

Rebeca Cenamor García

# Aggregating and modifying TV delivery to iOS devices

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Media Engineering

Thesis

Date 08.05.2013



Author(s) Title Number of Pages Date	Rebeca Cenamor García Aggregating and modifying TV delivery to iOS devices 72 pages + 3 appendices 8th May 2013
Degree	Bachelor of Engineering
Degree Program	Media Engineering
Specialization option	Digital Media
Instructor(s)	Erkki Aalto, Principal Lecturer Sasu Saarnia, Development Engineer
<p>The purpose of this thesis was to describe the current and most commonly used streaming system installed and used by students of Helsinki Metropolia University of Applied Sciences.</p> <p>This study aimed at creating written guidelines and tips for Metropolia students to help them with setting the system for streaming. The study first introduces briefly the TV production centre at Leppävaara campus. Secondly, broadcasting options available are described. Finally, guidelines for choosing the right broadcasting method are presented.</p> <p>This thesis also describes the equipment needed for streaming into the correct format and how to configure Adobe Live Encoder 3.2 to stream on the web.</p>	
Keywords	television, streaming, Apple, iOS, flash, Adobe



**Contents**

<b>Abbreviations and terms</b>	<b>0</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Theoretical background</b>	<b>2</b>
2.1 <i>TV production centres</i>	3
2.1.1 Studio floor	4
2.1.2 Control room	5
2.1.3 Master control room	8
2.2 <i>Television delivery</i>	9
2.2.1 DVB-C	10
2.2.2 IPTV	11
2.2.3 Web streaming	13
2.3 <i>Create video streaming</i>	14
2.4 <i>Adobe Flash</i>	16
2.5 <i>iOS supported formats</i>	18
2.6 <i>AAC</i>	19
2.7 <i>H.264</i>	19
<b>3 Methods and materials</b>	<b>20</b>
3.1 <i>Transmission and transfer system</i>	20
3.2 <i>Publishing system</i>	25
<b>4 Results</b>	<b>32</b>
4.1 <i>Placing the video on a web site</i>	32
4.1.1 JW Player 6	32
4.1.2 Coding guides	37
4.2 <i>Testing results</i>	38
4.3 <i>Host the web page for testing</i>	42
<b>5 Discussion</b>	<b>44</b>
<b>6 Conclusions</b>	<b>45</b>
<b>References</b>	<b>47</b>

<b>Apenddices</b>	<b>Ap1</b>
<i>Dolphin Studio Diagram for streaming [17]</i>	<i>Ap1</i>
<i>Matrox MXO2 LE and MXO2 LE MAX Connections [19]</i>	<i>Ap2</i>
<i>Web Script</i>	<i>Ap3</i>

## Abbreviations and terms

3GPP	3 <sup>rd</sup> Generation Partnership Project multimedia container format
ActionScript	Is an object oriented language based on ECMAScript. It is commonly used for web development and, embedded inside SWF files, for Adobe Flash Player software.
AMF	Action Message Format is the default format for Local Shared Objects, the form of the cookies stored on users' computers.
BNC	Bayonet Neill-Concelman connector used for coaxial cable.
DVB-C	Digital Video Broadcasting is the standard specifications for digital broadcasting across Europe, limiting the network to a physical cable connection.
DVB-S	Is the Digital Video Broadcasting standard applied to satellite communications.
ECMAScript	Scripting language used for client-side on the web. It uses languages such as ActionScript, Jscript and JavaScript.
EPG	Electronic Program Guide
F4V	Flash Video file extension
FLV	Flash Video is a container designed for transmitting video signals over Internet inlaid inside SWF files, using Adobe Flash Player.
FMS	Flash Media Server
GIF	Graphics Interchange Format is a bitmap image format which also supports animations.

H.264	This video format, also known as MPEG-4, uses a high compression codec for high definition videos, which significantly lowers the bitrate.
HD	High Definition
HDMI	High-Definition Multimedia Interface is a video and audio interface use for transferring uncompressed signals between the video/audio source and a monitor, video projector or digital television.
HD-SDI	High Definition Serial Digital Interface is a series of video interfaces standardized by SMPTE (The Society of Motion Picture and Television Engineers).
HLS	HTTP Live Streaming protocol was developed by Apple Inc. It divides the content into 10 s fragments encoded into MPEG-4 and sent in 10 s packages, requested from the client side as needed.
HTML5	HyperText Markup Language version 5 supports multimedia as the previous versions did not.
HTTP	Hypertext Transfer Protocol defines syntax and semantics of the software used in web architecture like clients, servers or proxies.
iOS	Mobile operating system created by Apple, originally named iPhone Operating System. It is based on the OSX operating systems series for Mac computers, but adapted to portable devices as iPod Touch, iPhone and iPad.
IPTV	Internet Protocol Television used to transmit TV content via Internet to a TV set top box.
JavaScript	Is a programming language created for being part of the user side on web browsers.



JPEG	Joint Photographic Experts Group is a lossy compression image format.
JSON	JavaScript Object Notation allows interoperability between Flash Player 11 and JavaScript.
M4A	MPEG-4 Part 14 Audio only
M4V	Video file format created by Apple, MPEG-4 Part 14
MHP	Multimedia Home Platform
MOV	Extension for the QuickTime multimedia file format.
MP4	MPEG-4 Part 14
MPEG-4	Also known as H.264. It is a compression method standardized by MPEG (Moving Picture Experts Group).
OS	Operating System
OSX	Is a series of operating systems created by Apple for Mac computers.
PDR	Program Direction Room, also known as control room in a TV production center.
PGM	Program signal is the definite transmitted by a television production centre.
Plugin	For a software application, a plugin is a complement that enlarges the abilities of this upper software.
PNG	Portable Network Graphics is a lossless compression image format.
PTV	Pay Television

PVW	Preview signal on the video mixer. This signal is the video flow selected to go to the program stream after the one is already been sent.
QAM	Quadrature Amplitude Modulation
RTMP	Real Time Messaging Protocol, owned by Adobe Systems, it allows the media streaming over the internet, for Flash player and Flash Media Server.
SD	Standard Definition
SNR	Signal-to-noise ratio is a measure that compares the signal level with the noise level. It is used as a quality measurement.
SWF	Small Web Format is a file format for vectorial graphics created by <i>Macromedia</i> (now <i>Adobe Systems</i> ). Can be generated by so many applications, but the original Adobe Flash uses an editable format with “.fla” extension. It is based on vectors and images, although it can have audio and video and some other interactive options. Its objective is to create small multiplatform files, working even on low bandwidth.
VBR	Variable Bit Rate is an encoding method that chose the bits per second used for codifying data. It varies depending on the complexity of the sequence treated.
VOD	Video on Demand is a system for watching video content by user request.
XML	eXtended Markup Language. It is a standard for the information exchange between different platforms in an easy, secure and reliable way.

## 1 Introduction

This project was born due to the necessity of transmitting TV delivery via streaming on the internet, based on *Adobe Flash Media Live Encoder*, for the periodical students' broadcastings, at Metropolia University's Leppävaara campus.

The desire is distributing to the maximum amount of people; this means that all the devices should receive the information packages properly, decode them and show the broadcasting in almost real time.

The purpose of this thesis is to describe the actual delivery system installed on the Dolphin laboratory of the Metropolia University of Applied Sciences. In addition, defines the appropriate method and system configuration for the data packages to be received by iOS devices, using the *Adobe Flash Media Live Encoder*.

The main objective of this thesis is to make it easier to students to setup and configure the system for future broadcastings by providing guidelines for fast and good results. The information given is necessary to understand the streaming process and to carry it out quickly and effectively.

## 2 Theoretical background

When introducing someone into the broadcasting process of audio-visual content, it is wise, first, to describe the necessary elements to make the transmission. Since this is not the main topic of this thesis, the different areas of a TV production centre, such as the studios, will be only briefly described. The methods needed for web-streaming, will be analysed with more detail in order to reach the widest audience possible. Which audience is growing in markets is the crucial question.

When broadcasting over the Internet, there are many different formats, software, plugins and ways to send and reproduce multimedia content. The use of them depends exclusively on users' choice and their preferences. One should keep in mind that it is actually a small part of the population who has the technical knowledge to distinguish over technologies and choose consequently. Therefore, there are no "technical reasons" to extol one technology over another. When it comes to getting a message across, what really matters is that the message actually reaches the receiver.

In Figure 1 there is a trend graph which shows, up to March 2013, the global market share of mobile operating systems mostly used.



Figure 1. Statistics of mobile operating systems users on the year 2013, up to March 2013 [1].

Although it can be observed that there is a high percentage of iOS users, the graph only shows the market tendency over the last year, and it is true that in the previous years the market was led by Android OS. With the release on September 2012 of the new iPhone 5, the number of iOS users has increased considerably and change the values in Apple's favour.

Because of the popularity of Apple products, the necessity of sharing the content also with iOS users is clear. Therefore this study focuses on formats valid for devices with iOS operating system installed.

## 2.1 TV production centres

This chapter describes what a production centre is and which elements are needed when producing TV content for broadcasting.

The complex system known as television studio is an installation in which audio-visual content is created, edited, and either broadcasted or recorded for future post-production. This system is subdivided into different rooms in order to make it more functional, since in each room different tasks needed to develop a TV program are performed. Usually, on big TV channels, the workflow is divided into sections according to TV programs, and within each section, there are physically separated rooms for different tasks. These rooms are interconnected and the workflow is divided for an effective result. The next paragraphs describe briefly the rooms that are actually installed at the Leppävaara campus of the Metropolia University of Applied Sciences, and their analogy with a professional television studio. In Figure 2 there is a graphic schema with the workflow at the Leppävaara TV production centre.

## Signal distribution system

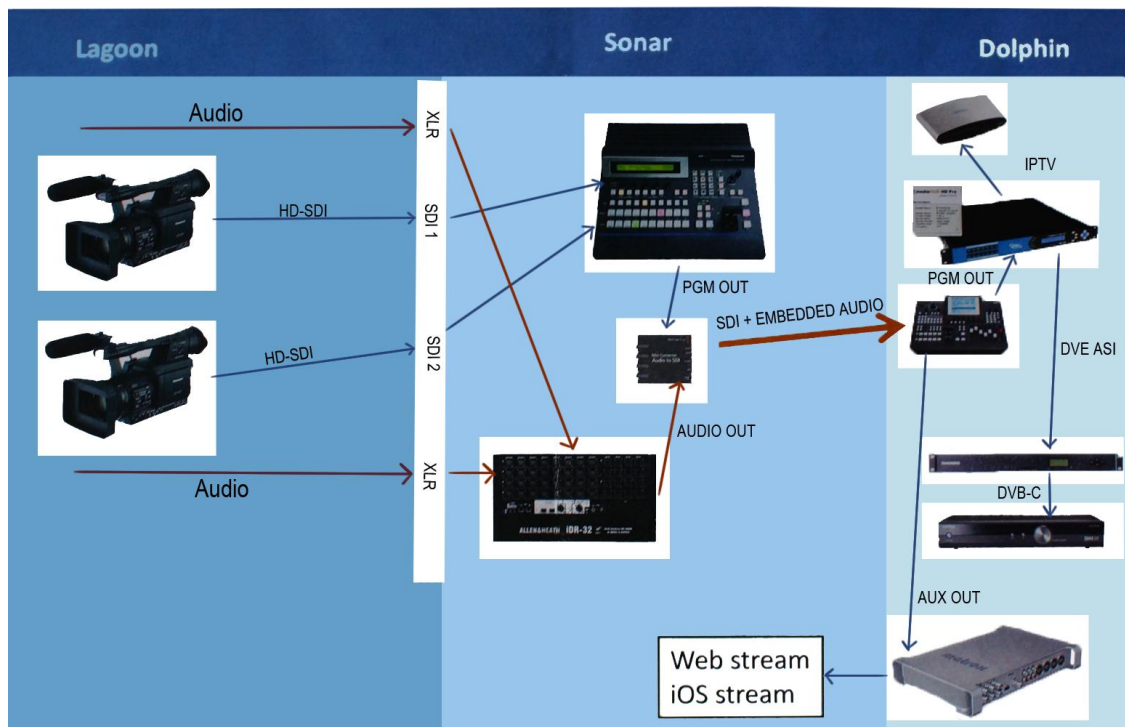


Figure 2. Signal distribution system. Modified from Erkki Aalto [2].

Figure 2 shows the path the signal follows from the cameras at Lagoon Studio, to the end of each of the three delivery methods available at Dolphin Studio.

### 2.1.1 Studio floor

In this room/s the recordings take place, using sets, video cameras, microphones, monitors, lighting and sometimes green screen for keying effects.

Those are Lagoon Studio and Hot Studio, where the audio-visual content is created and transmitted to the control room studio, in this case, Sonar Studio.

Figure 3 shows a photo of the Lagoon Studio, with a set of three cameras.



Figure 3. Lagoon Studio at Leppävaara Campus (Metropolia University of Applied Sciences).

The picture illustrates an example of a TV set, with the lighting, and cameras for shooting the material. Also, any kind of audio-visual material can be used for broadcasting, not only live content, but also pre-recorded files.

### 2.1.2 Control room

Also known as program direction room (PDR), its common task is compositing the outgoing program. Here is where audio and video content is received from the studio and processed and edited by an audio mixer and a video mixer. This is Sonar Studio and in Figure 4 there is an image of it.



Figure 4. Sonar Studio at Leppävaara Campus (Metropolia University of Applied Sciences).

The video is coming from the patch panel attached to the wall, on the right side, and it goes directly to the video mixer that can be found also on the right side of the desk. The audio goes the same way to the audio mixer, on the left side of the studio. In the monitoring system every signal is displayed, making it easier to select the right one on the video mixer, for sending it as a program signal (PGM). The operator can also manage pre-recorded files hosted in the server, for creating a complete TV program. In Figure 5 a zoom to the monitoring system has been done for explaining the incoming signals.





Figure 5. Sonar Studio monitoring display with incoming and outgoing signals.

On top of the screen the two main outputs are displayed. PGM is the final signal which goes directly to the next room. PVW is for a preview of the video flow selected to go to the program stream after the one that is already been sent. Some transition effects between both videos can be made by the operator, and some other stream from the other eight inputs can be selected for being emitted as program signal. The other input signals are the ones coming from the cameras at Lagoon Studio (input 1-3), but also other files hosted on the studio computer (input 7) or the server (input 8). This server is shown in Figure 6.

