Digital Fountains + P2P for Future IPTV Platforms: An Infrastructure Evaluation

G. Marfia, M. Roccetti, A. Cattaneo, A. Sentinelli, A. Vitali, L. Celetto, M. Gerla, *Fellow, IEEE*

***Abstract*—The opportunities laid by the joint use of Peer-to-Peer (P2P) streaming systems and advanced Digital Fountain (DF) codes in the context of Internet Protocol TeleVision (IPTV) platforms have set high expectations for their combined use. P2P streaming strategies, on one hand, have proven to be able to increase the bandwidth efficiency and the consequent scalability of IPTV platforms. DFs, on the other, have found a potential role in improving the bandwidth efficiency and the robustness to high peer churn rates of P2P streaming systems. However, given the complexity of both of such systems and the unexpected possible outputs that may result from their interaction, an effective assessment of their performances requires the design and setup of real world test-beds. In this article we describe the results deriving from a set of experiments performed on a P2P streaming test-bed that integrates DFs. The most prominent result of this experimentation campaign is that DFs do not provide notable improvements to a BitTorrent-based network of limited size, while as the network scales beyond a given size, it is possible to observe important advantages. This work, and, in particular, assessing when the use of DFs can be beneficial, can influence the devise of future IPTV service architecture which could be based on such technologies. We believe this represents a relevant contribution to understanding important aspects of the joint utilization of peer-to-peer multimedia streaming strategies and advanced digital fountain techniques.**

***Index Terms*—Coding techniques, digital fountains, peer-to-peer, streaming.**

I. INTRODUCTION

The recent successful and steady growth of the Internet Protocol TeleVision (IPTV) market, which analysts expect will keep its momentum going despite the recent economical breakdown, has triggered the research for innovative and efficient distribution strategies [1]-[3]. In fact, thanks to a trend where commercial IPTV channels distribute an increasingly wide range of content that spans from real-time streaming videos (e.g., soccer matches) to on-demand content (e.g., classic movies), about 27 million new IPTV contract services have been signed up in 2009. Considering such figure and recent researches that predict a 50% yearly worldwide market expansion for the next decade, the expected demand for multimedia content and, consequently, for larger amounts of bandwidth will steeply increase. In such scenario, the over provisioning of core and access network resources and the gradual adoption of multicast-capable routers may be too expensive and slow, respectively, to keep up with such exponential growth rate.

A possible way of guaranteeing the scalability of an IPTV distribution network, while confining the amount of investments and, contemporarily, achieving the desired results, is the integration of a Peer-to-Peer (P2P) infrastructure [4]-[12]. In particular, P2P distribution strategies have received great attention in the context of IPTV streaming platforms because of two particularly attractive properties: a) their software modules can be easily ported on Set-Top-Boxes (STB), and; b) they utilize the upload bandwidth of peers, resource that would otherwise probably be left unexploited and, therefore, unnecessarily wasted. Although promising, however, the opportunity of adopting P2P streaming strategies within commercial IPTV platforms is still under investigation. In fact, channels that rely on P2P streaming systems are usually affected by delays (i.e., end-to-end and startup delays) that are typically much higher than those we are all accustomed to when watching traditional analog TV sets, and are also affected by service degradations (e.g., losses of video frames) that can be caused by sudden changes in the P2P network topology (e.g., users that switch on or off their STBs).

Manuscript received October 12, 2010.

A. Cattaneo, A. Sentinelli, A. Vitali and L. Celetto are with STMicroelectronics, Agrate Brianza, Italy (email: alessandro.cattaneo@st.com, alexandro.sentinelli@st.com, andrea.vitali@st.com, luca.celetto@st.com).

G. Marfia (corresponding author) and M. Roccetti are with the Computer Science Department, University of Bologna, Bologna Italy (e-mail: marfia@cs.unibo.it, roccetti@cs.unibo.it).

M. Gerla is with the Computer Science Department, University of California, Los Angeles, USA (e-mail: gerla@cs.ucla.edu).

Digital Fountains (DFs) could be the missing piece of the puzzle as they can potentially combat the main shortcomings of P2P streaming strategies, alleviating, for example, the degradations that may occur as a consequence of high churn rates [13]. In brief, while in traditional P2P streaming networks packet losses directly turn into a loss of information, in DF-based P2P streaming networks the same events may be successfully dealt with combining other packets that have already been or will shortly thereafter be received.

