A Secure and Legal Network-aware P2P VoD System

Bertrand Mathieu, Pierre Paris,
Gaëtan Le Guelvouit
Orange Labs – Lannion – France
{bertrand2.mathieu; pierre.paris;
gaetan.leguelvouit}@orange-ftgroup.com

Soufiane Rouibia
TMG – S[t] Sébastien sur Loire – France
rouibia@tmg.eu

***Abstract* — File sharing applications using Peer-to-Peer (P2P) networks such as Bittorrent or eDonkey rapidly attracted a lot of people and proved the efficiency and interest of this P2P technology. Distribution of video and of live contents also experienced the P2P mechanisms with success. PPLive, UUSee and others have many of customers, hundreds of channels and thousands of concurrent users. However, major content providers are reluctant to use this technology because no solution to ensure the distribution of only legal contents is provided. In the same way, network operators do not really push towards P2P content distribution because bad organization of the overlay can lead to overload the network and consume a lot of networks resources. In this paper, a secure and legal network-aware P2P video system is introduced, which aims at overcoming those two drawbacks. The design of the system and the evaluation of a prototype showed good results and let us be optimistic about a possible deployment of P2P systems for video delivery, having the support of content providers as well as network operators.**

Keywords - P2P Video Streaming, network-awareness, secure distribution, legal contents, watermarking.

I. INTRODUCTION

Peer-to-Peer (P2P) technologies are well-known via their rapid deployment (and success) for file sharing applications (Kazaa, eDonkey, BitTorrent, etc.). For some years, new applications has appeared like Voice (e.g., Skype) and video applications, which are becoming more and more popular on the Internet and some of them affirm their success with many millions of users (PPLive, PPStream UUSee, SopCast, etc..). However, those P2P networks are mostly used with illegal contents because no function is offered to check the legality of the contents distributed in such systems (making content providers reluctant to use it). Some commercial companies tried to use this P2P technology to offer legal video services such as the BBC (British Broadcasting Corporation) that proposed their catch-up TV service in P2P architecture. However, due to its success, it rapidly loaded underlying physical networks [1] because the peer selection algorithm did not take into consideration the underlying networks (because there were no cooperation between the P2P providers and the network operators). Those two limitations make that content providers as well as network operators are still reluctant to deploy and use a P2P delivery system because of security issues (legality of content, security of the system that can be attacked or infected) and load of the network (since no network-awareness in the peer selection). This paper aims at presenting a system that provides solutions to address those two limiting points and thus offers a secure and legal network-aware P2P video system. In section II of this paper, we first present briefly the overall picture of our system. Section III presents the components that have been added in the P2P delivery for taking into consideration the network topology. Section IV introduces the adapted P2P client for VoD system. Section V highlights of the watermarking of contents, which enables to ensure that only legal content is distributed in the P2P system. Section VI focuses on the security of the system; authentication of end-users, controlling that only end-users having the right to watch the content can really do it, and new components that have been introduced in the system to detect and mitigate illegal or abnormal behaviors of peers (with a possible action to revoke them from the system). Section VII presents the evaluation of our prototype, performed in a real network environment, showing good results. Finally, section VIII presents some related work.. Section IX concludes this paper.

II. THE OVERALL SYSTEM ARCHITECTURE

The architecture of the system is composed of the functional blocks presented in Figure 1.

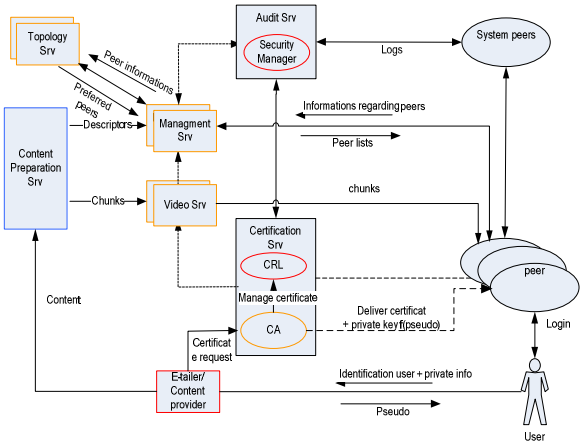

Figure 1. Architecture of the P2PIm@ges system

Firstly the user should download the P2P software client from an e-tailer (Content provider) who is directly in relation with the end consumer through his own portal. A certificate