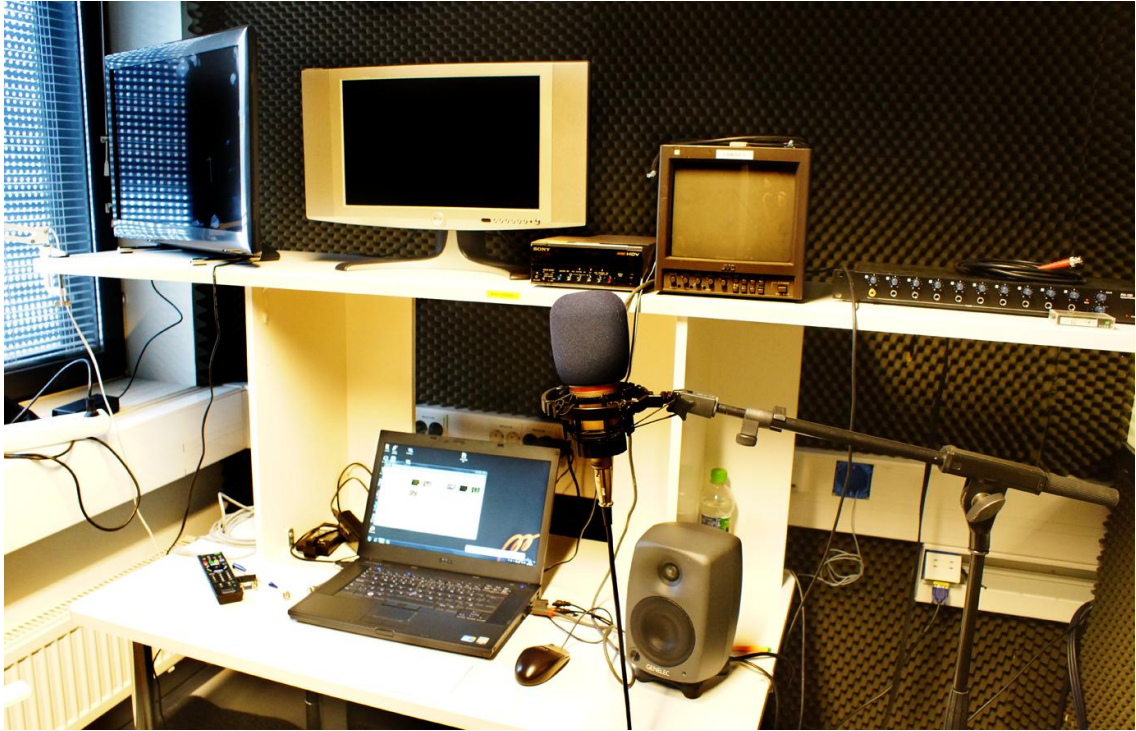


Figure 6. Sonar Studio second room at Leppävaara Campus (Metropolia University of Applied Sciences).

From the computer allocated in the secondary room of Sonar Studio, stored videos are sent to the video mixer for being part of the program as pre-recorded inserts.

During the process, program signal is sent as SDI with embedded audio to the next room, Dolphin Studio, the master control room of this production centre.

### 2.1.3 Master control room

In Dolphin Studio, all the program signals are received from the control rooms of the different studios of the production centre, in the case studied in this project, from Sonar Studio. In big centres there are sometimes different control rooms for the various sections that configure the TV channel.

At this point of the production, where all the contents are created, it is possible to switch between television programs, commercials or auto-promote videos, normally allocated on the video server accessible from Master room if they are not live programs. The channel logotype can be also added to the final program transmission.

It must be considered that the PGM signal produced in this room is the one to be broadcasted and received by the user.

The Metropolia example is shown in Figure 7.



Figure 7. Dolphin Studio mixing and monitoring desk at Leppävaara Campus (Metropolia University of Applied Sciences).

As can be seen at the monitor, the PGM signal selected and sent from Sonar Studio comes to the Dolphin Studio mixer and from there it is delivered to the different paths available for broadcasting. When knowing these three methods, it will be easier to understand the reason why Flash streaming is chosen.

## 2.2 Television delivery

As was already mentioned before, we can find three ways to deliver multimedia files at Dolphin Studio:

- DVB-C
- IPTV
- Web streaming

As an academic laboratory, its function is to lead the teaching to the widest level, by including all the industrial standard delivery systems of the hybrid television era. These systems have been collected in Figure 8.

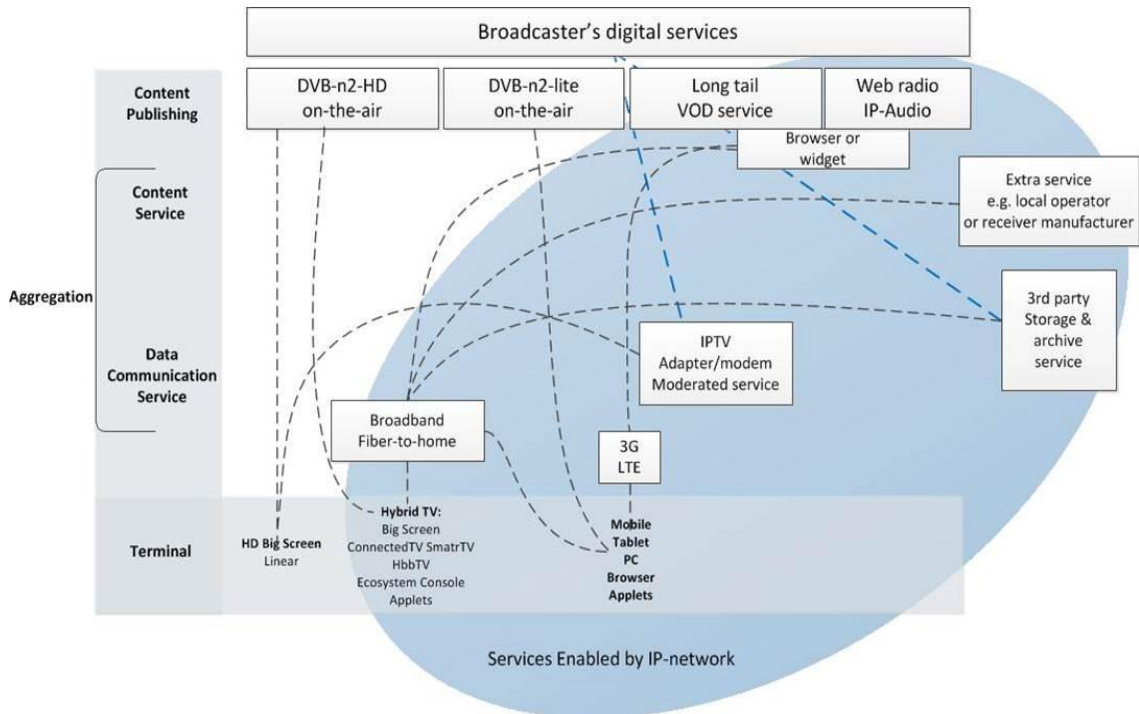


Figure 8. Broadcaster's digital services. Reprinted from Erkki Aalto. [3].

In this figure, the IPTV methods are remarked inside the blue ellipse, and the conventional antenna methods are shown on the left side.

### 2.2.1 DVB-C

Digital Video Broadcasting – Cable is a digital transmission system for cable television. As all the video signals used at the Metropolia production centre are digital, they must be modulated into analogue signals to be transmitted over an analogue channel. The DVB standard is used to assure that the modulation, synchronization and addition of error protection systems are correctly made to guarantee the quality and standardize the reception methods.

This standard uses modulation schemes between 16-256 QAM. Its channel bandwidth is 6-8 MHz, depending on the bitrate and the QAM modulation. As the diffusion is done

over cable, the interferences, noise and delays are minimum, presenting a SNR=30dB. [4].

The installation of the system is very important, because, as it happens in every cable connection, cables length and impedances adaptation between equipment, cable and charges at the end of them, could add some interference produced by the signal bouncing at the end of the transmission line.

In DVB – C can be also sent, multiplexed with the audio and video signal, additional data used to give access to the electronic program guide EPG, and other interactive data for a potential Interactive Television. [4].

It is a very expensive and complex way to broadcast because of the need to have a limited network, with physical cable, in which is possible to send and receive the data. However, it is cheaper than satellite broadcasting (DVB-S), which is completely out of the students' possibilities.

### 2.2.2 IPTV

Internet Protocol Television is a distribution system based on the delivery of television or video data packages using the internet connection. It is mainly based on Pay TV (PTV) and Video on Demand (VOD), accessing directly to the IP address where the content is being delivery to.

A conventional TV set-top-box has no the necessary elements for accessing to the internet connection, and decoding the digital video streaming. To add a computational layer to the set, an external device is added between a computer and the TV. In Figure 9 there is an example of the *Amino* device used at Dolphin Studio for testing the IPTV reception.



Figure 9. Amino device for receiving IPTV.

As a characteristic of any VOD, the content will reach the receptor when requested. There are two channel types, depending on if it is SDTV or HDTV. If a MPEG-4 (or H.264) codifying is used, 1.5 Mbps are needed for the first type, whereas 8 Mbps is used for connecting the second type. [5]. It means that the internet connection must have at least those speeds for every IPTV receptor of the network. Nowadays, ADSL velocity is higher than when it was developed, but not every home has enough connection speed for having a comfortable access to this service and the conventional Internet on their computers, and it requires a subscription to a provider company, usually a telephone company. Because of that, this trend is not really generalized. However, it can be used over a Local Area Network (LAN) connection in the same campus or business buildings. Sometimes it is used for videoconferences in these environments.

### 2.2.3 Web streaming

Very similar to the previous method, IPTV, web streaming makes use of video on demand, delivering multimedia content by explicit user request. The content is available and displayable when received by the user. The provider sends the multimedia stream and it can be seen before the entire data has been received. It is possible to send live video, in almost real time, or sent pre-recorded files. In this study, the first type will be described in the point 3.2.1 of this document.

Figure 10 illustrates how the common workflow for the whole streaming process, is organized.

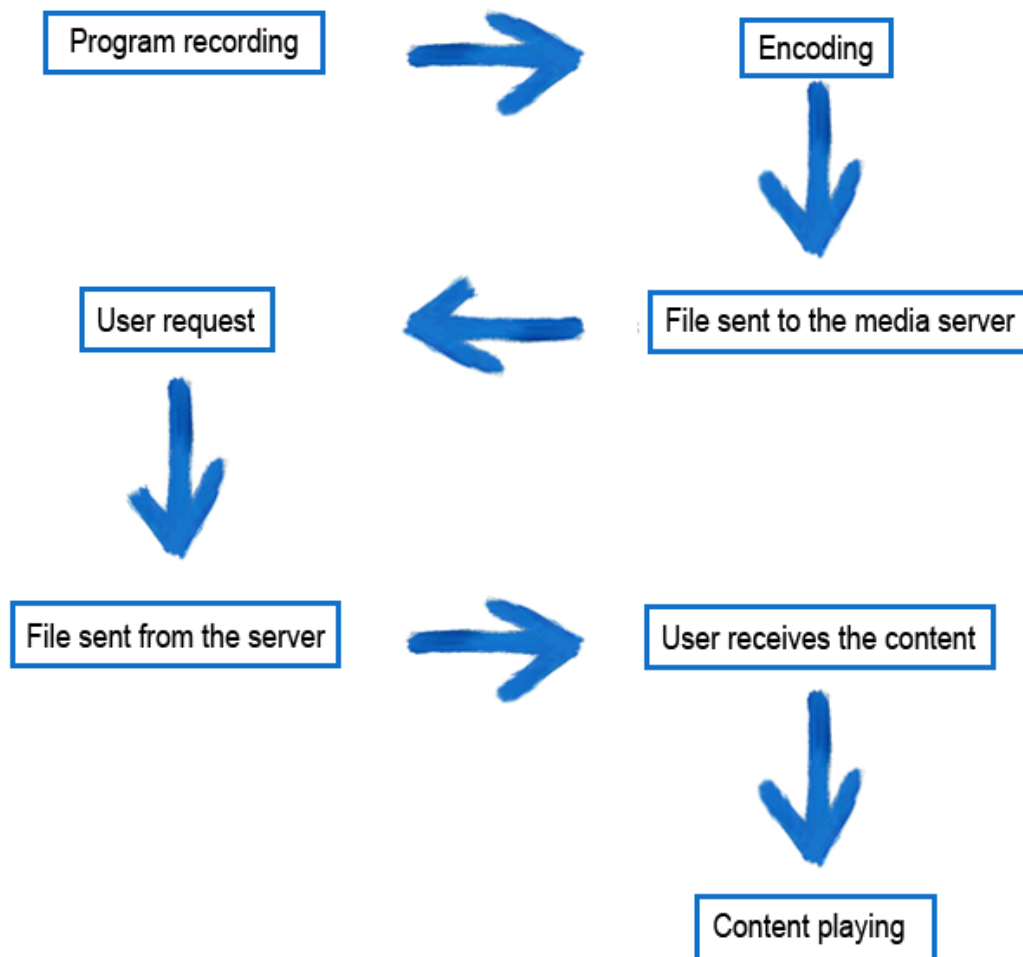


Figure 10. Work flow for a streaming delivery.

The first step is to acquire the multimedia data to be distributed. This will be done with audio/video capture devices, or taking the file from the server. The format depends on

the software to be used for streaming. Also the quality and the bitrate for delivering the packages at an optimal speed for the internet connection are chosen. The signal must be encoded and sent to the media server.

On the client side, all multimedia flows are received in packages, and kept on a buffer. In order to keep a smooth and uninterrupted playback, the buffer must be full enough to have enough material to show without any cut, while waiting for new packages. This also allows the user to pause and play the clip without losing all the information. The time used in this process depends on the internet connection, the format and the resolution, in other words, on the video size and the speed in which the server communicates with the user. Once the buffer is full enough, the transmission starts at the clip's bitrate previously chosen between the server and the client while the buffer is being filled.

The method described above is the most widespread because is the most accessible and comfortable, both for the user and the distributor. For the user, it takes just a web browser with the appropriate decoder plugin and an internet connection. For the distributor, there are many methods encoding the files and sending them. It is possible to purchase the service or to manage it by the distributors themselves. There are also different types of publishing material, depending on the desired service and the economic possibilities.

It is possible to place a video file on a server (proprietary or purchased) and link it on the web. This streaming is just on the user side, and requires less resources and money inversion on the distributor side. This method is known as *progressive download* or *HTTP streaming*. There is also the possibility of encoding and sending the media content while it is being created, using media servers and live encoders. This is the *true streaming* and is the chosen option for satisfying the needs of Metropolia students. [6, 220-224].

