Broadcasting of H.264/SVC video over BitTorrent-like networks

J.P. García Ortiz jportiz@ace.ual.es
J.M. Dana dana@ace.ual.es
V. González Ruiz vruiz@ual.es
I. García igarcia@ual.es

Computers Architecture and Electronics Dept.

University of Almería, Spain

Abstract

BitTorrent protocol-based networks are the most popular peer-to-peer (P2P) configurations for data sharing in the Internet. One of the keys for its success is the prioritization of the less popular parts of the files so peers send first those chunks owned by the smallest amount of clients. The same algorithm that maximizes the bit-rate also prevents the low-delay transmission of multimedia contents because data streams are not sequentially sent. An alternative approach -still compatible with the standard BitTorrent protocol- would be the application of the same selection policy but applied to a sliding window which is moved from the beginning to the end of the stream. Unfortunately, this solution imposes to peers severe temporal restrictions for retrieving data from the rest of peers, causing significant variations of the downstream bit-rate. The developed proposal addresses this drawback transmitting scalable video streams instead of non-scalable streams. In order to determine the efficiency of our approach, a set of experiments have been carried out. Results reveal that a SVC-based (Scalable Video Coding) approach outperforms the standard non scalable (AVC) under certain circunstances. Specifically, when high encoding bit-rates are selected and streams are truncated by a drop of the transmission bit-rate.

1 Introduction

This work[1] studies the capacity of BitTorrent networks to broadcast video streams. The main reason for selecting this kind of overlay network is that, nowadays, BitTorrent networks are the most used peer-to-peer applications for data sharing files in the Internet. Usually, files are too large to be transmitted as a whole and the BitTorrent protocol splits them into data blocks. Blocks are transmitted in an order that depends on its availability for the set of peers (swarm) that at least shares a part of the file. The selection algorithm selects to be transmitted first those blocks that are less popular in the peer neighborhood because this minimizes the overall transmission time of the file. However, this rarest-first selection policy also prevents the low-delay broadcasting of video because videos must be played (and therefore, transmitted) using a sequential ordering. A solution for this problem, fully compatible with BitTorrent, is the use of the rarest-first selection algorithm applied only to a small number of consecutive blocks. These blocks are contained into a temporal sliding window that scrolls from the beginning of the stream to its end, every time a block is played. Unfortunately, this approach decreases the throughput of the BitTorrent-like network because, depending on the number of blocks in the window (number that should be small enough to guarantee a low-delay transmission), peers have less time to retrieve sequentially the content of each of them. This problem, added to the fact that the peers can have Internet links with different "speeds", makes impossible to determine the reception bit-rate during the playback time. In this context, our proposal researches the advantages of using the H.264/SVC standard. This video codec can generate quality scalable streams that can be truncated without causing playback interruptions, allowing to peers to display a video whose quality will depend on the amount of data re-

[1]This work has been funded by grants from the Spanish Ministry of Science and Innovation (TIN2008-01117 and TIN2009-05737-E) and Junta de Andalucía (P08-TIC-3518), in part financed by the European Regional Development Fund (ERDF).

ceived. The main encoding parameters such as the number of layers, the bit-rate of these layers and the GOP (Group Of Pictures) size are analyzed in the experiments. Results demonstrate that even using a small number of layers, continuous playback can be achieved independently of the reception bit-rate.

2 The BitTorrent protocol

BitTorrent [2] is a P2P communication protocol designed for the distribution of large amounts of data on the Internet. This data must be stored in a static-content file that an original distributor (also called *seed*) transmits to other peers of the BitTorrent network. Using this model, as soon as each peer has a part of the file, it supplies pieces of data to other peers, reducing the band-width cost of the seed's link. It also decreases the propagation time of the data thanks to the possibility of multiple concurrent transmissions among the peers that build the overlay network.

Each file in BitTorrent is divided into contiguous segments of equal length called *chunks* that are managed independently. Chunks are usually transmitted in smaller segments called *pieces*. Chunks usually have a size of 128 KB or 250 KB, whilst 16 KB is the common size for pieces [8].

