Interactive Digital Teletext Service

Chengyuan Peng and Petri Vuorimaa

Telecommunications Software and Multimedia Laboratory,
Department of Computer Science and Engineering,
Helsinki University of Technology,
P.O. Box 5400, FIN-02015 HUT, Finland.
Email: pcy@tml.hut.fi and petri.vuorimaa@hut.fi

ABSTRACT

Digital television offers significantly increased capacity, which can be used to provide digital Teletext service. The existing digital Teletext belongs to enhanced broadcast application profiles. It has only local interactivity and no return channel is needed. The digital Teletext with two-way interactivity is highly expected both from the growing market and the research area. This paper presents a Java-XML based solution, which can add two-way communications to the digital Teletext service by using a return channel provided by digital television infrastructure. The approach is based on the client-server model and utilizes XML based messaging mechanism and Java Servlet technology. A simple TV shopping system was developed inside the digital Teletext browser as an example of interactivity and shows how messaging works. The order forms, which are frequently used in an interactive system, are represented in XML format. The form system separates the presentation and the form content. Therefore, the average latency of the system is decreased.

Keywords: digital Teletext, interactive service, client-server model, JAXM messaging, forms, Java, XML.

1. INTRODUCTION

The digital Teletext is a resident service of digital television in set-top box. It is an enhanced version of analogue television Teletext service (see Figure 1).

Figure 1. Analogue Teletext service.

With the digital Teletext, the pages are stored in data carousel and transmitted via broadcast network. The software and data can be downloaded to set-top box from broadcast object carousel and the pages can also be requested dynamically from data carousel instead of using vertical blanking intervals in analogue TV. It has rich graphics and multimedia user experience. The menu-driven navigation method is used instead of pressing three-digit number on a remote control. The content of digital Teletext can be represented in Extensible Markup Language (XML) [1]. However, only local interactivity, e.g., navigating service, downloading pages, activating another service (e.g., Xlet), etc., can take place in the existing digital Teletext services with a remote control.

The two-way interaction in digital Teletext means that the viewer is able to send request to the service provider and to get an acknowledgement from the service provider. A return channel and its network communication protocols are needed for developing an interactive digital Teletext service to achieve the two-way communication between the parties.

Figure 2 shows a general model for an interactive system. The broadcast channel is a unidirectional broadband broadcast channel, which carries broadcast content including video, audio and data using the MPEG-2 transport stream from the broadcaster to the viewers [2]. Data transmission may be simplex or full duplex and may be unicast, multicast, or broadcast.

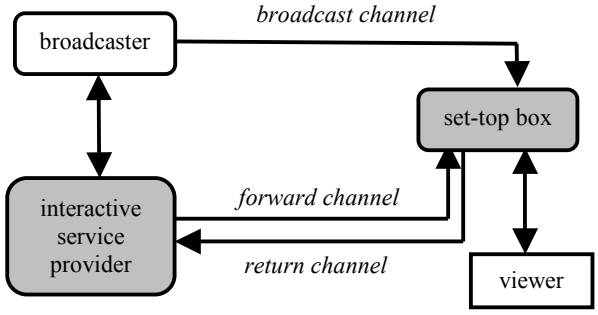

Figure 2. A general model for interactive system.

The interaction channel is a bidirectional interaction channel between the viewer and the service provider. It includes the return channel from the viewer to the service

ISBN: 980-07-8150-1 © SCI 2002

provider and the forward channel from the service provider to the viewer (cf. Figure 2).

The return channel is used to make requests to the service provider or to answer questions. The return channel carries the user's commands and responses. The forward channel is used to provide information from the service provider to the viewer and any other required communication for the interactive service provision. This channel may not be required in some simple implementations, which can make use of the broadcast channel alone for the carriage of data to the viewer.

The Digital Video Broadcasting (DVB) has defined various types of return channel systems. The return channel is usually implemented via a narrowband network, such as a Public Switched Telephone Network (PSTN)/Integrated Services Digital Network (ISDN). Depending on the level of interaction, the bandwidth requirement of the return channel varies. Implementing a return channel requires modifications in both the broadcaster site and the receiver.