### 2.3 Create video streaming

This study describes a concept commonly known as *true streaming*, but it can be also simulate as HTTP streaming, which is an easier and cheaper method. As mentioned earlier, it does not allow live streaming, since it only works with files hosted on a server, and they must be complete at the moment they are sent.



For creating a *true streaming* broadcast, the first step is to choose the file format. Very often users have their own preferences for reproducing files, and will use one in particular. As providers of the content, the format must be chosen knowing that it could not be the one the users have. Therefore this choice requires careful consideration to not limit the streaming to just one user group. As was already mentioned in the introduction of this section, one of the objectives of this study is to create simultaneously a video streaming for web and for iOS devices in order to follow this rule.

Some of the most common media players are [6,126-128]:

- QuickTime
- Adobe Flash
- RealPlayer
- Windows Media

Some of the most common file formats are:

- .mp4 (MPEG-4)
- .mov (QuickTime)
- .fla/.flv/.swf (Adobe Flash)
- .wmv (Windows Media)
- .rm (RealMedia)
- .avi (Audio Video Interleaved)

Once the format is chosen, a streaming media server is needed. This is an application which allows the service to detect users' connection speed and adjust the bitrate for it automatically before managing the information delivery, in other words, to provide the service. With a media server it is also possible to broadcast live events, as will be described in 3.2.1.

The easiest way to manage a media server is to sign up for a hosted server, but there is also the possibility to operate with own servers, purchasing the server and managing it by ourselves, which is the Metropolia case.

Figure 11 demonstrates the results of a study conducted in 2011 by Millward Brown.

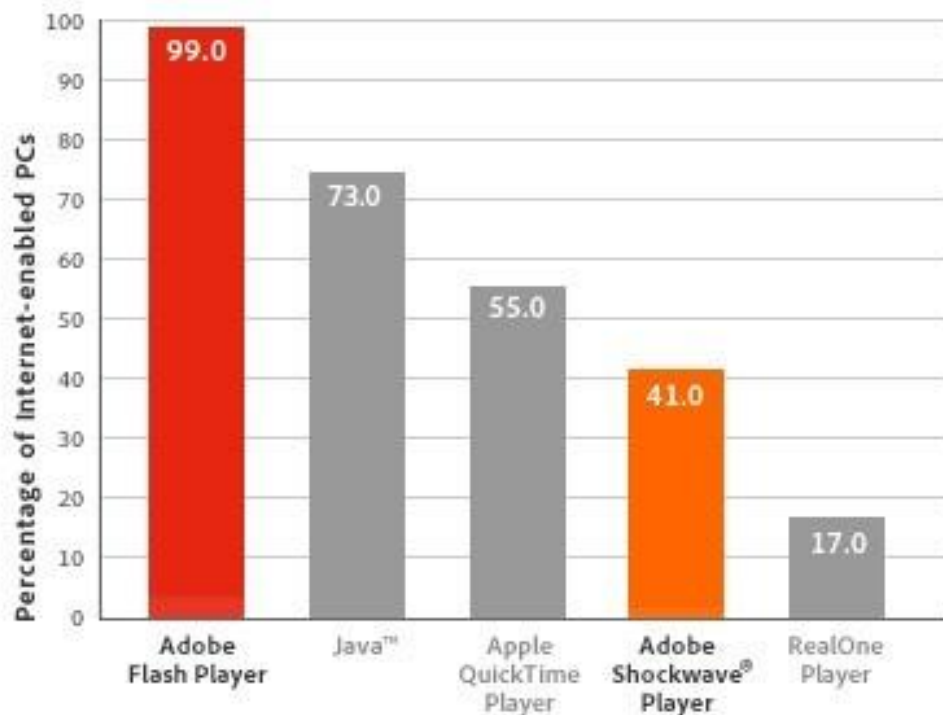


Figure 11. Browser media player market [7].

It represents the market penetration of the player system since it was released, and it seems clear that Adobe Flash is one of the most versatile and widespread options. This is why it has been chosen to be operated with at Metropolia. Adobe Flash Media Live Encoder (version 3.2) will be used for encoding the video and connecting to the server provided by Adobe Flash Media Server (FMS). The following paragraphs discuss why Adobe Flash was chosen.

## 2.4 Adobe Flash

Flash is the creation platform for multimedia content, created by *Macromedia* in 1996 and distributed nowadays by *Adobe Systems*. It allows creating SWF files (Small Web Format) containing the video file inside, and reproducing them with Flash Player. This player can appear in web browsers as a plugin, and in operating systems (such as mobile OS) as an application. [8].

Flash Player supports ActionScript language, based on object oriented programming, very similar to JavaScript, both based on the ECMAScript standard [9].

As the Millward Brown study shows, Adobe Flash is widely spread. One of the reasons is the easiness to design a multiplatform application. An application can be created and then ported into another platforms, so it makes it easy to developers to extend their ideas over the market. In 2010, Apple expressed its disagreement with Adobe Flash technology. Some of the reasons given were that it is a closed system, it is not secure, it was not using hardware acceleration and therefore, software acceleration was needed to decode videos. This however, uses too much power, as Steve Jobs stated in 2010. [10]:

*“Flash was created during the PC era – for PCs and mice. Flash is a successful business for Adobe, and we can understand why they want to push it beyond PCs. But the mobile era is about low power devices, touch interfaces and open web standards – all areas where Flash falls short.”*  
- Steve Jobs, April 2010. [10].

However, some months later, Flash Player 10.1 appeared, including hardware acceleration also in Mac, and using Cocoa interface for OSX. [9].

### **Supported formats**

Flash is a developer environment and it supports many different formats that must be known before choosing the proper one. It allows many data, multimedia and video formats such as: SWF, XML, AMF, JSON, FLV, PNG, JPEG and GIF. Audio is mostly codified on MP3 or AAC, but also MPEG-4 (H.264), F4V, MP4, M4V, M4A, 3GP, and multimedia formats such as MOV and 3GPP. [11; 12].

As the purpose of this project is to distribute the TV program to the maximum amount of people, the chosen format must fit most of the users' preferences for allowing them to watch the broadcasting.

Users with a PC and a web browser with Flash Player installed or integrated should not have any problem receiving the video streaming, but for Apple devices users there are some limitations, concerning the file format. In order to choose the right format for everyone, a few things must be known about Apple operating systems.

## 2.5 iOS supported formats

The Apple operating system for mobile devices, iOS (originally iPhone Operating System), was released in 2007 originally for iPod Touch and iPhone, but later it was developed to support iPad and Apple TV. It comes from the OSX operating system used on Mac, but with a touch based interface.

According to the latest iOS system (iOS 6), the audio and video formats allowed are [13, 22-23; 14]:

### Audio technologies:

- AAC
- Apple Lossless (ALAC)
- A-law
- IMA/ADPCM (IMA4)
- Linear PCM
- $\mu$ -law
- DVI/Intel IMA ADPCM
- Microsoft GSM 6.10
- AES3-2003

### Video filename extensions:

- .mov
- .mp4
- .m4v
- .3gp

### Video compression standards:

- H.264 up to 15 Mbps, 640x480 pixels, 30 fps, Low-Complexity of the H.264 Baseline Profile with AAC-LC audio up to 160 kbps, 48 kHz. It may include stereo audio in .mp4, .m4v or .mov file formats.
- H.264 up to 768 kbps, 320x240 pixels, 30 fps, Baseline profile up to Level 13 with AAC-LC audio up to 160 kbps, 48 kHz and the same file formats.

Until the HTML5 creation, playing videos on web pages required browsers to have a plugin to execute the video (excepting for Google Chrome which has already Flash Player integrated). Using the HTTP Live Streaming (HLS) iOS users have the possibility to receive and reproduce Flash video, in small doses of 10 s that are requested and received while they are needed after the first 10 s. HLS includes H.264 and AAC for audio and video content [15].

Reaching this point and comparing what Flash Player and what iOS supports, the best formats for the studied case are AAC format for audio compression and H.264 for video compression.

## 2.6 AAC

Advanced Audio Coding is a digital audio format. It uses lossy compression, losing some audio samples for having the highest compression level possible. Due to the high performance and the quality, is the most spread format on the Internet and it has been chosen by Apple as the main format for its audio devices.

This format uses a VBR (Variable Bit Rate). It is based on the Fourier's modified discrete cosine transform (MDCT) and removes acoustic redundancies of the signal.

It supports 96 kHz, 88.2 kHz, 48 kHz, 44.1 kHz, 24 kHz, 22.05 kHz, 16 kHz as sampling rates and various bit rates such as 128 kbps, 192 kbps, 320 kbps and 448 kbps [16].

AAC encoding is not supported by Adobe Flash Media Live Encoder on Windows, so Metropolia purchased the MainConcept AAC Encoder, which allows this encoding while using Windows.

## 2.7 H.264

It is a standard for defining a high compression video codec developed by the ITU-T (International Telecommunication Union) Video Coding Experts Group (VCEG) and the

ISO/IEC Moving Picture Experts Group (MPEG). It was designed to develop a standard to guarantee good quality of the images while compressing them, and reduce the bit rates compared to the previous versions.

It supports 4:2:0, 4:2:2 and 4:4:4 sampling with a sample bit depth precision ranging from 8 to 14 bits per sample and different compression profiles depending on the compression level for video resolution. Apart from the images I, P and B used on previous standards, it includes two new images, SP (Switching P) and SI (Switching I), to change between two different video streams fluidly and without a heavy image I. This compression method is based on Fourier's MDCT, as AAC format but lossless. [6, 89-126].

H.264 and AAC are used in both iOS and Flash, so that are the right choice nowadays, to achieve the objectives. In the next chapter everything described above, will be implemented, to obtain the desired video streaming.

### **3 Methods and materials**

This chapter analyzes the actual installation at Dolphin Studio, which resembles the Master control room of a common TV production centre, as explained in the point 2.1.3. Due to the fact that this studio was designed for an academic use, it must be considered as an example, knowing that it is not the only way to produce and deliver TV content.

During the following paragraphs, all the equipment will be described briefly. Also their significance for producing a simple broadcasting, is explained.

#### **3.1 Transmission and transfer system**

At Dolphin Master Control Room there are three ways of transmission available: for SD and HD quality TV (common digital terrestrial television), IPTV, and PC/Web. In Figure 12 there is a schema according to which the complete workflow at the Studio can be followed. The delivery methods are marked in the schema as shown in Figure 12:

1. DVB described at 2.2.1 *DVB-C*.
2. IPTV described at 2.2.2 *IPTV*.
3. Web streaming described at 2.2.3 *Web Streaming*.

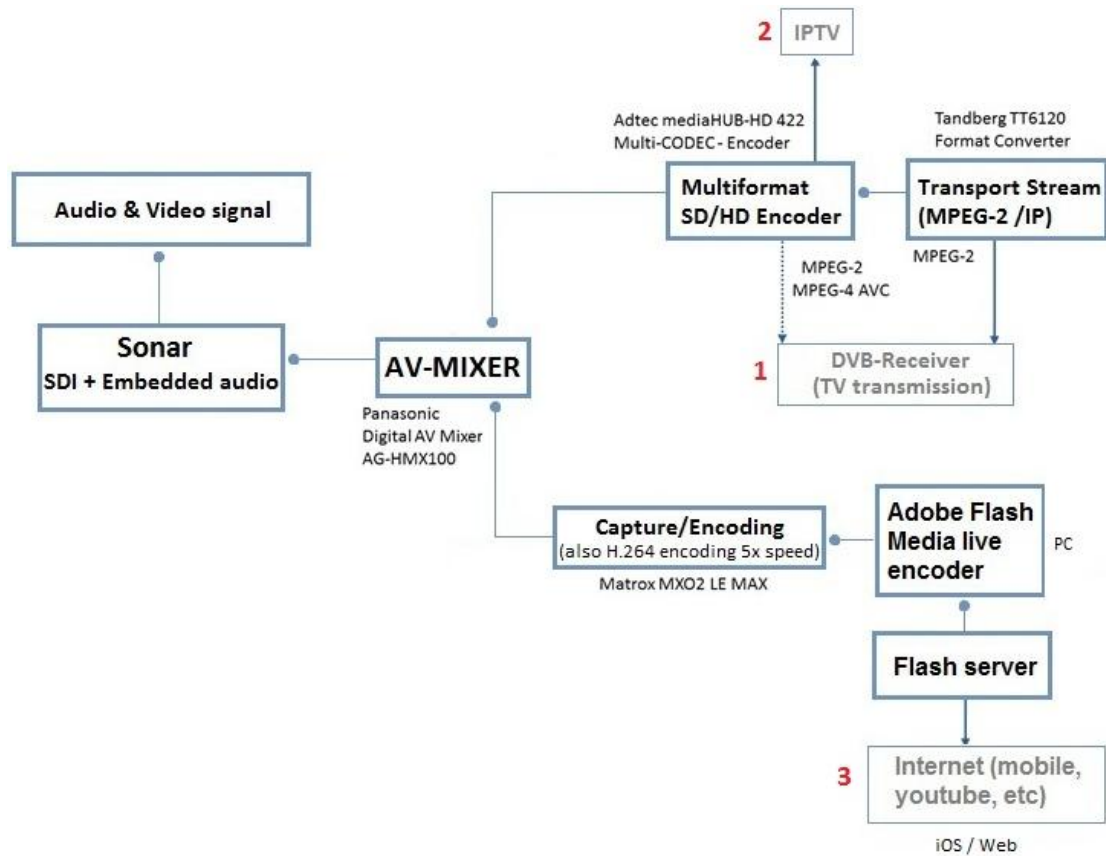


Figure 12. Workflow schema at Dolphin Studio. Modified from Erkki Aalto (2012) [2]

After being introduced in the whole system (*point 2.2*), only the third method and the equipment involved will be treated, starting from the AV-mixer. The third method corresponds to having chosen internet streaming through Adobe Flash.

## Equipment

The heart of Dolphin is the Panasonic AV-Mixer AG-HMX100. This device receives the signal coming from the Sonar Studio. It is HD-SDI signal, with audio embedded. The signal coming from Sonar has been already explained in the point 2.1.2.

Some of the main characteristics of this mixer can be found in Figure 13, making reference to the connections it has and the inputs and outputs type it allows.



