HES-SO // MASTER

L

Projet d'approfondissement,
orientation Technologies de l'information et de la communication (TIC)

Diffusion audiovisuelle sous MPEG DASH

rédigé par
DAVID FISCHER

Sous la direction de
Prof. Andrés Revuelta

de la MRU TIC de hepia

8 Jul 2012

Acknowledgments

I would like to gracefully thanks Mr. Bram Tullemans for his guidance and support during this work.

Thanks you, Mr. Tullemans, for the knowledge you shared with me and my colleague, Josué Beltran.

I will also thanks the EBU Technical staff for the motivating environment their provided and for the help the whole team members has provided for the work to be done.

Thanks to Josué Beltran, who is a invaluable colleague, very energetic and able to focus is energy to help me investigating the open source implementations of DASH not only for me, but also for his bachelor work !

Thanks to my PA follower, prof. Andrés Revuelta, for his good humour and for all the professional and personal experience we shared in common with our team on the telecommunications laboratory of hepia.

Thanks to my family, especially to my wife, Claire, for the personal support they provided to my during this work and also during the whole time I followed the Master courses.

... And sorry if I forgot anyone !

Abstract

MPEG DASH is an upcoming standard for streaming video over the internet. It unifies different propriety techniques for adaptive streaming via HTTP into one open standard. This preliminary work is concerned with the question how MPEG DASH can be implemented in a broadcasters work-flow using only open source software.

The deliverable will be a development plan to build this open source distribution chain with the intention to start a follow up programme to actually develop and demo this open source MPEG-DASH distribution chain from encoding to playout on different connected devices like pc, tablet, smart-phone and televisions. A evaluation of the demo will indicate what parts of the open source distribution chain need further development in order to be an alternative for propriety solutions.

Unification of different streaming techniques and the open source availability are very important for broadcasters when the technological convergence results in more efficiency and a better Quality of Experience for their audience. Efficiency will be realised when a single encoding format can be used for distributing video to different devices that are used.

At this moment of time broadcasters are forced to build different (propriety) platforms to deliver their video. Open source can, in theory, maximise the efficiency because it limits the possibility of vendor lock in, opens the possibility that the sharing of knowledge will lead to a better product and can help to reduce costs when the bills paid for support do not overcome the advantage that the software itself is free of charge.

Quality of video Experience consist for a great part on the performance of the network. When packages get lost the video starts halting. With adaptive streaming the available speed of the internet is verified by the player and an optimised bitrate is selected on the server. The end user will get the maximum available bits per second video in this end to end distribution chain and therefore experience the maximal possible quality at that moment of time.

Table of Contents

1 Executive summary

1.1 French

Here is the original executive summary.

Diffusion audiovisuelle sous MPEG DASH

Responsable	Revuelta Andrés
MRU	TIC / hepia
Orientation	TIC
Axes technologiques concernées	TIC / Systèmes d'information et multimédia
Entreprise	EBU / UER

Résumé

MPEG DASH ('Dynamic Adaptive Streaming over http') est le nouveau standard pour la diffusion de vidéos en ligne sur de multiples et nombreux périphériques.

Cette norme émergente contient une description de plusieurs profils, dont il semble que celui dédié à la diffusion directe possède un important potentiel technico-commercial.

Le présent projet vise la conception d'un démonstrateur 'open source' dans le but de vulgariser la technologie MPEG DASH.

Cahier des charges

- Réalisation d'un cahier des charges détaillé ;
- Étude de la norme DASH sous le profil de la diffusion directe ;
- Étude de faisabilité d'une plate-forme de démonstration en 'open source' utilisant FFMPEG et Apache.

Connaissances préalables

- Éléments de transmission et codage numérique multimédia ;
- Problématique de la transmission en temps réel sur IP.

Mots-clés

TIC IPTV; TIC Multimédia; TIC Streaming vidéo; TIC TV numérique

1.2 English

Multmedia streaming over MPEG DASH

Responsible	Revuelta Andrés
MRU	TIC / hepia
Orientation	TIC
Technological domain	TIC / Multimedia & informations technologies
Company	EBU / UER

Resume

MPEG DASH ('Dynamic Adaptive Streaming over HTTP') is the upcoming standard for online video deliverance to multiple devices.

In this upcoming standard different profiles are described and it seems that the live profile has the biggest potential to be picked up in the market.

The goal of the following project is the design of an 'open source' demonstrator in order to popularize MPEG DASH.

Specifications

- Project specifications refinement ;
- Study documentation about the live profile of MPEG DASH ;
- Description of work to be done to get an open source distribution chain.

Prerequisites

- Good knowledges of the multimedia transmission standards & codecs ;
- Understand the constraints of the real-time streaming over IP.

Keywords

TIC IPTV; TIC Multimedia; TIC Video Streaming; TIC Digital TV

2 Definitions

Broadband streaming

Multimedia content delivery through a broadband network, most (if not all) of the streaming technologies are based on the IP stack of the Internet protocols.

Broadcast streaming

Multimedia content delivery through a broadcast network, such networks are designed to provide an unidirectional way to transmit productions from an unique source to the mass. Classically, multimedia content is encapsulated in MPEG-2 TS packets delivered by Digital Video Broadcasting systems.

Adaptive streaming over HTTP

HTTP-based adaptive bit-rate streaming technologies. Such technologies are specifically designed in order to provide to client a way to handle network conditions variations[1] by continuously selecting an optimised bit-rate representation of the multimedia content. The delivery server must provides multimedia content encoded on multiple representations with specific bitrate and resolution.

Linear multimedia content

Multimedia content intended to be viewed in real-time, the audience will consume this type of content linearly from the beginning to the end. Typical applications : TV channels, books.

Non-linear multimedia content

Multimedia content that can be consumed on a non-linear way. The client can seek freely on such type of content. Typical applications : Video on demand, video games, ...

Representation

A representation is a specific content encoded on a specific parameter set (quality, geometry, bitrate, …).

1 Others metrics can be used at client side : Screen resolution, computational power ...

3 Abbreviations

3.1 Common

EBU	European Broadcasting Union
GIT	Git (version control system)
HbbTV	Hybrid Broadcast Broadband Television
HTML	Hypertext Mark-up Language
IEEE	Institute of Electronical and Electronics Engineers
IETF	Internet Engineering Task Force
IDE	Integrated Development Environment
ISP	Internet Service Provider
LAN	Local Area Network
MPEG	Motion Picture Experts Group
OIPF	Open IPTV Forum
QoE	Quality of Experience
QoS	Quality of Service
SDI	Serial Digital Interface
STB	Set-Top Box
SVN	Subversion (version control system)
TV	Television
VCEG	Video Coding Experts Group
VOD	Video On Demand
VLC	Video LAN Client
W3C	World Wide Web Consortium
XML	Extensible Markup Language

3.2 Video & Protocols

FLV	Adobe	Flash Video
H.264	VCEG/MPEG	Advanced Video Coding
HLS	Apple Inc.	HTTP Live Streaming
HTTP	IETF	Hypertext Transfer Protocol
IP	IETF	Internet Protocol
MMS	Microsoft	Microsoft Media Services
MP4	MPEG	MPEG-4 File Format
MSS	Microsoft	Microsoft Smooth Streaming
PNA/PNM	RealNetworks	Progressive Networks Streaming Protocols
RDT	RealNetworks	Real Data Transport
RTCP	IETF	Real-Time Control Protocol
RTMP	Adobe	Real-Time Messaging Protocol
RTP	IETF	Real-Time Transport Protocol
RTSP	IETF	Real-Time Streaming Protocol
TCP	IETF	Transmission Control Protocol
TS	MPEG	Transport Stream (e.g. MPEG-2 TS)
UDP	IETF	User Datagram Protocol

3.3 DASH

DASH	Dynamic Adaptive Streaming over HTTP
ISO BMFF	ISO Base Media File Format
MPD	Media Presentation Description
SAP	Streaming Access Point

4 Introduction

4.1 Context

Nowadays, video streaming over the Internet provide a serious opportunity for content producers to propose new services for their audience. The broadband networks are evolving rapidly and this evolution allow to distribute higher quality video content.

New services such as video on demand and 7 days catch-up are now a reality, thanks to the Internet. Now, it is possible to provide multimedia content to a wider range of devices, like smart-phones, tablets and computers.

However, the broadband networks are congested and their best-effort design does not permit to reach the quality of service guaranteed by the broadcast networks. Broadcast and broadband networks can be used together to deliver (BC) live video content to the mass and individualized services VoD (BB) to individual.

This is why Internet based streaming technologies are evolving rapidly in order to reach a quality of experience the users are used to. An upcoming technology, called adaptive streaming, seem to be a very promising way to dramatically reduce the common/main problems the clients are facing to when using IP based video services.

In this context, MPEG DASH is an upcoming standard unifying existing proprietary technologies for adaptive streaming over HTTP into an open source standard.

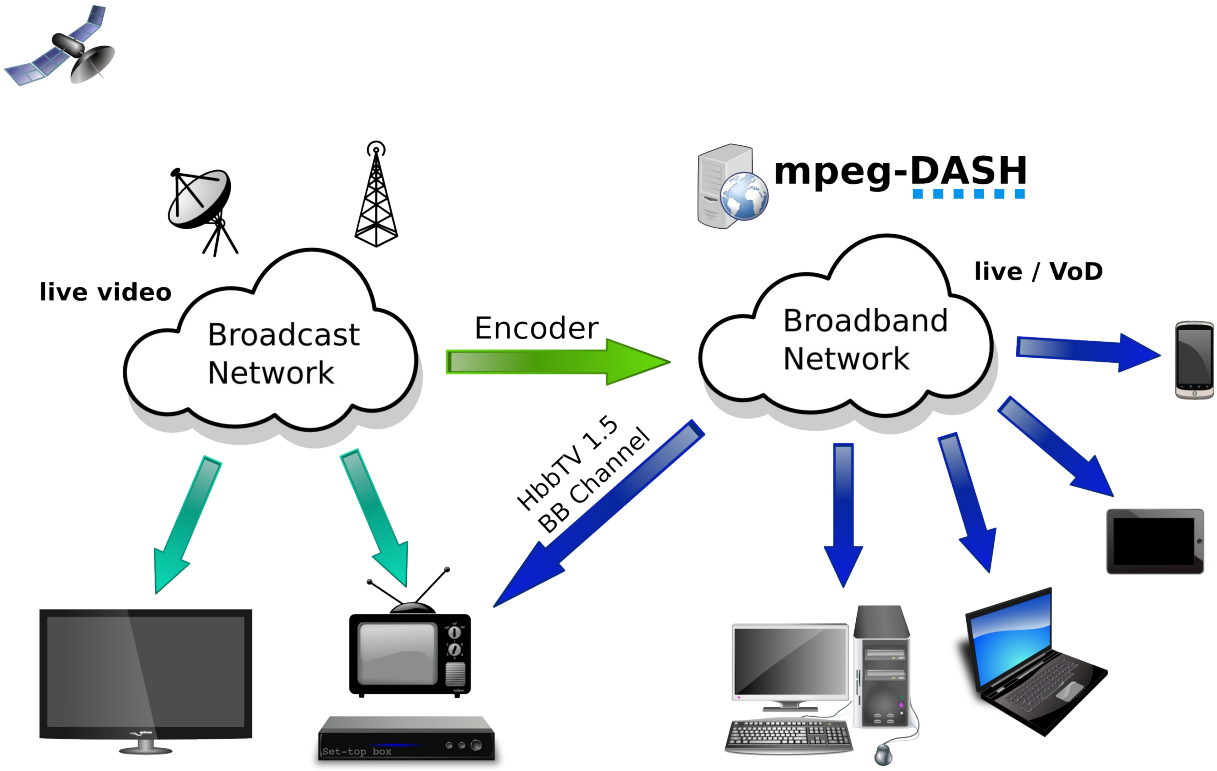


Figure 1: Multimedia content delivery through broadcast and broadband networks

4.2 Comparison BC & BB

	Broadcast	Broadband
Network	Dedicated	Shared best-effort (not only streaming)
Bandwidth	Guaranteed	Various Bitrate ← network load + ISP contract
Latency	Fixed	Various Latency → packets delivery jitter
Problems	Hardware failure	Packet dropping, out-of-order delivery
Protocols	Standardized	A lot of streaming protocols
Codecs	A few	A lot of different codecs
End devices	TV, Set-Top Box	TV, STB, PC, Tablet, Mobile phone
Screen size	Usually big	Various size
Characteristics	Dedicated HW	Various CPU capacity, memory, OS, plugins
Resulting quality	QoS guaranteed	QoE not guaranteed, from low to high

As you can see on this comparison a BB network is a challenging way to transmit content to the audience !

4.3 Problematic

By using broadband networks to provide multimedia content broadcasters are facing to the common problems generated by the best-effort design of such type of networks. These networks weren't designed to deliver media content on real-time :

- End-to-end Bandwidth can vary over the time, which means that the media content can't be delivered on time to client. The multimedia software will display a well known “buffering” error message saying that you must wait for the player to get more content prior to displaying it on the screen !
- Latency of IP packets delivery can vary over the time, this is called jitter. The multimedia software must handle this jitter by downloading a sufficient amount of content in his buffer prior to beginning the play-out, so one must be more or less patient. This problem can't be handled at 100%, if the latency/jitter change too much during the live show then it can be the case that the “buffering” error message will appear !
- Packets can be lost or dropped, out-of-order delivered → Buffering is needed, implying latency !
- The client will be behind a NAT/Firewall and this security equipment will probably be configured by the system administrator in order to filter Internet traffic to improve the security of the LAN he manage. As a result, the client may not be able to successfully receive the content one wants to watch ! The broadcasters must take into account that fact to avoid such kind of problems. The easiest way to bypass most of these standard security rules is to encapsulate traffic on HTTP requests.

By reaching to a wider range of devices, broadcasters should provide multiple representations of the same base content. The client should be able to select the best representation one can handle, based on the capabilities of the end-device hardware and the user preferences.

By reaching software based players, the broadcasters should take care of the wider range of plugins and codecs handled or not by the wide range of player implementations.

4.4 Broadcasters needs

Broadcasters needs confidence in the tools they use, in order to ensure or maximise the quality of the services they provide to the mass audience. To achieve this goal, new tools must be tested and integrated to the existing equipments prior to their use in production.

The audience usually watch/access to the multimedia content via these services :

- Live Television — The show is consumed in real-time in a non-interactive way
- Time-shift TV — The user can instantaneously seek back in actual TV program
- Catch-up TV — The user can replay any TV program of past week(s)
- Video On Demand — The user can earn a movie or TV program picked-up in a catalogue

Each of these 4 different scenarios/services has specific requirements as described here :

	Live television	**Time-shift TV**	**Catch-up TV**	**Video on demand**
End to end latency	Lowest	Lower	Low	Low
Media encoding	Online	Online	Online or Offline	Offline
Media publishing	Real-time	Real-time	~1 day	No requirements
Type of playback	Linear	Non linear	Non linear	Non linear

The following requirements must be fulfilled by the whole production chain.

It must be possible to deliver sign language and subtitles (if provided) for the deaf and hard-of-hearing audience.

Live sport events are typically covered by several cameras in multiple viewpoints and it would be a valuable additional service if a multi-view service can be provided in parallel to the main view.

Licensed external media productions needs to be protected against piracy and this is a real necessity to handle the protection of this content with DRM.

The MPEG consortium has taken all these requirements (and probably more) into account, it's why it can be an ideal choice for broadcasters to use MPEG DASH enabled tools !

In chapter 5.7 (page 35) I will explain how to integrate DASH in a production chain to provide these services.

4.5 Broadband Streaming Technologies

Streaming is a technique for transmitting content (audio, video, ...) as a steady and continuous stream over a network so that it can be processed in real-time by the streaming client. Streaming technologies and protocols are specifically designed to handle the problematic of data transferring through non ideal networks such as Internet.

