WP9 - Validation on Platform

Deliverable D9mm.3 - Multimedia Application V1

WP9 - Validation on Platform: Deliverable D9mm.3 - Multimedia Application V1
by Adrian Matellanes

Copyright © 2003 by Ocera

Table of Contents

List of Figures

Chapter 1. Introduction

1.1. Document Purpose

In this document we will describe the implementation of the Multimedia Application in its version 1 developed in the context of the OCERA project.

We refer to documents [OCERA D9mm1] and [OCERA D9mm1] for information about requirements and architecture not present in this document.

After a brief description of the application to make this document as self-contained as possible, we will present the implementation details of the multimedia application, called **VTWV**.

Chapter 2. Application Description

In this chapter, we will present a brief description of the application for document completeness. Further information on requirements and concrete architecture are described in [OCERA D9mm1] and [OCERA D9mm1].

2.1. Overview

Visual Tools will develop a digital video viewer for StrongArm based handheld platforms that will be integrated with its range of Digital Video Recorders and Transmitters (DVRT) using Wireless Communications. We will call this application Visual Tools Wireless Viewer, VTWV from now on. It is an application oriented to videosurveillance.

As already stated in previous documentation, this application is both, high resource consuming and running on a low performance platform (StrongARM) and thus the OCERA Resource Reservation components will be extensely used.

In what respects communications, the nature of the transmission media (radio) implies the necessity of controlling resources since, depending on particular conditions, the scenario is prone to interferences and the resource consumption can increase when transmiting the video stream.

We will develop an lightweight X Window application to receive wirelessly and present video streams transmitted by one (or several) Visual Tools DVRT. We will be able to find out which DVRT are present in the LAN and request them to stream video from any of its cameras.

Due to commercial reasons we will develop the application in a programming language suitable for an easy port to PocketPC, though the reference development environment will, of course, be done on a Linux platform using the OCERA components. This porting will, evetually, lack the benefits of the OCERA components wich are developed in and for a Linux environment.

2.2. Architecture

The development of VTWV will be done on a Compaq iPAQ 3850 with a Symbol 802.11b CompactFlash Wireless LAN card.

2.2.1. Linux Distribution

The target handheld will have the Linux Familiar distribution installed. See [OCERA D9mm1] for details.

The Familiar distribution will provide all external software required for the application. In particular, it will provide the different drivers needed for the CF Wireless LAN card, the iPAQ screen, etc... as well as specific software developed for this kind of platform, as the handwriting recognition software, orientation display manager, etc...

Instead of using the off-the-shelf kernel in the Familiar distribution, we will use the OCERA kernel, which is a vanilla kernel plus several patches that enhance its soft real-time capabilities, check deliverable D2.1 for further information about the OCERA kernel.

But we cannot make use of the standard OCERA kernel. In order to run a Linux kernel in the Compaq iPAQ we need to apply some patches to it that provide the specific drivers for this platform. Some examples of the particularities these patches deal with are the touchscreen, battery, etc... Once we apply those patches, we can boot the iPAQ and run the OCERA kernel with soft real-time capabilities.

2.2.2. Ewe

For the development of VTWV we will use the Ewe programming system (`http://www.ewesoft.com`).

Ewe allows to write Java programs for desktop computers as well as for mobile handhelds. It is particularly suitable for the development on handhelds platforms and will allow us not only to use it under the Linux kernel for the OCERA project but to port the application (though, unfortunately, making no use of the OCERA components) to the PocketPC OS, which is of particular importance from the commercial point of view.

VTWV have no hard real-time constraints, but in case we add, in future releases, some feature requiring hard real-time capabilities, we can still make use of Ewe since it provides an API for C extensions.

There are other options for Java development on mobile devices but we have chosen Ewe for its (rather) small footprint and its API completeness and its maturity. For a comparison between Ewe and PersonalJava see [Ewe vs PersonalJava].

2.2.3. Security

VTWV does not provide any security mechanism. We rely on the WLAN security mechanisms setup by the network administrator.

It is true that the current 802.11b wireless LANs are at risk of compromise. Since it is not easy to provide robust security mechanisms on the wireless LAN, to make the transmission as secure as possible, we will just encourage the use of WEP and, eventually, other propietary security mechanisms depending on the particular vendor that provides the wireless access points and stations. See [Arbaugh et al. 2001], [Walker2000] and [Borisov et al.] for details.

2.3. About Version 1

In this first version of VTWV we will ship the application with some restrictions on the original requirements and functionalities.

The aim of this first version is to get the OCERA kernel running on the Compaq iPAQ, to get the wireless connection working correctly and to ensure the detection and communication of the Visual Tools video servers on the LAN and finally to make the first experiments witht the OCERA resource reservation components.