**Video formats:**

480/59.94i, 576/50i

720/59.94p, 720/50p, 1080/59.94i, 1080/50i

**Video In:** Analog Composite x2, SDI x4, HDMI x2, DVI-I x1

**Video Out:** PGM: SDI x1, DVI-D x1

PVW: SDI x1

AUX: SDI x1

Multiview: SDI x1, DVI-D x1

**Audio In:** XLR x4 (L/R),

SDI (Embedded Audio) x4,

HDMI (Embedded audio) x2

AUX: Pin Jack x1 (L/R)

Mic: M6 x1

**Audio Out:** PGM: XLR x1 (L/R), SDI (Embedded) x1

PVW: SDI (Embedded) x1

AUX: SDI (Embedded) x1

Headphone: M6 x1

**Reference In:** BNC x2 (loop-through)

**Advance Reference Out:** BNC x1

**Tally Out:** D-sub 9 pin x1

**GPI:** BNC x1

**Remote:** D-sub 9 pin x1 (RS232C)

Figure 13. Main characteristics of the Panasonic AV-Mixer AG-HMX100. List data gathered from Erkki Aalto (2012) [2]

Once the mixer has received the signal, the operator has to select the content which is going to be sent as program content (PGM). This means a pre-chosen program list (EPG) must be followed, choosing between the available files to transmit TV programs or commercials in a temporary order previously established. For helping in this task, there are several monitoring devices. The most important one, with multi-screen option, shows every input that comes to the mixer. It provides a clearer view of the available material, and makes it easier to change between channels of the mixer for making the desired PGM signal. An example of this monitor was given in the point 2.1.2 *Control room* in the Figure 5 (page 7).

This signal is sent directly to the next device, the *Matrox* encoder. In Appendix 1 [17], a connexion diagram of the Internet streaming wing at Dolphin Studio can be found. It shows how the incoming signals from Sonar Studio reach the media server.



In Figure 14 and Figure 15, the Matrox device and its main characteristics are presented.



Figure 14. Matrox MXO2 LE MAX Encoder device, upper view.

Figure 14 shows the symmetric design of the device. It is a very useful guide, and acts as a bridge between transmission lines, where the signals are unwrapped, decoded, encoded and wrapped again in a different format.

**Video formats:**

NTSC/PAL/NTSC-EIAJ  
 720P 50/59.94  
 1080I 50/59.94  
 1080P 23.98/24/25/29.97/30  
 1080PsF 23.98/24/25/29.97/30

**Video In:** HDMI x1, SDI x1, Composite x1,  
 S-Video x1, Component x1

**Video Out:** HDMI x1, SDI x2 identical,  
 Composite x1, S-Video x1, Component x1

**Audio In:** unbalanced RCA x2, balanced XLR x2,  
 SDI(8/16 channels on PC/Mac) x1,  
 HDMI(8 channels) x1

**Audio Out:** unbalanced RCA x2, balanced XLR x2,  
 SDI(8/16 channels on PC/Mac) x1,  
 HDMI(8 channels) x1

**Resolutions:** from 64 x 64 to 1920 x 1080

**Bit rates:** 100 Kb/s to 50 Mb/s

**Frame rates:** 15, 23.98, 24, 25, 29.97, 30, 50, 59.94, and 60 fps

**Encoding controls:** GOP structure

2.0, 3.0, 3.1, 4.0, 4.1, and 4.2 level support  
 Constant bit rate, Constant quality,  
 Variable bit rate, Noise filtering,  
 De-interlacing, Data rate slider,  
 Sample rate monitoring, Scene detection

Figure 15. Matrox MXO2 LE MAX Encoder device supported signals main characteristics. Data gathered from Matrox website. [18].

Figure 15 illustrates the input connectors for the different supported signals.

In Appendix 2 three images have been added pointing the different inputs and outputs Matrox MXO2 LE MAX has [19].

Matrox MAX implements a fast H.264 encoding in various formats and resolutions. As it is not necessary to use the CPU processor, it can be used for simultaneous tasks, and so, dedicated to the exclusive integrated processor of the Matrox device only for the encoding task, making this process so much faster than using the computer resources. Matrox supports a direct connection with Adobe Media Live Encoder, the software chosen for communicate with the Adobe Media Server, avoiding having to install new plugins or software

The next step is as simple as receiving the video stream on the Dolphin Studio computer, already in H.264 format, and opening the Adobe Flash Media Live Encoder 3.2.

### 3.2 Publishing system

The next chapter will continue from the moment in which Adobe Flash Media Live Encoder 3.2 is opened and will describe and show the paces for configuring the program and reaching the publishing of the content.

#### Adobe Flash Media Live Encoder set up

The material is recorded, edited, mixed and transmitted to the PC from which will be then sent to the network.

Figure 16 represents an example of the screen that could be found as soon as the software starts.

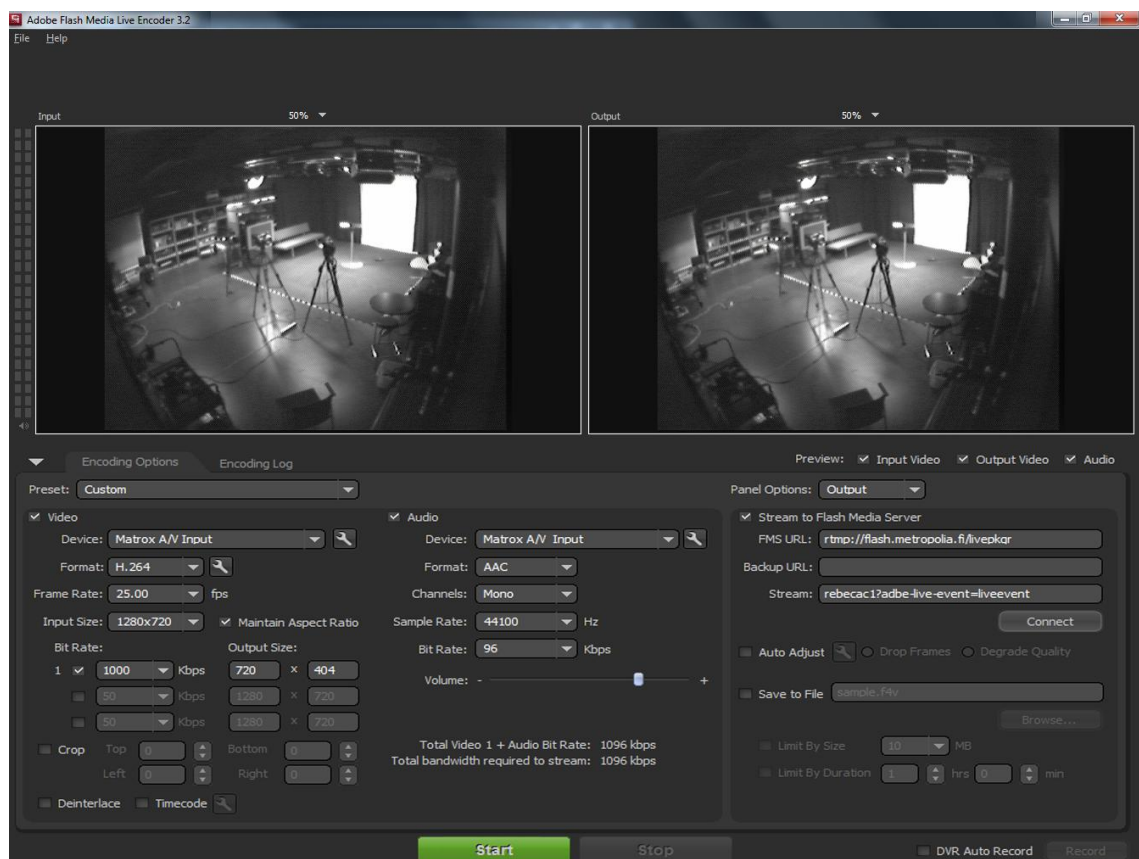


Figure 16. Adobe Flash Media Live Encoder 3.2 screen example.

Figure 4 and Figure 5 (*point 2.1.2 Control room* pages 6 and 7) show how Input 5, the security camera at Lagoon Studio, is sent as PGM signal. In Figure 7 (*point 2.1.3 Master control room* page 9), the equipment verifies that the signal has arrived to Dolphin and is sent as final TV program. The same image logically appears at the Live Encoder screen as input and output because it is the signal that Matrox as well as the PC receives. Figure 17 is a zoom in of the preview options.

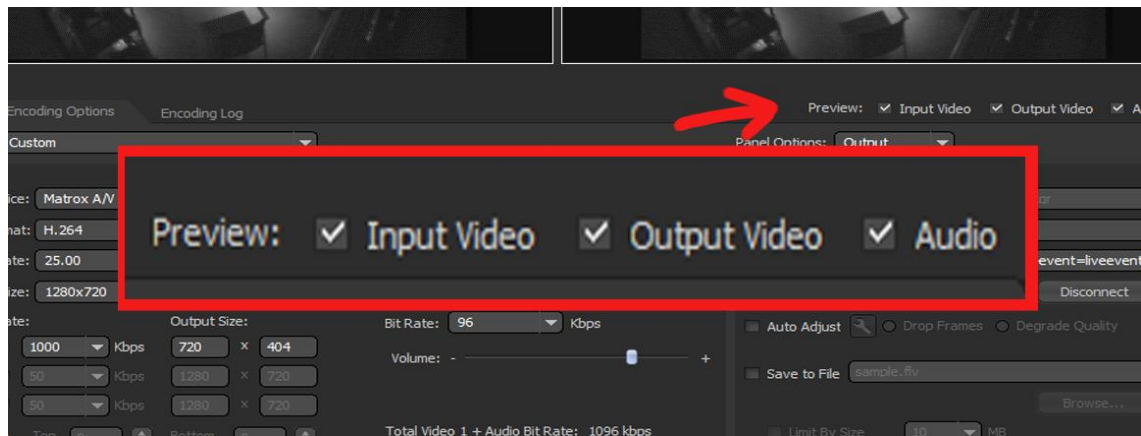


Figure 17. Preview options in detail.

Checking and unchecking the labels, the monitored screens Input and Output mentioned on the above paragraph are enabled or disabled, as well as the audio preview. The preview size of the video can be change also, for a better visualization.

Figure 18 below presents a zoom in on the Encoding Options menu, where the video and audio settings can be changed, choosing from some preset or customized options.

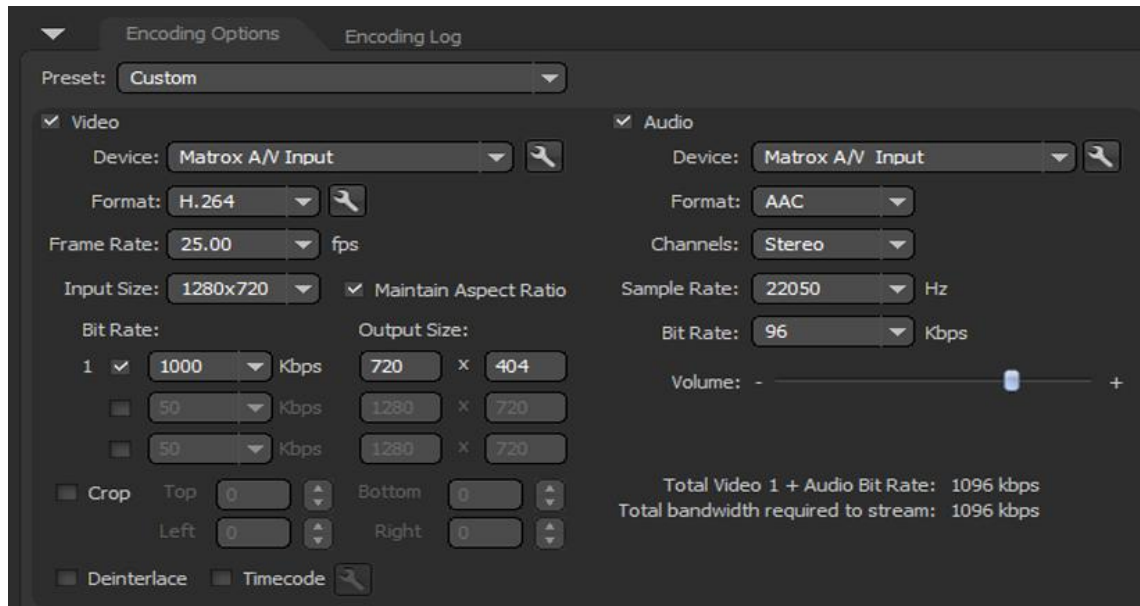


Figure 18. Encoding Options for video and audio.

Enabling or disabling the video or the audio labels, it is possible to choose between sending video/audio, or both.

The device options, in the study case, only offers Matrox device because it is the only installed equipment for this purpose. Some options are given for the chosen device, as can be seen on Figure 19 and Figure 20.

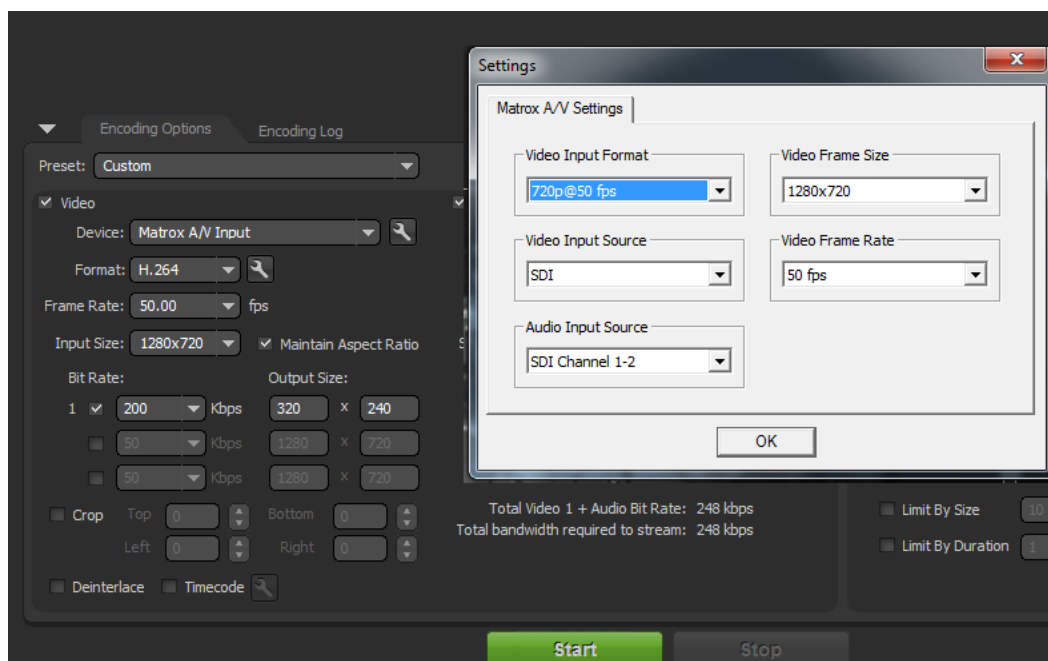


Figure 19. Device options available.

