The Negotiation of Multimedia Content Services in Heterogeneous Environment

Tayeb Lemlouma and Nabil Layaïda

Opera Project
INRIA Rhône-Alpes, France

{Tayeb.Lemlouma,Nabil.Layaida}@inrialpes.fr

Table of contents

- Introduction and Objectives
- Multimedia services negotiation
- The "*Profiling*" concept
  - *Client side*
  - *Document or Service side*
- The modularization principle and 'SMIL'
- Negotiation strategy
- Our approach
  - The '*TL Evaluation*' module
  - Protocol
- Conclusion

Introduction and Objectives

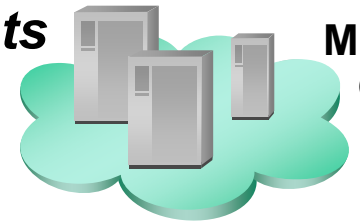
- With the explosive growth of computing technology, multimedia services (audio, video, animation, etc.) become necessary for most of the actual applications.

- By 2002, 75 % of web document viewing will be through non-desktop devices like palm computers, televisions, and other alternative platforms.

- The actual multimedia systems, include a wide range of clients which are subject of many constraints such as:
  - **» Low power**
  - **» Small user interface**
  - **» Small storage and processing capacities**
  - **» Limited access to the network**
  - **» Risks of data**

Problem

How can we allow different clients to access and to (re)use efficiently the network content?



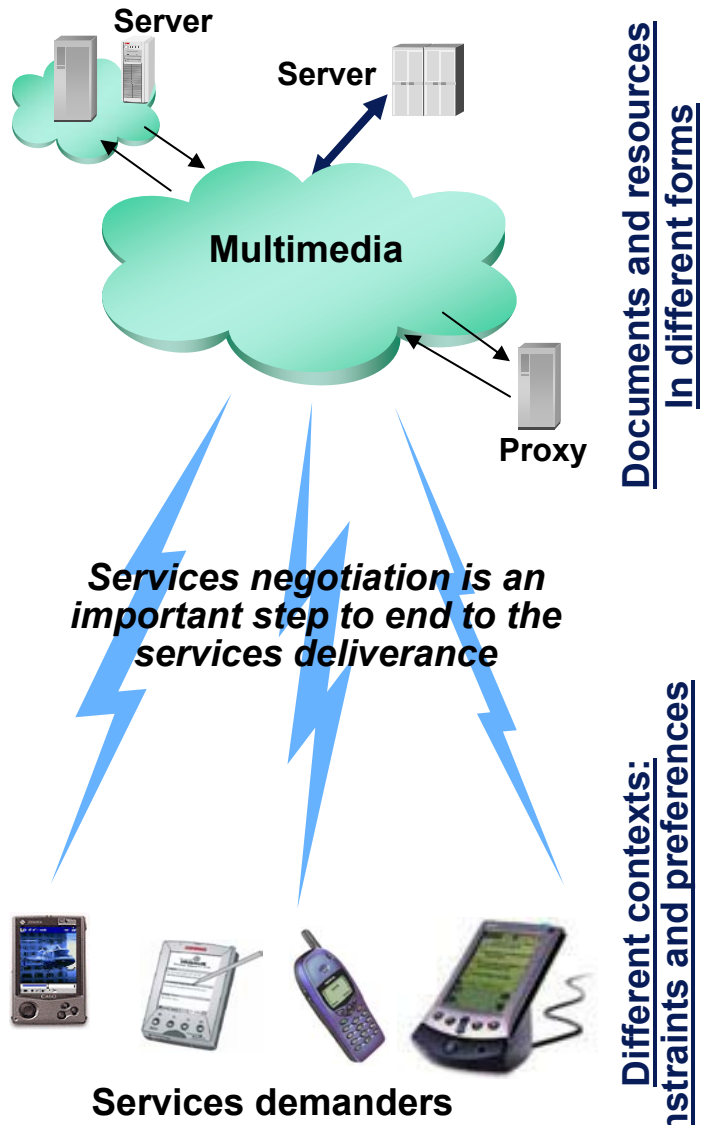
Multimedia content

Multimedia services deliverance

User1: Marjorie

User2: Manon

- ***Ensure a negotiation strategy is equivalent to:***
  - 1- Reply to all the diversity of users contexts that exists:
    - Consider different constraints posed by the user agents set.
  - 2- Determine the target format of the service in terms of selected modules.
  - 3- Determine the transformations to apply.
  - 4- Support context changes according to:
    - Client
    - Application
    - Environment (network, etc.)



