

Video streaming

a research on free video streaming technologies

Jaromil
dyne.org / rastasoft.org

Table of Contents

Introduction.....	1
We need CPU for compression	2
We need bandwidth for distribution	2
We need harddisk space for storage.....	3
Distribution of video archives	4
Embedded solutions.....	6
Streaming video software.....	8
About the author of this document	10
Video streaming terms glossary.....	11

Introduction

Given the vast panorama of video technologies available nowadays, this research could fill up way too much paper in the attempt of covering every aspect of this field, instead i'll just try to narrow the focus to certain advanced aspects of streaming video, also trying to give a quick reference guide to the usage of selected free software.

So let's first define the field in which we'll move: *streaming video* provides a continuous digital video and/or audio signal across a data network. As a viewer, you typically make a web browser-based player connection to a streaming server to receive a *webcast* (live program) or *video-on-demand* (previously recorded program). The program is sent from the server to your player in a continuous way, as opposed to having the entire program downloaded before viewing can begin. In fact streaming makes the fruition of content immediate, no copy of the entire program needs to be stored on the computer being used for viewing, it is just about a stream of data being interpreted while received.

The aim of this research is to highlight affordable and reliable compression technologies, transport protocols, software and hardware setups for video streaming.

Emphasis is put on efficient resource allocation in terms of hardware and network usage: in any case this document will privilege the use of older technologies as it is the author's interest to recycle hardware which is often declared obsolete (marketing lie) by proprietary software employed on it.

The computer architecture mostly taken in consideration in this research is the PC ix86 (manufactured by Intel, AMD, Transmeta and VIA among the others) for two main reasons: the Apple platform offers a non free and less flexible platform for anything different from desktop publishing (still its video cutting applications are state of the art, but that's not about video streaming) while it is way less affordable and available in the 'south' of the world, where the author comes from.

Now, given the assumption that professional quality can be achieved using free software, as of the definition given by the Free Software Foundation¹, the main concerns are image quality, audio/video synchronization, efficient network usage, end-user access and flexibility.

An up to date version of this document is made available at the internet address korova.dyne.org/video_streaming² and is downloadable in PDF format from korova.dyne.org/video_streaming.pdf³.

We need CPU for compression

The process of video encoding used to require very expensive gear in the past, while at the time being we can easily think of any user-end system as something capable to compress and stream video online, and thanks to the constant evolution we experience with computers, we can safely expect a wider access to these technology in the future.

A rough picture of a computer capable to handle live video encoding at full PAL resolution is a 1Ghz Pentium3 with 512MB of RAM, anything more is better; in particular the Pentium4 (as well processors from the same "generation") improves much the latency and the performance thanks to its special extended instructions⁴ made to deal exactly with realtime video processing.

Still, the quality achieved by a fast CPU is often made unnecessary: unless we really need superior quality (for certain artistic reasons for instance) we can lower our profile and settle on a broadcast quality level, which may vary in its definition.

In fact for content like interviews the video is way less important than the audio in vehicling the message, therefore its quality can be low, saving resources and widening the potential pool of listeners. Again in video conferencing the low detail of a figure speaking can be alternated to the higher detail of a still image (better if in vectorial format) accessed from the web at the right time - and such a setup can be also remotely controlled with tools like VNC or using the UNIX X Windows environment itself - whereas to stream everything via encoded video would be a sinful waste.

Such concerns about the kind of contents we are going to make available are of crucial importance when planning a succesful distribution of audio and video materials among a large audience, while they greatly affect our requirements in terms of CPU, as well network usage. So let's go talking about network.

1. <http://www.gnu.org>

2. http://korova.dyne.org/video_streaming

3. http://korova.dyne.org/video_streaming.pdf

4. on the PC, the SSE and SSE2 instruction sets can be considered as an evolution of the MMX set, also implementing accelerated 64bit floating point instructions, see the documentation on tommesani.com/Docs.html

We need bandwidth for distribution

A crucial parameter in network streaming is, of course, bandwidth. That is the amount of traffic generated by a stream, which strictly depends from our connectivity. It is directly proportional to the quality and resolution of images, therefore the general rule is: the more quality you need, the more bandwidth you need.