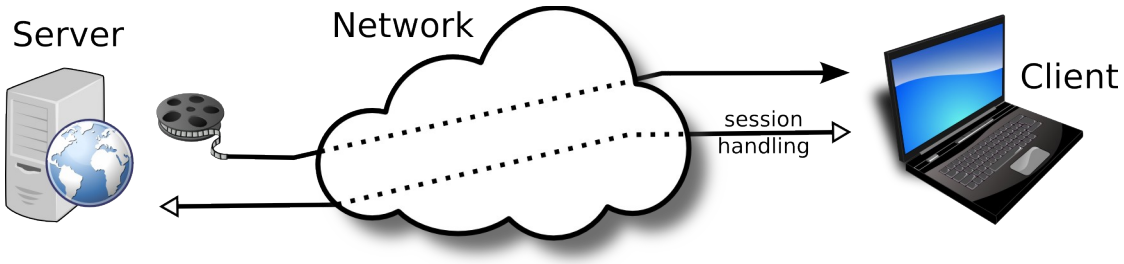


Figure 2: Context of the streaming. Here, stream and session channels are separated

The multimedia source can be transferred through the network in two different ways [1] :

- Treated as pure data by data transfer protocols. These type of protocols handles data correction (by retransmission) and they typically transfer data as fast as possible. For example : TCP/IP ;
- Treated as a continuous stream by streaming protocols. These type of protocols handles the real-time constraints and they adapt the speed of the data they transmit. The packet lost should be handled by the higher level protocol by adding a forward error correction in top of the streaming protocol. E.g. : UDP/IP ;

The multimedia session can also be managed in two different ways :

- Statefull : The server and client work together to maintain the session ;
- Stateless : The server will only respond to client's request, the client must handle the session ;

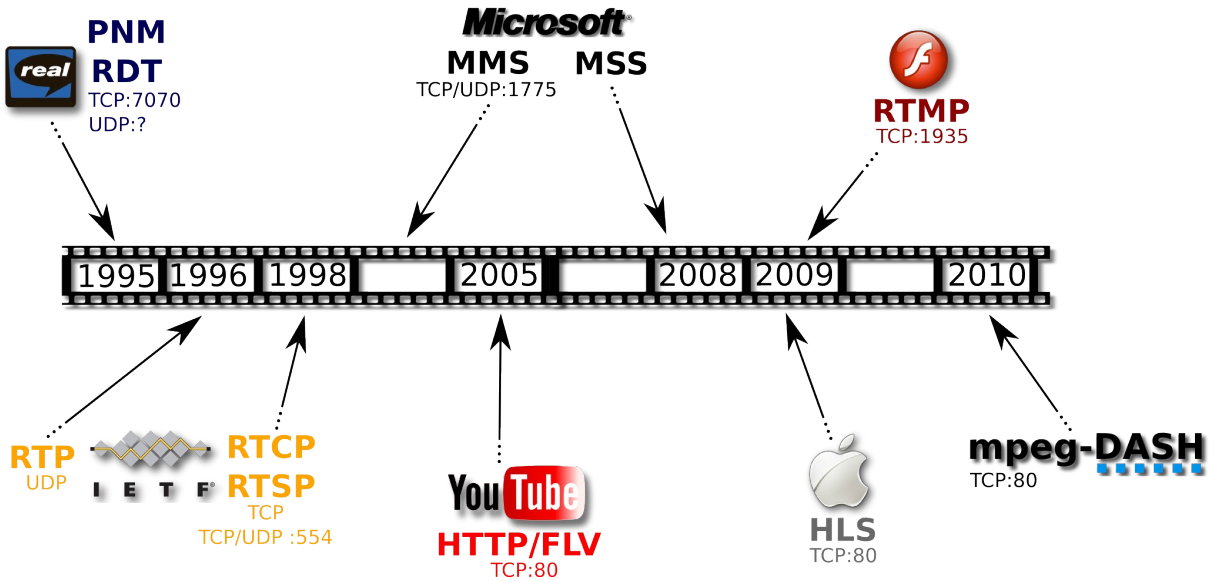


Figure 3: Streaming technologies appearance over the time

As a trend, we can remark that recent streaming technologies privilege the usage of the HTTP stateless protocol.

In the next chapters, I will introduce one of the most recent and promising streaming technology called MPEG-DASH. Then, I will explain why HTTP can be a good option despite this technology is not specifically designed to handle the constraints of real-time data transfer.

4.6 Dynamic Adaptive Streaming over HTTP

4.6.1 Introduction

Dynamic **A**daptive **S**treaming over **H**TTP (ISO/IEC 23009-1) is a multimedia streaming standard of the MPEG consortium started in 2010 and finally released as an International Standard in November 2011.

MPEG DASH is born of a common need of a standard for adaptive streaming over the Internet improving the quality of user experience (QoE). This standard will ensure the interoperability of the hardware and software devices dedicated to multimedia content encoding, delivery and consumption.

More than 20 companies joined a Promoters Group of DASH (DASH-PG) [2] including industry leaders such as Adobe, Microsoft, Qualcomm, Akami, Netflix, Cisco, NDS, Envivio, Samsung and Harmonic. This group called the DASH Promoters Group is dedicated to the promotion and implementation of the standard.

DASH in a nutshell

The multimedia content is partitioned into multiple (1 to N) media segments, each of them being represented on the web by an unique URL. An XML file is also provided in order to describe the media as a whole : The MPD.

The MPD describes the media components (main tracks, alternative tracks, representations, ...) and contains useful details about these components (URL, timing, media characteristics such as video resolution and bitrate).

The DASH media server is only responsible for providing files to client by answering their HTTP stateless GET requests. The client should manage the streaming session by downloading required segments at appropriate time.

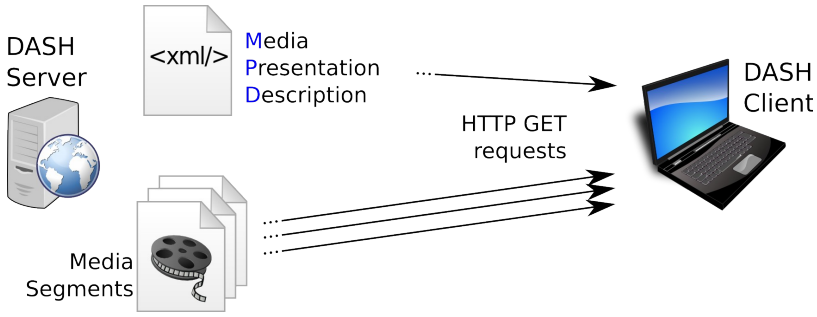


Figure 4: DASH main components (files, protocol)

The client will :

- Download the media's MPD ;
- Select the components/tracks to play ;
- Select appropriate representation of components/tracks at moment of time * ;
- Download “current” media segments of selected tracks representations ;

* The client will maximise is QoE by switching between the representation based on relevant factors such :

- Network conditions (bandwidth, ...) ;
- Device capabilities (screen resolution, CPU/GPU power, memory, ...) ;
- User preferences (language, default resolution, ...) ;
- Player screen space (windowed, full screen, ...) ;

What is specified and what is not [3]

This standard specify the format of the MPD and segments, the transfer protocol to use (HTTP/1.1) and subsets of the standard called profiles (see chapter 4.6.7, p. 23). The client engine is only described and not specified.

4.6.2 Advantages of this technology

The MPEG-DASH Promoters Group give an overview of DASH, here is an extract :

> *" HTTP streaming has become a popular approach for delivering of multimedia content over Internet. This approach has several benefits. First, the Internet infrastructure has evolved to efficiently support HTTP. For instance, CDNs provide localized edge caches, which reduce long-haul traffic. Also, HTTP is firewall friendly because almost all firewalls are configured to support its outgoing connections. HTTP server technology is a commodity and therefore supporting HTTP streaming for millions of users is cost effective. Second, with HTTP streaming the client manages the streaming without having to maintain a session state on the server. Therefore, provisioning a large number of streaming clients doesn't impose any additional cost on server resources beyond standard Web use of HTTP, and can be managed by a CDN using standard HTTP optimization techniques. " [4]*

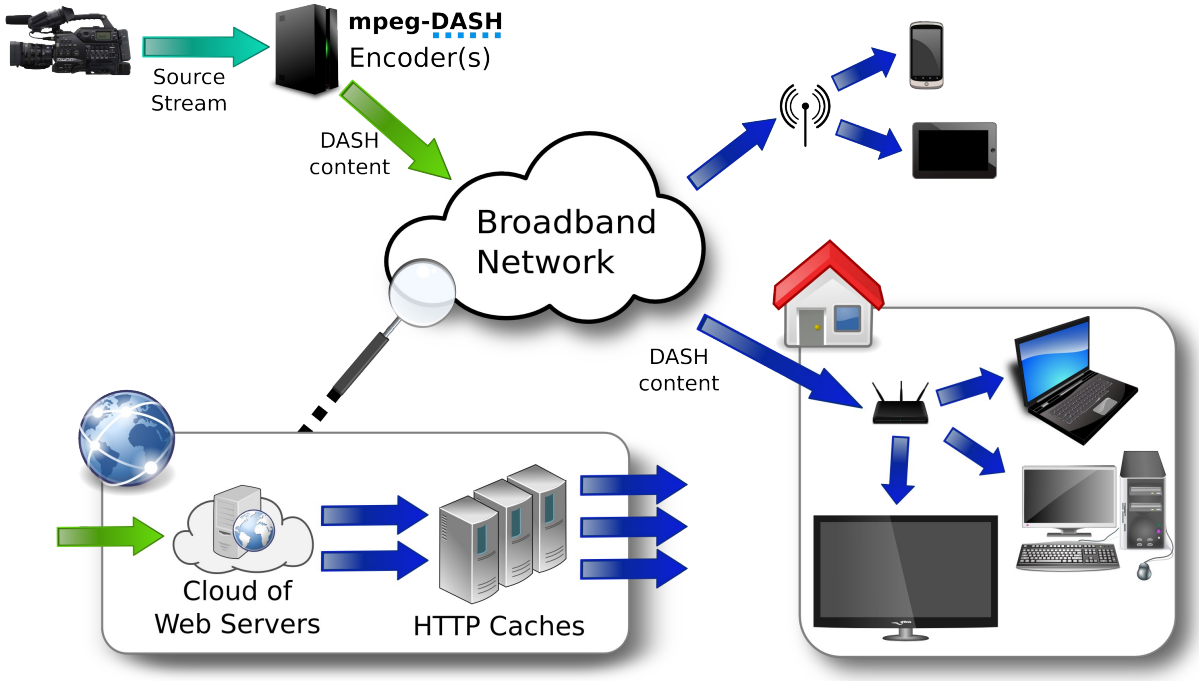


Figure 5: Zoom on the broadband side of DASH network topology

Advantages are numerous :

- Re-use widely deployed HTTP networks (caches, servers, …) ;
- Improve scalability by using stateless HTTP servers ;
- Firewalls pass-through (uses the well known port 80/443) ;
- Seamless adaptation to conditions (network, device, …) ;
- Avoid re-buffering and reduce start latency ;
- Is new services enabled (multi-view, multi-lingual, ...) ;
- Improve the user quality of experience ;

4.6.3 *An illustrative scenario*

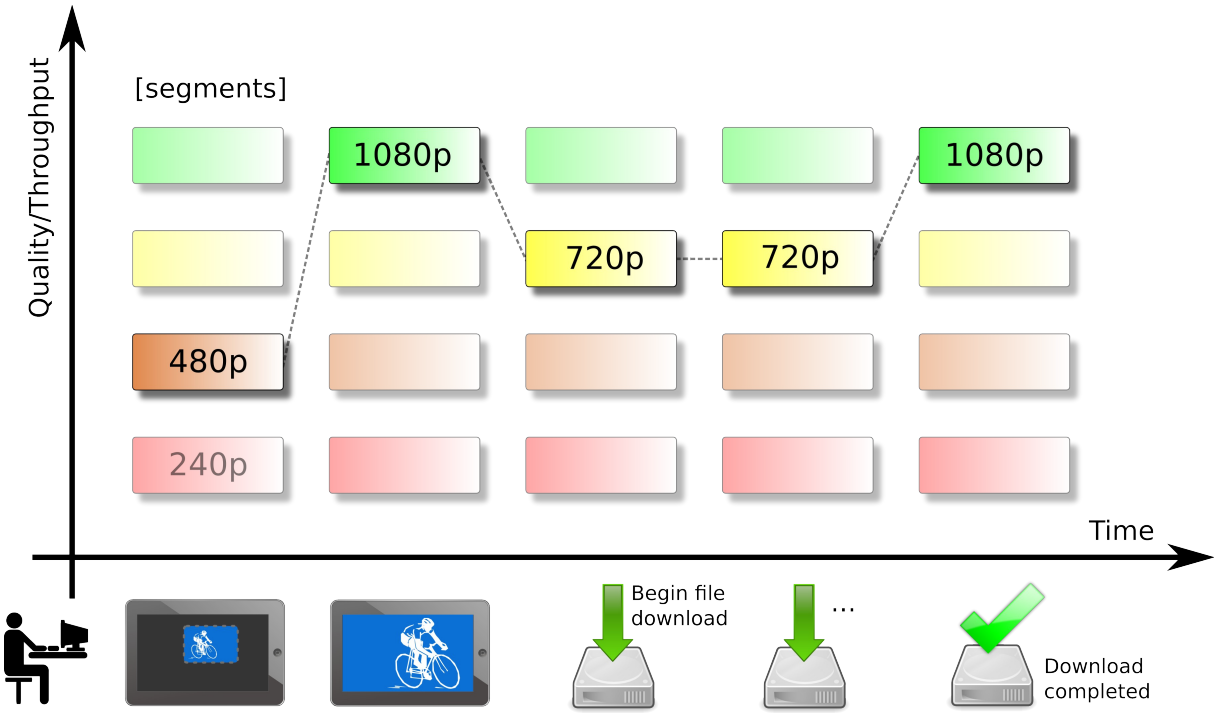
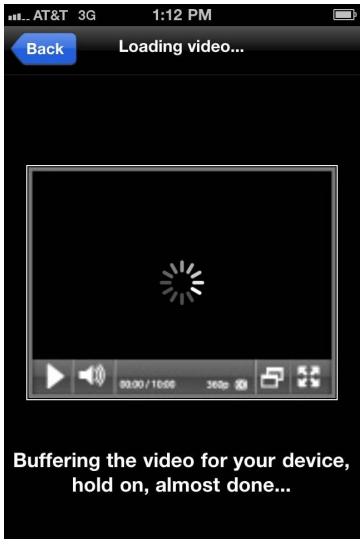


Figure 6: Client : Continuous adaptation of the throughput by switching between representations

Figure 6 is a scenario where an user, Daniel, click on is favourite Internet TV channel to watch to a bike-cycling event. A few minutes later, he discovered that is favourite team will probably win the race and he decide to switch to full-screen. His son, more interested in information technologies than bike-cycling, decide to download a GNU/Linux distribution from is computer. During this scenario, the adaptive multimedia streaming client was able to manage the conditions in order to maximise Daniel's QoE.

Sure, it looks cool, but what is avoided ?



4.6.4 State of the Art : Overview of Adaptive Streaming Technologies

	Adobe Dynamic HTTP Streaming	Apple HTTP Live Streaming	Microsoft Smooth Streaming	MPEG-DASH
Uses cases	Live, VoD	Live, VoD	Live, VoD	Live, VoD
Adaptive bitrate	Yes	Yes	Yes	Yes
Segmenter software	Free	Included in OS X	$199, includes encoder	Free[2]
Media container	MPEG 4 – Part 14, FLV	MPEG-2 TS	MPEG 4 – Part 14	3GPP, MPEG4, ISO BMFF
Video codec	H.264	H.264 Baseline Level	Agnostic	Agnostic
Default fragment length	4 seconds	10 seconds	2 seconds	Flexible
Media files storage	Contiguous	Segmented	Contiguous	Both
Maximum bitrate	Unlimited	1.6 Mbps	Unlimited	Unlimited
Caching support	Yes	Yes	Yes	Yes
End-to-end latency	6 seconds	30 seconds	>1.5 seconds	Flexible

Table 1: Comparison of adaptive streaming technologies

They are numerous already available applications of these technologies, here are few illustrative examples.

Microsoft Smooth Streaming in IIS Media Services and Silveright [5]

" Smooth Streaming, an IIS Media Services extension, enables adaptive streaming of media to Silverlight and other clients over HTTP. Smooth Streaming provides a high-quality viewing experience that scales massively on content distribution networks, making true HD 1080p media experiences a reality. "

Figure 7: IIS Smooth Streaming Media Workflow, Copyright : Microsoft Corp.

