A generalized low bit rate video distribution system for the Wearable Command Unit

Final report

Author
MIKAEL CORP, student

Industrial Supervisor
JOHN KESSLER, Saab Security Systems AB

Academic Supervisor
Dr. VLAD VLASSOV, Royal Institute of Technology

Abstract

A solution for the distribution and viewing of live video in the WCU 2.0 system is developed and implemented.

The WCU system is a command and control system for crisis management and infrastructure protection. The main goal of the WCU system is to bring situation awareness to its users, by using digital maps, voice communication, and text messages. As another way of conveying information, it was decided to extend the WCU system with video capabilities.

The WCU system is inherently mobile, which sets the constraints for a video solution. There are limited bit rate and processing capabilities available. A general solution was sought, one that would allow readily available low-end cameras to be connected with a minimum of configuration required.

The work included a thorough investigation of the current state of the art regarding digital video. A survey of the existing WCU architecture was also carried out. A few test pilots were constructed in order to evaluate the properties existing solutions. Finally, a design was proposed and implemented.

The solution shows that the proposed design works. The bit rate constraint seems to have a quite high impact on the results; the perceived quality of service with regards to the video signal is fairly low.

Also, the devised solution was not general in the sense that any camera may be connected. Still, the lack of standardization regarding the interface for accessing the video stream hinders a true generalized solution with the proposed solution.

Sammanfattning

En lösning för distribution och uppspelning av direktsänd video i WCU 2.0 utvecklas och implementeras.
WCU-systemet är ett ledningssystem för krishantering och skydd av infrastruktur. Målet med WCU är att ge lägesbild till sina användare, genom att använda digitala kartor, text- och röstmeddelanden. Som ett ytterligare sätt att överföra information, beslöts att WCU ska utökas till att hantera direktsänd video.
WCU-systemet är naturligt mobilt, vilket sätter villkoren för en tillämpning av video. Främst ligger begränsningen i låg bandbredd och relativt låg processorkraft. En generell lösning söktes, som tillåter att befintliga enklare kameror kan anslutas med ett minimum av konfigurering.
Arbetet inkluderar en grundlig undersökning av läget på forskningsfronten inom digital video. WCU-systemets arkitektur undersöktes också. Några testfall genomfördes för att testa existerande lösningar och dess tillämpbarhet i fallet med WCU. Ett lösningsförslag lades fram och implementerades.
Lösningen visar att förslaget fungerar. Villkoret med låg bandbredd visade sig ha en hög genomslagskraft på resultaten; den uppnådda servicekvaliteten vad gäller videosignalen upplevdes som ganska låg.
Den implementerade lösningen är inte så generell att den tillåter en godtycklig videokamera att anslutas. Bristen på standardisering i sätten på hur man kommunicerar med kamerorna hindrar en sann generalisering i den föreslagna lösningen.

Edition history

Edition	Date	Major changes
1.2	2007-04-18	Final changes after revision by examiner and opponent: Conclusions from section 4.6 added to chapter 7. Added a discussion on hybrid network architecture to section 4.4.3 and 7.2.2. Clarified comment on video quality in section 6.2. Clarified statement on RTSP in section 7.2.2. Clarified definition of low bit rate in section 2.3. Extended paragraph on existing pre-study section 4.5. Clarified paragraph about cameras used in section 4.5.5. Added legend to table in section 4.6.1. Changed a typo in section 5.1. Clarified a constraint in section 5.4.3.
1.1	2007-04-16	Added table with network cameras in section 5.5.
1.0	2007-04-08	Release version.
P1.0-4	2007-03-28	Expanded section 3 "Method". Expanded section 6.2 "Evaluation". Added a product-feature matrix in section 4.6. Added a solution strategy table in section 4.6. Added a figure in section 5.4. Several minor changes from revision by Vlad Vlassov.
P1.0-3	2007-02-21	Moved section on "Application and transport protocols" to Appendix B. Moved section on error-resilience to appendix. Expanded some sections in chapter 7 "Conclusions".
P1.0-2	2007-02-20	Changed the title. Changed page numbering in appendix section.

		Added Appendix E. Added 2.4 “Delimitations”. Moved section about cameras to Appendix. Moved section on transcoding complexity to Appendix C. Moved section on license issues to Appendix D. Changed title of section “Application and transport protocols”. Changed section 4.4 “Network architecture”. Rewrote section 4.6 “Conclusions”. Moved Use Cases to section 5.3.
P1.0-1	2007-02-19	First draft.
P1.0-0	2007-02-05	Created.

Table of contents

1.1 Table of figures

2 Introduction

2.1 Background

The Wearable Command Unit (WCU) is a command and control system developed by Saab Security Systems AB in Järfälla. The main goal of the WCU is to bring common situation awareness to its users [23]. It is a delicate task; the information must be accurate, it must not saturate the user, nor must it withhold any vital pieces.

The WCU system is intended to be used in crisis management situations and infrastructure protection scenarios.

The user may be mobile or stationary, and each user assumes a pre-defined role. In its current version, there are three roles: Command and control (C2) client, field client, and smart phone client. They will be covered in more detailed later.

The WCU is based on a centralized system, with the server acting as a pure message-broker. If a client wants to send a message to another client, it is the job of the server to distribute it to the intended receiver.

As another way of providing information to the user, the management at Saab Security Systems has decided to extend the WCU system to support live video transferring and viewing capabilities.

As a result, this project was initiated. It was supervised by John Kessler, product manager of the WCU version 2.0, and Dr. Vladimir Vlassov of the Royal Institute of Technology.

2.2 Problem statement

Digital video is not new to the world of computing, there has been extensive research on the topic and there are well-defined standards covering many aspects of the technique. However, digital video is new to the WCU concept and the system was not designed with video in mind. Compared to other features of the WCU system, video is a relatively demanding application.

What is interesting for this work is if a video distribution system would work in the WCU context, and if so how the solution would look. The main constraints in the WCU system are the relatively low bit rate and the computational resources available. Furthermore, a video system may not utilize all the available resources. The WCU system has other components that must not be starved of resources.

The WCU system is marketed towards customers with different needs. For some, video surveillance is very important, and they may have an existing video infrastructure that needs to be extended into the WCU. For others, video may be an interesting option but their budget does not allow state of the art solutions. For them, a solution comprised of cost-efficient cameras that are easy to connect and maintain and that fits into the WCU context would suffice. Also, since the advancements in digital video research has been quite fast during recent years, and is expected to continue to be, it is undesirable to be locked to a proprietary solution. The solution should seek to allow arbitrary cameras to be connected to the system.

Specifically, the issues to be resolved are:

- Existing techniques and software: What techniques are currently used and what is coming in the near future?
- Network architecture: How to distribute the sensor data to the client.
- Design issues with regards to building the prototype: Making the prototype fit into the existing WCU platform.
- Variable or adaptive data transfer bit rates: With variable network conditions, does the transfer rate have to be variable too?
- Remote controlling the camera: Is this possible with a generalized solution? Are there any standards?

2.3 Low bit rate

In order to define low bit rate for this work, a series of tests were setup to determine the throughput and latency of the common wide-coverage wireless networks, such as GPRS/EDGE and UMTS. The tests were performed several times over a few weeks in the fall of 2006.

Since the video sensors will be connected on these links, the limiting factor will be the uplink transfer rate. The tests showed that the maximum transfer rate was around 64 kilobits per second.

2.3.1 Network latency and throughput

To measure the TCP packet throughput, a tool named TPTEST5 [71] was used. The reference server was referens.sth.ip-performance.se (192.36.144.178). Cards tested: AirCard 775, Globesurfer iCon. Network provider: Telia. The AirCard was around 8% faster downstream and 23% faster upstream. Interestingly, the EDGE upstream throughput was consistently higher than UMTS.

UMTS throughput	Kilobits per second
Test runs	6
Median TCP downstream	360.6
Median TCP upstream	57.5

EDGE throughput	Kilobits per second
Test runs	8
Median TCP downstream (GlobeSurfer)	193.7
Median TCP upstream (GlobeSurfer)	65.5
Median TCP downstream (AirCard)	209.9
Median TCP upstream (AirCard)	84.5

To give an estimate of the wireless network latency, the network tool ping was used to observe the round-trip times. The reference server was the main KTH web server, www.kth.se, (130.237.32.107).

GPRS/EDGE round-trip times	Milliseconds
Ping packets sent/received	140/140
Median	432
Mean	436
Variance	3229

UMTS round-trip times	Milliseconds
Ping packets sent/received	138/138
Median	249

Mean	257
Variance	1093

2.4 Delimitations

2.4.1 Human-machine interface

This work is about determining whether a solution to the problem exists, and if so implementing a proof of concept within the WCU context. The importance of a systems user interface must not be underestimated, but in this work the GUI is not emphasized upon.

2.4.2 Security and privacy

Any time there is inter-machine communication, or there is a mere possibility thereof, security and privacy becomes an issue. There has been extensive research on that field in the WCU 2.0 project, but for this work, it is not a major factor.

2.4.3 Licensing issues

As previously stated, there has been extensive research in the field of digital video and the transport mechanisms involved. Hence, a range of commercial companies have patents on various parts of the techniques. In order to use any of this work, a thorough investigation of the licensing issues must be performed but that is outside the scope of this work. There is a short overview in Appendix A.

2.4.4 Time constraints

As always, time is a limiting factor. This work was performed during 20 hectic weeks and some parts were not possible to perform in that time span. The reader is encouraged to read section 7.3 "Future work".

2.5 Outline

Chapter 3 describes the method used to solve the problem.

Chapter 4 contains a survey of the state of the art in digital video and related areas, such as application and transport protocols. It also contains a detailed view of the relevant WCU components. Furthermore, it has an overview of existing solutions, and describes some test scenarios used to evaluate the solutions. There are descriptions of the cameras that were procured for this work.
Chapter 5 deals with the proposed design and the implementation.
Chapter 6 contains the analysis of the solution, including validation and evaluation.
Chapter 7 holds the conclusions.
Chapter 8 has an index of the referenced sources, and a table of the figures.

2.6 Acronyms, abbreviations and definitions

Keywords "must", "should", "required" etc. are used in line with the RFC 2119 "Key words for use in RFCs to Indicate Requirement Levels" [6].

Word, abbreviation, or acronym	Description
ActiveX	A specialized type of OLE. See section 4.2.2.
COM	The Microsoft Component Object Model. See section 4.2.2.
GPRS	General packet radio service; a data service available in some GSM networks.
ISO	International Organization for Standardization. See section 4.3.1.
Low bit rate	In the remainder of this work, low bit rate is defined as less than 100 kilobits per second.
.Net framework	A class library and a runtime for managed code from Microsoft. See section 4.2.2.
Managed code	Code that has its execution managed by the .Net framework runtime, in a way that the runtime always can retrieve information specific to the current CPU instruction, such as register or stack memory contents. In that way, the runtime knows what the application is about to do and can make certain guarantees, such as garbage collection, type

	safety and array bounds checking.
MPEG	Moving Picture Experts Group. See section 4.3.1.
OLE	Object Linking and Embedding. See section 4.2.2.
PSNR	Peak signal-to-noise ratio. See section 4.3.3.
Sensor	A device that generates a signal that can be interpreted or measured.
UMTS	Universal Mobile Telecom System; a 3rd generation mobile phone network standard.
Unmanaged code	In contrast to managed code, unmanaged code is a binary image loaded into memory. The program counter gets to points to the first address and the OS can make no assertions of what the code will do.
WPF	Windows Presentation Foundation. See section 4.2.2.
WCF	Windows Communication Foundation. See section 4.2.2.
WCU	Wearable Command Unit.

3 Method

This work was divided into several distinct phases. These are:

- Survey of the state of the art
- Test pilots
- Proposed solution
- Implementation
- Evaluation and validation

A fairly thorough survey of the state of the art regarding digital video, and a survey of the WCU design, was performed. The survey of digital video focused on the different international standards, and also covers the technique briefly. Even though it was decided to procure cameras with built-in video servers, it was necessary to study the nature of digital video in case some new module had to be written. The survey was mainly done by reading articles from various scientific publications. The relevant findings are reproduced in this document. Some findings that are considered important for the subject in general but not for this work in particular, can be found as an appendix to this report.

To see how the different existing solutions would work under the constraints of this particular problem, a few test pilots were performed. The results they yielded were less desirable, but some good lessons were learned from them. The test pilots were setup in the computer lab at Saab Security Systems in Järfälla. The lab was quite adequate for this work, with plenty of computers with different configurations as well as different networks in place, such as LAN, VPN and the common wireless network types. A number of tests were setup, to see how the different solutions would cope with latency, moving picture quality, and packet loss. The test pilots are described more detailed in section 4.5.5.

After the survey of digital video and the WCU, and the test pilots, a design for the solution was devised. The design had to conform to the existing WCU architecture and was implemented directly on the existing production code base.

The evaluation and validation was also performed at the Saab Security Systems in-house lab. With regards to networking equipment, the mobile network links (GPRS/UMTS) were leased from Telia. To validate the solution, the use cases defined in the design phase were used. To evaluate the solution, measuring the PSNR was chosen as the method of assessing quality of service. A reference video sequence was sent from one client over a UMTS link, and stored on the viewer machine. Then the PSNR was calculated on a frame-by-frame basis and finally the median value was taken.

4 Current state of the art

4.1 Overview of this chapter

In order to fully understand the subject, there was a need for a thorough literature study. There are a few fundamental parts needed to be investigated. One is of course the different international video standards. Also, how to transport video data over a network is an important part; this section is further divided into a network design part, and a part dealing with the existing protocols. I also look at existing solutions for video distribution.
To be able to implement a solution in the WCU, there is a chapter covering the current general design of the WCU and in particular the modules that will be affected by adding a video module.

This chapter is structured as follows:
Section 4.2 describes the Wearable Command Unit, which is in focus for the master thesis. It also includes an overview of the existing software architecture with descriptions of interfaces and classes.
Section 4.3 covers video compression and the current status in the international standards area. It includes a brief introduction of the coding techniques, quality of service, and error-resilience.
Section 4.4 looks at relevant network configurations.
Section 4.5 is about existing solutions, and lists the cameras that were procured for this work.
For a deeper coverage of some of the issues, the reader is encouraged to refer to the appendices. They are:

- Appendix A – Brief overview of the technique behind video coding,
- Appendix B – Application and transport protocols,
- Appendix C – Transcoding complexity,
- Appendix D – License issues,
- Appendix F – Error resilience in digital video.

4.2 Description of the WCU

The Wearable Command Unit (WCU) is a system for communication between different users, both mobile and stationary. One of its main goals is to establish "situation awareness" among its users. It has its heritage in military command and control (C2) systems.

The WCU client software operates on various hardware platforms, ranging from a laptop to a tablet pc to a PDA.

It is developed by Saab Security Systems in Järfälla.

The system is used by fire departments, rescue units, security guards and police.

4.2.1 General architecture

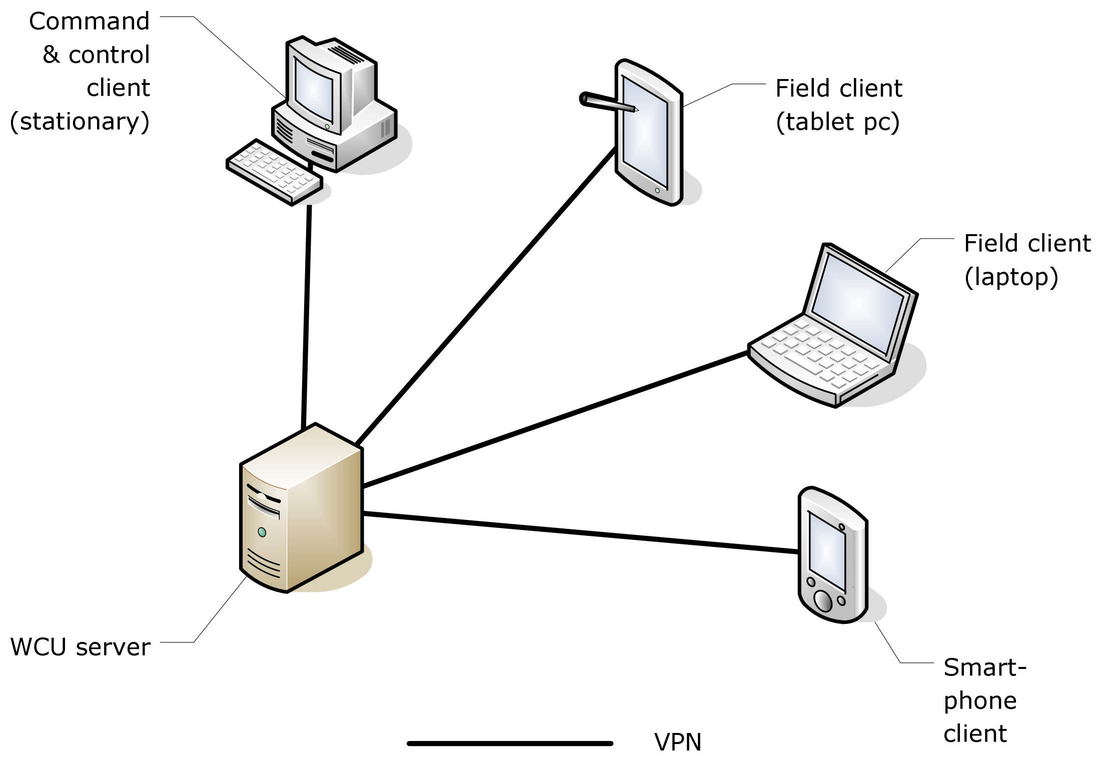


Figure 4-1: An overview of the WCU network.

