Agder University
Faculty of Engineering and Science

Master of Science Thesis

Optimized distribution of internationalized DRM-protected multimedia content

by

Christian Kroken
Sigbjørn Tvedt

Supervisor: Vladimir Oleshchuk

Grimstad, 2008

Contents

Acknowledgements

We would like to thank the people at Filmarkivet AS for providing us with the thesis. We would also like to thank our supervisor Vladimir Oleshchuk for providing us with valuable feedback and guidance during the project period.

Chapter 1

Introduction

1.1 Filmarkivet AS

Filmarkivet AS is a multimedia content distribution company based in Kristiansand. Their main product is a movie rental portal on the internet.

Filmarkivet.no was released publicly 18. November 2004 by Norsk filminstitutt[17] in cooperation with Agder Energi AS and NorgesFilm AS. The goal of this project was to be able to deliver movies using the internet as the distribution medium.

1.2 Background

The delivery of multimedia content on the web will give a lot of new challenges. One of these limitations is the last mile* capacity on the lines that connect the end users to the internet. Because of this, it is necessary to limit the data that is transmitted while keeping the quality of the content as high as possible. It would also be desirable to give the user a choice of which audio and closed caption track that should be played, almost like a dvd. Alongside with the technical challenges of delivering the content, we have the problem of customers accessing the content without permission and customers that are creating extra copies after renting a copy.

When delivering the content, there is always a risk that the end user would like to create an unlicensed copy, and it is therefore necessary to use a system that will keep the data in an encrypted state from leaving the server, until it is displayed to the end user. As keeping information secure is a key point for most e-commerce solutions,

*The last connection between the network operator and the user

it is possible to use a lot of techniques from this area, but unlike ordinary security systems where we have 2 legal users and one attacker, the role as an attacker has been moved over to one of the users. Due to the fact that the receiver and attacker are the same person, it is also necessary to have a way of keeping the content secure while delivering it to the customer.

Another way of handling this would be by using watermarking. The watermark method embeds data into the audio/video stream, and it will therefore be possible to identify the person that has shared the media. This method will not hinder the spread of the media, but will instead rely on the users fear of getting caught to prevent illegal distribution.

The reason why Filmarkivet want us to look into new technologies is that their site isn't compatible with all browsers, and is locked to the Microsoft Windows platform. They also want to update their security solution in order to use newer DRM methods, along with a more intuitive user interface. Our project will therefore be a study of different streaming technologies.

1.3 Thesis Definition

In this project we will investigate methods for distribution of internationalized multimedia content with DRM protection. The current solution requires that all movies on the video server must be stored with subtitles burned into the video, which means one full version copy for each audio track. This makes it more demanding to create and administrate the content, as well as requiring more storage on the server. In this project we will investigate solutions for creating, storing and distributing this content that are more efficient from a performance point of view. One possible approach would be to handle multiple audio and/or subtitles along with the video with synchronized delivering of the content. We will consider different ways of dealing with this while using DRM protection and determining the performance compared to non-DRM and partly-DRM protected solutions. We will analyze the way DRM works and how it impacts the usability and performance of the service. We will also consider how the new player technology may influence performance and usability, and make recommendations to Filmarkivet with respect to the new platform choice.

Keywords: DRM protection, watermarking, distribution of multimedia content

Chapter 2

State of the Art

In this chapter we will present several DRM solutions and how they work, player technologies and streaming protocols used when delivering a multimedia stream. We will also present a few ways of attacking the protected content, as well as presenting different schemes for delivering data.

2.1 Digital Rights Management

Digital Rights Management, or DRM for short is a way of protecting content in digital form. The use of DRM has grown in recent years, since businesses need a way to protect their content and prevent users from copying copyrighted material. It is possible to divide the DRM market into two parts. One part depends on restricting the end user so that they are unable to create copies of the information. This will include measures to prevent the user from moving content from one computer to another as well as preventing the use of screen capture software. There are different levels of protection, ranging from Microsofts Windows Media DRM in Microsoft Vista which can shut down the ability to use digital outputs, to software that will disable the save as choice in the browser. The other way of managing rights is by acknowledging that the most technical users will always be able to get past a restriction scheme. Since it will be possible to create copies, we need to make sure that the user keeps the copy for himself, by making sure that a person leaking the media will be identified.

2.1.1 Requirements for the DRM protection

The different DRM schemes will be deployed depending on the protection demanded from the holder of the rights. If it is only important that nobody except the intended receiver is able to read the content, a simple end to end encryption of the transport stream should be sufficient. If, on the other hand, the content also has to be protected from the recipient, we will need to employ some other methods. This could be the use of trusted zones like set top boxes, or it could be done in software. There are also a lot of different suggestions for DRM systems, but the thing they have in common, is that it should be possible to renew the security system when one is broken. There are a lot of different actors in this marked. The commercial ones, like Microsoft and Adobe usually employ proprietary formats and protocols that they try to protect from everybody else, thus monopolizing the marked and making sure that their players are the only one that are able to play the content. We also find open initiatives, like the MPEG4 IPMP[26], that suggest a extension of existing formats in order to deliver protection using a system that is public available. This will also give the advantage of making it possible for 3. parties to create players for the content.

2.1.2 Uses for DRM

Uses for DRM:

- Movies (DVD, HDDVD/Blue-Ray, Streaming)
- Music (CDs, Internet music)
- Documents

A simple version of DRM have been used on DVDs to protect movies almost since they came out and is still used, although it provides little protection compared to current DRM systems. A better system was developed for HDDVD and Blue-Ray movies to protect them, called Advanced Access Content System (AACS). [5]

DRM is widely used on the internet for Video On Demand (VOD) services that enable users to rent movies and view them in their browser in most cases, or alternatively offline. After checking through Norwegian and International movie sites we have noticed that most of these sites use Windows Media DRM to protect their content. Sites such as filmarkivet.no, sfanytime.no, cinemanow.com use Windows Media DRM. More sites are listed in the Windows Media DRM site. * Other companies, like

* http://www.microsoft.com/windows/windowsmedia/forpros/drm/9series/providers.aspx

Adobe have also shown their interest in making DRM technology with their Flash DRM, which will be discussed further in this chapter.

Some Audio CDs used to have DRM, but since this limited their use on some players it was eventually removed. Sony got a lot of problems for their attempt to protect their Audio CDs, since it installed software on people's computers without their consent and opened their systems to security vulnerabilities. [†]

Apple had their own DRM for music purchased on itunes, which locked the tracks to itunes or for use on iPods. After a time they made DRM-free tracks available at an increased price, but on October 17, 2007 they cost the same as DRM protected ones. [‡]

2.1.3 Known ways of accessing and downloading files illegally

A media stream that is distributed over the internet is always in danger of being downloaded and used in ways that are not accepted by the content owner. For unencrypted content, the simplest way of getting a media file is to extract the address and perform a direct download from the server. Another way that is almost as easy is to read the files from a cached location, usually the web browser cache. As these two methods are the easiest, they are of course easy to prevent. This can be prevented by hiding the URL by running the player in a plugin, thus hiding the real address from normal end-users. In order to create a stronger scheme, we will need to make sure that the address is concealed from the rest of the system, for instance by creating an encrypted tunnel between the plugin and the remote server. The reading of files from cache can be avoided by making the player discard data once it has been played back. This method will also depend on the ability of the application to conceal the data in a way, so that the user is unable to read the data as it is passed through the memory.
A method that requires a little more work will be to copy all incoming data before being fed to the player. This will usually be possible to prevent by using an encrypted connection between the server and the software used to playback the content.

2.1.4 Hardware solutions

As the receiver also will be the attacker, most cable and satellite systems will enforce the DRM by using custom made hardware solutions. These solutions usually consist of a hardware system that will handle the decryption and conversion of the signal, and a smartcard that will be used for user identification. The reason for deploying new hardware to the consumer will often be that this hardware can be seen upon as a trusted and secured system. The system will have been made in a way so that it will be resistant to tampering, thus creating a large barrier for intruders. This also enables the media provider to have a list of equipment that they accept in their network, and they will therefore be able to create and enforce the

[†] http://www.pcworld.com/article/id,124062-page,1/article.html
[‡] iTunes store sells DRM free music http://news.bbc.co.uk/2/hi/technology/6516189.stm

rules that they want. It could be that customers have to use a specific connector in order to play the signal, or it could be that the equipment must enforce a copy control scheme, like HDCP[§]. The biggest disadvantage with this system, is of course that every distributor will have to deliver a hardware unit and a smartcard to the customer. The system will on the other hand be more secure, and easier to support than a software solution running on a system of the user's choice due to the control over software and hardware.

2.1.5 Microsoft DRM

2.1.5.1 Windows Media DRM

Microsoft's Digital rights management, Windows Media DRM[22], has been on the market since 1999 and is used by many companies to protect their content.

It has been made in 3 versions since its conception in 1999, and has gained many new features since then. It is available in Version 1, Version 7.X/9 and Version 10, that was released in 2004. The system has several options that can be used to define what the media is allowed to do. This could be ways of limiting the use of the protected files to certain machines or restricting the user in other ways. Some of the options that are given to the creator of the content are:

- Expiration date
- Burn to CD
- Duration
- Counted operations (plays, transfers)

2.1.5.2 Microsoft PlayReady

Microsoft PlayReady is a system that is currently in development from Microsoft. It will be backwards compatible with the Windows Media DRM system, but will also add some new features. The most important feature, from a consumer's point of view, will be the "Domains" feature. This will enable the user to register several media players, up to a limit set by the content provider. After doing this, the user will be allowed to use the media on all devices registered for the user. The user will also be allowed to change the list of devices registered in order to play back old content on new devices.

2.1.6 Adobe DRM

Adobe has two solution for flash that provides protection of files, called Flash Media Server (FMS) [9], and Flash Media Rights Management Server (FMRMS) [8].

[§]http://www.digital-cp.com/

2.1.6.1 Flash Media Server

Version 2 provided some means of protecting content, but in April 2007 Adobe made some upgrades with DRM-like protection. Version 3 was released 4 December 2007. This new version includes H.264 video and high-quality audio support as well as increased protection for streaming content.

Unlike Windows Media DRM FMS doesn't encrypt the media file itself, but uses a different streaming protocol instead. FMS stops the client from caching the content and then reading the file from the browser's temporary files. It also makes sure that the location of the media is hidden from the user, thus protecting it from being downloaded using a downloading tool.
There are several ways to stream media with Flash Media Server 3 [10]:

- RTMP (Real-Time Messaging Protocol)
- RTMPT (RTMP tunneled over HTTP)
- RTMPE (Encrypted Real-Time Messaging Protocol)
- RTMPTE (Encrypted RTMP tunneled over HTTP)
- RTMPS (RTMP with SSL)

The client runs an swf[¶] file either in a web browser, or in Adobe Media Player. The server vill then be able to use a verification of the executable to make sure that it has not been altered and is still trustworthy before any content is transmitted from the server. After the client has been approved, the media will be streamed using the Adobe RTMP protocol. This stream is unprotected, but can then be protected using a extra layer of security like SSL.

A new addition to FMS is RTMPE, which adds a 128bit encryption, but unlike SSL this does not require a certificate. The benefits of RTMPE is that it's easier to implement, has better performance and less impact on server capacity. The Flash Media Server tech overview [10] states that RTMPE requires 15% more processing power than RTMP, and RTMPS(SSL) requires almost 50% more.

2.1.6.2 Flash Media Rights Management Server

FMRMS was released along with Adobe Media Player [4] and Adobe AIR [1], which is "a new cross-platform runtime for rich Internet applications (RIA)." RIA .It's a DRM technology, but instead of encrypting the stream it can protect content after it has been downloaded, more like a regular DRM i.e. Windows Media DRM.

Some of the features of the Flash Media Rights Management Server:

[¶] Adobe Flash executable

- Clients for Windows and Mac.
- Streaming and download protection for flv and fl4v files
- Flexible dynamic rights management gives the possibility of changing user rights and keeping track of media, even after distribution has started.
- Option of creating a custom authorization service and connecting the media to a user's account.

Policy features offered by Flash Media Rights Management Server:

- Start and end date for the policy
- Number of days the policy is valid
- Number of days the license is valid after the user downloads and authenticates the content
- Client applications and versions that can access content (for example, users must have a certain custom Adobe AIR application or a certain version of Adobe Media Player)
- Authentication domains
- Anonymous access allowed or denied
- Off-line user access to content allowed or denied
- Name of the external authorization service attached to the policy

FMRMS also makes sure that any embedded advertisements follow the content, and also contains security controls in case of attempted tampering. This is to ensure that the content owner can take advantage of advertisements even after the user has downloaded the media.

2.1.7 Watermarking

While it is normal to enforce restrictions on the users in order to prevent unauthorized spreading of media, the threat of easily being identified as the source could be a less aggressive method of protecting content. By embedding data identifying the customer that originally purchased the media, we will have a scheme that offers the customer full rights and flexibility to use the media in whatever way they want, while making sure that it is possible to identify and punish those who spread the media without permission.