In case the connection available is only a dialup modem we have a very tight bound for doing video streaming: the best solution is to use lower frame rates (1 frame per second) and a smaller image canvas of 320x240 pixels (approximately half PAL/NTSC resolution) or below.

With higher connection rates (cable modem, ADSL or if we are lucky T1 class connectivity) we can raise our image quality up to 24fps and larger sizes, keeping the balance between those two parameters.

A significant parameter is the geographical proximity to the stream listeners we want to reach: over long distances our transmission will be limited by the network nodes we are crossing and by the connectivity available to the people watching the stream; while over shorter distances (our same urban or national area, or within our specific Internet Service Provider) we might take advantage of a even larger bandwidth than what is nominally available to us.

In any case an empirical test in collaboration with somebody taking up our stream from the place where it should be received will give us a clear idea of the results of our efforts.

The measures used for bandwidth are two: MegaBytes per second (MB/s) and Megabits per second (Mb/s). They are equivalent and can be converted from one another thru a simple calculation: one MegaByte is 8 Megabits. These measures define the throughput of data required by a correct visualization of the video stream, as well for its correct transmission, therefore we are talking about an amount of bandwidth which is necessary on both ends of the streaming flow.

A rough estimation of the compression/storage ratio obtained by a common MPEG2 encoding is 1.8Gbytes per hour, for instance to store a full screen DVD in average quality on your harddisk. In terms of realtime streaming, this means about 4 Megabits per second (4Mb/s), which can also be quantified as 582 KiloBytes per second (582KB/s).

here is how some of the above calculations are done:

1. $2\text{GigaBytes of stored video} / 60 \text{ mins} / 60 \text{ secs} = 596523 \text{ bytes per sec}$
 - a. $596523 \text{ bytes per sec} / 1024 = 582 \text{ KB/s (Kilobytes per sec)}$
2. $596523 \text{ bytes per sec} * 8 = 4772184 \text{ bits per sec}$
 - a. $4772184 \text{ bits per sec} / 1024 / 1024 = 4 \text{ Mb/s (Megabits per sec)}$

Full quality encoding it is often meaning a bitrate of 6Mbit per second (768KB/s), which walking back thru the above calculus leads to a compression/storage ratio of 2.6Gbytes per hour on your harddisk.

As we can see the bitrate is directly proportional to the space required to store the video stream. Let's focus now to the aspect of storing the video we are going to stream, or which is in turn streamed to us.

We need harddisk space for storage

When thinking about video storage there is a fundamental distinction we have to do between compressed and uncompressed video: the first is video encoded in a form ready to be streamed and which shouldn't be edited anymore because of loss of quality at every reencoding⁵; the second is often occupying 5 to 10 times more space and usually employed for video cutting.

The harddisk is a crucial device when we have to acquire video from video sources like composite video or firewire. The harddisk is not involved in the operation only in case we are doing live stream and we don't want to store anything we broadcast; while often there is the necessity to acquire and store the uncompressed video to later cut it, which requires a sufficiently fast and large harddisk. In case we don't need to cut the video we acquire and we have a fast CPU, then we can compress on the fly the acquired video signal making CPU intensive compression operations on volatile memory before storing it, saving a lot of overhead to the harddisk which then doesn't need to be especially fast or big.

Anyway in case your harddisk or your CPU is too slow or you don't have firewire equipment, you shouldn't get desperate: you can still use a middle-layer lossless codec for video acquisition before compression. It is about a different codec to store the video data before the process of encoding it in compressed form: a state in between the raw video and the fully encoded video. When such a codec is lossless it can then be a suitable source for video editing, it will require a lighter computation than MPEG2 and MPEG4 while occupying more space on your harddisk. Such an effective solution is implemented by a free software codec named NuppleVideo⁶ developed in Vienna by Roman Hochleitner and supported as an input format by most of the software encoders available on GNU/Linux.