is delivered to the user by the certification authority after identification. The certification authority validates the P2P software clients that are allowed to access the system. After choosing contents to be downloaded, a content download requests are directly send to the management server. The management server asks the topology server to provide guidance to figure out which peers are better to select from the network perspective. Afterward, the Management Server replies to P2P client with: 1) a description of the content (and possibly the quality) to be downloaded, allowing reconstructing content once downloaded every chunk, 2) a list of P2P software clients that constitute its new peer-set.
The content preparation server cuts contents into chunks. The list of the chunks related to a given content is built, signed and transmitted to the Management server. This list contains the associated content ID, the chunk IDs and the chunk hashes. Later, the chunks will be transmitted in the network independently; some chunks of the content can have different paths as others of the same video content.

Security is conducted by the audit server. All logs from entities of the system (System peers, Management Severs, all other servers) are sent to the audit server. The latter runs a security manager which analyzes logs and collected alarms to inform the management server about malicious peers, possibly to be excluded from peer lists or revoked.

System peer is a conventional peer controlled by the system. It is launched by the Security Manager from the Audit Server. System peer behaves like "standard" peer within the swarm, but it sends information about users in the swarm in which it is connected, to the Audit Server.

Next sections detail the components of the system.

III. NETWORK-AWARENESS

In order to take into account the network-awareness in the peer selection algorithm, cooperation between network operators and service provider is defined. This cooperation leads to two distinct entities: one managed by network operators, called the Topology Server (TS) and one managed by service providers, called the Management Server (MS) (see Figure 2).

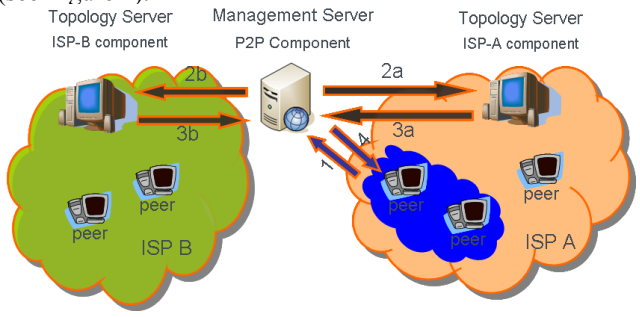

Figure 2. P2P Network-awareness Architecture

The *Topology Server (TS)* role is to maintain an accurate view of its network topology, with list of IP networks (or IP addresses) associated to different network groups (e.g., ADSL Points of presence or groups of cities) and weighted links for connecting them. Those weighted links are based on value of metrics such as bandwidth, delay, availability of links, cost of links, etc. The network topology information also encompasses information about peering relationships (i.e., if the operator has special agreements with some others operators). Based on this network topology, the TS will return back to the MS (step 3) a list of preferred peers to select, when requesting by the MS (step 2).

The *Management Server (MS)* is the P2P central entity which will provide the list of peers (step 4) when a peer joins the P2P network (step 1). Instead of sending peers selected randomly (from the list of peers sharing the contents) or based on application-level consideration, the MS will choose the peers that are preferred by the network operators, via a request/response protocol on a well-defined interface to the TS (step 2 and 3 in figure 2).

The interface between the MS and TS has two simple methods:

– connect (infohash, ip)

which allows the MS to inform the TS that a new peer (with this *"ip"* address) is joining the swarm identified by *"infohash"* and get back a list of preferred peers (to network operators point of view) who are in the same swarm.

– deconnect (infohash, ip)

which allows the MS to inform the TS that a peer (with this *"ip"* address) is leaving the swarm identified by *"infohash"*.

In this interface, information such as the content (identified by infohash value) or the end-user (identified by its IP address) are passed to the TS. Some people may object about it since it could be against user privacy but since the goal of our prototype is the delivery of a secure and legal content, having this information could help to control the legality of the content as well as the client. More information can be found in [17].

IV. P2P CLIENT SOFTWARE

Our system is for VoD delivery. Peer client software should then be adapted for VoD requirements; i.e., chunks download should be ordered to get chunks sequentially (or close to it) and not get chunks randomly. Instead of starting from crash, we adapted the Bittorrent Open Source code, developed in the Python Language, which is performing well, support large number of clients and is reliable.