The transmission protocol in this version 1 will be TCP/IP only. We will not provide streaming over UDP. We will also run the application within a network configured according to what we called *Basic Topology*, see Figure 2-1. For details about this topology check [OCERA D9mm1].

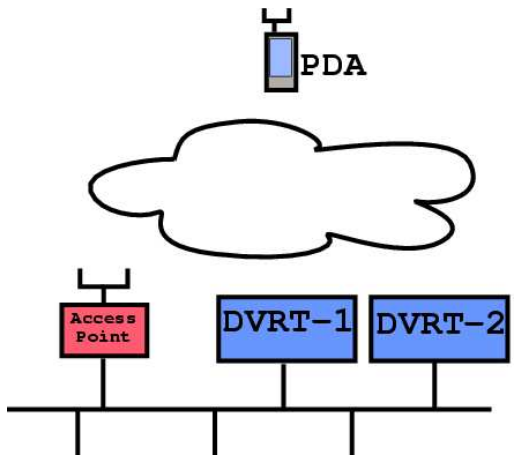

Figure 2-1. Basic Network Topology.

Chapter 3. Implementation

3.1. Methodology

Let's comment, in a few lines, the methodology we are following for the development of VTWV: Extreme Programming (or XP for short).

These are the guidelines of Extreme Programming:

- **Iterations**
  An XP project work in short iterations of one or two weeks.

- **Pair Porgramming**
  In an XP project all code is written in pairs. There are always two programmers working on the same computer, yes, computer!

- **Testing**
  In an XP project, the testing is done automatically. The team code the testing software the application has to pass. There are both **unit testing** and **acceptance testing**. Moreover, the testing software is written *before* the software itself.
  Unit testing validates each piece of software individually.
  Acceptance Testing validate the whole application. Since the development is **testing-driven**, the software is finished when the acceptance testing is passed. The software is developed to pass the tests!

- **Refactorization**
  An XP project work refactoring continuously the code.

- **Simple Design**
  The design has to be as simple as possible. You never do or design something for tomorrow. If tomorrow your needs change, "refactorize" the code.

- **Everybody owns the code**
  In an XP project everybody owns the code and then everybody can change any line!

- **Continuous Integration**
  Integration of the different modules is done continuously. There is always a machine with the latest software running and being tested. Since all code is released when it passes the tests, the integrated system always works. This way we avoid undefined integration periods.

3.2. Linux Kernel

The patches applied to the standard OCERA kernel to run on the iPAQ are `patch-2.4.18-rmk3` and `patch-2.4.18-rmk3-hh10` wich can be downloaded from the Linux Familiar Distribution website, `http://familiar.handhelds.org`. Fortunately the application of these patches was smooth and we had just to deal with some configuration issues. In particular the Familiar patches remove some of the configuration menus related to processor type and features, just the menu were most of the extra features of the OCERA kernel are located.

The Wireless LAN card driver, also particular for the iPAQ platform run smoothly.

3.3. Classes Structure

VTWV v1 is written in Java. Let's present now the classes structure.

3.3.1. `KoalaRequest`

This class is in charge of making requests to the video server. The protocol to command the video server is implemented in this class. Using this protocol, we can start and stop playing from the specified camera and video server.

The connection established to command the DVRT is done through a TCP/IP socket to the request port of the DVRT and there is one TCP connection per request, i.e., we do open a connection with the server, send the command, wait, synchronously, for the response and close the connection. When there is need for another request we open a new connection with the DVRT.

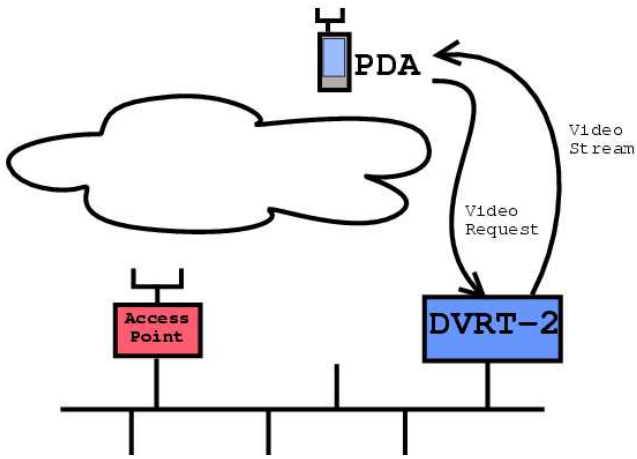

Figure 3-1. The handheld requests a video stream

3.3.2. `ImageReceiver`

This class is an abstract class that provides the common functionality needed to receive a video stream, independently of the compression codec used.