2 Depending on the availability of MPEG-DASH open-source implementations.

Akamai HD for iPhone & iPad [6]

" With Akamai HD for iPhone & iPad, you can send both live and on-demand audio and video content over HTTP for playback on Apple devices such as iPhone®, iPad®, or iPod® Touch. "

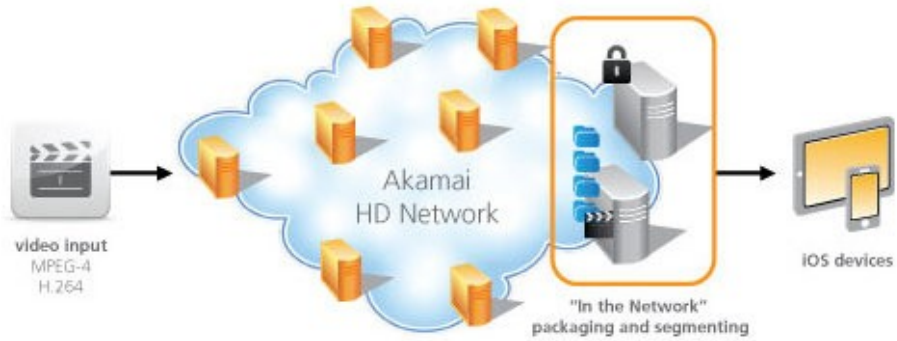


Figure 8: Akamai HD for iPhone & iPad, Copyright : Akamai Technologies

Akamai HD for the Adobe Flash Platform [7]

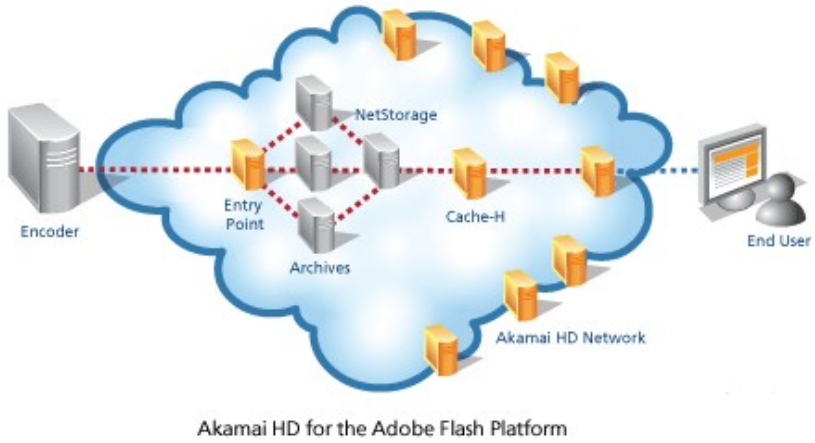
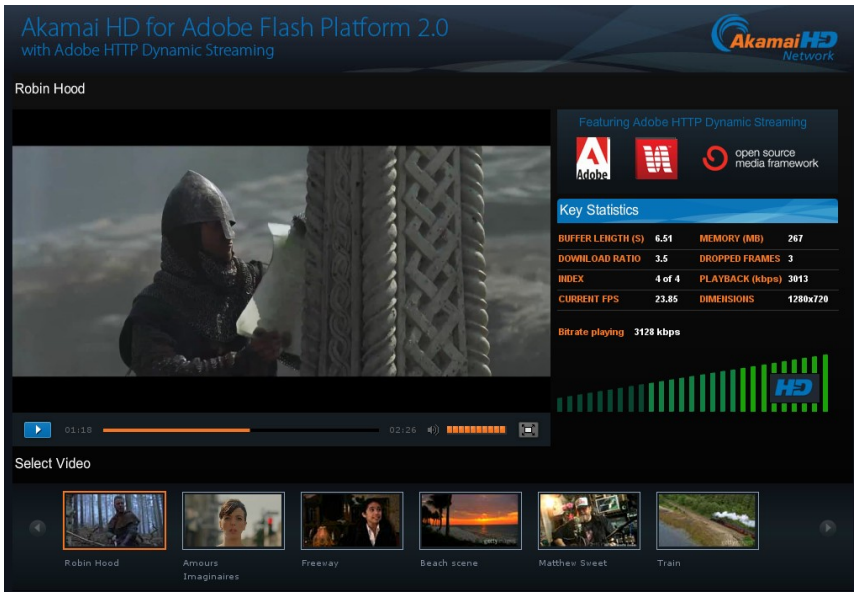


Figure 9: Akamai HD for Adobe Flash Platform, Copy. : Akamai T.

Here is a screen-shot of the Akamai demonstrator based on Adobe Flash HTTP Dynamic Streaming [8] :



Qtube – view and edit live production content, anywhere [9]

> *" QTube is game-changing software that enables content creators, administrators and managers to interact with their content wherever they are and wherever their content is located. QTube is already in daily use transforming content creation in the same way, globalizing workflows for media organizations of all sizes around the world. "*

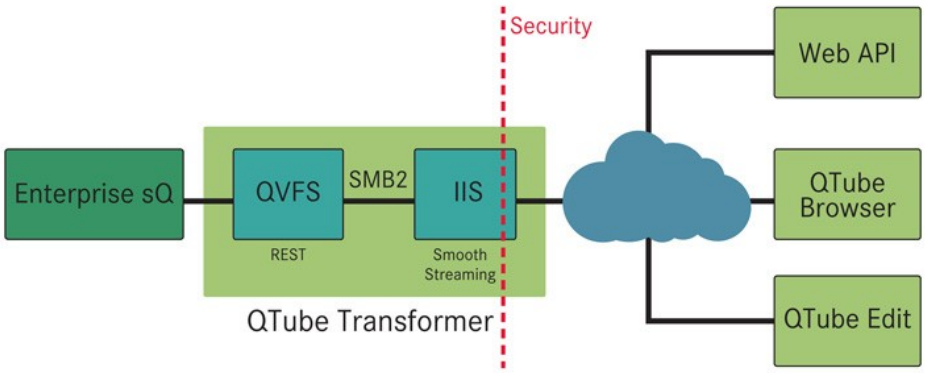


Figure 10: QTube workflow over IP, Copyright : Quantel

… And DASH !

As you noticed on the comparison table 1 (page 18), MPEG-DASH is by design the most flexible (adaptive) streaming technology available on the market. An implementation of this standard has the potential to replace concurrent technologies like Adobe, Apple and Microsoft adaptive streaming. This fact is not a coincidence but the result of the collaboration of the domain's major actors (industrial leaders or normalization groups) !

This emerging standard will be used by the industry's actors to release DASH enabled implementation of their proprietary adaptive streaming technology [10] to ensure a better interoperability of the multimedia devices and platforms. The open-source contributors are also part of it as they will implement DASH standard in their open-source software. This guarantee to BC a vendor free way to produce and consume DASH enabled content.

The main differentiating factor between DASH and the others is the use of a non-proprietary manifest file, the MPD describing the media characteristics (tracks, segments, ...).

This flexibility add complexity, this is why DASH describe subsets of the standard, called profiles. These profiles specify requirements and parameters adapted to common multimedia delivery uses cases and formats, for example Live, Video-On-Demand, … See chapter 4.6.7 page 23 for further details.

4.6.5 State of the Art : DASH Commercial Applications

Here are two commercial implementations of DASH.

MPEG DASH Developments in ST [11]

" Early prototype of the technology on ST/ST-Ericsonn embedded platforms
→ ST: 7108, Orly → ST-Ericsson: A9500-Snowball
DASH Client integration in multimedia SW middleware
→ Linux/GStreamer → Android/Stagefright "

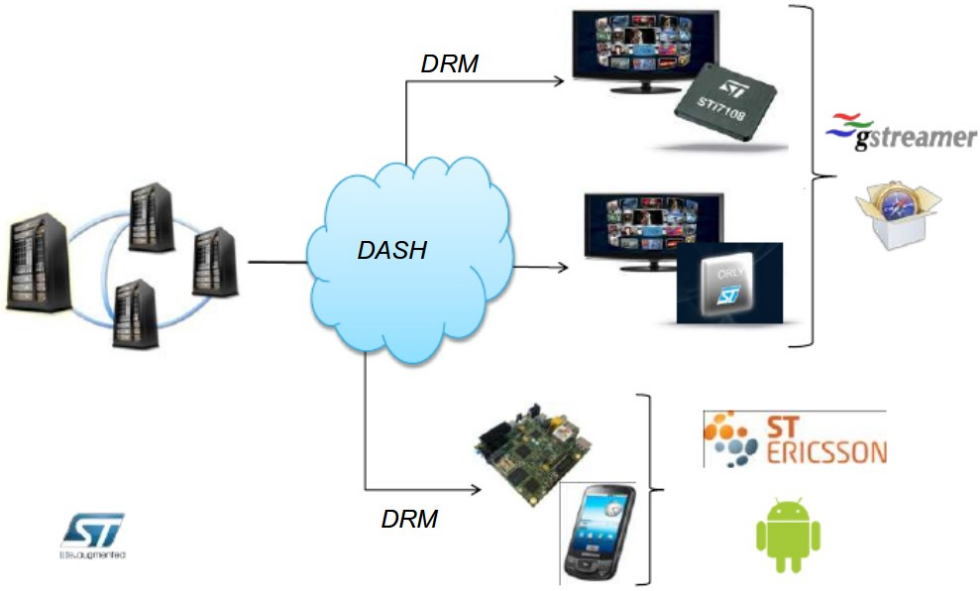


Figure 11: Target 2012. Multiplatform DASH Streaming, Copyright : STElectronics

RGB Networks is First with TV Everywhere Packager Support for MPEG DASH ... [12]

" Sunnyvale, California – March 29, 2012 – RGB Networks, the leading provider of scalable multiscreen IP video delivery solutions, today announced that its award-winning TransAct Packager now supports the MPEG Dynamic Adaptive Streaming over HTTP (MPEG DASH) protocol. Adding this to its support for Apple HTTP Live Streaming (HLS), Microsoft Smooth Streaming and Adobe HTTP Dynamic Streaming (HDS), RGB's TransAct Packager is the only packager to support all four protocols in live and on-demand environments. With this broad support, video service providers (VSP) can continue to deliver video to any IP-enabled device, even as the industry continues to evolve."

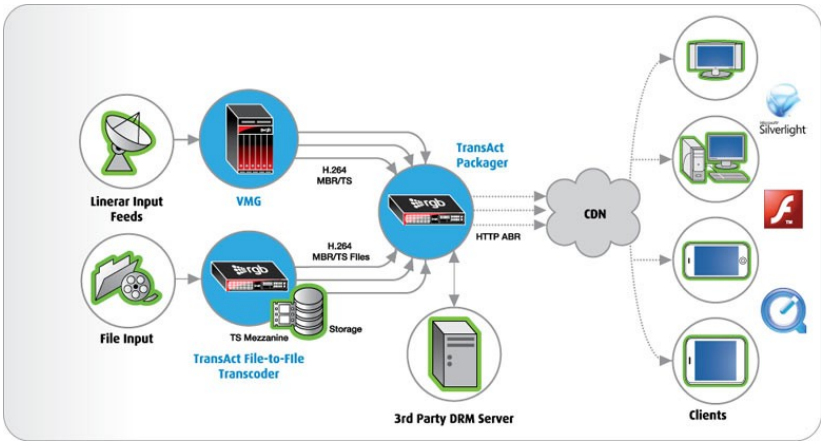


Figure 12: TransAct Packager, Copyright : RGB Networks

4.6.6 State of the Art : DASH Open-Source Implementations

First of all, I will thanks my colleague for his investigations, I can provide a good picture of the open-source implementations of DASH, here is the status at time of writing :

	ffmpeg	**gstreamer**	**GPAC / MP4Box**	**DASH Encoder**	**DASH-JS**	**GPAC / OSMO4**	**Stagefright**	**VLC**
Role	Encoding	Encoding	Encoding + MPD	Encoding + MPD	Client	Client	Client	Client
Platform	Linux, Mac, Win	Linux	Linux, Mac, Win, iOS, Android	Linux, Win, Mac	Chrome Browser	Linux, Mac, Win, iOS, Android	Android	Linux, Mac, Win
Licence	GPL / LGPL *	LGPL	LGPL	LGPL	LGPL	LGPL	?	GPL / LGPL
Repository	GIT	GIT	GIT	SVN	SVN	GIT		GIT
Version								
Activity								
MXF								
ISO BMFF Live								
ISO BMFF On demand								
ISO BMFF Main								
MPEG2-TS Main								
MPEG2-TS Simple								
Full								

* depending on the configuration (options) during the compilation

Color code :

- Red means that the profile is not supported by the tool ;
- Yellow means that the profile seem to be implemented ;
- Green means that the profile is implemented and tested ;

4.6.7 *Overview of DASH*

The Media Presentation Description

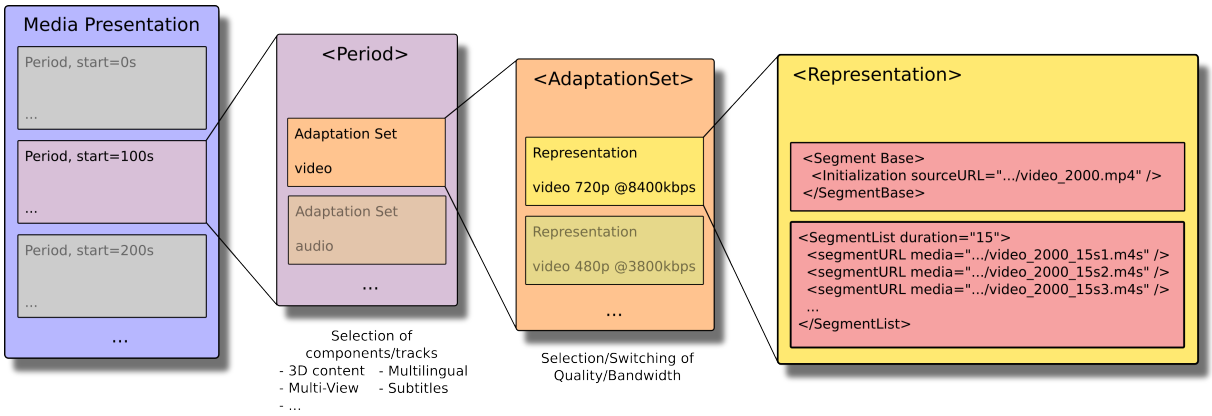


Figure 13: Hierarchical view of the Media Presentation Description XML file

Facts About DASH

- Medias (representations, ...) shares a common time-line to permit synchronization ;
- Period → Hierarchical structure (adaptation, representation, ...) does not change during a period ;
- AdaptationSet → type of stream (e.g. video, audio, subtitles, ...) ;
- Representation → specific codec settings (resolution, bandwidth) ;
- Segment → a segment [t,t+dt] of the stream = URL ;
- SAP → Codec independent concept of switching point (not described in this document) ;
- Client can switch representation between two segments by seeking or switching ;
- Client select the best representation according to metrics (see chapter 4.6.2 p. 16) ;
- Remark : It is better to avoid download of overlapping data by switching !

DASH Profiles

DASH Profiles are described in the standard at page 90, here is a short overview of them :

MPD Acronym	Profile Name	Related Applications and Technologies
isoff-live	ISO BMFF Live	HbbTV 1.5, Microsoft Smooth Streaming
isoff-on-demand	ISO BMFF On Demand	Supporters : Netflix, Qualcomm & few others
isoff-main	ISO BMFF Main	
mp2t-main	MPEG2-TS Main	Apple HTTP Live Streaming
mp2t-simple	MPEG2-TS Simple	
full	Full profile	

Table 2: Profiles defined in the standard of DASH

ISO Base Media File Format Live Profile

- Used by HbbTV 1.5 (a reduced set of live profile)
- MPD @type = dynamic
- MPD can be dynamically updated with new entries/segments -OR- templatized URLs
- Segments available when produced
- Short segments to minimize end-to-end latency
- Usage of segment availability time (wall-clock time)

ISO Base Media File Format On Demand Profile

- MPD @type = static
- All segments are available
- 1 unique segment per representation
- Subsegments aligned across representations within an Adaptation Set
- Subsegments must begin with a SAP
- Scalable & efficient use of HTTP servers & simple seamless switching

Draft of DASH264 Profile (not in the standard)

- Efficient low latency

5 Project Specifications

5.1 Motivation

MPEG DASH standard is in the process of being implemented by open source developers to their favourite open source software. The European Broadcasting Union (EBU/UER) is interested in testifying that a complete video delivery chain (from producer to end-users) can be build by integrating such pieces of software. This proof of concept would be freely available for broadcasters.

5.2 Work to be done

The investigation will consist of defining which tools should be used (FFmpeg and Apache are just examples) and what developments are needed to get a full open-source MPEG DASH distribution chain. Livelihood of the development community is an important requirement of the open source tools.