In addition to a return channel, the communication protocols are also needed for interactive digital Teletext service. The general protocol stack model includes three layers (i.e., application, transport, and physical layers [2]). The application layer is the interactive application software and runtime environment. This layer is open to competitive market (cf. [3]). All the physical (electrical) transmission parameters are defined in the physical layer. The transport layer defines all the relevant data structures and communications protocols, etc. Figure 3 shows an interaction channel (forward and return) protocol stack [4], which is used by the digital Teletext service.

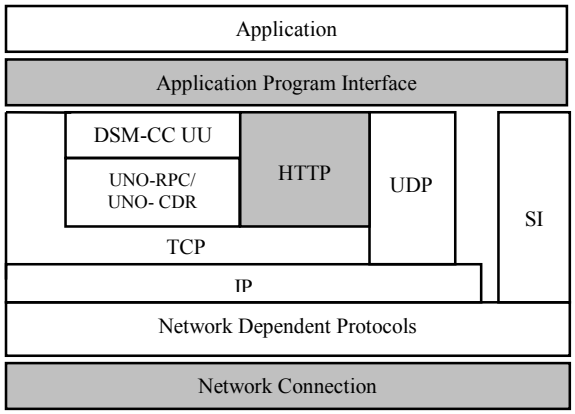

Figure 3. Communication protocols stack in the DVB-MHP.

There are several options above the network dependent protocols layer in this stack model. The DSM-CC UU indicates the Digital Storage Media – Command and Control User To User. The UNO-RPC means Universal Networked Object-Remote Procedure Call. The UNO-CDR means Universal Networked Object-Common Data Representation. The SI indicates service specific protocols, which are proprietary protocols and used by a service.

One of the important measurements of implementation of an interactive digital Teletext service is a fast response. Viewers are accustomed to instantaneous response. A short response time proves highly desirable in a television environment. Response time in set-top box mainly includes data access and content presentation delay. The presentation latency is important, as digital Teletext service will handle a large amount of pages, images, animation, etc [5]. Data access latency implies that the request and response time from the network [6].

Our system was built on the digital Teletext browser developed in Future TV project and the work in this paper is the follow-up research, which is based on [1] [5]. The digital Teletext browser adopts a non-XML web browser based approach. It is a lightweight and Java-based application, which can browse XML pages. The Java bytecode size was only about 81KB [3].

2. THE SYSTEM MODEL

The system with two-way interactivity is modeled as the traditional distributed client-server model and uses message-oriented mechanism to send requests and receive responses. With this model, set-top boxes can interact with the servers of service provider. Under this model, the processing tasks involved in running an interactive digital Teletext browser are divided between a client set-top box and a server. The set-top box is responsible for data accessing from the network, XML data parsing, and content rendering. The server performs message processing and database retrieving. Figure 4 shows the system architecture of the digital Teletext browser.

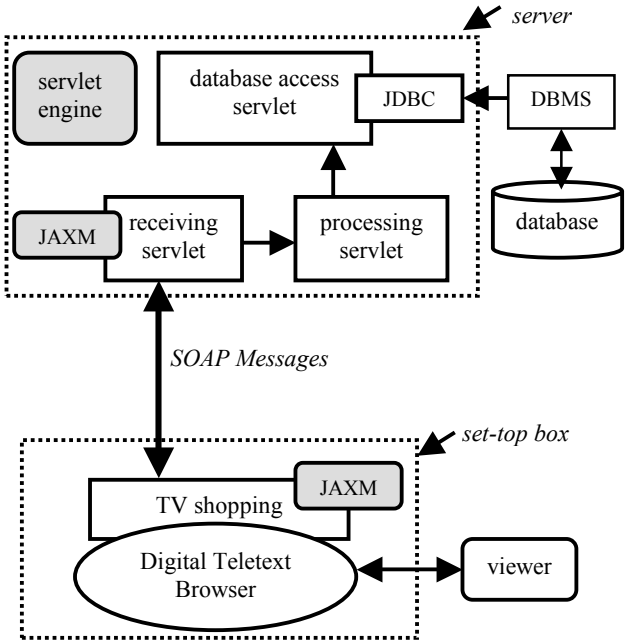

Figure 4. System architecture.