The options shown in Figure 19 depend on the selected device and its set configuration. For Matrox device, are the ones shown in the Figure 20.

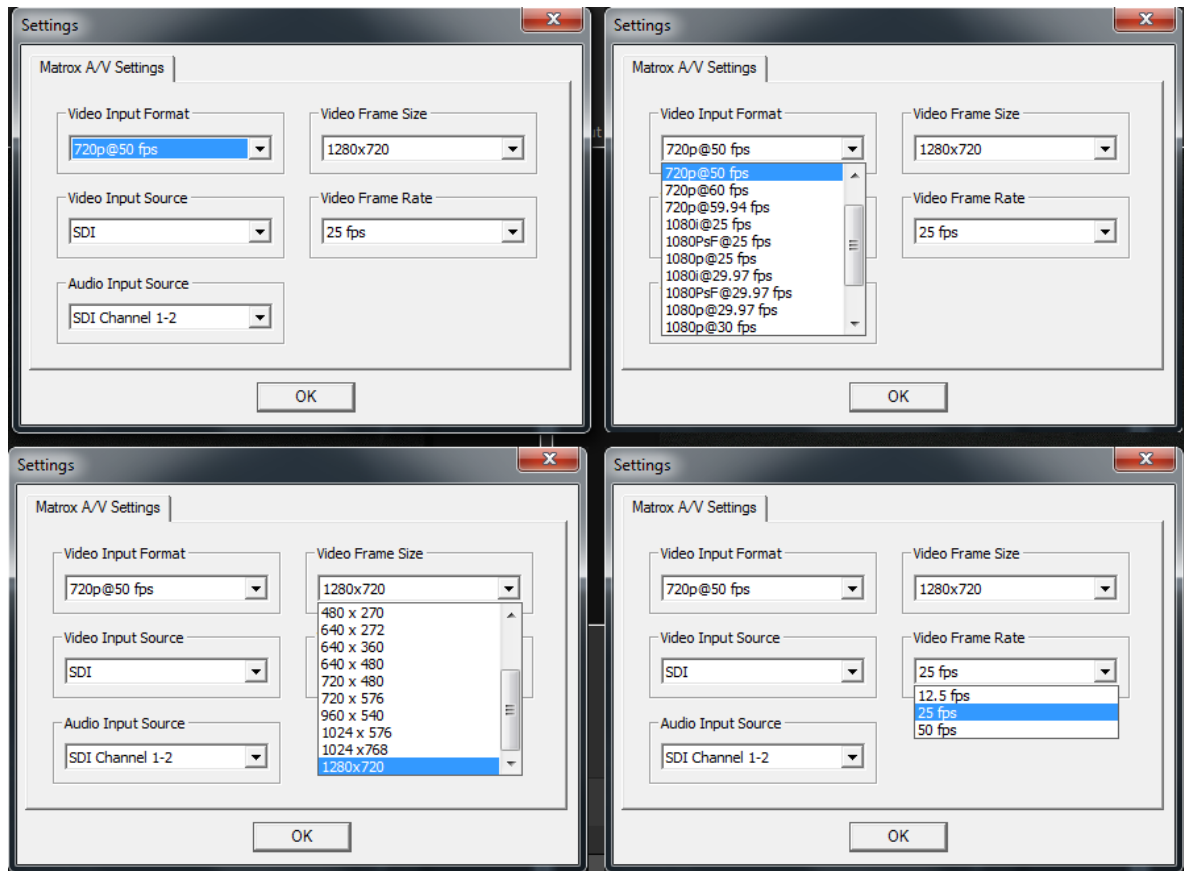


Figure 20. Some of the options offered in each label and the selected ones for the project.

The format chosen has been treated during the previous chapters, and the election of H.264 format for the video file was justified. The frame rate and the input size depend on the original video settings, during recorded.

The program allows selecting 3 different bit rates and output sizes that are going to be chosen depending on the user's connection, for improving the efficiency of the streaming. For receiving H.264 files, iOS devices require a 25 fps rate, and 1280 x 720 input size. The bit rate depends on the upload connection speed available for the streaming, and the number of users expected to receive the content. As more users receive the streaming, the service becomes less fluid, and more bit rate is needed. The output video size can be selected by the provider also. As *Maintain Aspect Ratio* label is selected, when converting an input video width of 1280 to 720, the height is converted into 404, as can be seen in this example. Standard values, as 640 x 360 can also be chosen.

For the audio settings, it was mentioned that AAC format would be used, with two stereo channels. A sample rate of 22050 Hz or 44100 Hz and 96 kbps of bit rate will lead to a good quality audio stream. The volume can be also adjusted before sending.

The next step is to connect to the Adobe FMS. Settings for connecting are at the Panel Options, as is shown in Figure 21.

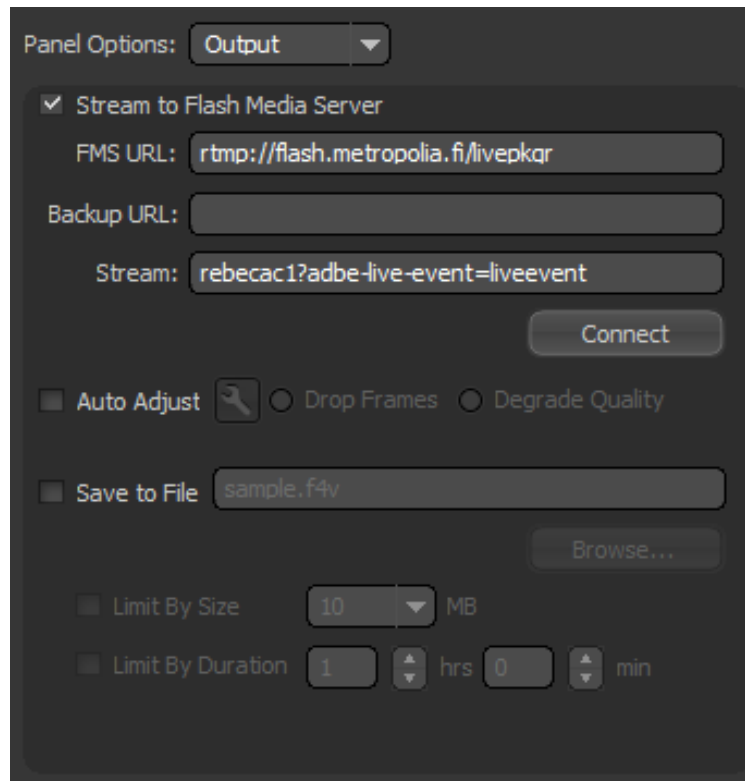


Figure 21. Panel options for connecting to FMS.

Metropolia University has its own domain, allocated at the URL: flash.metropolia.fi. For live streaming Adobe real time messaging protocol must be used (RTMP). As iOS devices are using HLS (HTTP Live Streaming), and it divides and sends the information into small fragments, it is necessary to publish the streams to the HTTP Live Packager service on the server. The Live Packager expects a concrete value to be written on the stream option, it is *name?adbe-live-event=liveevent*, which is a specification of the Adobe standard. [20].

As the file can also be recorded, there is the option to enable it and specify a file name. In this case it was not required, since it could be that other recording device can be installed between the Matrox device and the computer.

A resume of the important information requested to connect to the Media Server, is shown in Table 1.

Table 1. Data for connecting to the server and start the streaming.

	General	Metropolia UAS
<b>FMS URL:</b>	rtmp://domain/livepkgr	rtmp://flash.metropolia.fi/livepkgr
<b>Stream:</b>	livestream?adbe-live-event=liveevent	streamName?adbe-live-event=liveevent

It is very important to rename the stream each time a new one is made, because of the files storage on the server. It will not connect but will produce an error if this rule is not followed.

Once the connection with the server is made, it will appear the “Connected” message at the bottom of the screen as Figure 22 shows.



Figure 22. Connection is correct and start option is enabled to star streaming.

The Start option will become active and by pressing it, the streaming will start and the screen will change into the Encoding Log menu which can be seen in Figure 23.

Apart from the log generated during the connection process and the streaming beginning, a short resume is shown at the right side, and a new message appears at the bottom of the page, as it is zoomed in. Figure 24 shows this.



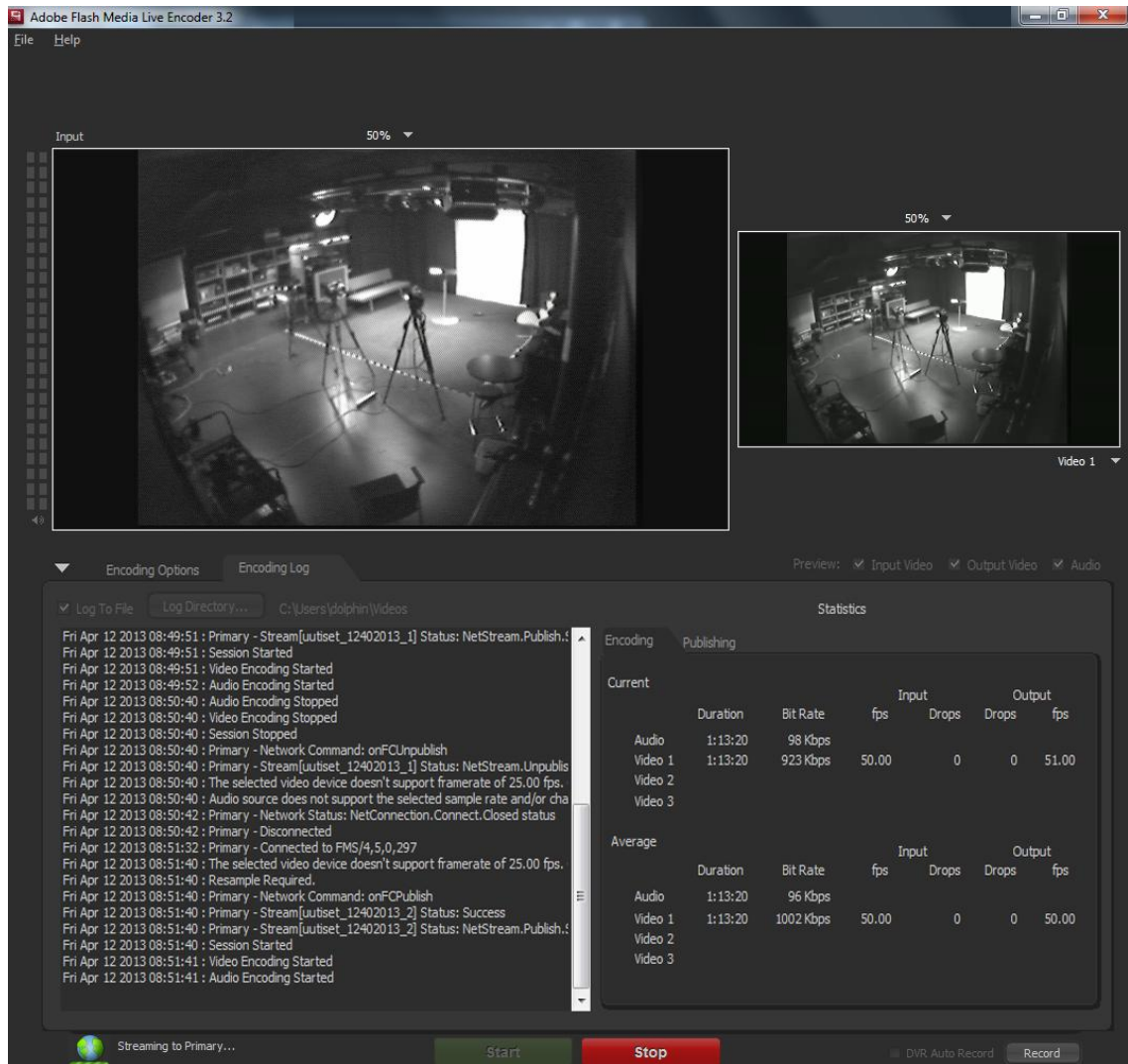


Figure 23. Encoding Log screen after server connection is made.

In this figure the difference in the aspect ratio between the input and the output images can also be seen.



Figure 24. Streaming is taking place.

Now that the process is ending, the last thing to test is the streaming, and to verify that they are working properly for web and for iOS. This testing process will be demonstrated in the next chapter.

## 4 Results

The streaming has now been completed. The signal is going out from the studio but, is it really going through the Internet? Is it possible to receive it and watch it? The next paragraphs will show the results of testing the correct working of the made delivery. But first, where is the video?

### 4.1 Placing the video on a web site

The video is being sent to the domain at the FMS, and it must be redirected to a media player for being played. There are many different web players compatible with Flash and for this example, *JW Player 6*, an easy and free to obtain media player for HTML5 and Flash, was chosen.

#### 4.1.1 JW Player 6

This section is for giving some information about the JW Player 6, created by Longtail Video. Here are some of the specifications and supported formats. [21]:

##### Media Support:

- Video files: MP4 (H.264/AAC), FLV (VP6/MP3), WebM (VP8/Vorbis).
- Audio files: AAC, MP3, Vorbis.
- Streaming protocols: Adobe RTMP, Apple HLS.

##### Device support:

- Desktop browsers: Chrome, Internet Explorer 8/9, Firefox, Safari, Opera.
- Mobile devices: iOS, Android 2.3, Android 4.
- Rendering modes: Support for Flash + HTML5 on desktops (auto-detected), HTML5 on devices. Built-in download fallback adds basic playback to other devices (e.g. Blackberry, Android 2.2).

#### Streaming:

- RTMP: Single MP4, FLV, AAC, MP3 or live stream. Dynamic streams w/automatic bitrate switching. Options for bufferlength.
- Apple HLS: Playback in desktop browsers (Flash mode but only in the paid version) and iOS.
- Flash pseudo-streaming: Supports mod\_h264 (apache/iis/nginx), mod\_flv (apache) or HttpFlvModule (nginx).
- Server support: Official server support for Wowza Media Server and Adobe Media Server.

It supports Adobe FMS, RTMP streaming, H.264 and AAC, so it is valid for the example treated in this study.

It is downloadable from the Longtail Video web (<http://www.longtailvideo.com/jw-player/download/> at 14<sup>th</sup> of April 2013).

As soon as the download is finished, a .zip file containing the necessary files for adding the JW Player to a web page will be available. These files are:

- jwplayer.flash.swf
- jwplayer.html5.js
- jwplayer.js

A “readme” file will also appear explaining and giving some tips for adding this player to a website.