In the WCU, there are two basic entities; the server and the client. There is only one kind of server, but the clients may be different, or rather assume different roles. There is a command and control client (C2), which acts as an administrator of the network. Its role is to keep a full view of the situation, and may command other units on dispatch. Another type of client is the field client.

The field client has the same view of the situation as the C2 client, but may not issue any commands in the same way as the C2 client. The smart-phone client is a stripped-down field client, adapted to be used on a limited device. Its main functions are navigation and text-messaging.

The WCU network is an overlay network, in the form of a specialized VPN. This brings several advantages:

- The traffic is encrypted,
- Clients on different networks (such as GPRS/3G) are treated as nodes on the same network,
- Some extra features are added such as reconnection of dropped links, access control of lost clients, and bandwidth throttling.

The server

The server basically functions as a message broker; distributing messages among the clients. There is no direct client-to-client communication; every message passes through the server. Each client has assumed a role, based on which the server manages subscriptions to various levels of information.

The server also keeps a database containing the data for each object in the system.

Figure 4-2: A WCU field client.

The command & control client

The command and control (C2) client handles and controls the available resources. It has a full view of all activities such as alarms and events, and it can dispatch units directly by drawing on the map. The C2 client is usually installed on a desktop or a high-performance laptop. It may have several wireless network adaptors, including GPRS, UMTS, and 802.11b/g.

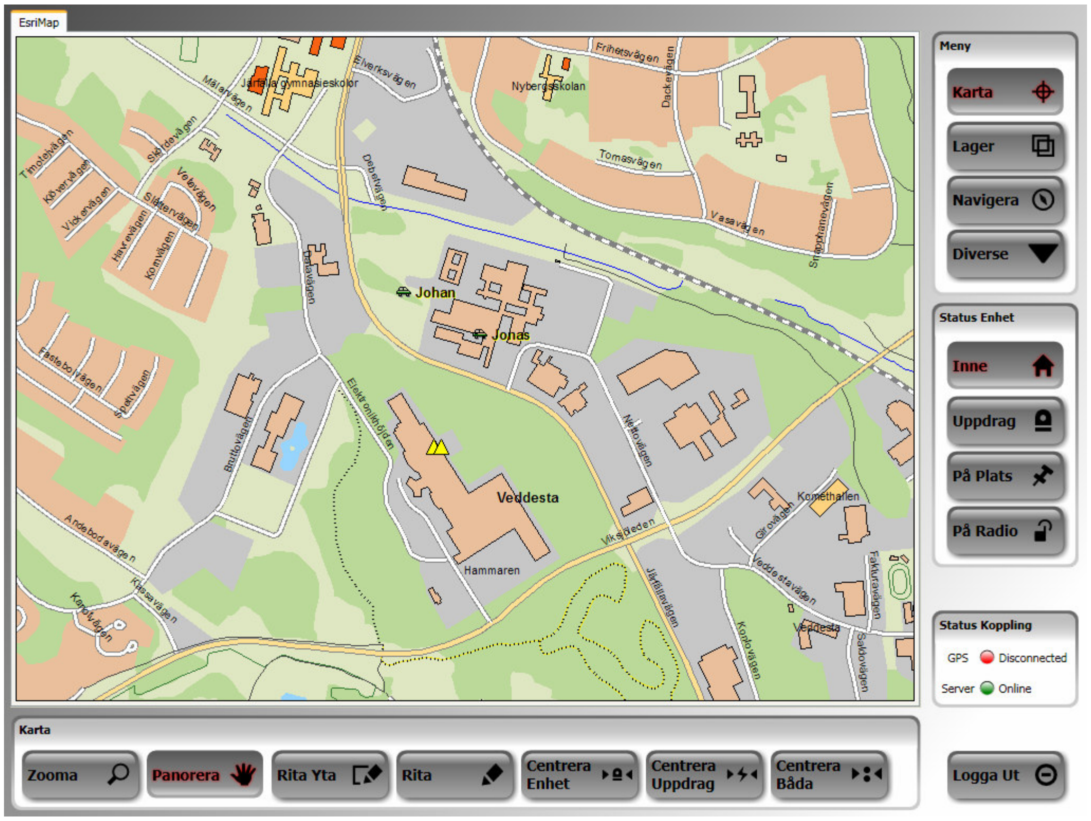


Figure 4-3: A screen capture from the field client.

The field client

Like its name reveals, it is used on the field and hence it has to be easy to carry. It is usually installed on a tablet pc or a laptop. It comes with a GPS receiver that continuously informs the C2 client of its position over the network interface. The geographical information system is the main feature of the field client. On the screen, the user can view other units or other things, such as alarms or events, as symbols. It may have several wireless network adapters, including GPRS, UMTS, and 802.11b/g.

The smart phone client

The smallest device used in WCU is the smart phone client, designed to run on a smart phone or PDA. Essentially, it is a stripped-down version of the field client, mainly missing file transfer capabilities, chat, and will initially not be considered for video applications.

4.2.2 The WCU inner workings

The design of the WCU 2.0 is traditional server-client architecture. The server actually has very few tasks; it mainly handles client authentication and the database. The other functionality is added through a plug-in architecture. The plug-ins are well-defined and can be loaded by any client, as long as the plug-in itself allows it. For example, the map is a plug-in that can be loaded by every kind of client. The GPS functionality is defined in another plug-in. Different plug-ins may also require certain WCU user privileges; the command and control client may require administrator rights to run, etc.
With this design proposal, there will be a video plug-in capable of receiving, decoding and displaying video streams. Also, the map plug-in should display symbols representing the video sensors.
Any type of WCU client should be able to load the video plug-in.

The framework

The WCU 2.0 is built tightly on top of the Microsoft .Net framework 3.0; which was released in its final form in November 2006. The new version is a core component of the Windows Vista operating system released at the same time. In the 3.0 release, Microsoft has created four new distinct areas: Windows Communication Foundation, Windows Presentation Foundation, Windows Workflow Foundation and CardSpace. The first two are the most relevant for the WCU and will be covered briefly below.

Windows Communication Foundation (WCF)

In an effort to collect all the different communication techniques (e.g. COM, DCOM, and RMI) from earlier versions of the .Net framework, the WCF initiative was taken [29]. According to that source, the main goal of WCF is to unify all different techniques into one, which is meant to be the very best option in all cases. It should provide performance that is about as good, if not better than, any other alternative.

I will let that last statement remain unchallenged for now, since it is outside the scope of this work to verify its validity.

In WCF, all communication is centered on the concept of a service, heavily influenced by web services.

Windows Presentation Foundation (WPF)

WPF is perhaps the biggest package of the new additions to the .Net framework. It is meant to be a replacement to Windows Forms, and is like a mix of dynamic web content and old Windows Forms. One of its drivers was to formally separate the user interface with program logic. In version 2.0 of .Net framework, using Windows Forms was the preferred way of generating the UI. Many user controls in Windows Forms are ActiveX components.

Microsoft has devised a new markup language called XAML. Based on XML, it is too a descriptive language and is used as a serialization format for objects from the WPF presentation stack [42].

From a user interface perspective, XAML-based applications may deliver more advanced content in a programmatically easier way.

OLE, COM, and ActiveX

In the late 1980's, Microsoft developed a system and protocol for distributed objects, named "Object linking and embedding" (OLE). The technique enables developers to embed controls and other objects very easily in applications. In 1993, Microsoft introduced the Component Object Model (COM). It was designed to allow implementation of objects so they can be used in other environments than they were created in, removing the dependency to any specific language.

In the late 1990's a part of this technique, which was then renamed to ActiveX, became popular on the web, by embedding different kinds of ActiveX controls such as video players and document readers.

When the first version of the .Net framework was released in 2001, it had to have support for COM objects since most of the existing Windows software was based on COM. The .Net framework 2.0 has good support for COM objects via built-in wrappers. And now, with the release of WPF, the same objects require yet another integration method.

To use ActiveX controls in WPF, an intermediate host must be used. This requires the UI to be wrapped twice; first wrapping the ActiveX control in a Windows Form, then wrapping the Form in a class called `WindowsFormsHost`.

4.2.3 Relevant WCU components

Internally, an empty plug-in should look like Figure 4-4. In the following sections, the relevant classes and interfaces are described.

Adding a plug-in such as video with the current requirements also requires some changes to the map plug-in. Therefore, the map plug-in design is also discussed.

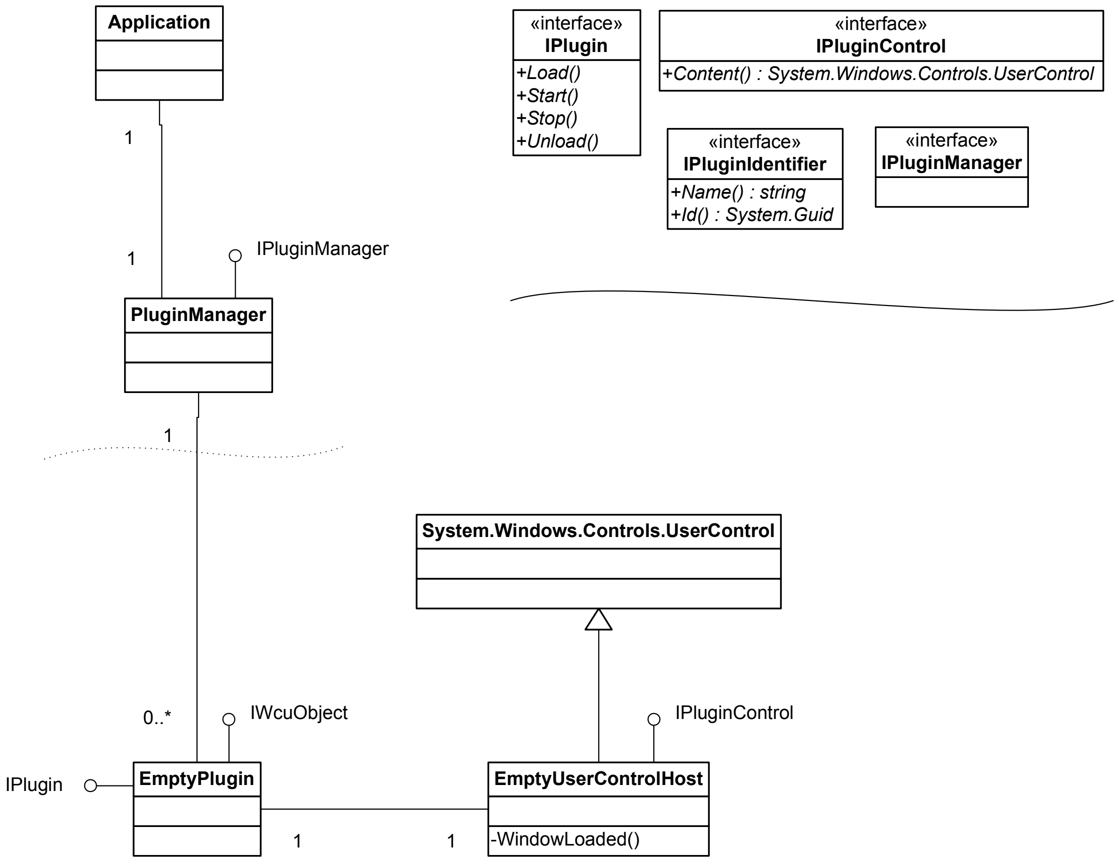


Figure 4-4: A class diagram showing an empty plug-in.

The `Application` and the `PluginManager` classes belong to the WCU framework.

The plug-in manager initializes an instance of the `EmptyPlugin`, which in turn creates an instance of the `EmptyUserControlHost`. A reference to this instance is passed to the `Application` instance via the `RegisterExtension` method. When the `Application` has added the plug-in to its GUI, the `WindowLoaded` method is invoked on the `EmptyUserControlHost` instance. Then the plug-in is fully loaded.

Framework interface descriptions

The most relevant interfaces are covered here. They are already part of the WCU framework.

IPlugin

`IPlugin` declares five important methods, whereof four deals with the lifetime of a plug-in. They are: `Load`, `Start`, `Stop`, `Unload`, and are invoked by the manager in that particular order.

- `Load():` The plug-in may listen to events, but may not fire any events.
- `Start():` The plug-in may fire events. Always invoked after Load.
- `Stop():` The plug-in may no longer fire events. Always invoked after Start.
- `Unload():` The plug-in must de-allocate all resources, stop all threads and dispose all UI components, etc. It is always invoked after Stop.

The fifth method is `OnPluginSettingsValueChanged`, which is invoked by the manager if any of the settings for a plug-in is changed.

IPluginIdentifier

This interface contains two abstract methods: `ItemName` and `Guid`. The first returns a string, the second a `Guid` instance. The purpose is to allow the manager to tell plug-ins apart.

IPluginControl

Inherits: `IPluginIdentifier`

It declares one abstract method: `Content()`, which returns the `this` pointer.

IWcuObject

This is the interface of the atomic unit of the WCU system. This interface declares several events and methods for event handling.

IWcuCommand

Inherits: `System.Windows.Input.ICommand`

This interface adds two identifiers, `name` and `guid`, to the .Net `ICommand` interface.

The `ICommand` interface works basically as a function pointer that may be passed around between different plug-ins. It declares two methods; `CanExecute()` and `Execute(object).` The first returns a boolean indicating whether the instance is able to execute, and an implementation of the latter contains the definition of the method to invoke.

ICommandManager

The `ICommandManager` interface declares methods `Register()`, `Unregister()` and `Execute()`. The names are rather self-explanatory; any implementers should handle the book-keeping of plug-ins with the first two methods. The `Execute(IWcuCommand, object)` method will be invoked by a client framework `Application` instance. The first parameter is the command and the second is the parameter(s) for the command. It may also be `null` if the command expects no parameters.

Framework class descriptions

The most relevant classes are briefly covered here. They are already part of the WCU framework.

Application (client framework)

Inherits: `System.Windows.Application`
Implements: `IApplication` (client framework)
This class serves as the top-level instance in the WCU 2.0 client architecture. It holds references to all the different managers, including the plug-in manager and the command manager.
It also launches the client GUI.

PluginManager

Implements: `IPluginManager`
This class is responsible for all the plug-ins. It finds all available plug-ins from a local storage directory, and then loads them.

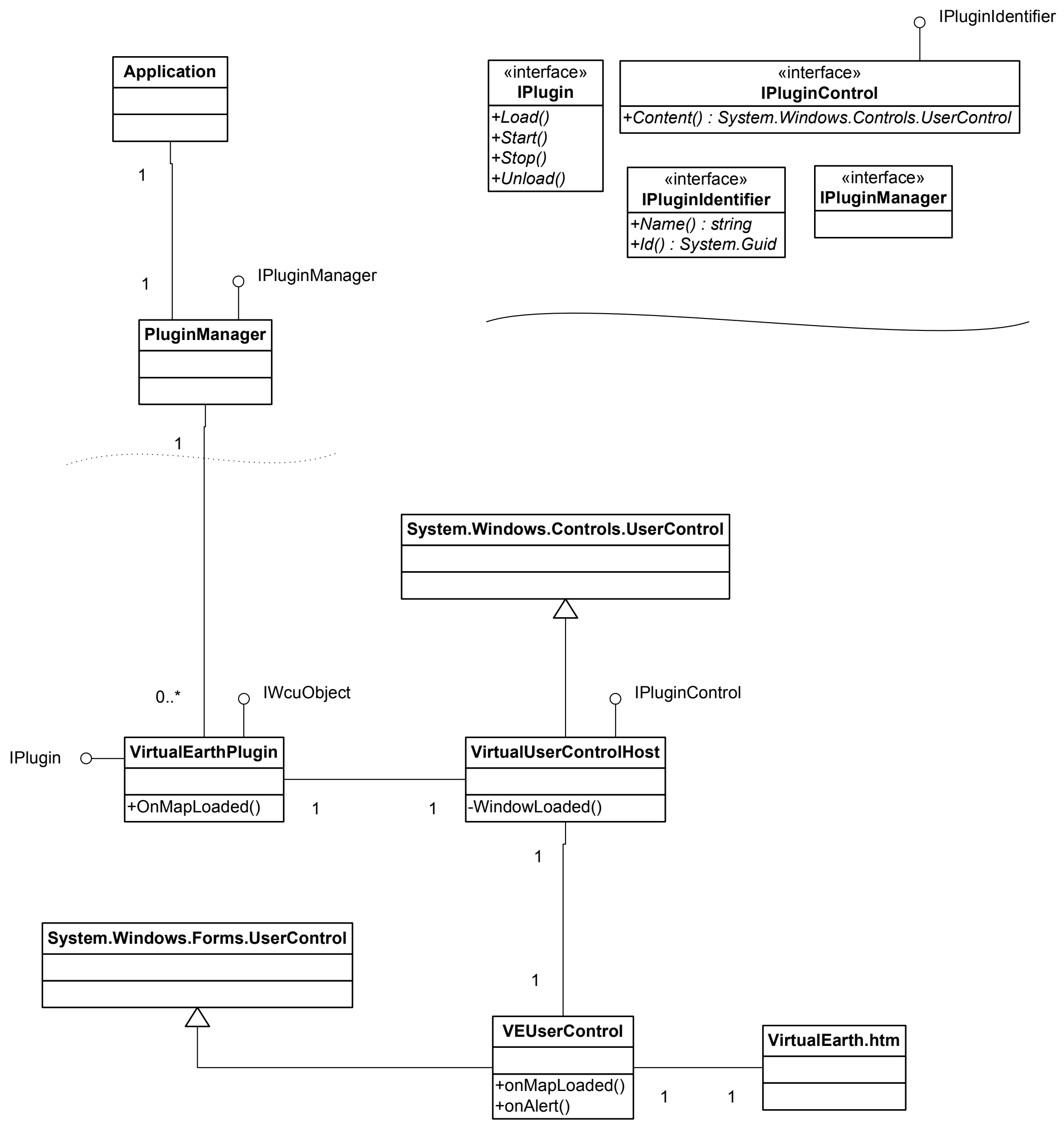
CommandManager

Implements: `ICommandManager`
The purpose of this class is to handle the book-keeping of plug-ins. It exposes the `Execute()` method that is invoked by instances of the client framework `Application` class.