2.1 XML Messaging

Message-oriented approach is much suitable for return channel communications. The client-server model is based on the pure Java API for XML Messaging (JAXM) framework. The JAXM provides a standard way to send messages over the Internet from the Java platform [7]. It is based on the Simple Object Access Protocol (SOAP) 1.1 and SOAP with attachments specifications. The request and response messages are modeled as SOAP messages (cf. Figure 5). The messages are always document-centric since the page of digital Teletext is an XML file format [1]. Also, the XML document self can contain some kind of information of operations.

One advantage of using messaging is that the client and server can belong to different companies. It's possible that the companies might not use the same programming language, or may use an asynchronous communication model so that neither will depend on the other's application being up and running [9]. Another advantage of messaging occurs that it can remove the dependency between an event and the system's reaction to the event performed on the server (like HTML), e.g., processing ordering forms.

JAXM is designed to be a lightweight API that abstracts away the underlying messaging infrastructure. The JAXM supports for reliable message delivery; a variety of message types and payloads (SOAP with attachments); and security, including message integrity, authentication and non-repudiation of messages and audit trails [7]. All these features are necessary for set-top applications.

The addition of XML to messaging means that we are able to make use of a flexible data formatting language for our messages. In messaging, both the client and the server need to agree on a message format. XML makes this easier by deciding many data formatting issues. No additional work is required to come up with a message format [9].

2.2 The Client (Set-top Box)

The client was designed as a standalone mode, i.e., synchronous (point-to-point) mode using a local provider. The sender cannot proceed until a response to the request being sent is received. A local provider is a pure library implementation of the JAXM API so that SOAP messages can be sent directly to the remote server [7]. The client side runtime libraries (cf. Figure 4) and the message-processing code were needed in the system architecture.

The client creates a message of a specific format known to both the client (i.e., SOAP Message) and the server and then sends it to the server over the return channel. The client does not depend on the server or the server's procedures. It is only dependent on the message format.

The messages (i.e., SOAPMessage) sent over the return channel include request and response messages. The response message is simple and only consists of SOAP part. No attachment part is needed. The request message is made up of two parts, a SOAP part and an attachment part (cf. Figure 5). The SOAP part consists of a SOAPEnvelope object containing a SOAPHeader object and a SOAPBody object. The XML page file (e.g., containing shopping cart data) is encapsulated in the SOAPBody. The attachment part of a message contains a plain text file (e.g., encrypted and extracted user order forms).

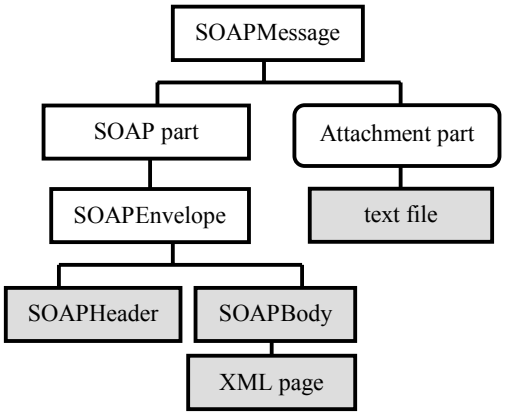

Figure 5. SOAP message protocol used in the system.

One of the reasons of wrapping the content in SOAP is that developers can focus on building, sending, receiving, and decomposing messages for the applications instead of programming low-level XML communications routines. Another reason is that the SOAPMessage can be routed to more than one destination, which can be specified in a message's SOAPHeader [7].

2.3 The HTTP Server

The HTTP server was used (cf. Figure 4) in the system. It contained an add-on Java Servlet engine using Java Servlet API, which is a standard Java extension. The server consisted of three servlets, a DBMS system, and a database.

The receiving servlet derived from JAXMServet is responsible for handling the messages. Once a message is received, this servlet will interact with the servlet engine through requests and responses. It then sends the appropriate response back via the same route to the client. Meanwhile, it activates the processing servlet to unpack the SOAPMessage and pass documents (e.g., the order and shopping cart data) to database access servlet. The database access servlet uses the JDBC (i.e., database connectivity software) to contact the relational database.