**Note:** *some coding will be needed for that purpose, and in this section will appear some examples.*

As it says, the first thing to do is to copy the *jwplayer* folder attached at the .zip file to the root of the website. This will give access to the *jwplayer.js* script that is inside the folder, to the website. For this, the script must be included in the *<head>* section of the web page code. The script appears at Listing 1. [22].

```
<script type="text/javascript" src="/Jwplayer.js"></script>
```

Listing 1. Code for adding jwplayer.js script to <head> part. Reprinted from JW Player manual. [22].

After placing the Listing 1 code at the <head> section, the JW Player location must be defined at the <body> section with the <div> element that appears at Listing 2. [22].

```
<div id="myElement">Loading the player ...</div>
```

Listing 2. Code for indicates the player position. Reprinted from JW Player manual. [22].

Immediately bellow, the script for invokes the JW Player *setup()* on the position defined with the Listing 2 code. [22]. The script can be found at Listing 3.

```
<script type="text/javascript">
  jwplayer("myElement").setup({
    file: "/uploads/example.mp4",
    height: 360,
    image: "/uploads/example.jpg",
    width: 640
  });
</script>
```

Listing 3. Script for setup the player. Reprinted from JW Player manual. [22].

The Listing 3 scripts setup the player over the element defined by Listing 2. There are some options editable on this setup, as file, height, image and width, but other options exist to configure and edit the player to appear as desired. [22]. This script is just an example, and options must be changed according to the current streaming. As already mentioned before, RTMP was used for streaming, and for this protocol, different scripts must be added to the HTTP code of the web site for embedding the stream into JW Player 6. [23]. Listing 4 shows the changes to be done for RTMP streaming.

```
<script type="text/javascript">
  jwplayer("myElement").setup({
    file:"rtmp://example.com/application/mp4:myVideo.mp4",
    image: "/assets/myVideo.jpg",
    height: 360,
    width: 640
  });
</script>
```

Listing 4. Example script for RTMP streaming. Reprinted from JW Player manual. [22].

The code written on Listing 4 is the script to be added to the *<body>* section of the HTTP code, instead of the one on Listing 3, which is not for RTMP streaming. If the scripts given as examples are taken and joined in the HTTP code, will be obtained the code shown in Listing 5.

```
<head>
<script type="text/javascript" src="//Jwplayer.js"></script>
</head>

<body>
<div id="myElement">Loading... </div>
<script type="text/javascript">
  jwplayer("myElement").setup({
    file:"rtmp://example.com/application/mp4:myVideo.mp4",
    image: "/assets/myVideo.jpg",
    height: 360,
    width: 640
  });
</script>
</body>
```

Listing 5. Resume of the example code. Modified from Reprinted JW Player manual. [22; 23].

The code in Listing 5 is not still operable for the created streaming, the file, image, height and width options must be changed. The proper values are shown in Listing 6.

```
<script type="text/javascript">
  jwplayer("myElement").setup({
    file:"rtmp://flash.metropolia.fi/live/<?php
      echo $streamName; ?>",
    image: "/assets/1.png",
    width: 720,
    height: 420
  });
</script>
```

Listing 6. Definite script for calling the current streaming to the JW Player. Reprinted from Sasu Saarnia [24].

in Listing 6 had appear new parameters as are `<?php; ?>`, `echo()` and `$streamName`. Some information will be given about code in the next point 4.1.2. But first, for finishing all the JW Player section, the explanation will continue concerning the scripts the player needs. All the same, the script appearing in Listing 6 is valid when conventional streaming is made. It can be observed that the file parameter is just referring to `/live` server location, while it was already mentioned that, for iOS devices, the Live Packager was required for simulate the HSL. For this reason, as it appears in Table 1 (page 30) the file parameter must refer access to the `livepkg` at `flash.metropolia.fi`. The complete file parameter is shown in Listing 7.

```
file:"rtmp://flash.metropolia.fi/livepkg/<?php echo
  $iOSname;?> ?adbe-live-event=liveevent"
```

Listing 7. *file* parameter for streaming for iOS devices. Reprinted from Sasu Saarnia. [24].

in Listing 7 appears the instruction `?adbe-live-event=liveevent` that was already explained, and the `<?php echo $iOSname;?>` parameter that will be explained in the point 4.1.2. As can be seen in Listing 7, now the file is reached at the `/livepkg` server location, where the created streaming is being sent.

However, this solution may just work for desktop browser reception, even when it is made for iOS devices. A satisfactory solution is to link directly to the server content, with the `<a>` command as Listing 8 shows.

```
<a href="http://flash.metropolia.fi/hls-live/livepkgr  
/_definst_/liveevent/<?php echo $iOSname;  
?>.m3u8?adbe-live-event=liveevent">Text for the link to  
Apple streaming.</a>
```

Listing 8. Direct link to the content sent to the server. Modified from Sasu Saarnia. [24].

The reason is that for HSL stream, stream location must be provided for the M3U8 file [25].

Now that JW Player can be provided with the parameters it needs, some comments about the code will be given in the next point.

#### 4.1.2 Coding guides

In this chapter, it is important to explain why parameters such as `<?php echo $iOSname;?>` should be in the code.

In the website HTML code should initialize the php function with the command `<? used for opening and closing tags. In this function, the name of the streaming is going to be saved in the variables $streamName and $iOSname, getting into it the parameter “b”. This parameter will come in the browser URL as “?b=streamName”. By initializing the function, $iOSname will contain the stream name that was given according to the step shown in the Figure 21 (page 29). Once the name is stored, the function echo(), which actually does not need brackets (it is not really a function, but a language construct). It “calls” the parameter ordered in its argument, in this case, the string character stored into the variable $iOSname.`

Because of this, the file parameter of the JW Player script will be shown in Listing 9.

```
file:"rtmp://flash.metropolia.fi/livepkgr streamName
?adbe-live-event=liveevent"
```

Listing 9. Hypothetic result of the file parameter.

The code shown in Listing 9 must be preceded by the Listing 10 code.

```
<?php
$streamName = $_GET['b'];
$iOSname = $_GET['b'];
?>
```

Listing 10. Code for initializing *php* function and store in the variables the value of the stream name.

Listing 10 contains also the way to get the stream name and store it into the variables.

Now that these points have been clarified, can the devices receive and play the video? Some images have been taken during the testing process and appear in the next point, 4.2.

## 4.2 Testing results

In this section some pictures will demonstrate the correct work of the process. First, a web page for testing the streaming was created, and the *JW Player* was added to it by the code explained in point 4.1.1. This web page was used for testing the streaming to desktop browsers and to iOS devices, accessing from an iPad. In Figure 26 and Figure 27 there are two screen shots of the web page, showing the *JW Player* working as desired. The web page code can be found in Appendix 3, and it is offered for free use to test the correct working of the *JW Player* and the streaming. It can be used only for academic purpose. It comes with the *style.css* file for the format, and all the instructions are given in the Appendix.

The web page created was allocated at Metropolia server for having access to it from the internet. The process to place it on the server is explained in point 4.3. The URL needed to access the stream is the one shown in Figure 25.

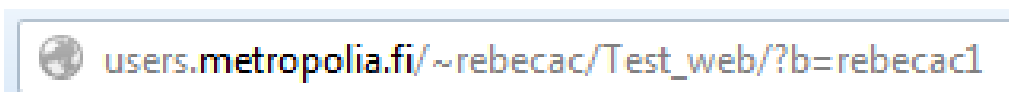
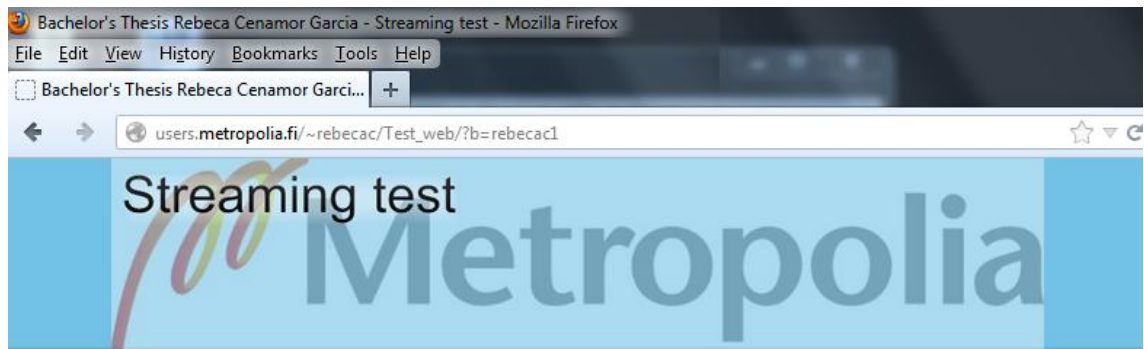


Figure 25. Address box the the URL of the web page.



Specified on the URL shown in Figure 25, appears the stream name used on the Adobe Flash Media Encoder set up, on Figure 21, “rebecac1”.



Test browser streaming



Test iOS streaming



Click for Apple HLS

[Apple livestream](#)

Figure 26. Screenshot of the test web site loading the video.

In Figure 26 the JW Player is loading the content sent to the server. The name of the streaming must be added to the URL as shown in Figure 25, to the proper work of the

function shown in Listing 10. With the command `?b=streamName` it has been told to the JW Player scripts that the name of the streaming was “rebecac1”.

#### Test iOS streaming



[Click for Apple HLS](#)

[Apple livestream](#)

Figure 27. Zoom in into the *JW Player* box when the streaming is loaded.

In Figure 27 the JW Player receives the sent signal. As expected, it is shown on the player established initially for iOS, which means that the `/livepkg` server section is receiving properly the stream. It can be seen the link placed for accessing to the Apple HSL from an iOS device, generated with the code shown in Listing 8. This link was tested, and the result appears in the next Figure 28.



Figure 28. iPad view with the received streaming.

The process passed the final test, and with the next Figure 29 this section concludes.



Figure 29. Complete example of a streaming rehearsal from Dolphin Studio. Picture made by Sasu Saarnia.

This picture resumes quite well the whole process to delivery TV content through the three ways available at Dolphin Studio at the Leppävaara campus of the Metropolia University of Applied Sciences. The screen allocated at the top left side is the one sent and received via DVB-C method. The one at the top right side is received by the *Amino* device and corresponds to the IPTV method. At the left bottom side there is a mobile device, and at the right bottom side the desktop browser is being shown. The middle monitor, as was already explained in Figure 5 (page 7), shows (from top left to the opposite corner in zigzag order) the PVW signal selected from input 1 which is the signal received from the cameras allocated at Lagoon Studio showing a TV set; second, PGM signal with input 7 selected where a pre-recorded insert is delivered; third, previews of the eight inputs, which are: Lagoon Studio cameras, pre-recorder insert hosted on the server, Hot Studio camera, monitoring system, colour bars for equipment adjustment, the insert sent as program signal and another monitoring display for checking the signal. There are two more monitors, one showing the exact time synchronised with all the studios where the time can be seen, from every one of the studios, the broadcast state (off air, on air, rehearsal), and the studio computer with the Adobe Flash Live Encoder opened and delivering the program.

When the study was been conducting, some issues were detected. These problems will be presented in the next chapter 5.

#### 4.3 Host the web page for testing

In this section a short guide for placing a web site at the Metropolia server will be given with the example used on the previous section.

The first step is opening the program WinSCP from a computer in the Metropolia Network. It will appear the window shown in Figure 30.

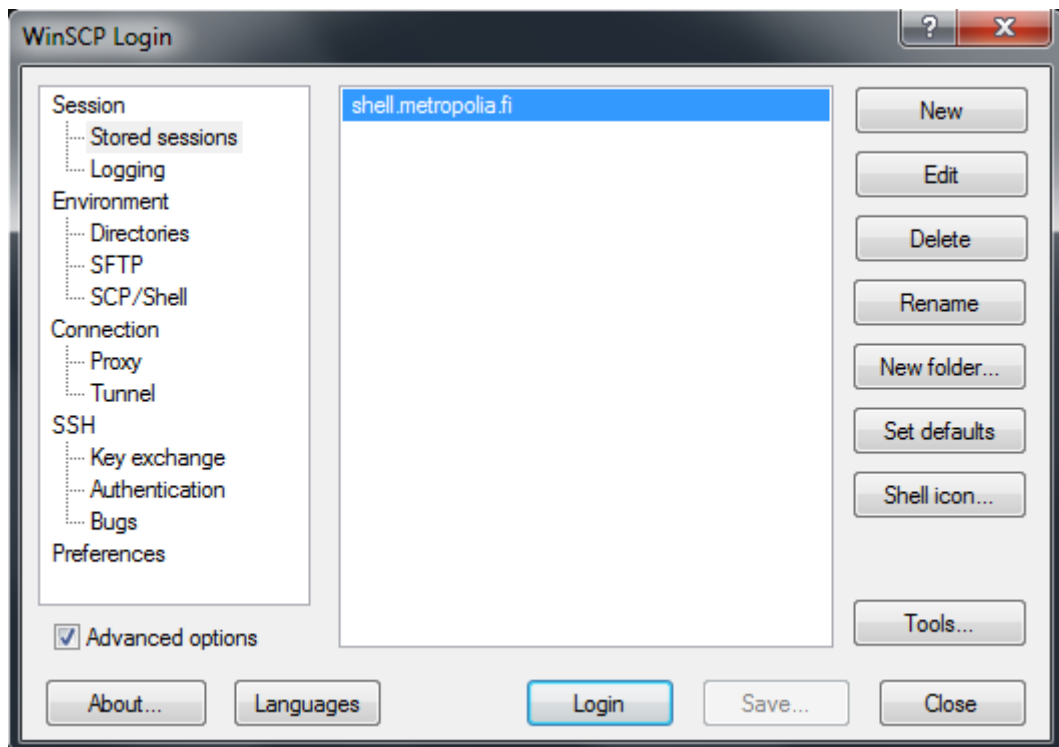


Figure 30. WinSCP window for starting the login.

Figure 30 demonstrates the Metropolia server. After clicking “Login” it will ask the users’ name and password, as shown in Figure 31.

