

# Protected Delivery of Multimedia Contents over Multicast IP Networks: an Open-Source Approach

Lorenz Klopfenstein   Saverio Delpriori   Matteo Valentini   Andrea Seraghi   Alessandro Bogliolo

DSBF – STI – University of Urbino

Urbino, Italy

e-mail: lck@klopfenstein.net, saveriodelpriori@gmail.com, valentin@sti.uniurb.it,  
{andrea.seraghi, alessandro.bogliolo}@uniurb.it

**Abstract**—Internet traffic trends and forecasts clearly show that audio-video streaming is a killer application for next generation networks. Providing multicast support within operators’ managed networks is mandatory to sustain the exponential growth of demand for multimedia contents without saturating the bandwidth available in core, backhauling, and access infrastructures. This work presents an experimental setup, built on top of an open source platform called *openBOXware*, which enables protected delivery of multimedia contents over unprotected multicast IP networks.

**Keywords** – *multicast; multimedia; streaming; open source; protection;*

## I. INTRODUCTION

According to the Visual Networking Index Forecast published by Cisco in June 2010 [1], consumer IP traffic will account for 87% of the overall aggregate traffic in 2014 and nearly 60% of this share will be taken by Internet video streaming and download. Although only 5% of Internet videos are currently watched on a TV set, the amount of Internet-to-TV traffic is expected to double every year, reaching a share of 17% of the overall Internet video traffic in 2014. In the meantime, mobile video will reach a share of 66% of mobile data traffic, with the highest annual growth among all mobile applications. These data confirm the leading role of multimedia applications in the development of the Internet, and the growing demand for customers to be able to watch video contents on any device, anywhere, and anytime. As more and more multimedia contents migrate to IP networks, a paradigm shift is required to enable different customers to choose their preferred use mode, ranging from the “lean forward” experience offered by web applications running on PCs to the “lean back” living room experience of traditional television.

If, on one hand, this unrestrainable trend promises to pave the way to the development of next generation networks, on the other hand, its potential could be hindered by several challenging issues including: the bandwidth saturation in under-provisioned IP access networks, the lack of portable platforms enabling a thorough “develop once and deploy many” paradigm, the common adoption of business models which are unsuitable to promote innovation [2], and the persistence of cultural and infrastructural digital divide in sparsely populated and rural areas [3].

It is a well known fact that multicasting could be used to deliver multimedia contents to many end-users without adding to bandwidth requirements. Moreover, it has been recently shown that the inherent broadcasting capabilities of wireless access points could be suitably exploited to make it possible to distribute high definition TV channels with bandwidth and computational requirements independent of the number of simultaneous end-users watching them [4]. However, multimedia contents streamed to a multicast group can be freely accessed by unregistered users who associate to the group. This impedes the application of billing models and raises content protection issues, which are particularly challenging in case of wireless open access networks [5]. In a typical IPTV scenario these issues are overcome by adopting broadcaster-specific *digital rights management* (DRM) techniques which rely on dedicated hardware [6], while no solutions exist in more open network environments.

This work demonstrates the possibility of using an open source platform (namely, *openBOXware*) to safeguard multimedia contents delivered on top of open, unprotected multicast IP networks. OpenBOXware is introduced in Section II, while Section III outlines the protection method and the experimental setup used for demonstration purposes.

## II. OPENBOXWARE

OpenBOXware is a portable open-source framework for the development of bandwidth-aware multimedia applications. Its main feature is the capability of handling multimedia flows streamed from heterogeneous sources to both local and remote targets. The framework automatically creates the pipelines between media sources and media targets, possibly including the required transcoding stages. The capability of handling multiple simultaneous pipelines while streaming them to remote targets makes it suitable to be installed not only at the receiving end point, but also at the intermediate nodes of a content delivery network. In particular, openBOXware provides support for incoming and outgoing multicast streams, thus enabling the implementation of bandwidth-aware content distribution mechanisms within managed IP networks.

OpenBOXware has a layered architecture which grants portability by abstracting the underlying HW platform and

software components. The first level, which is the *kernel*, includes all actual components and makes them interact with each other. The second level, which is composed of the *application programming interfaces* (API), provides the abstraction to the kernel functionalities. The highest level includes all kinds of software components which rely on the API. In particular, a special component, called *skin*, acts as application manager and determines the main user interface of the platform. By changing the skin it is possible to change the usage experience while maintaining compatibility with all other applications. *Media sources* and *media targets* are also implemented as high-level addins which make the corresponding resources available to any other component willing to use them. For instance, a media library application could allow the end-user to bind a media source to a media target and to automatically create the pipeline to stream data from the preferred source to the target of choice. Other applications include: custom pipelines with their own user interfaces, client-side web applications (ranging from social network clients and feed readers to a fully fledged web browser), and server-side applications (such as both media related and home automation UPnP services, or any kind of web services). Moreover, any application can expose a remote interface possibly built by taking advantage of the embedded web server.

Each application can run in fullscreen, in the sidebar, in a custom widget, or in background. The execution modes implemented by a given application are declared in its manifest, which is an XML file providing to the framework the information required to present it to end-users and to load its components. All applications, including those running in background, can also make use of the API to interact with a local user (by showing notifications and opening dialog boxes) or with a remote user or application (by exposing web/UPnP services).

OpenBOXware is currently implemented on top of *Mono*, an open source implementation of the .NET framework providing the required abstraction from the underlying hardware and software components. The multimedia subsystem relies on *GStreamer*, while the graphic user interface is based on *Qyoto*, a binding library of the *Qt* framework for .NET. Video output is enabled through the *XV* overlay mechanism of *X server*.