The modifications we did consist of adaptations of the ranking of chunks index in BitTorrent buckets and the organization of the bucket of buckets, whose main interest is to manage the priority of chunks. The chunks are then organized in order to have sequential and not random chunk download. We also added a sliding window for the chunks download (this window can depend on the play-out time). All chunks that are within this window are preferably (and sometimes urgently) required are organized accordingly in the buckets. When all chunks are downloaded, the windows moves and then the following chunks are requested. In our implementation, we define the window as the size of 50 chunks.

In order to watch the video content, the peer software includes a wrapper to VLC player. Adaptation has been made in order to have player automatically started as soon as enough data has been downloaded by the peer. One main concern for our peer implementation was to reduce as much as possible this delay between the start of the download and the play-out of the video (see Section VII).

Finally, since our system is for secure and legal video content delivery, some added-value functions for security have been added: the watermarking function and the authentication module.

V. VIDEO WATERMARKING

Legal video distribution usually comes with content protection technologies, especially with Digital Rights Management (DRM). This kind of protection is very restrictive for the customer, since it only works with compatible systems, players and devices. Thus, we decided to use a system-agnostic content protection system based on digital watermarking.

Digital watermarking consists in embedding invisible information within content samples, i.e., pixels for image and video. This information must be robust: even if marked content is processed (e.g., video compression), we must be able to reliably read embedded data. Digital watermarking can deal with video, sound and pictures. Nevertheless, the algorithm we describe here can only handle images and video.

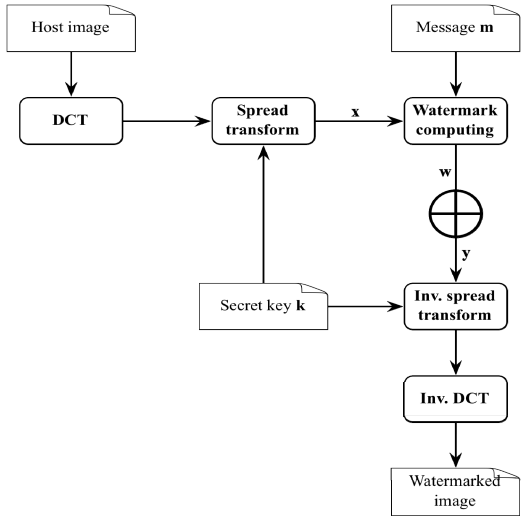

Figure 3. Watermarking algorithm.

Figure 3 depicts the defined watermarking algorithm. Given an image to be watermarked, its 8×8 Discrete Cosine Transform (DCT) is computed [18]. This transformed domain was chosen because is it is also used for MPEG video compression algorithms and allows the design of less visible marks. Selected DCT coefficients are projected on pseudo-random carriers thanks to a spread transform. Carriers are computed from a secret key. This leads to the host vector denoted **x**.

The watermark vector **w** is obtained from **x** and the message **m** to be embedded using a Turbo Trellis Coded Quantization (TTCQ) algorithm [19][20]: this coding technique provided state-of-the-art watermarking robustness. Finally, **w** is added to **x**, then inverse spread transform and inverse DCT are computed, leading to the watermarked image.

The watermark reading function uses same steps: image DCT then spread transform. TTCQ algorithm is then used to decode the message from $\mathbf{y} = \mathbf{x} + \mathbf{w}$.

In our system, each video frame is watermarked with 20-bits long message. The same message is used for all frames. Robustness benchmark done with SD movies shows that the message can be recovered even if watermarked movies are compressed with MPEG-4 at 200 Kb/s rate (very poor video quality).

Watermarking function is integrated within peers' video player (VLC player). After video decoding, images are watermarked prior to display. The embedded message corresponds to the peer's certificate (see Section VI). Thus, if a movie is illegally distributed, we can find out the client who leaked it. Each distributed movie copy is then unique thanks to watermarking. In order to detect illegal copies, we use system peers, deployed in the network (see Section VI).

While DRM protections are often seen as a major drawback of legal distribution platforms (because the technologies are not compatible with each other and sometimes prohibit legitimate uses of content), the watermarking technology we use is a transparent form of protection for video. The user can convert, copy and read watermarked movies on any player or system. But he knows that if he illegally distribute the content he bought, he may be accused. Of course, customers must be notified in advance of the operation of protection under the conditions of use of the platform.