As a result of the possible synergies that may exist between DFs and P2P streaming techniques, a plethora of approaches have been devised aiming at optimizing their combined usage. However, much of the work that has been so far carried out bases its conclusions on theoretical analysis and simulations. In this paper, instead, we will describe our experience in designing and building a DF + P2P streaming test-bed and show the advantages that, in reality, can be expected from their combined use. Our major finding, in particular, is that DFs can provide significant advantages to a P2P streaming architecture based on a mesh topology, but only as the network exceeds a given size. When mesh P2P streaming networks are of small size (below 100 nodes), in fact, DFs actually reduce the download speed of multimedia flows, as the amount of overhead introduced by their use weighs more than their beneficial effects. Such information, for example, is very useful when predicting the size beyond which an IPTV platform can successfully rely on a DF + P2P architecture, although it results tricky to obtain without a real test-bed evaluation.

The rest of this paper is organized as follows. In Section II we provide an illustrative example of how digital fountains can be useful alleviating the degradation effects that are due to packet losses. In Section III we describe the architecture of our test-bed infrastructure, while in Section IV we provide a comparison between a P2P architecture that employs DFs and one that does not. We finally conclude with Section V.

II. Background

Since P2P streaming techniques have received great attention in the past decade and their underlying mechanisms are supposedly well known to most audiences, we will here concentrate our efforts to describing how digital fountains work.

The main idea that lies at the basis of digital fountains is the introduction of redundant information in the packets that are sent over the network, so that packet losses and client failures may be more easily dealt with. This process, however, should not be confused with a simple procedure where the same packets are merely retransmitted to the same peers. With digital fountains, in fact, all packets are unique, although they enclose and convey information related to, possibly, the entire source of information (e.g., a multimedia stream). To explain this concept more effectively, we will rely on the following comparison.

Let us now consider a scenario where a node, which we will from now on call *K,* attempts to download from its peers, namely *A*, *B*, *C*, *D*, *E*, and *F,* a video stream composed of three pieces: *x*, *y* and *z*. Assume digital fountains are not employed and that *K* proceeds making the following requests: piece *x* to *A* and *B*, *y* to *C* and *D*, and *z* to *E* and *F* (leftmost part of Figure 1). Furthermore, letting us also assume that there is a 50% chance that a requested piece is lost, the following might occur. *K* receives the pieces sent by *A*, *B* and *C* (ending up with two copies of *x* and one of *y*), while misses the ones sent by *D*, *F* and *E* (being unable to retrieve a copy of *z*). Therefore, after this first round, *K* still needs to issue a request to receive *z*. This example exposes one drawback of the Automatic Repeat request (ARQ) paradigm: longer delays and greater overheads are induced by retransmission requests.

Now, we will describe what would have happened if a trivial digital fountain scheme such as a Forward Error Correction (FEC) mechanism were employed. Using FEC at all peers, the symbols that are transferred result from the XOR operation between the pieces that compose the stream.