Virtual Earth overview

Microsoft has developed a mapping and location service called Virtual Earth. It has general mapping functionality and comes with a fairly extensive API that enables developers to add custom layers, symbols and routing functionality. The core component is an html file with embedded JavaScript code to manipulate the objects. In default mode, the map tiles are downloaded dynamically, but the systems allows for local caching. In the WCU, the Virtual Earth plug-in is connected with the GPS plug-in and is used for mapping and routing.

Virtual Earth plug-in class descriptions



VirtualEarthPlugin

Implements: `IPlugin`, `IWcuObject`

When an instance of this class is created and has its `IPlugin.Load()` method invoked, it creates a `VirtualEarthUserControl` instance. A reference to this instance is registered with the parent `Application` instance. It also defines a delegate method, `OnMapLoaded`, which is registered as an EventHandler with the `VirtualEarthUserControl`.

VirtualEarthUserControl

Inherits: `System.Windows.Controls.UserControl`

Implements: `IPluginControl`

This class is the user control which the user interacts with. However, since the Virtual Earth object is an html file, which is hosted by a `System.Windows.Forms.WebBrowser` instance, the `VirtualEarthUserControl` needs to hold a `WindowsFormsHost`. The html file is hosted by the `VEUserControl` class.

VEUserControl

Inherits: `System.Windows.Forms.UserControl`

This class is a wrapper class for the methods in the Virtual Earth API that are used in the WCU. The Virtual Earth html file is loaded into a `System.Windows.Forms.WebBrowser` instance. The Virtual Earth object is interacted with by calling a general `InvokeScript` method on the document of the `WebBrowser` instance, and supplying the script name and a parameter. It also exposes two callback methods for the Virtual Earth object; `onMapLoaded` and `onAlert`. It is possible to add more callback methods. The former method is invoked when the map is loaded in the GUI and it fires the `MapLoaded` event upwards. The latter is used for error notification. This class also keeps a dictionary with all the current objects in the map for easy referencing.

4.3 Digital video

Simply put, digital video is moving pictures that have been digitized to be viewed, stored, or processed on a computer.

Historically, the major obstacle for digital video is the amount of raw data that is required when an analog video signal is digitized. As an example; a typical TV quality digital signal, with no compression, 720 × 576 pixels at 25 Hz in RGB mode with 8 bits per color, would require a data stream of 720 × 576 × 25 × 3 × 8 ≈ 248 M bits per second. Storing 90 minutes of such a stream requires around 156 GiB.

This is where compression, or encoding, comes in. The goal of any compression technique is to be able to reproduce the original signal with no perceived loss of quality. In the reality of transcoding, there is a fundamental trade-off between fidelity and bit rate.

In their 2005 paper, [50], Gary J. Sullivan and Thomas Wiegand define four important characteristics for a video codec (i.e. the system comprising of a coder and decoder):

- Throughput of the channel (transmission channel bit rate and protocol overhead),
- Distortion of the coded video (errors introduced by the encoder and transmission),
- Delay (startup latency and end-to-end delay),
- Complexity (in terms of computation, memory consumption, and memory access requirements).

In this section, I will present the history of international video coding standards, the application of coding techniques in low bit rate environments and their error-resilience.

4.3.1 Main bodies

ITU

ITU is the International Telecommunication Union. It was founded in 1865 and its main objective is to recommend standards in the area of telecommunications for interoperability among countries. It is a specialized agency serving under the United Nations. There are several branches under ITU, with ITU-T being the one specifically dealing with telecom issues. ITU-T is relevant for this work because of their involvement in developing video coding standards.

ISO/IEC MPEG

Moving picture experts group (MPEG) is a working group of ISO/IEC. Founded in 1988, MPEG consists of representatives from the industry, universities, and research institutions. Its main task is to define standards for video and audio coding.

History of international video standards

Figure 4-2 depicts a timescale with an overview over when the international video standards were developed. In section 4.3.2, the standards are further detailed.

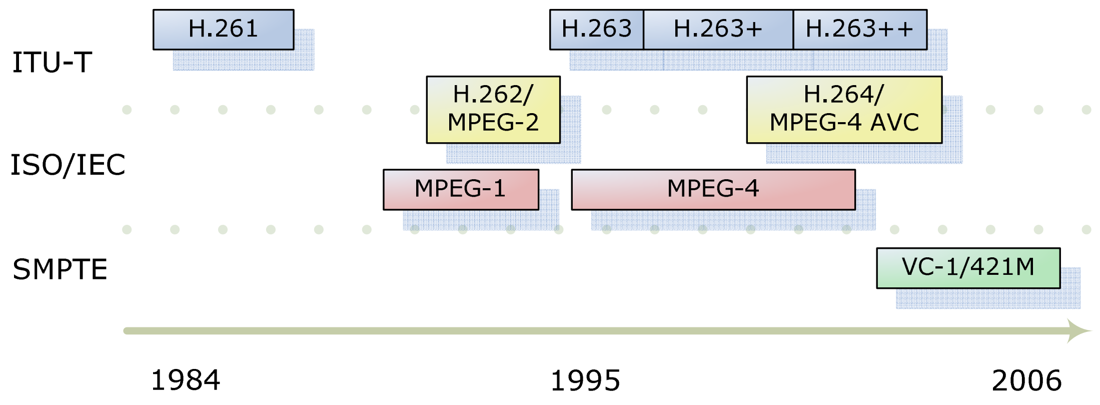


Figure 4-5: A chronology of international video coding standards (from [14], [48], [45]).

4.3.2 Overview of international video standards

MPEG-1

In the early 1990's, when development on MPEG-1 started, the goal was to create a format that would allow storage and retrieval of moving pictures at SIF resolution (352 × 240) at 25 Hz. The target bit rate was 1.15 M bits per second, an effective compression of 25:1 [14].

MPEG-2

MPEG-1 was not targeted at high-definition video [43], and that prompted the development of the successor called MPEG-2. It is also referred to as H.262, since it was jointly developed by ISO/IEC and ITU-T [14]. MPEG-2 supported video at higher resolution than its predecessor and hence higher bit rates. Since its deployment, MPEG-2 has been adopted by a number of different applications, such as digital video broadcasting [3], and DVD [64]. MPEG-2 defines several "profiles" aimed at the different applications. With the profile intended for DVD movies, "main profile at main level" 720 × 576 at 25 Hz, a bit rate of 9.8 M bit/s is generated [19].

H.263

With the growth of the Internet during the 1990's, wireless networks such as GSM and the development of 3rd generation networks, the need for a better compression technique grew.

ITU-T responded by releasing the H.263 standard in 1995. Its main objective was "video coding for low bit rate communication" [21] and applications like video-conferencing. In [41], performance evaluations show that H.263 is more than 20 percent more efficient than MPEG-2. Though considered a legacy coding technique, H.263 is used extensively by popular video distribution web sites, such as YouTube and Google Video.

MPEG-4 visual

In 1998, ISO/IEC introduced MPEG-4, which aimed at applications such as streaming media, digital television, and conversation (e.g. video phones) [10]. MPEG-4 actually consists of several standards, called parts, of which two directly refers to video; part 2 and part 10 (released in 2003 [49]). Part 2 is also called "Visual" and part 10 is called "Advanced video coding" (AVC). In [41], MPEG-4:2 ASP showed to be 40 percent more efficient than MPEG-2.

H.264/AVC

For MPEG-4 part 10, MPEG again joined forces with ITU-T and hence the standard is also called H.264 [62].
Both part 2 and part 10 are comprised of various levels, each targeting a certain resolution and frame rate. In comparison with MPEG-2 encoded video, H.264/AVC has shown to be up to three times as efficient [49], and yielded a 40 percent lower bit rate when compared to MPEG-4:2 ASP in [41].

VC-1

Society of Motion Pictures and Television Engineers, SMPTE, announced in early 2006 the release of standard 421M, also known as VC-1. The Microsoft WMV9 codec is compatible with VC-1 [54] and they are for the rest of this document considered equal. Microsoft has implemented a second codec, called WMV9 advanced profile, which handles high definition content.
The VC-1 codec is very similar to H.264/AVC in construct, and hence also performance [45]. In the paper, Srinivasan compares the two codecs by looking at the peak signal-to-noise ratio. Though very similar, H.264/AVC seems to be a little bit (less than 1 dB) better on higher bit rates (range 2-6 M bits per second).

Motion-JPEG

As this work is focused on a low bit rate application, another coding technique is also presented; called Motion JPEG, though not officially recognized as an international standard. M-JPEG encodes video on a picture-by-picture basis based on the JPEG standard [57]. In the JPEG-2000 standard, part 3 is defined as Motion JPEG-2000, enabling moving pictures. One fundamental difference between the two is that JPEG-2000 employs the discrete wavelet transform (see section 0) and its predecessor JPEG uses discrete cosine transform (see section 0). In an assessment, [38], JPEG-2000 shows a significant PSNR gain of 3-4 dB compared to JPEG, in lossy, low bit encoding.

Dirac

Not an official standard, but interesting as what may come, is the Dirac coding technique. It is currently being developed by the British Broadcasting Company (BBC) and differs from the other codecs by not using DCT. Instead, like JPEG-2000, Dirac employs the discrete wavelet transform. The Dirac codec is targeted towards high definition applications. Low bit rate video is mentioned in [66]:

> *"[The codec] has been further developed to optimize it for Internet streaming resolutions and seems broadly competitive with state of the art video codecs."*

The codec is still in development; however the specification is essentially completed and freely available, so one can assume implementations will emerge over time.

4.3.3 Assessment of perceived quality of service

As this work is focused on distribution of video over low bit rate links, it will involve selecting the most suitable video compression technique. The term "most suitable" includes several things, one of them being perceived quality of service. The process of assessing the quality of a video stream is non-trivial, and there have been numerous studies on the topic [52], [60], [9].

The assessment metrics can be divided into two main categories; objective and subjective. In objective assessments, the most recognized method is measuring the peak signal-to-noise ratio. In subjective testing, ITU-T has recommended to use a "mean opinion score" metric [20]. Both methods are discussed below.

Peak signal-to-noise ratio

In electrical engineering, one way to determine the quality of a signal is to look at the signal-to-noise ratio. The signal is the information, and noise is the undesired element in the transmission medium. It is conveniently expressed in a logarithmic decibel scale. In image processing, a special case of SNR has been adopted, the peak signal-to-noise ratio. Here, noise has come to mean artifacts in the image introduced by compression. Sometimes is it called peak signal-to-reconstructed image ratio. The PSNR method quantifies the *luminance distortion* in a compressed image, by comparing it to the uncompressed image.

The way it works is as follows: Two monochrome $m \times n$ images, image A (before compression) and image B (after compression), are compared by looking at the luminance value of the individual pixels. The *mean square error* is calculated as the following:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left\| A[i, j] - B[i, j] \right\|^2$$

Then the PSNR is calculated as:

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE} \right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}} \right),$$

where MAX_I is the maximum pixel value of the image. E.g., for an 8 bits-per-pixel image it would be 255.

Typical reasonable values for PSNR are 20-40 dB. To give the reader an example of different PSNR values, Figure 4-6 and Figure 4-7 show a DCT-encoded (JPEG) image and the resulting PSNR calculations.

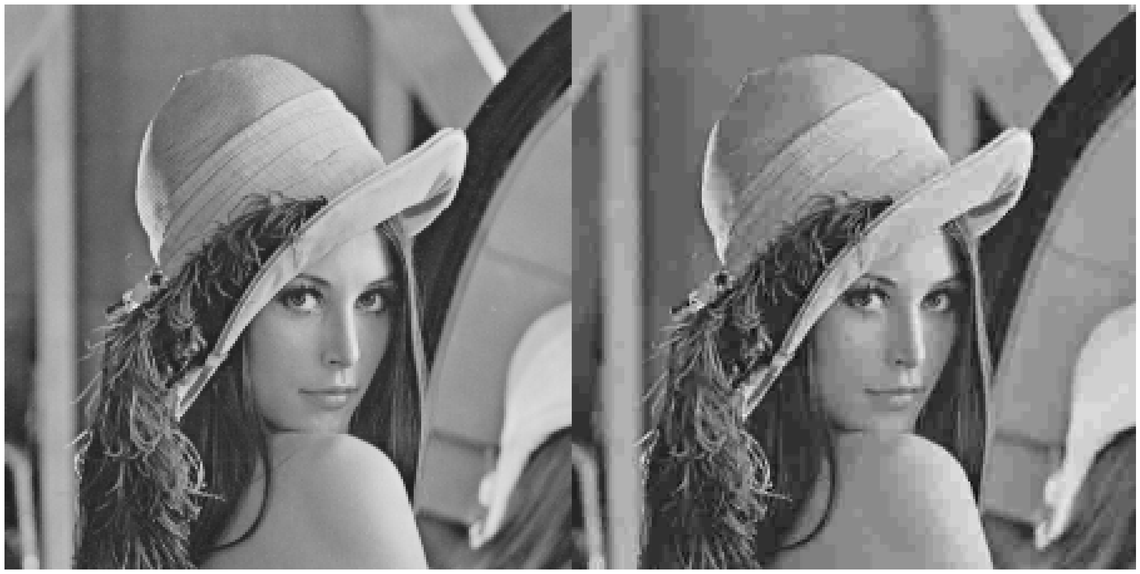


Figure 4-6: "Lena"; PSNR 33.4 dB. Original image to the left.

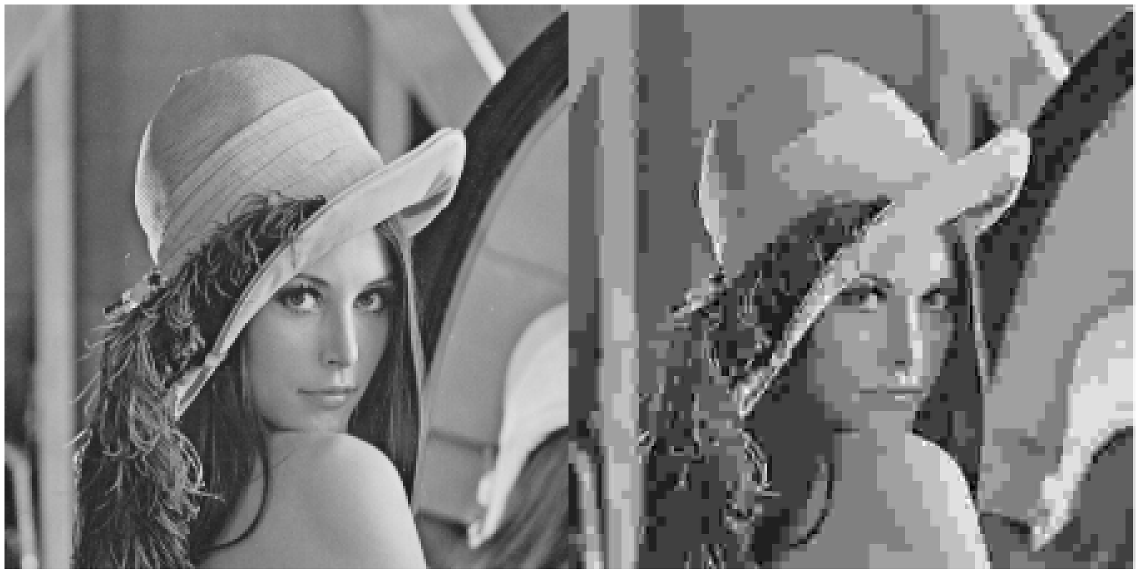


Figure 4-7: "Lena"; PSNR 27.2 dB. Original image to the left.

