INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Scalable Video Streaming over ALC (SVSoA): a Solution for the Large Scale Multicast Distribution of Videos

Christoph NEUMANN — Vincent ROCA

INRIA Rhône-Alpes, Planète project, France
christoph.neumann|vincent.roca@inrialpes.fr
http://www.inrialpes.fr/planete/

N° 4769

March 13th, 2003

THÈME 1

Rapport de recherche

ISSN 0249-6399

Scalable Video Streaming over ALC (SVSoA): a Solution for the Large Scale Multicast Distribution of Videos

Christoph NEUMANN , Vincent ROCA

INRIA Rhône-Alpes, Planète project, France
christoph.neumann|vincent.roca@inrialpes.fr
http://www.inrialpes.fr/planete/

Thème 1 — Réseaux et systèmes
Projet Planete

Rapport de recherche n° ???? — March 13th, 2003 — 28 pages

Abstract: This paper introduces a novel approach, SVSoA, for the streaming of hierarchically encoded videos using multicast IP, which fully benefits from the Asynchronous Layered Coding (ALC) reliable multicast protocol. The main assets of this approach are: (1) it is massively scalable, (2) it is naturally TCP friendly, (3) it is immediately deployable and does not rely on any QoS service in the network, (4) it supports clients heterogeneity and ensures a minimum video quality to each of them since the enhancement layers are only received in a second step, after the base video layer, (5) it is up to a certain point immune to long bursts of packet losses, and finally (6) it is compatible with any video hierarchical encoding scheme (temporal, spatial or qualitative, with a fine or rough granularity). Many of these features result from the intelligent use of ALC as the underlying transport protocol. This solution is well suited to the large scale distribution of videos or television programs over the Internet. Yet it is not suitable for interactive applications like video-conferencing because of the playing delay it induces.
This paper details the key parameters of the approach and the associated trade-offs. Experiments carried out with a full featured implementation of both ALC and our streaming solution, using a spatially encoded MPEG-4 video, confirm its benefits, especially in congested environments.

Key-words: Video streaming, Multicast, Asynchronous Layered Coding (ALC), Reliable Multicast

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Scalable Video Streaming over ALC (SVSoA): une Solution pour la Distribution Multipoint de Vidéo à grande Echelle

Résumé : Ce travail introduit une nouvelle approche de transmission streamée de vidéo hiérarchiquement encodée utilisant IP-Multicast et qui tire profit du protocole de transmission multicast fiable "Asynchronous Layered Coding" (ALC). Les principaux avantages de cette approche sont : (1) elle est massivement scalable en terme de nombre d'utilisateurs, (2) elle est naturellement équitable avec TCP (TCP-friendly), (3) elle est immédiatement déployable et ne repose sur aucun mécanisme de gestion de QoS au sein du réseau, (4) elle supporte l'hétérogénéité des récepteurs et assure à chacun d'eux la réception d'une qualité vidéo minimale puisque la(les) couche(s) d'amélioration ne sont reçue(s) que dans un deuxième temps, après la couche vidéo de base, (5) elle permet un certain niveau d'immunité vis-à-vis des longues rafales de pertes de paquets, qui ne conduisent pas nécessairement à un changement brusque de la qualité vidéo coté récepteur, et (6) elle est compatible avec n'importe quel type d'encodage vidéo hiérarchique (temporel, spatial ou qualitatif, présentant ou non une granularité fine). Plusieurs de ces caractéristiques dérivent de l'utilisation d'ALC en tant que protocole de niveau transport. Cette solution est bien adaptée à la distribution à large échelle de vidéos ou de programmes TV sur l'Internet. Cependant elle n'est pas adaptée à des applications interactives telles que la vidéo-conférence en raison des délais importants qu'elle induit.
Ce papier détaille les paramètres clefs de l'approche et les compromis associés. Des expérimentations menées avec une implémentation complète d'ALC et de notre solution de streaming, utilisant une vidéo MPEG-4 avec encodage spatial hiérarchique, confirment les bénéfices de l'approche, tout particulièrement dans le cas d'un environnement réseau congestionné.

Mots-clés : Streaming Video, Multicast, Asynchronous Layered Coding (ALC), Multicast fiable

1 Introduction and Related Works

This work deals with the streaming of videos, that are either dynamically produced or pre-recorded, to clients who receive and play information on-the-fly. It targets a massively scalable distribution, with potentially several millions of concurrent clients, and multicast-IP is therefore unavoidable. Because of the ubiquity of IP, this multicast routing infrastructure can take advantage of many different technologies on both the core network and the access network (satellites, terrestrial links, cable, DSL, etc.). In this work we merely assume the availability of a multicast routing service without making any assumption on its nature (source specific versus any source), nor on the nature of the underlying physical technology. Because of these assumptions, the client set is generally highly heterogeneous. This heterogeneity must therefore be considered to enable each client to receive the video stream that best fits with its networking and processing capabilities.

1.1 Video Scalability

In this context, the advent of recent video codecs like MPEG-2, H.263+, MPEG-4, or H26L has largely improved the streaming possibilities, making it possible to optimize the video quality *over a given bit rate range* instead of *at a given bit rate*. The present work focuses on (but is not restricted to) MPEG-4. The video scalability feature of MPEG-4 [18], also known as hierarchical video coding (both names will be used indifferently in this paper), refers to the possibility to see a video at several spatio-temporal resolutions by parsing appropriate portions of the bit-stream. In MPEG-2 and 4, three scalability techniques exist:

- *Temporal scalability:* this technique codes a video sequence into several layers at the same spatial resolution but different frame rates, where the enhancement layers provide the frames missing in the base layer to provide a higher frame rate. Various possibilities exist, depending on what frames ("Predicted" P frames and "Bidirectional" B frames) are affected to the enhancement layers and their relationships. When several enhancement layers are defined to provide a finer granularity, great care must be taken on the interdependencies between the P and B frames.
- *Spatial scalability:* this technique codes a video sequence into two layers at the same frame rate, but at different spatial resolutions.
- *Qualitative (or SNR, Signal-to-Noise Ratio) scalability:* this technique codes a video sequence into two layers at the same rate and spatial resolution, but using different quantization accuracy (i.e. number of DCT coefficients).

In all cases a partial reception of the enhancement layer will provide very little benefit (see [9] for a detailed explanation). This is why the Fine Granularity Scalability (FGS) [9] and Progressive FGS (PFGS) [23] scalability schemes are actively studied for MPEG-4. Although the FGS coding technique also codes a video sequence into two layers, a partial reception of the enhancement layer bit-stream provides a partial enhancement proportional to the number of bits decoded for each frame. The enhancement layer can therefore accommodate a wide range of bit-rates and offers the possibility to continuously adapt to the available networking bandwidth. Finally the FGS scalability can be combined with temporal scalability to further improve its flexibility. [8] discusses the Multiple Description (MD) video coding scheme that produces multiple independent layers of the video stream, each of the same importance. In the remaining of this paper we do not consider these MD coding schemes but focus on cumulative scalable encoding.
This discussion highlights several points: (1) video scalability is a complex feature of recent video codecs that often produces a *single* enhancement layer. (2) Splitting artificially this enhancement layer, or receiving only a subset of the associated data, does not always produce the expected result. (3) An exception is the MPEG-4 FGS scalability since the enhancement layer can be split into an arbitrary number of sub layers or can be partially received. We will see that our proposal is compatible with any scalability approach and does not assume the presence of a fine grained hierarchical encoding, even if it can be highly beneficial.

1.2 Layered and Single Layer Streaming Approaches

Many approaches exist for video streaming [8].

1.2.1 Layered Streaming Approaches

The streaming of scalable videos fits well with a transmission in cumulative quality layers. A traditional solution consists in mapping these video layers onto several multicast groups. In order to perform congestion control, each receiver dynamically adapts the number of layers received according to the experienced losses [15]. In order to behave correctly, this solution requires that the video layer granularity be fine enough, and a temporal scalability scheme is almost always assumed. Another requirement is that packets sent on the base layer experience no losses, because such losses usually trigger an important distortion in the reconstructed video (inter-layer dependencies). One solution to provide this transmission discrepancy is to protect data sent on the base layer with FEC (Forward Error Correction) techniques [13]. Another solution is to rely on a QoS differentiation mechanism within the network (Int-Serv or Diff-Serv), and to affect packets of the base layer to a prioritized service [2] [22]. The major practical limitation of this approach is the requirement to have a QoS service deployed between the source and each potential client (e.g. throughout the Internet in case of a public streaming service).
In [22] the authors introduce two Source-Adaptive Multi-layered Multicast (SAMM) algorithms to improve the transmission of layered video, by adjusting the number of layers and the bit-rate of each layer depending on feedback information sent either by network elements (network-based SAMM) and/or by receivers (end-to-end SAMM). This approach has several limitations: it requires that the source can dynamically adjust the number and bit-rate of the streams. Besides both variants require special features for the routers (priority drop preference, flow isolation, plus congestion notification with network-based SAMM).

1.2.2 Single-Layer Streaming Approaches

Another streaming solution consists in having a single video stream, mapped onto a single multicast group [3] [16] [17]. In that case the source adapts the transmission rate (e.g. by changing the video coding) according to RTCP feedback messages that give an indication on the experienced packet loss rate at receivers. Even if RTCP packets are rate-controlled (e.g. not to exceed 5% of the total session bit rate), this solution is not massively scalable. Besides, this solution is single-rate and consequently does not take into account the client heterogeneity.
A variant, called Simulcast in [9] and Destination Set Grouping (DSG) in [6], consists in generating multiple bitstreams of different bit-rates for the same content. Each client chooses the most adequate video bitstream according to its networking and processing capabilities, but switching to another bitstream dynamically is possible. This solution addresses client heterogeneity but requires to decide, at coding time, for a fixed total bit rate, how many bitstreams should be generated and their bit-rate. DSG addresses the second problem by using feedback messages (which is not the case of Simulcast) and adjusting the transmission rate of each channel within some predefined limits.
Our proposal completely departs from all of these approaches, makes no assumption neither on the services deployed within the backbone network nor on the video scalability scheme used. Our proposal is in fact mid-way between reliable multicast file transfer and streaming, and largely relies on the ALC layered reliable multicast protocol.