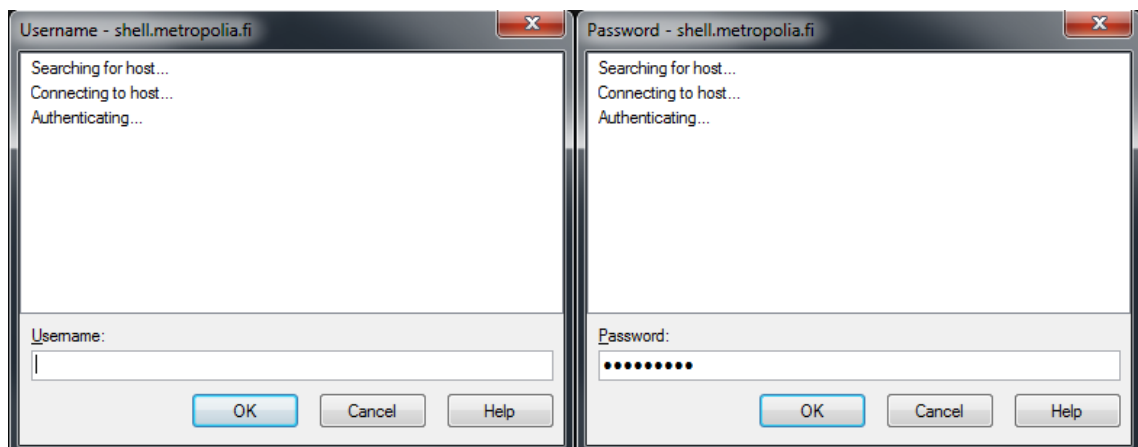


Figure 31. WinSCP windows for introducing Metropolia username and password.

After introducing the Metropolia account information, the server window will appear with all the content allocated in the user’s folder. The important folder is *public\_html*. In the

next Figure 32 a screenshot of the program and the location of the *public\_html* folder can be seen.

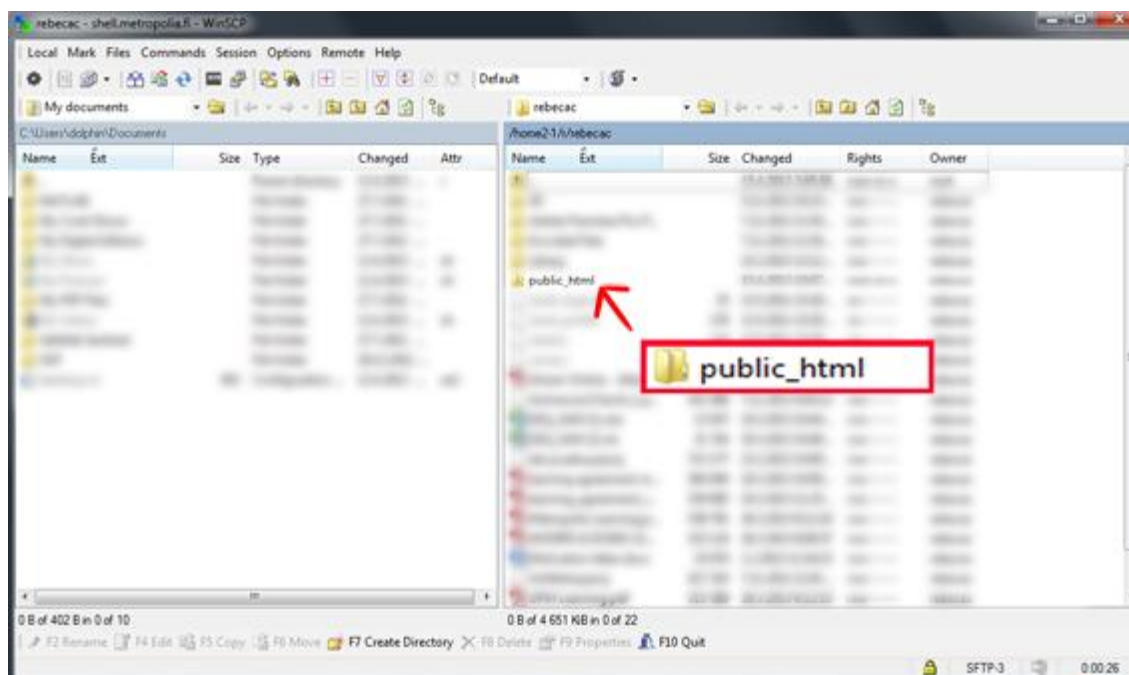


Figure 32. WinSCP window with the user folder at the Metropolia server.

The web site folder must be copied inside *public\_html*. Once it is done, from a browser it is just necessary to write the URL shown in Figure 25, being the generic one: *users.metropolia.fi/~userName/webFolder/?b=streamName*

## 5 Discussion

The streaming has been made now, but, what happens with other mobile operating systems, for example, *Android 4* and later versions, since they do not support Adobe Flash anymore?

This streaming was made with a system Android does not support, and iOS was not supporting the flash streaming for a long time. After the controversy raised by the Steve Jobs' criticism to Adobe, more and more developers started thinking the same: Flash is a closed system and is getting a bit obsolete. [10]. With the HTML5 birth, doors have opened to applications supported by most of the systems. It is yet a protocol under development, and it was not accepted by all the important content providers, but it has

a strong potential and this will bring HTML5 the opportunity to grow and spread with considerable rapidity. However, Adobe still has the majority of the market share and it will be hard to push their system out of the spotlight. Most of the people will continue owning computers with Flash plugins installed, and the most probable future will be a perfect communion of both systems, as it has been for some time.

The main problem appears on portable devices, because the low performance and power they have. It is more difficult to reach the same effectiveness as desktop systems have, also because of the mobile network speed in comparison with common home ADSL connection. But the reality is that mobile technology is developing amazingly fast, and nowadays it is relatively easy to find mobile devices with more computational power than many of the computers that were being used for 5 years. This means that some decisions must be taken concerning developing software for mobile devices, but the capacity they can have from now on should not be underestimated.

Other problem, as could be deduced from the point 4.1.1, is that when placing the player on a web site, so many considerations must be taken. The written code must work in any case, and sometimes it is not an easy task, because of the freedom that exists choosing the system by providers and users. By using widely spread formats, this project has tried to not incur into the problem.

## **6 Conclusions**

Directly linked with the previous chapter, the first conclusion can be taken. Was the objective of the project fulfilled? As a reminder, the main objective of this project was to define a streaming service that satisfies most of the users, by sending the TV content in a way that can be received by the majority. In the point 4.2 *Testing results* it is shown how the streaming works in browsers and in Apple devices. So, basically, each device with a browser installed and the Flash plugin installed can have access to the content, as well as Apple devices. By the HTML5 use, the current problem with Android 4 devices could be solved, and JW Player is supporting this protocol, in a way that can be easily adjusted in the future.

As additional support, in Appendix 3, is attached the full script for a simple website example to test the results.



## References

- 1 Market Share Statistics for Internet Technologies. Net Market Share [online]. California; March 2013.  
URL: <http://netmarketshare.com/> . Accessed 28<sup>th</sup> of March 2013.
- 2 Aalto, E. Hybrid Systems - Production, Value-chains and Technologies in Media Publishing course project (Information) [pdf format]. Espoo (Finland) 2012.
- 3 Aalto, Erkki. New Electronic Media. 2016 Foresight. VTT Technology 31. 2012. [pdf format]  
URL: <http://www.vtt.fi/publications/index.jsp>
- 4 Ortiz Berenguer, Luis I. TV Digital: MPEG2 y DVB. 3<sup>rd</sup> Edition. Dpto. de Publicaciones de la E.U.I.T. de Telecomunicación, Madrid 2003.
- 5 Díaz López, José Manuel. Vídeo en Multimedia [lectures material]. Chapter 5 (Video over the Internet). Madrid 2011.
- 6 Simpson, Wes. Video over IP: IPTV, Internet video, H.264, P2P, web RV, and streaming: a complete guide to understanding the technology. 2<sup>nd</sup> edition. Oxford, Elsevier/Focal Press. 2008
- 7 Adobe Systems. Adobe Flash Platform runtimes. Statistics: PC penetration [online]. Millward Brown survey; July 2011.  
URL: <http://www.adobe.com/la/products/flashplatformruntimes/statistics.html> .  
Study methodology URL: <http://www.adobe.com/la/products/flashplatformruntimes/statistics.displayTab4.html> . Accessed 28<sup>th</sup> of March 2013.
- 8 Richter Stefan, Ozer Jan. Hands-On Guide To Flash Video: Web Video and Flash Media Server. United States of America: Elsevier Inc.; 2007. Chapters 10, 16 and 17.

- 9 Adobe Systems. Adobe Flash Player Help/Release notes. Flash Player 10.1 [online].  
URL: <http://helpx.adobe.com/flash-player/release-note/release-notes-flash-player-10.html> . Accessed 29<sup>th</sup> of March 2013.
- 10 Jobs, Steve. Thoughts on Flash. [online letter]. April 2010.  
URL: <http://www.apple.com/hotnews/thoughts-on-flash/> . Accessed 5<sup>th</sup> of April 2013.
- 11 Adobe Systems. Adobe Flash Platform, ActionScript 3.0 Developer's Guide. Understanding video formats for Flash Player 9 and later and Adobe AIR 1.0 and later [online].  
URL:[http://help.adobe.com/en\\_US/as3/dev/WS5b3ccc516d4fbf351e63e3d118a9b90204-7d46.html](http://help.adobe.com/en_US/as3/dev/WS5b3ccc516d4fbf351e63e3d118a9b90204-7d46.html) . Accessed 29<sup>th</sup> of March 2013.
- 12 Adobe Systems. Programming ActionScript 3.0, Flash Player APIs. Basics of working with bitmaps [online].  
URL:[http://livedocs.adobe.com/flex/3/html/help.html?content=Working\\_with\\_Bitmaps\\_02.html](http://livedocs.adobe.com/flex/3/html/help.html?content=Working_with_Bitmaps_02.html) . Accessed 29<sup>th</sup> of March 2013.
- 13 Apple Inc. iOS Technology Overview. iOS Developer [pdf]. California 2012.  
URL:<http://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf> . Accessed 28<sup>th</sup> of March 2013. p. 22-24
- 14 Apple Inc. Apple TV Specifications. [online].  
URL: <http://www.apple.com/es/appletv/specs.html> . Accessed 2<sup>nd</sup> of April 2013.
- 15 Apple Inc. HTTP Live Streaming [online]. 2012  
URL: <https://developer.apple.com/resources/http-streaming/> Accessed 2<sup>nd</sup> April 2013.
- 16 Pohlmann, Ken C. Principles of Digital Audio, 5<sup>th</sup> Edition. McGraw-Hill, April 2005. Chap. 11 (354).

- 17 Aalto E. Master Control Room diagram [pdf format].
- 18 Matrox digital Video Solutions. Products, Matrox MXO2 LE Tech. Specs. [online].  
URL: [http://www.matrox.com/video/en/products/mxo2\\_le\\_max/specs/](http://www.matrox.com/video/en/products/mxo2_le_max/specs/) .  
Accessed 10<sup>th</sup> of April 2013.
- 19 Matrox digital Video Solutions. Products, Matrox MXO2 LE Connections. [online].  
URL:[http://www.matrox.com/video/en/products/mxo2\\_le/io\\_connections/#close](http://www.matrox.com/video/en/products/mxo2_le/io_connections/#close) . Accessed 10<sup>th</sup> of April 2013.
- 20 Adobe Systems. Adobe Flash Media Server 4.5 Help/Stream live media (HTTP). [online].  
URL:<http://helpx.adobe.com/flash-player/release-note/release-notes-flash-player-10.html> . Accessed 10<sup>th</sup> of April 2013
- 21 LongTail Ad Solutions. Products. JW Player. Tech Specs. [online].  
URL: <http://www.longtailvideo.com/jw-player/tech-specs/> . Accessed 10<sup>th</sup> of April 2013.
- 22 LongTail Ad Solutions. Support. JW Player. Embedding the Player. [online].  
URL: <http://www.longtailvideo.com/support/jw-player/28839/embedding-the-player> . Accessed 11<sup>th</sup> of April 2013.
- 23 LongTail Ad Solutions. Support. JW Player. Using RTMP Streaming. [online].  
URL: <http://www.longtailvideo.com/support/jw-player/28854/using-rtmp-streaming> . Accessed 11<sup>th</sup> of April 2013.
- 24 Saarnia, Sasu. Index HTML code for his web site. [text file]. 2013

- 25 LongTail Ad Solutions. Support. JW Player. Using Apple HLS Streaming. [online].  
URL: <http://www.longtailvideo.com/support/jw-player/28856/using-apple-hls-streaming> . Accessed 12<sup>th</sup> of April 2013.

## Literature

Richter, S; Ozer, J. Hands-On Guide To Flash Video: Web Video and Flash Media Server. United States of America: Elsevier Inc; 2007.

Apple Inc. Technical Note TN2224. iOS Developer Library [online]. 14 August 2012.  
URL: <https://developer.apple.com/library/ios/#technotes/tn2224/index.html> Accessed 2<sup>nd</sup> April 2013.

Apple Inc. HTTP Live Streaming [online]. 2012  
URL: <https://developer.apple.com/resources/http-streaming/> Accessed 2<sup>nd</sup> April 2013.

Apple Inc. iPhone iOS [online]. 2012  
URL: <http://www.apple.com/iphone/ios/> Accessed 8<sup>th</sup> April 2013

Adobe Systems. Adobe Flash Player Help/Release notes. Flash Player 10.1 [online].  
URL: <http://helpx.adobe.com/flash-player/release-note/release-notes-flash-player-10.html> . Accessed 29<sup>th</sup> of March 2013.

Ortiz Berenger, Luis I.; Rodríguez Vázquez José L. Ingeniería de vídeo en entornos digitales. Dpto. de Publicaciones de la E.U.I.T. de Telecomunicación, Madrid 2008.

Luther, Arch; Inglis Andrew. Video Engineering. 3<sup>rd</sup> edition. McGraw Hill Professional, 1999.

Ortiz Berenger, Luis I. TV Digital: MPEG2 y DVB. 3<sup>rd</sup> edition. Dpto. de Publicaciones de la E.U.I.T. de Telecomunicación, Madrid 2005.

Ortiz Berenger, Luis I.; Blanco Martín, E. Comunicaciones en Audio y Vídeo. 2<sup>nd</sup> Edition. Dpto. de Publicaciones de la E.U.I.T. de Telecomunicación, Madrid 2008.

Díaz López, José M. Vídeo en Multimedia [lectures material]. Chapters 4 (Interactive Television, Content creation, Broadcast Server, Transmission, Receptors) and 5 (Video over the Internet, IT Network, Streaming, Webcasting). Madrid 2011.

Pérez Fuentes, José Antonio. Real-time streaming video from/to mobile devices. [final research project, thesis]. E.U.I.T. de Telecomunicación (read at Mälardalen University – Västerås, Sweden), Madrid 2009.