The client profile

- The client profile describes resources and capabilities of the user.
- The CC/PP model represents an efficient tool to ensure such description.
- A CC/PP profile describes client capabilities and preferences in terms of a number of "CC/PP attributes" for each component.
- A CC/PP profile can include, for example, the three main components:
    - The software plate-form upon which all applications are hosted
    - The hardware plate-form upon which software is executing
    - The individual application used by the client, such as a browser or a player

An example of a Client Profile

```
[ClientProfile]
 |
 +--ccpp:component-->[TerminalHardware]
 |                   |
 |                   +--rdf:type--> [HardwarePlatform]
 .                   +--display----> "320x240"
 .
 [UADefault]
 |
 +--rdf:type---> [BrowserUA]
 +--name-------> ''WinWAP''
 +--version----> ''3.0.4.179''
 +--vendor-----> ''Slob-Trot Software Oy Ab''
 +--wmlVersionSupported--> [ ]
 |                         |
 |                         +--rdf:type--> [rdf:Bag]
 |                         +--rdf:_1----> ''1.2''
 +--ScriptsSupported----> [ ]
```

Document Profile or service

- 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)
- The central aspect of this approach is the definition of elementary functionality required for the rendering of multimedia services

Documents

Document profiles definition

Document 1	} *Profile 1*
Document 2	
Document 3	- *Profile 2*
Document 4	- *Profile 3*
.	.
.	.
.	.
Document N	- *profile N*

Profile

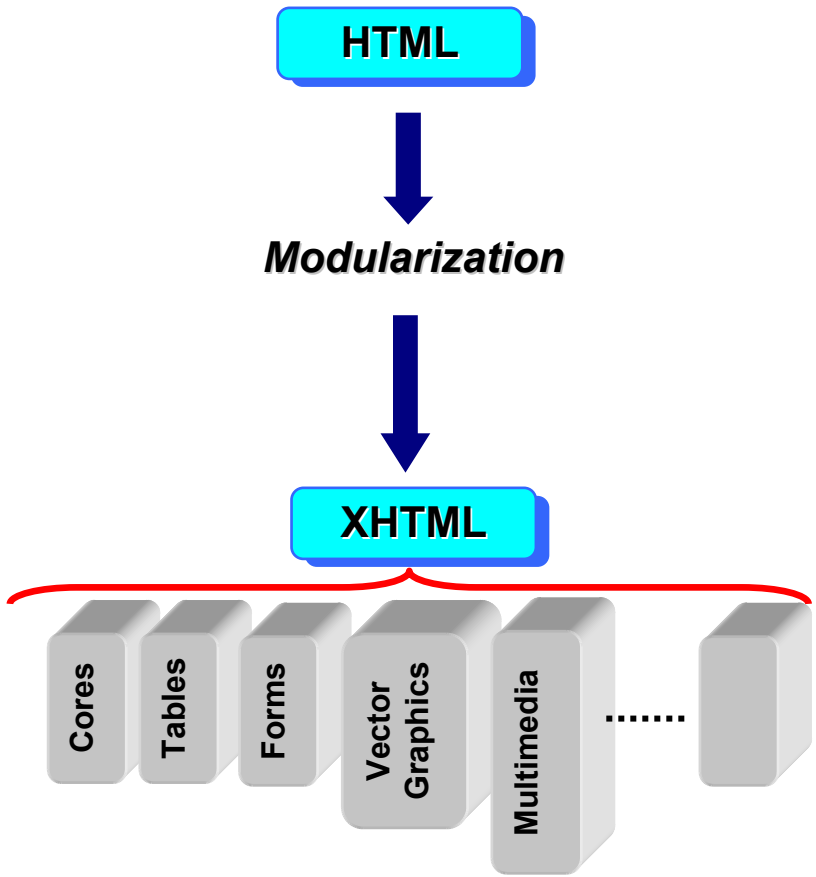
= { functionalities set }

The modularization principle

- **Objective:**
  resolve the heterogeneity problem and respond to adaptability needs

- A module represents a description of a set of functionalities. It can be seen as the basic element of a profile

- The modularization allows to support new devices and applications, bye defining subsets of modules and recombining them