1.3 Introduction to ALC and its Congestion Control Protocol

The Asynchronous Layered Coding (ALC) protocol (RFC 3450) [10] of the IETF RMT working group is a layered reliable multicast protocol. Each receiver chooses how many layers to receive, depending on the bandwidth of its individual access network and on competing traffic. This receiver-driven decision is taken by an associated TCP-friendly layered congestion control protocol (e.g. RLC [21], FLID-SL/DL [4] [14], or WEBRC [12] [11]). Transmissions take place on the session layers either at some fixed predefined bit-rate (RLC, FLID-SL) or using a cyclic, dynamically changing bit-rate (FLID-DL, WEBRC), depending

on the associated congestion control protocol. Since neither ALC nor the congestion control protocol use any feedback to the sender, this solution is massively scalable in terms of number of receivers.
ALC is well suited to the transmission of popular content in an "on-demand" mode, where clients join an ALC session, retrieve data, and leave at their own discretion. This is made possible by the large use of FEC (Forward Error Correction) encoding [13], and by the transmission of all the packets (data and FEC) in a random order and continuously on the various ALC layers [5] [20]. This "on-demand" mode is very specific to ALC and other reliable multicast approaches (e.g. NORM and TRACK) are limited to a "push" synchronous model where all clients are supposed to be ready before the transmission starts. We will see that our approach relies on this "on-demand" model and is intrinsically linked to ALC.
The remainder of this paper is organized as follows: Section 2 introduces the general ideas of our proposal. Section 3 discusses how to initialize the various parameters and the associated trade-offs. Section 4 introduces some experimental results obtained on a local testbed with a full implementation of our proposal and of the ALC/RLC protocols. Finally Section 5 concludes the paper.

2 The Scalable Video Streaming over ALC (SVSoA) Approach

2.1 Principles

The SVSoA approach relies on ALC (and associated protocols)/UDP/IP as the transport/network layers, and is placed beneath RTP and the server or player application. The general architecture is illustrated in figure 1. Note that no RTCP back channel (e.g. to carry feedback information to the source) is used in SVSoA.

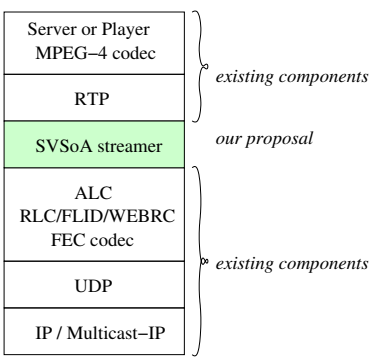


Figure 1: General architecture of the SVSoA approach.

2.1.1 Sender behavior

Let's consider a server who needs to stream a video that has been hierarchically encoded using one of the scalability schemes of section 1.1. The video consists of a base layer plus one or more enhancement layers. Unlike many previous works, we do not assume the presence of fine granularity, so having a single enhancement layer is sufficient.
The sender first partitions the video stream into *segments* of approximately the same duration, VSD (Video Segment Duration). By default $VSD = 60$ seconds, but other values are possible (section 3). Each video layer produces a *block*, of duration VSD. Each block is then sent independently on a *distinct ALC session* and thus on a different set of multicast groups as shown in figure 2 (note that video layers and ALC layers are two different notions). After VSD seconds, the server automatically switches to the next segment (at $t0+VSD$ on figure 2), and for each ALC session, the transmission of block n is stopped and replaced by block $n+1$.
During each period, the packets of a block are not sent sequentially but in a random order and cyclically, in order to offer an "on-demand" delivery mode. FEC packets included by ALC in the data stream enable

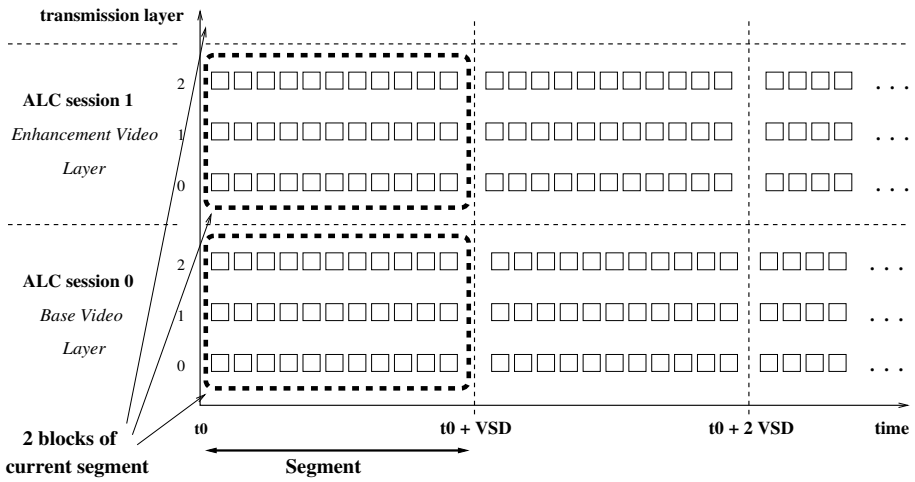


Figure 2: Sender behavior.

receivers to efficiently reconstruct some missing packets (either lost or not-yet received). This on-demand mode is required since receivers do not necessarily join at the beginning of a block transmission, especially on ALC sessions 1 and above. This is one of the reasons why ALC is absolutely required.

2.1.2 Receiver behavior

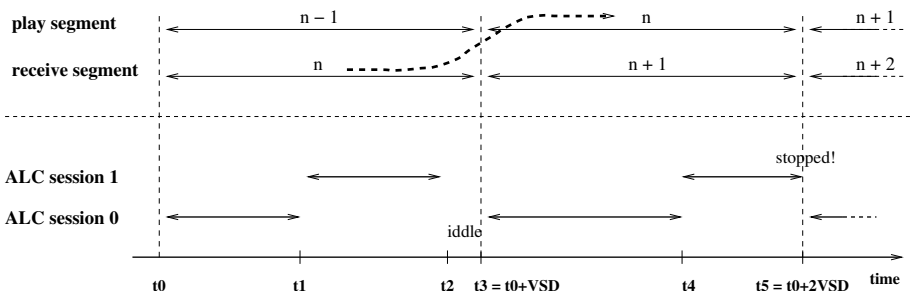


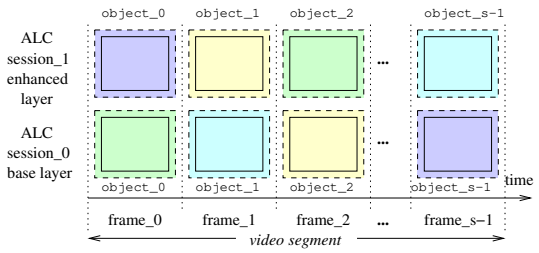
Figure 3: Receiver behavior.

At the beginning of a period or when a new receiver joins the SVSoA session, a receiver first subscribes to the ALC session where the base video layer of the current segment, n, is sent (from t_0 to t_1 in figure 3). When this block is successfully received, the receiver subscribes to the ALC session of the next enhancement video layer (from t_1 to t_2). This process stops (1) when all blocks have been successfully received, if ever (e.g. at time t_2), or (2) when the transmission of the next segment, $n+1$, begins. When transmissions for segment $n+1$ start, the receiver plays the video of segment n and switches to the first ALC session, and so on. Therefore a receiver always plays the previous video segment while receiving the current one, which of course introduces a playing latency of VSD seconds (section 2.3).

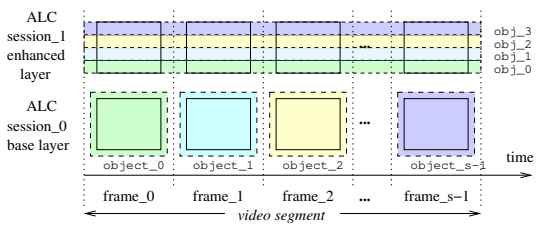
2.1.3 Receiver Synchronization

In order to keep receivers synchronized a descriptor is sent by the source for each video block with the following information: the number of frames in the current block, the first and the last RTP timestamp, the last RTP sequence number, and the duration, VSD, of the current segment. With this information a receiver knows when he completely received a block, and can detect the end of the current segment in order to switch to the next one. These descriptors can either be sent in-band (the solution we use), or via an out-of-band mechanism (which is out of the scope of this paper).

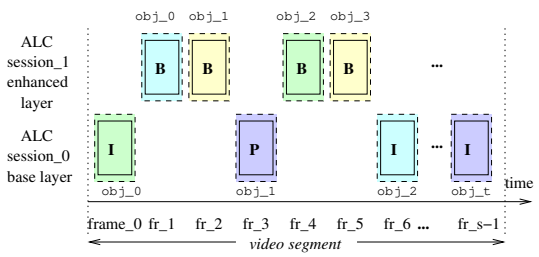
2.1.4 Definition of ALC Objects According to the Video Scalability Scheme



(a) Framing with Spatial Scalability



(b) Framing with FGS Scalability



(c) Framing with Temporal Scalability (with an IBBPBBP sequence)

Figure 4: Object framing according to the video scalability scheme.

The global reception efficiency is largely affected by the definition of ALC objects. Two points must be considered: (1) the relationships with the video scalability scheme, and (2) the size and number of the objects.

Object Framing The Application Level Framing (ALF) paradigm [7] tells us that each object should be autonomous and contain enough information to be processed by a receiver independently of other objects. We applied this principle and identified several framing possibilities: figure 4 (a) shows a possible framing in case of non-FGS spatial scalability. Since a receiver cannot take any benefit from receiving a subset of an enhancement frame, each frame is carried as a distinct ALC object. With an FGS video, figure 4 (b), the source can further split the frames of the enhancement block. Each slice, that now spans all the block frames, forms an ALC object. Each object is independent from others and provides some refinement over the whole video segment duration. *The video quality is therefore constant during each VSD period* and finely reflects the networking quality. Finally, figure 4 (c) shows the ALC object definition in case of a temporal scalability video. In this example each I and P frames of the base layer are independently

carried in a different object, as well as each B frame of the enhancement layer (note that many other I/P/B mappings are possible).

Object Size The other constraint is the object size. Having a single large object in an ALC session is usually more efficient, from a transmission point of view, than having a large collection of small objects because of the so-called coupon-collector problem [5]. This is obvious with a large block FEC codec [13] since the coupon-collector problem is then totally avoided. This is less true with a small block FEC codec, since this codec requires to split a large object into various blocks over which FEC is applied. In that case having several objects of the size of the FEC block (often around 64 kilobytes) does not bring any further limitations, but having smaller objects will.
This aspect must be considered, along with the desired granularity, when doing the framing of the *enhancement layer of an FGS video*. The optimal object size of the enhancement layer depends on the $desired_granularity$, the number, $frame_nb$, and size, $frame_sz$ (assumed constant) of the frames in the block, the maximum FEC block length, max_blk_len, (FEC dependent) as well as the minimum desired FEC block length, min_blk_len:

$$target_len = \frac{frame_nb * frame_sz}{desired_granularity}$$

$$obj_len = max(min_blk_len; min(target_len; max_blk_len)) \quad (1)$$

In practice the maximum and minimum FEC block length can limit the actual granularity obtained in the FGS enhancement layer.
With a non FGS or a temporal scalability scheme, the object framing is anyway determined by the frame size, since each frame is carried independently, and equation 1 cannot be applied. The transmission efficiency is then reduced since FEC encoding is applied independently to each small frame (object). Improving this behavior is discussed in section 4.2.3.