When a peer requires to download a file, it needs firstly to find the associated *torrent* file (which usually has the `.torrent` extension), that contains, among other information, the *tracker* IP address and the hash code of all the chunks. These codes allow to validate each chunk (in order to detect transmission errors) when it is retrieved by a peer. The tracker is the only centralized component in BitTorrent. Its main function is to administrate the different peers that are downloading the maintained file. The tracker is not forced to serve this file, but only to manage the connected peers.

Once the appropriate torrent file is retrieved, the peer connects to the tracker to be included into the *swarm*. A swarm is the set of peers that are downloading the same file. The tracker sends to the peer a random set of swarm peers to communicate with them. Within a specific swarm, there are two different types of peers: *leechers* and *seeders*. Leechers have only some or none of the chunks of the file and are communicating to retrieve the rest, while seeders have the complete file already downloaded but stay connected to help the previous ones.

Each leecher requests chunks from all the peers it is connected to, following the *rarest-first* selection policy. With this chunk selection mechanism each peer selects to download first the chunks that are less replicated among their neighbors, allowing a dissemination of the file chunks along the whole swarm. In order to avoid *free-riding* [9], BitTorrent uses a *tit-for-tat* scheme, where each peer chooses to upload to another peer as long as it takes something in return.

3 Transmission of video over BitTorrent networks

The main obstacle for adapting BitTorrent to streaming of multimedia content is the rarest-first policy. Under this policy, chunks are not retrieved by a peer in a sequential order, necessary for this kind of content, but in a special order that ensures a good scattering of the data along the swarm. Removing completely this policy means the elimination of one of the most powerful features of BitTorrent, and hence to decrease significantly its performance. A certain balance between the rarest-first order and the sequential one must be found in order to make feasible the multimedia streaming as well as to maintain a good data distribution.

C. Dana et al. proposed BASS [3] that is one of the first solutions for multimedia content, specifically for video-on-demand, using BitTorrent as P2P base system. Authors reduce the limitation of the rarest-first paradigm introducing a little modification and an external server. Peers can not download any data prior to the current playback point, and can download data directly from the external server in sequential order. Despite the promising results exposed by the authors, this solution includes a non-scalable external server, that reduces its applicability.

BiToS [12] proposed by A. Vlavianos et al. in order to enhance BitTorrent for supporting streaming applications. They proposed to group chunks into two different sets: the high priority set, with chunks near the playback point, and the set with the remaining chunks. Therefore a peer chooses with a certain probability ρ to download a chunk of either a set or another. Within a set, peers choose the chunk following the rarest-first rule. The main drawback of

this approach is the difficulty of choosing appropriate (optimal) values for ρ and for the length of the high priority set.

Among the most relevant proposals, only LiveBT [6], of Jianming Lv et al., changes entirely the rarest-first policy by another different one. In this proposal peers would download data following the *most-wanted-block-download* strategy. Therefore a peer always chooses to download the chunks that are most wanted (are nearest the playback point) by its neighbor peers. This new chunk selection method is rather promising according to the showed results, but it is not as much evaluated in real scenarios as the rarest-first one.

Finally, the approach made by Purvi Shah and Jehan-Franvois Paris [11] is maybe one of the most interesting ones. Authors introduce only little modifications to the original BitTorrent paradigm. The rarest-first policy is maintained, but applied within a sliding window that is moved according to the playback point and the received data. The neighbor selection algorithm is also slightly modified in order to improve the throughput. Neither of these modifications prevent a fully compatibility with any existing BitTorrent-like system.

Although these proposals offer a solution acceptable for general multimedia data none of them are oriented to scalable video.

4 The H.264/SVC video codec

H.264/SVC [10] is the scalable extension of the H.264/AVC video coding standard [13]. SVC is an scalable video coding framework that can produce quality, spatial and temporal scalable streams. A stream is composed of a collection of consecutive sub-streams or *layers*, where each layer increases the temporal resolution, the spatial resolution or the quality of the decoded video. One of these layers is called the *base layer* and the rest are named *enhancement layers*. A base layer can be decoded by both an AVC and a SVC decoder, but enhancement layers can be only handled by SVC ones.