2.1.7.1 Introduction to watermarking

Watermarking has been used for a long time to mark physical documents in order to make them difficult to copy and impossible to remove the watermark without destroying the original object. In later years, researchers have been working on moving the watermarking schemes

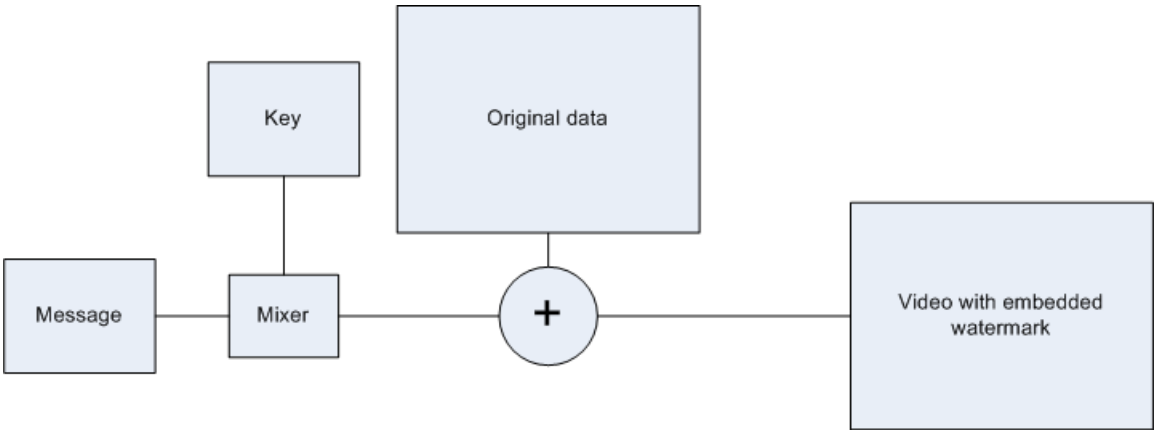

Figure 2.1: Generic watermarking system

into the digital domain. In the digital world, watermarking can be used to prevent playback or recording of media, identify untrustworthy users or distributors or just mark the content so that everybody are able to see that the works are copyrighted.
Displaying content owner information and copyright information, is relatively simple to implement. All that has to be done, it to embed a visible image/watermark on top of the image that should be protected.
The fingerprinting of a single user, or distributor, will require that we equip every copy with a unique watermark. This could be done in the video, and/or in the audio stream depending on the processing power available and if there are parts of the content that are more important than other. This way it will be possible to identify everybody that has had access to the media.
Playback prevention has been implemented in some media players, like the HD-DVD players. All HD-DVD players has to scan for a watermark embedded in the audio stream. If the copy has audio from a movie theater, playback will stop immediately, but if it is a commercial disc, it will check that the disk is factory made or shut down if it is a home made disc.

2.1.8 Multimedia watermarking and identification

In order to be able to identify a person that has uploaded content without permission, we need to be able to insert a personalized Id into every video frame. When doing this with a single image, speed is not critical as the user will usually be able to wait for a few seconds. When embedding this into a movie, we will have to encode 25-30 fps[‖]. If we choose to do this by decoding and encoding this amount of data, it will take a lot of time, and will thus require us to have several large clusters of servers if we are going to be able to perform real time watermarking of all media streams that are transmitted. Solutions to this that have

[‖] Frames per second

been discussed, are to move the watermarking procedure to the end user. This will have the advantage of relieving the servers of the work that has to be done in order to watermark each stream, but as the hardware and software is accessible to the customer, there is a chance that the user will be able to circumvent, or change the watermarking mechanism. A second solution that has been suggested involves the use of the routers along the path [24]. Each router will then add a specific watermark, thus making it possible to trace each router even if we are multicasting the data to a large group. This will ensure that the watermark is added without the server having to perform all the work, while the end user will be unable to modify the equipment in order to defeat the watermark. Another, and probably best solution that researchers are investigating, is a solution where the server only decompresses and incorporates the watermark into parts of the video stream. This way, we will be able to mark all frames in the movie, while it could be possible to do it on demand from the server.

2.1.8.1 Attack types

** As an unreadable watermark will be worthless for identification purposes, we must assume that there will be attacks against any solution that implements this scheme. These attacks could be done intentionally or unintentionally. Either way the system will have to be built in a manner that ensures that the information embedded is readable even after an attack has been attempted.

A threat towards watermarks are the collision attacks. These could be divided into two types. If the same watermark is embedded into several medias, it is possible to identify the watermark by comparing them and identifying similarities. The other method is to collect several "samples" of the same content, but with different watermarks. It is then possible to identify changed regions, thus identifying where the watermark has been hidden. Identifying the watermark will not only enable the user to strip his own watermark from the content, but it will also be possible to add another watermark thus blaming another user for copyright infringement or other crimes. A watermark should therefore change position for every user that receives the content. If we are to use the same watermark at different intervals, it will be possible to predict where the watermark will be present. It could therefore be intresting to use the approach suggested by M. Noorkami [28]. He suggests that we will be using the same reocurring pattern in scenes that are similar, thus making it difficult for an attacker to predict where the watermark will occur. In order to make it even more difficult, a scheme that gives the watermark a unpredictable property [29] can be used, making it possible to read the watermark from a arbitrary frame, but it will still be impossible to statistically predict where the watermark will be.

Some of the attacks that are threatening the existence of watermarks are in reality quite simple optimization or enhancement techniques. These include changing the gamma level, cropping

**These attacks are described in more detail in the thesis "Security Issue and Collusion Attacks in Video Watermarking" [25]

the image frame, using sharpening algorithms or just encoding the video into another format. Other things like changing the framerate, editing of the video and changing the aspect ratio will also impact the watermark.

2.2 Player Technologies

There are several new player technologies for streaming content over the internet that came out in 2008. Some of these have been released in early version without the full range of features that will be in the final version. These players include several features that enable them to better meet todays requirements. As we have mentioned in the previous section, DRM is a solution content creators use to ensure that their content is protected. These player technologies support different kinds of DRM encryption, based on its developer.

2.2.1 Microsoft Silverlight

Microsoft Silverlight [15] is a new cross browser and cross platform plugin for interactive web pages. It supports Microsoft's .NET platform, High definition video, streaming and controls.

Silverlight aims to support the most popular browsers and both Mac and Windows. They're mainly concentrating on Internet Explorer, Firefox and Safari for Mac or Windows. Microsoft states that they're cooperating with Novell to give Silverlight Linux support. Silverlight supports technologies such as AJAX, IronRuby, IronPython, C# and Visual Basic.

The Silverlight platform is currently released in two versions, a stable version 1.0, and version 2 Beta 1 and supports several features that can also be found in Adobe Flash Player [3].

2.2.1.1 Version 1.0

Silverlight 1.0 is the current stable release of Microsoft Silverlight. As this is the first release, it still lacks several technologies, and is currently limited to JavaScript as the only development language.

Some of the important features in Silverlight 1.0:

- Cross browser
- 2D vector graphics
- Cross platform
- JavaScript support
- Ability to stream media

2.2.1.2 Version 2.0 Beta 1

Silverlight 2.0 (earlier named Silverlight 1.1) will be released during 2008. It will be a more mature version of Silverlight 1.0, and contain a lot of new methods for creating content. According to the roadmap [20], Microsoft will continue to extend the Silverlight platform to new platforms and operating systems.

Among the more interesting features for our project that will be added to this release are:

- More programming languages
- Support for controls (TextBox, RadioButton, Calendar etc.)
- Better security
- Content protection of media (DRM)

2.2.1.3 Tools & Formats

Microsoft are currently offering two tools for developing Silverlight applications. It is also possible to use external tools, alternatively you can write code in a text editor.

Microsoft Expression Studio
Expression studio is intended for use by designers. It supports creation of vector graphic, web pages, user interfaces, as well as editing and encoding of media. Expression Studio 2008 will support Silverlight 2 applications.

Microsoft Visual Studio
Visual Studio is an integrated development environment (IDE) for Windows. It supports several languages, and it is possible to extend it to support even more than those who are delivered by Microsoft.

JavaScript
In order to use Silverlight, we need to load it into the web page we are going to use it on. To load the Silverlight plugin we include a JavaScript file supplied by Microsoft. This file contains all the information needed to identify the operating system and browser used to access the web page.

XAML
The XAML format is used in the Microsoft .NET 3.0 and newer versions of the framework for describing user interfaces. XAML, or Extensible Application Markup Language uses the syntactic structure of XML, and is therefore easy to read for humans, and easy to parse and check for errors both manually and by using software.

2.2.2 Adobe Media Player

Adobe Media Player is a new off-line cross-platform media player with a similar function to Windows Media Player. It can play media while online, or off-line from either RSS streams, Flash Video (FLV) files or other supported formats.

It is built on Adobe AIR, which is a cross-operating system runtime that runs on Windows, Mac and Linux (under development). Adobe AIR combines Flash, HTML, Ajax and Flex. Adobe Media Player was released on April 9, 2008.

The player has features to browse, manage and discover tv shows on the internet. Adobe has teamed up with many companies to provide entertainment, such as CBS, Nickelodeon, Comedy Central and MTV, as well as smaller companies.

Features:

- For Windows, Mac & Linux (Under development)
- Plays FLV (Flash Video) files
- Provides content via RSS feeds
- DRM content protection
- Advertisements

2.2.2.1 Tools & Formats

The main thing you need to create a feed for Adobe Media Player is an RSS editor, where you store information about the layout, content and advertisements. You will also need to encode media to the correct format since AMP supports the following: MOV, MP4, MP4V, M4V, 3GP, 3GPP2 and 3G2.

Adobe Flash
You can use Adobe Flash for creating content, since it plays SWF files, in case you want to include applications in your feed. For more info about Adobe Flash consult the next subsection.

Flash Media Server
FMS can act as a server for Flash Video Files (FLV) and protects the stream with the RTMPE protocol. More information about this technology is available in the DRM section of this chapter.

Flash Media Rights Management Server
FMRMS was released along with Adobe Media Player and AIR to provide more ways to protect media content, further info about this technology was provided in the DRM section. RSS The player is basically a multimedia RSS[19] reader, so you can provide content with the use of RSS feeds. You also provide things like banners and backgrounds in the feed. Since

the page is continually updated, you can also provide such things as seasonal themes. RSS feeds are quite often used on web pages today for providing information about news, page updates and more.

2.2.3 Adobe Flash

Adobe Flash is a multimedia browser plugin that's been on the market for a long time. It's very popular on the internet for creating design and animations on websites, as well as creating small games. Lately it is also being used for what Adobe calls "Rich Internet Applications", which is basically more advanced applications, making the experience more interactive. You can create applications with animations by placing objects along a timeline, or place controls such as combo boxes and buttons for user interfaces.

Adobe Flash is used in popular sites such as Youtube, Google Video, and a large amount of websites that use an interactive user interface. Adobe states on their website that 98% of computers have Flash player installed [6].

By using Flash Media Server (FMS) it is possible to stream flash video files with the RTMP protocol or the encrypted RTMPE protocol that protects the media.

Features:

- For Windows, Mac & Linux
- Encrypted streaming with Flash Media Server with RTMPE or SSL

2.2.3.1 Tools & Formats

Adobe Flash editor
This is a tool created explicitly for making content for Adobe Flash, and uses a script language called ActionScript for writing code. Adobe Flash CS3 is the latest version [2]. This is mainly the tool you need for creating flash applications, except for any image editors or similar programs. The Adobe Flash editor is for both programming and design of the application.

Flash Media Server
FMS can act as a server for Flash Video Files (FLV) and protects the stream with the RTMPE protocol. More information about this technology is available in the DRM section of this chapter.

2.3 Streaming and packaging

2.3.1 Transport containers

In order to stream a file, it is necessary to package it in a way so that we do not need to transmit the whole file before it is possible to start playback at the client. The common way to solve this, is to encapsulate the different media streams in a so called container. The container will then contain the data needed for ensuring that the data is received in the correct order, while the inner data will contain all the media that we would like to transport. The way this has been solved is to use a container that encapsulates the audio and video we want to transmit. Some containers like MPEG-TS [12] , OGG [18] have been created for transfer of specific formats, while others like Matroska [14] ,and asf [13] are able to carry most known formats.

2.3.2 ASF

ASF, Advanced Systems Format, is a video container format developed by Microsoft that is especially designed for streaming of multimedia content. The container will be able to carry most media formats,

2.3.3 MPEG

2.3.3.1 MPEG-TS

MPEG-TS, or MPEG-Transport stream, is a protocol defining how to stream media over an unreliable connection. It has been designed to be used in environments where there is a chance of data loss or desynchronization. It demands that the streamed content is divided into several small packages (188 bytes) before transmitting, and it is therefore possible to conceal small packet loss if the receiving software has been designed to do so.

2.3.3.2 MPEG-4 part 10

MPEG-4 part 10, or H.264 as it is usually called, is one of the newest video compression codecs released from the MPEG and VCEG. It supports HD video, and will usually yield a much better compression rate compared to MPEG-2. H.264 is currently being used as the preferred video format in Blu-ray, HD-DVD and some of the new digital terrestrial networks, like the one in Norway[††]. [11]

2.3.4 Matroska

Matroska is a container format that is licensed under the GPL license. It supports several audio and video formats, like H.264, AVI, AC3, AAC, Vorbis, MP3, MP2 and DTS. It also

[††]http://www.ntv.no/

has support for several audio and video tracks, menu systems, tagging of some or several tracks inside the container, as well as support for closed caption. Although no DRM or other encryption schemes have been designed specifically for this format, the developers state that it is possible to add several layers of protection to the media files [‡‡].

2.4 Multiplexing methods

2.4.1 Files on server that are multiplexed in advance

The usual way of creating, securing and delivering content, is to compress audio and video one or more times. The multiple passes will be done in order to achieve higher compression rates. The compressed audio and video streams will then be multiplexed into a file using a container format like AVI or MKV. This way, we will end up with one file containing the movie, audio and , if desired, the closed caption. This approach is quite easy, and will enable the option of doing a large amount of pre-processing as the encoding and adding of protection is not time critical. It will on the other hand require that the content creator has all data available at the time of creation, and that no more data will be added later. We will also have the problem of being unable to distribute different versions of a movie without creating several almost identical copies. A third problem that has to be handled, is that we could be interested in distributing several versions of a media file. This method will then require that we create one copy of each file we want to distribute before they are placed on the web server.

Advantages:

- Easy to create and distribute as it is only one file
- It is possible to do the DRM encoding only done once

Disadvantages:

- No choice of different media qualities for the user unless we are to create several files containing almost the same material.
- Waste of human and machine resources as we must create several versions of each media clip
- Difficult to update content as everything must be reencoded , protected and packaged in order to do changes to audio, video or closed caption.

2.4.2 Multiple sources. Assembly done by the server

This solution will use several source files, but they will be multiplexed into one file stream before they are delivered onto the internet. This will require the use of a fast DRM encoder,

[‡‡] http://www.matroska.org/technical/specs/notes.html#Encryption

and will require that the original content is available at the server. We will also be required to store the private keys used to encrypt the files on the server, thus creating a large security risk. By doing this, it could be possible to add DRM to the multiplexed file, or to add it to the separate streams before they are multiplexed into one file stream.