ITU-T mean opinion score

In 1999, the standardization branch of ITU released a recommendation for a "non-interactive subjective assessment method for evaluating the quality of digital video images" [20]. It is a straight-forward approach where a large enough audience is gathered and subjectively rates different video clips on a 1-5 scale, where 5 is the best rating. The mean of the scores for each video clip is calculated and can then be used to evaluate different encoding techniques.

Other methods

There have been efforts to devise new assessment methods for determining video quality. In [59], Zhou Wang et al. describe a method that adds two factors to the PSNR method; loss of correlation and contrast distortion.
In their 2003 paper [58], Zhiheng Wang et al. discuss commonly used metrics such as PSNR and mean opinion score, but also introduce a set of alternative objective streaming video metrics and present results based on experiments.

4.3.4 Low bit rate video

Three of the coding techniques were designed with low bit rate applications specifically in mind; H.263, H.264/AVC, and VC-1. For the latter two, high-definition video with high bandwidth is also a big objective, but it is not relevant for this work.
In the case of VC-1, there is a tool developed specifically for low bit rate applications [44]. It is the ability to encode a frame at multiple resolutions, by scaling down either or both dimensions. The decoder is informed that the frame has been down-scaled, and up-scales the image before displaying it. In this way, the range of quantization is extended by a factor of $\sqrt{2}$ each time the image is down-scaled by a factor 2.

Video in wireless environments makes not only coding-efficiency interesting. Also, error-resilience, end-to-end delay, and jitter are relevant and will continue to be, since in 2003, Stockhammer et al. stated in [47]:

> *"[…] it is worth noting at this point that new directions in the design of wireless systems do not necessarily attempt to minimize the error rates in the system, but to maximize the throughput."*

In the paper [7], the writers evaluate the performance of H.264/AVC in 802.11b ad-hoc wireless networks. They present solutions for the random packet-loss problem and how to recover from bursts-errors. Another paper [61] discusses the error-resilience and presents tools included in the H.264/AVC standard to mitigate the problems.

4.4 Network architecture

The cameras intended to be used in this system come with a built-in network interface and have built-in video servers.
From that perspective, there are two solutions for the communication between the source (camera) and the destination (viewer). The first is using direct connection and the second employs an intermediate relay agent. The two are discussed further below.

4.4.1 Direct connection

Since the cameras are assigned an IP address like any other node and have a built-in video server, the easiest way of communicating is a direct connection between the viewer and the camera. When a viewer wants to view a particular camera feed, it just opens a connection to that camera. When the viewer decides to stop viewing, it just closes the connection.
The main drawback with this scheme is for applications where the camera is on a low bit rate link. If several viewers connect to the same camera, the same stream is sent multiple times over the link. Another drawback using this method is that each client viewer needs to know how to communicate with each camera.

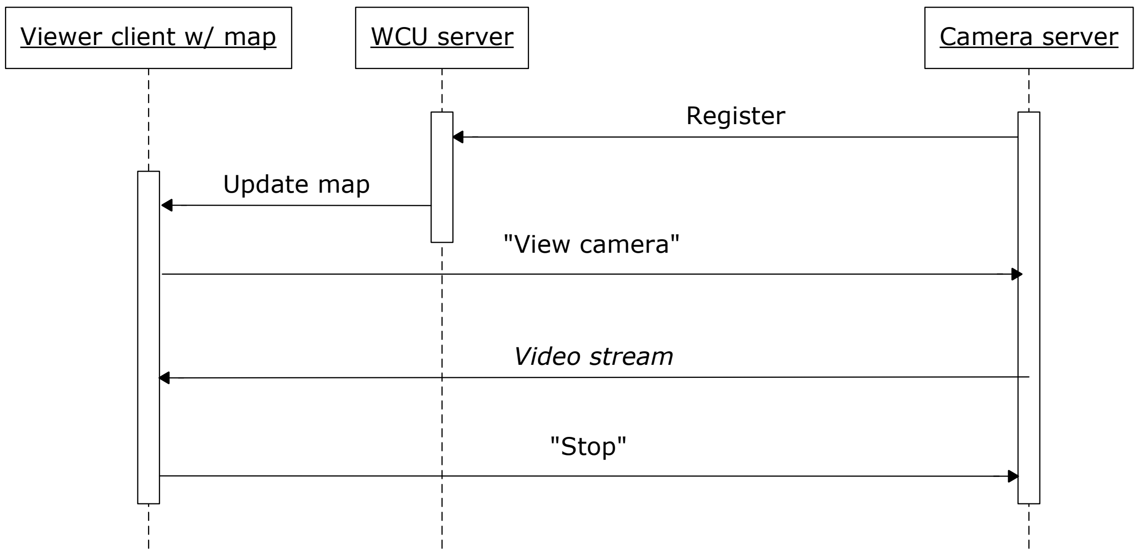


Figure 4-8: A diagram showing the sequence in which to start viewing a video from a camera.

4.4.2 Connection via an intermediate relay agent

To encounter the problem with many viewers connecting to the same camera over a low bit rate link, an intermediate relay agent could be used.

By having the viewer connect to the relay requesting a particular stream, the relay would then connect to the camera and initiate the stream. If another viewer wants to view the same stream, the relay would just clone the stream and not increase the load on the link between the camera and the relay agent.

The main advantage of using this setup is of course that the link between a camera and the relay agent only carries one stream at a time. Another advantage is that it is only the relay agent that needs to know how to communicate with each camera. The relay agent may then translate the stream into some common form that all client viewers understand.

The real drawback is that there will be one single point of failure. If the intermediate relay agent fails, no video will be able to be distributed. There are ways to mediate this fact, such as using redundant relay agents. Another drawback is that the system will be more complex in its design.

A sequence diagram showing the necessary steps are shown in Figure 4-9.

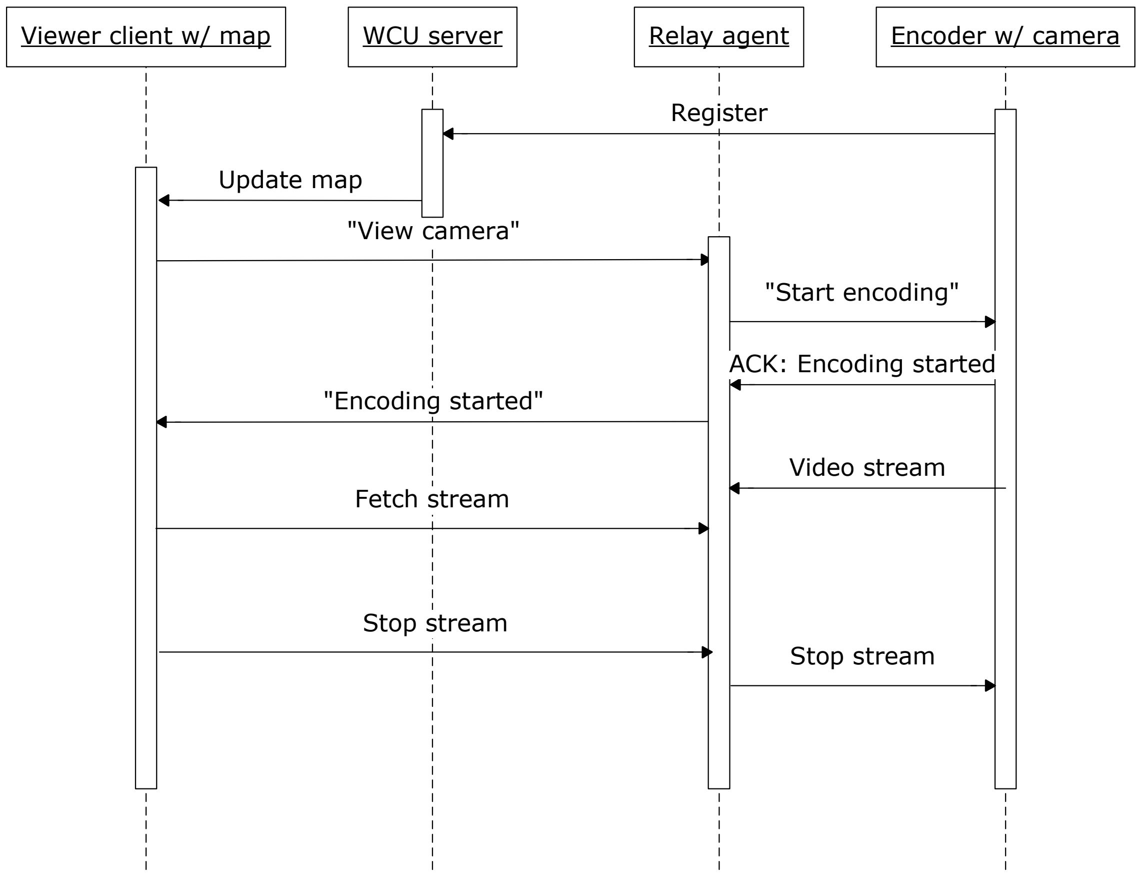


Figure 4-9: A sequence diagram showing the steps taken to view video through a relay.

When this work refers to the network design, it generally refers to the transport layer and above. It assumes that there is an IP network below and the addressing is in place. There is extensive research on different transport and application overlay network designs, and this work will not extend that. On the application layer, there are a few interesting techniques that this work will evaluate; they are accounted for below.

4.4.3 Hybrid architecture

A hybrid network architecture could be to have the first client connect to the camera, and any subsequent viewer connect to this client. Due to the significant latency problem when using a relay, this option was not further investigated.

4.5 Existing solutions

There has been a pre-study titled "Video för WCU" [56], performed by Ph.D. student Christoffer Wahlgren. He bought a ready-made system from a commercial vendor and successfully integrated it on the WCU client. However, for this work, that system was too expensive and thus is not viable.

4.5.1 Microsoft Media Services and Windows Media Encoder

The WCU clients and infrastructure are built in a Microsoft Windows environment. Therefore, it is natural to investigate the Microsoft product family.

The "raw" digital video signal needs to be compressed before being transferred on the network. The Microsoft Windows Media Encoder (WME) is designed to perform such a task. It is a stand-alone program, and comes with an SDK that allows it to be seamlessly integrated into other applications.
The WME can pass its data in two ways; pull or push. When the data is pulled, the server (or another client) initiates the connection. This is a simple setup, but can place a lot of burden on the encoder in case there are multiple incoming connections. Another drawback is the fact that in the WCU environment, the field clients use connect over GPRS from an ISP. The ISP uses NAT and only assigns IP addresses in the private address range, and does not allow port-forwarding. Hence, it is not possible to access a field client externally.
By using the push method, the encoder initiates a connection to the server, and pushes the data upstream. Anyone that wants to access the data connects to the server, which acts as a relay. In this way, the NAT problem is avoided, and a server is more capable of handling multiple external clients than a client.
It is also in line with the overall design of the WCU concept.

Microsoft Windows Server 2003 has a built-in module for streaming media, called Windows Media Services (WMS). A, for this work, relevant aspect of the WMS is its capability to act as a relay for streaming video. As described above, a Windows Media Encoder can push its data to the server, which then can re-distribute it to multiple clients.
Windows Server 2003 is already used in the WCU infrastructure, so it requires relatively little effort to add the media streaming role.

4.5.2 Videolan VLC/VLS

This piece of software originates from a university project at the École Central Paris. In the beginning, two different versions were developed; one server module (VLS) and a client module (VLC). These were then merged into one (VLC).
VLC version 0.86 comes with support for a large array of platforms and video formats. It also ships with an ActiveX control library.
VLC is licensed under the GPL, which makes it difficult to integrate with non-GPL projects. However, integrating the ActiveX control is permitted, since it would constitute as two different programs. The ActiveX library is loaded into a separate address space and all method calls are interop.

4.5.3 SerVision Gateway family

SerVision is a company dedicated to the security industry, with products aiming at video surveillance.

SerVision has a product line called SVG, which is basically a video server that encodes analog video and streams it over the network. It is capable event-based automatic video capture and storing, and supports multiple source cameras. The server is mainly designed for streaming over high bit rate links, but it is also capable of low bit rates. All their products use MPEG-4:2 compression.

4.5.4 Bosch Security Systems IP network video

Bosch Security Systems is focused on surveillance products, intrusion alarm systems, and PA systems.
Bosch has several solutions in the area of transmitting video over networks. Most of the products are geared toward higher bit rates and higher fidelity, but there is some support for low bit rate applications.

Bosch uses both a distributed server system (X-range), comprised of a small video server attached to each camera, and a centralized system (Vidos), with capabilities such as storing of video, image interpretation, and remote controlling.

4.5.5 Test pilots

In order to gain a deeper understanding of the off-the-shelf products, a series of pilot tests were conducted. This section gives a summary of the findings with regards to the latency of the video signal

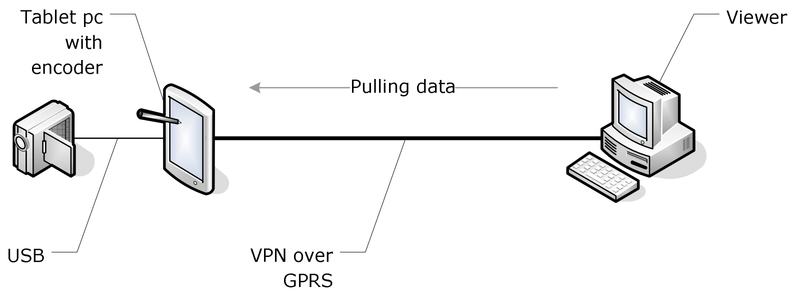


Figure 4-10: Test setup 1.

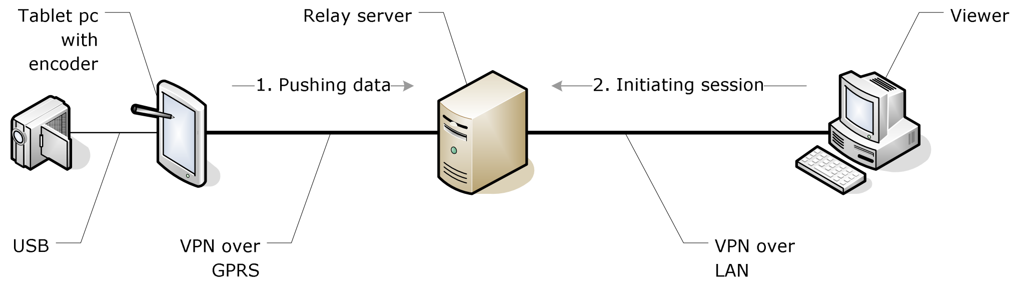


Figure 4-11: Test setup 2.

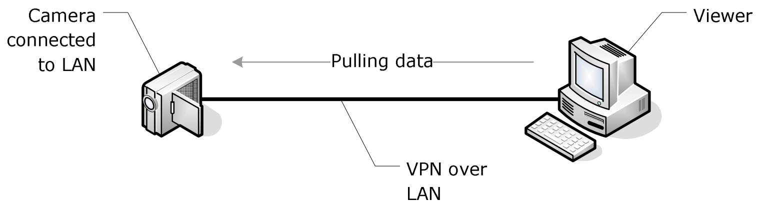


Figure 4-12: Test setup 3.

.

Windows Media Encoder/Services

The software family for live video distribution by Microsoft was tested in two different ways.

First, there was a simple setup consisting of a camera connected via USB to a tablet pc. The tablet pc was running Windows Media Encoder in "stand-alone" mode, i.e. acting as a server for incoming connections. The tablet pc was connected via a GPRS link. Since the ISP is using NAT with no access to the port forwarding, a VPN was installed to allow for access to the server. Please refer to Figure 4-10.

Windows Media Encoder was setup to encode 160×120 pixel video in the WMV9 format in 32 kbits per second with no audio. The viewer was running Windows Media Player and connected to the same VPN.

In the other setup, the only difference was that instead of acting as a server, the Media Encoder pushed the data over the GPRS link to a Windows Media Server on the same VPN. Please see Figure 4-11.

In the former scenario, the signal was delayed about 25 seconds initially. After reducing the buffer sizes as much as possible on both the sender and the receiver, the latency was reduced to around 18 seconds.

In the other scenario, using the Media Server as a relay, the latency was never under 30 seconds. All buffers, the encoder, server and client, were reduced as much as possible.

In the test setup, when using ping to assess the round-trip latency, the typical value was around two seconds.

Another thing noticed was that the CPU load was constant at around 40 percent during the encoding.

Videolan VLC

The setup in this test was similar to what was used with the Microsoft software.

Two scenarios were constructed. In the first, the camera was connected via USB to the tablet pc, which was connected on a VPN over GPRS. VLC acted as an encoder and a server, as in Figure 4-10. VLC encoded the video in 160×120 pixels WMV9 at 32 kbits per second.

In the second scenario, the setup was as in Figure 4-11, with an instance of VLC running as an encoder and one as a relay server. Also, VLC was used on the viewer side.