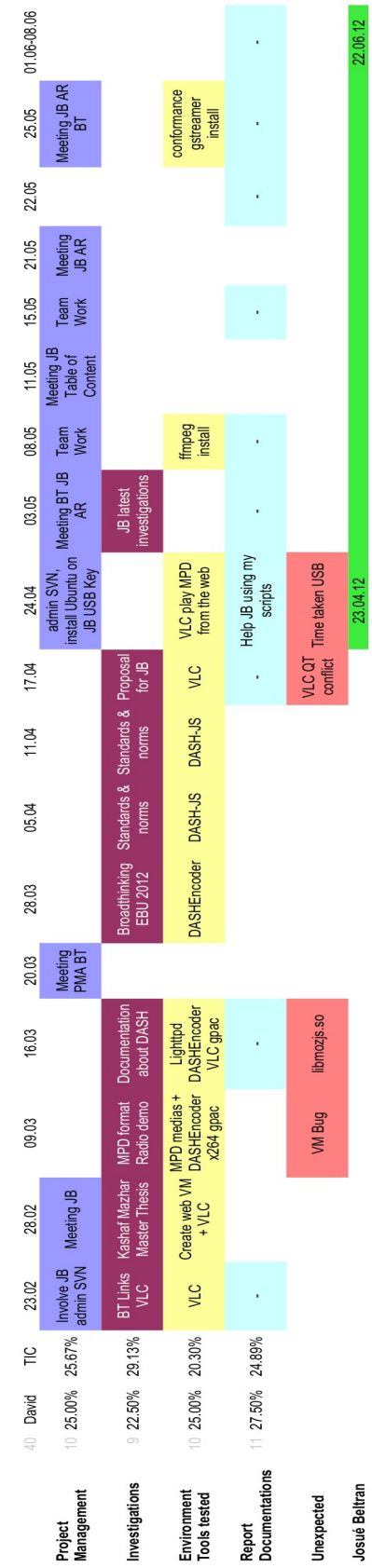
The clients support in this project in order of importance :

1. PC
    a. Browser (HTML5+JS)
    b. Player (e.g. VLC)
2. Television
    a. HbbTV 1.5
    b. MHEG5
    c. Youview
    d. MHP
3. Android smart-phones
4. iOS smart-phones

The specific scope of the work to be done is :

1. Gather specific requirements for On-Demand an Live DASH video distribution over IP ;
2. Description of chain plus tools ;
3. Evaluation of the constraints of the chain and tools in relation to the requirements ;
4. Description of work to be done to get an open source distribution chain ;

5.3 Planning



I began my project by a meeting with Josué Beltran, a bachelor student finishing his semester project on MPEG DASH at hepia [13].

He gave me a good overview of what was in development by the open-source community, he discovered that one of the most active open-source contributor is the Information Technology Institute of the Klagenfurt University in Germany (ITEC). [14]

When I started, ITEC's team was developing a library (libdash) [15], as well as an encoder based on gpac and ffmpeg (DASHEncoder) [16]. They was also implementing DASH in VLC [17]. A few weeks later, they provided a DASH MPD Validator web service [18] and DASH-JS [19], a JavaScript implementation targeting WebM compatible browsers like Chrome and Chromium.

Mr. Beltran faced some trouble when managing to get the source code of the tools and specially when he tried to compile it.

Based on these facts, I decided to create a development environment based on a lot of bash scripting. As a result, I produced scripts in order to easier Install, compile, test and deinstall all the open-source tools.

I also did investigations on the topic, read a rich master thesis (Compliance Procedures for DASH by Kashaf Mazhar) [20] the draft of the standard ...

Mr. Tullemans invited me to a really interesting seminar hosted by the EBU/UER called EBU BroadThinking 2012 [21]. This two days seminar helped me to understand the background of the broadcasting, the way the broadcasters develop and integrate new technologies to their production environment and few other interesting topics !

The 23th of April, Mr. Beltran start his bachelor thesis. Convinced that it would be far more interesting to work together as a team, I took the decision to involve him in my environment by granting him an access to my (now I must say our) project's subversion repository.

It was necessary to supervise and give him advices and directions. Once a week, we discussed about the planning, the remaining tasks and the unexpected problems he faced during the past week.

As a result, we were able to share our experience and knowledge.

Mr. Beltran focused on the deep investigations (open-source tools, do tests, reading source code, which profiles are implemented and so on ...).

For my part, I improved the scripts I developed, wrote some documentation and continued my investigations in order to write this preliminary report.

Remark : In 2008 the MSE HES-SO has released a study about the skills needs in the information technology industry. This is the TIC column in my planning and my work matched this profile.

5.4 MPEG DASH Demonstrator : A First Picture

The primary development platform I used during this project was my personal laptop, an Asus ZenBook UX31 with Ubuntu 12.04 LTS x86_64 installed on it.

Has a backup and synchronization system I installed and configured an SVN server on a Synology NAS.

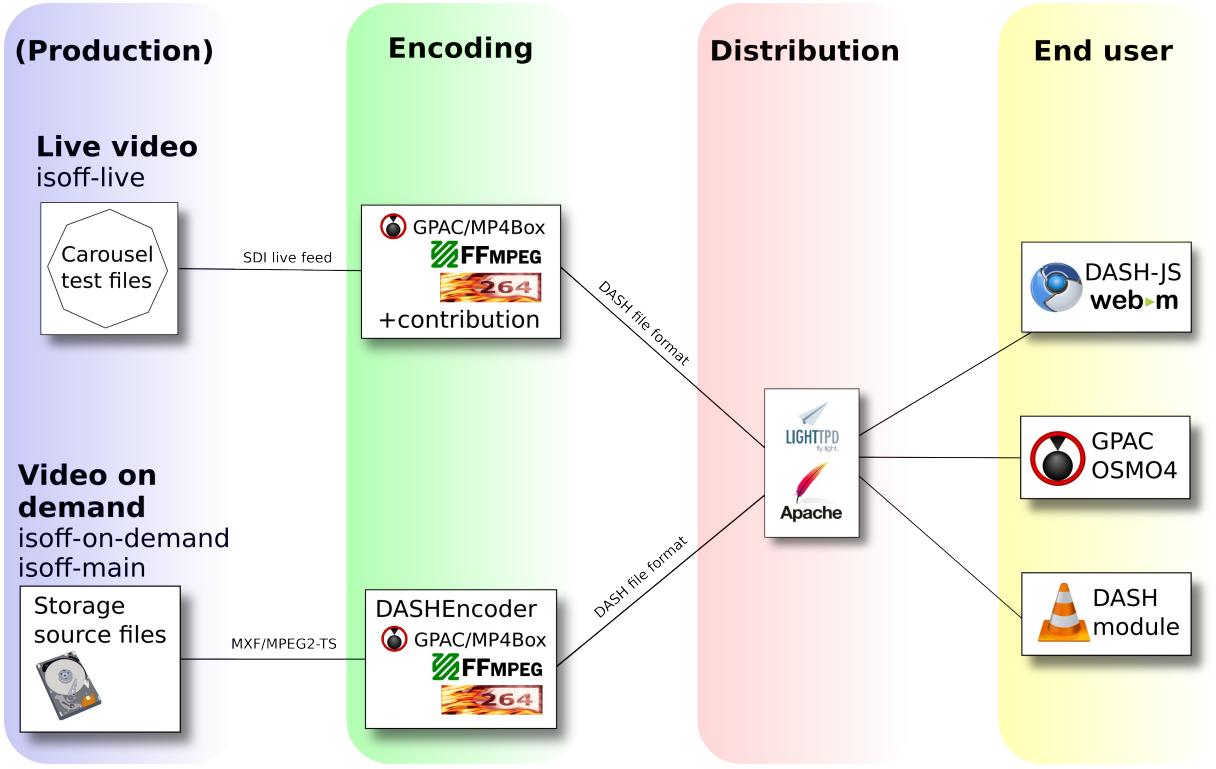


Figure 14: Most of the opens-source DASH chain that is investigated

As you can see on the figure, a DASH distribution chain can be decomposed in 4+1[3] main components.

The source media file/stream produced by the contributors is grabbed from the production's output and encoded in media segment(s). The DASH MPD file is generated and the required DASH meta-data's are appended to media segments. Finally, all these DASH compliant files are stored in a web-server farm/cloud to make them available on the Internet. The last but not least component (not shown in this figure) is the brain of the system !

DASH Video-On-Demand demonstrator scenario :

1. The tester will go to the web-page of the demonstrator and click on the Video-On-Demand demo ;
2. The tester will upload a media file to the FTP repository of the demonstrator ;
3. The demonstrator will detect new files and encode it on the fly ;
4. The tester will receive a mail with a link to the generated/resulting content page ;
5. The tester can play the content on his browser (if compatible) or uses his DASH player (if installed) ;

DASH Live demonstrator scenario :

1. The tester will go to the web-page of the demonstrator and click on the Live demo ;
2. The tester can play the content on his browser (if compatible) or uses his DASH player (if installed) ;

In the next sub-chapters I will briefly introduce and explain the role of the most promising open-source components of the media production chain, not forgetting the development tools not directly involved in this chain.

3 Production, encoding, distribution, end user and the management system !

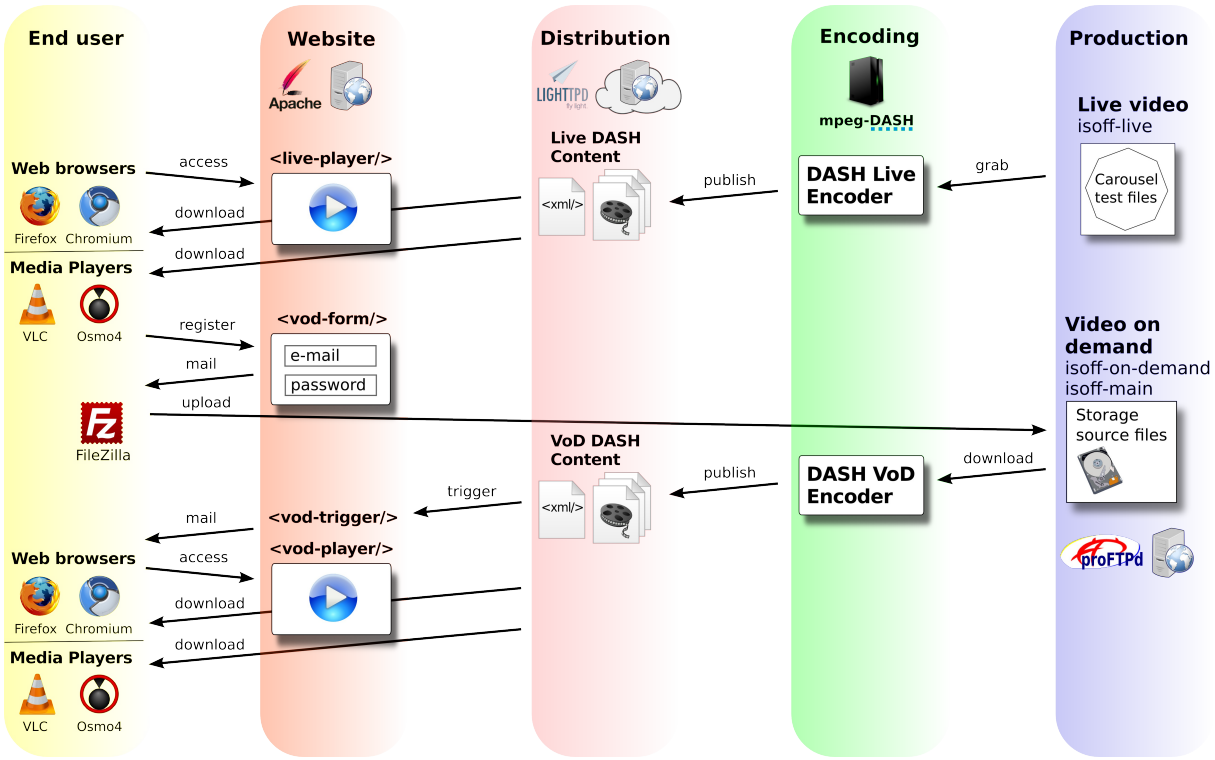


Figure 15: DASH Live and Video-On-Demand scenarios (encoding & management soft. are not shown)

5.4.1 Operating system : Ubuntu 12.04 LTS

> *" Ubuntu is a computer operating system based on the Debian Linux distribution and distributed as free and open source software, using its own desktop environment. ... Ubuntu is sponsored by the UK-based company Canonical Ltd., owned by South African entrepreneur Mark Shuttleworth. Canonical generates revenue by selling technical support and services related to Ubuntu, while the operating system itself is entirely free of charge. The Ubuntu project is committed to the principles of free software development; people are encouraged to use free software, improve it, and pass it on. " (source wikipedia)*

5.4.2 Development tools : SVN and GIT

What is SVN & GIT ?

Subversion is a versioning and revision control software based on a client/server architecture. Developers use this kind of tool to maintain current and historical versions (called revisions) of their work such as source code and documentation of their project(s).

Subversion is one of the most widely used versioning system on the open-source community.

The SVN server hosts the projects repositories and the developers will use a SVN client to get a local copy of the project. The developer will work locally on his copy (add, remove, rename, modify files and directories) and then propagate (commit) these modifications to the repository.

GIT is a versioning and revision control software based on a distributed architecture. Git was initially designed and developed by Linus Torvalds for the development of the Linux kernel, the biggest open-source project ever made. Unlike Subversion, every Git local copy is a full-fledged repository with complete history. This design permit to maintain large distributed projects in a efficient manner.

Why using SVN & GIT ?

DASH is a cutting-edge standard and the open-source community is actively implementing DASH in their favourite open-source piece of software. This is why it is necessary to access to the latest revision of the source code of these softwares to understand what is implemented and what is not (profiles, ...) !

It is also necessary to manage backups of this project and Subversion is the best tool for that. Every backup (commit) is done manually and every single revision has a purpose, such a new functionality (code), a bug fix (code) or some files to backup (documents). The revision history permit revert changes if necessary and it also permit to get a statistical overview of the work (with statsvn for example).

This tool is also a must have to synchronize the contributions (work units) of the team members !

5.4.3 Media Encoder : FFmpeg

" FFmpeg is a free software project that produces libraries and programs for handling multimedia data. The most notable parts of FFmpeg are libavcodec, an audio/video codec library used by several other projects, libavformat, an audio/video container mux and demux library, and the ffmpeg command line program for transcoding multimedia files. FFmpeg is published under the GNU Lesser General Public License 2.1+ or GNU General Public License 2+ (depending on which options are enabled). " [source wikipedia]

5.4.4 Media Packager : GPAC/MP4Box

" GPAC stands for GPAC Project on Advanced Content (a recursive acronym). It is an implementation of the MPEG-4 Systems standard written in ANSI C. GPAC provides tools for media playback, vector graphics and 3D rendering, MPEG-4 authoring and distribution. … GPAC provides … A multimedia packager : MP4Box " [source wikipedia]

5.4.5 DASH Encoder : DASHEncoder

" For our dataset we developed a new content generation tool – on top of GPAC's MP4Box – for DASH video on demand content called DASHEncoder.

DASHEncoder generates the desired representations (quality/bitrate levels), fragmented MP4 files, and MPD file based on a given config file or by command line parameters respectively. Given the set of parameters the user has a wide range of possibilities for the content generation, including the variation of the segment size, bitrate, resolution, encoding settings, URL , etc., which is shown by the example of a DASHEncoder config file in the git repository of DASHEncoder.

The DASHEncoder is available as open source with the aim that other developers will join this project. The content generated by DASHEncoder is compatible with our DASH VLC plugin which can be used as a decoder and player respectively. " (source Itec, University of Klagenfurt)

5.4.6 Web-Server : Apache and Lighttpd

" The Apache HTTP Server, ... is web server software notable for playing a key role in the initial growth of the World Wide Web. In 2009 it became the first web server software to surpass the 100 million website milestone. Apache was the first viable alternative to the Netscape Communications Corporation web server (currently named Oracle iPlanet Web Server), and since has evolved to dominate other web servers in terms of functionality and performance. Typically Apache is run on a Unix-like operating system. " [source wikipedia]

" lighttpd (pronounced "lighty") is an open-source web server more optimized for speed-critical environments than common products while remaining standards-compliant, secure and flexible. It was originally written by Jan Kneschke as a proof-of-concept of the c10k problem - how to handle 10,000 connections in parallel on one server, but has gained worldwide popularity." [source wikipedia]

5.4.7 DASH Web-Browser Client : DASH-JS

“ DASH-JS is a seamless integration of the Dynamic Adaptive Streaming over HTTP (DASH) standard from MPEG into the Web using the HTML5 video element. Moreover, it is based on JavaScript which uses the Media Source API of Google’s Chrome browser to present a flexible and potentially browser independent DASH player. DASH-JS is currently using WebM-based media segments defined in. “ (source Itec, University of Klagenfurt)

5.4.8 DASH Smartphone/Tablets Client : GPAC/OSMO4

“ GPAC stands for GPAC Project on Advanced Content (a recursive acronym). It is an implementation of the MPEG-4 Systems standard written in ANSI C. GPAC provides tools for media playback, vector graphics and 3D rendering, MPEG-4 authoring and distribution. … GPAC provides … A multimedia player, cross platform … with a GUI : Osmo4 “ [source wikipedia]

5.4.9 DASH PC/Software Client : VLC

“ VLC media player (also known as VLC) is a highly portable free and open-source media player and streaming media server written by the VideoLAN project. It is a cross-platform media player, with versions for Microsoft Windows, Mac OS X, Linux, BeOS, MorphOS, BSD, Solaris, iOS, and eComStation.