Thanks to this software, a 800Mhz CPU with a fairly modest harddisk (as of models sold on the consumer market in 2001) can function as a MPEG2 or MPEG4 encoding station, acquiring video from composite or s-video using a cheap WinTV/Hauppauge card.⁷

Distribution of video archives

Before stepping into a more specific view on hardware and software solutions, this research will analyze a situation for video streaming technologies, envisioning a possible *real life* application for video streaming, other than the well known scheme of broadcast one-to-many.

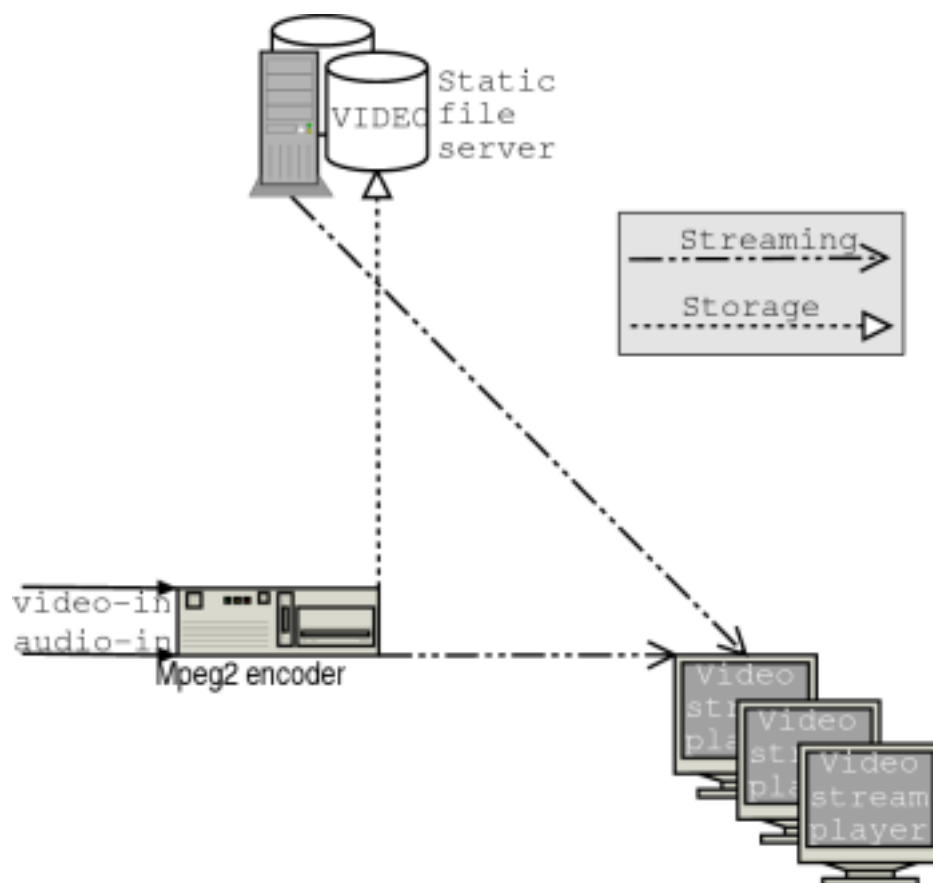
Let's assume a scenario in which we need to import high quality video (MPEG2 6-8Mb/s) storing it into our video repository to make it available to different players on the same network; live streaming is required as well, to feed players with some non-stored video which gets compressed on the fly.

This is in fact a need for Montevideo's exhibitions, for instance.

5. that's why most compression algorithms are defined lossy.

6. <http://frost.htu.tuwien.ac.at/~roman/nupplevideo>

7. Brooktree or Conexant chipset, ~50 to ~120 euros on the market



here is a diagram of our high quality video network

In such a situation we distinguish 3 fundamental components of this workflow: the *Storage*, the *Encoder* and the *Players*.

The *Storage* is where all the recorded video is stored, it is capable of streaming on demand its "static" files to *Players* and it offers write access to its storage for the *Encoder*.

The *Encoder* can get input from other audio/video devices and will transform it into MPEG2 format, perfect for high quality streaming and storage; the result of the transformation can be saved in the *Storage* and/or streamed to *Players*.