As a message receiver, the receiving servlet can handle multiple requests concurrently, and can synchronize requests. This servlet is the JAXM client application and was deployed in the Servlet container. According to the

JAXM specification, the JAXM applications in server side are also called JAXM clients [7].

3. FORMS DESIGN

Interacting with forms is an important part of implementation of interactive digital Teletext browser. Forms are mainly used here to pass user input to the remote server. Forms processing includes form content definition, data processing, and user interface rendering.

The forms used in the digital Teletext browser are not XForms specified in W3C. The XForms represents the next generation of Forms for the Web browser. The digital Teletext browser is a standalone application as mentioned in the introduction part.

3.1 Form Format Definition

Figure 6. An order form in Digital Teletext browser.

The forms definition is described in XML format. They include form components, e.g., text field, text area, combo box, check box, radio button, button, etc. Each form component is defined as an XML element and contains the attributes, which are used for carrying data value and rendering information. The following piece of code shows a form definition in Document Type Definition (DTD) format (cf. Figure 6).

```
<!ELEMENT Form ( (Component+ ) ? )>
<!ELEMENT Component ( ( radio+ )?, (checkbox+)?)>

<!ATTLIST Component
        Width     CDATA   #REQUIRED
        Height    CDATA   #REQUIRED
        X         CDATA   #REQUIRED
        Y         CDATA   #REQUIRED
        Type      CDATA   #REQUIRED
        Value     CDATA   #REQUIRED
        Caption   CDATA   #REQUIRED>

<!ATTLIST radio
        Value     CDATA   #REQUIRED
        Caption   CDATA   #REQUIRED>
<!ATTLIST checkbox
        Value     CDATA   #REQUIRED
        Caption   CDATA   #REQUIRED>
```

Data processing of the form components includes data gathering (i.e., read and write the values of input elements in the forms into an XML document), data validating, and key event processing. Validating is a pre-process, which is used to verify the form prior to submission. This allows the server-side scripts to be simpler and reduces the amount of data that must be sent between the client and the server. The valid form data is then sent in a SOAP message with some operations (or behaviors) to the remote server.

3.2 Event Model

Navigation of forms is difficult using a remote control. In addition to considering the usability issues, user interface must complete complicated navigation functions. Some UI components may have text input function; some components have their own navigation functions (e.g., Radio buttons, Combo box, check box, etc.).

Consequently, one delegation key event model was designed to ease navigation (see Figure 7). Key event processing was designed so that each form component can listen to other user interface components as well as its own component (i.e., input data). And the viewer may only press "up" and "down" buttons to browse each components, "+" and "-" buttons to add or remove items from the shopping cart, etc. (cf. Figure 6). Also, the number of form components on the screen is reduced and long forms that ask viewers to enter large amounts of text were avoided.

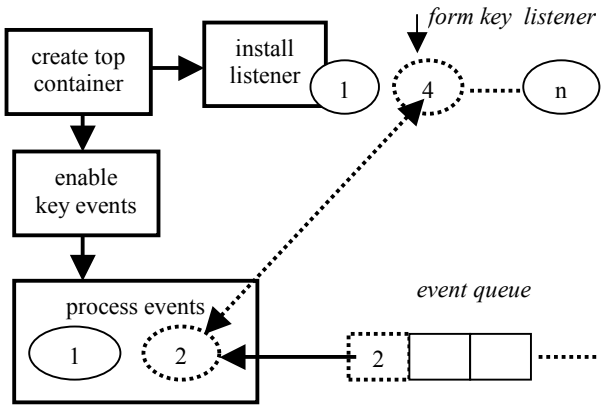

Figure 7. Remote control navigation model.

In Figure 7, the top container contains form components. After receiving the creating of forms request, the top container installs the form key listener, and then enable key events. Each event (e.g., button press, checkbox press, etc.) in the event queue is customized Java AWT event and has a unique event ID. Each customized event is associated with the customized form key listener, which processes corresponding events. The customized form key listener is responsible for receiving keyboard events and registered with top container. The key events are multicast to the form components. Each key listener can be added or removed from the top container by its

parent container. This approach is efficient and can solve most of navigation problems with fast response.