In the first scenario, the latency was typically 4-5 seconds. In the second setup, the latency was increased to about 10 seconds.

Using built-in video server

The camera used in this test, the Axis 207, had a built-in server and DHCP client. The viewer in this case was an ActiveX control plug-in for Internet Explorer. In this setup, there was no noticeably delay. One should note that a fixed-line LAN was used in this setup; hence no significant delay from the network was added.

The other cameras procured for this work (see section 5.5), are similar to the Axis 207 in the sense that they also have a built-in video server.

4.6 Conclusions

4.6.1 Summary

	Windows Media Server	Windows Media Encoder	VLC with relay	VLC direct	Camera SDK
Latency	High	Mid	Mid	Low	Low
Lost packets	Low	Mid	Low	Mid	High
Adaptive bitrate	x	-	-	-	-
RTSP support	x	x	x	x	-
Scalability	High	Low	High	Low	Low
General appl.	Low	Low	High	High	High

Table 1: An product-feature matrix. Green fields symbolize a desired feature, yellow means indifferent and red marks negative impact.

The above table illustrates a comparison of the features of the different techniques in the solution space. It is very simplified, and should be used as a brief overview. "Camera SDK" means that the software shipped with each camera is used for the connection.

"Latency" means the delay of the video signal from the source to the viewer. "Low" indicates a latency of less than 2 seconds, "mid" is less than 10 seconds, and "high" is everything above.

"Lost packets" refers to different buffering/error recovery facilities in the software. Usually, this contrasts the latency since buffering inherently increases latency.
"Scalability" indicates how well the system may handle an increased number of clients. For example, a direct connection to an IP camera is limited by the built-in video server. On the other hand, Windows Media Server may be run on more sophisticated hardware and therefore handle more clients.
"General applicability" refers to how well the system works with different cameras. Windows Media Services/Encoder can only stream video via the DirectShow interface, hence has no support for IP camera. VLC can connect to IP cameras provided they support RTSP.

4.6.2 Coding technique

For encoding low bit rate video today, MPEG-4:2 is considered the best option.
The reasons supporting the recommendation:

- It is a standard developed by one of the major expert organizations in the field, thus it can be assumed there is wide-spread support in software and hardware.
- Despite being only a few years old, it has to be regarded as a mature technology, also reinstating its near-ubiquitous position.
- The compression properties are regarded as among the best at the moment.
- Its error-resilient properties are good.

The H.264/AVC is an even better choice bit rate wise, and seems to be the best option for the future, but it has not yet reached widespread adoption like MPEG-4:2.
For very low bit rate environments, when frame rate has to be sacrificed for fidelity, motion-JPEG may be a suitable candidate.
Some aspects of MPEG-4:2 and H.264/AVC that may be regarded as not favorable:

- Even though Microsoft has developed WMV9 based on MPEG-4, there is no support for standard MPEG-4 in Windows XP.
- There are somewhat difficult licensing issues around MPEG-4 decoders.

In the specific case of implementation in a Microsoft software environment and as a recommendation for the future, the VC-1 codec is the best option. Its efficiency is much better than MPEG-4:2, which makes it ideal for low bit rate applications. There is a codec available from Microsoft, which makes support virtually ubiquitous within Microsoft products like Media Player and Media Encoder.

4.6.3 Network design and protocols

With the given problem specification, and with the WCU concept in mind, there are a few viable choices when it comes to network design.
First is the direct connection between viewer and camera. This is the by design simplest design and most fault-tolerant. It does not scale very well, since it depends on the cameras and their links how many concurrent viewers they allow.

A second alternative is to use an intermediate translation proxy tier, using existing solutions. However, with the results of the test pilots in mind, using this scheme adds an unacceptable latency to the signal and it is not feasible for this project.

A third alternative is to design and construct a specialized relay proxy for the WCU. This topic needs to be researched more and probably requires working with the Microsoft DirectShow library. Therefore, it is outside the scope of this project.
A solution based on an intermediate tier, as in alternative two and three, also needs to take redundancy into account.

When it comes to application and transport protocols, all the selected cameras support HTTP and two of them RTSP over RTP. The latter would be the recommendation for this project if it was supported by all, solely because it does not need a permanent TCP connection, as HTTP traffic over TCP does. On unreliable wireless links, the connection-less property is important.

The table below tries to give an overview of the basic pros and cons of a few solutions that are viable for this work.

Solution	**Pro**	**Con**
Using a custom relay server	Constructing a dedicated server that acts as a router and just fetches the packets from the cameras and allow several clients is probably the best option, for many reasons; scalability,	The time constraint for this project makes it impossible.

	redundancy, not saturating the camera network link.	
Using Windows Media Server	Fine buffering capabilities, has adaptive bit rate.	Latency is way too high.
Using VLC as server	Lower latency than Windows Media Server and has good support for IETF standards.	Latency is still too high, requires RTSP support in the camera application layer.
Using cameras own SDK	Makes it possible to utilize camera-specific features, provides minimal latency.	Not truly generalized. An adapter for each camera is required unless they support RTSP or similar. As this solution implies a point-to-point connection, it may saturate the camera network link when several viewers are connected.

5 Design and implementation

5.1 Overview of this chapter

This chapter presents a proposed design for the implementation of video distribution, within the existing WCU platform.
This chapter is in many ways based on the literature study performed in an earlier phase of this project. The decisions are also made with the overall project goals in mind, namely to create a generalized and cost-efficient solution.
The design will be implemented in the WCU 2.0 production environment currently in place at Saab Security Systems AB. The environment is based on the Microsoft Visual Studio 2005 Team Foundation.

5.2 Target platform

The design is based on the WCU 2.0 design. As the WCU 2.0 is currently in implementation phase, and some design decisions are revised, it is expected that this design will change accordingly.
The WCU 2.0 client is built for Microsoft Windows XP. It is built on top of the .Net framework, version 3.0. The product is intended to run on several different hardware configurations, ranging from low-capacity tablet PCs to high-capacity servers.
The implementation of a video component is supposed to be fully integrated with the existing solution.

5.3 Use cases

5.3.1 Registering a video sensor in the server

Pre condition
Camera is powered on and fully connected to the network.

Basic path

1. Add the sensor to database in the WCU server.

Post condition

The video sensor is added to the database.

5.3.2 Start viewing video

Pre condition

Video player plug-in is loaded; a video sensor is active and visible on the map.

Basic path

1. The user clicks on the desired video sensor symbol on the map.

Post condition

The video player plug-in starts to show the video stream from the sensor.

5.3.3 Stop viewing video

Pre condition

Video player plug-in is loaded and showing a video stream from a sensor.

Basic path

1. The user clicks on "Stop" button.

Alternative path

1. The user closes the video plug-in.

Post condition

The video player plug-in stops showing the video stream.

5.4 Architecture

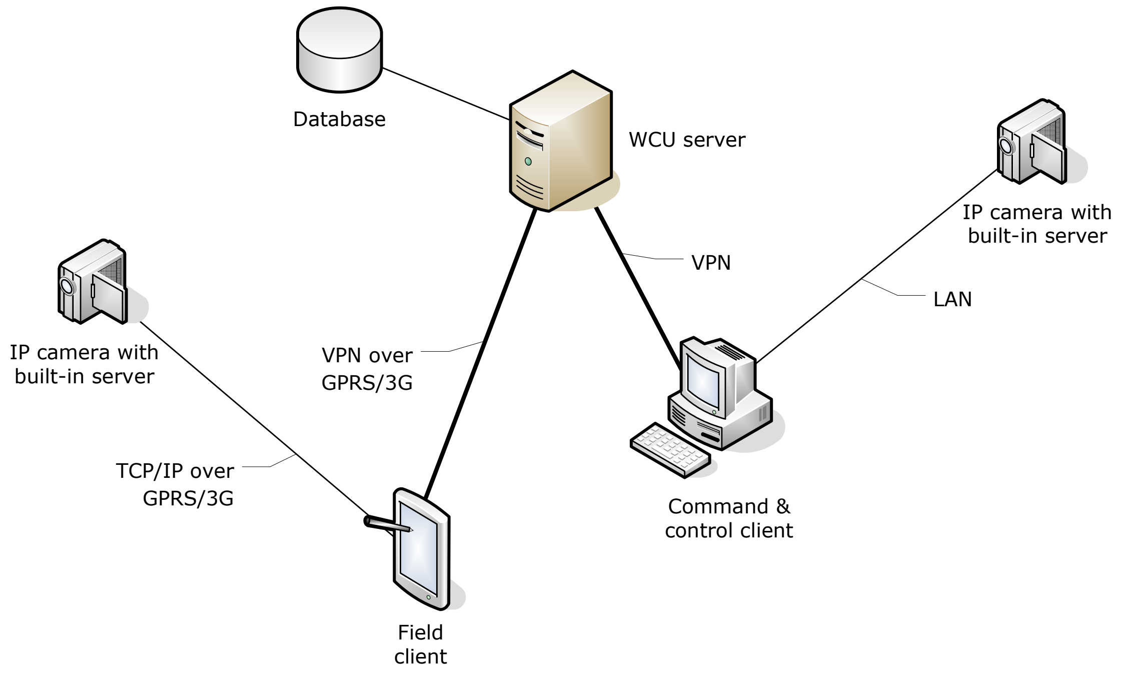
5.4.1 Strategies

I have looked at several different ways of solving the problem, each with its strengths and weaknesses.
To make the solution truly generalized, perhaps the best way would be to use an intermediate translation layer. Then, the clients only need to handle one type of data stream, and changes or updates only need to be done at one place, the translation tier. A prototype was constructed to test this concept, using two commercial video products. The first was using Microsoft Windows Media Services and the second was using the Videolan VLC software. However, one major drawback was found using this solution. A considerable latency was built into the system; in the case with the Microsoft product, around 30 seconds. With the Videolan software, it was not as hefty, but still too much at around five seconds. The reason for this is that those systems are more geared towards video quality, rather than live video. Thus, a lot of buffering is used, particularly in the case with Microsoft's product.
A second obvious drawback is that the reliability of the system depends heavily on the translation node. Of course, redundancy could be added but it is not viable for this work.

Another solution is to build a specialized translation layer that would have the properties needed in this work. This may very well be a recommendation for the future, but there is not enough time to fit such a task in this work. One of the reasons for that is that it would most likely involve the Microsoft DirectShow library, which is not ported to the .Net framework yet. A drawback with this scheme is that there will be a single point of failure, since if the translation layer fails; no video at all will be distributed.

The third alternative, also the proposed design, is to place the intelligence at the leaf nodes; the clients. In this solution, there will be a direct connection between the client viewer and the video source. In this case, the client will need to be able to handle the different streams from the various cameras. If there is a new camera connected, the client needs to learn how to decode the video stream. A prototype using this technique was constructed, and it shows that the latency is kept very low. This solution would be a more distributed system, not always depending on one relay node.

However, with this solution, there are two obvious drawbacks. First, the more clients that are viewing a particular stream, the higher the bandwidth is needed at the camera. Second, if a new camera is connected, every client that wants to view a stream from it will need to learn how to decode the stream. Also, distributing the video servers is not as scalable as using one intermediate relay, which can rather easily be upgraded to accommodate more traffic.



Above figure shows an overview of the proposed solution. The clients (field and C2) are connected to the WCU server over VPN. The server knows about the camera sensors; i.e. has a record of them in the database, so the clients will see the cameras on the map. The record in the database also contains the address and model of the camera, so the viewer can connect.
Unfortunately, none of cameras support VPN.

5.4.2 Dependencies

One dependency worth mentioning is the network. Since the network connections are acquired from an external ISP, the solution depends on their network conditions.
The cameras selected for this work are inherently different, since a goal for the work was the ability to function in a general way with different cameras. However, this also implies that

different cameras may have to be accessed in different ways. Thus, this solution is dependent on the cameras attached, and their application interface.

5.4.3 General constraints

The field clients are connected to the server over GPRS/EDGE or UMTS wireless data links. Typically, they both provide around 60 kbps uplink bandwidth (see 2.3). There is also more latency over these links compared to fixed network, in the order of at least one magnitude.

The ISP supplying the network connections uses a network address translator in their network, assigning private addresses to all data clients. This introduces the classic NAT problem; that a client is not directly accessible from the server.

There are several solutions to this problem; an elaborate scheme is to use so-called UDP hole-punching. Another way is to put in place an overlay network, such as a VPN. This solution is currently being investigated in the WCU 2.0 project, and it would allow direct access from server to client.

A third alternative is to have the client push its video data to the server. This requires the client to continuously stream data, even though no other client is viewing. If no VPN solution is in place, this is the preferred solution.

Otherwise, a VPN would solve that problem.

Encoding video is a really resource-demanding task; both regarding network and processor utilization. On the field client hardware, empirical tests have shown that encoding video at VGA resolution in the H.264 format requires around 95 percent of the available CPU time.

Memory consumption is not as severe as the CPU utilization. The empirical tests show that H.264 encoding in resolution of 1280×720 pixels, one magnitude of compression, peak consumption is around 50 MB of RAM, well below what is available.

Power consumption is disregarded for this work.

5.4.4 Program units

There are two fundamental parts of this solution; one is the video plug-in, the other is the changes to the pre-existing data model and virtual earth.

5.5 IP network cameras

This sections looks at different cameras on the market with a short review of each. They are all similar if one looks at the specification. They are all capable of sending MPEG-4:2 video in various resolutions/frame rates. All of them except the Axis also come with adaptors for wireless networks. One of them, the Linksys unit, is equipped with pan-tilt-zoom controls.

	Axis 207	**Linksys WVC200**	**D-Link DCS-2120**	**Vivotek IP7135**
MPEG 4:2	x	x	x	x
10/100 Mb eth	x	x	x	x
WLAN		x	x	
PTZ		x		
SDK	x			x
Max res	VGA	VGA	VGA	VGA
Price SEK	2396	1992	1995	1755

Table 2: Overview of network camera features.

Vivotek IP7135

Main features:

- MPEG-4:2, JPEG
- Max res: 640 × 480 pixels @ 30 Hz
- 10/100 Mbps Ethernet network adaptor, 802.11g
- Price: SEK 1755 excl VAT

The Vivotek camera ships with an ActiveX control intended to be embedded in a Windows application or the Internet Explorer browser. It also comes with a quite impressive SDK that allows access to all the features of the camera, as well as a library for storage of video.

Linksys WVC200

Main features:

- MPEG-4:2, JPEG

- Max res: 640 × 480 pixels @ 30 Hz
- 10/100 Mbps Ethernet network adaptor, 802.11g
- Pan-tilt-zoom control.
- Price: SEK 1992 excl VAT

The Linksys camera ships with software for viewing the camera but no software for explicitly embedding the viewer in an application. However, the ActiveX control may be added to a new application but contains no documentation of the features. Reverse-engineering of the control revealed the most basic features such as viewing and controlling its PTZ features.

D-Link DCS-2120

Main features:

- MPEG-4:2
- Max res: 640 × 480 pixels @ 30 Hz
- 10/100 Mbps Ethernet network adaptor, 802.11g
- Price: SEK 1995 excl VAT

The D-Link camera, just as the Linksys, only ships with ready-made software for viewing. It comes with an ActiveX control, but there is no support for embedding the viewer in other applications. An attempt at reverse-engineering the control failed, and the only way I found it to work was to access the RTSP stream from an instance of the VLC application.
One thing worth to notice is that it seems like the D-Link camera is actually a Vivotek camera, manufactured as OEM. In that case, it seems like it would be at least possible to distribute an SDK.

Axis 207

Main features:

- MPEG-4:2, M-JPEG, JPEG
- Max res: 640 × 480 pixels @ 30 Hz
- 10/100 Mbps Ethernet network adaptor
- Price: SEK 2396 excl VAT

Axis provides an ActiveX control intended to be embedded in other applications. They also provide a rich SDK with very good documentation.

5.6 Program unit: Video plug-in

5.6.1 Introduction

The viewer may be regarded as the end-point of the communication. Its main task is to display a given video stream.

5.6.2 Interface descriptions

IVideoPlayer

This interface is supposed to make it easy to replace the actual viewer in the system. It declares two methods that any implementer must define:

- `PlayUrl(string)`: Start fetching the stream from the given URL and display it.
- `Stop()`: Stop fetching the stream.

With the methods declared in this interface, the use cases in sections 5.3.2 and 5.3.3 can be accomplished; by supplying the URL of the video stream when invoking the `PlayUrl()` or `Stop()` methods.