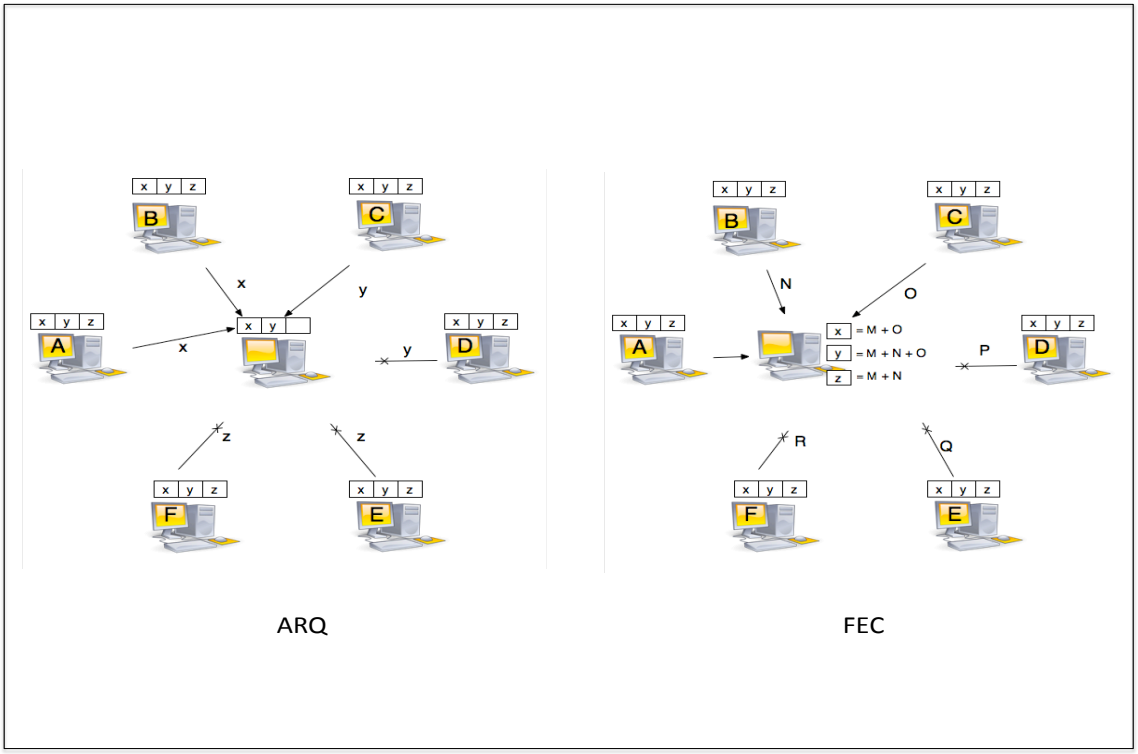

Fig.1. ARQ vs. FEC strategies.

Therefore, the following might have occurred (rightmost part of Figure 1): *K* receives *M* (*x* XOR *y*) from peer *A*, *N* (*x* XOR *y* XOR *z*) from *B* and *O* (*y* XOR *z*) from *C*. Alike the previous case, 50% of the requested pieces are lost, but now *K* can obtain the original video decoding the received pieces (i.e., *x* = (*N* XOR *O*), *y* = (*M* XOR *N* XOR *O*) and *z* = (*M* XOR *N*). Clearly, in this case, no further requests are necessary and, consequently, no further use of bandwidth resources. However, quantifying the potential of the combined use of digital fountains and P2P schemes, providing a thorough understanding of their capabilities (e.g., increased bandwidth use) and possible negative aspects (e.g., overhead and complexity), requires a test-bed evaluation, as we will see in the following Section.

III. Test-Bed

Aiming at quickly obtaining the implementation of a client that resulted compliant with BitTorrent, we first proceeded integrating a digital fountain scheme on top of a modified version of the mainline BitTorrent client [11]. After this step, in order to realize an assessment, we installed our client on the nodes of the PlanetLab infrastructure and performed a set of experiments that we will here shortly describe [14]. However, before moving on to the results of our experimentation campaign, we will summarize how our client works.

First of all, each client applies DF encoding on the available 16KB long blocks that compose the pieces of a multimedia stream (i.e., streams are composed of multiple pieces, which, in turn, are composed of multiple blocks). The particular DF instance that we implemented is Raptor coding, using the Python programming language [15]. A set of tests (500 runs) performed on our encoder and decoder modules have shown that on PlanetLab nodes, for a piece size of 128 blocks, the coding and decoding times never exceeds 500 and 400 milliseconds, respectively.

Second, at the application layer our client, say *A*, behaves as a BitTorrent client that is in end-game mode, thus always asking all of its known un-choked (i.e., peers that may answer requests) neighbors for random blocks (BitTorrent adopts this strategy only in the final phase of a download). Moreover, in order to increase the number of random blocks that propagate through the network,

A sends two requests, instead of one, per each block within a piece: a high priority one and a low priority one (a peer serves low priority requests only after all the high priority ones have been dealt with). In turn, when a peer, for example *B*, receives a request from *A* for a random block inside a piece, say *P*, it proceeds: i) randomly selecting a subset of blocks within *P*, ii) XORing them, and, iii) obtaining a random block that is readily sent. When *A* receives the block that was sent by *B*, it stores it in a buffer specifically allocated to piece *P* and attempts to obtain *P* using all of the random blocks that have been so far accumulated in the buffer. If the decoding process fails (e.g., it can be determined performing a hash check), *A* issues new requests to collect a number of random blocks that is sufficient to reconstruct *P*.

Finally, at the transport layer, instead of the TCP connections utilized in BitTorrent, our DF-based client utilizes unreliable UDP segments to transfer data between peers. Such decision follows from the general design choice of dealing with the loss of any data with DF techniques, rather than with ARQ ones.