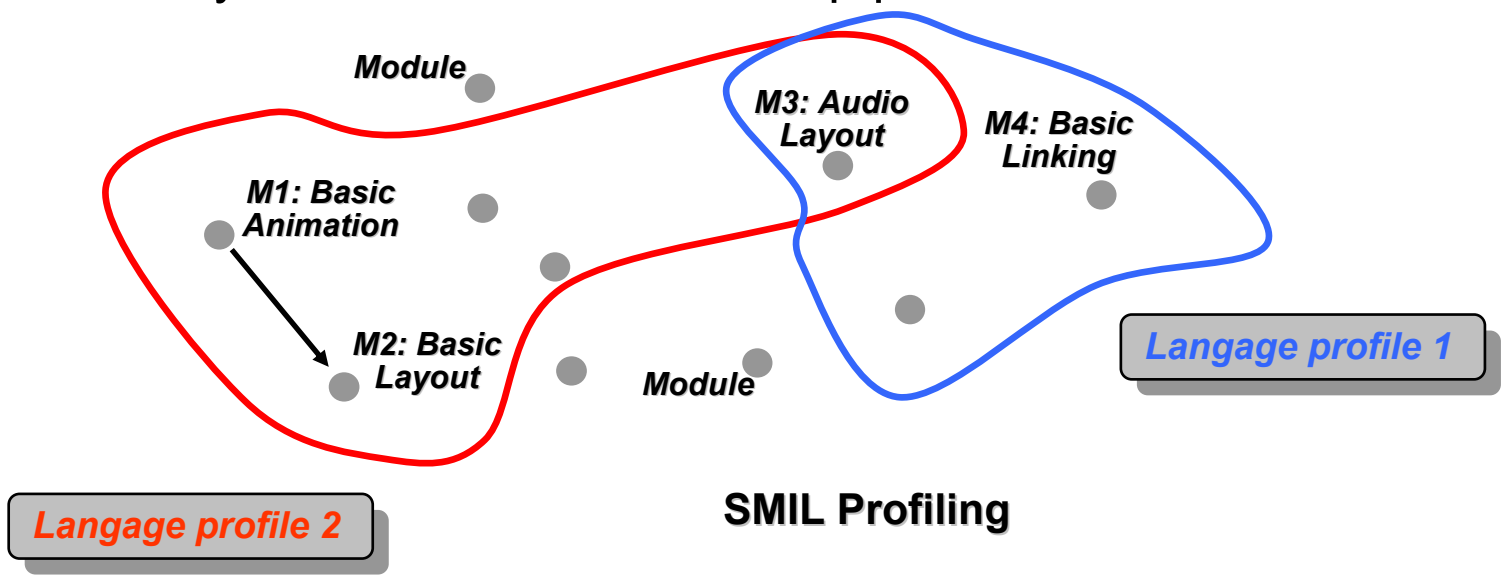


XHTML: An Example of modularization

SMIL and the Modularization

- **SMIL 2.0:** Covers a wide diversity of functionalities that can exist in a multimedia document.
- A language profile must include its basic modules and all the modules on which depends other included modules.

- **SMIL Basic:** Consists of a reduced subset of the full SMIL modules, which offer a common core more generic especially for contexts which impose many constraints like PDA, wap phones, etc.



SMIL Profiling

The negotiation strategy

- Must end to a consensus between the content server and the client that initiates the request.