3.3 Form Presentation

The form components, e.g., text field, button, combo box, etc., were designed as drawing objects instead of separate components, which are usually derived from Java AWT/Swing user interface components and added to the top container. The drawing objects can be rendered in the form top container at runtime. The container does the rendering tasks (cf. Figure 6). Also, there is no need to place the form components in front of other containers [1]. The XML form content script can change during the content updating. This design approach has proved to be very efficient on reducing both rendering and components initialization time. The memory consumption can also be significantly decreased.

4. IMPLEMENTATION

On-line TV shopping is a good example of showing interactive service with two-way interactivity. With the digital Teletext browser, the viewer can use a remote control to browse, select, and securely purchase goods from various electronic storefronts through their television set. Figure 8 is a screenshot of the TV shopping system. In the system, the Internet was used as the return channel at the test phase and the HTTP protocol (cf. Figure 3) was selected as the return channel communication protocol.

Figure 8. The TV shopping system.

The design and implementation of a TV shopping system demonstrates a framework of digital Teletext browser with two-way interactivity. The design was focused on the client part since the complex database payment system was not implemented in the first stage of the project.

The client configuration in the prototyping system included a PC with Windows 98 as the operating system. The application layer protocols consist of (from top to bottom) applications, middleware, and software and hardware resources layers [3]. The middleware layer uses JAXM 1.0 reference implementation early access 2, Java API for XML Processing (JAXP) 1.1, Java Media Framework 2.1.1, and JDK 1.3.3_01.

The server configuration consisted of an Apache 1.3.22 HTTP server, Tomcat 4.0 servlet engine, JAXM 1.0 reference implementation early access 2, JDK 1.3.1_01, (with JDBC), and a database. Tomcat 4.0 is servlet container, which was installed as an add-on component to the server.

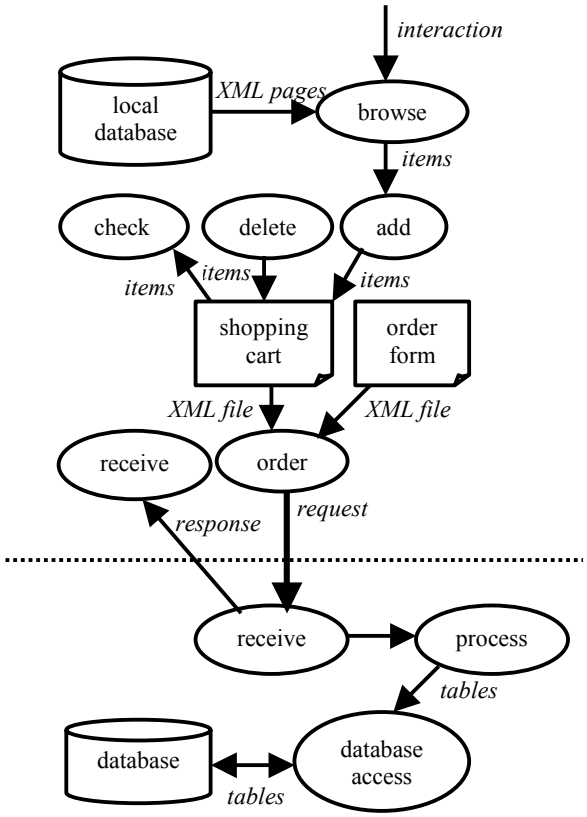

Figure 9. Data flow diagram of the XML-based messaging system.

Figure 9 shows a data flow diagram of the system messaging procedure. The viewer can *browse* through the products available in XML pages transmitted via broadcasting network (here via local database). When the viewer decides to buy a product after *adding* it in a shopping cart, he/she can *order* the product and fill in an order form. The shopping cart and the order form are files in XML format, which are defined by the service provider and transmitted via broadcasting network. The data both in the order form and the shopping cart is extracted and encapsulated in the SOAPBody and attachment part, respectively. They are then sent to the server in SOAPMessage format.

After the server has received the SOAPMessage, it sends back a response in SOAPMessage without an attachment part. It tells whether the credit card transaction has been accepted, etc. Meanwhile, the server will continue to process the SOAPMessage arrived. The database

connectivity software will pass the gathered data to the database.