IV. Results and Discussion

We performed a testing campaign using a single seed (i.e., one multimedia server) and 170 PlanetLab peers located all around the world. All nodes within the P2P streaming network, including the seed, were allowed a maximum upload speed of 1.6 Mb/s.

Our first test has been that of assessing the download completion time at all peers of a 30 MB video for a fixed piece size of 2 MB. In Figure 2 we plot the download completion time as a function of the peer index for four different P2P streaming client types: A) BitFountain (i.e., our DF-based BitTorrent client), B) a BitFountain client that employs TCP at the transport layer and where all the coding and decoding operations are pre-computed in advance, C) a BitFountain client where, again, all the coding and decoding operations are pre-computed, and, D) BitTorrent.

Interestingly, a comparison between the completion times of BitFountain (i.e., A) and BitFountain with pre-coding (i.e., C) indicates that about 30 seconds are wasted in the coding and decoding process. Moreover, as lower completion times typically correspond to better IPTV performance values, a first analysis of Figure 2 indicates that BitFountain clients with pre-coding achieve more or less the same performance of their BitTorrent counterparts (we are allowed to assume coding and decoding delays negligible, as the modules that perform these operations can be efficiently implemented in hardware). However, with a more attentive inspection, we also find that the BitFountain clients with pre-coding improve on the BitTorrent clients when the network size exceeds 100 peers. This fact can be readily explained as follows. A larger network means that more un-choked peers are available for downloading blocks. Nonetheless, when utilizing BitTorrent clients, only part of the peers owns the requested blocks and, contemporarily, is provided of sufficient upload bandwidth. This event, instead, rarely occurs when utilizing BitFountain clients, where there is a high chance that the nodes that can dispose of spare upload bandwidth also own random blocks that may be useful in the decoding process. Interestingly, we are only capable of capturing the beginning of a trend where the positive features of DFs void the negative effects of overhead in BitFountain, thus improving over the performance of BitTorrent.

For completeness, we also tested the effects of utilizing TCP at the transport layer of BitFountain (i.e., client B). As we could have expected, the interferences produced by the interactions between the ARQ mechanism of TCP and the Raptor code implemented within BitFountain provide poor results (a similar result concerning the interference of protocol layers may be found in [16]-[18]).

Finally, we decided to verify the effect of varying the piece size within BitFountain with pre-coding and BitTorrent while keeping the default block size (16 KB) fixed. Figure 3 shows that increasing the piece size, from a value of 64 KB to a value of 8 MB increases the completion time of both schemes, although BitFountain in this case suffers this test less than BitTorrent, as it is capable of slightly improving its performance. This result was largely expected for both schemes, as it is well known that

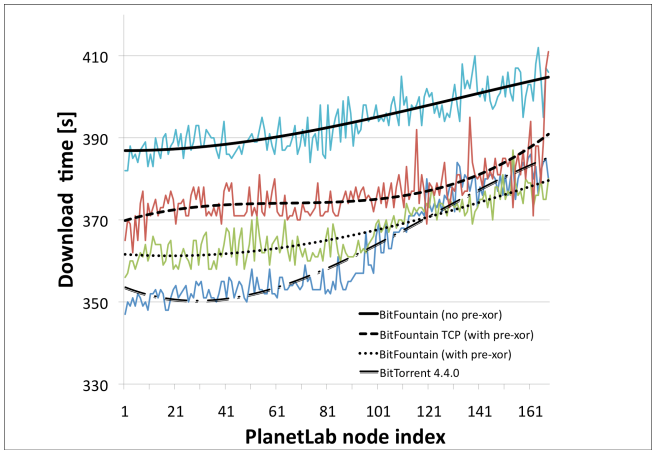

Fig.2. BitFountain vs. BitTorrent download times with and without pre-encoding and pre-decoding.

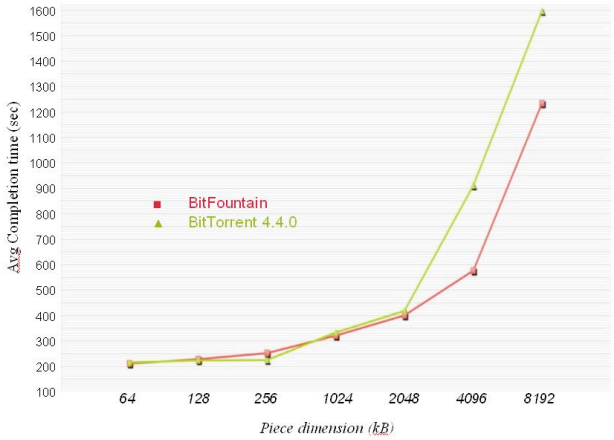

Fig. 3. BitFountain vs. BiTorrent download times as a function of the piece size.

smaller piece sizes correspond to more efficient download experiences in BitTorrent [11].