Advantages:

- The user will be free to select the media that should be played
- Easy to add/remove parts of the media that are available to the customer
- Synchronization between streams are done by the server
- Possible to add DRM to the stream before or while transmitting

Disadvantages:

- The streams will probably have to be restarted if one media is changed.
- The server will have to do a lot of work in order to assemble and distribute the files.
- Keys for encryption must be kept available to the servers.
- Original media must be stored on each server in order to ensure maximum speed when assembling and encrypting the stream.

2.4.3 Separate files assembled on the client

While creating one file containing the relevant data is easy, we will loose a lot of flexibility. A way of handling this could be to separate the audio and video track into different files. We will then be able to deliver a custom "package" containing the relevant audio and video track. Another advantage of doing this, is that it is possible to change the audio or video track while playing. We will however encounter some problems when using this method.
The main problem this method has, is that if one of the streams are changed or experiences a buffer underrun, we will have a unsynchronized playback. This will have to be handled in the client software, and it is therefore necessary for the client software to check that the streams are synchronized all the time. This will require that it is possible to extract exact positions from the media player, and that the client software is able to synchronize the streams several times per second without choking the performance of the player.
Another problem that we need to handle, is the DRM protection. If we want to add encryption to the content, like windows media protection, we must do this on a per file basis. After adding this protection, the user will have to receive a key for each piece of content unless we use the same key on all media from the same source. This can make it possible to create a system that enables the user to pay different depending on the quality or parts of the movie they want to access.

Advantages:

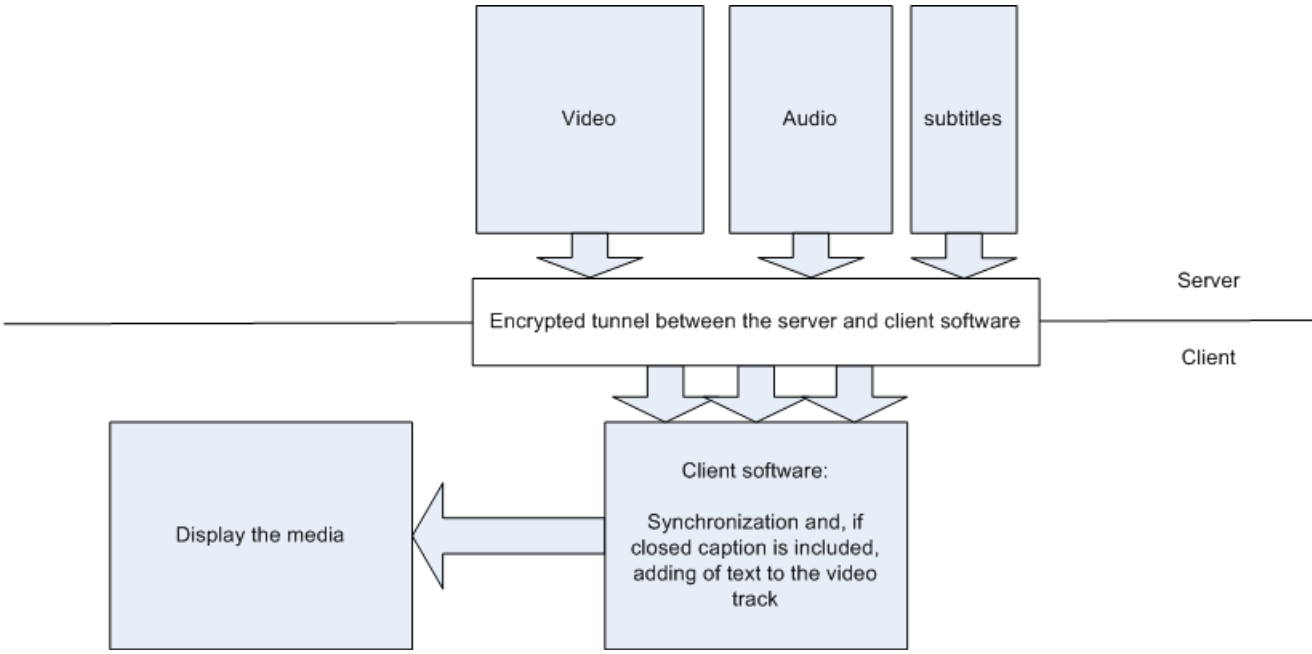

Figure 2.2: Model of a system where the files are delivered as separate streams to the client

- The user will be free to select the media that should be played
- Easy to add/remove parts of the media that are available to the customer

Disadvantages:

- Difficult to maintain synchronization between the streams if errors occur
- Will probably require several keys in order to unlock the content
- Larger overhead
- Needs to install software at the client

2.4.4 Multiple sources. Assembly done by the server and client

The previous schemes we have discussed will lay the burden of assembly on the server, or on the client, but by sharing the burden, we should be able to create quite flexible solutions. The principle here is a solution based on the watermarking scheme. When a client requests information from the server, it will be presented with two, or more media streams. It will then have to handle the synchronization and presentation of this content. The server will be required to create a unique stream of data for the customer. It could either watermark the whole video track in order to prevent any copies from being made, or just parts of it, thus

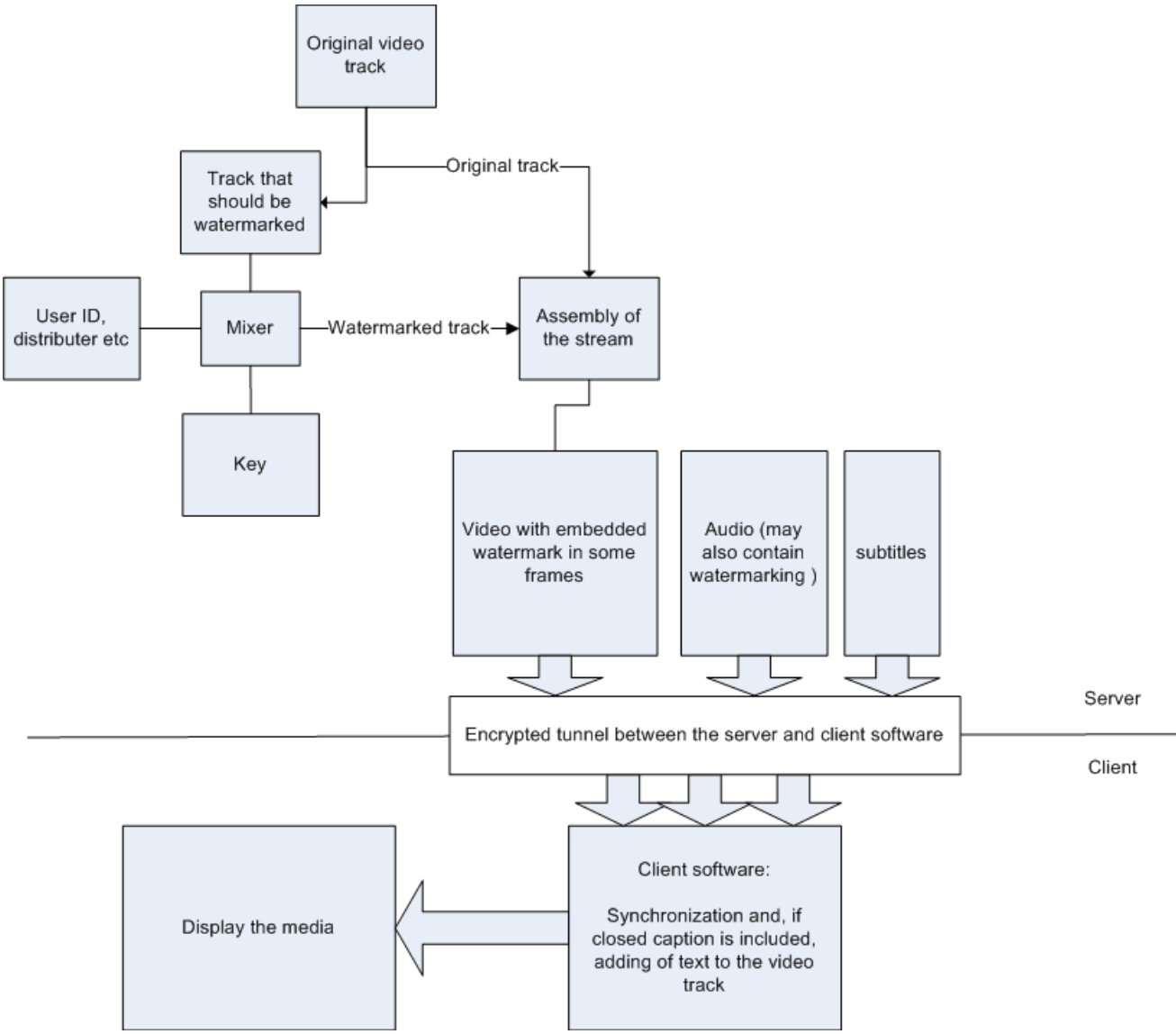

Figure 2.3: DRM watermarking with the different streams delivered separately to the client

generating a lighter load on the system, while maintaining the option of tracking down users that distribute the movie illegally. If it is required we could also be interested in protecting the audio track. This could also be protected by using watermarks, but it should be enough to protect the video track. After the server has assembled the different media streams, it will transmit them to the end user. In order to protect the streams from eavesdroppers that could distribute a stream "on behalf" of the intended user, it should employ some kind of encrypted end to end transportation mechanisms, like SSL.

2.5 Currently used solutions

Currently most sites providing Video on Demand, or more specifically those who provide protected content, like movies, use Windows Media DRM, and sometimes Adobe Flash. Windows Media DRM is a technology that is often used to protect copyrighted content like movies. Adobe Flash is used for the site itself. The video will then be played in a window on the web page and can, if the supplier of the content allows it, be set to full screen. Unfortunately most of these sites usually only support Internet Explorer. A few of them have support for other browsers, like Firefox, but the use of alternative browsers are usually at the users own risk. Support for Mac is relatively rare, but on occasion you will find some sites that support Mac.

As more people are starting to use alternative web browsers and operating systems, we see that there is an increasing demand for content providers that support the systems that these users have chosen to use. Unfortunately Windows Media DRM is not an open system, thus making it difficult to use for users outside of the Microsoft software ecosystem. Flash, on the other hand, have been ported to several operating systems, like Windows, Mac OS X and Linux. Flash DRM Technology, like FMRMS has been developed lately, but it's new at the time of this writing so it's not yet in use in any major video on demand sites. Since there hasn't been any active solutions for Flash DRM until now Windows Media DRM is dominating this market.

For streaming of unprotected content, both Windows Media Video (WMV) and Adobe Flash Video (FLV) are used, as well as other players like RealPlayer, DivX Web Player and VLC. Most of these exists on more platforms than the Microsoft Windows platform, and will therefore be able to target a broader audience than the Windows Media DRM allows.

Chapter 3

Technology Survey: Comparison and Test setup

In this chapter we will be going more in depth with the technologies we need for our Master thesis. We will review all the information needed for our implementation and testing. There will also be a comparison between different platforms for use by Filmarkivet AS.

3.1 DRM Performance and Feature Test Setup

3.1.1 Setup

The testing will be performed with two hosts. One will be acting as a server while the other will act as a client. For testing with encrypted files, we may have to use a third server in order to get hold of the licenses required to playback the content. This data will not be recorded, as the amount of data sent in order to acquire the license will be in a size that, compared to the movie, will be so small that it will not have an impact on the large amount of data that will be transmitting.
For Adobe Flash DRM we are testing Flash Media Server 3 with the different versions of the RTMP protocol, namely RTMP (Real-Time Messaging Protocol) and RTMPE (Encrypted Real-Time Messaging Protocol). We will test both CPU and bandwidth usage to determine what kind of performance hit there is when using the encrypted version of the RTMP protocol. Although there is another, more traditional form of DRM available from Adobe, FMRMS. This was released not long ago, and would require contacting Adobe to acquire a trial version. Because it was released when we were working on the Master thesis it was not be possible to test this. Flash Media Server on the other hand is available as a free trial for use with a very limited amount of concurrent streams.

3.1.2 Test criteria

The goal will be to test how easy it is for the customer and distributor to create and stream the desired media. We will therefore need to look into the following areas:

- Content creation
- Ease of deployment
- Changeability
- Added Overhead
- Resource usage

3.1.2.1 Content Creation

The content that is going to be streamed to the end users will be created by the company selling the solutions. It is therefore of great value to find a solution that will make sure that it is easy and quick to create the content and to add the required protection scheme.

3.1.2.2 Ease of deployment

As a solution that is difficult to deploy will hamper the people using the solution, a deployment scheme that is relatively simple to use should be found. This must include the process of moving a media file into the distribution system, and to register it in the license management solution. We will need to have a look at the authentication processes, and the different schemes that are supported for the solution that we want to use, and how difficult it is to implement. This will be important in order to be able to offer different solutions, and subscriptions to the customers, as well as to the how flexible it is in regards to the type of back end software the solution will require.

3.1.2.3 Changeability

As the ease of replacing and modifying the content is one of the things we want to investigate, we will discuss how difficult it is to do this for the different scenarios we have suggested.

3.1.2.4 Added Overhead

Depending on the way we are securing the content from being copied, there is a possibility of adding some overhead to the data that will be sent from the server to the client. When streaming the media, there could be an added overhead depending on the method we are using to transport the data.

3.1.2.5 Resource usage

Each solution should also be compared to the amount of resources it requires for storing and distributing the media. This will include disk space, network bandwith and CPU usage for the server and client.