SVC exploits temporal redundancy using MCTF (Motion Compensated Temporal Filtering) [7]. Figure 1 shows an example of the use of MCTF on a GOP of 8 images, producing 4 temporal resolution levels. The number of spatial resolution levels is 2 and there is also 2 quality layers for each combination of temporal-spatial resolution. Therefore, the number of layers is 16.

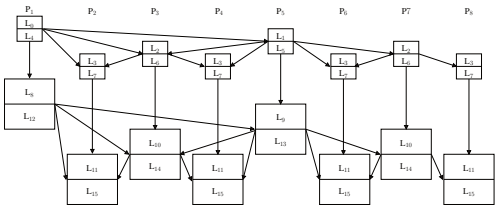

Figure 1: An example of the image dependencies in H.264/SVC with 4 temporal resolution levels (GOP size of 8), 2 spatial resolution levels and 2 quality levels. P_i denotes the i-th picture and L_i the i-th layer. Oblique arrowed lines represent temporal dependencies between images, vertical arrowed lines shows spatial dependencies and horizontal lines in the middle of a picture indicate a dependency between two quality levels.

Although layers can be decoded independently, there is a logical dependency between them that must be respected in order to obtain a valid reconstruction of the video. For example, L_0 (the base layer) does not depend on any other layers, but L_2 depends on L_1 (which also depends on L_0). All these dependencies between layers impose that in most of cases layers should be sent in order if a truncation of the stream could happen.

5 P2P low-delay transmission of scalable video

The proposal presented in this paper is based on the approach described in [11]. In this section only the main modifications done to this proposal in order to benefit from the video scalability are presented. As mentioned in Section 1, although the H.264/SVC video codec offers three kinds of scalability, that is, temporal, spatial (resolution) and quality (SNR), this work focused only on the SNR scalability, which is the most flexible way of scalability. Our solution can also be applied using all scalability alternatives, but it would require a further study, maybe as a future work.

In our video streaming system, each GOP is compressed independently and each GOP bit-stream is stored within a chunk in order to avoid data dependencies between GOPs at decoding time. Since the

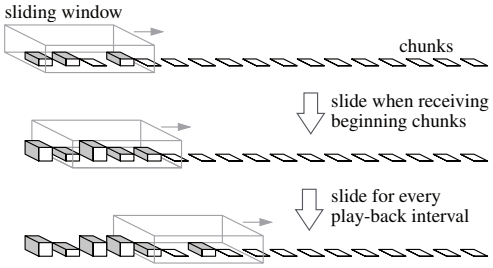

Figure 2: Example of the sliding window mechanism.

chunk size is usually equal to 128 or 256 KB, GOP bit-streams have been generated considering these limits.

Fig. 2 shows how the sliding window mechanism of P. Shah and J-F. Paris works [11]. As aforementioned, the movement of window is caused by a complete reception of the first chunk or by a playback interval. The window size selection issue is also tackled by the authors. They verified that a near optimal value for the window size is given by $s_w = d \cdot b/s_c$, where s_c is the chunk size (bytes), d is the playback delay (seconds) and b is the original video consumption rate (bytes/seconds).

Under the original approach each peer applies the rarest-first policy to select a chunk to download only within the window. This behavior has been modified a little bit, allowing peers to select chunks outside the window only if, when they are *unchoked* (are allowed for requesting) there is no chunk inside it to download. When a peer is unchoked by another peer, it can not waste this opportunity to retrieve whatever available data and contribute to the swarm scattering.

When streaming real time video, this modification does not affect at all, but it may become quite relevant when streaming video files. Since the data of the sliding window has a higher priority than the rest, any transmission of a chunk outside the window is replaced by another one for a chunk that were inside it as soon as possible. This is made although the first chunk reception were not complete.

The BitTorrent protocol imposes that a peer informs its neighbors about the existence of a new available chunk only when this is completely retrieved and checked with the hash codes of the Torrent file. In the context of scalable video streaming this is an unnecessary limitation. Certain ratio of errors is allowed in video transmissions so the content of a chunk might be published before being checked. Furthermore, thanks to the quality scalability, the content of a chunk can be exploited although it were not complete.