VLC media player supports many audio and video compression methods and file formats, including DVD-video, video CD and streaming protocols. It is able to stream over computer network and to transcode multimedia files.

VLC used to stand for VideoLAN Client, but since VLC is no longer simply a client, that initialism no longer applies.

The default distribution of VLC includes a large number of free decoding and encoding libraries, avoiding the need for finding/calibrating proprietary plugins. Many of VLC's codecs are provided by the libavcodec library from the FFmpeg project, but it uses mainly its own muxer and demuxers and its own protocols. It also gained distinction as the first player to support playback of encrypted DVDs on Linux and Mac OS X by using the libdvdcss DVD decryption library. “ [source wikipedia]

5.4.10 Cloud IaaS Infrastructure : OpenStack

“ OpenStack is an Infrastructure as a Service (IaaS) cloud computing project by Rackspace Cloud and NASA. Currently more than 150 companies have joined the project among which are AMD, Intel, Canonical, SUSE Linux, Red Hat, Cisco, Dell, HP, IBM and Yahoo!. It is free open source software released under the terms of the Apache License.

OpenStack integrates code from NASA's Nebula platform as well as Rackspace's Cloud Files platform.

OpenStack has a modular architecture that encompasses three components :

- Compute (Nova) is a cloud computing fabric controller (the main part of an IaaS system). It is written in Python and utilizes many external libraries such as Eventlet (for concurrent programming), Kombu (for AMQP communication), and SQLAlchemy (for database access).

- Object Storage (Swift) is a massively scalable redundant storage system.

- Image Service (Glance) provides discovery, registration, and delivery services for virtual disk images.

Several components have been added for the next release :

- Open Stack Identity Management (Keystone)

- User Interface Dashboard (Horizon)

OpenStack has APIs compatible with Amazon EC2 and Amazon S3 and thus client applications written for Amazon Web Services can be used with OpenStack with minimal porting effort. “ [source wikipedia]

5.5 DASH Demonstrator : The Constraints

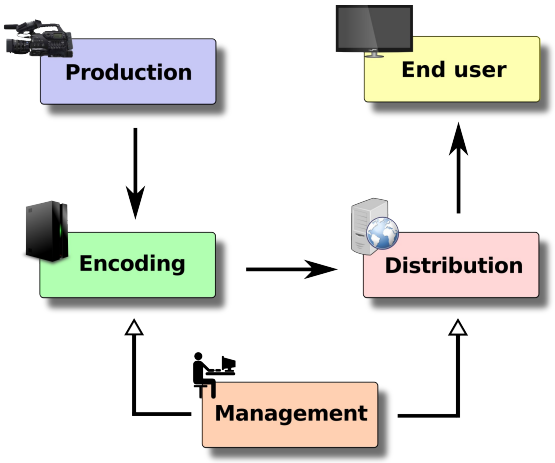


Illustration 16: Components of the chain

The chosen **encoding** implementation should handle the following constraints :

- **Files naming** : The segments must be named according to the source informations (like the pip of the program) and the specific segment's name template (e.g. numbering) ;
- **Time-stamping** : The media streams must share a common time-line, it does mean that the implementation should handle the synchronization problematic (e.g. A/V lip sync) ;
- **Errors handling** : The system must be reactive when errors occurred. If any of the processes fail or crash, it is absolutely necessary to replace the faulty process and to re-encode the missing segment by using a cached version of the source stream or by transcoding a higher quality already encoded segment of the same part of the stream ;
- **Latency & Scalability** : The architecture of the system must designed in order to reduce the encoding latency (e.g. for live content delivery) and to be as scalable as possible (e.g. VoD encoding farm) ;

The chosen **distribution** implementation should handle the following constraints :

- **Easy set-up** : The web-server should be as simple as possible to install and configure ;
- **Low memory footprint** : The web-servers will only deliver static content, allowing low memory footprint ;
- **High Availability & Scalability** : It will be easy (at least possible) to configure the web-servers to integrate them to a highly available delivery server farm / cloudy architecture.

The chosen **end-user** implementation should handle the following constraints :

- **Multi-Platform** : Implementing DASH to a widely used and multi-platform player will ensure that the work done will be useful for a larger audience.
- **Session Handling** : This is a de facto constraint : The client must handle the streaming session. The session handling algorithm should correctly parse the media's MPD and mix-up metrics (e.g. network BW, resolution, ...) to maximise the QoE of the playout by selection the right segments. If the stream is seekable, the client should provide a way for the client to do this operation.
- **QoE Feedback** : If the client software will give quality of experience feedback to the servers, it would be possible to adapt the service based on the audience (e.g. which bit rates to provide ?). It will be also really interesting to send such feedback as it provide a way for the service providers to give statistics to users (e.g. real-time state of the network).

5.6 DASH Demonstrator : Work To Be Done

When I wrote this report, a complete DASH distribution chain cannot be implemented with the current version of the open-source tools. It seem, based on Mr. Beltran's and my investigations that the live profile is currently not implemented at all or it will require so much tricks that the resulting demonstrator cannot be used on a “real” production environment. In contrast, it is already possible to find partial or complete implementations of the others DASH profiles (for example Video-on-Demand).

In the next sub-chapters I will specify work to be done to complete the open-source DASH distribution chain for the Live profile.

5.6.1 Media Production Component

No specific work, the SDI live feed will be already available (fingers crossed).

5.6.2 DASH Encoding Component

Hardware

I developed my project on Ubuntu, so I must choose a GNU/Linux compatible SDI card.

As a first step, the demonstrator can be as cheap/simple as possible (single computer). Then, if the resulting work seem promising, it will be interesting to think bigger !

Software

The media encoding must be done in parallel with the DASH formatting and MPD generation processes.

I see at least three different ways to implement this component :

- **Divide to conquer** : The encoding and packaging parts are two separated softwares (e.g. FFmpeg and MP4Box) driven by a third element (my contribution). This specialized tool could be firstly developed with shell scripts (fast prototyping) and then implemented on a compiled language (to improve performances, error handling, …) ;
- **May the best man win** : The encoder could be integrated to the packager (by using encoder's libraries if available). A separated management tool would spawn an instance of the “encoder+packager” for each source media. This tool can be (if required) directly implemented on the “encoder+packager” or keep separate ;
- **A new start** : A completely new and specialized (only for DASH Live profile ?) tool could be developed from the scratch by using encoder's and packager's libraries (e.g. libavcodec + libgpac) ;

Divide to conquer

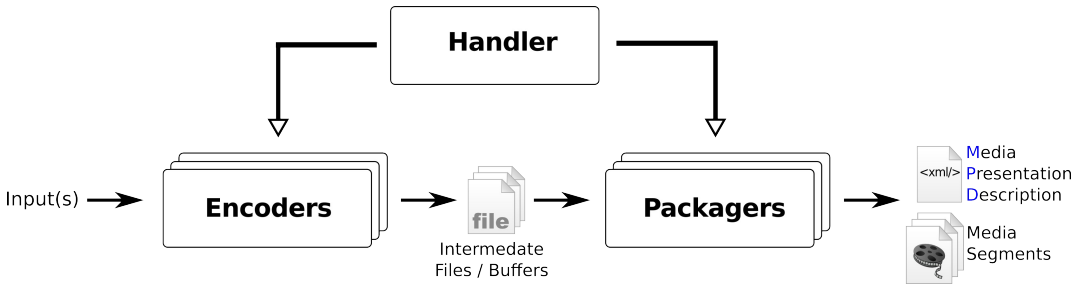


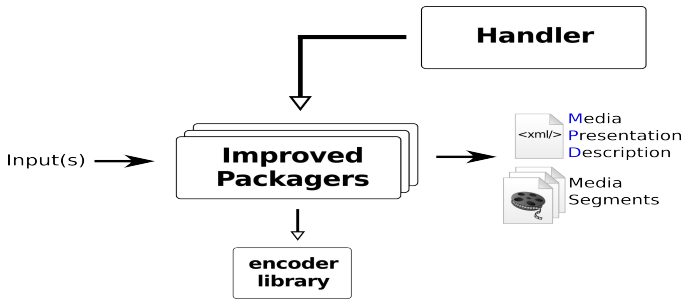
Figure 17: Divide to conquer implementation of a DASH Encoder

An unmodified version of the encoder (e.g. FFmpeg) would be used to grab the input SDI stream and produce the short length media segments. In parallel, a modified version of the packager (e.g. MP4Box) will be used to add DASH meta-data's to segments and to produce the corresponding MPD.

It will also be necessary to spawn multiple encoder instances to generate the multiple presentation of the same content. If the end-media is composed of multiple streams, it would be also necessary to encode all these streams separately in order to minimize the latency of the encoding. The packager should be aware of that (by implementing specific command line arguments for example) otherwise it will not be able to produce a valid MPD file related to the end-media's composition. Finally, the packager must be modifier in order to respect the producer/consumer paradigm (encoder → *intermediate files* → packager).

May the best man win

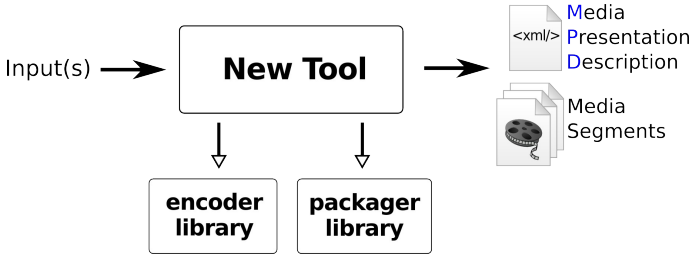
Encoder capabilities would be integrated to the packager (e.g. MP4Box + libavcodec). This alternative may require more developments/work but it would be easier to handle some of the most problematic constraints (e.g. encoding+formatting latency, synchronization, ...) and to do optimizations (e.g. fewer intermediate files).



A new start

A completely new tool (maybe inspired by DASHEncoder) would be developed, this tool will uses encoder's and packager's libraries (e.g. libavcodec + libgpac) in order to produce DASH compliant content.

This highly specialized tool must be completely designed from the scratch, thus allowing more freedom for the developer.



It may require more work to maintain this tool has it uses two, maybe more different libraries (developed by two different open-source community of contributors).

5.6.3 DASH Distribution Component

I recommend the usage of the lightweight and optimized open-source lighttpd web-server for media delivery.

The main web-page of the demonstrator will probably be hosted by a more powerful web-server like Apache.

5.6.4 DASH Management Component

A main difference between a demonstrator and a final product is it ability to handles errors and the possibility to manage it. The management component can be the icing of the cake for this demonstrator !

It would be possible to provide a highly available, fault tolerant DASH production chain by handling the following :

- SDI signal error (green signal or on/off switching) ;
- Encoder/packager components crash ;
- Hardware failure, seamless replacement ;

The management system should provide a way to :

- Define the flow chart of streams (inputs → encoders [parameters] → … → storage → web ...) ;
- Start/stop the components of the chain (encoder, packager, publisher, ...) ;
- Log the activities of the production chain (system access, media treatments, crashes, ...) ;
- Give an overview of the chain's status (health, performances, …) ;
- Get feedback from the end-users (QoE, audience, ...)[4] ;

Finally, it may be interesting to increase the scalability and the performance of the production chain by driving a cloud based service (e.g. OpenStack) to launch redundant/multiple instances of the components !

4 This option requires that the end-user's DASH client implement **feedback** functionality

5.7 DASH : Media Distribution Uses Case

Here is a example of how DASH profiles can be used in common BC scenarios :

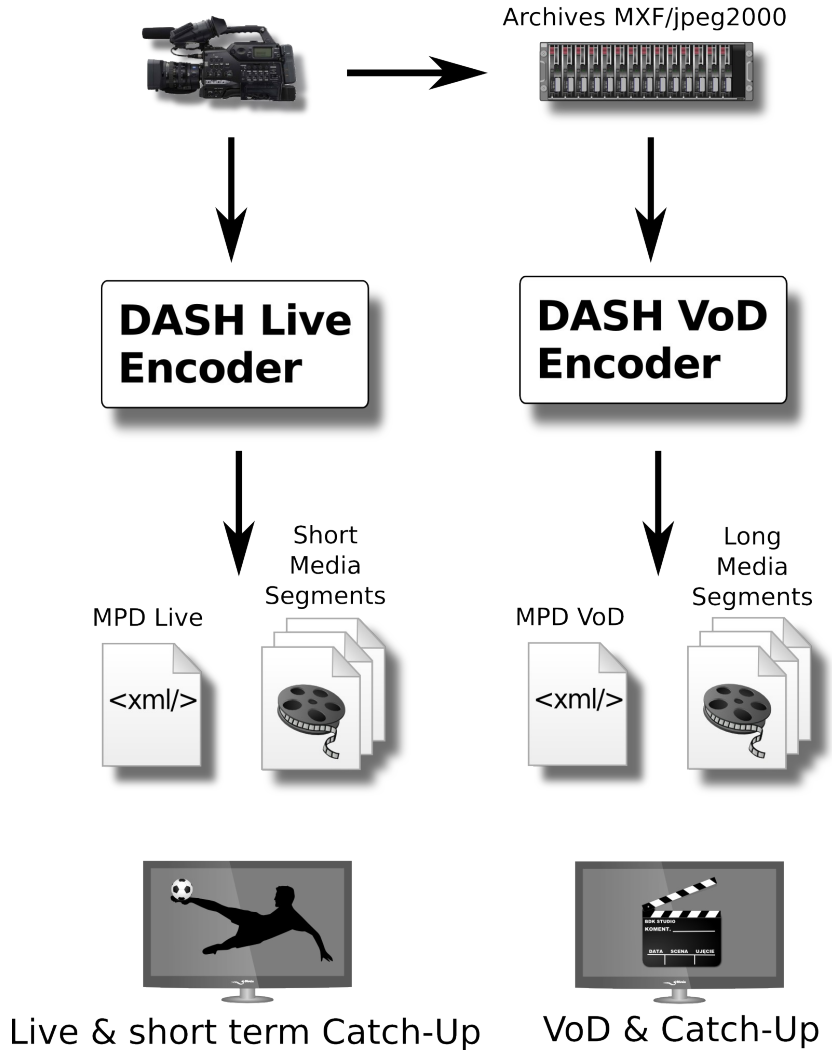


Figure 18: Broadcasting uses cases

Left side : Live Television and Short term Catch-Up TV

The source stream is encoded in real-time by a fast one-pass DASH Live Encoder. This content is then packeted in short length segments and a MPD is created or updated with required informations as specified in the Live Profile of the DASH Standard. The MPD and the segments are published as soon as possible.

Right side : Video-On-Demand and Catch-Up TV

The source stream is meaningfully segmented (e.g. 1 TV show = 1 segment) and stored in a archiving system.

Then, these files are encoded by a high-quality multi-pass DASH VoD Encoder and packeted in full-length segments. The MPD is completely filled with required informations as specified in the VoD Profile of the DASH standard. The MPD and the segments are published when this process is finished.

6 Conclusion

Broadcasters needs confidence in the tools they uses, they need to ensure or maximise the quality of the services they provide to the mass audience. They also want to provide new services to their audience by using new tools.

To achieve this goal, new tools must be tested and integrated to the existing equipments prior to their usage in production. MPEG DASH standard and open source implementations are these kind of exiting new tools to use !

I began my project by a meeting with the people involved in this project and by discovering what is MPEG DASH.

I created a development environment based on a lot of bash scripting. As a result, I produced scripts in order to easier Install, compile, test and deinstall all the open-source implementations of MPEG DASH.

Helped with those tools I was able to test the implementations and in parallel I also do investigations on this topic.

I attended to the two days EBU/UER BroadThinking 2012 seminar in order to understand the background of the broadcasting, the way the broadcasters develop and integrate new technologies to their production environment and few other interesting topics !

A few weeks later, Mr. Beltran started his bachelor thesis. Convinced that it will be far more interesting to work together as a team, I took the decision to involve him in my environment by granting him an access to my project's subversion repository.

It was necessary to supervise and give advices and directions to him. Once a week, we discussed about the planning, the remaining tasks and the unexpected problems he faced during the past week. As a result, we were able to share our experience and knowledge.

My colleague focused on the deep investigations and for my part, I refined the scripts I developed and wrote this preliminary report.