VI. SECURITY OF THE P2P SYSTEM

The aim of the security in such project is (1) to authenticate users, in order to allow only authorized people to join the system, (2) to verify the identity of passing content to allow only authorized content to be exchange (see section V), (3) to analyze the flows exchanged between the various entities in the system (including peers) to detect malicious behaviors that may attack the system or alter its efficiency.

In order to identify and authenticate entities in the system, two mechanisms have been identified: Secure Sockets Layer (SSL) communication and the use of certificates. Each software client needs to acquire a certificate to be able to access the system. A unique ID (called ClientSoftwareID) is attributed to each software client. The ID is directly taken from the Registration Certification Authority (RCA). All certificates are generated off-line by the Certificate Authority (CA) and transmitted to the RCA. The CA keeps track of all ClientSoftwareID to be sure they remain unique. The ClientSoftwareID can be

revoked by the CA if the owner of that certificate behaves maliciously (e.g., id detected by system peers, see below).

In order to detect malicious behaviours, information about peers are collected by so-called System Peers and gathered and analysed by the Audit Server. A System peer is a peer controlled by the system. It is launched by the Security Manager from the Audit Server. In the swarm the system peer behaves like any standard peer but when communicating with others peers, it collects information about them and sends it to the Audit Server. When a Security manager activates a System Peers it gives them a swarm target. Several attacks can be detected directly by the System Peer or through the correlation of logs sends by several System Peers, such as free rider attacks, network flooding attacks, network filtering, incorrect use of the protocol… Section VII presents the attacks or misbehaviors we have implemented and tested. Statistical analysis is recovered by the Audit server that can decide, depending on the event, to take some actions (alerting peer, revoking it, etc.). Several techniques to restrict access to malicious peers are used according to the gravity of the behavior. The various entities that send logs to the Audit server are shown in Figure 4. The Host Intrusion Detection System (HIDS) alarms sent by authentication components (Registration Authority, Delivery Authority and Certification Authority) (RA, DA, CA) to the Audit Server are already encrypted (use of prelude manager tools). All information sent to the Audit Server are on an encrypted Syslog channel.

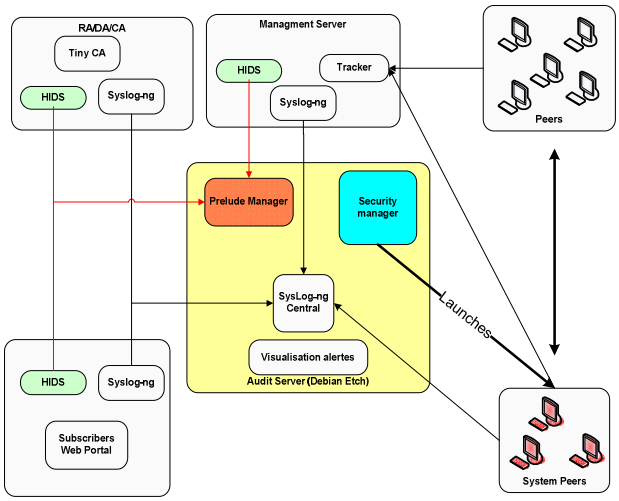

Figure 4. Security Detection System

VII. EVALUATION

The main objective of our evaluation was to identify the quality of the content that can be delivered to end-users enjoying the services in a *real-life configuration.* This mainly means end-users connected behind an Internet Box (e.g., Livebox) at a given throughput (e.g., 1 or 10 Mbit/s) using a real ISP connection. Several clients in different locations (8 in Lannion with 2 dedicated for the MS and the TS, 2 in Rennes, 6 in Nantes, with 3 dedicated for the System peers) were used in order to have a distributed network. The quality of the video has been evaluated in an objective way; the end-users themselves will note the quality of the video, according to the MOS scale. Picture 5 depicts this environment in the end-user side.

In the current implementation, the MS and the TS are written in the Python language. The network topology is a text file, which is updated according to network conditions changes. The interfaces between the MS and TS are sockets and data are formatted using JSON (JavaScript Object Notation) for transmission.