This class is in charge of connecting to the video server through the appropriate protocol (in V1 we will just have TCP transmission available) and receive the video stream.

3.3.2.1. `JPEGReceiver`

This class is in charge of receiving the JPEG frames sent from the server. We support a particular stream of JPEG images composed of a header containing image size, resolution, etc... and the proper JPEG buffer.

We use this class to ensure that a the whole image have been received and prepare it to be decompressed and displayed.

3.3.2.2. `MPEGReceiver`

MPEGReceiver is the corresponding class to receive the MPEG stream from the network. Using this class we get the MPEG buffer ready for decompression and display. See below how we deal with decompression and display.

3.3.3. `Buffer`

Class to manage byte buffers. Basically it is a wrapper of a byte array that provides extra information about it.

3.3.3.1. `DecodedBuffer`

Derived class of the Buffer class to hold the buffers of the decoded stream.

3.3.3.2. `EncodedBuffer`

Derived class of the Buffer class to hold the MPEG encoded buffers.

3.3.4. `FFMPEGDecoder`

This class is responsible of MPEG stream decompression. To decompress a MPEG stream we rely on native methods that in the end access the `ffmpeg` open source library.

The native methods are implemented using ENI, the Ewe Native Interface, with wich we are able to call the decompression routines of the open source library `ffmpeg`. In order to facilitate the development of these native methods, the Ewe SDK provides the Jewel tool that creates, from the `FFMPEGDecoder` Java class, the C++ source code squeleton which implements the native code.

Right now we have just implemented the decompression into RGB image format since that is what we actually need in order to render the image. In fact, `FFMPEGDecoder` native methods access other dynamic link libraries for YUV to RGB conversion, filtering, and MPEG user data extraction.

3.3.5. `DecoderException`

Class defining the exceptions produced during the MPEG stream decompression.

3.3.6. `VSFinder`

VSFinder is the Java class that finds out the Visual Tools DVRT present in the network. Once the detection process is finished, the discovered DVRTs are presented in a browsable list in the GUI. See below.

3.3.7. `VTWV`

VTWV is the class holding the GUI. It contains the design and behaviour of the graphical user interface. We have, however, separated as much as possible the engine from the user interface. This has been specially useful, other considerations apart, to be able to build the automatic tests corresponding to each class.

The GUI is composed of one panel that contains the image to be displayed and two browsable lists that contain the DVRT present in the local area network and the corresponding cameras. These lists allow the user to select which server and camera she wants to play video from. See Figure 3-2.

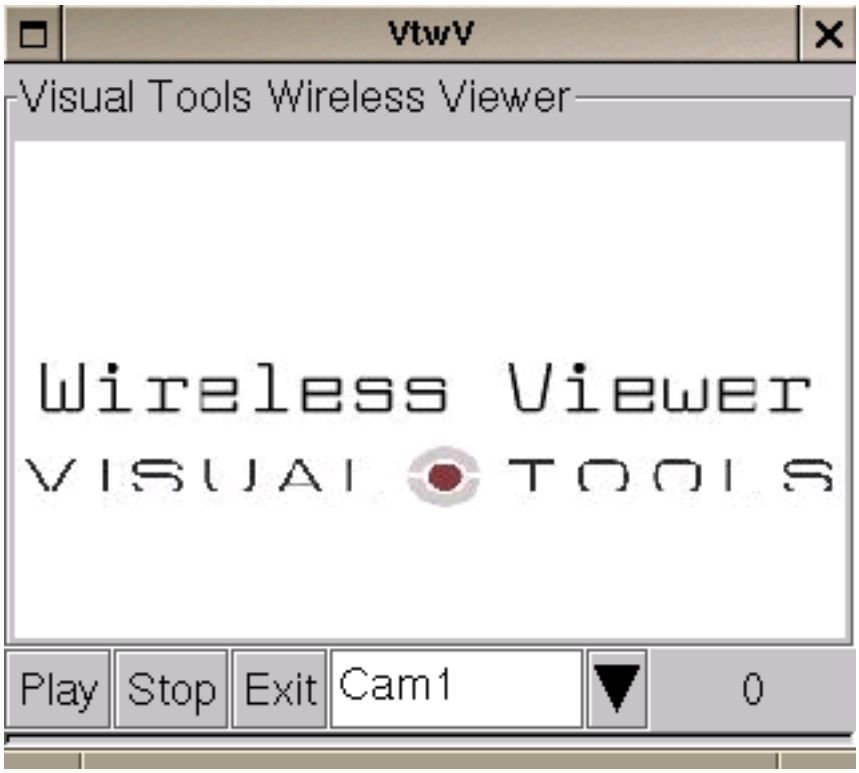

Figure 3-2. VTWV screenshot with debug play/stop buttons.