3.2 Watermarking over a secure connection

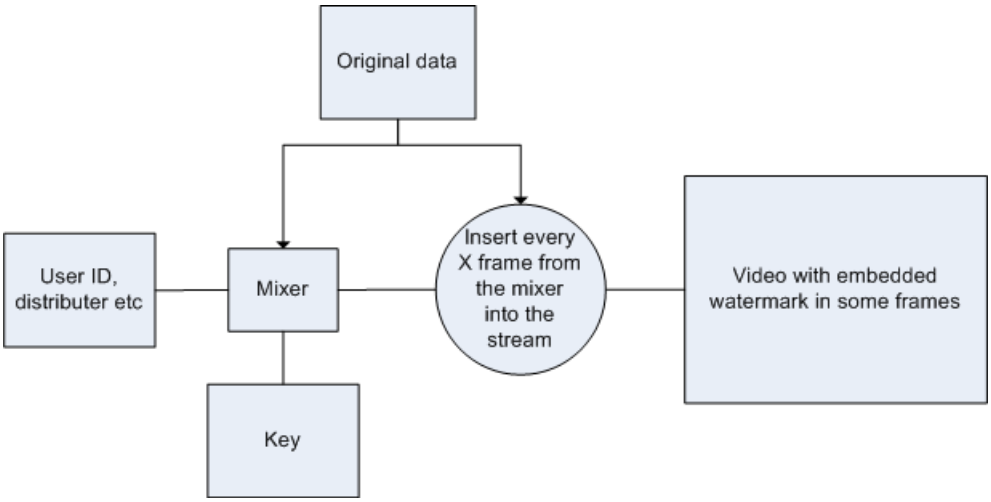

Figure 3.1: Model of a system where only parts of the media are watermarked

3.2.1 Watermarking by decoding and reencoding

Adding a watermark by decompressing and re-encoding the video stream is possible, but will have one large drawback. As video processing is a CPU intensive operation, we will need to pre process the videos before delivering them. Alternatively, we will need to have really fast computers performing watermarking as the content is streamed to the user.

3.2.2 Watermarking without fully decompressing the video stream

Recent research has shown that it is possible to add a watermark to a high definition video stream without fully decompressing it. This will be a great advantage, as we will be able to perform the watermarking much more efficiently than using a scheme where we have to decode and reencode the video stream. This is usually done by taking advantage of format and compression specific properties. This has lead to proposals for utilizing DTS*[23] or DEW† in order to perform on demand watermarking.

*Discrete Cosine Transform
†Differential Energy Watermarks

3.3 Choice of platform

In the last chapter we outlined two technologies for streaming video over the Internet, Microsoft's Silverlight and different Adobe Flash Technologies. These are the most suited technologies for our employer, and we need to compare them to determine which one of these would be best to build their new video on demand platform. The choice of these technologies are closely related to the DRM technology to be used, since Silverlight and Adobe Flash uses separate methods for this.

3.3.1 Core differences

There are a couple of core differences between Silverlight and Adobe Flash technologies. Silverlight and Adobe Flash are browser plugins, while Adobe Media Player and Adobe AIR are desktop applications and not to be used in a browser. This is definitely the biggest difference, since Silverlight is basically a contender to Adobe Flash, while Adobe Media Player is aiming more towards competing against Windows Media Player.

Media files in Silverlight are encoded with Microsoft's WMA, VC-1 and WMV79 format as well as MP3, while Adobe Flash are encoded as a variant of H.264, called Sorenson Spark[‡] or VP6[§].

Silverlight uses the XAML format, created in Microsoft Expression or another tool to store user interface and design, while AMP stores both the content, design and interface in an RSS feed, that you can edit in any text editor. Adobe Flash uses the tool by the same name for all content.

These technologies also use their own versions of DRM. Microsoft uses their new version of Windows Media DRM, while Adobe has created new technology, namely Flash Media Server and Flash Media Rights Management Server.

3.3.2 Requirements

- Meet movie company standards (DRM)
- Cross-platform
- Ease of use

There are certain requirements placed on the movie distributors from movie companies such as Universal, Warner Bros etc. These must be fulfilled for them to be allowed to host their movies for rental or sale. Requirements are placed on how the customers are allowed to access the movies, and the settings on the DRM technology they are using. This is the most important requirement because they can't run their service without a secure DRM solution.

[‡]http://www.adobe.com/macromedia/proom/pr/2002/flash_mx_video.html
[§]http://www.on2.com/index.php?313

The technology must be cross-platform to allow for the broadest customer base. It must be able to run on Windows, Mac and preferably Linux. It must be able to support major web browsers like Internet Explorer, Firefox and Safari. Support for more browsers is a plus.

The player must be easy to use, for both the viewer, developers and designers.

3.3.3 Fulfillment of Requirements

3.3.3.1 DRM Technology

Silverlight 2 will use a DRM called Microsoft PlayReady, which is based on Windows Media DRM, that Filmarkivet currently uses. Adobe Flash technology uses either Flash Media Server or Flash Media Rights Management Server.

Windows Media DRM has been used for a long time in the industry already, and since PlayReady is backwards compatible, there should be no problems with this new version. Adobe's Flash DRM is new on the market and has yet to be used by any major movie companies. We have to be sure that they meet the standards that are required, or Filmarkivet won't be able to use this technology.

3.3.3.2 Cross-platform

Silverlight 1 supports Internet Explorer, Firefox and Safari on Windows and Mac. They have announced a cooperation with Novell to support Linux in Silverlight 2. Opera isn't working in Silverlight 1 or Silverlight 2 beta, and it remains to be seen if this will be added later.

Flash 9 supports Internet Explorer, Firefox, Opera and Safari on Windows, Mac and Linux. Adobe Media Player supports Windows and Mac. Adobe has stated that it will include Linux support at a later date.

Flash 9 has the best cross-platform support, while the others are lacking Linux support at present. Linux support has been suggested for both Silverlight and AMP, but if and when this will be done are still unknown.

3.3.3.3 Ease of use

Silverlight is a new technology with support for the .NET platform as well as a newer version of the established Windows Media DRM. It has tools such as the new Microsoft Expression and Microsoft Visual Studio for programming.

Adobe Flash 9 has an own editor for creating content, the current version called Adobe Flash CS3. Adobe AIR and Adobe Media Player require Adobe Flash and an RSS editor, respectively. Something that sets it apart from the other technologies is that they require an installation that runs separately from the Internet browser. This might not be so convenient for people who don't want to install software outside of their browser and want to keep it simple and accessible. In Adobe Media Player you have one player that you use for all your RSS feeds, while in Adobe AIR you need to install a desktop application for different services.

3.4 Multiplexing Test Setup

For the multiplexing test, we created a new set of media files. The movie file is 87 MB in size, and is encoded with H.264[11]. The audio file is 13 MB in size, and is using DTS [7] encoding. Both files where then packaged inside a matroska container[14] using MKVMerge from the MKVToolnix[16] software. We also made a file containing the video and audio using this tool (100 MB).
For the streaming, we used Apache 2 as the server for delivering files. The client was then set up with Videolan[21] as the media player.

3.4.1 Files multiplexed in advance

When the server only has to deliver a file, the bottleneck will probably only be the connection and speed of the hard drive. This test will therefore be done in order to see what load a server would be experiencing under the solutions used by the windows media DRM.

3.4.2 Files assembled on the server

As the client will be able to select audio track, language, quality and such, it is necessary to assemble this into a package that are easier to handle than X different files that has to be transmitted in the correct order. In order to do this, we will be using MKVmerge, a windows and Linux tool that assembles several files into a MKV package. The resulting file will then be put on a temporary storage as it could be unique and personalized for every client.

3.4.3 Files assembled on the client

Here we will use the ability of the client to playback 2 or more streams at once. The server will only be responsible for serving content, and if we implement a watermarking scheme, it will have to do the watermarking to. The client will have to synchronize the streams and fetch the correct files.

Chapter 4

Implementation and Testing

4.1 DRM Performance

4.1.1 Impact of Windows Media DRM

The performance testing of Windows Media DRM has been done on an encrypted and unencrypted video stream (A.1, A.2). As we would expect, the encrypted data stream will require more cpu power in order to be played back. The amount of data will also be increased, but the amount of added data is only about 2 KB, and will therefore not be noticeable as the encrypted files will usually be in the size between 0.5 GB and 5 GB. The added data will probably be there due to the fields containing a file ID and a link to the place where the player should redirect the user in order to acquire the license for the media.

4.1.2 Impact of Flash DRM

Performance testing on Flash Media Server 3 has been done with a regular RTMP stream (B.1) and an RTMPE stream (B.2).

According to the tests RTMPE requires slightly more processing power than an RTMP stream. In these charts RTMPE takes an average of 0.43% more CPU power than RTMP. The difference would have been larger on a slower system, the test system statistics are shown in Appendix B. When reversing the roles of our server and client, i.e. using a Quad Core processor as the client the difference between RTMP and RTMPE are hardly noticable, so the RTMPE protocol has virtually no performance hit on a high-end CPU.

Bandwidth testing on Flash Media Server 3 has been done with a regular RTMP stream (B.3) and an RTMPE stream (B.4).

The bandwidth figures show that RTMPE requires a bit more bandwidth, in the figures

provided about 3.2kbps more than RTMP. After getting the average of several tests, the difference is slightly lower, at about 1.32kbps average difference after 4 tests. These tests were used on a Flash Video file with video bitrate at 700kbps and audio bitrate at 128kbps.

4.2 Multiplexing data and its influence on performance

In this section we will have a look at how the proposed schemes of multiplexing works. We will use the files that are pre processed as the "best" solution, and compare the other solutions to this. The CPU and bandwidth usage results from this tests are found in appendix C.

4.2.1 Files assembled on the server

The goal here is to create a custom package that will be delivered to the user. In order to create this package, we used MVKmerge to assemble the files into one package. We then ended up with a package containing only the audio and video track that we wanted. The results of doing this was that we had an optimized package, containing as little information as possible. This will also lead to the existence of several versions of the same media if we were to distribute several versions with almost the same content.

4.2.2 Files multiplexed in advance

As these files have already been processed, we will use this as the ideal condition for our tests. The common property for these files is that they contain all the data that we want to move to the client. Since we will not want to add or remove data from these files, we will be able to use a multi pass compression that will ensure that the files are as small as possible. Unfortunately, this will not be feasible to do on demand due to the long processing time and amount of CPU power required. The bandwidth used for this configuration will be the same as in the previous section, Files assembled on the server (4.2.1), but will require a little less CPU power from the server. This is due to the fact that we are only going to transmit a file without doing any pre processing, such as assembling the files the first time the client requests it.

4.2.3 Files assembled on the client

In order to maintain maximum flexibility, and ease of changing the content, we could attempt to deliver the content as several separate streams. The testing with 2 separate streams were done using Videolan on the client, and Apache 2 on the server side. The test where done in a environment that minimized the chance of desynchronization of the streams. We have therefore not implemented a system for handling synchronization. The synchronization part would add a little extra overhead to the client CPU usage as it will need to check if the streams are synchronized properly. We could also see a little more network traffic, but this

will depend on how often a desynchronization occurs as well as the caching method we are using.

4.2.4 Watermarking

Due to few free implementations of a watermarking scheme, and the fact that we have no expertise when it comes to video processing, we had to use existing tools in order to do the performance testing of the watermarking scheme. As the computation and insertion of the watermark should take a lot less time than the decoding and reencoding of the video stream, we decided to do the test with a static watermark. The test where performed using FFmpeg* to place a image over the original video, as this will give us a indication of the speed we can achieve if we are to do on demand watermarking on the server by fully decompressing and compressing the video. According to the results from the experiments[27] done by G.C. Langelaar, we should be able to add a DEW watermark† to a video more than 50 times faster than the time decompression and reencoding takes‡.

*ffmpeg.mplayerhq.hu
†Differential Energy Watermarks
‡see Figure 6.4.1

Chapter 5

Discussion

5.1 DRM

5.1.1 Methods for internationalization

According to the data we have collected from the multiplexing tests, it is clear that the data should be pre processed before delivering it to the client if we have little bandwidth available. We would however recommend that all tracks, video, audio and subtitles, should be saved as separate tracks.

Having separate tracks will ensure that international users can get an audio track and subtitles in their native language, or any other language that they prefer. They can also get a choice on what kind of quality they want depending on their connection. This will also enable a distribution model where the customer could be charged differently depending on the features they want. This will probably require that the stream is saved to RAM, or to disk. Depending on the system, it should then be possible to add encryption to the stream, or to watermark it, thus hopefully preventing unauthorized distribution. The fact that pre multiplexing a HD stream will reduce the processor usage by 2 % on a quad 2,6 GHz processor, also means that more customers will be able to use the content without having to upgrade their equipment.

5.1.2 Watermarking

Since we would like to reduce the load on the server in order to be able to deliver video on demand to a large amount of customers simultaneously, we need to speed up the process of watermarking. Our recommendation for this will be to add the watermark to parts of the movie. By only processing small parts, and adding watermarks onto this, we should be able to handle fairly large amounts of data at the same time. By replacing a few frames of

the video now and then, we should be able to identify users that have uploaded the movie without permission, but we could also lose the ability to detect those who only upload a small part of the content. We could also try to use the DEW watermarking of the video, but this will require that we are using H.264 streams. It will therefore make it less versatile than the approach where we decode and reencode the video. Another option is to embed the watermark in the audio signal, but as this does not protect the visual content in any way, it will probably not be accepted as a good enough solution for the movie companies.
As this is a drm scheme created on the basis of making the user act legitimate of his own free will, as opposed to schemes that are restricting the access rights, this solution should be easier to use for most people, and will also be platform independent.

5.1.2.1 Suggestions for increasing the security

Since a attacker will be able to frame a person for distributing the data if he is able to intercept the communication between the seller and customer, we need a way of making sure that nobody are able to steal the data while it is being transmitted. Since the content could be rented from an online store we should also try to protect it from the user. As we have previously stated, most DRM systems will be possible to bypass with enough time and effort. We would therefore like to have a system that removes the easiest ways of attacking the content. By creating a player that is embedded into software that runs in the web browser we should be able to create a secured connection from the application to the streaming host. This will reduce the risk of people being able to read the data that is transmitted, and it will also be possible to hide communication between the server and client software. Another measure that we could implement are hidden URLs. In order to make sure that nobody are able to use a download software to download the files, we will recommend hiding the URL inside the application, and if possible, generate a new URL each time a customer accesses a file. By doing this, and requesting a new URL from the server each time, we will also be able to prevent people from reading the URL and using it later in order to be able to replay the files in another player.