We made different tests for evaluating the quality of video as well as the efficiency of our system regarding security issues.

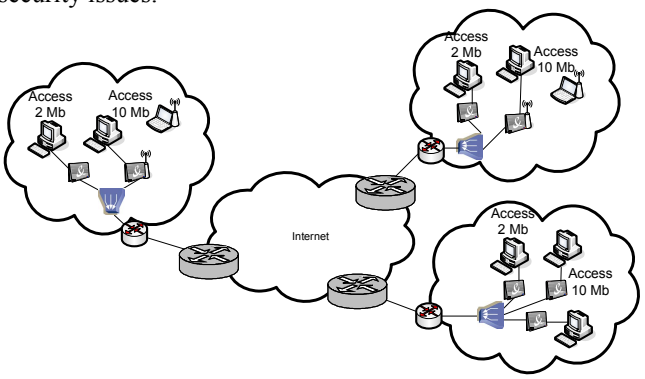

Figure 5. Evaluation testbed

For the video quality, we made tests with peers wanting the content (video is watched during downloading) in Lannion and Rennes using 1M and 10M links and with one peer offering the content (seeder) located in Lannion. Following table shows the results.

Client	Complete Download in	Video Player starts after
Lannion 10M	3'20"	15"
Lannion 1M	14'	1'15" (player blocked, buffer empty waiting data)
Rennes 10M	3'46"	22"

TABLE I. DOWNLOAD TIME & PLAYOUT

This test proved that the download of a video in a P2P fashion is feasible for users having a high bandwidth within a good time and that the end-users does not have to wait for a long time before the start of the video (time needed to fill in the buffer). After the start of the video, the video is smoothly played-out without any interruption or freeze. But users behing 1M links can not watch the video smoothly since the link is too slow (buffer empty, waiting for data).

For video quality, we made tests to evaluate the behaviour of our client on interrupted session. Whereas the interruption occurred because of end-users stop or because of a network failure, the download restarted where it was (not starting from the beginning).

For the tests highlighting the network-awareness of our solution, we used peers located in the three sites: Lannion, Rennes and Nantes, and having links between sites with different priority based on the cost of using the physical link: (1) higher priority level for link *Rennes-Nantes*, (2) medium priority for link *Lannion-Rennes*, (3) lower priority level for link *Lannion-Nantes*. It means that link *Lannion-Nantes* should be avoided as often as possible.

We have tested different scenarii such as:

a) one seeder was started in Nantes,

b) one client (lecher) was started in Lannion; Since no local seeder is available then the seeder in Nantes was selected,

c) one seeder as started in Lannion: the peer in Lannion must connect to seeder in Lannion and disconnect from seeder in Nantes as soon as possible,

d) one peer was started in Rennes: it must be connected to seeder Nantes (higher priority).

or others with the requirement to have at least several seeders for a good delivery of video, such as :

a) we started one seeder in Nantes and one in Lannion,

b) we started one client peer in Lannion: downloading from seeders in Nantes and Lannion,

c) we started one client peer in Rennes: downloading from seeders in Nantes and Lannion,

d) start a new Seeder in Lannion: Peer in Lannion must to connect to new seeder in Lannion, must disconnect from seeder in Nantes.

In all cases, the tests were successful and the disconnection/reconnection from/to seeders was done transparently to end-users, and not detectable since the video quality was good, without freeze nor blocking artifacts.

Those tests proved that the topology-awareness when selecting peers help in reducing network bandwidth and has also no impact in the end-user QoE. The change of one seeder to another one does not lead to freeze in the video. The apparition of one seeder in the region is automatically detected and the lecher peer connects to this new local seeder and disconnect from remote seeder.

Security tests were more functional tests, aiming at checking the efficiency of our implemented solutions to detect and mitigate illegal behaviour of peers. An interface was developed to conduct tests on the security platform. A Bash file is generated through this interface and run on the security platform. Logs of machines concerned by the test are displayed trough this interface. All tests were conducted with 8 machines (1 MS, 3 Peers, 3 Systems Peers). Audit Server and Security Manager were on the same machine. All the functional tests were successful. The tests we have done are:

* *Flooding of secure connections detection*: The System Peer relates this attack to Audit Server when it receives lots of secure connections in a small amount of time.