5.6.3 Class descriptions

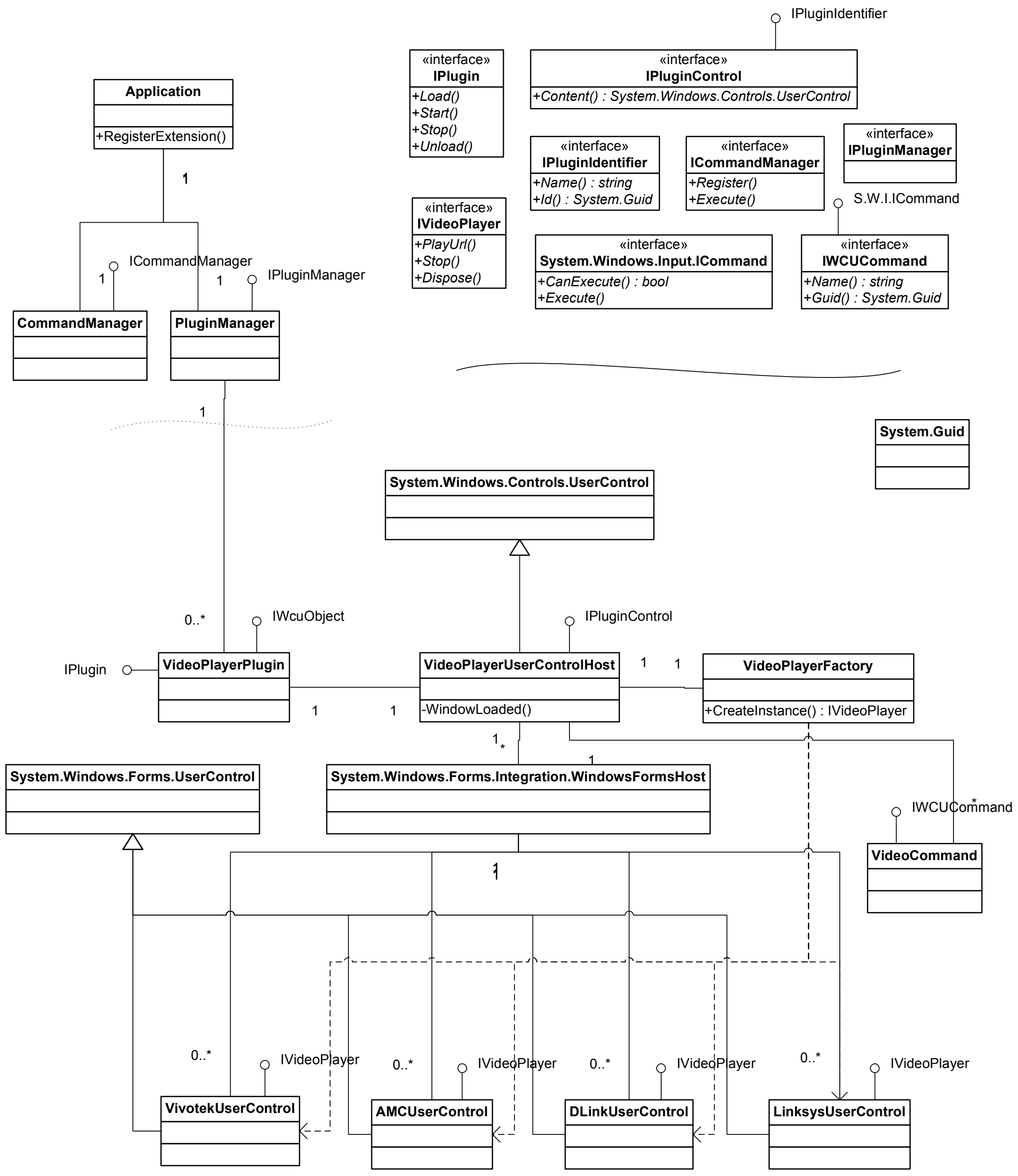


Figure 5-1: Class overview of proposed design.

Class VideoPlayerPlugin

Implements: `IWcuObject`, `IPlugin`

- A reference to an instance of `VideoPlayerPlugin` class is held by the plug-in manager. Through the `IPlugin` interface, four important methods are defined: `Load`, `Start`, `Stop`, and `Unload`.

In `Load()`, the plug-in registers itself with the plug-in manager by invoking `RegisterExtension()` on the `Application` instance.

This class holds two command delegates that are registered with the `CommandManager`; `VideoPlayerPlayUrlCmd` and `VideoPlayerStopCmd`. The idea is that any other plug-in may interact with the `IVideoPlayer` instance by using these two. The `IVideoPlayer` instance is manipulated by using the property `PlayerUserControl` in the `VideoPlayerUserControlHost` instance.

Class VideoPlayerUserControlHost

Inherits: `System.Windows.Controls.UserControl`

Implements: `IPluginControl`

In order to use an ActiveX control within a WPF user interface, it has to be hosted by a class called `WindowsFormsHost`. It resides in the `WindowsFormsIntegration` library, which is part of the .Net platform.

This class is also where the XAML component of the UI is connected to the program logic. When an instance of this class is loaded by the GUI, a method called `WindowLoaded()` is invoked. This signals that the GUI has loaded the plug-in and that it is ready to use. This is also the place where the `WindowsFormsHost` gets initialized.

Through the interface `IPluginControl`, another interface `IPluginIdentifier`, is also implemented. It declares two methods, both having to do with plug-in identification, so the plug-in manager can tell plug-ins apart.

This class is rather passive; it exposes a property, for setting and getting a reference to an instance of its `IVideoPlayer`.

The idea is that when the `PlayUrl` command in the `VideoPlayerPlugin` object is invoked, a new `IVideoPlayer` is created through the `VideoPlayerFactory`. A reference to that `IVideoPlayer` object is then assigned to this property, and the `IVideoPlayer` is manipulated through this reference.

Class VideoPlayerFactory

This class is intended to work like the Factory pattern. It contains a static method `CreateInstance` that takes an argument of the type `EVideoSensor enum` and returns the corresponding `IVideoPlayer`.

Class AmcUserControl

Inherits: `System.Windows.Forms.UserControl`
Implements: `IVideoPlayer`
This class is a specialization of a Windows Forms `UserControl`. It holds an ActiveX control that enables connecting and playback of the Axis 207 camera video data stream.

Class DLinkUserControl

Inherits: `System.Windows.Forms.UserControl`
Implements: `IVideoPlayer`
This class is a specialization of a Windows Forms `UserControl`. It holds an ActiveX control that enables connecting and playback of the D-Link DCS-2120 camera video data stream.

Class LinksysUserControl

Inherits: `System.Windows.Forms.UserControl`
Implements: `IVideoPlayer`
This class is a specialization of a Windows Forms `UserControl`. It holds an ActiveX control that enables connecting and playback of the Linksys WVC200 camera video data stream.

Class VivotekUserControl

Inherits: `System.Windows.Forms.UserControl`
Implements: `IVideoPlayer`
This class is a specialization of a Windows Forms `UserControl`. It holds an ActiveX control that enables connecting and playback of the Vivotek IP7135 camera video data stream.

5.6.4 Necessary changes to VirtualEarthPlugin

VirtualEarthPlugin

The event handler method `OnMapObjectClicked`, listening for mouse click events, is needed.

It should be defined in this class, because it needs a reference to the `Application CommandManager` property, where it can access and execute commands on the `VideoPlayerPlugin`.
When the `OnMapObjectClicked` event handler is invoked, a reference to the corresponding `IWcuObject` is passed as a parameter. This object should be of type `VideoSensorMapObject`, and contains the type of camera and its address. This information is passed as a parameter with the `Application.CommandManager.ExecuteCommand` call. The `ExecuteCommand` method takes two arguments; the name of the specific command (`string`) to invoke and a parameter (`object`).

VirtualEarthUserControl

This class needs to add the `VirtualEarthPlugin.OnMapObjectClicked` delegate as an event handler on its `VEUserControl` reference.

VEUserControl

When the `onMouseClicked` callback is invoked, a string object is passed from the map. It holds the id of the object clicked, and since this class keeps a dictionary of all current map objects, it can perform a look-up to find the corresponding `IWcuObject`. It then fires an `OnMapObjectClicked` event and passes the `IWcuObject` reference as a parameter.

VirtualEarth.htm

Objects in the Virtual Earth map are represented as `VEPushPin` objects. However, a drawback with the Virtual Earth API is that the `VEPushPin` objects are not clickable. The only object that can listen for mouse click events is the map itself, which is not very useful. However, there is a workaround that can be used. When a `VEPushPin` has been added to the map, it is part of the `document` tree structure, and can then have mouse event handlers registered with it. The drawback of that method is that any metadata is not easily retrieved, such as the `VEPushPins` id. To solve this, I decided to use the `onMouseOver` event in an unconventional way. The handler for the `onMouseOver` event just sets a member variable to point at the last object the mouse hovered above. When a `VEPushPin` is clicked, this variable contains the clicked object and metadata may easily be retrieved.
The `onMouseClicked` event handler invokes the `OnMouseClicked` method on the parent, in this case the `VEUserControl` class.
Methods (event handlers) added: `onMouseClicked()` and `onMouseOverCallback()`.

5.6.5 User interface

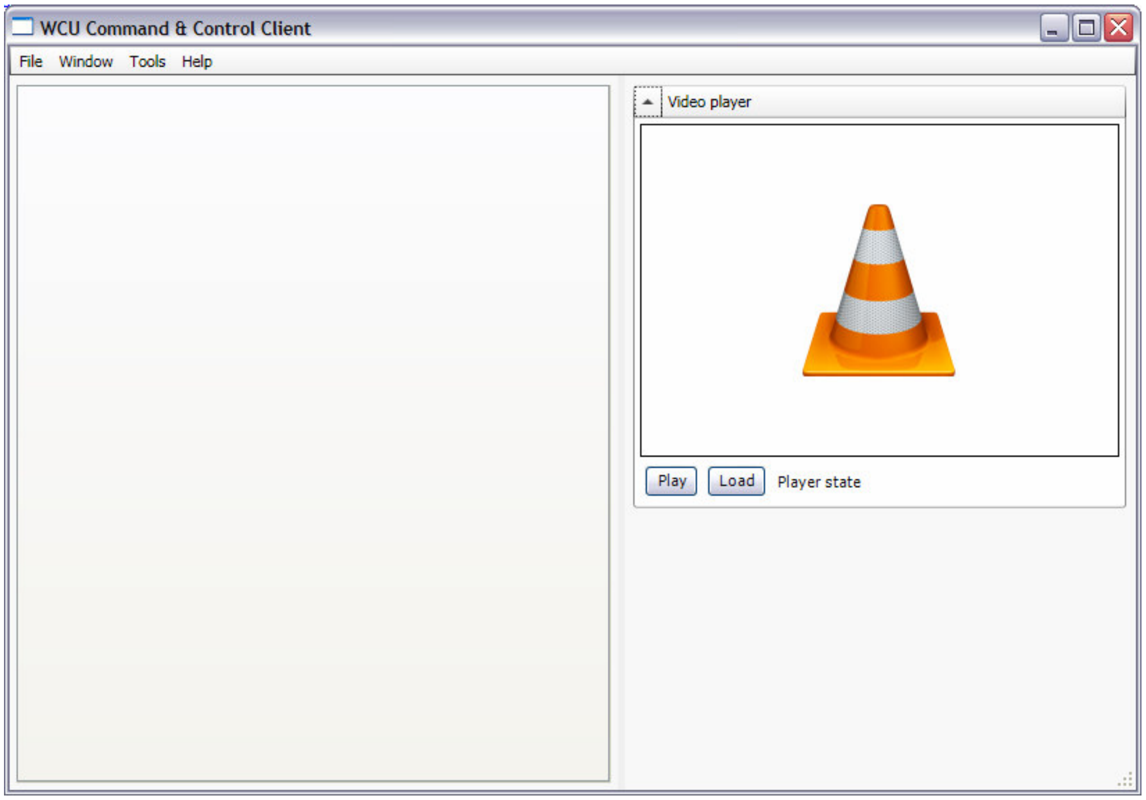


Figure 5-2: A sketch of the user interface for the viewer plug-in.

5.6.6 Necessary changes to the existing data model

In the existing WCU 2.0 data model, there is no object representing a video sensor yet. Therefore, the class `VideoSensorObject` was created.

Class description VideoSensorMapObject

Inherits: `WcuMapObject`

This class extends the WcuMapObject by adding two properties; `Url` and `Model`.

The property `Url` of type `string` contains the URL of the camera. The Model property is of type `EVideoSensor`, and reveals the type of the camera, used in the `VideoPlayerFactory` class.

6 Analysis

6.1 Validation

6.1.1 Generalized solution

The goal with the work was to construct a generalized solution that would allow the system to work with an arbitrary camera. One of the problems with the market for IP network cameras is that there is no standard way of accessing them. They use MPEG-4 as encoding technique, but the actual access to the camera is different for each model. The method the vendors prefer is using Internet Explorer, which downloads an ActiveX control from the camera and then allows for viewing and configuring of the available features. It makes it difficult to embed the viewer in other applications.

Hence, the goal was not accomplished fully. The solution is general for the four camera models that were procured, in the way that the user does not need to be aware of the kind of camera that is viewed. However, since all cameras have different interfaces for communication, it was not possible to devise a solution that would work with any give camera model.

6.1.2 Low bit rate

If we revisit the problem statement, it said that the WCU clients operate on a link with very low bandwidth. Furthermore, the bandwidth had to be shared by several applications, such as the regular WCU traffic.

The solution that was implemented did work under the bandwidth constraint as well as the computational boundaries, including memory consumption. The WCU system was not developed fully, and no tests could be carried out to see how regular WCU traffic would perform in a real environment.

The perceived quality of service was rather low. Tests showed that the PSNR of the video signal was around 25 dB over a GPRS link. The tests did not take lost frames into account.

6.1.3 Functionality of the solution

In the design phase, some use cases were defined to demonstrate the functionality of the system. The video plug-in that was implemented carries that functionality.

6.2 Evaluation

As defined in section 2.3, low bit rate is defined as the region of up to 200 kilo bits per second. Since the camera is supposed to send a video stream, the uplink speed of the camera network interface sets the limit. The networks used; GPRS/EDGE and UMTS, offer asymmetrical bit rates and the boundary for those networks is the available uplink throughput, which is usually below 64 kbits/s.

In the final solution, a practical limit seemed to be at around 40 kilo bits per second. If the bit rate exceeded 40 kbits/s, the quality of the signal down-graded rapidly, with stuttering effects and sometimes no picture at all.

To measure the quality of the signal, PSNR measurements were performed. A short reference video sequence (“Foreman”, 176x144 pixels) was encoded with MPEG 4:2 and sent over the UMTS network at 40 kbits/s. On the receiver side, the sequence was stored and on a frame-by-frame basis the PSNR was calculated (402 frames). This test was repeated several times over a period of one week to include varying network conditions. Then, the median value of all the tests was taken. This is somewhat a blunt tool for evaluating the quality, since what is actually measured is the product of the quality of the encoder and any lost packets. The compression was around 1:50.

PSNR measurement, “Foreman”, 176 x 144 pixels, 402 frames, MPEG 4:2 at 40 kbits/s (values in dB):

Color	**Median**	**Mean**	**Variance**
R	28.86	28.81	2.82
G	29.79	29.82	2.99
B	27.74	27.60	2.63
All	28.69	28.74	3.63

7 Conclusions

This chapter provides a summary of the work and conclusions.

7.1 Summary

A solution for the distribution and viewing of live video in the WCU 2.0 was researched and implemented. The solution is general in the sense that the user is unaware of the camera being viewed. The solution has proven to work over low bit rate links, with relatively high latency.
As video codec, MPEG-4:2 is used, and transported over RTSP/HTTP. The solution was implemented in C# on the .Net 3.0 framework. The perceived quality of service is acceptable.

7.2 Conclusions

7.2.1 Video format

MPEG 4:2 was selected as the video format for this work.
The reasons supporting the decision:

- It is a standard developed by one of the major expert organizations in the field, thus it can be assumed there is wide-spread support in software and hardware.
- Despite being only a few years old, it has to be regarded as a mature technology, also reinstating its near-ubiquitous position.
- The compression properties at low bit rates are regarded as among the best at the moment.
- Its error-resilient properties are good.

Another video format that was considered is the H.264/MPEG 4 AVC. However, due to the processing requirements being too high, it was overlooked.

7.2.2 Network design

The proposed solution is based on a direct connection between the viewer and the source. This decision was based on the fact that the latency was too high on any of the other designs that were investigated. There are a few immediate drawbacks of using this design, such as scalability and robustness.

Another design that was tested, using an intermediate relay server, does not share the drawbacks like scalability. However, the latency problem was significant and disqualified that architecture. Yet another design is to use a hybrid network setup, one in which the client would serve other clients. Due to the problems with latency when using a relay, this option was not investigated further.

With the advancements in the field of digital video, the modern compression techniques have made it possible to stream live video over low bit rate networks such as GPRS/EDGE and UMTS. One can expect the bit rates in the wireless networks to increase, and at the same time, the compression techniques getting more effective and thus provide better visual quality for the end user.

In this work, what turned out to be the toughest constraint was the bit rate available. If the bit rates were to increase in the networks, the higher data rate would most likely demand more processing on the decoder side. In the WCU clients the processing capabilities are adequate and there is room for higher bit rates. A trend in the hardware industry is to move video decoding from the CPU to the graphics processor, which is currently only available for high-end video cards. When higher bit rates are available in the wireless networks, specialized hardware support for video decoding should be available even on low-end cards.