The *Players* can connect to a live stream on the *Encoder* or play a file from the *Storage*, thus being able to show archived videos as well videos which are not stored but realtime converted into a playable format.

This setup eases the displaced reproduction of videos on a local network, keeping all the videos stored in a central server, thus simplifying the backup of the stored videos. Last but not least it offers a wide range of possibilities to customize and automatize the playback of the *Players* for applications like distributed video jukebox and kiosks for browsing digital video bibliotèques, being also a scalable solution for *Storages* which are likely to grow in future.

Free software implementing the *Storage* and *Player* functions is already available: *videolan* is the video streaming client/server framework which should be used on every component of our setup, in order to let the MPEG2 video being streamed to *Players* across the network. But it's still missing a software to implement some of the *Encoder* functionalities: a way to remote control its actions and to send the

encoded video to the Storage as well as streaming it to Players.

Embedded solutions

Dynebolic GNU/Linux bootable CD

Dynebolic GNU/Linux is an operating system working directly from the CD, without the need to install or change anything on harddisk. It is an embedded solution which can easily adapt any PC computer to be a fully configured workstation ready to achieve most of the tasks being discussed here and equipped with most of the software we are going to mention in the next chapters.

This operating system focuses on multimedia production, offering tools for audio and video recording, editing, playing and live manipulation; being all compiled from scratch, it takes advantage of specific binary optimizations to achieve superior performance over other generic purpose GNU/Linux distributions, often resulting in smoother processing and responsiveness on older hardware.

In fact, keeping our focus on the practice of 'recycling' hardware, dynebolic has a basic requirement of a PC with 200Mhz CPU and a IDE CDRom player: on such a setup it can be easily used as a comfortable surf station. When runned on a 400Mhz CPU it can be a perfect stream box for the encoding and broadcasting of a internet radio station; while it can turn a 800Mhz CPU machine into a perfect MPEG4 audio/video streaming box.

All the above tasks can be solved using dynebolic without requiring any particular technical knowledge and using a low profile setup: devices for acquisition of audio and video for the tasks mentioned above range from cheap webcams and TV cards to normal audio cards with mic/line input.

Being GNU GPL free software, dynebolic is free to be redistributed and is made available for internet download (as a burnable ISO raw cd image file) from its website dynebolic.org⁸ along with detailed documentation. You are very welcome to join and share your experience with the enthusiastic community of users and developers around it, gathered together in the *dynebolic@dyne.org* mailinglist.

Multipurpose mini-ITX box

The thaiwaneese manufacturer VIA is producing in its EPIA class products some very interesting motherboards called Mini-ITX⁹. They are very compact and customizable computers which can be adapted to serve very well the various purposes of encoding, streaming and playing high quality video transmitted thru the net.

In particular the VIA EPIA board called Nehemiah M10000 includes TV-OUT and a onboard chip for MPEG2 decoding¹⁰: it is perfect for playing high quality streams and DVD movies, while still being a fully functional desktop computer supported by GNU/Linux softwares. The support for MPEG2 hardware decoding on video-out is implemented by the VIA enhanced Xine Player¹¹ project.¹²

8. <http://dynebolic.org>

9. <http://www.mini-itx.com>

10. the CLE266 functionality has been reverse engineered in a V4L module to make it work with Linux kernels, see <http://www.ivor.it/cle266>

11. <https://sourceforge.net/projects/viaexp/>

12. 2.4.23 linux kernel has specific support for EPIA architecture

Standalone video stream player

Other interesting embedded devices for video streaming are those produced by Amino, a company based in Swavesey, north of Cambridge. They are focusing on the so called *IPTV*¹³ market by producing integrated solutions based on Linux technology, not as easy to be customized for various needs but surely very compact and efficient. In particular the AmiNET110¹⁴ is a very small and compact streaming video player¹⁵ which is apparently going for consumer grade production: it can play MPEG2 RTSP streams from the network and is fully compatible with the Videolan streaming server, a rough estimation for its price floats around 150EU.