5.1.3 Performance

In our tests Windows Media DRM added a $\sim$1.6% increase in CPU usage compared to an un-encrypted file. This is a bigger increase than FMS introduces, although Windows Media DRM is a traditional DRM and encrypts the media itself, instead of the stream. The small increase in CPU usage has no noticeable effect on our test systems, and won't require enough resources to a be a problem for low-end systems. Windows DRM has been in use by many video on demand services over the world, and as far as we have heard this has not limited the customer base due to demands to the processor.

Our test system for Flash Media Server is listed in Appendix B. Our CPU usage and Bandwidth tests show that Adobe's RTMPE protocol has very little performance hit (~0.43%) on our systems. Reversing the server and client i.e. using a Quad Core processor as a client showed us that the RTMPE protocol really doesn't require much processing power on high-end systems, or on our server system. This means that it won't require much resources on mid-end systems, and lower-end computers won't have trouble running it, so the technology can be used by all customers.

Bandwidth-wise RTMPE only took about 1.32kbps more on average than the un-encrypted RTMP protocol, so with todays internet connections this should have no noticeable effect on performance.

5.1.4 Ease of deployment

For companies with video on demand services that already use Windows Media DRM the transition between their older WMDRM and the new PlayReady technology that will be released with the next version of Silverlight is relatively easy, since it's backwards compatible with their existing encryption.

For companies that are changing from Windows Media DRM to Adobe's Flash Media Server or FMRMS they will have to convert their media into compatible formats like FLV, since these products doesn't support Microsoft's formats nor can they decode media files encrypted with the Windows Media DRM protection.

For companies that already have a large library of media files based on one DRM system the transition will require a large amount of resources, since they will have to convert all their files to a new format. This means that they will have to consider the pros and cons of changing their DRM technology and platform with this in mind.

5.2 Choice of platform

5.2.1 Cross-platform

Cross-platform support in Silverlight is limited to Windows and Mac at the moment, and browsers such as Opera isn't supported either. Support for Opera and more browsers might be added to Silverlight in the future. Microsoft has stated that they are cooperating with Novell to create a Linux version, but they haven't given any info about when this will happen. Although Linux has very small user-base compared to Microsoft Windows and Mac, it's important to reach as many customers as possible. Linux developers are constantly making their distributions more user-friendly, so more people may be able to use it in the future.

Adobe Media Player is an off-line player that supports Windows and Mac in its current version. As with Silverlight it remains to be seen if it will support Linux, but since Adobe already has support for Linux with Flash 9 this might not be so improbable.

Flash 9 supports Windows, Mac and Linux, so it definitely has the best cross-platform support. This is a big plus for Flash, enabling users with any operating system to use applications and websites made with this technology. If cross-platform is the highest priority, then this is a good choice. The other platform technologies might get better support in the future, but only Flash 9 has this at the moment.

5.2.2 Ease of use

Flash 9 and Silverlight requires less effort from the users point-of-view than Adobe Media Player or Adobe AIR, since it's a browser plugin and you don't have to install separate software. You only have to install the plugin, and then you can view the video in your browser. With AMP you will have to run the player separately, like Windows Media Player, or Quicktime. With Adobe AIR and you will have to install an application for each product. Say you have a video on demand application, and then another company has a game. The user will then have two applications just like any standard desktop application.

This might have an effect on how the customers perceive the product and ultimately the product they chose. Although some customers, say more experienced computer users might not care much if they have to run a separate application instead of just their web browser, this does not apply to everyone.

From a developer's point-of-view the difference lies more in the tools available than features themselves, since the base features are pretty similar. Adobe products use the Flash editor, and for AMP you mainly need an RSS editor. Microsoft Silverlight uses Microsoft Expression for design and Visual Studio .NET for development. Adobe Flash has been around for a long time on the internet, often used to create design and user-interface for websites, for games or animated videos, so it's well-tested for these purposes. You can use any .NET programming language to program in Silverlight, and these languages has been tested for both conventional and web programming. What tools you like the most will depend on the personal preference, but they both have the features necessary to make a video on demand site.

5.2.3 DRM Technology

The Windows Media DRM and Adobe DRM will protect the content from the customer, but they will employ different means of doing so. As the Windows Media DRM scheme will require that the files are pre encoded, we will probably get a lower CPU usage on the servers, but this will be on expense of the flexibility as we will need to decrypt and encrypt all the files if we are to change the certificate or distribution system. The Adobe DRM will encrypt content while it is delivered, thus making these changes easy to do, but as the tradeoff here is a increase in cpu use, the system will be less usable in a environment serving a lot of customers at the same time. A large problem for Adobe would be if their new DRM technology isn't approved by the content owners. Windows Media DRM has already been used extensively on

the internet, and are known to be accepted by the large media companies, and are therefore the "safe" choice of platform for now.
Adobe Flash has been used extensively on the Internet for many years now, in Youtube and many other web applications. That's something that Microsoft is trying to counter with their Silverlight technology. Many video on demand sites currently use Flash for the design and Windows Media with Windows Media DRM for the video itself. It remains to be seen what the trend will be in the future, depending on what level of protection Movie companies require. If they deem Flash Media Server as a valid solution for protecting media then Flash might quickly become popular, since they can get maximum cross-platform capability and only use one technology for the player.

5.3 Multiplexing results

	Pre multiplexing	Client multiplexing	on demand multiplexing
CPU use at the server	Low	Low	High when creating archive.
Bandwidth	Low	Medium	Low
Storage needed on server	High	Medium	Extreme
Flexibility	Low	High	High
Ease of replacing parts of the content	Difficult	Easy	Easy/medium

Table 5.1: Comparison of different multiplexing methods if we are to serve several different audio/video streams for the same content

As we can see in table 5.1, the way of deploying content that demands the least from the distributor, will be to use the existing methods of distribution. This has the disadvantage of low flexibility, and will therefore not be suitable when distributing to a multinational audience where it should be easy to add or remove information, such as subtitles or audio tracks, or if the content provider wants to offer the content in different quality. The 2 other methods we have outlined are more suitable for this since they are both able to handle change of content. They have, on the other hand, some disadvantages compared to the normal way of distributing content. Delivering 2 or more streams to a client will, as shown in appendix C.4, demand more CPU power, and require the client to be responsible for synchronization of the streams. It will also introduce more network overhead due to the fact that the streams will be completely independent of each other. This overhead could be quite large if we are

distributing HD video and audio, as seen in C.2.

Getting the server to create a container containing the information that the client has requested on demand, will remove the problems when it comes to synchronization and extra bandwidth, but we will introduce other problems. As long as we only want to distribute one audio and video track, this will have the same performance values as the scenario where the files where multiplexed in advance. The main difference between this method and the pre multiplexing, are that we will use a lot more storage space on the server as it, depending on the amount of tracks offered, will exist several almost identical copies on the server. Another disadvantage will be that we also require that the customer have to wait for the server to assemble the file, which could take several seconds to complete on a video of SDTV quality. This could certainly be addressed by utilizing caching techniques, but this would introduce problems keeping the pre processed media up to date when replacing content with newer versions. The server multiplexing could also be done by adapting the multiplexing software so that we are able to assemble only the part of the media that the clients are requesting. This would give the advantage of customers being able to select the exact content they want without using a lot of disk space on the server. This solution has some drawbacks. While the server is assembling the files on the fly, we will use more CPU capacity than we would use when streaming a file from a cache. We could also get problems if the user chose to jump to another position or needs to have parts of the content retransmitted. The largest problem is however that we probably will need to create tools that work with proprietary video formats or containers. Earlier attempts on doing this have been shut down and we have no reason to believe that companies like Microsoft will change their practice and allow these modifications.

Chapter 6

Summary and Conclusion

6.1 Conclusion

6.1.1 DRM

The DRM technologies we have tested have had very small performance hits on our test systems. Since the performance hit is so small, lower-end systems should have no problems running them either. Although the two DRM technologies we have tested show a little difference in the amount of CPU usage they need, this is minor ($\sim$1.2%). According to our tests there is no reason to choose one of the technologies over the other because of performance concerns.

Watermarking has the main advantage of letting the end user decide the software they want to use in order to play back the content, as well as making the playback completely independent of operating system. We will on the other hand not be able to perform any playback restrictions in order to stop the end user from saving the video stream and play it back as many times as they want. Watermarking is unfortunately not a solution we would recommend. As VOD sites will want to limit the time where a user should be able to use the content, watermarking will not be of any use. The process of watermarking media will also require a lot of CPU power, thus requiring more powerful servers. Another problem is that even if we are able to identify the user that has distributed a pice of content, the content will stay open on the internet. It is also possible to blame such activities on hackers and viruses in order to get away without getting blamed. The last problem with watermarks will be that they are only detectable to the company that inserted the watermarks, as a public method of identifying the watermarks will give everybody a way of detecting and destroying them.

6.1.2 Choice of platform

Since cross-platform is an important criteria to reach a broader market, Flash 9 is currently the best option for delivering the media, since AMP and Silverlight currently doesn't support Linux, and Silverlight (being a browser plugin) doesn't support browsers such as Opera. Since the last two are newer technologies, features might be added later in the development phase. Flash and Silverlight are run in the browser which, in our opinion makes it more practical for customers, as opposed to a separate application like Adobe Media Player. The biggest potential problem with using Flash 9 as a streaming platform would be DRM issues, since this will make it impossible to provide video on demand services. Whether or not this is a problem will be known as the technology is tested.

6.1.3 Method of content distribution

As long as it is possible, we would recommend that all the different media types are stored as single streams on the server. By assembling them on demand, it will be possible to give the customers a option of selecting the quality they want on the content. This will be highly relevant for supplying the customers with media that they are able to play on their system, or are able to receive due to bandwidth restrictions. This will also be possible to extend into a market where the customers are buying the movie and paying depending on the quality of the content and features they want. The largest drawback of this is that the more customers that want to use the service, the more disk space will be used. As disk space is extremely cheap at the moment, this should not be a big problem as long as we are restricting the time the copy is kept on the server.

6.2 Summary

In this project we have looked into different streaming technologies, transport methods and ways of securing the content. We have performed an analysis of the performance in different scenarios, and tested different ways of distributing the content. We have also looked into some of the newest player technologies, and found the reasons for using them, or to look for alternatives. We have found that the best way to deliver content is to use multiple streams and assembling them on the client. This will take more bandwidth, but with todays internet connections this will be acceptable. The solution that we have suggested will enable people to pay extra for quality, while it is also possible to do both watermarking and encryption on the output stream.

We have also had a look into watermarking as a DRM protection, but as this will only prevent the spread of content, it will not protect the media stream from being saved by the end users.

Our recommendation for Filmarkivet AS would be to keep their current DRM system, since they already have a large amount of content protected by Windows Media DRM.

Converting this content would take too many resources compared to the gain. The best option would be to develop their site using Microsoft Silverlight since they can use their existing DRM. To add internationalization to their site we recommend that they use separate video, audio and closed caption streams, since this will ensure maximum flexibility.

Bibliography

[1] Adobe air http://www.adobe.com/products/air/ last visit: 26.05.2008. [cited at p. 9]

[2] Adobe flash cs3 http://www.adobe.com/products/flash/ last visit: 26.05.2008. [cited at p. 16]

[3] Adobe flash http://www.adobe.com/products/flashplayer/ last visit: 26.05.2008. [cited at p. 13]

[4] Adobe media player http://www.adobe.com/products/mediaplayer/ last visit: 26.05.2008. [cited at p. 9]

[5] Advanced access content system http://www.aacsla.com/what/overview. [cited at p. 6]

[6] Amount of flash users http://www.adobe.com/products/player_census/flashplayer/ last visit: 26.05.2008. [cited at p. 16]

[7] Dts: http://www.dtsonline.com/consumer/technology/ last visit: 26.05.2008. [cited at p. 28]

[8] Flash media rights management server http://www.adobe.com/products/flashmediarightsmanagement/ last visit: 26.05.2008. [cited at p. 8]

[9] Flash media server http://www.adobe.com/products/flashmediaserver/ last visit: 26.05.2008. [cited at p. 8]

[10] Fms tech overview http://livedocs.adobe.com/flashmediaserver/3.0/docs/flashmediaserver_tech_overview.pdf last visit: 26.05.2008. [cited at p. 9]

[11] *H.264 FAQ http://www.apple.com/quicktime/technologies/h264/faq.html last visit: 26.05.2008.* [cited at p. 17, 28]

[12] http://www.chiariglione.org/mpeg/technologies/mp02-ts/index.htm last visit: 26.05.2008. [cited at p. 17]

[13] http://www.microsoft.com/asf/ last visit: 26.05.2008. [cited at p. 17]

[14] Matroska http://www.matroska.org/ last visit: 26.05.2008. [cited at p. 17, 28]

[15] Microsoft silverlight http://www.microsoft.com/silverlight last visit: 26.05.2008. [cited at p. 13]

[16] Mkvtoolnix. http://www.bunkus.org/videotools/mkvtoolnix/ last visit: 26.05.2008. [cited at p. 28]

[17] Norsk filminstitutt http://www.nfi.no/ last visit: 26.05.2008. [cited at p. 3]

[18] Ogg http://www.vorbis.com/ last visit: 26.05.2008. [cited at p. 17]

[19] Rss: http://iws.cit.cornell.edu/iws2/technology/techinfo.cfm last visit: 26.05.2008. [cited at p. 15]

[20] Silverlight2 roadmap: http://blogs.msdn.com/usisvde/archive/2007/05/23/silverlight-1-1-developer-roadmap.aspx last visit: 26.05.2008. [cited at p. 14]

[21] Videolan media player http://www.videolan.org/ last visit: 26.05.2008. [cited at p. 28]

[22] Windows media drm http://www.microsoft.com/windows/windowsmedia/forpros/drm/default.mspx last visit: 26.05.2008. [cited at p. 8]