* *Network flooding detection*: The audit server (Security Manager) can detect a network flooding attack by correlating logs coming from System Peers. The System Peer can report a Free-Rider peer by analyzing its behaviour. The audit Server takes a decision concerning this peer if the same behaviour is reported by the other System Peers.

Figure 6 shows this flow chart[1].

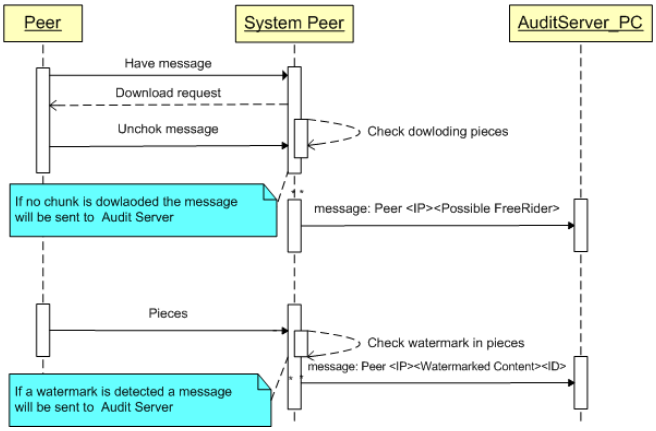

Figure 6. Network flooding detection flow chart

* *Detection of an incorrect use of P2PIm@ges protocol*: System Peer analyse all messages exchanged with peers. If the protocol used by the peer is incorrect, the system peer sends logs to the Audit Server with information (IP, CN, etc.) of the concerned peer. After that, the revocation of the certificate of peer is performed by the Audit Server.

* *Detection of compromised certificate/ private keys*: The misbehaviour is detected when an attacker tries periodically to connect to the MS with a bad or expired certificate (Certification authority not recognized). This lead to (1) connection refused by Tracker, (2) message displayed with IP of Attacker and (3) after several tries, attacker IP is blocked

* *SSL flooding attacks type "Denial of Server" (DoS)* on protected servers are detected when attacker tries periodically to the MS without certificate. The connection is refused ans the the peer IP is blocked after several tries.

For watermarking tests, we made tests aiming at evaluating the amount of time needed to detect a watermark from the beginning of the downloading process. System peers provide a watermark detection daemon, which checks video chunks during their download. If a watermark is detected, an alert is sent to security logger with the extracted ID and movie file name.

For this test, we limited the upload/download rate to 20 Kb/s for all peers. The following results were obtained, depending of the file size (3 files) and the number of system peers that were used (1 or 3 system peers).

.This test proved that our system can detect illegal content in the P2P network quite quickly. It is longer when several peers are used because correlation is done between information provided by system peers. A compromise is made between detection accuracy and detection time; in those results, we focused on very low false detection rates.

[1] It should be noted that such flowcharts exist for all tests but are not included in this paper for space limitations.

Movie file	Size (Mo)	Compression (Kb/s)	# of system peers	Time to detect watermark
Testa.avi	616.27	148	1	1 to 3 minutes
Testa.avi	616.27	148	3	5 minutes
Testb.avi	209.38	160	1	2 to 3 minutes
Testb.avi	209.38	160	3	7 minutes
Testc.avi	356.56	133	1	2 to 3 minutes
Testc.avi	356.56	133	3	6 minutes

TABLE II. WATERMARKING DECTECTION

VIII. RELATED WORK

In this section, we present some related work and highlight how our system differs.

Several research projects have lead to different systems such as P2VoD [2], GnuStreeam [3], ZigZag [4], Pals [5], Promise [6] but they mainly focus on the overlay level. They present solutions about the way to structure the overlay network: tree [4] or mesh [7] and the way data should be fetched, receiver-driven solution [5] or sender-driven solution [4]. Some also proposes to take into account the network, via application probes [3][6], but without relation with network operators. [8], [9] or [10] includes such network-awareness concept as currently under definition in IETF ALTO (Application Level Transport Optimisation) group [11] or adaptation but as others, they do not propose watermarking solution to control the contents, neither the system peers concept to monitor the network behavior.

There also exist deployed solutions, such as RayV [12], Peerialism [13], Peering Portal [14] but they do not implement functions for cooperation with network operator, neither watermarking options. Finally, well-known and used solutions [15] [16] such as PPLive, PPStream, TVants for Live content, do not take into consideration the network and do no take care about legality of contents.