Hardware MPEG2 Encoder

The Hauppauge PVR250 and PVR350 are MPEG2 encoder cards are fully supported on the GNU/Linux platform¹⁶¹⁷ by the IVTV Linux driver module and so far can be employed using two softwares: MythTV¹⁸ and Videolan¹⁹. A simple and efficient software to do MPEG2 capture from the card is still missing, therefore the best way to do that still seems to be the commandline technique:

```
~ #cat /dev/video0 > capture.mp2
[Enter] and CTRL-c to stop
```

The documentation on how to setup this card in GNU/Linux is fairly extended, the best and most direct tips are found on the IVTV wiki pages²⁰, both for PAL and NTSC setup of the card.

Example 1. setup for PAL video input MPEG2 encoding

compile and install the ivtv kernel module, then copy the following lines in your /etc/modules.conf

```
alias char-major-81 videodev
alias char-major-81-0 ivtv
alias /dev/v4l ivtv
options ivtv debug=0 ivtv_pal=1
options tuner type=5 pal=1
options saa7127 enable_output=1 output_select=0 pal=1
options msp3400 once=1 simple=1
add below ivtv msp3400 saa7115 saa7127 tuner
add above ivtv ivtv-fb
```

then load the module with a simple *modprobe ivtv*, the /dev/video0 device will give out encoded video with a simple "cat" command, while the /dev/fb0 device will be decoding MPEG2 and can also be used as an additional framebuffer display device.

13. Internet Protocol TV

14. <http://www.aminocom.com/products/aminet110.html>

15. Specs: MIPS 350 CPU with Flash memory, 10/100Eth, MPEG2 8Mb/s decoder, composite video out to 4:3 and 16:9 formats.

16. <http://ivtv.sf.net/tiki/tiki-index?page=SupportedHardware>

17. most probably the IVTV driver will be included in the 2.6 series Linux kernel.

18. <http://mythtv.sf.net>

19. <http://www.videolan.org>

20. <http://ivtv.writeme.ch/tiki-index.php>

The PVR price floats around 240EU while they have many of the features of other expensive competitors. The compressed video can be acquired either from live video plugged into the card (composite or s-video) or from memory mmap.²¹

High definition TV media player

The Roku HD1000 is a recent product shipped by Rokulabs²² getting on the consumer grade market at the price of 499\$ as the "the world's first high-definition digital media player". The product displays content from memory cards or networked PCs on High Definition TVs (HDTVs) including LCD and Plasma TVs. The HD1000's open, Linux-based architecture makes it especially interesting to embedded hackers, and Roku encourages such use.

It shows up a range of media capabilities comprises digital photos, video, music, and "dynamic media applications"; content is displayed through memory card slots for CompactFlash, MMC, SD, Memory Stick, and SmartMedia. Or, the Roku HD1000 can connect via Ethernet or Wi-Fi to a home network. The device works with any TV, though it was built specifically for HDTVs.

The so called Roku OS is the supposedly open platform that includes Roku's advanced media APIs and the Linux Kernel. Roku says that developers can quickly craft innovative and custom applications that take advantage of the TV-centric user interface elements, network and memory card access, MP3, MPEG, windowing system, graphics library, and other media engines. A C/C++ SDK will be available by the end of the year 2004, but still nothing is there.

What's under the hood? The HD1000 is based on an ATI Xilleon x225 processor which includes a 300MHz MIPS architecture CPU core, 2D and 3D graphics engines, video and graphics scalers, and a high-definition MPEG2 decoder. It comes with 32MB of 133MHz DDR system RAM and 32MB of 133MHz DDR video RAM. The device boots a custom bootloader from a small serial flash, the bootloader then looks for a bootfile (the Linux kernel) on the CompactFlash slot. If it doesn't find one, it boots the Linux kernel from the 16MB internal NAND Flash memory chip.

Streaming video software

Mpeg4lp

MPEG4IP provides an end-to-end system to explore MPEG-4 multimedia. The package includes many existing open source packages and the "glue" to integrate them together. This is a tool for streaming video and audio that is standards-oriented and free from proprietary protocols and extensions.