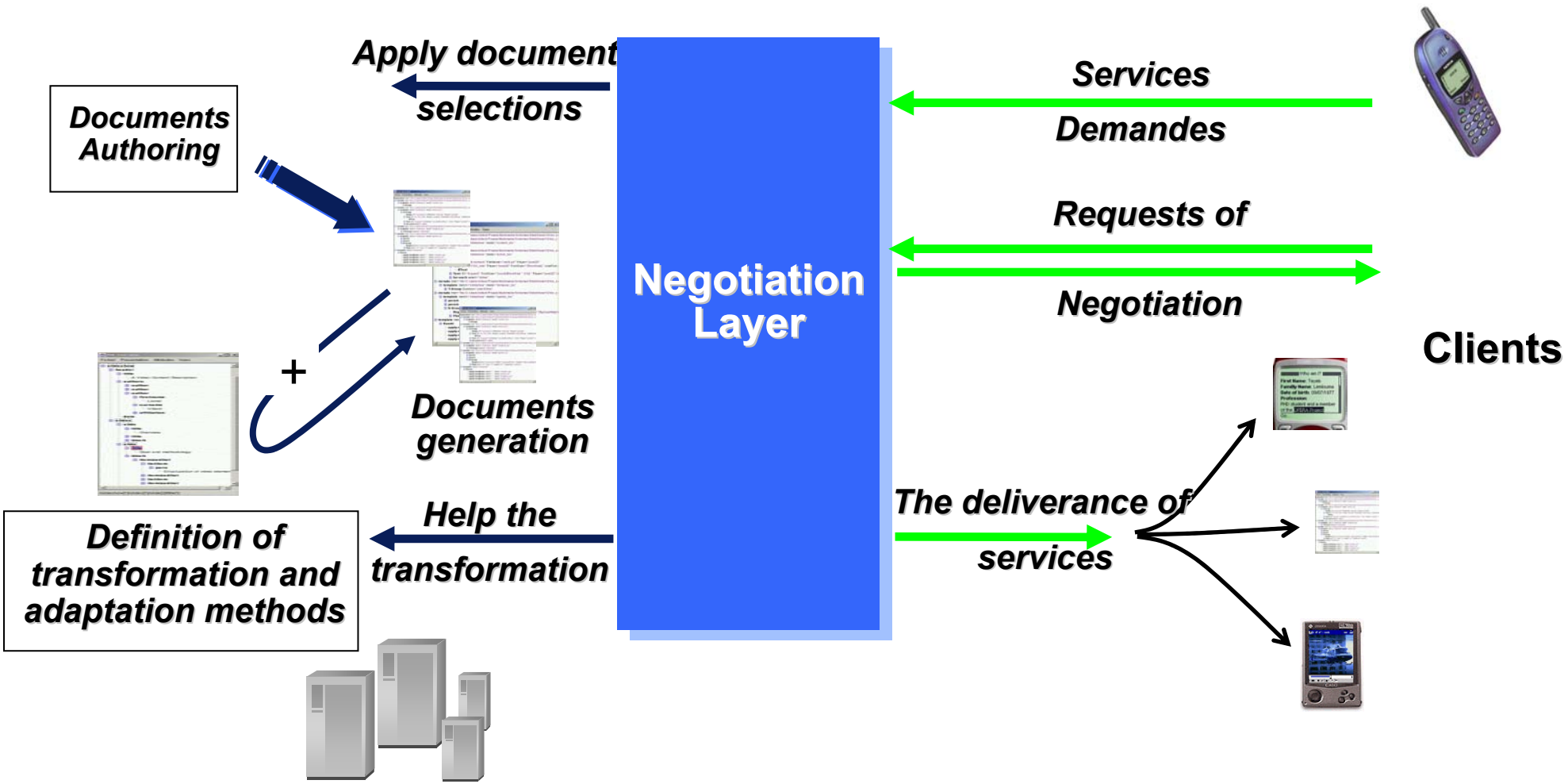
Basic initial steps

1- Creation of profiles on the servers of content, i.e. profiles of the content susceptible to be used by client. Tool: SMIL+CC/PP

2- Preparation of profiles on the client side, i.e. profiles of different user agents. Tool: SMIL+CC/PP

3- Enrich the server environment by adaptation methods. Tool: adaptation methods: XSLT + other transformation programs

The negotiation layer



Principle

The delivered content:

1- mustn't include an unsupported functionality according to the client profile

2- must cover the maximal supported functionalities

- Client side: The client supports the following atomic functionalities: **{X,Y,Z}**
- Server side: The server content is described by : **{Y},{X,Y},{X,Y,Z,T}**

* It's clear that the use of the content described by {X,Y,Z,T} is not permitted, because the T module isn't supported.

* The use of the content {Y} isn't preferable, because we have another content which covers more supported functionalities