2.2 Benefits of the SVSoA Approach

The benefits of this approach are numerous, many of them being derived from the use of ALC as the underlying transport protocol.

2.2.1 Massively Scalable

This is a direct consequence of the use of ALC. Because no feedback of any kind is used at either ALC, the layered congestion control protocol, or SVSoA, it makes no difference to the video server whether there are very few or several hundreds of millions of simultaneous clients. Note that RTCP (the control protocol associated to RTP) is not used to carry feedback information to the source in SVSoA because it would be useless.

2.2.2 Exploits the Intra-ALC Congestion Control

Our approach takes advantage of the layered congestion control protocol used within each ALC session. SVSoA automatically benefits from the latests developments in TCP-friendly congestion control protocols. This is a major asset since a recent protocol like WEBRC proved to be highly effective: receivers quickly reach the equilibrium point, achieve a good TCP-friendliness and are not affected by the IGMP leave latency issue [12].

2.2.3 Immediately Deployable Anywhere

The SVSoA approach does not rely on any privileged transmission service nor on any specific feature within the backbone and can therefore be immediately deployed anywhere. This is made possible by the fact that a client receives only one video layer at a time, starting by the most important one (base layer). This solution therefore maximizes the probability of receiving the most important data correctly.

On the opposite, traditional layered streaming approaches often rely on the presence of a QoS mechanism within the core backbone to protect packets sent on the base layer (section 1.2.1). This is in practice a major limitation which adds much complexity to the solution and restricts its use to QoS-capable networks.

2.2.4 Addresses the Heterogeneity of Clients

Because ALC addresses the heterogeneity of clients, each SVSoA client receives the amount of video data made possible by its access network, independently of other clients. Besides, all clients are guaranteed to receive a minimum video quality before trying to receive any enhancement information.

2.2.5 Tolerant to Packet Loss Bursts

Because ALC is a reliable protocol, packet loss bursts are easily recovered, even in case of long lasting bursts (e.g. several tens of seconds), without any major impact on the video quality perception. The only requirement is that enough time is left to enable a receiver to receive at least the base video layer during the segment duration. More details are given in section 3.5.

2.2.6 Independent from the Video Scalability Scheme

Another benefit of SVSoA is that the congestion control efficiency does not depend on the number of the enhancement layers provided by the scalable video codec, and the nature of scalability used by this codec. This is a major advantage over traditional approaches that rely on a direct mapping between video layers and transmission layers (multicast groups) (section 1.2.1) and who implicitly assume the presence of a fine granularity video encoding. In practice this granularity is usually very low (e.g. the MPEG-4 ISO reference codec produces a single enhancement layer, section 1.1)!
In presence of a fine granularity scalability (FGS) video encoding (section 1.1), our approach can either rely on several ALC sessions, one per enhancement video layers, or a single ALC session for the enhancement video layer, since the FGS scheme enables a receiver to fully exploit a partial reception of the enhancement video layer. This feature will be highly beneficial but this is not made mandatory by SVSoA.

2.3 ... And the Price to Pay

2.3.1 A High Playout Latency

Since perfection rarely exists, the price to pay for these benefits is an important latency:

- when a new client joins an ongoing SVSoA session: this client experiences an initial latency comprised in:
$$0 < BLRT \leq initial_join_latency < VSD + BLRT < 2 * VSD$$
where $BLRT$ (Base Layer Receive Time) is the minimum time required to get the whole base video layer. The minimum join latency is experienced when the client joins the session $BLRT + \epsilon$ seconds ($\epsilon \ll 1$) before the end of the current segment since he has enough time to get the whole base video layer and can display it immediately after switching to the following VSD period. There is no enhancement layer during this first VSD period but the most important information is displayed. The worst join latency is experienced when the client joins the session $BLRT - \epsilon$ seconds before the end of the current segment, since he needs to wait an additional VSD period.

- during the video streaming: the video playout is always delayed by the segment duration parameter, VSD (typically 60 seconds, section 3.7). This feature prevents using SVSoA when interactivity (e.g. with tele-teaching) or immediate delivery (e.g. for a sport event coverage) are required.

A playout latency also exists in traditional unicast streaming solutions. This latency is caused by the need to buffer video data in order to counteract transmission jitter and packet loss bursts that may take place

later on during the streaming session. If this latency is usually smaller than the one experienced with SVSoA, the major difference is that SVSoA is designed for massively scalable video content distribution, whereas the number of simultaneous clients served by a unicast streaming server is by definition limited.

2.3.2 Additional Traffic

Another drawback is a high cumulative transmission rate at the source, since all layers for all ALC sessions are by default active. Yet multicast routing limits the traffic carried on the backbone by avoiding the transmission on branches that do not lead to a receiver. In practice, in the absence of receiver, the first hop multicast router prunes this traffic which only flows on the LAN of the source (which is usually not a problem).
Let's now consider a client. ALC introduces several inefficiencies: data and FEC packets are of finite number and can be duplicated. For instance the same packet can be received on two different layers at different times, or a packet for an already decoded block can be received later on. Some FEC codes also have intrinsic decoding inefficiencies. This is the case of "non-systematic" FEC codes like LDPC or Tornado where $(1+\epsilon)k$, $\epsilon \geq 0$, symbols are required to recover the original k data symbols. Some additional traffic will therefore be received compared to the strict minimum, which is unavoidable with a reliable multicast protocol. In section 4.2.1 we show that even in case of a rate limited environment, the SVSoA approach behaves efficiently and the extra traffic, in fact, enables clients to recover losses.

3 Analysis of the SVSoA Parameters

When deploying our solution some parameters must be adapted to the target environment (e.g. is it deployed in a closed environment like an hotel, or in the Internet), and to the video features (e.g. the bitrates of the base and enhancement video layers). In this section we explain how to optimally initialize two key parameters:

- the video segment duration (VSD), and
- the base transmission rate of each ALC session (b_0).

Several contradictory aspects must be considered when choosing a value for VSD. The first idea is to have a very short VSD in order to reduce:

- the storage capacity required at the source and at receivers, and
- the initial join latency and the playing delay.

But several considerations are against short video segment durations:

- the impacts of the IGMP leave latency when switching between two ALC sessions,
- the impacts of the congestion control protocol during the startup phase,
- and the maximum loss burst length that can be transparently recovered.

We now provide a theoretical analysis of each of these aspects.

3.1 Storage Requirements

The first limitation is the required storage capacity *at a server*. Since efficiency requires that packets are sent in a random order in each ALC session [20], the whole data set is usually stored in physical memory rather than on disk in order to avoid inefficient random I/Os. A direct consequence is that the available storage capacity (i.e. RAM) is quickly limited, and this is all the more true as the same server can stream several video contents simultaneously. The storage requirements amount to:

$$source_storage_req = VSD * cumul_enc_rate * FEC_ratio \quad (2)$$

Description	*Parameter*	*Default value*
SVSoA parameters		
Video Segment Duration	VSD	60 seconds
Object length	obj_len	
Desired granularity (in case of FGS)	$desired_granularity$	
IGMP inefficiency ratio	$igmp_ineff_ratio$	10%
Congestion control parameters		
ALC base layer transmission rate	b_0	
Increase rate	C	2 for RLC and FLID-SL 4/3 for WEBRC
Time cycle duration of the base layer	t_0	0.25 sec for RLC depends on b_0 and pkt_sz with FLID-SL
Time Slot Duration for FLID-SL	TSD	1 sec
Layer addition cycles for WEBRC	$epoch$	0.5 sec
ALC parameters		
Number of layers in an ALC session	$alay_nb$	
FEC ratio ($\frac{n}{k}$ ratio)	FEC_ratio	2
minimum FEC block length	min_blk_len	
maximum FEC block length	max_blk_len	64 KBytes
Reception inefficiency	rx_ineff	1.66
Packet size	pkt_sz	1024 Bytes
Network features		
Leave latency	$igmp_leave_lat$	3 sec
Video features		
Encoding rate of a single video layer	enc_rate	
Cumulated encoding rate over all video layers	$cumul_enc_rate$	
Number of video layers	$vlay_nb$	2
Number of frames per VSD	$frame_nb$	
Average frame size	$frame_sz$	

Figure 5: Overview of all parameters.

where $cumul_enc_rate$ is the cumulative video encoding bit rate over all layers, and FEC_ratio is the ratio of the number of packets after FEC encoding (FEC plus data) over the number of data packets ($\frac{n}{k}$ ratio).
Figure 6 illustrates the memory requirements at a source as a function of VSD for various encoding rates and
$FEC_ratio = 2$ (as many FEC packets as original packets). It shows that a high quality video encoded at 2 Mbps requires only 29.3 MB of storage capacity in the ALC component with $VSD = 60$ seconds, which is fairly reasonable. Having higher VSD values (e.g. 120 seconds) is not a problem either since the required memory only amounts to 58.6 MB. *We therefore consider that the storage capacity at a server is not a problem.*
The memory requirements *at a receiver* are lower since the FEC_ratio has no influence here (FEC decoding takes place immediately, as soon as enough packets have been received). But two segments can be buffered, the one being displayed and the one being received (in practice data displayed is immediately freed, so equation 3 is a pessimistic upper bound). Therefore:

$$receiver_storage_req = 2 * (VSD * cumul_enc_rate) \quad (3)$$

Besides, unlike a video server, a receiver is typically involved in a single video stream at a time. Buffering can yet be a problem for low end receivers (e.g. mobile devices). But in that case, because of the

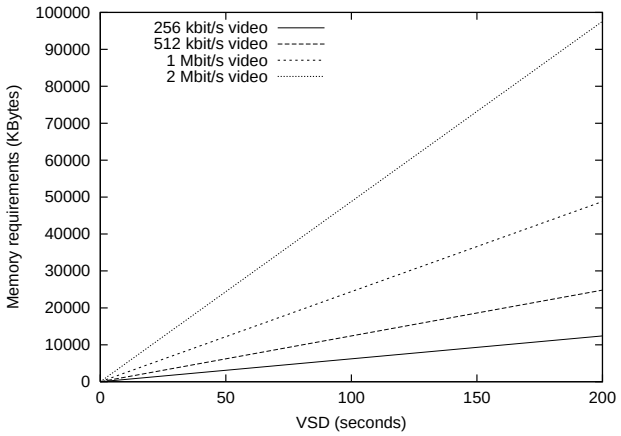


Figure 6: Storage requirements at a source.

limited bandwidth of the access network and/or the limited display capabilities, the video encoding rate will also be low, thereby limiting the storage requirements (e.g. 1.9 MB with a 128 kbps video, and $VSD = 60$ seconds). *We therefore consider that the storage capacity at a receiver is not a problem.*

3.2 Impacts of the IGMP Leave Latency

The second limitation are the *impacts of the IGMP leave latency*, i.e. the delay between when the last receiver of a LAN leaves a multicast group and its effect. This latency is usually 3 seconds (there are up to three instances of the IGMP polling message, with a typical 1 second polling delay each), but can be higher depending on the IGMP implementation. In addition to this delay, the multicast routing protocol itself can add its own pruning delay. Let $igmp_leave_lat$ be the sum of these latencies (assumed constant).