When it comes to the network architecture, the model used in this work is robust when it comes to the overall functionality. A failure in a client or a video camera will only disable that device, not the entire video functionality. The server is a special case, since it does the book-keeping of all the connected sensors, but that is more of a WCU-related problem. In case the server fails, there is no service whatsoever, let alone video.

The solution is not very scalable though. The number of concurrent viewers connected to a particular camera is limited by the camera or the link bit rate. This could be solved by using an intermediate relay agent, provided it can keep the added delay low. In the applications where video is intended to be used, no latency is the ideal case.

Regarding the generalization, it turned out that the cameras use standard coding techniques and standard transport protocols. However, on the application protocols, they differ, thus making it difficult to create a truly generalized solution that works for an arbitrary camera. Until there is a uniform way of accessing the cameras, a truly generalized solution is very difficult to achieve. The RTSP protocol or similar has the potential to allow a unified solution by eliminating the need to write special application-layer drivers or adapters.

7.3 Future work

In the devised solution, there is certain knowledge about the different cameras in each client. If another camera is added, all existing clients need to be updated in order to be able to connect to the new camera. Another solution, which uses an intermediate translation layer, would be to prefer, provided that it does not add any significant latency.
It would be a real benefit to similar projects if there would be a uniform way of accessing the cameras. Two of the cameras support the real-time streaming protocol (RTSP) [40], which seems to be a promising protocol that could provide the uniformity needed when it comes to viewing.
If a uniform access is achieved by some standard protocol like RTSP, there is still the problem of accessing the camera-specific features, like PTZ, storage or motion-detection. To unify all this into one standard is a big task and may not be feasible.
Another area where there is a possibility for advancements is the relay agent. It would be nice to have a low-latency, robust intermediate relay agent. Of course, such an agent would share the same problem sets with regards to camera-specific features.

8 References, indices

8.1 References

8.1.1 Bibliographical

[1] 3GPP (3rd Generation Partnership Project), "Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP)," TS 26.244 release V7.0.0, Jun 2006.

[2] Antonini, M. et al., "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205-220, Apr 1992.

[3] Balk, A. et al., "Adaptive MPEG-4 video streaming with bandwidth estimation", in proceedings of *QoS-IP 2003: Quality of service in multiservice IP networks*, Milano, Feb 2003.

[4] Benoit, H., "Digital television: MPEG-1, MPEG-2 and principles of the DVB System," 1st ed., Focal Press, 2002.

[5] Björk, N., Christopoulos, C., "Transcoder architectures for video coding," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 1, pp. 88-98, Feb 1998.

[6] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", *IETF RFC 2119 "best current practice"*, RFC Editor, 1997.

[7] Calafate, C.M., Malumbres, M.P., "Testing the H.264 error-resilience on wireless ad-hoc networks," in *Proceedings EC-VIP-MC 2003*, vol. 2, pp. 789-796, 2003.

[8] Christopoulos, C., et al., "The JPEG2000 still image coding system: An overview," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 4, pp. 1103-1127, Nov 2000.

[9] Cranley, N, et al., "User perception of adapting video quality," *International Journal of Human-Computer Studies*, vol. 64, no. 8, pp. 637-647, Aug 2006.

[10] Ebrahimia T., Horneb C., "MPEG-4 natural video coding – An overview," *Signal Processing: Image Communication*, vol. 15, no. 4-5, pp. 365-385, Jan 2000.

[11] Feig, E., "A fast scaled-DCT algorithm," *Proceedings of the SPIE – The International Society for Optical Engineering*, vol. 1244, pp. 2-13, 1990.

[12] Fielding et al., "Hypertext Transfer Protocol – HTTP/1.1," *IETF RFC 2616 "draft standard"*, RFC Editor, 1999.

[13] Golja, M. et al., "WMPStat: a tool for measuring the correlation between application and network layer traffic of streaming video over Internet ," *Proceedings of the 12th IEEE Mediterranean electro technical conference*, vol. 2, pp. 657-660, May 2004.

[14] Golston, J., "Comparing media codecs for video content", *Embedded Systems Conference*, San Francisco 2004.

[15] Handley, M., et al., "SDP: Session description protocol", *IETF RFC 4566 "proposed standard"*, RFC Editor, 2006.

[16] Hong, D. P. et al., "Evaluating the impact of emerging streaming media applications on TCP/IP performance," *IEEE Communications Magazine*, vol. 39, no. 4, pp. 76-82, Apr 2001.

[17] Horowitz, M. et al., "H.264/AVC baseline profile decoder complexity analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 704-716, Jul 2003.

[18] ISO standard 7498-1, "Information processing systems - OSI Reference Model. The basic model", paragraph 5.2.1.1, 1994.

[19] ITU-T, "Generic coding of moving pictures and associated audio information: video," *ITU-T Recommendation H.262*, 1994.

[20] ITU-T, "Subjective video quality assessment methods for multimedia applications", *ITU-T Recommendation P.910*, Sept 1999.

[21] ITU-T, "Video coding for low bit rate communication," *ITU-T Recommendation H.263*, v. 3, Nov 2000.

[22] Kamaci, N., Altunbasak, Y., "Performance comparison of the emerging H. 264 video coding standard with the existing standards," In Proceedings of *ICME '03 International Conference on Multimedia and Expo*, vol. 1, pp. 345-348, 2003.

[23] Kessler, J, et al., "Product Specification WCU 2.0", [company confidential document], Sept 2006, Saab Systems.

[24] Li A. H. et al., "Data partitioning and reversible variable length codes for robust video communications," *in proceedings on Data Compression Conference*, pp. 460-469, Mar 2000.

[25] Loguinov, D. and Radha, H., "Measurement study of low bit rate internet video streaming," in *Proceedings of the 1st ACM SIGCOMM Workshop on internet Measurement* (San Francisco, California, USA, Nov 1-2, 2001). IMW '01. ACM Press, pp. 281-293.

[26] Marpe D., et al., "H.264/MPEG4-AVC fidelity range extensions: Tools, profiles, performance, and application areas", *IEEE Int'l. Conf. on Image Proc.*, Genova, Sept. 2005.

[27] Marpe, D. et al., "Performance evaluation of Motion-JPEG2000 in comparison with H.264/AVC operated in pure intracoding mode," *Wavelet Applications in Industrial Processing*, volume 5266, Ed. Frédéric Truchetet, SPIE, pp. 129-137, 2004.

[28] Marpe, D., Cycon, H.L., "Very low bit-rate video coding using wavelet-based techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 1, pp. 85-94, Feb 1999.

[29] McMurtry, C., et al., "Microsoft Windows Communication Foundation Hands-on," Beta ed., Sams Publishing, May 2006.

[30] Microsoft Corporation, "Advanced Systems Format (ASF) Specification," revision 01.20.03, Dec 2004.

[31] Osterman, J. et al., "Video coding with H.264/AVC: Tools, performance, and complexity," *IEEE Circuits and Systems Magazine*, vol. 4, no. 1, pp. 7-28, 2004.

[32] Postel, J, "Transmission control protocol," *IETF RFC 793 "standard"*, RFC Editor, 1981.

[33] Postel, J, "User datagram protocol," *IETF RFC 768 "standard"*, RFC Editor, 1980.

[34] Radha, H. M., van der Schaar, M., Yingwei Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 53-68, Mar 2001.

[35] Reibman, A. R. et al., "Network monitoring for video quality over IP," *in proceedings of 2004 Picture Coding Symposium*, pp. 1-6, 2004.

[36] Rejaie, R., et al., 1999. Quality adaptation for congestion controlled video playback over the Internet. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (Cambridge, Massachusetts, United States, 1999). SIGCOMM '99.

[37] Rhee, I. "Error control techniques for interactive low-bit rate video transmission over the Internet," *SIGCOMM Comput. Commun. Rev.* 28, 4, pp. 290-301, October 1998.

[38] Santa-Cruz, D. et al., "JPEG 2000 performance evaluation and assessment," *Signal Processing: Image Communication*, vol. 17, no. 1, pp. 113-130, Jan 2002.

[39] Schulzrinne H. et al., "RTP: A Transport Protocol for Real-Time Applications," *IETF RFC 3550 "standard"*, RFC Editor, 2003.

[40] Schulzrinne, H. et al., "Real Time Streaming Protocol (RTSP)," *IETF RFC 2326 "proposed standard"*, Apr 1998.

[41] Schwarz H., Wiegand T., "The emerging JVT/H. 26L video coding standard," *In proceedings of IBC 2002*, 2002.

[42] Sells C., Griffith, I., "Programming Windows Presentation Foundation," 1 ed., O'Reilly Media, September 2005.

[43] Sikora, T., "MPEG digital video-coding standards," *IEEE Signal Processing Magazine*, vol. 14, no. 5, pp. 82-100, 1997.

[44] Srinivasan, S., et al., "Windows Media Video 9: overview," *Signal Processing: Image Communication*, vol. 19, no. 9, pp. 851-875, Oct 2004.

[45] Srinivasan, S., Regunathan, S. L., "An overview of VC-1," *Proceedings of SPIE - the International Society for Optical Engineering*, no. 2, pp. 720-728, 2005.

[46] Srinivasan, S., Regunathan, S. L., "Computationally efficient transforms for video coding," *IEEE International Conference on Image Processing*, vol. 2, pp. 325-328, Sept 2005.

[47] Stockhammer T., Hannuksela, M. M., Wiegand T., "H.264/AVC in wireless environments," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 657-673, July 2003.

[48] Sullivan, G. J., "Overview of international video coding standards (preceding H.264-AVC)", ITU-T, July 2005.

[49] Sullivan, G. J., et al., "The H.264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions". In *SPIE Conference on Applications of Digital Image Processing XXVII*, (2004).

[50] Sullivan, G. J., Wiegand, T., "Video compression – from concepts to the H.264/AVC standard," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 18-31, Jan 2005.

[51] Talluri, R., "Error-resilient video coding in the ISO MPEG-4 standard," *IEEE Communications Magazine*, vol. 36, no. 6, pp. 112-119, June 1998.

[52] Tan, K.T., Ghanbari, M., "A multi-metric objective picture-quality measurement model for MPEG video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 7, pp. 1208-1213, Oct 2000.

[53] Tun M., Fernando, W.A.C., "An error-resilient algorithm based on partitioning of the wavelet transform coefficients for a Dirac video codec," *Proceedings of Information Visualization (IV'06)*, pp. 615-620, July 2006.

[54] Tung, Y. S. et al., "DSP-based multi-format video decoding engine for media adapter applications," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 1, pp. 274, 2005.

[55] Vetro, A., Christopoulos, C., Huifang S., "Video transcoding architectures and techniques: an overview," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 18-29, Mar 2003.

[56] Wahlgren, C, "Video för WCU", [company confidential document], Saab Systems, Feb 2006.

[57] Wallace, G.K., "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii-xxxiv, Feb 1992.

[58] Wang, Zhihang et al., "Studying streaming video quality: From an application point of view," *Proceedings of the ACM International Multimedia Conference and Exhibition*, pp. 327-330, 2003.

[59] Wang, Zhou et al., "A universal image quality index," *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81-84, Mar 2002.

[60] Watson, A.B. et al., "Digital video quality metric based on human vision," *Journal of Electronic Imaging*, vol. 10, no. 1, pp. 20-29, Jan 2001.

[61] Wenger, S., "H.264/AVC over IP," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 645-656, Jul 2003.

[62] Wiegand, T., Sullivan, G.J., et al., "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560-576, Jul 2003.

[63] Xiong, Z. et al., "A comparative study of DCT- and wavelet-based image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 5, pp. 692-695, Aug 1999

[64] Yasuda, M. et al., "MPEG2 video decoder and AC-3 audio decoder LSIs for DVD player," *IEEE Transactions on Consumer Electronics*, vol. 43, no. 3, pp. 462-468, Aug 1997.

[65] Zimmermann, R., "Streaming of DivX AVI movies," *Proceedings of the 2003 ACM symposium on Applied computing*, pp. 979-982, 2003.

8.1.2 Online

[66] BBC, "Dirac video codec", *available at http://dirac.sourceforge.net/overview.html*, [retrieved Feb 5 2007].

[67] ISO/IEC MPEG Licensing authority, "MPEG-4 visual attachment 1," *available at http://www.mpegla.com/m4v*, [retrieved Feb 5 2007].

[68] Stallman, R., "GNU General public license," *available at http://www.gnu.org/copyleft/gpl.html*, version 2, 1991 [retrieved Feb 5 2007].

[69] Stallman, R., "GNU Lesser general public license," *available at http://www.gnu.org/copyleft/lesser.html*, version 2.1, 1999 [retrieved Feb 5 2007].

[70] Streaming Download Project, "MMS protocol," *available at http://sdp.ppona.com/*, Dec 2003 [retrieved Feb 5 2007].

[71] TPTEST 5.02, *available at http://www.tptest.se/*, Feb 2007 [retrieved Feb 2007]

Appendix A – Brief overview of the technique behind video coding

This appendix looks briefly at the technique behind coding digital video.

MPEG-2, H.263, MPEG-4:2, H.264/AVC, VC-1 and JPEG are all block-based, DCT-based, encoding techniques [21], [64], [62], [45], [57].

Predictive coding and motion compensation

In video, large areas of the image usually only change due to motion; hence one type of video compression is to encode only the difference between two different images [48]. The technique is usually called *motion compensation.*

There are three main types of pictures; intra-pictures, predicted pictures, and bi-directional predictive pictures. Intra-pictures (I-pictures or key frames) do not hold reference to any other image than themselves. For video that tends to be more static, I-pictures may be transferred more seldom, whereas in high action video, a higher frequency of I-pictures is needed. Predicted pictures (P-pictures) may reference previous pictures, thus requiring fewer bits to encode. Bi-directional predictive pictures (B-pictures) may reference two or more already decoded pictures, both previous and subsequent, to average the prediction. B-pictures require even fewer bits to encode than I-, or P-pictures.

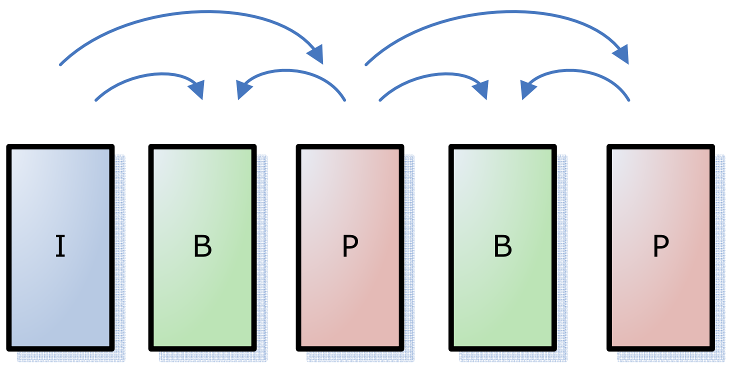


Figure 8-1: Example of predictive coding of pictures (from [48]).

The actual sequence of pictures is based on the type of video and selected at the discretion of the encoder.

The discrete cosine transform

The DCT, discrete cosine transform, is often used in image/video processing due to its energy compaction properties [63].
The image to be encoded is arranged into *m* × *n* pixel blocks, which are then transformed by the forward DCT given below (here as specified in the ITU-T H.263 recommendation [21], on 8 × 8 blocks):

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2x+1)v\pi}{16}\right]$$

where

$x, y =$ spatial coordinates in the pixel domain;

$u, v =$ coordinates in the transform domain;

$C(u), C(v) = {}^{1}\!/\!_{\sqrt{2}}$ for $u, v = 0$;

$C(u), C(v) = 1$ otherwise.

The resulting 64 coefficients are clipped into a signed 12-bit range (-2048 – 2047) and can then be fed into the inverse transform, IDCT, shown below:

$$f(u,v) = \frac{1}{4}\left[\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2x+1)v\pi}{16}\right]$$

where

$C(u), C(v) = {}^{1}\!/\!_{\sqrt{2}}$ for $u, v = 0$;

$C(u), C(v) = 1$ otherwise.

The ITU-T recommendation specifies using at least 64-bit floating point accuracy [21]. For a block of 8 × 8 pixels, the DCT can be performed with only 54 multiplications [11].

The discrete wavelet transform

Another transform that has gained popularity in recent years, is the DWT, discrete wavelet transform. It has been adopted by the JPEG-2000 image encoding standard [8] and by Dirac.
The nature of the DWT will not be detailed here as no video standard currently employs the technique. There is an extensive paper [2] by Antonini et al.
While a straight-forward approach of using DWT seems to yield less obvious gains compared to DCT in [63], other evaluations show a slight gain in favor of DWT [28].
Xiong et al. came to the conclusion in [63], that the difference between DCT and DWT is very modest. A quote from their paper:

> *"[…] the main factors in image coding are the quantizer and entropy coder rather than the difference between the wavelet transform and the DCT".*

Worth noting is that one paper, [53], introduces an algorithm to make the DWT more resilient to transmission errors. Their idea, to partition the transform coefficients into groups and then independently process each group, resulted in a significant (5 dB PSNR) improvement over the default case.