By investigating on this hot topic, I will help the EBU/UER and the broadcasters to integrate this new technology on a complete multimedia content distribution chain based on such open source tools.

Has a future work, I would like to develop such a proof of concept in the form of an open source demonstrator.

7 Annexes

7.1 References

The following documents are valuable resources.

Specifications

DASH	ISO/IEC 23009-1:2012(E)
RFC 2616	HTTP server compliance / DASH client compliance §9.3 => HTTP GET / partial GET
RFC 2818	HTTP over TLS / transport security
RFC 3986	xlink:href URI definition

Websites

Open Clipart Library

http://openclipart.org/

Streaming Media Protocols List

http://all-streaming-media.com/faq/streaming-media/faq-streaming-media-protocols.htm

HTMLMediaElement

http://html5-mediasource-api.googlecode.com/svn/trunk/draft-spec/mediasource-draft-spec.html

HTTP 1.1 - Persistent connections pipelining and chunking

http://www.subbu.org/blog/2004/11/persistent-connections-pipelining-and-chunking

DASH promoters group

http://dashpg.org/

DASH - Conformance Software

https://docs.google.com/document/d/1Vboii4BufzWojoFjEGhYlHsFJwmOY_VhsyCXK029ZJI/edit?pli=1

DASH - STMicro and Fraunhofer show off DASH prototype

http://www.itproportal.com/2011/10/06/stmicro-and-fraunhofer-show-dash-prototype/

Streaming Media - A very good article about MPEG-DASH

http://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=79041

Slideshare - DASH with MPEG2-TS

http://www.slideshare.net/agiladi/alex-giladi-dashtsr3

ITEC - Dynamic Adaptive Streaming over HTTP

http://www-itec.uni-klu.ac.at/dash/

ITEC - DASH dataset of several dash video streams and MPD versions

ftp://ftp-itec.uni-klu.ac.at/pub/datasets/mmsys12/

… And so many others.

7.2 Bibliography

1: Prof. Andrés Revuelta, Multimedia transmissions, Streaming chapter, 2005

2: , DASH Promoters Group, , dashpg.com

3: Thomas Stockhammer (Qualcomm), MPEG DASH Webinar, 2012, http://tech.ebu.ch/events/webinar043_mpeg-dash/cache/off?id=16979

4: mpeg-DASH promoters group, Overview of MPEG-DASH Standard, 17.11.2011, http://dashpg.com/?page_id=25

5: Microsoft Crop., Microsoft Smooth Streaming, 2012, http://www.iis.net/download/smoothstreaming

6: Akami Technologies, Akami HD for iPhone & iPad, 2012, http://www.akamai.com/html/technology/products/hd_delivery_iphone.html

7: Akamai Technologies, Akamai HD for the Adobe Flash Platform, 2012, http://www.akamai.com/html/technology/products/hd_delivery_flash.html

8: Akamai Technologies, Akamai HD for Adobe Flash Platform 2.0, 2012, http://wwwns.akamai.com/hdnetwork/demo/flash/zeri/index.html

9: Quantel, QTube - view and edit live production content, anywhere, 2012, http://www.quantel.com/page.php?u=fc919cb52f36247c19a4fdade742b8ce

10: Kevin Towes, Adobe announces support for MPEG-DASH streaming standard, 27.02.2012, http://blogs.adobe.com/ktowes/2012/02/adobe-announces-support-for-mpeg-dash-streaming-standard.html

11: Emanuele Quacchio, MPEG DASH Developments in ST, 16.02.2012, http://www.slideshare.net/imtcorg/mpeg-dash-st2

12: RGB Networks, RGB Networks is First with TV Everywhere Packager Support for MPEG DASH ..., 29.03.2012, http://www.rgbnetworks.com/newsandevents/pr/pr_032912-mdash.php

13: , Ingénierie des Technologies de l'information à hepia, , http://hepia.hesge.ch/index.php?id=63

14: , Institute of Information Technology at Klagenfurt University, , http://www.uni-klu.ac.at/tewi/inf/itec/

15: Christopher Mueller, libdash - Simplifies the usage of DASH, 07.02.2012, http://www-itec.uni-klu.ac.at/dash/?p=422

16: Stefan Lederer, DASHEncoder, 23.11.2011, http://www-itec.uni-klu.ac.at/dash/?p=291

17: Christopher Mueller, VLC 2.0 major release with basic DASH support, 20.02.2012, http://www-itec.uni-klu.ac.at/dash/?p=497

18: Markus Walti, MPEG-DASH MPD Validator released, 13.03.2012, http://www-itec.uni-klu.ac.at/dash/?p=711

19: Stefan Lederer, DASH-JS A JavaScript-based DASH library for Google Chrome, 27.03.2012, http://www-itec.uni-klu.ac.at/dash/?p=792

20: , Compliance Procedures for Dynamic Adaptive Streaming over HTTP (DASH), 2011

21: EBU/UER, EBU BroadThinking 2012, 2012, http://tech.ebu.ch/events/broadthinking2012

7.3 Project FAQ

How to get a copy of this project ?

My project is hosted in a subversion repository. If you want to get the latest update of my project, you can send a mail to david.fischer@hesge.ch to ask for a user credential to our repository.

```
$ svn co https://claire-et-david.dyndns.org/prog/DASH
```

I invite you to read the *README* and *INSTALL* files located in the base path of the project !

How to install the development environment provided by this project ?

With this project, some tools are provided as bash scripts. The first thing I invite you to do is to add these utilities to your *PATH* by opening a Terminal (the black thing of the evil) and executing :

```
$ sh dashExports
```

After that, you need to re-open another Terminal anywhere you want to be :

```
$ dashUpdate
```

```
$ dashInstall
```

dashInstall is a script able to help anyone who want to install all the tools necessary for this project to work.

How to update my copy of the project ?

In order to get the latest revision of the project & tools, just open a Terminal :

```
$ dashUpdate
```

Most of my scripts are gentle, so they ask for your acceptance prior to doing things. Sorry for the inconvenience it can generate if you need to execute them a lot of time.

How to compile ?

```
$ dashCompile
```

How to run tests and developments demo's ?

```
$ dashRun
```

```
$ dashDemo
```

The final demonstrator itself is not implemented at time of writing, so these scripts are here for us and you to be able to discover and test functionalities of the open source implementations of DASH we will use to develop it.

How to clean-up things ?

If you just want to clean the compilation results (prior the recompile the tools), you can execute that :

```
$ dashClean
```

If your need to completely deinstall the tools of your system just run :

```
$ dashUninstall
```

7.4 Source code

7.4.1 Licensing

The source code we provide is licensed under the GNU Genera Public Licence v3.

```
#*****************************************************************************#
#         OPEN-SOURCE DYNAMIC ADAPTATIVE STREAMING OVER HTTP DEMONSTRATOR
#
#  Author    : David Fischer
#  Contact   : david.fischer.ch@gmail.com / david.fischer@hesge.ch
#  Project   : DASH (The open source demonstrator)
#  Copyright : 2012 DASH Team. All rights reserved.
#*****************************************************************************#
#
# This file is part of DASH (The open source demonstrator).
#
# This project is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This project is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this project. If not, see <http://www.gnu.org/licenses/>
#
# Retrieved from:
#   svn co https://claire-et-david.dyndns.org/prog/DASH
```

7.4.2 dashExports (revision 205)

```
if ! grep dashExports $HOME/.bashrc >/dev/null; then
  echo "export DASH_PATH='`pwd`'" >> $HOME/.bashrc
  echo ". ''`pwd`/dashExports'" >> $HOME/.bashrc
  echo "DashExports successfully appended to $HOME/.bashrc"
fi

if ! grep "alias ls='ls --color=tty'" $HOME/.bashrc >/dev/null; then
  echo "alias ls='ls --color=tty'" >> $HOME/.bashrc
  echo "Colored LS successfully appended to $HOME/.bashrc"
fi

export PATH=$PATH:$DASH_PATH/scripts
```