It is well known that the IGMP leave latency will affect the behavior of a layered congestion control protocol like RLM, RLC or FLID-SL. The only exceptions are the FLID-DL and WEBRC protocols which counteract this latency thanks to a dynamic layering approach [4] [12]. Let's now assume that a dynamic approach is used.

Even with such a protocol, our approach is still affected by the IGMP leave latency whenever a receiver changes of ALC session, for instance to receive the enhancement layer, or when switching to a new video segment. During $igmp_leave_lat$ seconds, packets of the previous ALC session still flow up to the receiver LAN, thereby preventing a normal behavior of the new ALC session. Two cases must be considered:

- the host is the only client in the LAN: in that case during $igmp_leave_lat$ seconds, packets of the previous ALC session still flow up to the receiver LAN where no receiver exist any more, thereby preventing a normal behavior of the new ALC session.
- other clients in the LAN are still joined to the previous ALC session. In that case two (or more) ALC sessions will coexist for some time. This second case highlights a limitation of the host heterogeneity handling with layered multicast approaches: heterogeneity is managed with a LAN granularity, not a host granularity.

The impacts of this latency are given by equation 4:

$$igmp_ineff_ratio = \begin{cases} (vlay_nb * igmp_leave_lat)/VSD & \text{if } vlay_nb > 1 \\ 0 & \text{if } vlay_nb = 1 \end{cases} \quad (4)$$

where $vlay_nb$ is the total number of video layers (base plus enhancement(s)) used by a receiver. The higher this inefficiency ratio, the higher the percentage of time wasted because of the IGMP and multicast routing protocol latencies. Figure 7 shows the evolution of this ratio as a function of the VSD and number of video layers produced by the video codec. In our case we are limited to two video layers ($vlay_nb = 2$) and assume that $igmp_leave_lat = 3$ seconds. In this case the $igmp_ineff_ratio$ amounts to 10% with $VSD = 60$ seconds, and 5% with $VSD = 120$ seconds. With five video layers, this ratio amounts respectively to 25.0% (prohibitive) and 12.5% respectively.
The IGMP leave latency largely impacts the solution efficiency and using a video segment duration of 60 *seconds is only possible with two video layers* (the common case). Having a higher number of video layers requires to significantly increase the VSD parameter.

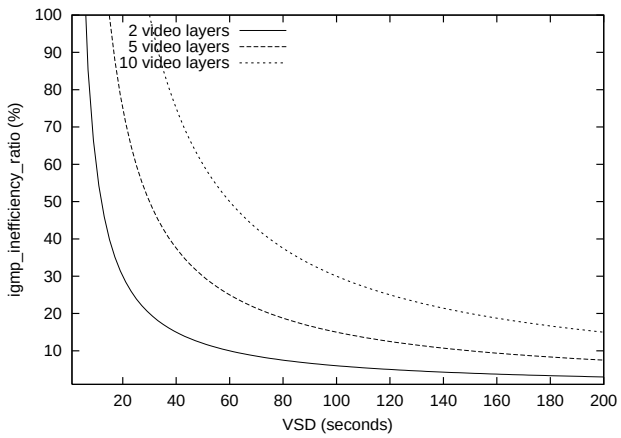


Figure 7: Impacts of the IGMP leave latency (igmp_ineff_ratio).

3.3 Impacts of the Congestion Control Protocol during Startup Phase

The third aspect to consider is the *congestion control protocol behavior during the startup phase for a given ALC session.* Because of this protocol, the reception rate at a client progressively increases until it reaches a "fair share" (the exact fairness depends on the protocol in use) of the available bandwidth between the source and the client. Depending on the protocol, the time required to reach the steady rate is not so small compared to the VSD parameter and must be considered. This problem affects a client each time he joins a new ALC session, for instance to receive the enhancement video block or when switching to the following video segment. Starting from scratch after joining the following ALC session is the default behavior. In section 3.4 we describe an optimization whereby a client bypasses the startup phase and instead keeps the subscription level of the previous ALC session. Yet, under some circumstances, this optimization is not possible and the results of the present section apply. We also assume in this section that a client does not experience any loss. The case of lossy transmissions will be considered in section 3.5.
We now give a mathematical model of the startup behavior of various congestion control protocols and define a minimum value to VSD taking into account the video features, in particular its encoding rate.

3.3.1 Startup Behavior with RLC

In RLC [21] a receiver experiencing no loss can add a new layer upon the reception of a dedicated "increase signal". These signals are exponentially distributed over the layers, using a factor of two, making the opportunities of adding higher layers less frequent than with lower layers. The minimum delay, t_l, after which layer l can be added, if no loss occurs, is:

$$t_l = (2^l - 1)t_0$$

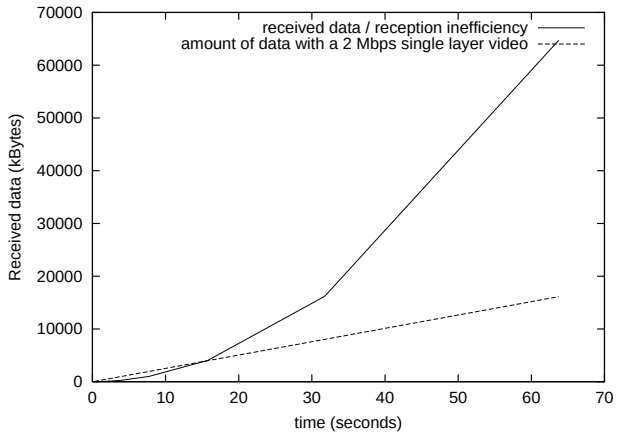


Figure 8: Amount of received data with RLC.

where t_0 is a fixed period (we use $t_0 = 0.25$ seconds in our experiments). Then each "increase signal" of layer l is repeated with a period:

$$T_l = 2^l t_0$$

The transmission rate of layer $l \in \{0; alay_nb - 1\}$, where $alay_nb$ is the total number of layers in an ALC session, follows a doubling scheme:

$$b_l = \begin{cases} b_0 & \text{if } l = 0 \\ 2^{l-1} b_0 & \text{if } l \geq 1 \end{cases}$$

Therefore, the amount of data received through a single ALC session in the startup phase, at time $t = i * t_0$, multiple of the RLC's time slot period, is:

$$\begin{aligned} Rx(t = i * t_0) &= data_received_on_base_layer + \\ &\quad \sum_{l \in \{active_upper_layers\}} data_received_on_layer\ l \\ &= b_0 \sum_{k=0}^{i-1} t_0 + \sum_{l:\ l>0\ and\ 2^l < i+1} 2^{l-1} b_0 \sum_{k=2^l-1}^{i-1} t_0 \\ &= b_0 t_0 \left(i + \sum_{0<l<\log_2(i+1)} 2^{l-1}(i+1-2^l) \right) \end{aligned}$$

If we only consider the moments when a new layer is added, i.e. if $\exists\ L : i + 1 = 2^L$, then this equation can be simplified:

$$Rx(t = i * t_0) = \frac{i(i+2) b_0 t_0}{3} \tag{5}$$

In practice, the use of ALC introduces some inefficiency (section 2.3.2). To take it into account, we introduce a *global reception inefficiency ratio*, assumed constant:

$$rx_ineff = \frac{nb\ of\ received\ pkts}{nb\ of\ usefull\ pkts} = 1 + \frac{nb\ of\ extra\ pkts}{nb\ of\ usefull\ pkts} \geq 1$$

To recover len bytes of data, the following inequation must be true:

$$\frac{Rx(i)}{rx_ineff} \geq len$$

We can now calculate the minimum time required to entirely receive one video block, of enc_rate encoding rate, with the associated ALC session. This is the minimum solution i of equation:

$$\frac{Rx(i)}{rx_ineff} \geq enc_rate * i * t_0$$

Figure 8 shows the $Rx(i)/rx_ineff$ curve and the amount of video data curve as a function of time, when using the following parameters: $b_0 = 160$ kbps, $t_0 = 0.25$ sec, $rx_ineff = 1.66$ [20], and $enc_rate =$ 2 Mbps. The minimum duration of a segment VSD_{min} is the intersection of the two curves. We find: $VSD_{min} \approx 63 * t_0 = 15.75$ sec. With two video layers, each encoded at 2 Mbps, we have to double VSD_{min} (31.5 sec).