[23] *Watermarking of uncompressed and compressed video*, volume 66. Elsevier, 1998. [cited at p. 25]

[24] I. Brown, C. Perkins, and J. Crowcroft. Watercasting: Distributed Watermarking of Multicast Media. *Proceedings of the First International Workshop on Networked Group Communication*, 300, 1999. [cited at p. 12]

[25] Gwenael DOERR. *Security Issue and Collusion Attacks in Video Watermarking*. PhD thesis, Universite de Nice Sophia-Antipolis, 2005. [cited at p. 12]

[26] M. Ji, SM Shen, W. Zeng, T. Senoh, T. Ueno, T. Aoki, Y. Hiroshi, and T. Kogure. MPEG-4 IPMP Extension for Interoperable Protection of Multimedia Content. *EURASIP Journal on Applied Signal Processing*, 2004(14):2201–2213, 2004. [cited at p. 6]

[27] G.C. Langelaar. *Real-Time Watermarking Techniques for Compressed Video Data*. Ph.d. thesis, Delft University of Technology, 155 p., ISBN: 90-901-3190-6, Delft, January 2000. [cited at p. 31]

[28] M. Noorkami. *Secure and Robust Compressed-Domain Video Watermarking for H. 264*. PhD thesis, 2007. [cited at p. 12]

[29] MD Swanson, B. Zhu, and AH Tewfik. Multiresolution scene-based video watermarking using perceptual models. *Selected Areas in Communications, IEEE Journal on*, 16(4):540–550, 1998. [cited at p. 12]

Appendices

Appendix A

Windows media DRM performance

Test configuration:

	Client
CPU	2.4 GHz Intel Quad
RAM	6 GB
Operating system	Windows 2008 x64
Network speed	10 Mbit

Table A.1: Client configuration for windows media testing

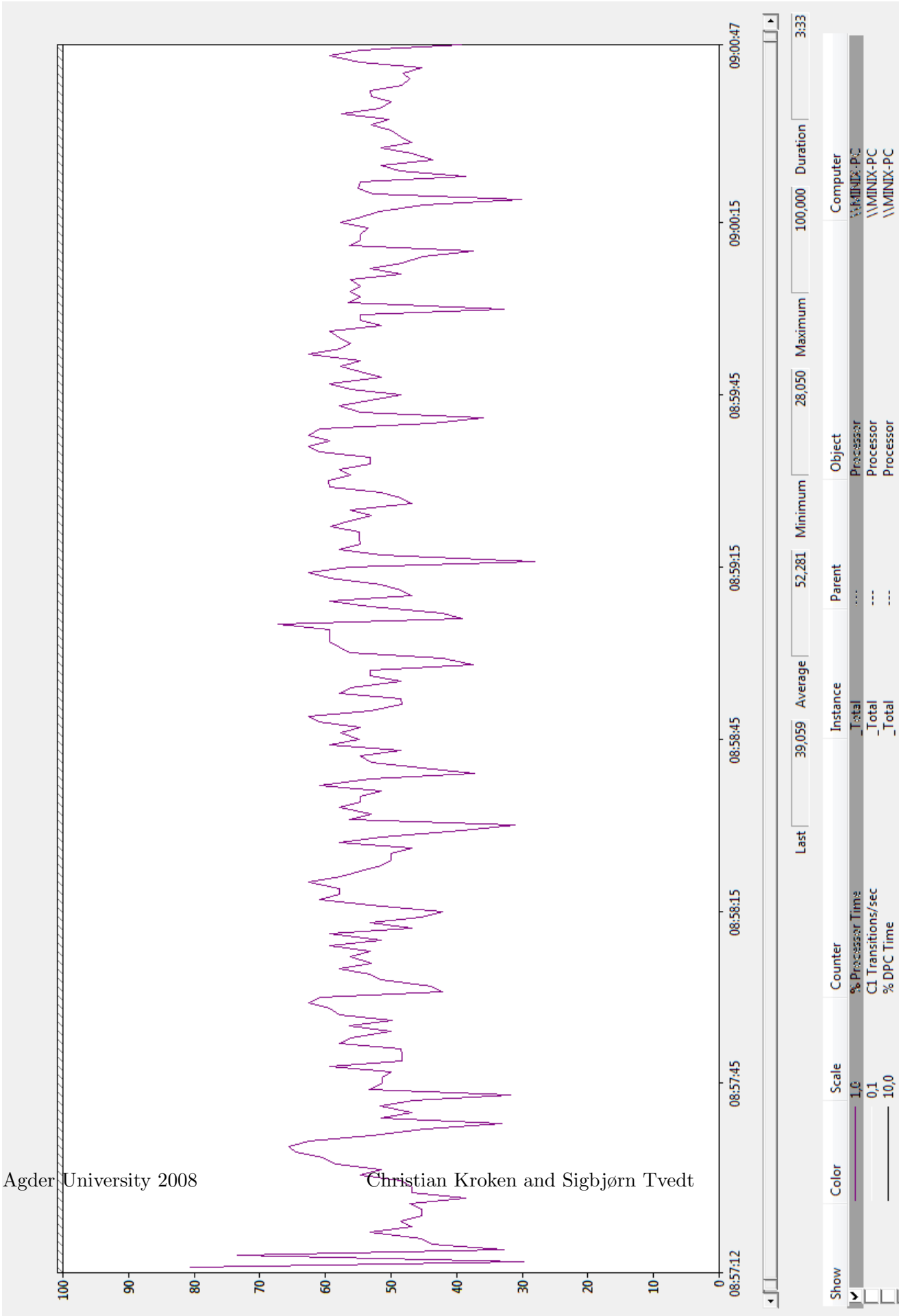

Figure A.1: CPU usage with Windows Media DRM enabled

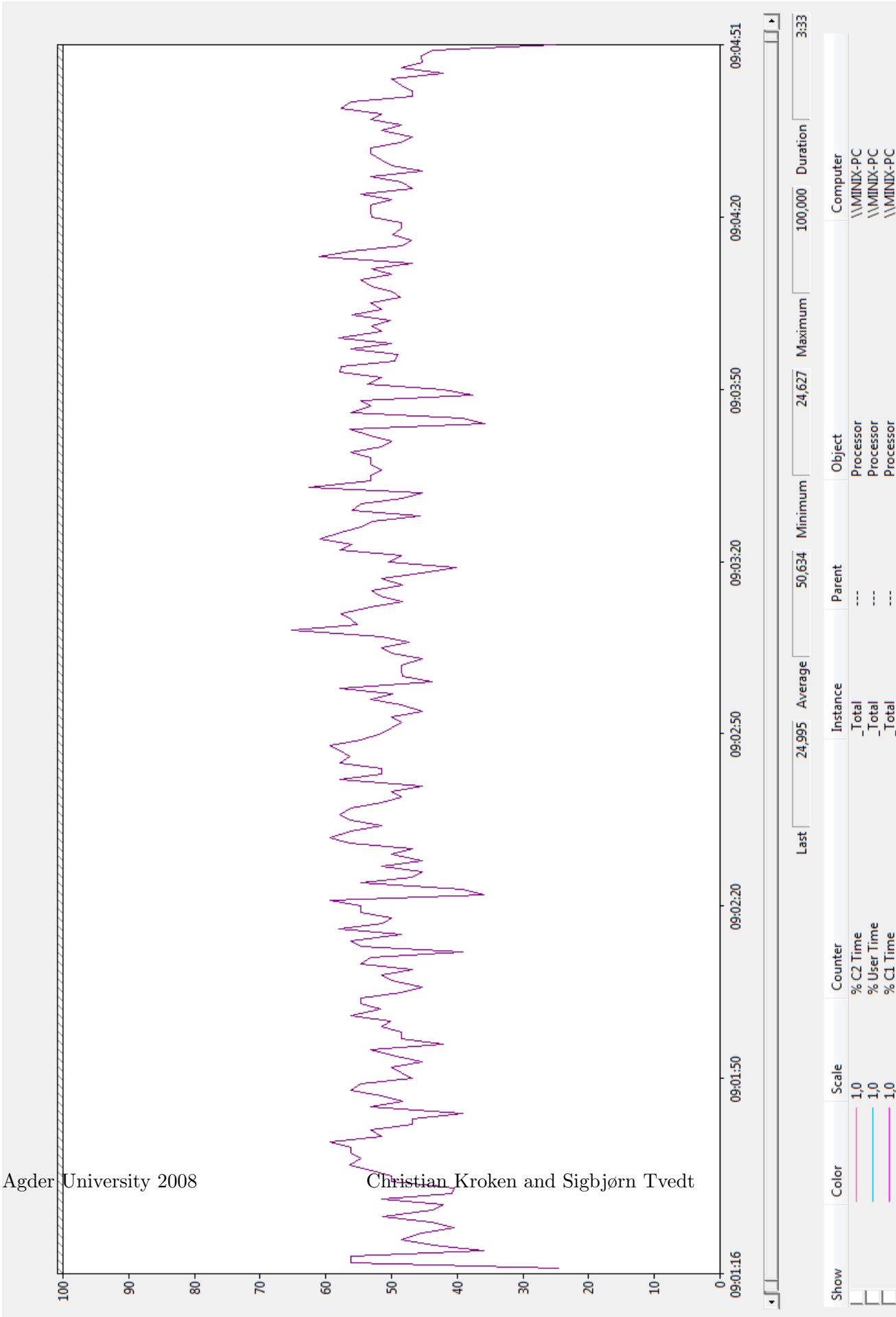

Figure A.2: CPU usage with Windows Media DRM disabled

Appendix B

Flash Media Server performance

Test configuration:

	Server	Client
CPU	2.7 GHz Intel Quad	1.83 GHz Core 2 Duo
RAM	2 GB	1 GB
Operating system	Windows Vista x64	Windows XP SP2
Network speed	1 Gbit	1 Gbit