Austerberry, David. The technology of video and audio streaming. 2<sup>nd</sup> Ed. Focal Press, Oxford, October 2004.

Simpson, Wes. Video over IP: IPTV, Internet video, H.264, P2P, web RV, and streaming: a complete guide to understanding the technology. 2<sup>nd</sup> edition. Oxford, Elsevier / Focal Press. 2008.

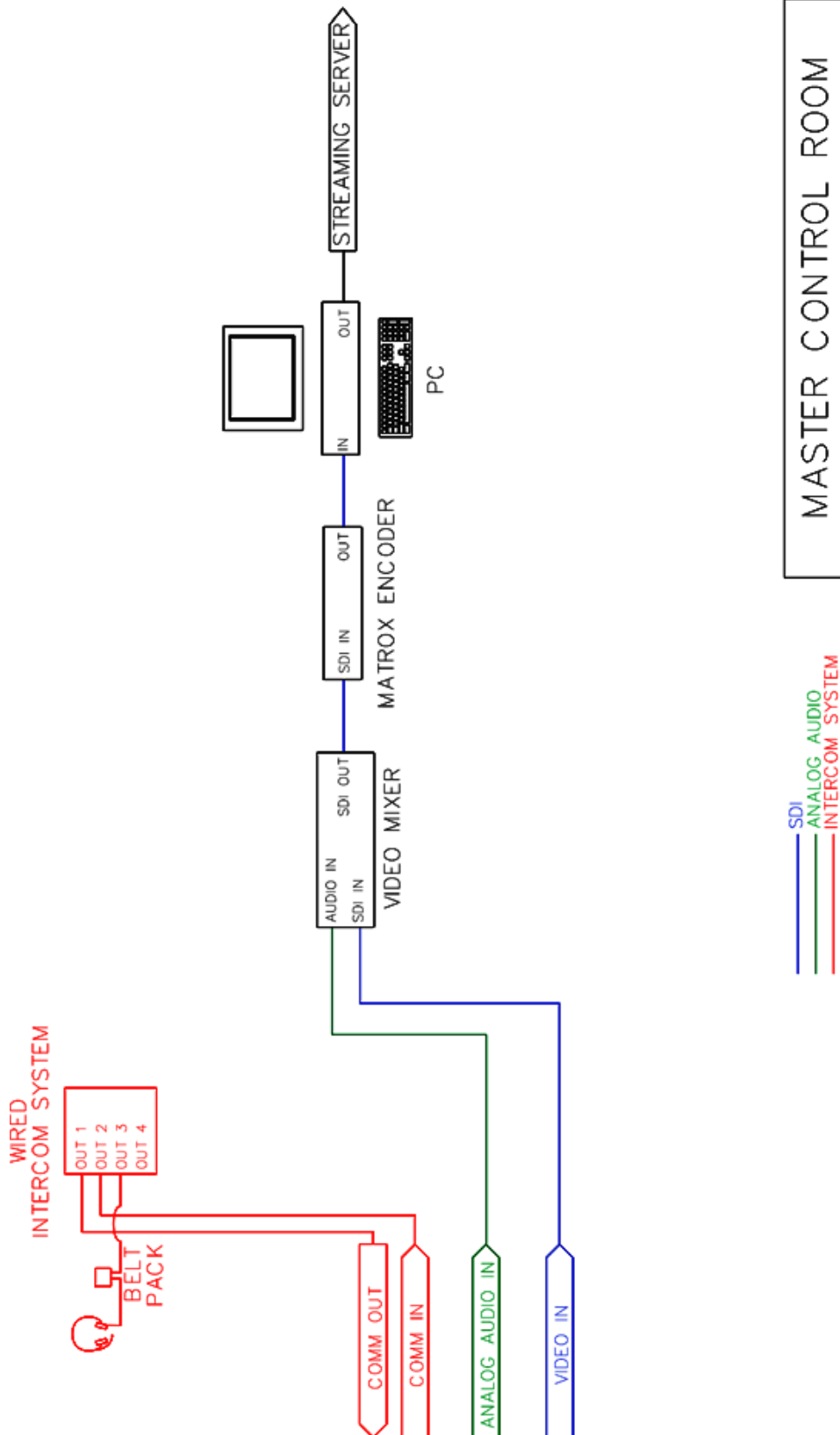
The PHP Group. PHP The manual. 2013. [online].

URL: <http://www.php.net/manual/en/index.php> . Accessed 13<sup>th</sup> of April 2013



## Appendices

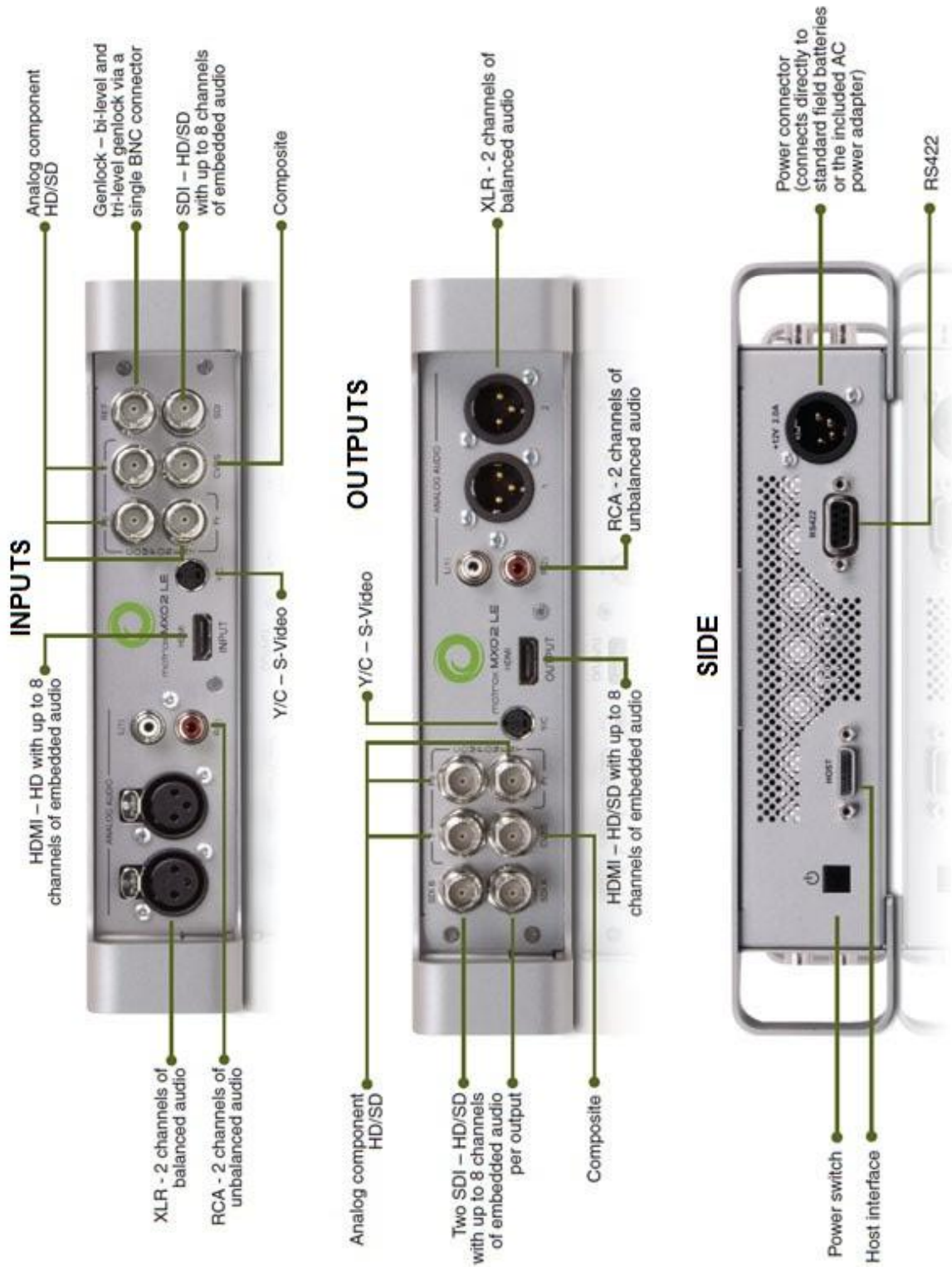
### Dolphin Studio Diagram for streaming [17]







**Matrox MXO2 LE and MXO2 LE MAX Connections [19]**





## Web Script

```
<!--
-----
Autor: Rebeca Cenamor Garcia
Bachelor's Thesis: Aggregating and modifying TV delivery to iOS
devices.
Metropolia University of Applied Sciences, May 2013.

This template is for academic use, for testing the correct
working of the studied process at the thesis.
-----
-->

<!DOCTYPE HTML>

<?php
$streamName = $_GET['b'];
$iOSname = $_GET['b'];
?>

<html>
  <head>
    <title>Bachelor's Thesis Rebeca Cenamor Garcia -
    Streaming test</title>

    <link id="theme" rel="stylesheet" type="text/css"
    href="style.css" title="theme" />

    <script type="text/javascript" language="javascript"
    src="js/addon.js"></script>
    <script type="text/javascript" language="javascript"
    src="js/custom.js"></script>

    <script type="text/javascript"
    src="jwplayer/jwplayer.js"></script>
    <script type="text/javascript"
    src="jwplayer/jwplayer.html5.js"></script>

  </head>

  <body>

    <div id="BannerWrapper">
      <div id="Banner">
    </div>

    <div id="content">
      <div id="main">
        <h3>Test browser streaming</h3>

```

```

<!-- web player -->
<div id="myElement">Loading... </div>
<script type="text/javascript">
    jwplayer("myElement").setup({
        file:"rtmp://flash.metropolia.fi/live/
        <?php echo $streamName; ?>",
        image: "/assets/1.png",
        height: 420,
        width: 720
    });
</script>

</br>

<h3> Test iOS streaming </h3>
<div id="myElement2">Loading... </div>
<script type="text/javascript">
    jwplayer("myElement2").setup({
        file:"rtmp://flash.metropolia.fi/livepkgr/
        <?php echo $iOSname; ?>?adbe-live-
        event=liveevent",
        image: "/assets/2.png",
        height: 420,
        width: 720
    });
</script>

<br/>

<h3> Click for Apple HLS </h3>
<p>
<a href="http://flash.metropolia.fi/hls-
live/livepkgr/_definst_/liveevent/<?php
echo $iOSname; ?>.m3u8?adbe-live-event=liveevent">
Apple livestream</a>
</p>
</br>
</div> <!--main ends-->
</div> <!--main ends-->

</body>
</html>

```

Listing 11. Index script for a simple web page with two media player added and a link to the stream to Apple devices.

For this template to work, a style sheet has used, in css format. It is attached below and must be saved in the same folder than the *index.php* file.

```
/*-----  
Autor: Rebeca Cenamor Garcia  
Bachelor's Thesis: Aggregating and modifying TV delivery to iOS  
devices.  
Metropolia University of Applied Sciences, May 2013.  
  
This template is for academic use, for testing the correct  
working of the studied process at the thesis.  
-----*/  
  
#BannerWrapper {  
    width:100%;  
    height:200px;  
    background-color:#72c2e5;  
    background-image:url(css/shadow.jpg);  
    background-position:center top ;  
}  
  
#Banner {  
    margin:0 auto;  
    width:920px;  
    height:200px;  
    background-image:url(css/MetropoliaBanner.jpg);  
    background-repeat:no-repeat;  
}  
  
#content {  
    margin:0 auto ;  
    width:920px;  
    text-align:left;  
}  
  
#main {  
    margin-right:40px ;  
    width:780px;  
    float:left;  
}  
  
#content a:link, #content a:visited {  
    color:#4556EC;  
    font-weight:bold;  
}  
  
a:link, a:visited {  
    color:#838595;  
    text-decoration:none;  
}  
  
a:hover {  
    text-decoration:underline;
```

```
}  
  
body {  
    color:#444444;  
    font-family: Arial;  
    font-size: 15px;  
    font-style: normal;  
    font-weight: normal;  
    text-transform: normal;  
}  
  
h3 {  
    margin:10px 0 5px 0;  
    color:#000000;  
}
```

Listing 12. Css style sheet that must be saved as “style” in the same folder than the previous Listing 12 was saved as “index”.

In the same folder than *style.css* and *index.html* must be placed a folder named “css” where the following images must be stored for the good looking of the simple web site.



Figure 33. Save this image as *MetropoliaBanner.png* inside the *css* folder inside the web site folder.

Figure 34 image must be placed also in the same folder named as “css”.



Figure 34. Save this image as *shadow.png* inside the *css* folder inside the web site folder.

The directory of the web site folder must have:

- Index.php file (Listing 11)
- style.css file (Listing 12)
- Folder named *css* with the images *MetropoliaBanner.png* and *shadow.png* (Figure 33 and Figure 34)
- Folder named *jwplayer* downloaded in the zip file as described in the point 4.1.1 (page 343).

In Figure 35 is shown how the web folder inside the *public\_html* folder must look, for this example, in the WinSCP program for managing the server folder.

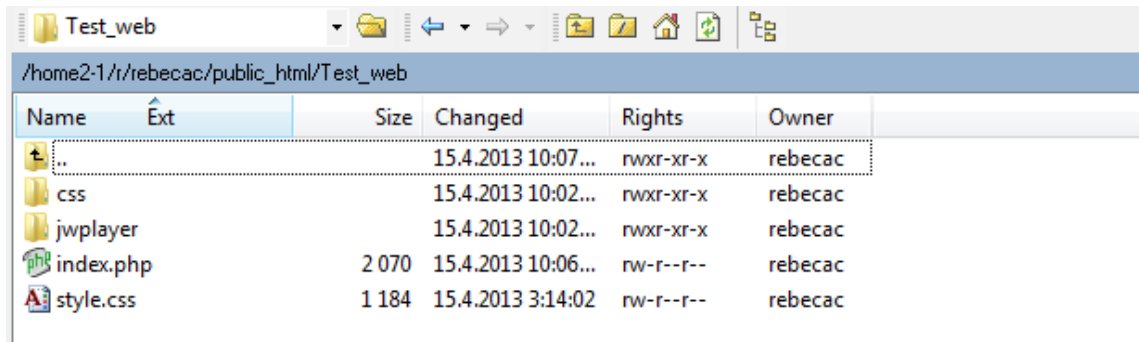


Figure 35. Web folder in the server with all the necessary files.

The web must look as appears in Figure 26.