3.3.2 Startup Behavior with FLID-SL

We now consider the FLID-SL (Static Layer) congestion control protocol [4] which shares many similarities with RLC. The b_l can follow a multiplicative scheme:

$$b_l = \begin{cases} b_0 & \text{if } l = 0 \\ (C^l - C^{l-1})b_0 & \text{if } l \geq 1 \end{cases}$$

where $C > 1$ is the multiplicative factor. A particular case is a doubling scheme, like RLC, where $C = 2$. The main difference between FLID-SL and RLC concerns the period T_l between two "increase signals" on layer l. T_l depends on a probabilistic function p_l which indicates the probability to increase the subscription layer in each time slot. On average T_l is given by:

$$T_l = \frac{TSD}{p_l}$$

where TSD is the the Time Slot Duration. [4] proposes a heuristic for p_l to mimic TCP. [14] suggests a simpler scheme that we consider here:

$$p_l = min\left(1.0, \frac{20 * pkt_sz * TSD}{2^l b_0}\right)$$

We can assume that the right expression is smaller than 1.0 (usually true expect for small values of l) and consequently we do not consider the $min()$ function. Then:

$$T_l = \frac{TSD}{p_l} = C^l \frac{b_0}{20 * pkt_sz} = C^l * constant$$

is similar to RLC where $C = 2$, with $t_0 = \frac{b_0}{20 * pkt_sz}$. The amount of data received through a single ALC session in the startup phase, at time $t = i * t_0$, with the above value of t_0, is given by:

$$\begin{aligned} Rx(t = i * t_0) &= data_received_on_base_layer + \\ &\quad \sum_{l \in \{active_upper_layers\}} data_received_on_layer\ l \\ &= b_0 \sum_{k=0}^{i-1} t_0 + \sum_{l:\, l>0 \text{ and } C^l < i+1} (C^l - C^{l-1}) * b_0 \sum_{k=C^l-1}^{i-1} t_0 \\ &= b_0 t_0 \left(i + \sum_{0<l<\log_C(i+1)} (C^l - C^{l-1}) * (i + 1 - C^l) \right) \end{aligned}$$

If we only consider the moments when a new layer is added, i.e. if $\exists\, L : i + 1 = C^L$, then this equation can be simplified:

$$\begin{aligned} Rx(t = i * t_0) &= \left(i + \frac{i(i + 1 - C)}{C + 1} \right) * b_0 t_0 \\ &= \frac{i(i + 2) b_0^2}{(C + 1) * 20 * pkt_sz} \end{aligned} \tag{6}$$

If $C = 2$, we have:

$$Rx(t = i * t_0, C = 2) = \frac{i(i+2)b_0^2}{60 * pkt_sz} \tag{7}$$

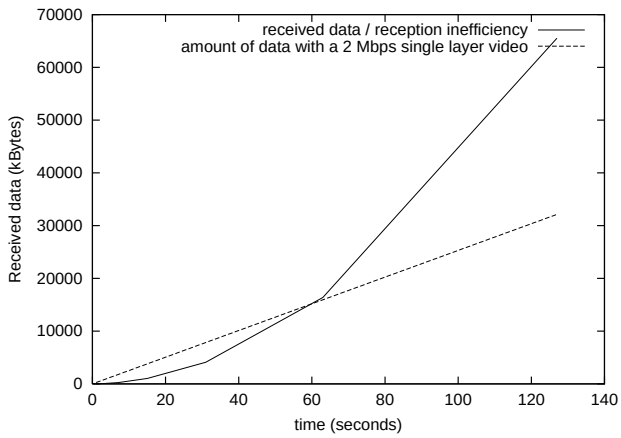


Figure 9: Amount of received data with FLID-SL, when $C = 2$.

As for RLC, a reception inefficiency ratio must be considered. Figure 9 shows the $Rx(i)/rx_ineff$ curve as a function of time, when using the following parameters: $C = 2$ (doubling scheme), $b_0 = 160$ kbps, $pkt_sz = 1024$ bytes, $rx_ineff = 1.66$ [20], and a single video layer encoded at $enc_rate = 2$ Mbps. We obtain the minimum segment duration at the intersection of the two curves: $VSD_{min} \approx 62$ seconds.
We see that FLID-SL and RLC give similar results, even if FLID-SL, with the default parameters suggested in the literature, leads to a slower reception rate progression and requires a higher VSD value. We do not consider the FLID-DL (Dynamic Layering) scheme [4] here, since WEBRC replaces FLID-DL favorably.

3.3.3 Startup Behavior with WEBRC

WEBRC behaves differently than RLC or FLID-SL and is capable of adding layers much faster during the startup phase. Indeed the reception speed is multiplied by a factor of $C = \frac{4}{3}$ each epoch (by default 0.5 second), creating an exponential increase (TCP does the same since the transmission rate is doubled each RTT during the first stage of the slow-start algorithm), and then stabilizes around a fair share of the available bandwidth, determined through a TCP throughput equation. Consequently *SVSoA behaves much better with WEBRC* than with the RLC/FLID-SL protocols and the startup phase is less a limiting factor.
Let's now continue with a mathematical model of the WEBRC startup. Every *epoch*, the reception rate is increased by a factor C. The amount of data received through a single ALC session in the startup phase, at time $t = i * epoch$, multiple of the *epoch* time slot period, is:

$$Rx(t = i * epoch) = \sum_{l=0}^{i-1} C^l * b_0 * epoch$$

By resolving the sum we obtain:

$$Rx(t = i * epoch) = \frac{(C^i - 1)b_0 * epoch}{C - 1} \tag{8}$$

This equation confirms the exponential increase of the reception rate (in $(\frac{4}{3})^i$ instead of i^2 with RLC). Figure 10 compares the WEBRC and RLC behaviors. The WEBRC startup is rather slow, and then, after 21 seconds, the WEBRC curve increases exponentially, crossing the RLC curve. This curve shows that *WEBRC is well suited to long VSDs* since a receiver benefits from the exponential behavior of the reception rate and quickly reaches the TCP equivalent throughput.

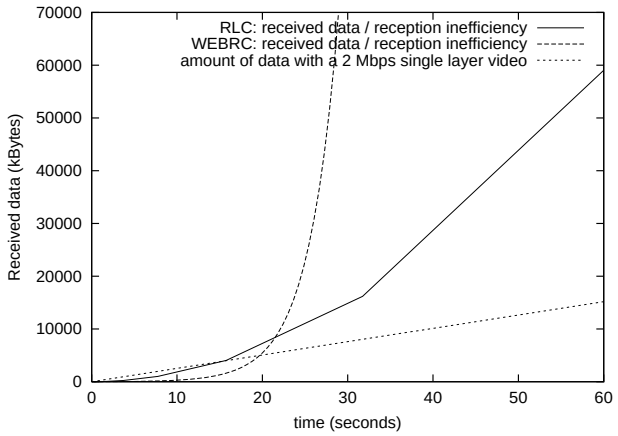


Figure 10: Amount of received data with WEBRC and comparison with RLC.

3.4 Avoiding the Startup Phase of the Congestion Control Protocol

The behavior of congestion control discussed in section 3.3 is penalizing. Therefore we now introduce an optimization that, under some circumstances, avoids this startup phase. The idea is to keep some reception rate information when switching from ALC session, either to receive the following enhancement block or at the end of the segment. With this information, the client starts immediately at a reception rate equal to $\alpha \leq 1$ times the previous reception rate, where the α parameter is a security (e.g. with RLC, it can be equal to 0.5, which means one layer less than in the previous session).
This optimization can only be used if:

- there is *no significant idle period* before switching to the new ALC session. If a high speed client finishes receiving all video blocks well before the segment end, then the networking conditions may have changed when joining the ALC session of the following video segment. In that case a standard congestion control startup phase must be used.
- there is a *single client after the bottleneck link or router*. This will be usually the case if clients are connected through a low-rate line (e.g. 128 kbps ISDN). Otherwise several concurrent ALC sessions could compete for the bandwidth and only a standard congestion control protocol can regulate this situation.

Note that SVSoA can be used without this optimization, and adopting a conservative solution with the standard startup congestion control phase should be preferred in environments where the above requirements are not guaranteed.

3.5 Packet Loss Recovery Capabilities

The VSD parameter has a direct impact on the packet loss recovery capabilities of SVSoA. Losses in the Internet usually occur in bursts, because of router congestion problems or routing instability. Thanks to ALC's reliability mechanisms (in particular the use of FEC within ALC), these losses can usually be recovered, at least for the base layer which contains the most valuable video information. Intuitively, the longer the video segment duration (VSD), the greater the immunity to losses, and the longer the loss burst that can be recovered. In this section we analyze the SVSoA robustness assuming that a single burst, of duration $loss_dur$, occurs during a video segment.
The goal of this analysis is to *have an idea on how to initialize the VSD parameter to obtain a certain target robustness*, and what are the other parameters that affect this robustness. The simplification made (single loss burst) does not catch the SVSoA behavior in front of other loss models (e.g. with random isolated losses, or in case of several small loss bursts rather than a single long burst). Yet our scheme also brings some robustness in front of other loss models. The exhaustive analysis of the SVSoA's behavior

is left to future works. By default, we only consider the base video layer in this analysis. No guaranty is given for the enhancement layer(s).

3.5.1 Loss recovery capabilities with RLC

This analysis is based on the equations obtained in section 3.3 giving the amount of data transferred after a certain time, when no loss occurs, and taking into account the inefficiency ratio of ALC, rx_ineff. In equations 5 we can ignore the i factor in front of i^2 (remind that $i * t_0$ is the elapsed time). We can also ignore the non-continuous behavior of the congestion control protocol and use a continuous time instead. We can then write:

$$\frac{Rx(t)}{rx_ineff} \approx c * t^2$$

where c is a constant. By comparing with equation 5:

$$\frac{i^2 b_0 t_0}{3 * rx_ineff} = c(i * t_0)^2$$

and the value of the constant c is given by:

$$c = \frac{b_0}{3 * t_0 * rx_ineff} \tag{9}$$

Figure 11 illustrates the robustness problem when $VSD = 60$ seconds and with a video encoding rate $enc_rate = 2$ Mbps. Let t_{min} be the time required to receive the amount of video data sent during VSD seconds:

$$\frac{Rx(t_{min})}{rx_ineff} = VSD * enc_rate$$

In that case the maximum loss duration is the extra time available at the end of the video segment: $VSD - t_{min}$, and we find (graphically) a value of ≈ 32 seconds.

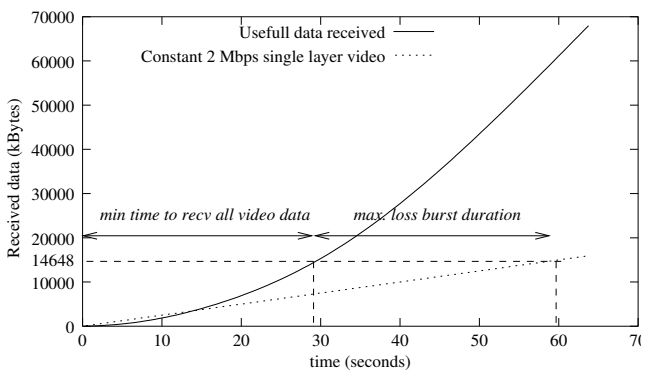


Figure 11: Maximum recoverable loss burst length; here the loss burst occurs at the end of the video segment period (60 s).

This is in fact an upper bound and the robustness is largely impacted by the position of the loss burst in the video segment. The maximum recoverable loss period is indeed reduced when the loss period starts in the middle of the video segment, because of the congestion control algorithm which slows down the reception rate after a loss. Depending on the length of the burst, the congestion control algorithm restarts reception at a subscription level j smaller than the subscription level i before the start of the burst: $0 \leq j \leq i - 1$. The worst case where all layers are dropped ($j = 0$), is illustrated in figure 12. The maximum recoverable burst length is then only 17 seconds which is now a lower, pessimistic, bound.

