Contributions to the development of distributed multimedia applications

Introduction

This thesis consists of strategies and methods for improving a part of characteristics of the content management systems. The approach is written based on the CSMDS project developed by the authors during the PhD stuff. Motivation for this work comes from two directions. Firstly, for building multimedia distributed systems close to the real world – a 2D character talking to the end-user in an eGovernment virtual service provider, for instance, seems to be an assistant interpreting the low or the same 2D character selling products within an eCommerce site seems to be the real vendor in a real store. The second motivation is that the Internet and the CTI systems are converging to the same point using the same speech-based user interaction.

The classic distributed systems have provided a normal HTML graphical user interface for simplifying the user interaction and avoiding the network problems. In this way, the end-user can interact with the system filling the input text-based graphic components and pressing the buttons and hyperlinks. The Internet is growing up, involving a lot of network technologies of highest speed and invention. More, the biggest technology providers are ready to put on the market a lot of unbelievable strategies for developing distributed systems that allow the system architects to design personalized and interactive 2/3D graphical user interfaces connected to the optimized application servers.

The CSMDS framework gives a new face to the distributed systems, improving the user interaction and changing the main features within the look and data delivery modules. These features can be the basis of a new distributed system or to be integrated in an existing one because of the integration capabilities provided by the CSMDS API. Any developer, extending the framework components, is able to implement, without too much effort, an interactive 2D user interface meeting all the customers’ requirements and providing them all the requested information, via audio RTP streams. The framework has been designed for providing web-based services but it can be extended to offer the service access for mobile devices such as PDAs or smart phones.

The CSMDS framework consists of three main components:

- **2DCharacter** component implements the look in the user interaction. It will get the human image and comes from the idea to have a graphical user interface very close to the real world. It also contains the synchronization module that processes the notifications about the data delivery. In this way, the 2D character seems to speak to the end-user in a real manner, making all the movements synchronized with the audio streaming.

- **voiceComponent** package implements text-to speech/speech recognition capabilities and data delivery. The voice client component captures the audio data from the microphone, encodes and transmits it to the voice server that receives the audio streaming and forwards the audio data to the speech engine. The speech recognition components generate a simple command that will be processed by the business delegate component. This command represents an interface between the controller and model. The model components process the information and forward the response to the text-to-speech engine. The speech engine is managed by an AudioManager component that avoids the sound blaster, as default, in order to realize the data streaming from the voice server to voice client, via RTP.

- **commClient** package implements the communication features of an integrated communication environment: chat, video conference, whiteboard, or file management capabilities.

CSMDS Architecture
The CSMDS framework respects the multi-tier architecture of the distributed systems and uses value object, business delegate or session façade J2EE design patterns. It can be integrated to communicate with EJB or Java Servlets components and allows the possibility to build a reliable and easy-to-use distributed system providing high degree of scalability and speech-based user interaction.

The multi-tier application consists of:
- **Data tier** contains a set of databases hosted by a MySQL server. These databases store all the information related to the remote services, users, etc.
- **EJB tier** manages a set of session bean components that implements the business logic and entity bean components for retrieving the information stored into the databases.
- **VoiceServer** block receives the audio RTP requests from the VoiceClient, transforms the audio data into an internal command (the interface between controller and model components) and generates the speech based on the information from the model components.
- **Web server** receives the HTTP requests the 2D Flash scene required, calls the remote service and sends the response, via HTTP.

Two J2EE design patterns are used in the framework development – value object and session façade. They come to be used for reducing the number of service invocations on the server-side. More, the value object design pattern is used for interchanging the data between the Flash application and Java service – the Flash application does a service request and the Java-based application server answers providing the response. For instance, only one method will be called on a value object, where few methods implemented by an entity bean have been invoked. The result, provided by the method will be used in the ActionScript code, reducing the traffic and the overloading of an EJB component. Session façade represents the only one contact point with the server-side components. It reduces the network traffic and allows different client types to access remote services. Thus, the remote services will be accessed via business delegate component.

**CSMDS Components**

**2D graphical user interface**

The view part of a system consists of a set of graphic elements for displaying text or graphics, related to the users’ events. The framework provides one 2DCharacter component that has to implement the look of the system. In the design process of this framework the architect has been interested by the following features:
- Very friendly graphical user interface makes the application close to the real world
- The presentation logic (implemented by the client-side components) is separated by the business logic the server-side components implement.
- Low-cost deployment will be ensured
- The client-side performs the routines, transmits the data and invokes the remote services

The block diagram of the 2DCharacter component will be illustrated bellow:

![Fig. 2 2DCharacter - block diagram](image)

2D graphical user interface comes from the on-line entertainment applications with the scope to make the system more attractive. Another advantage considered is about the implementation of this concept – it can be easily implemented using Java or Macromedia Flash MX. Both of them are platform-independent and allow the developer to build the graphical components connected to the server-side resources in a very simple manner, via HTTP.