V. Conclusion

We here offered a set of experimental results performed on a real digital fountain P2P streaming test-bed. We believe this work provides a useful aid in determining how P2P and coding strategies can be put to good use for the implementation of next-generation IPTV platforms.

References

[1] C. E. Palazzi, N. Stievano, M. Roccetti and G. Marfia, "Ensuring Fair Coexistence of Multimedia Applications in a Wireless Home," in *Proc. 2nd IFIP/IEEE Wireless Days Conference*, IFIP/IEEE, Paris, December 2009, pp. 1-5.

[2] G. Marfia and M. Roccetti, "Dealing with Wireless Links in the Era of Bandwidth Demanding Wireless Home Entertainment," in *Proc. 6th IEEE International Workshop on Networking Issues in Multimedia Entertainment,* IEEE, Singapore, July 2010, pp. 1376-1381.

[3] C. E. Palazzi, A. Bujari, E. Cervi, "P2P File Sharing on Mobile Phones: Design and Implementation of a Prototype," in *Proc. of the 2nd IEEE International Conference on Computer Science and Information Technology*, IEEE, Beijing, 2009, pp. 136-140.

[4] P. Baccichet, T. Schierl, T. Wiegand and B. Girod, "Low-delay peer-to-peer streaming using scalable video coding," in *Proc. 16th Packet Video Workshop*, Lausanne, 2007, pp.173–181.

[5] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: high-bandwidth multicast in cooperative environments," in *Proc. 19th ACM Symposium on Operating Systems Principles*, Bolton Landing, 2003, pp. 298–313.

[6] C. Gkantsidis, P.R. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, Miami, 2005, pp. 2235–2245.

[7] W. Mea, L. Baochun, "R2: random push with random network coding in live peer-to-peer streaming," *IEEE Journal on Selected Areas in Communications,* vol. 25, no. 9, pp. 1655–1666, December 2007.

[8] D. Kostic, A. Rodriguez, J. Albrecht and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *Proc. 19th SIGOPS Oper. Syst. Rev.*, Bolton Landing, 2003, pp. 282–297.

[9] W. Chuan, L. Baochun, "rStream: resilient and optimal peer-to-peer streaming with rateless codes," *IEEE Trans. Parallel and Distributed Systems,* vol.19, no.1, pp.77–92, Jan. 2008.

[10] N. Thomos, P. Frossard, "Collaborative video streaming with raptor network coding," in *Proc. 2010 International Conference on Multimedia and Expo,* Hannover, 2008, pp. 497–500.

[11] B. Cohen, "Incentives build robustness into bittorrent," in *Proc. 1st Economics of Peer-to-Peer Systems Workshop*, Berkeley, 2003.

[12] A. Sentinelli, G. Marfia, S. Tewari, M. Gerla, L. Kleinrock, "Will IPTV ride the peer-to-peer stream?," *IEEE Communications Magazine,* vol. 45, no. 6, pp. 86-92, June 2007.

[13] M. Mitzenmacher, "Digital fountains: a survey and look forward," *Information Theory Workshop, 2004. IEEE* , vol., no., pp. 271- 276, 24-29 Oct. 2004.

[14] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, Jul. 2003.

[15] A. Shokrollahi, "Raptor codes," *IEEE Trans. on Information Theory*, vol. 52, no. 6, pp. 2551-2567, June 2006.

[16] V. Ghini, G. Pau, M. Roccetti, P. Salomoni, M. Gerla, "Smart download on the go: a wireless Internet application for music distribution over heterogeneous networks," in *Proc. IEEE International Conference on Communications*, Paris, 2004, pp. 73- 79.

[17] M. Wang and B. Li, "Lava: A reality check of network coding in peer- to-peer live streaming," in *Proc. 26th IEEE International Conference on Computer Communications*, Anchorage, 2007, pp. 1082-1090.

[18] M. Wang and B. Li, "Network Coding in Live Peer-to-Peer Streaming", *IEEE Trans. On Multimedia*, vol. 9, no. 8, pp. 1554-1567, 2007.