Provided are an MPEG-4 AAC audio encoder²³, an MP3 encoder, two MPEG-4 video encoders, an MP4 file creator and hinter, an IETF standards-based streaming server, and an MPEG-4 player that can both stream and playback from local file.

21. mmap capability in encoder cards is not always present, it means the possibility to encode video generated from inside the computer (softwares and such), and not from a source plugged via a cable: this is a key feature for transcoding functionalities.

22. <http://www.rokulabs.com>

23. interestingly enough, this is the only GPL'd AAC encoder implementation that i've found around, and is yet not a shared library included in other generic frameworks as transcode: useful component, indeed.

Mpeg4Ip can be recommended for its stability and easy employment, being a simple yet powerful tool to produce MPEG4 streams that can be served by the popular Darwin Streaming Server distributed by Apple. It provides an intuitive graphical user interface and can record on harddisk while stream at the same time, the produced streams can be played on most platforms by commonly found players, while mpeg4ip itself provides players for multiple platforms. A remarkable feature is also the complete support for RTP/RTSP and Multicast protocol.

VideoLan

The VideoLAN project targets multimedia streaming of MPEG-1, MPEG-2, MPEG-4 and DivX files, DVDs, digital satellite channels, digital terrestrial television channels and live videos on a high-bandwidth IPv4 or IPv6 network in unicast or multicast under many OSes.

VideoLAN also features a cross-platform multimedia player, VLC, which can be used to read the stream from the network or display video read locally on the computer under all GNU/Linux flavours, all BSD flavours, Windows, Mac OS X, BeOS, Solaris, QNX, Familiar Linux...

The latter feature is one of the most important points for VideoLan, being in fact the only free software being able to fill the gaps between video and audio formats on different platforms, while it's even supporting ARM and MIPS based handheld devices, being ready to be integrated in consumer-grade embedded solutions.

VideoLan supports the functionalities offered by the IVTV driver, taking advantage of hardware encoding on a growing number of cards²⁴. It is basically made out of two software components: VideoLanClient (VLC) and VideoLanServer (VLS). VLC is a multipurpose streaming a/v client and source: it can play streams as well it can capture and send a stream to another VLC or VLS. VLS is just a server which has no visualization output for the streams it handles, its fully capable of capturing and streaming from the local machine, as well reflect streams coming from other VLC/VLS nodes.

The way VideoLan distributes functionalities among its nodes is therefore very flexible and permits to easily build streaming topologies to distribute realtime audio/video streams for various needs.

Video conversion: Transcode

Transcode is a linux text-console utility for audio/video stream processing and transcoding between different formats and codecs, complete documentation and sourcecode are available on it's homepage²⁵.

Decoding and encoding is done by loading modules that are responsible for feeding transcode with raw video/audio streams (import modules) and encoding the frames (export modules). It supports elementary video and audio frame transformations, including de-interlacing or fast resizing of video frames and loading of external filters.

A number of modules are included to enable import of DVDs on-the-fly, MPEG elementary (ES) or program streams (VOB), MPEG video, Digital Video (DV), YUV4MPEG streams, NuppelVideo file format and raw or compressed (pass-through) video frames and export modules for writing DivX;-), DivX 4.02/5.xx, XviD, Digital Video, MPEG-1/2 or uncompressed AVI files with MPEG, AC3

24. PVR250/350, Yuan MPG600/160, Avermedia M179 and probably more in future, see <http://ivtv.writeme.ch/tiki-index.php?page=SupportedHardware>

25. <http://zebra.fh-weingarten.de/~transcode>

(pass-through) or PCM audio. More file formats and codecs for audio/video import are supported by the avifile library import module (capable of using WIN32 codecs directly from .dll), the ffmpeg codec library (MPEG4, DivX family codecs, Real) while limited Quicktime export support and DVD subtitle rendering is also available.

In particular it can be useful for MPEG4 compression, realized thru the FFMpeg module. It supports various m\$mpeg and divx formats, as well H.263. While unfortunately is still lacking support for the H.264 video codec and the AAC audio codec, due to patent pending. Honestly enough, i think that even if more efficient than other codecs, those proprietary solutions will have less and less ground for consumer grade diffusion if they will not open their policies: right now it's only possible to play H.264/AAC encoded MPEG4 files on Mircosoft Windlows platform.