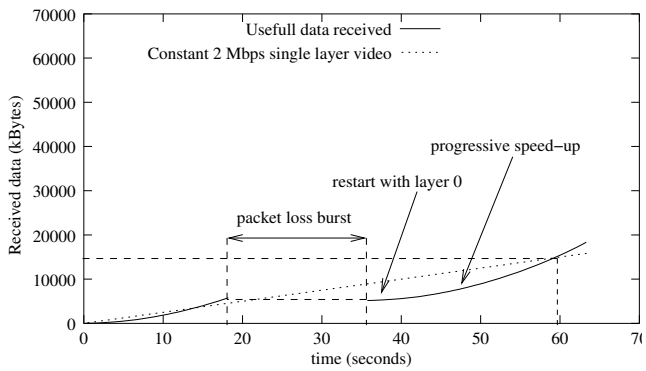


Figure 12: Impacts of a loss burst in the middle of the video segment period (60 s).

Let's now continue with a formal discussion of the problem. Let's t_1 and t_2 be respectively the time before and after the loss burst, of duration t_{loss}. We pessimistically assume that the burst leads the client to drop all layers. We have:

$$VSD = t_1 + t_{loss} + t_2$$

Transmission is successful if:

$$\frac{Rx(t_1) + Rx(t_2)}{rx_ineff} \geq enc_rate * VSD$$

or:

$$c * t_1^2 + c * t_2^2 \geq enc_rate * VSD$$

After replacing t_2 and extracting t_{loss}:

$$t_{loss} \leq VSD - t_1 - \sqrt{\frac{enc_rate * VSD}{c} - t_1^2} = f(VSD, t_1) \tag{10}$$

with the following definition interval:
$t_1 \in [0; t_{1_max} = \sqrt{\frac{enc_rate*VSD}{c}}]$. The recovery capability is maximum for $t_1 = 0$, then a minimum recovery capability is reached at $t_1 = \sqrt{\frac{enc_rate*VSD}{2c}}$:

$$t_{loss_min}(VSD) = VSD - \sqrt{\frac{2 * enc_rate * VSD}{c}} \tag{11}$$

and then the recovery capability increases up to the same maximum obtained for t_{1_max}. Figure 13 illustrates the maximum recoverable burst length as a function of VSD and t_1. It uses (eq. 9): $c = \frac{160000}{3*0.25*1.66} = 128,000$ bits/sec^2. We find: $t_{loss_min}(60s) = 16.70$ sec. These curves confirm the high importance of the position of the loss burst in the video segment, and that of the VSD parameter.

3.5.2 Loss recovery capabilities with WEBRC

With WEBRC, layers are added much faster and SVSoA performs better and tolerates longer loss bursts. Similar to RLC and FLID-SL, We can approximate the behavior of WEBRC (simplify the formula 8) by considering the exponential startup behavior of WEBRC and write:

$$\frac{Rx(t)}{rx_ineff} \approx c * C^t$$

where c is a constant. By comparing with equation 8:

$$\frac{C^i * b_0 * epoch}{(C-1) * rx_ineff} = c * C^i$$

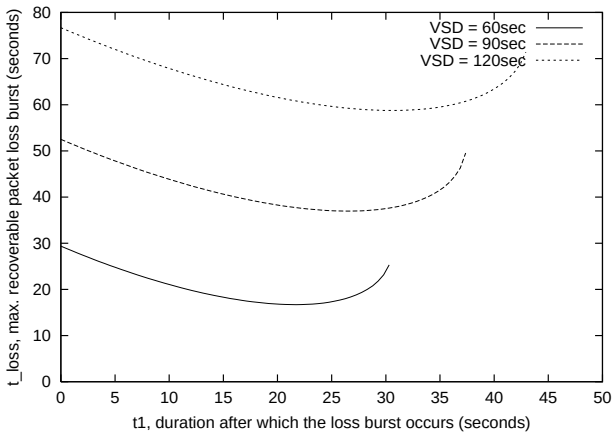


Figure 13: Maximum recoverable packet loss burst length when using RLC.

and the value of constant c is given by:

$$c = \frac{epoch * b_0}{(C-1) * rx_ineff}$$

We now continue our considerations with $C = 4/3$ and $epoch = 0.5$ seconds as proposed in [12]. c is now:

$$c = \frac{3 * b_0}{2 * rx_ineff} \tag{12}$$

Let's now continue with a formal discussion of the problem. Let's t_1 and t_2 be respectively the time before and after the loss burst, of duration t_{loss}. We pessimistically assume that the burst leads the client to drop all layers. We have:

$$VSD = t_1 + t_{loss} + t_2$$

Transmission is successful if:

$$\frac{Rx(t_1) + Rx(t_2)}{rx_ineff} \geq enc_rate * VSD$$

or:

$$c * (4/3)^{t_1} + c * (4/3)^{t_2} \geq enc_rate * VSD$$

After replacing t_2 and extracting t_{loss}:

$$t_{loss} \leq \frac{ln(\frac{enc_rate*3^{t_1}VSD - c*2^{2t_1}}{c}) - 2 * t_1 * ln(3/2) + VSD * ln(3/4)}{ln(3/4)} \tag{13}$$

with the following definition interval:
$t_1 \in [0; t_{1_max} = \frac{ln(\frac{c}{enc_rate*VSD})}{ln(3/4)}]$. The recovery capability is maximum for $t_1 = 0$, then a minimum recovery capability is reached at $t_1 = \frac{ln(\frac{2*c}{enc_rate*VSD})}{ln(3/4)}$:

$$t_{loss_min}(VSD) = \frac{ln((\frac{c}{enc_rate*VSD})^{\frac{2*ln(2)}{ln(3/4)}})}{ln(3/4)}$$
$$- \frac{2 * ln(\frac{c}{enc_rate*VSD}) * ln(3/2) - (VSD * (ln(3/4)) - 2 * ln(2)) * (ln(3/4))}{ln(3/4)^2} \tag{14}$$

and then the recovery capability increases up to the same maximum obtained for t_{1_max}. Figure 14 illustrates the maximum recoverable burst length as a function of VSD and t_1. It uses (eq. 12): $c = \frac{3*160000}{2*1.66} = 144578$ bits/sec. With the same parameters as with RLC, we find: $t_{loss_min}(60s) = 18.09$ sec which can be compared with the 16.70 sec of RLC. The benefits of WEBRC compared to RLC increase when VSD increase.

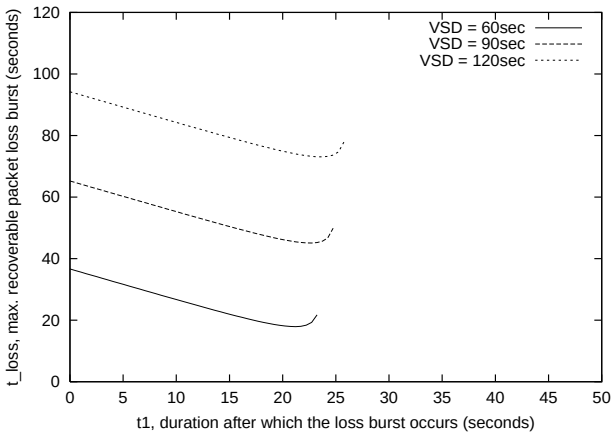


Figure 14: Maximum recoverable packet loss burst length when using WEBRC.

3.6 Transmission Rate of the ALC Session

Another important parameter is b_0, the transmission rate of the base ALC layer. A higher b_0 leads to a faster reception in the startup phase, linearly with RLC, in b_0^2 with FLID-SL. But a high b_0 also limits the possibilities to serve low end receivers. Therefore the following aspects must be considered:

- Required reception time in front of the video encoding rate: The higher the video encoding rate, the larger b_0 should be.
- The target environment (closed network, Internet,...):
  This target environment defines the possible heterogeneity of the users and their access networks. When serving Internet clients, b_0 should be compliant with the slowest possible client.

3.7 Initializing SVSoA in Practice: a Summary

In practice SVSoA can be correctly initialized according to the following steps:

step–1: Retrieve the average encoding rate enc_rate and number of layers of the video, $vlay_nb$ (usually 2).

step–2: if $vlay_nb \geq 2$, then use $VSD = 30 * vlay_nb$ to keep the $igmp_ineff_ratio$ constant and equal to 10%, assuming an IGMP latency of 3 seconds (equation 4). if $vlay_nb = 1$, then IGMP has no effect, and using $VSD = 60$ seconds for instance is appropriate.

step–3: in case of an FGS video encoding, define an appropriate obj_len from equation 1.

step–4: With RLC, retrieve the t_0 parameter, and with WEBRC the *epoch* parameter.

step–5: Estimate the reception inefficiency of ALC and its FEC code (e.g. we use 1.66).

step–6: Define the transmission rate on the base ALC layer, b_0. This choice depends on the target environment since this is the minimum reception rate.

step–7: Calculate the minimum loss burst immunity for the base video layer. With RLC use equation 11: $t_{loss_min} = VSD - \sqrt{\frac{6*enc_rate*VSD*t_0*rx_ineff}{b_0}}$. With WEBRC, use equation 14. If this value is judged too low, it means that the video bit-rate is too high compared to the transmission rate. So increase the b_0 value and reiterate at step 6. Alternatively we can set a higher VSD and reiterate step 7.

Many parameters are only approximations. For instance the video encoding rate is only an average (see figure 15), and the rx_ineff is not known in advance. This is not a major issue yet since several equations of previous sections (e.g. $t_{loss_min}(VSD)$) are for the worst case and rely on pessimistic assumptions (e.g. the client drops all the layers after the loss burst).

4 Experimental Results

4.1 Experimental Conditions

4.1.1 The Streaming Approaches Compared

We implemented our proposal using the MCL library that implements the ALC/RLC protocols [19] (we do not use the FLID-SL protocol during these tests even if it is available), and the MPEG4IP [1] MPEG-4/RTP streaming application.
We compare the SVSoA performances with a "classical" streaming scheme using a *single ALC session*, where each video layer is mapped onto a distinct ALC layer, and using the same RLC congestion control protocol as SVSoA. The sender sends each frame sequentially and no FEC protection nor any QoS mechanism is used. We are aware of the fact that the use of some protection mechanism for at least the base video layer is highly recommended. Yet adding proactive FEC consumes additional bandwidth and assuming a QoS service would not enable a fair comparison with SVSoA. This "classical" streaming scheme is obviously suboptimal (section 4.3) but is sufficient to better highlight some distinctive features of SVSoA.