Details about the expected behaviour of the application are given in the Acceptance Testing section.

Chapter 4. Testing and Benchmarking

4.1. Testing

As already stated in the Methodology section above, there are two main steps in the testing of VTWV, the unit testing and the acceptance testing.

4.1.1. Unit Testing

For each class presented in the Classes Structure section there is a testing class. Therefore we have testing classes to validate:

- The DVRT Request protocol

  This class, TestKoalaRequest will validate the correctness of the protocol to be used to command with the video server.

- The streaming of video over the WLAN

  There are two classes that take care of the streaming: `TestJPEGReceiver` and `TestMPEGReceiver`. These classes will validate, respectively, that VTWV receive the streams sent from the server through the network.

- The decompression of video the stream

  The class to test the decompression of the video stream received from the server is, TestMPEGDecoder. Notice that there is no need for a decompression class for JPEG since it is built-in into Ewe.

- The discovery of Visual Tools video servers on the LAN

  This class, TestVSFinder will test that all Visual Tools video servers present in the network are found by VTWV.

4.1.2. Acceptance Testing

The acceptance testing will indicate whether we provide the functionalities needed or not and if they work according to specification. Aside from the fact that the streaming application "runs okay", we have done testing to verify that the appropiate use of the OCERA resource reservation components give the desired results.

The basic points in the acceptance tests are the following:

1. Discovery of the DVRT present in the LAN.

   VTWV will find all the Visual Tools DVRT present in the LAN. To make the GUI as simple as possible this is done only at the application start. If the user plugs another DVRT into the local area network and wants the application to detect it, she must reboot the application. This decision has been taken because it is not likely that new DVRTs appear on the LAN very often and we can thus save resources: CPU used to discover the new ones while probably we are receiving a video stream and network bandwidth. It is also important to keep the GUI as simple as possible since we are targeting small devices.

2. Selection of a video server and camera.

   All discovered video servers will be presented in a browsable list. The user can select one of those servers at a time. Once the user has selected one server, the desired camera from that server can also be selected from a browsable list.

3. Video playing.

   So far the application has detected the DVRT present in the network and the user has selected the desired video server and camera.

   From the very first moment of camera selection, the application connects to the video server and request a video stream.

   The DVRT requested proceeds to stream live video of the selected camera until the user selects whether other camera or another video sever. In each case the application request the corresponding video stream to the appropriate server.

   In order to stop streaming the user have to stop the application, again, to facilitate the use and save GUI we have avoided the inclusion of a STOP button or any other mean to tell the application to stop the video playing.

4.2. Benchmarking

We explained in [OCERA D9mm1] that VTWV is not a hard real-time application but a soft real-time application.

Therefore we will not have to meet specific deadlines or satisfy any kind of hard Real-Time constraint. Nevertheless we will provide extensive information about the behaviour of VTWV under several load conditions, and resource reservation configurations.

In particular we will check how the resource reservation components actuation will affect:

- Latency
- Framerate
- Wireless LAN range (a priori not related to resource reservation)
- Delay when changing camera from the same server
- Delay when changing video server
- Network consumption

Bibliography

[OCERA D9mm1] Adrian Matellanes, 2003, *"Multimedia Application Requirements and Platform Analysis" http://www.ocera.org*.

[OCERA D9mm2] Adrian Matellanes, 2003, *"Multimedia Application Component Specification" http://www.ocera.org*.

[OCERA D2.1] Adrian Matellanes et al., 2003, *"OCERA Arquitecture and Components Integration" http://www.ocera.org*.

[Familiar] `http://familiar.handhelds.org` .

[EWE] Michael Brerenton, 2003, *"Ewe Application Development" http://www.ewesoft.com*.

[Ewe vs PersonalJava] Michael Brerenton, *"Comparison of the Ewe VM with a PersonalJava VM" http://www.ewesoft.com/EweDetails/VersusPersonalJava.htm* .

[Arbaugh et al. 2001] William A. Arbaugh, Narendar Shankar, and Y.C. Justin Wan, 2001, *"Your 802.11 Wireless Network has No Clothes"*.

[Walker2000] J. Walker, March 2000, Tech. Rep. 03628E, IEEE 802.11 Committee, *"Unsafe at any size key: an analysis of the WEP encapsulation"*.

[Borisov et al.] N. Borisov, I. Goldberg, and D, Wagner, *"Intercepting Mobile Communications: The Insecurity of 802.11."*.