**VoiceClient block**

*VoiceClient* implements the speech-based user interaction capabilities: it captures the audio data, encodes it into RTP streams, transmits these streams to the *VoiceServer*, receives the RTP streams from the *VoiceServer*, decodes and renders the audio data to the end-user. *VoiceClient* is designed making part of thin client category and all the data processing will be implemented by the server-side components. The block diagram is illustrated bellow:

![Fig. 3 – VoiceClient - block diagram](image)

**Capture/transmission functionalities**

One Java Applet covers the capture and transmission functionalities. For capturing the audio data from the microphone, encoding it into RTP streams and transmitting these streams to the "VoiceServer" component, the default init() method of the Applet has to implement the following steps: create the capture device, define the data source, define the type of the audio format, get the Processor object used for realizing the RTP streaming.

**Receiving/Rendering Capabilities**

For receiving, decoding and rendering audio data from a RTP streaming the Java Applet has to use the capabilities provided by a JMF Player object. In this way it has to implement the following steps within the initializing session: define the type of the audio format, get the Processor object for the RTP streaming, get the Player object for the RTP streaming.

**VoiceServer block**

*VoiceServer* block implements the speech recognition/synthesis capabilities. It also has to receive the RTP data streaming from the *VoiceClient* block, decode and forward this data to the *Recognizer* object. The information from the model components will be sent to the *Synthesizer* instance and a new RTP audio stream will be generated.
By default, the Recognizer and Synthesizer objects are designed to work with the sound blaster. Other component, called AudioManager, manages these objects to avoid the sound blaster on the server machine and to communicate via RTP according to the following steps:

- get the AudioManager using the Recognizer object already created
- create the Processor object
- get the link between AudioManager component and the RTP data source using the Processor object
- Recognizer object is ready for receiving audio data

The application grammar is automatically created using the RuleGrammar object for defining the grammar and the Rule one for defining the predefined commands. These commands can be keywords for a remote service and can be stored into a database table and automatically put into an EntityBean EJB component when the remote service is initialized.

After the application grammar is defined a set of steps must be implemented:
- Recognizer object is created
- Recognizer object on the ALLOCATED status
- And AudioManager object will be created, in the context of Recognizer instance
- If the Microsoft API 5.1 is used, the ResultAudioProvided property has to be TRUE and the Recognizer returns FinalResult objects.
- DataSink object is created in the context of AudioManager instance
- The Recognizer on the LISTENING status waiting for the RTP streaming

If the VoiceServer component extends the ResultAdapter class, it has to override its resultAccepted (ResultEvent ev) method.

When a command is recognized, the remote service is invoked using the recognized command as parameter and the result provided by the invocation is forwarded to the Synthesizer object for generating the RTP audio streaming.

The distributed synthesis is done in the same manner and the authors avoid repeating the information.

**Communication components**

The communication environment provides a set of features very important in the collaborative work: on/off-line messaging, audio/video P2P link, whiteboard capabilities, file management features, users’ management, message management.

All the object-oriented languages provide APIs already tested for developing communication environments but there are a set of network constraints that create problems when using the communication service, for instance, proxy/firewall protections or the bandwidth, especially for video conference tools, it never can be considered enough.
Macromedia Flash Communication Server provides a set of capabilities for managing any network trouble – Macromedia Real-Time Messaging Protocol avoids the firewall protections becoming RTMPT (tunneled) and running over the port 80 as the HTTP request/response mechanism. Macromedia also introduced the shared objects concept that allows the developer to manage the client-side in a new manner. The rich client concept is another very important property that allows the developer to build rich Flash scenes containing audio-video capabilities and graphics features.