Example 2. commandline video format conversion

convert videos from MPEG2 to MPEG4, resizing to half-PAL size, encode with MP3 64Kbit/s audio and WindowsMediaPlayer compatible 300Kbit/s video:

```
~ #transcode -i mpeg2/video.mpg \
    -y ffmpeg -F msmpeg4 \
    -Z 300x240,fast -w 300 \
    -v -N 0x55 -b 64 \
    -o mpeg4/video.mpg
```

special care has to be taken with the -Z resizing flag, which has to preserve the screen size ratio in order to avoid interlaced movements in the transcoded video. More information can be obtained by reading the *transcode --help | more* and the *man transcode* help and manual page; useful is also the GTranscode²⁶ graphical interface to Transcode which helps to form commandline formulas by schematizing the wide range of possibilities being offered.

Transcode was originally written by Thomas Oetreich and is now mantained by Tilmann Bitterberg, it links several different libraries and includes code contributed by many different researchers and programmers since june 2001.

About the author of this document

Jaromil aka Denis Rojo is the author and mantainer of GNU GPL free software available on dyne.org²⁷, more specifically HasciiCam²⁸ (ascii video streaming software), MuSE²⁹ (network audio streaming software for online radios), FreeJ³⁰ (realtime video manipulation framework) and dyne:bolic³¹ (a compact and easy to use GNU/Linux bootable CD distribution for multimedia production). His software is included into various GNU/Linux and BSD distributions.

26. <http://fuzzymonkey.org/newfuzzy/software/granscode>

27. <http://dyne.org>

28. <http://ascii.dyne.org>

29. <http://muse.dyne.org>

30. <http://freej.dyne.org>

31. <http://dynebolic.org>

All his productions, and further informations about him, are available on his homepage RASTASOFT.org³².

This document was commissioned by the Nederlands Instituut vor Mediakunst / Montevideo Time Based Arts in 2003, during a period of artistical residence in its facilities, and it's being mantained and made available at the address korova.dyne.org/video_streaming³³.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with the Invariant Section being the Front-Cover Text. A copy of the license is made available on the [gnu.org](http://www.gnu.org/licenses/fdl.html) website at the address: <http://www.gnu.org/licenses/fdl.html>

Video streaming terms glossary

CODECS

MPEG

the Moving Picture Experts Group defines industry standards for the encoding, management, and delivery of content through various media.

MPEG-1

a codec designed for near-VHS quality video. MPEG-1 is primarily designed for CD-ROM delivery of content through various media.

MPEG-2

a codec designed for high-quality video. MPEG-2 is primarily used for DVD disc encoding and other high-quality archival solutions, but can be streamed over high-bandwidth connections, such as Internet2. MPEG-2 playback often requires additional software and/or hardware.

MP3

a codec designed for audio. MP3 is the most popular standard used for distribution on the Internet and in portable music players, such as Apple's iPod. Note that MP3 stands for MPEG Audio Layer 3, not MPEG-3 (there is no MPEG-3).

32. <http://rastasoft.org>

33. http://korova.dyne.org/video_streaming

MPEG-4

a "codec container" built around free and proprietary codecs, MPEG-4 excels at multi-architecture compatibility and efficient compression for low bandwidth streaming. Being a cross-platform container, its existence makes redundant other platform specific container protocols: as of QuickTime, Real and WindowsMedia.

The best audio/video codec tuples for MPEG4 are MP3/H.263 (widely supported on most players) and AAC/H.264 (highest efficiency).

QuickTime

built around several non-proprietary codecs, QuickTime excels at medium to high bandwidth clips. It supports MPEG-4.

Real

built around proprietary RealNetworks codecs, Real format excels at low to medium bandwidth clips. It supports MPEG-4. Producers can use the SMIL markup language to add interactivity.

WindowsMedia

built around proprietary Microsoft codecs, Windows Media excels at medium bandwidth clips. It does not correctly support the MPEG-4, but a modified version of it (MSMPEGv.3).