5. RESULTS AND PERFORMANCE

The presentation and response time are very important for measuring the performance of an interactive service as mentioned in the introduction. The presentation delay (including initialization and rendering time) for displaying forms in set-top box client is about 2.6 seconds (order forms, cf. Figure 6). After initialization, the presentation time is instantaneous. Since each set-top box is independent of each other, we only need to get the delay from one set-top box.

The delay time for the message to arrive to the server from the set-top box was about 1.2 seconds. The response time (i.e., the message was sent back to the set-top box) was 3.5 seconds. This includes the database accessing time. This result was got, when there was only one client. We also gathered the response time, when twenty clients were sending the messages to the same server simultaneously. The response time was almost the same. These results show that the server's configurations are reasonable and independent of the number of clients. The time was consumed mainly by database access. This can be improved in the real implementation.

The performances show that the design and implementation of the prototyping system works well and can be deployed in the real broadcasting environment in the future state. Our work would be a valuable reference for companies and practitioners.

6. CONCLUSIONS

This paper presented a messaging framework for establishing two-way communication in digital Teletext service by using the Internet as a return channel. The components configured in the client-server model were proved to be the best choice among other alternatives. One of the key ingredients enabling standards-based interoperability is the widespread adoption of a transport-neutral packaging model and agreements on message header structures, manifests, and so on. This messaging frame can also be used in other applications that require asynchronous one-way (from set-top box to service provider or other receivers) interactivity.

The forms designed in the system can be used by other applications. They provide a flexible interaction tools. The XML language, which was used to define the forms as well as other digital Teletext pages, demonstrates the advantages of common platform independent and easy authoring. When a new XML form file is downloaded, user interface rendering is done automatically. There is no need to re-download digital Teletext application code. The SOAP messages must be transmitted using HTTP protocol, thus the HTTP server and Java Servlet were selected in the system. The Servlets provide a stable and secure mechanism for server-side code execution. These are important features for e-commerce applications. Servlets are protocol- and platform-independent server side components, written in Java, which dynamically (on demand) extend Java enabled servers [8]. They provide a general framework for services built using the request-response paradigm. Servlets are executed as a separate thread within the servlet engine, whereas CGI scripts require a whole new process to be started.

The XML pages were in local storage instead of Digital Storage Media - Command and Control (DSM-CC) data carousel. This can be improved in the next stage of service development. Also, the narrowband return channel will be used instead of the Internet.

7. REFERENCES

[1] C. Peng and P. Vuorimaa, "Digital teletext service," *proceedings of the international conference on Computer Graphics, Visualization and Computer Vision, WSCG'2001*, Czech Republic, February 5 - 9, 2001, pp. 120-125.

[2] ETS 300 802, Digital Video Broadcasting (DVB); Network-independent protocols for DVB interactive services, *European Telecommunications Standards Institute*, November 1997.

[3] C. Peng, P. Cesar, and P. Vuorimaa, "Integration of Applications into Digital Television Environment," *the International Workshop on Multimedia Technologies, Architectures, and Applications*, September 26-28, 2001, Taipei, Taiwan, pp. 266-272.

[4] TS 101 812, Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0.1, *European Telecommunications Standards Institute*, October 2001.

[5] P. Vuorimaa and C. Sancho, "XML based text TV," *Web Information Systems Engineering*, *WISE2000*, Hong Kong, June 19-20, 2000.

[6] Milan Milenkovic, Delivering Interactive Services via a Digital TV Infrastructure, *IEEE Multimedia*, Volume 54, Oct. - Dec., 1998, pp. 34-43.

[7] N. Kassem, A. Vijendran, and R. Mordani, JAXM specifications. Java API for XML Messaging (JAXM) v0.93. *Public Review Draft 2*, August 2001.

[8] D. Coward, Java Servlet Specification version 2.3, Final Release, August 13, 2001.

[9] Dirk Reinshagen, "XML messaging, Part 1, Write a simple XML message broker for custom XML messages," *Javaworld*, March 2001. http://www.javaworld.com/jw-03-2001/jw-0302-xmlmessaging.html.