The current 1.2 software release is freely available from the official website of the openBOXware project, including its source code (<http://www.openboxware.net/trac/>). This first release has been developed and tested on x86 PCs running the *GNU/Linux* operating system, while work on a port to an ARM based embedded platform (namely, the *IGEPv2* board equipped with a *TI OMAP* processor [7]) is currently under way, specifically targeting the *MeeGo* operating system [8], the core distribution of which includes all required components and exists in both x86 and ARM flavors.

### III. DEMO SETUP

The experimental setup shown in Figure 1 encompasses 5 instances of openBOXware running on different devices (labeled from A to E) and playing different roles in the content distribution chain. In particular: A represents an HTTP streaming server managed by the broadcaster who also takes care of access control and billing; B is a proxy server which handles the unicast-to-multicast conversion to deliver the contents to end-users registered with A; D and E represent registered clients; C represents an unregistered client trying to associate with the multicast group.

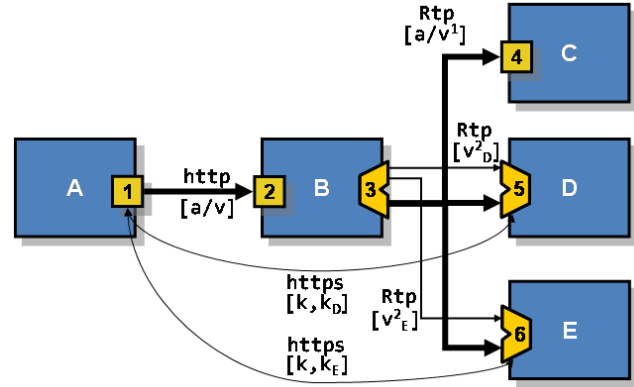


Fig. 1: Experimental setup.

The tradeoff between bandwidth sharing and selective delivery is achieved by splitting the multimedia stream into two parts delivered across separate channels. The first part (hereafter called S1 and denoted by the thick arrows labeled  $[a/v^1]$  in Figure 1) contains both the audio stream and most of the video packets, while the second one (called S2 and denoted by thin arrows labeled  $[V^2_D]$  and  $[V^2_E]$  in the figure) contains the remaining packets of the video stream. The original video stream is split randomly, by ensuring that at least a percentage of the original packets are sent to the lesser  $V^2$  streams. The  $V^1$  stream, which is much more bandwidth consuming, is multicasted and encrypted with a shared common key ( $k$ ) made known to all registered users. The second stream, which has a small impact on bandwidth, is encrypted with user-specific keys (namely,  $k_D$  and  $k_E$  in our setup) and delivered separately in unicast to each registered end-user. Both shared and user-specific keys are provided by A across an SSL channel upon user registration, making B transparent to end-users. An unregistered user (say C) who associates to the multicast group might be able to decrypt S1 (provided that he/she has come to know the shared key) but he cannot play it back without the missing part of the stream (S2), which is protected both by the direct unicast transmission and by the personal keys owned by registered end-users (namely, D and E in the experimental setup).

Components numbered from 1 to 6 represent the addins developed on top of openBOXware to implement the functionalities required at each node. Component 1 acts as a web server taking care of access control, session

management, billing, and content delivery to B. The proxy is implemented by combining a HTTP media source (2), receiving the original content from A, and a media target (3) which splits the audio/video stream into S1 and S2, encrypts them according to the strategy outlined above, and sends them to the multicast group and to the target clients. Component 4 implements a generic multicast media source, which is not able to decode the stream. Components 5 and 6 are instances of the same addin, which receives both S1 and S2, decrypts them, and re-assembles the original stream.

The entire distribution chain is initiated by the first client (say, D) issuing a request for a specific content. Subsequent requests for the same content issued by other clients (say, E) entail only the transmission of an additional unicast stream with a limited impact on bandwidth consumption.

Both bandwidth consumption and quality of experience will be monitored during the demo by instrumenting the experimental setup by means of IPTV testing equipment. The scalability and portability of the proposed solution will be also assessed by performing a stress test on the proxy running on different hardware platforms.

#### ACKNOWLEDGEMENT

The research leading to these results has received funding from the EU IST Seventh Framework Programme

([FP7/2007-2013]) under grant agreement n 25741, project ULOOP (User-centric Wireless Local Loop), and from the Italian ICT4University Programme, project U4U (University for University).

#### REFERENCES

- [1] Cisco, *Visual Networking Index Global Mobile Data Traffic Forecast*. Cisco White Paper, 2010.
- [2] A. T. Kearney, *A Viable Future Model of the Internet*, A. T. Kearney report, 2010.
- [3] M. R. Vicente and A. J. López, "Assessing the regional digital divide across the European Union-27", *Telecommunications Policy*, Vol. 35, No. 3, pp.220-237, 2011.
- [4] L. Klopfenstein, A. Seraghiti, S. Bonino, A. Tarasconi, and A. Bogliolo, "Multicast TV Channels over Wireless Neutral Access Networks", in *Proc. of Int. Conf. on Evolving Internet*, pp. 153-158, 2010.
- [5] J. Barceló, A. Sfairopoloulou, and B. Bellalta, "Wireless open metropolitan area networks", *SIGMOBILE Mob. Comput. Commun. Rev.*, Vol. 12, No. 4, pp. 34-44, 2008.
- [6] J. Maisonneuve, M. Deschanel, J. Heiles, Li Wei, Liu Hong, R. Sharpe, Wu Yiyuan. "An Overview of IPTV Standards Development", *IEEE Transactions on Broadcasting*, Vol. 55, No. 2, pp. 315-328, 2009.
- [7] IGEPv2 Board homepage, <http://www.igep.es/>, visited on March 2011.
- [8] I. Haddad. "Introduction to the MeeGo software platform", *Linux Journal*, Vol. 2010, No. 198, 2010.