Appendix B – Application and transport protocols

Application and transport protocols

Data communication network protocols can be divided into different layers. The property of each layer is such that it only uses functions of layers below and its own functions can only be used by layers above [18]. This section will describe relevant application and transport protocols more detailed.

Transport containers

From the perspective of the layered Internet protocol suite, a video stream can be divided into different layers.
One of the layers is the transport stream. The purpose of the transport stream is to multiplex audio and video, and to handle the synchronization among them. Depending on the container type, other streams may be multiplexed as well, such as subtitling in MPEG-4:17 or synthetic texture streams in MPEG-4:19.

Below, a few relevant examples of container formats are accounted for.

MPEG-TS – MPEG-2:1

The MPEG Transport stream was defined in the MPEG-2 standard, part 1. It is designed to carry synchronized video and audio. It has some support for burst-errors, by using Reed-Solomon error-correction. It is used in the digital video broadcasting (DVB) system. The MPEG-2 standard also defines another container, called program stream. Since it was designed for reliable media, such as a hard disk, it is not considered in this paper.

MP4 – MPEG-4:14

MP4 was specified together with the MPEG-4 standard in 1998. It is based on Apple Computers QuickTime container file format. MP4 can hold video and audio streams, but can also hold other streams such as subtitles and still images. It is designed with Internet streaming in mind.

3GP – Third generation partnership project video file format

3GP is a simplified version of MP4, designed for use in 3G mobile phones [1].

ASF – Advanced systems format

This container type is an open but proprietary format developed by Microsoft. The main features of ASF are (from [30]):

- Synchronized audio video streams,
- May also include meta data such as artist name,
- Is designed with digital rights management in mind, meaning that the ASF container will hold the encryption keys and cipher.

Its specification is freely available, but the end user license agreement stipulates that implementations are distributed in "object code form only", i.e. no open source code implementations. ASF, like MP4, was constructed for streaming purposes.

AVI – Audio video interleave

AVI was developed in 1992 by Microsoft, and is a relatively old container and regarded by many as obsolete. It supports multiple synchronized audio and video streams, but lacks built-in support for some features that are used in advanced encoding techniques, such as B-frames. Furthermore, AVI was not designed for streaming purposes, since the decoder has to read the file in a non-sequential fashion [65].

Application layer streaming protocols

In this section, I will discuss various application protocols developed for streaming applications.

RTSP – Real time streaming protocol

RTSP was developed on an initiative from IETF in 1998 [40]. Its main objective is "control over the delivery of data with real-time properties". It is a stateful protocol, where a "unique" session number is used in every transaction, thus there is no need for a permanent TCP connection. RTSP packets can be transported in RTP packets. However, since no ports are defined explicitly for RTP to use [39], RTP packets may have problems traversing firewalls.
Some of the commands:

- "Setup" and "Teardown" commands are used for initiating and closing the session.
- "Play" and "Pause" commands control the stream.

For this work, RTSP may suit as a uniform way of accessing each different kind of camera. If the camera had a RTSP server, a generic RTSP client would suffice in order to access an arbitrary camera.

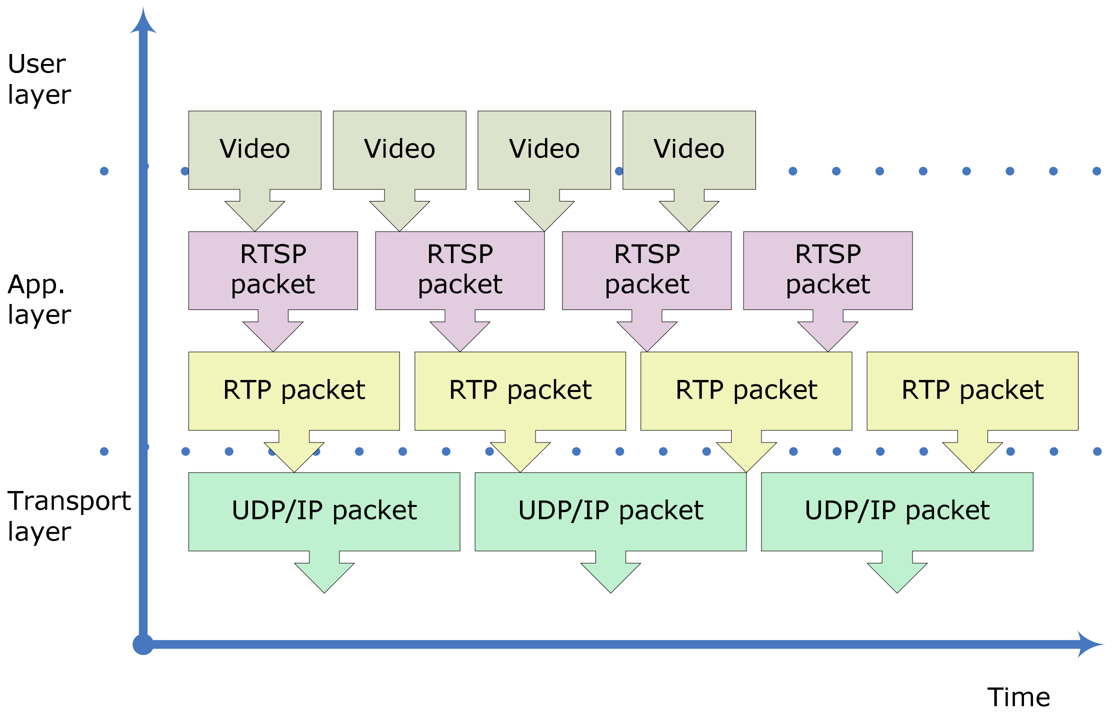


Figure 8-2: An overview of packet encapsulation, with RTSP/RTP.

MMS – Microsoft media server

MMS is a proprietary and closed application protocol used in the Microsoft Windows Media Series. One of the main features of MMS is to disable the end-user from storing the streaming content as a file, and an attempt to reverse-engineer the protocol has been made in [67]. With the latest release Window Media 9 Series, Microsoft has added support for RTSP [13]. The properties of MMS are very similar to RTSP. MMS can use both TCP and UDP transport packets [47]. It is sometimes denoted MMST and MMSU, respectively.

RTP – Real-time transport protocol

This protocol is designed to provide end-to-end delivery of data with real-time properties. It does not guarantee timely delivery; this is meant to be handled by underlying transport protocols. It can use either TCP or UDP and does not have any specific ports assigned, which may cause problems with firewalls. There are two components of RTP; the data and control messages are separated. For this work, it is worth noticing that RTP over UDP requires no permanent connection.

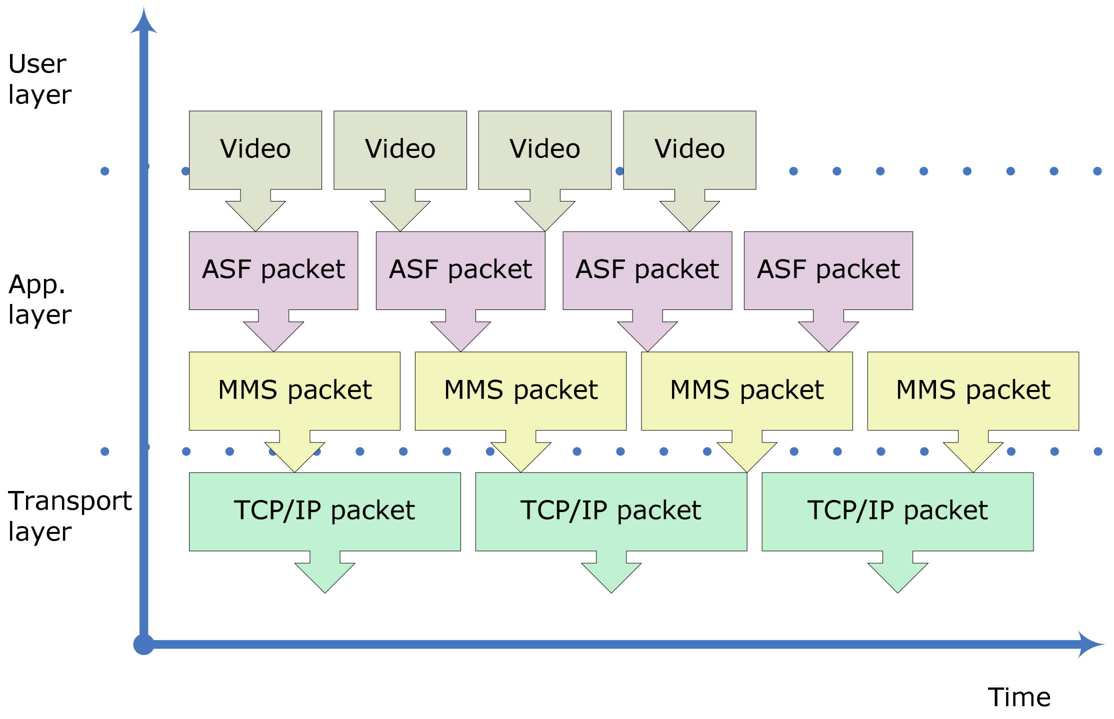


Figure 8-3: An overview of packet encapsulation, with ASF/MMS (from [35]).

Appendix C – Transcoding complexity

Transcoding complexity

One important characteristic put forward in [48], is the complexity of the codec. Even though hardware continues to develop according to Moore’s law, especially on small handheld devices, transcoding complexity is still an issue. In [44], evaluations show that the WMV-9/VC-1 decoder (“main” profile) requires 2-3 times less computations than H.264/AVC (“baseline”).
In another paper, [17], Horowitz et al. present a thorough analysis of the computational complexity of a software-based H.264/AVC decoder. One of their conclusions is that the H.264/AVC is two to three times higher in terms of time complexity than for H.263.

Appendix D – License issues

License issues

In this section, license issues regarding implementation and usage of coding techniques, protocols and software libraries are covered briefly. A complete coverage of the issues is well out of the scope of this work.

MPEG standards

Even though MPEG-2 and MPEG-4 are open standards approved by ISO/IEC, certain rights of usage are limited. Only for MPEG-4:2 visual, 26 companies have filed approximately 600 patents [67]. An organization called MPEG LA has been established to handle all issues regarding MPEG licenses.
Generally, a license fee is required for each decoder. However, exceptions to the default case do exist. This needs further examination, and probably requires an attorney at law.

ITU-T standards

H.264/AVC licenses are handled by MPEG LA, according to themselves the licensing process is deliberately made simpler than is the one for MPEG-4. For H.263, a license is required both for encoder and decoder software.

Microsoft products

According to [30], the data format ASF is free to implement in products but they may not be distributed in source code form.
Other Microsoft technologies, such as MMS and Windows Media Player, require licensing before redistribution. However, there is not necessarily a fee or royalties required.

GPL

The Gnu general public license (GPL) version 2, [68], has gained a lot of popularity since its release in 1991. It is called a *free software* license; free meaning that the end-user is free to modify the source code and re-releasing it, under the condition that the GPL still applies. If the software is redistributed in binary form, the source code must be available upon request. The licensee may charge a fee for the software, even for the source files, as long as the fee for the source code is less than or equal to the fee for the binaries.

If a program is combined with a GPL’d program so that they form a larger program, then the whole program must be licensed under GPL. “Combined” could mean they are runtime linked into the same address space. Pipes, sockets and command-line arguments usually do not count as “combining”. Ultimately, it would up to a court of law to decide.

LGPL

The LGPL was designed to be used when licensing software libraries. It is a less restrictive license than the GPL. Work under the LGPL license may be linked from another program, and redistributed under any chosen terms subject to the following condition, as stated by Stallman in section 6 of [69]:

> *“[…] the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.”*

Appendix E – Network latency and throughput

Network latency and throughput

To give an estimate of the wireless network latency, the network tool ping was used to observe the round-trip times. The reference server was the main KTH web server, www.kth.se, (130.237.32.107).

GPRS/EDGE round-trip times	Milliseconds
Ping packets sent/received	140/140
Median	432
Mean	436
Variance	3229

UMTS round-trip times	Milliseconds
Ping packets sent/received	138/138
Median	249
Mean	257
Variance	1093

To measure the TCP packet throughput, a tool named TPTEST5 [71] was used. The reference server was referens.sth.ip-performance.se (192.36.144.178). Cards tested: AirCard 775, Globesurfer iCon. Network provider: Telia. The AirCard was around 8% faster downstream and 23% faster upstream. Interestingly, the EDGE upstream throughput was consistently higher than UMTS.

UMTS throughput	Kilobits per second
Test runs	6
Median TCP downstream	360.6
Median TCP upstream	57.5

EDGE throughput	Kilobits per second
Test runs	8
Median TCP downstream (GlobeSurfer)	193.7
Median TCP upstream (GlobeSurfer)	65.5
Median TCP downstream (AirCard)	209.9
Median TCP upstream (AirCard)	84.5

Appendix F – Error resilience in digital video

As stated in Appendix A, modern video encoding techniques usually involve some form of picture-prediction and motion-compensation.

When a compressed video signal is subjected to noise, as typically can be found in wireless links, errors can affect the signal seriously. Errors introduced are propagated with each P-, or B-picture, and only expensive I-frames can resurrect the signal. Standard error-detection and error-correction measures, like forward error correction (FEC) and variable-length coding (e.g. Huffman), are necessarily employed on the encoder side. However, those measures are not enough.

In [51], Talluri describes in detail four tools that have been introduced in the MPEG-4 standard to mitigate errors in the bit stream.

They are:

- Video packet resynchronization,
- Data partitioning,
- Reversible VLC, and
- Header extension code.

They are briefly recounted for below.

Video packet resynchronization

The image is divided into macroblocks, and each macroblock is separately encoded with variable-length coding. One of the types of errors that can happen if bits are missing is loss of synchronization. With VLC, the decoder cannot determine where the next code word starts, and without any counter-measures the picture quality would degrade to zero. One counter-measure is to use resynchronization markers. When the decoder has lost track, it can look for a marker and then it can find the next code word. The drawback is that there must be no picture-prediction across these markers, which increases the number of bits in the encoded data.

Data partitioning

If an error is detected within a video packet, usually all macroblocks are discarded. In MPEG-4, efforts are made to recover the healthy macroblocks by partitioning the data into texture and motion vector data. In this way, if an error is detected within the texture data, the motion vector data can still be decoded. The other way around is not possible though, since texture data references the motion vectors.

Reversible VLC

Variable-length codes are code words with redundant data added to make it easier to detect bit errors. Reversible VLC work like VLC, but they can also be decoded backwards. With this property, the decoder has an easier task of determining where the error resides, and fewer bits need to be discarded. H.263+ and MPEG-4 have adopted the RVLC scheme.

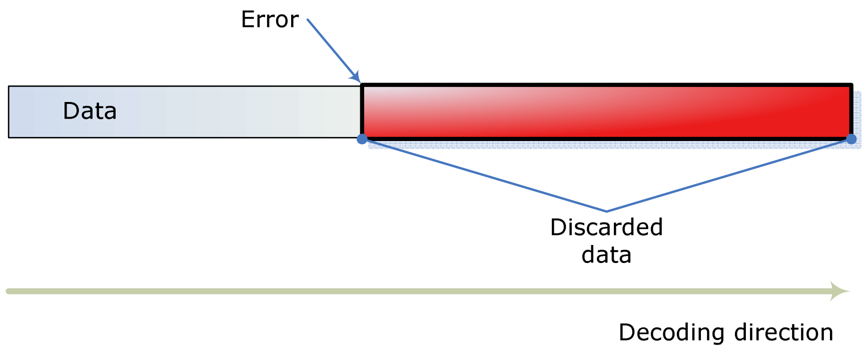


Figure 8-4: Error detection with VLC decoding (from [24]).

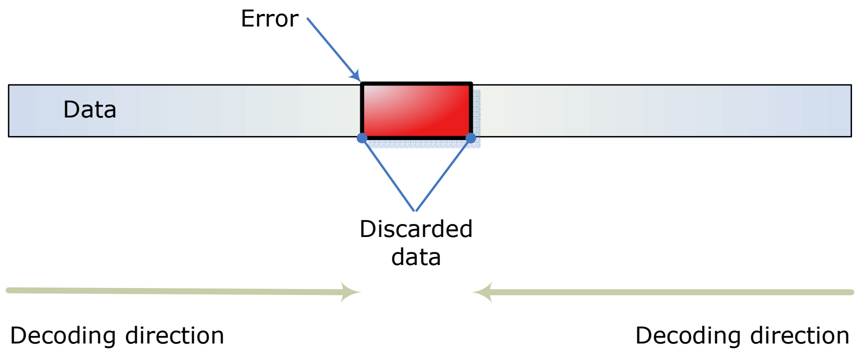


Figure 8-5: Error detection with RVLC decoding (from [24]).

Header extension code

The MPEG-4 decoder needs the header of each video packet in order to decode the bit stream correctly. The header contains metadata such as information on the type of picture-prediction and time stamps. If there is an error in the header, the decoder will discard the whole packet.

The header extension code is a 1-bit flag used to indicate that each header is sent redundantly with each frame. By comparing the packet header with the frame header, the decoder can determine bit errors and fewer frames need to be dropped.