Conclusions

Current personal computers, workstations and servers are designed to handle traditional forms of data. Their performance is optimized for a scientific or transaction-oriented type of workload. These systems do not perform well for multimedia data, requiring fast data retrieval and guaranteed real-time capabilities. The I/O capacity is usually a severe bottleneck. Our system provides the following set of characteristics:

- **Multi-spectator virtual environment capabilities** are implemented. For instance, several students and professors/tutors can experience the same environment (virtual classroom, university) but at a moment just one can actually interact with it.
- **Multi-user virtual environment capabilities** are provided. Within an eCommerce platform, several virtual customers/administrators can concurrently be presented in and interact with the virtual environment.
- **Minimal response time**: A crucial factor for the success of multimedia services is the response time seen by the client. The server must be able to minimize response times to live up to the expectations of the user.
- **Fast processing capability and low data access rates**: To guarantee fast response times, client requests should be processed fast and data access rates should be minimized.
- **Reliability and availability**: Like any other kind of server, a multimedia server must be reliable. The number of users and volume of data is handled by the server; the more difficult it is to guarantee reliability. To provide fault tolerance special software mechanisms must be employed. Since client requests may arrive at any time, the time the server is unavailable should be minimized.
- **Ability to sustain guaranteed number of streams**: Another important factor is the maximum number of data streams the server can simultaneously handle. This affects the total number of clients the server can serve.
- **Real-time delivery**: To be able to deliver multimedia data, the server should support real-time delivery. This poses profound requirements on the resource scheduling at the operating system level. The server should be able to guarantee real-time delivery for individual streams as well as for all the streams combined together. For this accurate real-time operating systems have to be developed.
- **Quality of Service (QoS) requirements**: The Quality of Service (QoS) is a set of parameters describing the tolerable end-to-end delay, throughput, and the level of reliability in multimedia communication and presentation. QoS requirements of clients are an important factor that affects the usage of the server. The server should be able to provide and adapt itself to different QoS requirements, according to the characteristics of the client's terminal, the network connection and the requested data type.
- **Cost effectiveness**: A very important requirement governing the future of multimedia servers is the cost effectiveness. The server must be affordable.
References

[VLA023] Integrated Environment for distance Learning, Aurel Vlaicu, Cosmin Porumb & co, COST #276, Budapest, October 2002

[VLA03] Integrated Environment for Distance Learning – ieL, Aurel Vlaicu, Cosmin Porumb & co, 2003 EDEN Annual Conference, Rhodes


[POR05] Rich Client Concept in Interactive Learning, Aurel Vlaicu, Sanda Porumb, Cosmin Porumb, Acta Technica Napocensis, January 2005

[VLA041] Client-Side Manager for Multimedia Distributed Services, Aurel Vlaicu, Cosmin Porumb & co, Bucharest, October 2004

[VLA042] Intelligent and Personalized Graphical User Interface in e/mCommerce Systems, Aurel Vlaicu, Cosmin Porumb, Sanda Porumb & co, Salonic, May 2004

[POR03R3] Cercetări privind sistemele multimedia distribuite, Referat doctorat, Cosmin Porumb, Cluj-Napoca, December 2003

[POR03R2] Metode și tehnologii folosite în aplicații IT distribuite, Referat doctorat, Cosmin Porumb, Cluj-Napoca, September 2003

[VAI02] J2EE, Aplicații Multimedia, Mircea Vaida, Cosmin Porumb & co, Cluj-Napoca, Mai 2002

[POR01R1] Stadiul actual al aplicațiilor multimedia distribuite, Referat doctorat, Cosmin Porumb, Cluj-Napoca, Decembrie 2001


[MAC] Macromedia Flash Remoting MX, Macromedia Documentation


[VAD99] Programarea in Java, Calin Vaduva, Cluj-Napoca 1999

[Mac03] Rich Client, Macromedia Flash MX Specification, Macromedia Documentation

[VLA011] Management and Assessment/Student Evaluation Tools for Open Virtual Educational Environments, Aurel Vlaicu, Cosmin Porumb & co, TEMPUS Programme, Tg Mures, 15.03.2001

[VLA012] Intelligent Agents for improving the performances in telecommunication and e-commerce, Aurel Vlaicu, Cosmin Porumb & co, COST #276, Madrid, November 2001

[VLA021] Intelligent Communication Environment using multiagent system, Aurel Vlaicu, Cosmin Porumb & co, COST #276, Florence, March 2002

[VLA022] Information Management using Multi-Agent Platform Systems, Aurel Vlaicu, Cosmin Porumb & co, COST #276*, Budapest, October 2002

[VLA023] Integrated Environment for distance Learning, Aurel Vlaicu, Cosmin Porumb & co, COST #276, Budapest, October 2002

[VLA03] Integrated Environment for Distance Learning – ieL, Aurel Vlaicu, Cosmin Porumb & co, 2003 EDEN Annual Conference, Rhodes