7.4.3 dashCommon.lu-dep (revision 205)

```
if [ ! "$DASH_PATH" ]; then
  echo 'ERROR: DASH_PATH must be "exported" !'
  exit 1
fi

# Constants ===================================================================
```

```
if uname -a | grep -q x86_64
then LINUX_VERSION='64'
else LINUX_VERSION='32'
fi

DASH_BUILD_PATH=$DASH_PATH'/build'
DASH_SCRIPTS_PATH=$DASH_PATH'/scripts'
DASH_FILES_PATH=$DASH_PATH'/scripts_files'
DASH_PACKAGES_PATH=$DASH_FILES_PATH'/packages'
DASH_MEDIAS_PATH=$DASH_PATH'/medias'
DASH_OUTPUTS_PATH=$DASH_MEDIAS_PATH'/Out'
DASH_MPD_PATH=$DASH_MEDIAS_PATH'/MPD'
DASH_CONFORMANCE_PATH=$DASH_BUILD_PATH'/DASH_Conformance'
DASH_ENCODER_PATH=$DASH_BUILD_PATH'/DASHEncoder'
DASH_FFMPEG_PATH=$DASH_BUILD_PATH'/ffmpeg'
DASH_GPAC_PATH=$DASH_BUILD_PATH'/gpac'
DASH_GSTREAMER_PATH=$DASH_BUILD_PATH'/gstreamer'
DASH_HTML5_API_PATH=$DASH_BUILD_PATH'/html5-mediasource-api'
DASH_JAVASCRIPT_PATH=$DASH_BUILD_PATH'/DASH-JS'
DASH_LIB_PATH=$DASH_BUILD_PATH'/libdash'
DASH_STERICSSON_PATH=$DASH_BUILD_PATH'/STMicro'
DASH_VLC_PATH=$DASH_BUILD_PATH'/vlc'
DASH_X264_PATH=$DASH_BUILD_PATH'/x264'
DASH_STATS_PATH=$DASH_PATH'/statistics'
DASH_VMS_PATH=$DASH_PATH'/vms'

# find any (first) segment sourceURL on any of the 'local' MPD
c=`find "$DASH_MPD_PATH" -name '*_local.mpd' -exec cat "{}" \; | grep sourceURL`
DASH_BW_SEG_FILENAME=`expr match "$c" '.*sourceURL="\(.*\)"/>'`
DASH_WEB_IP=`cat "$DASH_VMS_PATH/VmDASH-Web.ip"`
# START OF LOGICIELS UBUNTU UTILS (licencing : LogicielsUbuntu project's licence)
# Retrieved from:
#   svn co https://claire-et-david.dyndns.org/prog/LogicielsUbuntu/public

# Colored echoes and yes/no question ============================================

TXT_BLD=$(tput bold)
TXT_BLK=$(tput setaf 0)
TXT_RED=$(tput setaf 1)
TXT_GREEN=$(tput setaf 2)
TXT_YLW=$(tput setaf 3)
TXT_BLUE=$(tput setaf 4)
TXT_PURPLE=$(tput setaf 5)
TXT_CYAN=$(tput setaf 6)
TXT_WHITE=$(tput setaf 7)
TXT_RESET=$(tput sgr0)

if echo "\n" | grep -q 'n'
then e_='-e'
else e_=''
fi

#if [ -z $DISPLAY ]
#then DIALOG=dialog
#else DIALOG=Xdialog
```

```
#fi
DIALOG=dialog

techo() { echo $e_ "$TXT_GREEN $TXT_BLD$1$TXT_RESET"; } # script title
pecho() { echo $e_ "$TXT_BLUE $1$TXT_RESET";           } # text title
mecho() { echo $e_ "$TXT_YLW $1$TXT_RESET";            } # message (text)
cecho() { echo $e_ "$TXT_YLW > $1$TXT_RESET";          } # message (code)
recho() { echo $e_ "$TXT_PURPLE $1 !$TXT_RESET";       } # message (remark)
qecho() { echo $e_ "$TXT_CYAN $1 ?$TXT_RESET";         } # message (question)
becho() { echo $e_ "$TXT_RESET$1";                     } # message (reset)

xecho() # message (error)
{
  echo $e_ "${TXT_RED} [ERROR] $1$TXT_RESET" >&2
  pause
  exit 1
}

pause() # menu pause
{
  echo $e_ 'press any key to continue ...'
  read ok </dev/tty
}

readLine() # menu read
{
  qecho "$1"
  read CHOICE </dev/tty
}

# use sudo only if we're not root & if available
if [ "$(id -u)" != '0' -a "`which sudo`" != '' ]
then udo='sudo'
else udo=''
fi

service="$udo service"
if ! which service > /dev/null; then
  service() # replace missing 'service' binary !
  {
    if [ $# -ne 2 ]; then
      xecho "Usage: `basename $0`.service name argument"
    fi
    $udo /etc/init.d/$1 $2
  }
fi

if which apt-get > /dev/null; then
  autoremove="$udo apt-get autoremove"
  buildDep="$udo apt-get build-dep"
  install="$udo apt-get -y -f install"
  installPack="$udo dpkg -i"
  remove="$udo apt-get -f -y remove"
  update="$udo apt-get -q update"
  upgrade="$udo apt-get upgrade"
elif which ipkg > /dev/null; then
```

```
  autoremove="xecho 'autoremove not implemented' #"
  buildDep="xecho 'buildDep not implemented' #"
  install="$udo ipkg install"
  installPack=$install
  remove="$udo ipkg remove"
  update="$udo ipkg update"
  upgrade="$udo ipkg upgrade"
else
  xecho 'Unable to find apt-get nor ipkg in your system'
fi

#if ! pushd . 2>/dev/null; then
#  recho 'pushd/popd as internal functions'
#  dirLifo=''
#  pushd()
#  {
#    if [ $# -ne 1 ]; then
#      xecho "Usage: `basename $0`.pushd path"
#    fi
#    dirLifo="`pwd`:$dirLifo"
#    cd "$1"
#  }
#  popd()
#  {
#    dir=`echo $dirLifo | cut -d ':' -f1`
#    dirLifo=`echo $dirLifo | cut -d ':' -f2-`
#    if [ "$dir" ]; then
#      cd "$dir"
#    else
#      xecho 'Paths LIFO is empty !'
#    fi
#  }
#else
#  recho 'pushd/popd as shell built-in'
#  popd
#fi

# unit-testing of the implementation !
#pushdTest()
#{
#  pushdUnitFailed="pushd/popd unit test failed !"
#  here=`pwd`
#  pushd /media && echo $dirLifo
#  if [ "`pwd`" != '/media' ]; then xecho "$pushdUnitFailed 1/5"; fi
#  cd /home
#  pushd /bin && echo $dirLifo
#  if [ "`pwd`" != '/bin' ]; then xecho "$pushdUnitFailed 2/5"; fi
#  popd && echo $dirLifo
#  if [ "`pwd`" != '/home' ]; then xecho "$pushdUnitFailed 3/5 `pwd`"; fi
#  popd && echo $dirLifo
#  if [ "`pwd`" != "$here" ]; then xecho "$pushdUnitFailed 4/5"; fi
#}

# Asks user to confirm an action (with yes or no) ------------------------------
#> 0 (true value for if [ ]) if yes, 1 if no and (defaultChoice) by default
#1 : default (0 = yes / 1 = no)
#2 : question (automatically appended with [Y/n] ? / [y/N] ?)
```

```
yesOrNo()
{
  if [ $# -ne 2 ]; then
    xecho "Usage : yesOrNo default question\n\tdefault : 0=yes or 1=no 2='force
yes' 3='force no'"
  fi

  local default="$1"
  local question="$2"
  case $default in
  0 ) qecho "$question [Y/n]";;
  1 ) qecho "$question [y/N]";;
  2 ) REPLY=0; return 0 ;;
  3 ) REPLY=1; return 0 ;;
  * ) xecho "Invalid default value : $default";;
  esac

  while true; do
    read REPLY </dev/tty
    case "$REPLY" in
    '' ) REPLY=$default ;;
    'y' | 'Y' ) REPLY=0 ;;
    'n' | 'N' ) REPLY=1 ;;
    * ) REPLY='' ;;
    esac
    if [ "$REPLY" ]; then break; fi
    default='' # cancel default value
    recho "Please answer y for yes or n for no"
  done
}

# Utilities ========================================================================

# Add a repository if it isn't yet listed in sources.list ----------------------
# repositoryDebUrl : the debian URL (deb http://... natty contrib) of the repo.
addAptRepo()
{
  if [ $# -ne 1 ]; then
    xecho 'Usage : addAptRepo repositoryDebUrl'
  fi

  if ! grep -q "$1" /etc/apt/sources.list; then
    $udo sh -c "echo '$1' >> /etc/apt/sources.list"
  fi
}

# Add a 'ppa' repository trying to fix TODO ------------------------------------
# repositoryPpa : the PPA (eg : ppa:rabbitvcs/ppa) of the repo.
# repositoryName : the PPA name without ppa:/... TODO
addAptPpaRepo()
{
  if [ $# -ne 2 ]; then
    xecho 'Usage : addAptPpaRepo repositoryPpa repositoryName'
  fi

  local repositoryPpa="$1"
```

```
  local repositoryName="$2"

  local ok=''
  local here="`pwd`"
  local last="`lsb_release -cs`"
  cd /etc/apt/sources.list.d
  $udo rm -rf *$repositoryName*
  $udo add-apt-repository $repositoryPpa
  repositoryFile=`ls | grep $repositoryName`
  if [ ! "$repositoryFile" ]; then
    xecho "Unable to find $repositoryName's repository file"
  fi
  mecho "Repository file : $repositoryFile"
  for actual in "$last" 'oneiric' 'maverick' 'lucid'
  do
    $udo sh -c "sed -i -e 's:$last:$actual:g' $repositoryFile"
    mecho "Checking if the $repositoryName's repository does exist for $actual ..."
    if $update 2>&1 | grep -q $repositoryName; then
      mecho "Hum, the $repositoryName's repository does not exist for $actual"
      recho "Ok, trying the next one"
    else
      ok='yes'
      break
    fi
    last=$actual
  done
  cd "$here"
  if [ "$ok" ]
  then mecho "Using the $repositoryName's repository for $actual"
  else xecho 'Unable to find a suitable repository !'
  fi
}

# Add a GPG key to the system ----------------------------------------------
# gpgKeyUrl : the URL (deb http://....asc) of the GPG key
addGpgKey()
{
  if [ $# -ne 1 ]; then
    xecho 'Usage : addGpgKey gpgKeyUrl'
  fi

  wget -q "$1" -O- | $udo apt-key add -
}

# Install a package if it isn't yet installed ------------------------------
# packageName : name of the package to install
# binaryName  : name of the binary to find
autoInstall()
{
  if [ $# -ne 2 ]; then
    xecho 'Usage : autoInstall packageName binaryName'
  fi

  local packageName="$1"
  local binaryName="$2"
```

```
  # install the package if missing
  if which "$binaryName" > /dev/null; then
    recho "Binary $binaryName of package $packageName founded, nothing to do"
  else
    recho "Binary $binaryName of package $packageName missing, installing it"
    $install $packageName || xecho "Unable to install package $packageName !"
  fi
}

# Install a package if it isn't yet installed ---------------------------------
# libName : name of the package to install (library)
autoInstallLib()
{
  if [ $# -ne 1 ]; then
    xecho 'Usage : autoInstallLib libName'
  fi

  # install the libs package if missing
  if dpkg --get-selections | grep "$1" | grep -q install; then
    recho "Library $1 founded, nothing to do"
  else
    recho "Library $1 missing, installing it"
    $install $1
  fi
}

# Install a package (with a setup method) if it isn't yet installed ------------
# setupName  : name of the (setup) method to execute
# binaryName : name of the binary to find
autoInstallSetup()
{
  if [ $# -ne 2 ]; then
    xecho 'Usage : autoInstallSetup setupName binaryName'
  fi

  local setupName="$1"
  local binaryName="$2"

  # install the package if missing
  if which "$binaryName" > /dev/null; then
    recho "Binary $binaryName of setup $setupName founded, nothing to do"
  else
    recho "Binary $binaryName of setup $setupName missing, installing it"
    $setupName
  fi
}

# Extract a debian package ----------------------------------------------------
debianDepack()
{
  if [ $# -ne 1 ]; then
    xecho "Usage: `basename $0` debianFilename"
  fi

  local name=`basename "$1" .deb`
  dpkg-deb -x "$1" "$name"
```

```
  mkdir "$name/DEBIAN"
  dpkg-deb -e "$1" "$name/DEBIAN"
}

# Create a debian package of a folder ----------------------------------------
debianRepack()
{
  if [ $# -ne 1 ]; then
    xecho "Usage: `basename $0` debianPath"
  fi

  dpkg-deb -b "$1"
}

checkDepend()
{
  if [ $# -ne 1 ]; then
    xecho 'Usage : checkDepend binaryName methodName'
  fi

  if ! which "$1" > /dev/null; then
    xecho "Dependency : $2 depends of $1, unable to find $1"
  fi
}

validateNumber()
{
  if [ $# -ne 1 ]; then
    xecho "Usage: `basename $0`.validateNumber input"
  fi
  if [ "$1" -eq "$1" 2>/dev/null ]
  then return 0
  else return 1
  fi # FIXME do the same thing with less code ?
}

# Get the Nth first digits of the IPv4 address of a network interface ----------
#> The address, ex: 192.168.1.34 -> [3 digits] 192.168.1. [1 digit] 192.
# ethName : name of the network interface to get ...
# numberOfDigitsRequired : number of digits to return (1-4)
getInterfaceIPv4()
{
  if [ $# -ne 2 ]; then
    xecho 'Usage : getInterfaceIPv4 ethName numberOfDigitsRequired'
  fi

  local ethName="$1"
  local numberOfDigitsRequired="$2"

  # find the Nth first digits of the ip address of a certain network interface,
  # this method use regular expression to filter the output of ifconfig
  cmd=`ifconfig $ethName`
  case "$numberOfDigitsRequired" in
  '1' ) REPLY=`expr match "$cmd" '.*inet ad\+r:\([0-9]*\.\)[0-9]*\.[0-9]*\.[0-
9]*'`;;
```

```
  '2' ) REPLY=`expr match "$cmd" '.*inet ad\+r:\([0-9]*\.[0-9]*\.\)[0-9]*\.[0-
9]*'`;;
  '3' ) REPLY=`expr match "$cmd" '.*inet ad\+r:\([0-9]*\.[0-9]*\.[0-9]*\.\)[0-
9]*'`;;
  '4' ) REPLY=`expr match "$cmd" '.*inet ad\+r:\([0-9]*\.[0-9]*\.[0-9]*\.[0-
9]*\)'`;;
  * ) xecho 'numberOfDigitsRequired must be between 1 and 4' ;;
  esac

  # FIXME : check du parsing !
}

validateIP()
{
  if [ $# -ne 1 ]; then
    xecho "Usage: `basename $0`.validateIP ip"
  fi
  if [ `echo $1 | sed -n "/^[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*$/p"` ]
  then return 0
  else return 1
  fi # FIXME do the same thing with less code ?
}

validateMAC()
{
  if [ $# -ne 1 ]; then
    xecho "Usage: `basename $0`.validateMAC mac"
  fi
  if [ `echo $1 | sed -n "/^\([0-9A-Za-z][0-9A-Za-z]:\)\{5\}[0-9A-Za-z][0-9A-Za-z]
$/p"` ]
  then return 0
  else return 1
  fi # FIXME do the same thing with less code ?
}

# http://freesoftware.zona-m.net/how-automatically-create-opendocument-invoices-
without-openoffice

# Apply sed in a [Libre|Open] Office document ----------------------------------
# oooSrcFilename : name of the source [Libre|Open] office file
# oooDstFilename : name of the destination [Libre|Open] office file
# paramsFilename : name of the params file (a couple of param value by line)
oooSed()
{
  if [ $# -ne 3 ]; then
    xecho 'Usage : oooSed oooSrcFilename oooDstFilename paramsFilename'
  fi

  local work_dir='/tmp/OOO_SED'
  local oooSrcFilename="$1"
  local oooDstFilename="$2"
  local paramsFilename="$3"

  mecho "Apply sed in a [Libre|Open] Office document"
  mecho "Source          : $oooSrcFilename"
  mecho "Destination     : $oooDstFilename"
```

```
  mecho "Sed parameters : $paramsFilename"

  rm -rf $work_dir
  mkdir  $work_dir
  # FIXME local filename instead of filename, + test behaviour !
  filename=`basename $oooSrcFilename`
  filename=`echo ${filename%.*}`

  cp $oooSrcFilename $work_dir/my_template
  cp $paramsFilename $work_dir/my_data.sh

  # preparation
  cd     $work_dir
  mkdir  work
  mv     my_template work
  cd     work
  unzip  my_template > /dev/null
  rm     my_template

  # replace text strings
  local content="`cat content.xml`"
  local styles="`cat styles.xml`"

  # parse params list line by line to find
  #          param value
  while read param value
  do
    if [ "$read$param" ]; then
      echo "s#$param#$value#g"
      content=$(echo $content | sed "s#$param#$value#g")
      styles=$(echo $styles | sed "s#$param#$value#g")
    fi
  done < ../my_data.sh # redirect done before while loop

  rm -f content.xml
  echo "$content" > content.xml

  rm -f styles.xml
  echo "$styles" > styles.xml

  # zip everything, rename it as .od* file and clean up
  find . -type f -print0 | xargs -0 zip ../$filename > /dev/null
  cd ..
  mv ${filename}.zip $oooDstFilename
  cd ..
  rm -rf $work_dir
}

# http://dag.wieers.com/home-made/unoconv/

# Convert a [Libre|Open] office document to a PDF with unoconv -----------------
# oooSrcFilename : name of the [Libre|Open] office document to convert
oooToPdf()
{
  if [ $# -ne 3 ]; then
    xecho 'Usage : oooToPdf oooSrcFilename'
```

```
  fi

  unoconv -v --format pdf $1
}

# END OF LOGICIELS UBUNTU UTILS
```

7.4.4 dashInstall (revision 205)

```
. dashCommon

main()
{
  # gpac installation inspired by
  # http://gpac.wp.institut-telecom.fr/2011/04/20/compiling-gpac-on-ubuntu/

  # vlc installation inspired by
  # http://linuxers.org/howto/how-install-vlc-11-compiling-git-ubuntu-linux

  # lighttpd installation and configuration inspired by
  # http://library.linode.com/web-servers/lighttpd/ubuntu-9.10-karmic

  $buildDep ffmpeg gpac vlc x264
  $install libtool build-essential automake git libxcb-composite0-dev subversion
  $install chromium-browser chromium-browser-l10n chromium-codecs-ffmpeg-extra

  total=11

  pecho "01/$total Getting DASH Conformance software"
  if [ ! -d "$DASH_CONFORMANCE_PATH" ]; then
    cd "$DASH_BUILD_PATH"
    svn co https://subversion.assembla.com/svn/DASH_Conformance/ \
        --username=dash_conformance
  fi

  pecho "02/$total Getting DASH Encoder source code"
  if [ ! -d "$DASH_ENCODER_PATH" ]; then
    cd "$DASH_BUILD_PATH"
    git clone --depth 1 https://github.com/slederer/DASHEncoder.git
  fi

  pecho "03/$total Getting FFMPEG source code"
  if [ ! -d "$DASH_FFMPEG_PATH" ]; then
    cd "$DASH_BUILD_PATH"
    git clone --depth 1 git://source.ffmpeg.org/ffmpeg.git
  fi

  pecho "04/$total Getting GPAC source code"
  if [ ! -d "$DASH_GPAC_PATH" ]; then
    cd "$DASH_BUILD_PATH"
    #svn co https://gpac.svn.sourceforge.net/svnroot/gpac/trunk/gpac@3744 gpac
    svn co https://gpac.svn.sourceforge.net/svnroot/gpac/trunk/gpac gpac
  fi

  pecho "05/$total Getting GSTREAMER source code"
  if [ ! -d "$DASH_GSTREAMER_PATH" ]; then
    cd "$DASH_BUILD_PATH"
```

```
    git clone --depth 1 git://sourcecode.opera.com/gstreamer/gst-opera.git
gstreamer
  fi

  pecho "06/$total Getting <HTML5 /> MediaSource API source"
  if [ ! -d "$DASH_HTML5_API_PATH" ]; then
    cd "$DASH_BUILD_PATH"
    svn co http://html5-mediasource-api.googlecode.com/svn/trunk html5-mediasource-
api
  fi

  pecho "07/$total Getting DASH Javascript source code"
  if [ ! -d "$DASH_JAVASCRIPT_PATH" ]; then
    cd "$DASH_BUILD_PATH"
    git clone --depth 1 git://github.com/dazedsheep/DASH-JS.git
  fi

  pecho "08/$total Getting DASH Library source code"
  if [ ! -d "$DASH_LIB_PATH" ]; then
    cd "$DASH_BUILD_PATH"
    svn co https://svn-itec.uni-klu.ac.at/repos2/dash/trunk libdash
  fi

  pecho "09/$total Getting DASH Library for Android source code"
  if [ ! -d "$DASH_STERICSSON_PATH" ]; then
    cd "$DASH_BUILD_PATH"
    git clone https://code.google.com/p/test-code-hosting-manuele/ STMicro
  fi

  pecho "10/$total Getting VideoLan Client source code"
  if [ ! -d "$DASH_VLC_PATH" ]; then
    cd "$DASH_BUILD_PATH"
    git clone --depth 1 git://git.videolan.org/vlc.git
  fi

  pecho "11/$total Getting x264 source code"
  if [ ! -d "$DASH_X264_PATH" ]; then
    cd "$DASH_BUILD_PATH"
    git clone --depth 1 git://git.videolan.org/x264.git
  fi

  #gpacFromNightlyBuilds # gpac from the nightly builds, no more headaches ;-)
  #vlcFromNightlyBuilds  # vlc  from the nightly builds, no more headaches ;-)

  name=$USER
  root='server.document-root'
  user='server.username'
  group='server.groupname'
  $install lighttpd
  $udo sh -c "sed -i -e 's:$root.*:$root = \"$DASH_MEDIAS_PATH\":' \
                     -e 's:$user.*:$user = \"$name\":' \
                     -e 's:$group.*:$group = \"$name\":'
/etc/lighttpd/lighttpd.conf"
  $udo mkdir -p /var/log/lighttpd/
  $udo chown $name:$name /var/log/lighttpd/ -R
  $service lighttpd restart
```

```
}

gpacFromNightlyBuilds()
{
  pecho "12/$total Installing GPAC binaries from nightly builds"
  #trap "rm -rf '$DASH_BUILD_PATH/tmp-gpac' 2>/dev/null" INT TERM EXIT
  here=`pwd`
  mkdir -p "$DASH_BUILD_PATH/tmp-gpac"
  cd "$DASH_BUILD_PATH/tmp-gpac"
  wget http://gpac.wp.institut-telecom.fr/downloads/gpac-nightly-builds/
  line1=`cat index.html | grep 'Latest available linux64/gpac binaries'`
  line2=`cat index.html | grep 'Latest available linux64/libgpac-dev binaries'`
  url1=`expr match "$line1" '.*<a href="\(.*.deb\)">.*'`
  url2=`expr match "$line2" '.*<a href="\(.*.deb\)">.*'`
  if [ ! "$url1" -o ! "$url2" ]; then
    xecho 'Unable to parse gpac-nightly-builds page to get latest packages'
  fi
  for url in "$url1" "$url2"; do
    pack=`basename "$url"`
    name=`basename "$pack" .deb`
    wget "$url" && debianDepack "$pack" || exit 1
    # FIXME dirty dependencies update
    sed -i -e 's:52 :53 :g' \
           -e 's:49 :51 :g' \
           -e 's:libswscale0:libswscale2:g' \
           -e 's:libdirectfb-1.2-0:libdirectfb-1.2-9:g' "$name/DEBIAN/control"
    debianRepack "$name" || exit 1
    if ! $udo dpkg -i "$pack"; then
      recho 'no worry, just resolving gpac dependencies ...'
      $install
    fi
  done

  if ! which MP4Box > /dev/null; then
    xecho "Unable to install $pack"
  fi

  if MP4Box 2>&1 | grep -q libmozjs.so; then
    recho 'installing shared library libmozjs.so for MP4Box'
    # resolve MP4Box: error while loading shared libraries: libmozjs.so: ...
    $install kompozer
    $udo find /usr/lib -name libmozjs.so -exec ln -s "{}" /usr/local/lib \;
    $udo ldconfig
  fi

  cd "$here"
}

vlcFromNightlyBuilds()
{
  pecho "13/$total Installing VideoLan Client binaries from nightly builds"
  if ! addAptPpaRepo 'ppa:videolan/master-daily' 'videolan'; then
    xecho 'Unable to install vlc nightly builds repository !'
  fi
  $install vlc
}
```

```
main
```

7.4.5 dashUninstall (revision 205)

```
. dashCommon

#cd "$DASH_ENCODER_PATH" && $udo make uninstall
cd "$DASH_FFMPEG_PATH"    && $udo make uninstall
cd "$DASH_GPAC_PATH"      && $udo make uninstall
cd "$DASH_GSTREAMER_PATH" && $udo make uninstall
cd "$DASH_VLC_PATH"       && $udo make uninstall
cd "$DASH_X264_PATH"      && $udo make uninstall

$remove lighttpd
$autoremove
```