IX. CONCLUSION

In this paper, a secure and legal network-aware P2P system has been presented. The main goals of this study was: (1) to reduce the network load in the network via a better organization of the P2P network (better peer selection algorithm) while ensuring a good Quality of Experience to end-users, (2) to offer a secure and legal content distribution system that could convince content providers to distribute their contents using a P2P system since solutions to protect them about security issues are provided in the system. The evaluation that has been performed both in lab and in a real-life configuration (end-users behind DSL boxes and interconnected via Internet) proved the quality of the prototype and the efficiency of the proposed system. This paper has proved that technologies exist and can be deployed for delivering video content in a secure way. The next step is to convince content providers as well as network operators to widely deploy such systems.

ACKNOWLEDGMENT

The work presented in this paper has been partially funded by the French P2Pima@ges project.

REFERENCES

[1] http://www.telegraph.co.uk/scienceandtechnology/3357003/Who-will-pay-for-the-new-web.html

[2] T.Do, KA. Hua, and M. Tantaoui, “P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment.”, in the Proc. of the IEEE ICC, Paris, France, 20-24 Juin 2004

[3] X. Jiang, Y. Dong, D. Xu, and B. Bhargava, "GnuStream: a P2P Media Streaming System Prototype" IEEE International Conference on Multimedia and Expo Baltimore, MD, July 2003.

[4] Duc A. Tran, Kien A. Hua, and Tai T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming.", In Proceedings of IEEE INFOCOM 2003, San Francisco, CA, USA, 30 Mars – 03 Avril 2003

[5] R. Rejaie, and A. Ortega, "PALS: Peer-to-Peer Adaptive Layered Streaming". in Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video, Monterey, California, June 2003

[6] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: Peer-to-Peer Media Streaming Using CollectCast.", In Proc. of ACM Multimedia 2003, pages 45--54, Berkeley, CA, Novembre 2003

[7] F. Pianese, J. Keller, and E. W. Biersack, “PULSE, a Flexible P2P Live Streaming System”, in Proc. of 9th IEEE Global Internet Symposium 2006, Barcelona, Spain, 28 & 29 April 2006

[8] S. Traverso, E. Leonardi, M. Mellia, and M. Meo, " Network Awareness in P2P-TV Applications", 15th Open European Summer School and IFIP TC6.6 Workshop on The Internet of the Future, Barcelona, Spain, 7-9 September 2009

[9] H. Xie, A. Krishnamurthy, A. Silberschatz, and Y.Richard Yang, "P4P: Explicit Communications for Cooperative Control Between P2P and NetworkProviders", http://www.dcia.info/ documents/ P4P_Overview.pdf

[10] M. Eberhard, L. Celetto, C. Timmerer, E. Quacchio, H. Hellwagner, and F. Rovati, “An Interoperable Streaming Framework for Scalable Video Coding based on MPEG-21”, Proceedings of the 5th IET Visual Information Engineering Conference Conference (VIE’08), Xi’an, China, July 2008

[11] http://www.ietf.org/html.charters/alto-charter.html

[12] http://www.rayv.com/cms/index.html

[13] http://www.peerialism.se/

[14] http://www.peeringportal.com/

[15] Y. Liu, X. Hei, C. Liang, and K. W. ROSS. “Insight into pplive: A measurement study of a largescale p2p iptv system”. 2006

[16] T. Silverston and O. Fourmaux, “P2P iptv measurement: A comparison study.”, http://www.arxiv.org/abs/cs.NI/0610133, 2006.

[17] B. Mathieu, P. Paris, "A Topology-Aware P2P Video Streaming System", GIIS 2009, Hammamet, Tunisia, 23-26 June 2009

[18] P. Duhamel and C. Guillemot., "Polynomial Transform Computation of the 2-D DCT", IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Albuquerque, USA, 1990.

[19] V. Chappelier, C. Guillemot and S. Marinkovic, "Turbo trellis coded quantization", in Int. Symp. on Turbo Codes, Brest, France, September 2003.

[20] G. Le Guelvouit. "Trellis-coded quantization for public-key watermarking", IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Philadelphia, USA, 18-23 March 2005.