Table B.1: Testing configuration for Flash Media Server

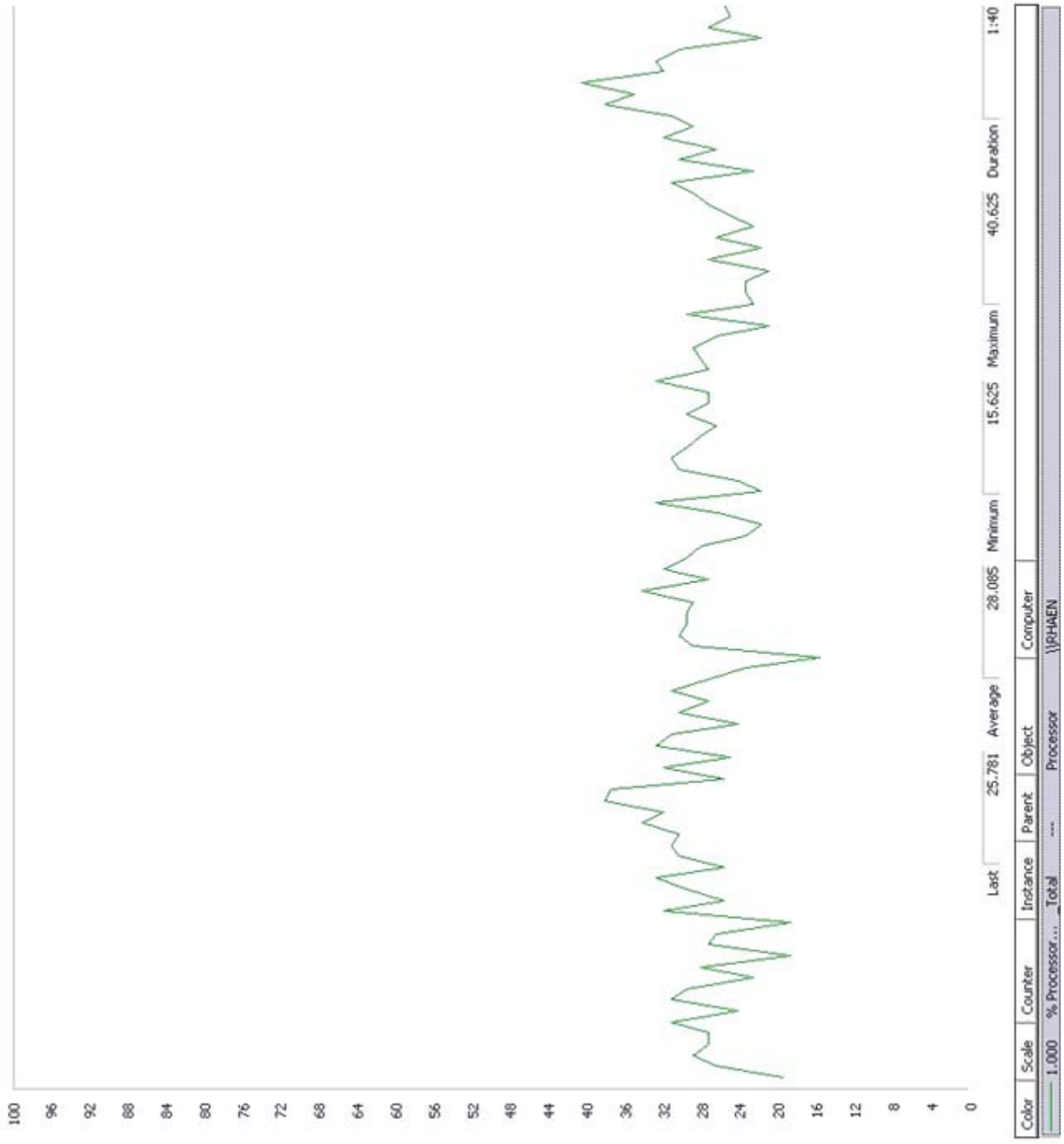

Figure B.1: CPU usage with RTMP

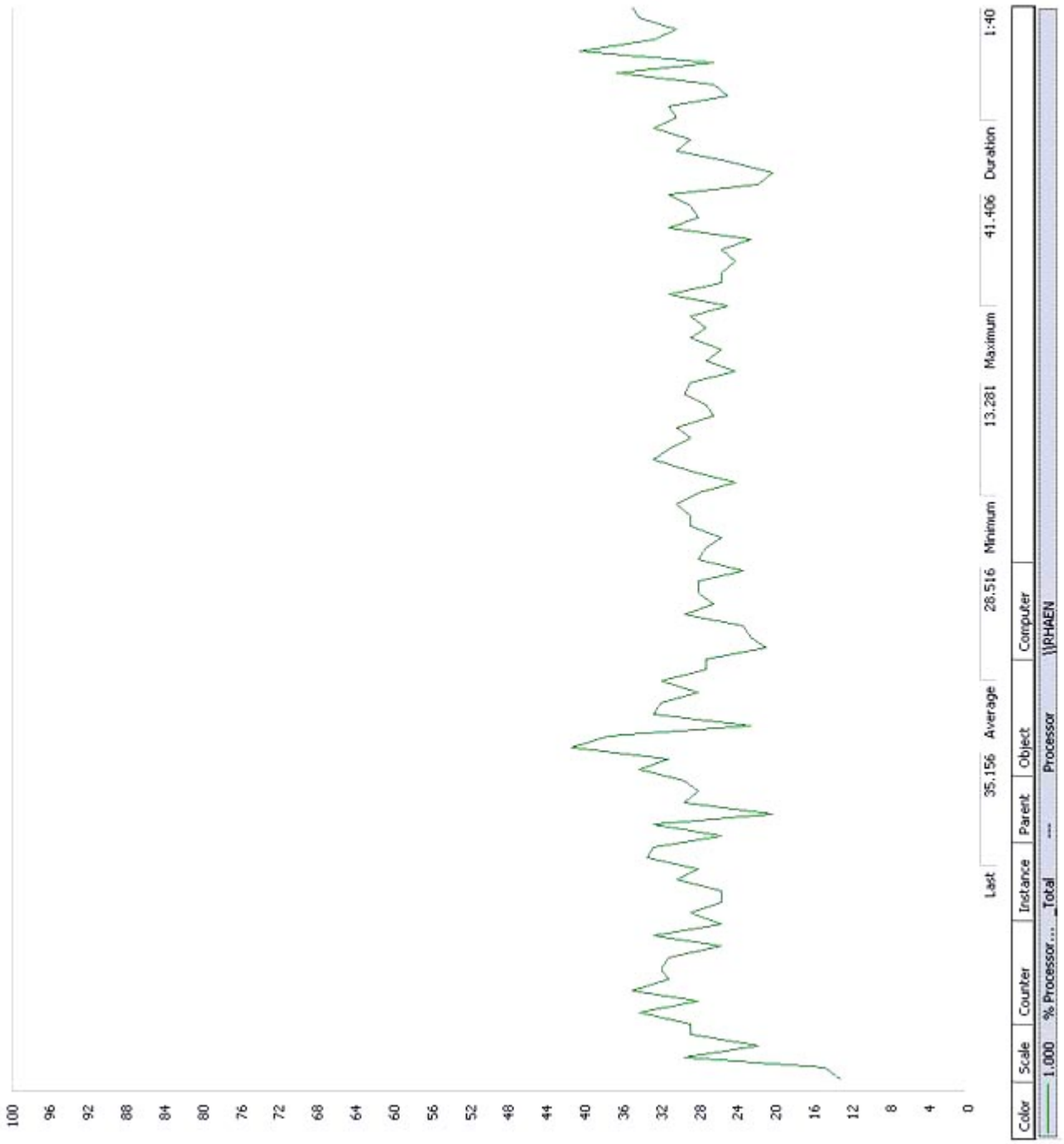

Figure B.2: CPU usage with RTMPE

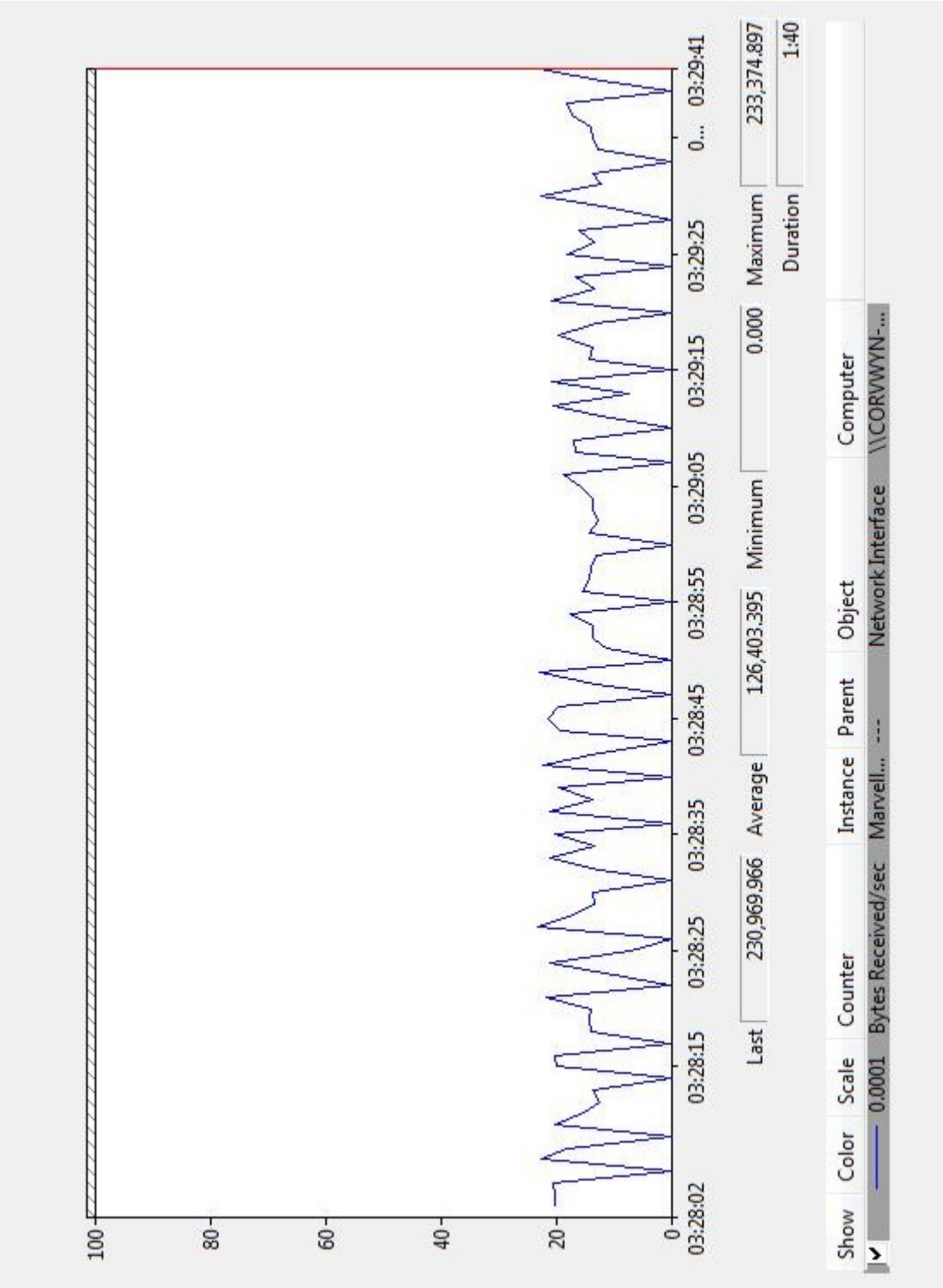

Figure B.3: Bandwidth usage with RTMP

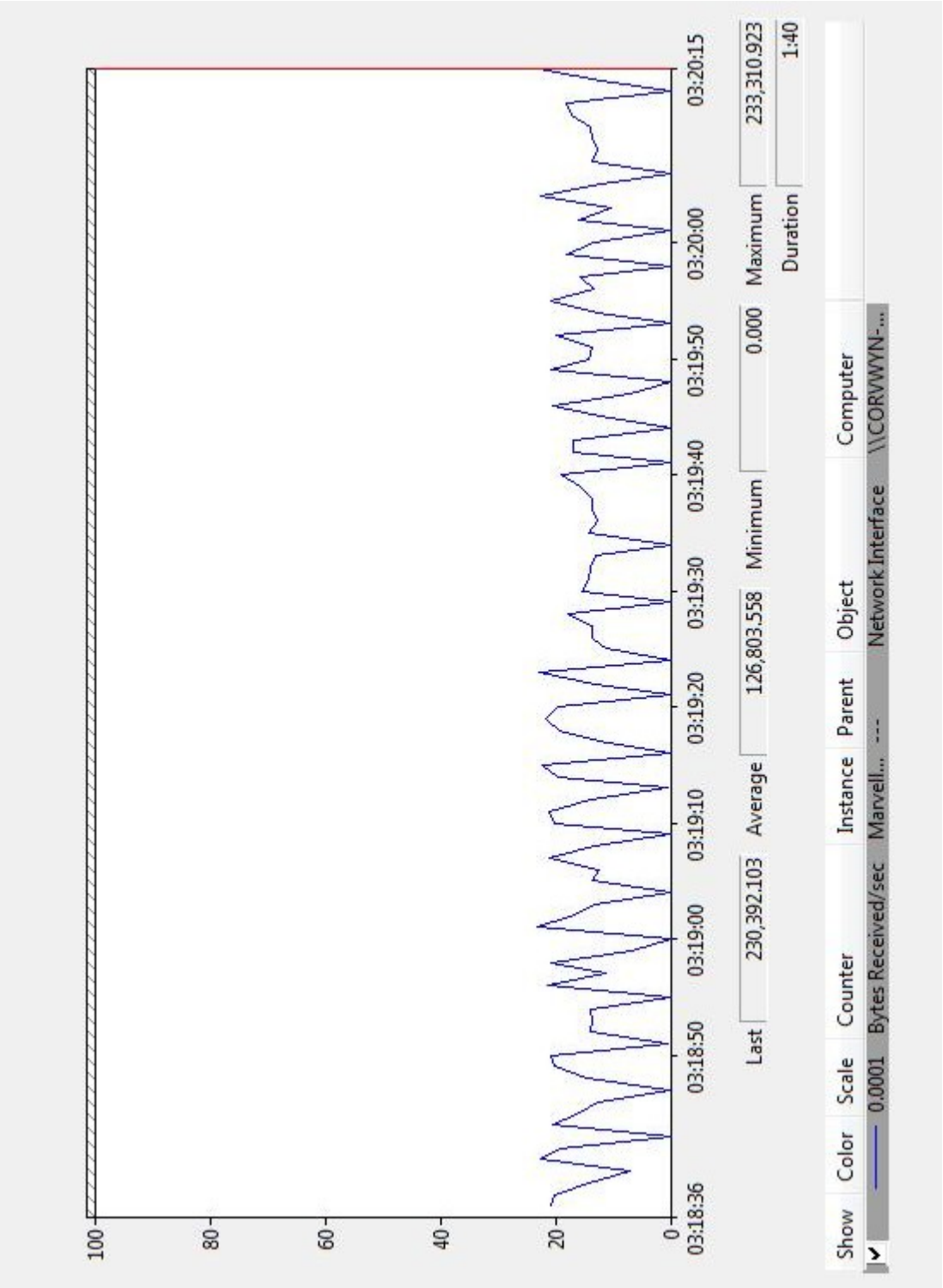

Figure B.4: Bandwidth usage with RTMPE

Appendix C

Bandwidth and CPU usage for different multiplexing scenarios

Test configuration:

	Server	Client
CPU	1.86 GHz Celeron	2.4 GHz Intel Quad
RAM	2 GB	6 GB
Operating system	Linux	Windows 2008 x64
Network speed	100 Mbit	1 Gbit

Table C.1: Testing configuration for multiplexing tests

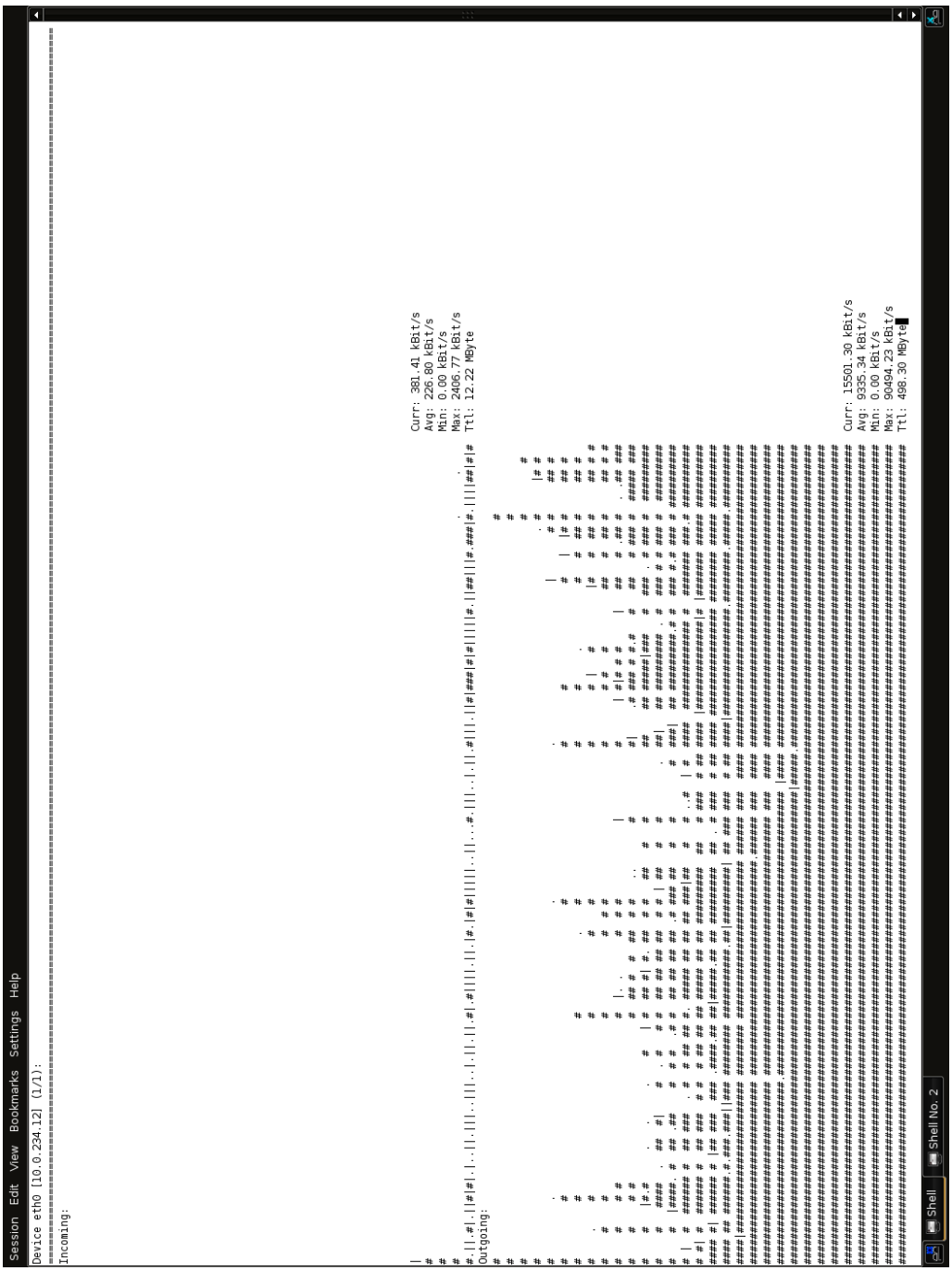

Figure C.1: Bandwidth usage when multiplexing is done before transferring the data