7.4.6 dashUpdate (revision 205)

```
. dashCommon

dashUpdate()
{
  techo "DASH Open-Source Demonstrator [EBU/UER Technical]" # TODO is it okay ?
  techo '------------------------- copyright David Fischer'
  techo
  pecho "Update DASH (project) local copy"
  pecho

  if [ $# -gt 1 ]; then
    xecho "Usage: `basename $0` revisionNumber"
  fi

  yesOrNo 0 'update DASH (project) local copy'
  if [ $REPLY -eq 0 ]; then
    cd "$DASH_PATH"
    autoInstall subversion svn
    if [ $# -eq 1 ]
    then svn update -r $1
    else svn update
    fi
    find . -maxdepth 1 -type f -not -path "*.svn*" -name "dash*" \
      -exec svn propset -q svn:executable '' {} \; -exec chmod u+x {} \;
    find scripts -type f -not -path "*.svn*" \
      -exec svn propset -q svn:executable '' {} \; -exec chmod u+x {} \;
    # generate dashCommon = dashCommon.lu-dep + logicielsUbuntuUtils
    # It does add some utilities from my personnal project "LogicielsUbuntu" !
    if which lu-importUtils > /dev/null; then
      lu-importUtils . 'no'
    fi
  fi

  updateGit "$DASH_ENCODER_PATH"    'DASH Encoder'
  updateGit "$DASH_FFMPEG_PATH"     'FFMPEG'
  updateSvn "$DASH_GPAC_PATH"       'GPAC'
  updateGit "$DASH_GSTREAMER_PATH"  'GSTREAMER'
  updateSvn "$DASH_HTML5_API_PATH"  '<HTML5 /> MediaSource API'
```

```
  updateGit "$DASH_JAVASCRIPT_PATH" 'DASH Javascript'
  updateSvn "$DASH_LIB_PATH"        'DASH Library'
  updateGit "$DASH_STERICSSON_PATH" 'DASH Library for Android'
  updateGit "$DASH_VLC_PATH"        'VideoLan Client'
  updateGit "$DASH_X264_PATH"       'x264'

  yesOrNo 0 'update local MPDs URLs'
  if [ $REPLY -eq 0 ]; then
    ip='127.0.0.1'
    echo "$ip" > "$DASH_VMS_PATH/VmDASH-Web.ip"
    mecho "VmDASH-Web.ip generated"

    listing=/tmp/$$
    trap "rm -f '$listing' 2>/dev/null" INT TERM EXIT
    find "$DASH_MPD_PATH" -name "*.template" > $listing
    while read template
    do
      dest=`basename "$template" .template`
      sed "s:VM_IP:$ip:g" < "$template" > "$DASH_MPD_PATH/$dest"
      mecho "$dest generated"
    done < $listing
  fi
}

updateGit()
{
  if [ $# -ne 2 ]; then
    xecho "[BUG] Usage: `basename $0`.updateGit path name"
  fi

  path=$1
  name=$2

  if [ -d "$path" ]; then
    c1="cd $path"
    c2='git reset --hard'
    c3='git pull'
    cecho "$c1"; cecho "$c2"; cecho "$c3"
    yesOrNo 0 "update $name local copy"
    if [ $REPLY -eq 0 ]; then $c1; $c2; $c3; fi
  fi
}

updateSvn()
{
  if [ $# -ne 2 ]; then
    xecho "[BUG] Usage: `basename $0`.updateSvn path name"
  fi

  path=$1
  name=$2

  if [ -d "$path" ]; then
    c1="cd $path"
    c2='svn cleanup'
    c3='svn update'
```

```
    cecho "$c1"; cecho "$c2"; cecho "$c3"
    yesOrNo 0 "update $name local copy"
    if [ $REPLY -eq 0 ]; then $c1; $c2; $c3; fi
  fi
}

dashUpdate
```

7.4.7 dashCompile (revision 205)

```
. dashCommon

dashCompile()
{
  cpuCount=`grep -c 'model name' /proc/cpuinfo`
  jobCount=$((2*cpuCount))

  while true
  do
    qecho "how many // jobs for compilation [$jobCount]"
    read n
    if [ "$n" -eq "$n" 2>/dev/null ]; then
      jobCount=$n; break
    elif [ ! "$n" ]
    then break
    else recho 'Illegal number, please try again'
    fi
  done

  compile "$DASH_ENCODER_PATH"   DASHEncoder 'patchNull'
  compile "$DASH_FFMPEG_PATH"    ffmpeg      'patchFfmpeg'
  compile "$DASH_GPAC_PATH"      gpac        'patchGpac' 'bin/gcc/libgpac.so'
  compile "$DASH_GSTREAMER_PATH" gstreamer   'patchNull'
  # FIXME HTML5_API
  # not needed DASH-JS
  # FIXME LIB
  compile "$DASH_VLC_PATH"       vlc  'patchVlc'
  compile "$DASH_X264_PATH"      x264 'patchX264'
}

compile()
{
  if [ $# -ne 3 -a $# -ne 4 ]; then
    xecho "[BUG] Usage: `basename $0`.compile path name patch [library]"
  fi

  path=$1
  name=$2
  patch=$3
  library=$4

  # is already compiled ?
  if [ -f "$path/$name" -o $# -eq 4 -a -f "$path/$library" ]
  then default=1
  else default=0
  fi
```

```
  bootstrapTime='-'
  configureTime='-'
  makeTime='-'
  installTime='-'

  if [ -f "$path/bootstrap" ]; then
    yesOrNo $default "bootstrap $name (needed for the 1st compilation)"
    default=$REPLY
    if [ $REPLY -eq 0 ]; then
      cd "$path" && $patch 'bootstrap' && chmod u+x bootstrap || exit 1
      startDate=`date +%s`; ./bootstrap || exit 1
      endedDate=`date +%s`
      bootstrapTime=$((endedDate-startDate))
    fi
  fi

  if [ -f "$path/configure" ]; then
    yesOrNo $default "configure $name (needed for the 1st compilation)"
    default=$REPLY
    if [ $REPLY -eq 0 ]; then
      cd "$path" && $patch 'configure' && chmod u+x configure || exit 1
      startDate=`date +%s`; ./configure $configureOptions || exit 1
      endedDate=`date +%s`
      configureTime=$((endedDate-startDate))
    fi
  fi

  yesOrNo $default "compile $name ($jobCount parallel jobs)"
  default=$REPLY
  if [ $REPLY -eq 0 ]; then
    cd "$path" && $patch 'make' || exit 1
    startDate=`date +%s`; make -j$jobCount $makeOptions || exit 1
    endedDate=`date +%s`
    makeTime=$((endedDate-startDate))
  fi

  yesOrNo $default "install $name on this system"
  default=$REPLY
  if [ $REPLY -eq 0 ]; then
    cd "$path" || exit 1
    startDate=`date +%s`; $udo make install
    endedDate=`date +%s`
    installTime=$((endedDate-startDate))
  fi

  mecho "Some statistics about $name compilation :"
  mecho "$bootstrapTime secs for boostrap"
  mecho "$configureTime secs for configure"
  mecho "$makeTime secs for make"
  mecho "$installTime secs for install"
}

patchNull()
{
  configureOptions=''
  makeOptions=''
```

```
  return 0
}

patchFfmpeg()
{
  if [ $# -ne 1 ]; then
    xecho "[BUG] Usage: `basename $0`.patchFfmpeg mode"
  fi
  mode=$1
  configureOptions='--enable-pthreads --enable-libv4l2 --enable-gpl --enable-
libx264' # --extra-cflags=-fPIC'
  makeOptions=''
  # for the android app of STMicroelectronics : --enable-libstagefright-h264'
  return 0
}

patchGpac()
{
  if [ $# -ne 1 ]; then
    xecho "[BUG] Usage: `basename $0`.patchGpac mode"
  fi
  mode=$1
  if [ "$mode" = 'configure' ]; then
    # resolve undefined reference to `jpeg_std_error@LIBJPEG_6.2'
    mecho 'patching gpac configure ...'
    e='GPAC_SH_FLAGS=-lpthread'
    f="GPAC_SH_FLAGS='-lpthread -ljpeg'"
    sed -i -e "s:$e:$f:" "$DASH_GPAC_PATH/configure" || return 1
  fi
  configureOptions='' # '--extra-cflags=-fPIC'
  makeOptions=''
  return 0
}

patchVlc()
{
  if [ $# -ne 1 ]; then
    xecho "[BUG] Usage: `basename $0`.patchVlc mode"
  fi
  mode=$1
  if [ "$mode" = 'make' ]; then
    mecho 'patching vlc make-alias ...'
    # resolve /bin/bash: ./make-alias: Permission non accordée
    chmod u+x "$DASH_VLC_PATH/make-alias" || return 1
  fi
  configureOptions='--disable-postproc'
  makeOptions=''
  return 0
}

patchX264()
{
  if [ $# -ne 1 ]; then
    xecho "[BUG] Usage: `basename $0`.patchX264 mode"
  fi
  mode=$1
```

```
  if [ "$mode" = 'configure' ]; then
    mecho 'patching x264 config.* ...'
    # resolve ./configure: ... ./config.guess: Permission non accordée
    # resolve ./configure: ... ./config.sub: Permission non accordée
    chmod u+x "$DASH_X264_PATH/config.guess" || return 1
    chmod u+x "$DASH_X264_PATH/config.sub"   || return 1
  fi
  configureOptions=''
}

dashCompile
```

7.4.8 dashClean (revision 205)

```
. dashCommon

cd "$DASH_ENCODER_PATH"   && make clean
cd "$DASH_FFMPEG_PATH"    && make clean
cd "$DASH_GPAC_PATH"      && make clean
cd "$DASH_GSTREAMER_PATH" && make clean
cd "$DASH_VLC_PATH"       && make clean
cd "$DASH_X264_PATH"      && make clean
```

7.4.9 dashRun (revision 205)

```
. dashCommon

title='DASH Run Menu TODO'

dashRun()
{
  autoInstall dialog dialog

  $service lighttpd restart

  tmpfile=/tmp/$$
  listing=/tmp/$$.list
  trap "rm -f '$tmpfile' '$listing' 2>/dev/null" INT TERM EXIT

  while true
  do
    runList="runEncoder    DASHEncoder \
             runJavascript DASH-JS     \
             runVlc        VLC"
    $DIALOG --backtitle "$title" \
            --menu 'Select one of the following program' 0 0 0 \
            $runList 2> $tmpfile

    retval=$?
    runItNow=`cat $tmpfile`
    [ $retval -eq 1 -o $retval -eq 255 -o ! "$runItNow" ] && return

    $runItNow
    pause
  done
}
```

```
runEncoder()
{
  if [ ! -f "$DASH_ENCODER_PATH/DASHEncoder" ]; then
    xecho "Compiled DASH Encoder binary doesn't exist !"
  fi

  $udo ldconfig /usr/lib/firefox # resolve unable to load libxul.so, ...

  configList=''
  find "$DASH_MEDIAS_PATH" -type f -name "*.config" | sort > $listing
  while read config; do configList="$configList$config - "
  done < $listing

  $DIALOG --backtitle "$title" \
          --menu 'Select a media to encoder by DASH Encoder' 0 0 0 \
          $configList 2> $tmpfile

  retval=$?
  config=`cat $tmpfile`
  [ $retval -eq 1 -o $retval -eq 255 -o ! "$config" ] && return

  yesOrNo 0 "encode it now : $config"
  if [ $REPLY -eq 0 ]; then
    cd "$DASH_ENCODER_PATH"
    sed "s:DASH_MEDIAS_PATH:$DASH_MEDIAS_PATH:" <"$config" >DASHEncoder.config
    ./DASHEncoder
  else
    recho 'operation cancelled by the user'
  fi
}

runJavascript()
{
  if ! which chromium-browser > /dev/null; then
    xecho "Chromium browser must be installed !"
  fi

  chromium-browser http://www-itec.uni-klu.ac.at/dash/?page_id=746
}

runVlc()
{
  if [ ! -f "$DASH_VLC_PATH/vlc" ]; then
    xecho "Compiled VLC binary doesn't exist !"
  fi

  sourceList=''
  find "$DASH_MPD_PATH" -maxdepth 1 -type f -name "*.mpd" | sort > $listing
  while read mpd
  do
    name=`echo $mpd | sed "s:$DASH_MPD_PATH/::g"`
    if echo "$name" | grep -q 'local'
    then mode='local'
    else mode='remote'
    fi
    sourceList="$sourceList$name $mode "
```

```
  done < $listing

  find "$DASH_MPD_PATH" -maxdepth 1 -type f -name "*.list" | sort > $listing
  while read list
  do
    name=`echo $list | sed "s:$DASH_MPD_PATH/::g"`
    sourceList="$sourceList$name web-page "
  done < $listing

  $DIALOG --backtitle "$title" \
          --menu 'Select a source to play by VLC' 0 0 0 \
          $sourceList 2> $tmpfile

  retval=$?
  source=`cat $tmpfile`
  [ $retval -eq 1 -o $retval -eq 255 -o ! "$source" ] && return

  if echo "$source" | grep -q 'local'; then
    source="http://$DASH_WEB_IP/MPD/$source"
  elif echo "$source" | grep -q '.list'; then
    dashParseList "$DASH_MPD_PATH/$source" || return
  else
    xecho 'not implemented'
  fi

  yesOrNo 0 "play it now : $source"
  if [ $REPLY -eq 0 ]; then
    cd "$DASH_VLC_PATH"
    ./vlc "$source"
  else
    recho 'operation cancelled by the user'
  fi
}

dashParseList()
{
  if [ $# -ne 1 ]; then
    xecho "[BUG] `basename $0`.dashParselist filename"
  fi

  source=''

  urlsList=''
  while read url
  do
    urlsList="$urlsList$url - "
  done < "$1"

  $DIALOG --backtitle "$title" \
          --menu 'Select a source to play by VLC' 0 0 0 \
          $urlsList 2> $tmpfile

  retval=$?
  url=`cat $tmpfile`
  [ $retval -eq 1 -o $retval -eq 255 -o ! "$url" ] && return 1
```

```
  index=/tmp/$$-index
  wget -q "$url" -O $index
  if [ ! -f $index ]; then return 1; fi

  mpdsList=''
  while read line
  do
    mpd=`expr match "$line" '.*href="\(.*\.mpd\)".*'`
    if [ "$mpd" ]; then
      mpdsList="$mpdsList$mpd - "
    fi
  done < $index
  rm -f $index

  $DIALOG --backtitle "$title" \
          --menu 'Select a source to play by VLC' 0 0 0 \
          $mpdsList 2> $tmpfile

  retval=$?
  mpd=`cat $tmpfile`
  [ $retval -eq 1 -o $retval -eq 255 -o ! "$mpd" ] && return 1

  source=$url$mpd # Finally we got it !
  return 0
}

dashRun
```

7.4.10 dashStatistics (revision 205)

```
. dashCommon

#dashStatistics()
{
  techo "DASH Open-Source Demonstrator [EBU/UER Technical]" # TODO is it okay ?
  techo '------------------------- copyright David Fischer'
  techo
  pecho "Update DASH (project) statistics with StatSVN"
  pecho

  #options="-include scripts/**:report/** -exclude
administration/**:build/**:medias/**:references/**:statistics/**:todo/**:**/*.ova:*
*/*.pdf:**/*.jpg:**/*.png:**/*.mpd*"

  yesOrNo 0 'update DASH (project) statistics'
  if [ $REPLY -eq 0 ]; then
    cd "$DASH_SCRIPTS_PATH"
    autoInstall statsvn statsvn

    tmpFilename=/tmp/$$
    trap "rm -f '$tmpFilename' 2>/dev/null" INT TERM EXIT

    mecho '1/2 getting repository verbose log ...'
    url=`svn status | grep ^URL | sed 's/URL.*: //'`
    svn log -v --xml > $tmpFilename

    mecho '2/2 using statsvn to generate statistics ...'
```

```
    rm -f "$DASH_STATS_PATH/*" 2>/dev/null
    statsvn $options $tmpFilename . -output-dir "$DASH_STATS_PATH"
  fi
}
```

7.4.11 dash-bwSet (revision 205) abandon-ware

```
. dashCommon

if [ $# -ne 1 ]; then
  xecho "Usage: `basename $0` limitKbps"
fi

autoInstall wondershaper wondershaper
if [ $1 -eq 0 ]
then $udo wondershaper clear eth0
else $udo wondershaper eth0 $1 $1
fi
```

7.4.12 dash-bwTest (revision 205) abandon-ware

```
. dashCommon

while true
do
  wget "$DASH_BW_SEG_FILENAME" -O /dev/null
done
```