4.1.2 Video Encoding and SVSoA Parameters

Our tests use a 120 second video, encoded with a constant 25 fps frame-rate, and using a spatial scalability encoding which provides two video layers, a base layer and a single enhancement layer. The bit-rate of both layers is on *average* 667 kbps respectively, but the instantaneous bit-rate fluctuates around this average, as shown in figure 15 (this figure represents the sum of the two video layers bit-rate).
In practice a dedicated MPEG-4 decoding hardware would be used by clients for real-time decoding. Since this facility is not available in our testbed, the video is encoded off-line, and the decoding/playing features at a client are turned off. We thus focus on transmission aspects and avoid any interference with CPU intensive tasks.
SVSoA is initialized with $b_0 = 131$ kbps and $VSD = 60$ seconds. The theoretical minimum loss recovery capability is $t_{loss_min} = 21.0$ seconds.

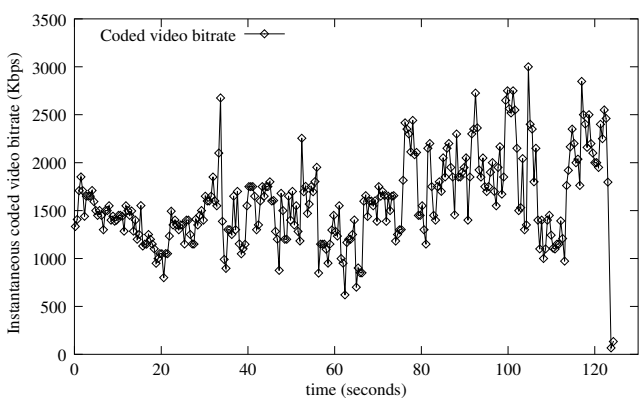


Figure 15: Instantaneous video bitrate of the two layers.

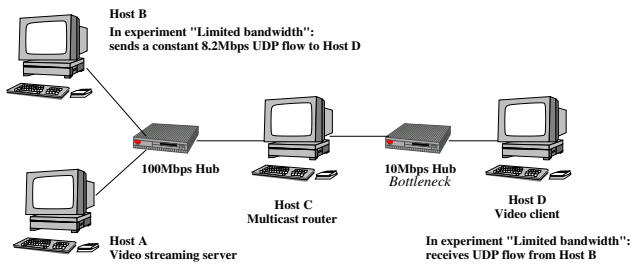


Figure 16: Our testbed.

4.1.3 Testbed and Scenarios

Our testbed is represented in figure 16. It is composed of four PC P-III/Linux attached to two different Ethernet subnets. The right subnet has a limited bandwidth of 10 Mbps. Host C in the middle is the multicast router and uses the `mrouted` multicast routing daemon. Host A is the SVSoA streaming server and host D the SVSoA client. Two scenarios are used:

1. *Limited bandwidth:* We evaluate both streaming approaches in presence of a concurrent network flow and in a limited bandwidth environment.

2. *Long burst of packet losses:* This test demonstrates the error recovery capabilities of SVSoA (section 3.5).

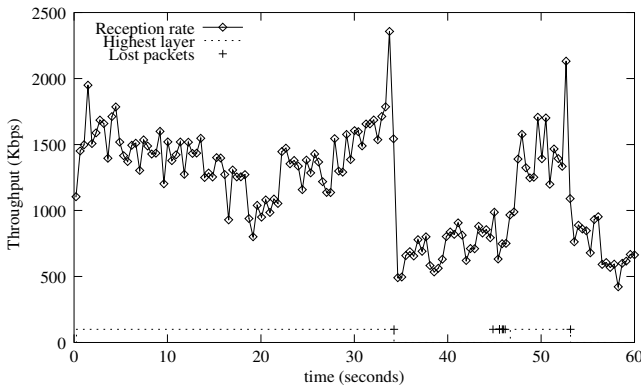
4.2 Experimental Results

4.2.1 Behavior in a Limited Bandwidth Environment

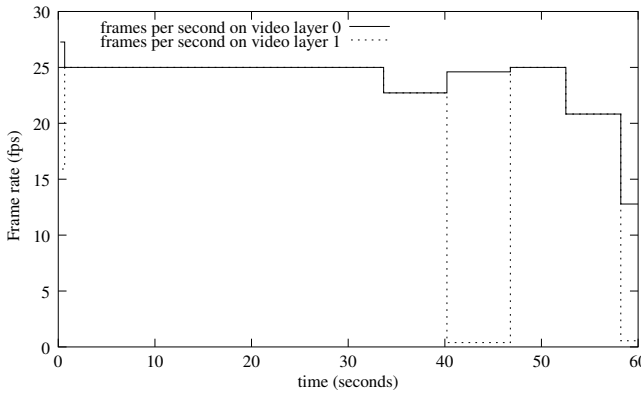
We evaluate both streaming approaches in presence of a concurrent network flow and in a limited bandwidth environment. During this test, host B sends a constant bit-rate 8.2 Mbps UDP flow to host D. Host D receives both the UDP flow from B and the video from A.
With the classical approach, the transmission rate for each video layer depends on the instantaneous video bit-rate. When the cumulative rate on both layers exceeds the available bandwidth (e.g. after 35 seconds in figure 17 (a)), packets get lost. According to RLC, the client then drops a layer, which reduces the video quality, in order to adapt to the available network bandwidth. Some time after (47 seconds), the enhancement layer is once again added, which triggers losses some time after, and the client drops the higher layer one more time. We see that the rough granularity provided by the spatial scalability encoding does not enable an efficient behavior. We also see that losses affect both video layers, and each loss on the base layer prevents the client to decode the associated frame on the higher layer if received which amplifies the phenomenon (figure 17 (b)).
This is not the case with SVSoA where the reception rate is only determined by the congestion control protocol, independently of the video bit-rate (compare figures 17 (a) and 18 (a)). Transmissions are smoothed over the time which is an advantage from a networking point of view. Figure 18 (b) shows that the frame rate of the base layer is constant during the whole test, and a minimum video quality is provided to the client. However the frame rate of the enhancement layer is relatively low since the spare bandwidth is not sufficient to receive more frames.

4.2.2 Behavior With a Long Burst of Packet Losses

This test demonstrates the error recovery capabilities of SVSoA (section 3.5). 10 seconds after the start of the video transmission, we unplugged the network cable during 15 seconds, and reconnect it up to the end of the test. No background traffic is used in this test.
The reception for both approaches is totally interrupted while the cable is unplugged (between $t = 10s$ and $t = 25s$ in figures 19 (a) and 20 (a)). With the classical approach, the reception restarts with the base video layer only, the enhancement layer being only added after some time, according to RLC. SVSoA



(a) Reception rate with the classical approach



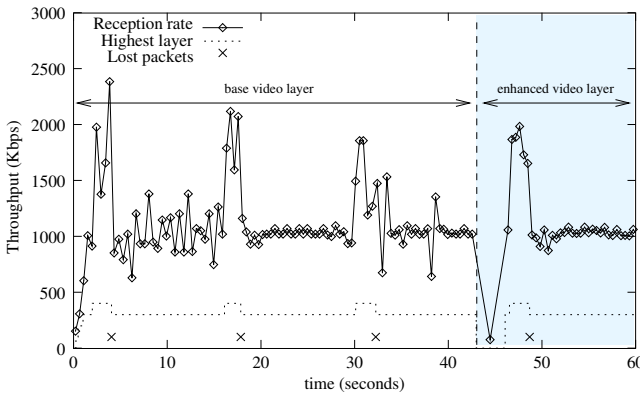
(b) Frame rate with the classical approach

Figure 17: Behavior with limited bandwidth and the classical approach (first 60 seconds).

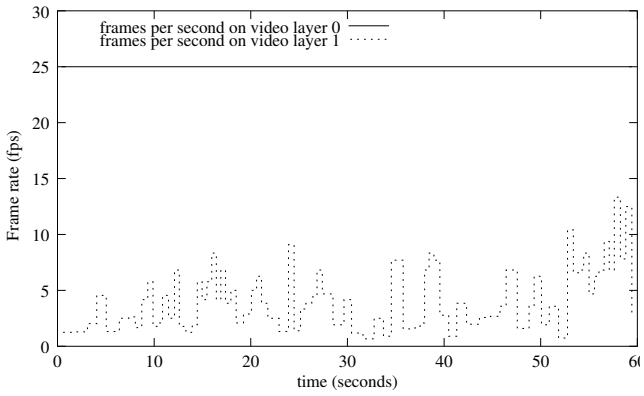
restarts receiving normally after the burst but stays in the base video layer as long as it is not completely received.
The frame rate of the resulting video at the client shows a big hole on both video layers with the classical approach (figure 19 (b)). No frame can be displayed while the cable stays unplugged (there is a small delay due to buffering for jitter compensation). For the enhancement layer, this time is a bit longer since the ALC layer is only added after an additional delay once the reception becomes normal.
With SVSoA the frame rate stays constant as if there was no disturbance (figure 20 (b)). The client receives the whole base video layer, thereby assuring a minimum video quality. The frame rate of the enhancement video layer is not optimal though for the first segment. This is due to fact that the client needs more time to reconstruct the base layer , which leaves less time to receive the enhancement video layer. But the lost frames of the enhancement layer are spread equally over the whole segment, which provides a *globally constant video quality*. The second segment is perfectly received.

4.2.3 Evaluation of the rx_ineff Ratio

We evaluated the practical rx_ineff ratio of SVSoA during a standard transmission (no background traffic). We find for the ALC session 0 where is sent the base video layer: $rx_ineff = 1.191$ which is rather good. On the opposite, for the ALC session 1 where is sent the enhancement video layer: $rx_ineff = 2.576$ which is really poor. The reason for this difference is that we use in fact a single ALC



(a) Reception rate with SVSoA



(b) Frame rate with SVSoA

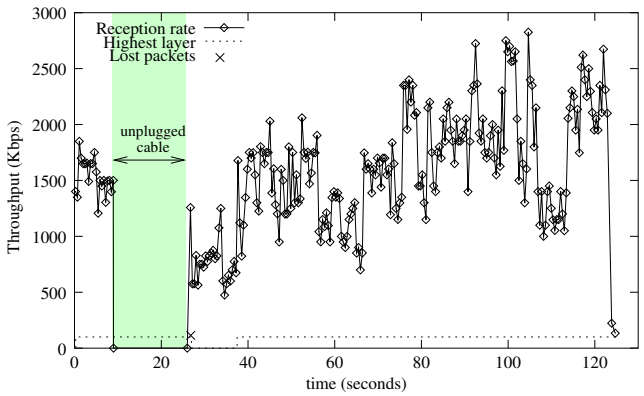
Figure 18: Behavior with limited bandwidth and SVSoA (first 60 seconds).

object (6.4 MB) for all frames of the base video layer in order to improve FEC encoding[1], whereas there are 1500 ALC objects (one per frame) on the enhancement layer, each around 4.3 KB long. Managing a huge number of very small ALC objects naturally creates severe inefficiencies with the current ALC implementation used. We are currently working on an FEC encoding scheme that encompasses several (small) objects. This solution should definitively solve the issue and largely improve the SVSoA efficiency.