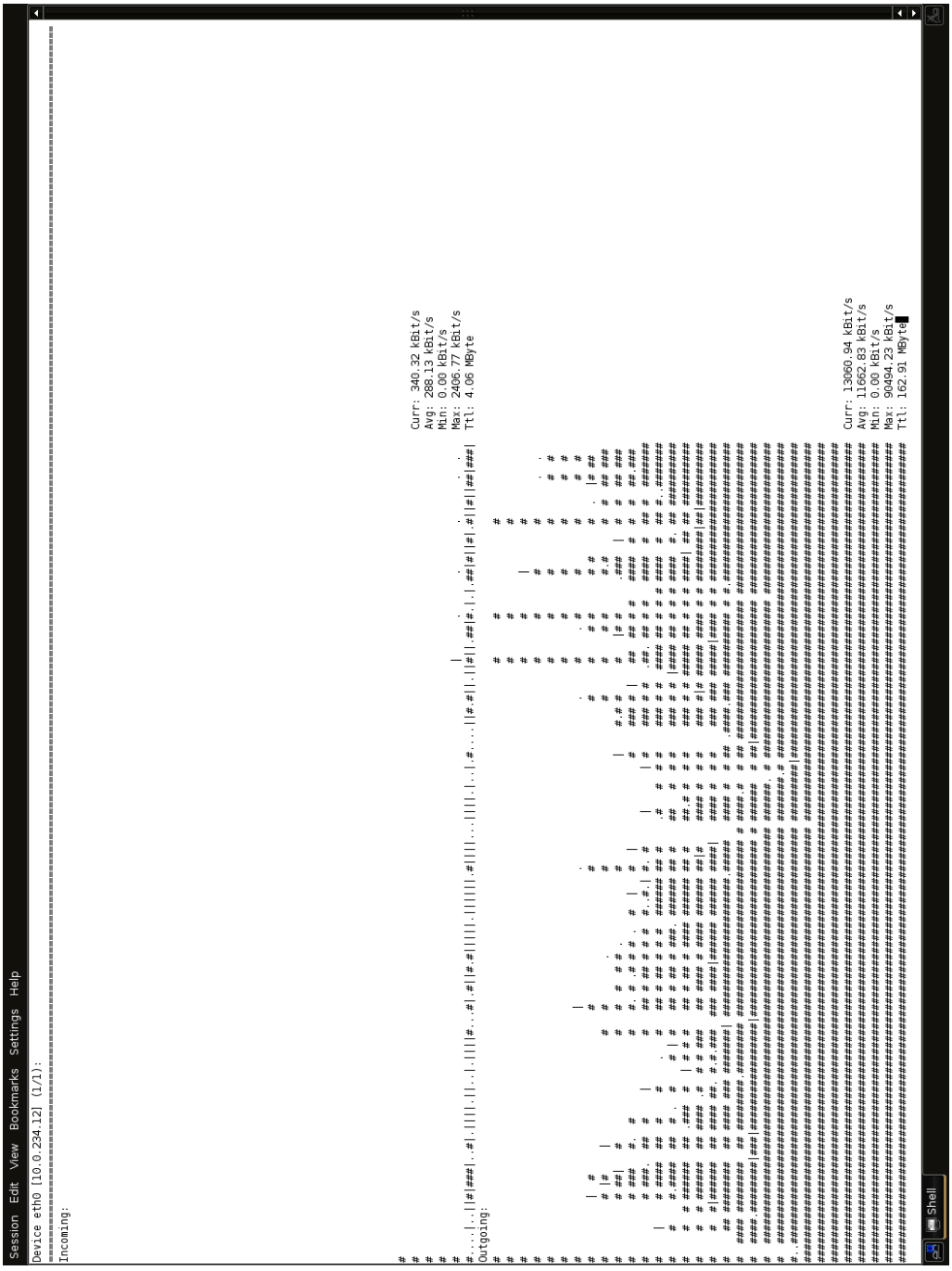

Figure C.2: Bandwidth usage when delivering data as 2 streams

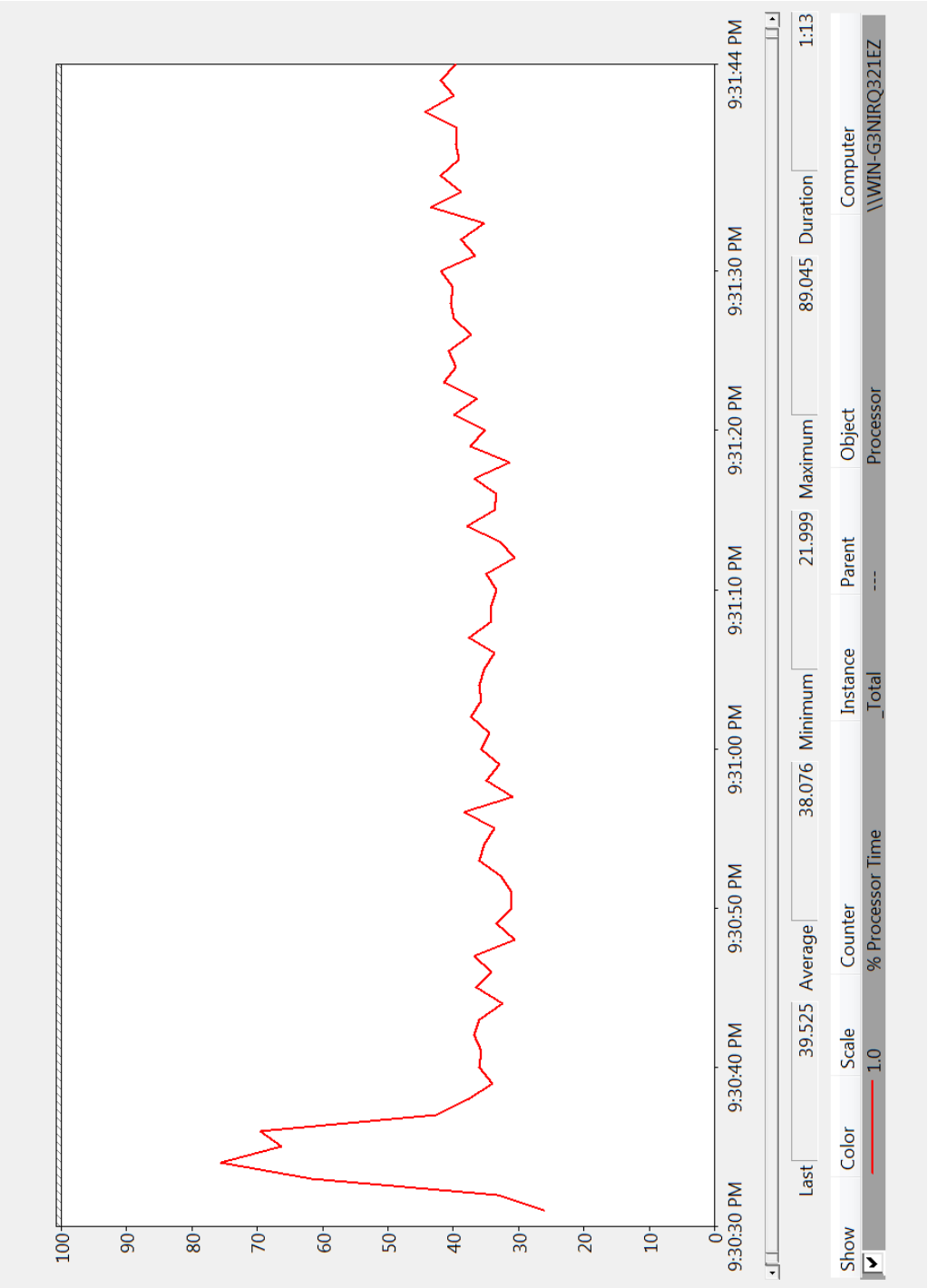

Figure C.3: CPU usage when delivering data as 2 streams

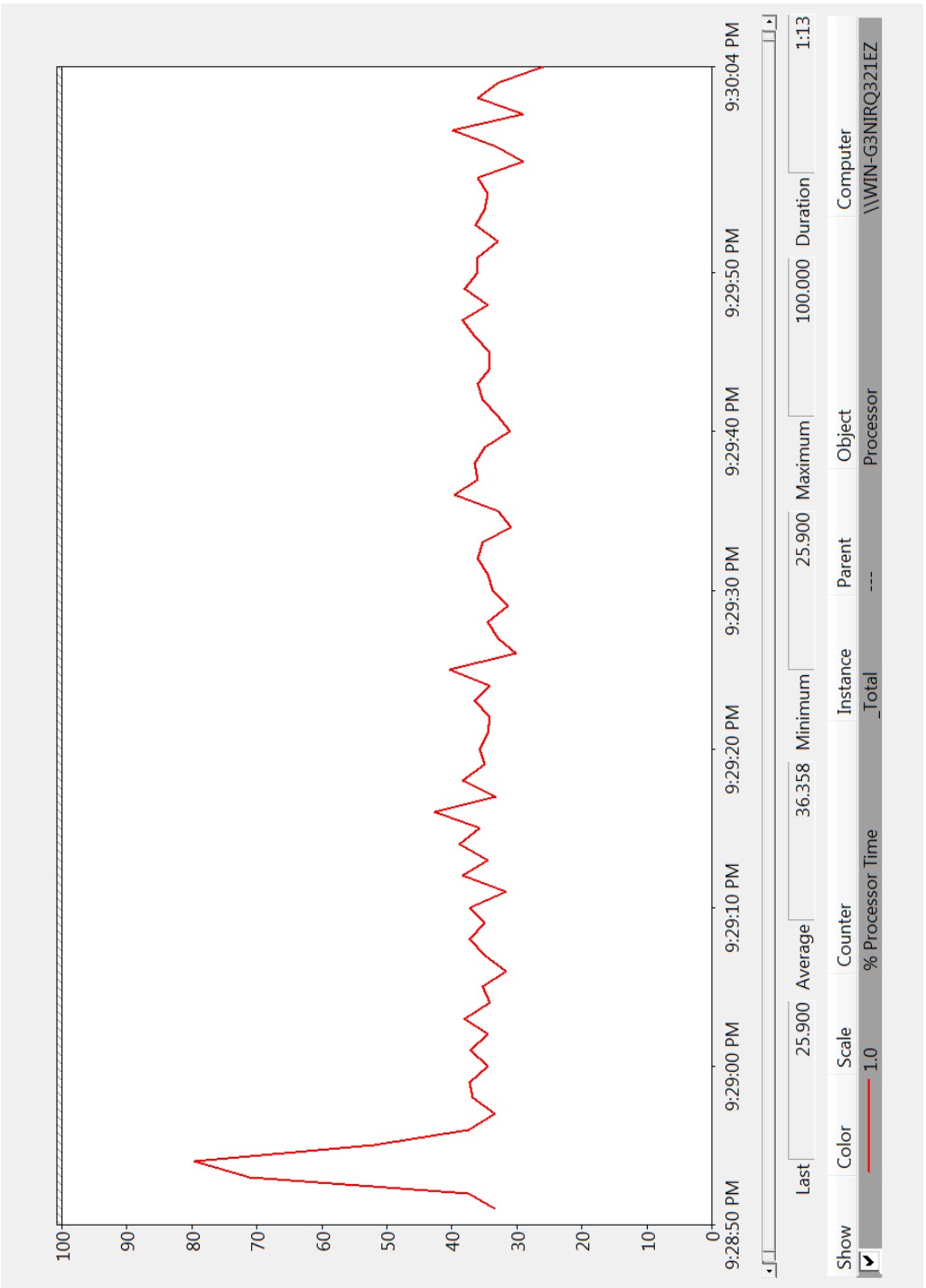

Figure C.4: Client CPU usage when multiplexing is done before transferring the data

List of Symbols and Abbreviations

Abbreviation	Description
VCEG	ITU-T Video Coding Experts Group
MPEG	ISO/IEC Moving Picture Experts Group, http://www.chiariglione.org/mpeg/
FPS	Frames per second
HD Video	Video with a resolution of 720 or more pixels vertically.
SDTV	TV resolutions under HD quality. Usually 625 or 480 pixels vertically.
FMS	Flash Media Server
FMRMS	Flash Media Rights Management Server
AMP	Adobe Media Player
AACS	Advanced Access Content System
VOD	Video On Demand
WMDRM	Windows Media DRM

List of Figures

List of Tables