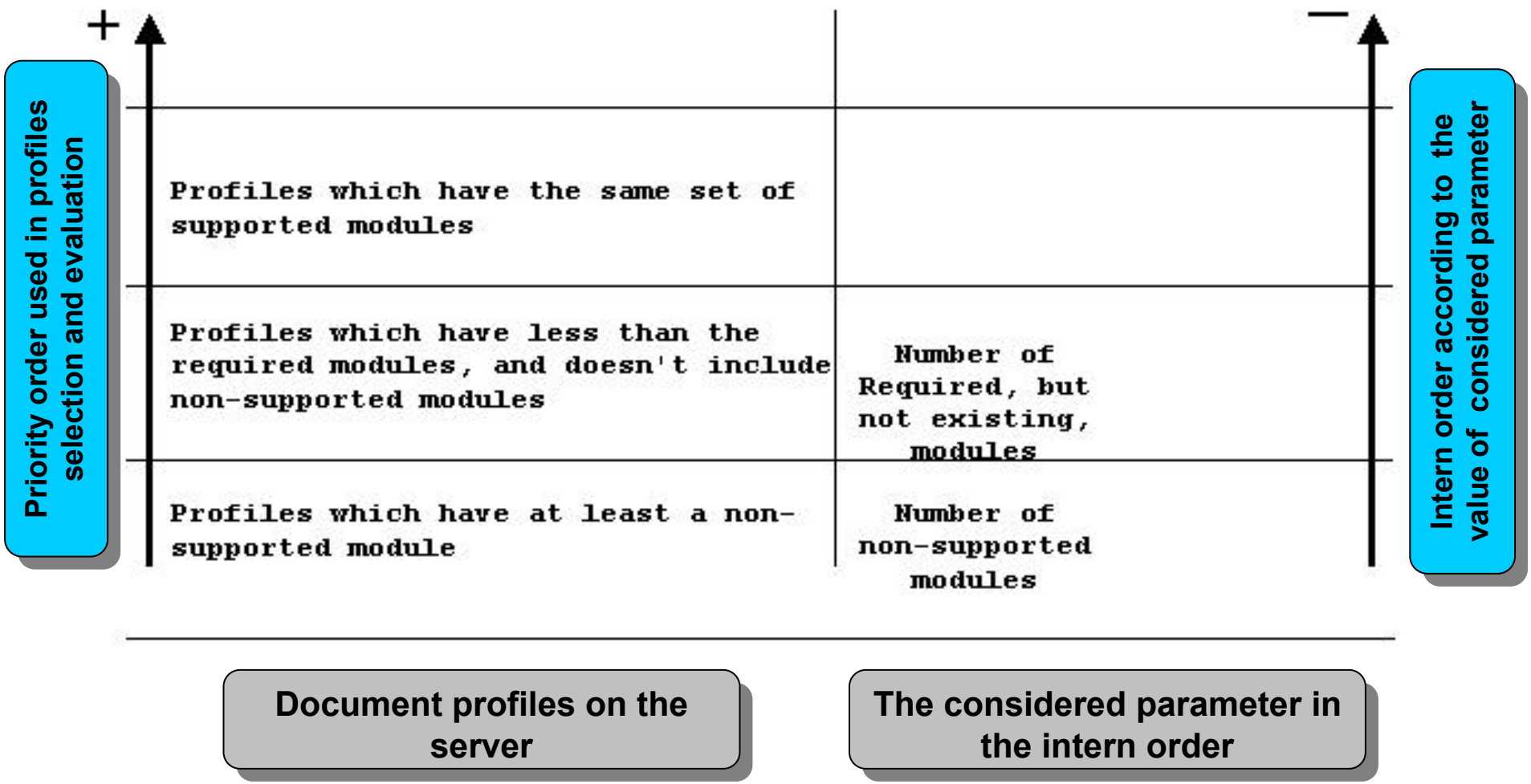
- The best negotiation strategy ends to the deliverance of the content having as profile **{X,Y}**

The '*TL Evaluation*'

- **We use in our approach, a profiles evaluation method that we call "TL Evaluation": Tailored Level evaluation.**

- **Allows selection and evaluation of profiles usable by a client and ordering them according to a priority level and constraints that can change**

- The evaluation used in our approach is base on the following priority order:



Requests formats

Client request

Client Identifier | *Server Identifier* | *Document Identifier* | ***User Agent Supported Functionalities*** | ***User Agent Preferred Functionalities***

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***

Overall view : Client

- Determine actual user agent supported functionalities;
- Determine actual user agent preferred functionalities;
- Determine document server;
- Send (*Client_Request*) to *Content Server* ;

---- After receiving server reply ----

- Select the best profile of the server proposed profiles:
  Selected Profile = **TL_evaluator** (server set, client constraint profile);
- Send (. . . Service identifier, Selected Profile, . . .) to server;

---- After receiving server reply :
(client identifier, server identifier, D: document) ----

- If D is not empty then use the service;

Server -1

--- When receiving a client request ---

Determine the supported and preferred modules (USP)

- Determine existing profiles that respond to supported functionalities:

  TL_evaluator(Supported functionalities, server profiles);

- Add profiles after applying existing transformations methods:

  TL_evaluator(Supported functionalities, profiles after adaptation);