We add a new protocol message with the aim to avoid these constraints. This message would be used only between those peers that implemented this solution, hereinafter called *videoers* to differentiate them from leechers and seeders. Thus videoers must recognize each others when a connection is established between them. We have used for this task the reserved 8 bytes (originally included to allow possible protocol extensions) of the BitTorrent handshake message. Videoers would include a specific fix value that would be ignored by leechers and seeders, but would allow videoers to distinguish each other.

The rarest-first policy is slightly adapted to accommodate this new message and its philosophy. The weight of each chunk is not given by its rareness in terms of number of complete chunks, but it is instead given in terms of total number of pieces. Therefore, within the sliding window, a videoer peer ρ would choose to download a new piece of the chunk c that had the minimum $W(\rho, c)$ weight value, according to Equation 1. $N(\rho)$ equals the set of neighbor peers of the peer ρ, and $P(\rho, c)$ returns the number of pieces that the peer ρ has of the chunk c.

$$W(\rho, c) = \sum_{i}^{i \in N(\rho)} P(i, c) \qquad (1)$$

This weighting function is exactly the same as the original BitTorrent one when a videoer has only normal leechers as neighbors.

The randomized tit-for-tat policy also proposed in [11] is used in this approach as well. Hence each peer always selects random peers to unchoke at the beginning of every playback. As the authors analyze in [11], this policy gives more free tries to a larger number of peers in the swarm, achieving a better success ratio and network throughput.

Layer	Resolution	Pic. rate	Av. Bit-rate	Quality
1	704x576	0.9375	35.70	0
2	704x576	1.8750	60.90	0
3	704x576	3.7500	96.30	0
4	704x576	7.5000	142.10	0
5	704x576	15.0000	195.90	0
6	704x576	30.0000	264.80	0
7	704x576	0.9375	36.50	1
8	704x576	1.8750	92.10	1
9	704x576	3.7500	186.70	1
10	704x576	7.5000	316.70	1
11	704x576	15.0000	470.70	1
12	704x576	30.0000	681.70	1

Figure 3: A description of the layers generated by SVC encodes the "crew" video sequence with one spatial resolution level, two quality levels and six temporal resolution levels. The average bit-rate is expressed in Kbps.

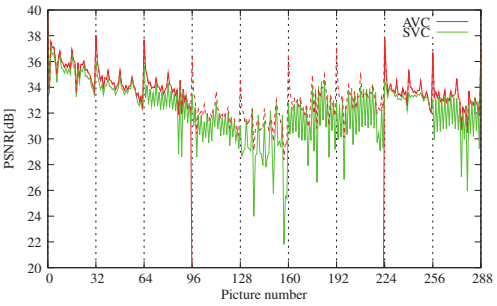

Figure 4: Quality given by AVC and SVC codec using the sample video "crew" when 6 temporal levels were used (GOP of 32 pictures).

6 Evaluation

A video broadcasting on a peer-to-peer network that uses the protocol proposed in Section 5 has been simulated using the NS-2 [1].The developed NS-2 code has been based on the code written by Kolja Eger et al. for the analysis they carried out of BitTorrent [4].

A flash crowd scenario with 100 peers, with only one initial seeder, has been simulated. Each peer has an upload band-width of 1 Mbps and a download band-width of 8 Mbps. All the peers are connected to each other by means of a link with a random delay according to a uniform distribution from 1 ms. to 50 ms. The chunk size used was set to 128 KB, but a piece size of 16 KB. The used sliding window has a length of 10 chunks.

With the transmission bit-rates produced by the network simulator two different experiments have been carried out. The first experiment shows the effects of the stream truncation when a non scalable stream (AVC) have been transmitted and the second with a scalable stream (SVC). In both cases the SVC Reference Software has been used [5].