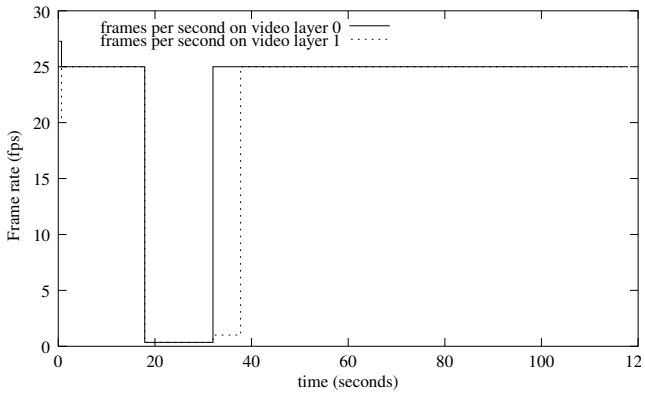
4.3 Summary of Experiments and Discussion

Non surprisingly, the SVSoA performances largely outperform that of a classical streaming approach. This classical approach, which serves as a reference, is obviously not optimal, and could benefit from several improvements, like: (1) the use of a QoS service to protect the base layer, (2) the addition of redundancy (FEC) to protect the base layer, (3) the use of a finer video scalability granularity, and (4) a really constant bit-rate video encoding. Yet SVSoA assumes neither the availability of a QoS service, nor the presence of fine granularity, nor a constant bit-rate encoding, which are three major, very restrictive, assumptions. Adding redundancy in a proactive way raises major problems. How much redundancy should be added, since it also increases the total bandwidth and only provides a limited

[1] This is a quick solution to bypass the problem, while waiting for the solution suggested hereafter. A major limitation of doing it is the risk of not receiving the base video layer at all since all the frames of the base layer are managed atomically.



(a) Reception rate with the classical approach



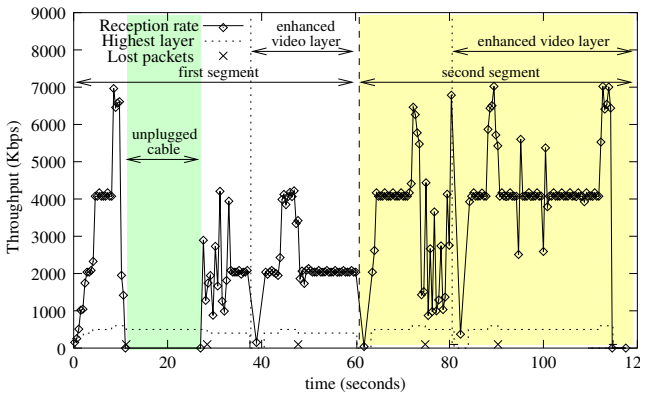
(b) Frame rate with the classical approach

Figure 19: Behavior with a 15 second loss burst and the classical approach.

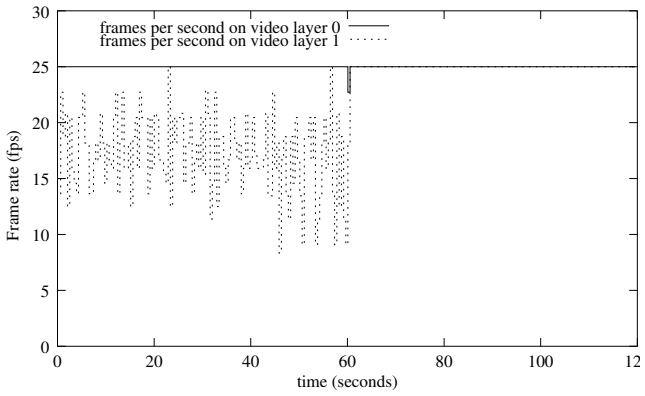
recovery capability? All solutions requiring some feedback information from the receivers to adjust the amount of proactive FEC create scalability problems. Besides the ideal amount is not the same for all clients, all the time, and defining several parallel streams is then unavoidable. This is why we chose to only consider the most elementary scheme for our comparisons.

The experiments show that SVSoA behaves well in a congested environment, no matter how many layers the video codec produces. Congestion control enables to adapt to the available bandwidth, and ensures to each client the reception of the minimum video quality made possible by its access network. Because of the natural buffering capability and the file transfer mode of SVSoA, packet losses, even in case of long bursts, do not automatically lead to frame losses and sudden video quality changes. They only increase the time spent receiving a given layer, and thus reduces the probability of receiving the enhancement information. Because of the high randomness of transmissions introduced in an ALC session, the video quality is almost constant during each VSD period.

Finally the SVSoA performances are limited by some sub-optimal solutions in the ALC implementation used. More recent congestion control algorithm, like WEBRC, more efficient FEC large block codes, and the cross-object FEC encoding (section 4.2.3) would largely improve the performances by reducing the reception inefficiency. We are currently working on these aspects and are confident on the possible improvements.



(a) Reception rate with SVSoA



(b) Frame rate with SVSoA

Figure 20: Behavior with a 15 second loss burst and SVSoA.

5 Conclusions

This paper introduces a novel multicast streaming solution, SVSoA, for hierarchically encoded videos. Our solution is in fact mid-way between reliable multicast file transfer and streaming. It directly benefits from the ALC reliable multicast protocol assets in terms of unlimited scalability, congestion control (TCP-friendly behavior), and error recovery (packet bursts are easily recovered). Thanks to ALC, our approach addresses the client heterogeneity issue and lets each of them experience the best possible display made possible by their access network. Our approach limits video quality instability by smoothing the effects of packet losses over periods of one minute. Finally, the solution is immediately deployable since it does not assume the presence of any specific service (like QoS support) within the network, and is compatible with any hierarchical video encoding scheme, no matter whether it offers a fine granularity or not.
This paper explains how to initialize the various parameters of SVSoA, what are the associated trade-offs, and provides answers to several issues like the IGMP leave latency or the impacts of the slow startup phase of the congestion control protocol. We implemented our approach and carried out a set of experiments on a local testbed. Results shows that SVSoA behaves appropriately, in line with the theory.
Future work will consist in reducing the reception inefficiency of the underlying ALC implementation, in order to be better compete with other streaming solutions.

References

[1] *MPEG4IP project home page*. http://sourceforge.net/projects/mpeg4ip/.

[2] S. Bajaj, L. Breslau, and S. Shenker. Uniform versus priority dropping for layered video. In *ACM SIGCOMM'98*, Sept. 1998.

[3] J. Bolot, T. Turletti, and I. Wakeman. Scalable feedback control for multicast video distribution in the internet. In *ACM SIGCOMM'94*, Oct. 1994.

[4] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, and W. Shaver. Flid-dl: Congestion control for layered multicast. In *2nd Workshop on Networked Group Communication (NGC2000)*, Nov. 2000.

[5] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *ACM SIGCOMM'98*, Aug. 1998.

[6] S. Cheung, M. Ammar, and X. Li. On the use of destination set grouping to improve fariness in multicast video distribution. In *IEEE INFOCOM'96*, Mar. 1996.

[7] D. Clark and D. Tennenhouse. Architectural considerations for a new generation of protocols. In *IEEE SIGCOMM'90*, Sept. 1990.

[8] B. Li and J. Liu. Multirate video multicast over the internet: an overview. 17(1), Jan. 2003. IEEE Network, "Multicasting: an enabling technology".

[9] W. Li. Overview of fine granularity scalability in mpeg-4 video standard. 11(3), Mar. 2001. IEEE Transaction on Circuits and Systems for Video Technology.

[10] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, and J. Crowcroft. *Asynchronous Layered Coding (ALC) protocol instantiation*, Dec. 2002. Request for Comment 3450.

[11] M. Luby and V. Goyal. *Wave and Equation Based Rate Control Building Block*, Dec. 2002. Work in Progress: <draft-ietf-rmt-bb-webrc-04.txt>.

[12] M. Luby, V. Goyal, S. Skaria, and G. Horn. Wave and equation based rate control using multicast round trip time. In *ACM SIGCOMM'02*, Aug. 2002.

[13] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft. *The use of Forward Error Correction (FEC) in reliable multicast*, Dec. 2002. Request for Comments 3453.

[14] M. Luby, L. Vicisano, and A. Haken. *Reliable multicast transport building block: Layered Congestion Control*, Nov. 2000. Work in Progress: <draft-ietf-rmt-bb-lcc-00.txt>.

[15] J. O. M. Nilsson, D. Dalby. Layered audiovisual coding for multicast distribution on ip networks. In *6th IEEE, European Workshop on Distributed Imaging*, Nov. 1999.

[16] H. N. Feamster, D. Bansal. On the interactions between layered quality adaption and congestion control for streaming video. In *12th International Packet Video Workshop, Pittsburgh, PA*, Apr. 2002.

[17] D. E. R. Rejaie, M. Handley. Layered quality adaption for internet video streaming. In *IEEE Journal on selected areas of communications (JSAC)*, 2000.

[18] R.Koenen. *Overview of the MPEG-4 Standard*, Mar. 2001. ISO/IEC JTC1/SC29/WG11 N4030.

[19] V. Roca. *MCL project home page*. http://www.inrialpes.fr/planete/people/roca/mcl/.

[20] V. Roca and B. Mordelet. Improving the efficiency of a multicast file transfer tool based on alc. In *IEEE International Symposium on Computer Communications (ISCC'02)*, July 2002.

[21] L. Vicisano, L. Rizzo, and J. Crowcroft. Tcp-like congestion control for layered multicast data transfer. In *IEEE INFOCOM'98*, Feb. 1998.

[22] B. Vickers, C. Albuquerque, and T. Suda. Source adaptive multi-layered multicast algorithms for real-time video distribution. 8(6), Dec. 2000. IEEE Transaction on Networking.

[23] F. Wu, S. Li, and Y. Zhang. A framework for efficient progressive fine granularity scalable video coding. 11(3), Mar. 2001. IEEE Transaction on Circuits and Systems for Video Technology.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399