- Apply further evaluation according to:
  - 1- client preferences: **TL_evaluator**(profile set, USP);
  - 2- server & network constraints:

    TL_evaluator(profile set, Server constraints);

- Send result profiles to the client

Server -2

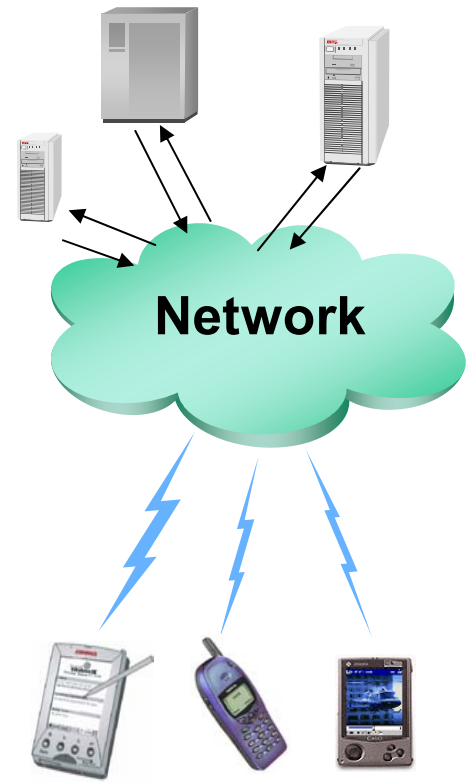
--- After receiving the client reply ---

- If the selected profile “SP”, is directly available:
  - Send the service having as profile SP
- Else: find the adaptation method “t” that corresponds to “SP”:
- Apply *t* to the service denoted by the document identifier
- Save the document profile (optional)
- Save the service version (optional)
- send the adapted document

send(Client Identifier, Server Identifier, Document) **to** Client;

Conclusion

- The problem posed by the Web, is that its content is made to be used in the classical model by desk top PCs
- An adaptation and negotiation architecture aims to allow small and limited devices such as: cell phones, personal device assistants, workstations etc. to access to multimedia services
- XML and related tools represent a good model to guarantee such architecture
- In a multimedia system, the negotiation permits to orient the adaptation process to offer the best service to clients
- The negotiation strategy that we have proposed:
  - Ensures an efficient opened multimedia system
  - Supports heterogeneity
  - Guarantees the extensibility (support new devices)
  - Supports environment changes (with changing declared profiles)



Future work

- The implementation of an XML protocol of services negotiation covering multiple of existing protocols (WAP, HTTP, etc.)
- Developing an XML-based proxy architecture
- Integrating a SMIL player (developed locally) of the overall architecture
- The enrichment of the server side by adapting methods: XSLT slide sheets (ex: SMIL 2.0 to SMIL Basic and XHTML), transformation programs (ex: HTML to WML, etc.)
- Developing an approach of automatic generation of XSLT style sheets according to client profiles

Thank you
Any questions?

More information on latest news:

www.inrialpes.fr/opera/people/Tayeb.Lemlouma/index.html

Contact:

Tayeb.Lemlouma@inrialpes.fr

Internet Explorer

Tools Help

Search Favorites Media

mation\HTML000.html

WAP Technology

WAP phones use Wireless Markup Language (WML) instead of HTML.

WML is very simple by comparison of HTML, and easy to be automatically created from monitoring scripts

Saviez-vous que les arbres parlent ?
Ils se parlent entre eux et v
L'ennui, c'est q
ils n'ont jamais appris à écouter les Indiens,
aussi je doute qu'ils écoutent les autres voix de la nature.
Pourtant, les arbres m'ont beaucoup appris :
tantôt sur le temps, tantôt sur les animaux,
tantôt sur le Grand Esprit.

Tatanga Mani ou Walking Buffalo,
Indien stoney (1871-1967)

Thank you for visiting this page

ex2.xsl | transfgen.xsl | XML2LaTeX.xsl | t1.smi | HTML2WML.xsl

```
<?xml version='1.0' encoding="iso-8859-1"?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Tr
<xsl:output doctype-public="-//WAPFORUM//DTD WML 1.3//EN

<xsl:template match="html">

<wml>
<card id="main" title="{head/title/text()}" newcontext="
            <xsl:apply-templates select="body/node()" /
        </card>
</wml>
</xsl:template>

<xsl:template match="h1">
<big><b><xsl:value-of select="."/></b></big><br/>
```