The test video sequence used for the experiments has been "crew" which is composed of 300 images of resolution 704x576 that must be displayed with a frequency of 30 Hz. GOPs of 32 and 16 pictures have been tested, introducing an encoding delay of 1.06 and 0.53 seconds, and a network delay of 10.6 and 5.3 s., respectively, that can be acceptable for most real-time broadcasting scenarios. For both, AVC and SVC, the scheme used to remove the temporal redundancy has been MCTF. In the case of AVC only one layer was generated with all the temporal levels (5 when the GOP is 16 pictures long and 6 when it has 32 pictures). For SVC, 2 quality levels were used, selecting an average bit-rate of 300 Kbps/quality-level. Therefore, the AVC version of the video was encoded at 600 Kbps and the SVC version was encoded at 300+300 Kbps. These bit-rates were selected after calculating the average and minimum bit-rates available for the peers of the network. Finally, only one spatial level was selected. This produced a total of 10 layers for the case of 16 pictures/GOP (5 temporal levels $\times$ 2 quality level) and 12 (6 temporal levels $\times$ 2 quality levels) for the case of 32 pictures/GOP.

Figure 3 shows an example of layer order. As it can be seen for this video, in average, a truncation of the stream between 681.70 and 470.70 Kbps decreases the number of decoded layer in 1. This means that, although the reconstructed sequence has 30 pictures/second because the layer 6 is decoded, odd pictures have better quality that the even ones. This effect can be observed in Figure 4 for the GOPs 3, 4, 5, 6, 7 and 9. It can be seen also that the AVC is slightly better that SVC in GOPs 1, 2, and 8, although this difference is larger for GOPs 3 and 9, where the SVC stream was truncated and the AVC stream was not (the SVC bit-rate was slightly superior to the AVC bit-rate in these GOPs). On the other hand, AVC could not reconstruct GOPs 4, 5, 6 and 7. Similar results were obtained for a GOP of 16 pictures.

7 Conclusions

The transmission of non-scalable (AVC) and scalable video (SVC) over a P2P network that uses a BitTorrent-compatible protocol has been analyzed. Results demonstrate that when typical variations of the transmission rate happen, SVC can reconstruct at the receivers a video without interruptions while AVC can not. The loss of quality of SVC respect to AVC is, in general, small in those parts of the video in which the AVC stream in not truncated. As a major remark, our experiments reveals that SVC can improve significantly the quality of the video broadcasting on P2P overlay networks when large variations of the receive bit-rate have happened.

References

[1] The Network Simulator - NS-2. *http://www.isi.edu/nsnam/ns*.

[2] Bram Cohen. Incentives Build Robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, pages 229–232, June 2003.

[3] Chris Dana, Danjue Li, David Harrison, and Chen-Nee Chuah. BASS: BitTorrent Assisted Streaming System for Video-on-Demand. In *IEEE Workshop on Multimedia Signal Processing*, pages 1–4, October 2005.

[4] Kolja Eger, Tobias Hosfeld, Andreas Binzenhofer, and Gerald Kunzmann. Packet and flow level simulations of bittorrent-like p2p networks. *Multiagent Grid Systems*, 5(2):217–232, 2009.

[5] Image Communication Group (Fraunhofer HHI). SVC Reference Software (JSVM software). *http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm*.

[6] Jianming Lv, Xueqi Cheng, Qing Jiang, Jing Ye, Tieying Zhang, Iming Lin, and Lei Wang. LiveBT: Providing Video-on-Demand Streaming Service over BitTorrent Systems. In *Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 501–508, December 2007.

[7] J.-R. Ohm. Three-dimensional subband coding with motion compensation. *IEEE Transactions on Image Processing*, 3:559–571, 1994.

[8] Vivek Rai, Swaminathan Sivasubramanian, Sandjai Bhulai, Pawel Garbacki, and Maarten van Steen. A Multiphased Approach for Modeling and Analysis of the BitTorrent Protocol. In *International Conference on Distributed Computing Systems*, pages 10–19, June 2007.

[9] L. Ramaswamy and Ling Liu. Free riding: a new challenge to peer-to-peer file sharing systems. pages 1–10, January 2003.

[10] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the h.264/avc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, September 2007.

[11] Purvi Shah and Jehan-François Pâris. Peer-to-Peer Mutimedia Streaming Using BitTorrent. In *IEEE International Performance, Computing, and Communications*, pages 340–347, April 2007.

[12] Aggelos Vlavianos, Marios Iliofotou, and Michalis Faloutsos. BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In *IEEE International Conference on Computer Communications*, pages 1–6, April 2006.

[13] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.