A Framework for Delayed Real-Time Services over GEO Satellite Distribution Networks

Vincenzo Mancuso, *Member, IEEE*, and Antonio Panto'

***Abstract*— This paper tackles an interesting family of services, the delayed real-time services, and aims at defining a concrete framework of the system architecture at network, transport and application layers. Our goal is to support such kind of services with given QoS, security and robustness requirements. This work is particularly suitable for, but not limited to, GEO satellite distribution systems and terrestrial mobile users.**

A delayed real-time service exploits the possibility of introducing a buffer at receiver-side, which can also be implemented via a specific proxy that manages several users. This allows applications to decouple the physical download of real-time streaming from the actual play-out. It also allows mobile terminals to manage the retransmission of lost/corrupted packets, in order to mitigate the effect of shadowing due to blockage or multipath and unpredictable interference.

In order to cope with bandwidth optimization, we propose to adopt a multicast protocol that collects user's requests and provides users with special "multicast recovery connections", i.e. cumulative segment retransmissions that avoid the duplication of overlapping requests. The concurrency of multiple recovery connections over a shared bandwidth is handled by implementing AIMD (Additive Increase Multiplicative Decrease) in the multicast transport, and by regulating AIMD parameters through a policy function that operates a cross-layer optimization between application and transport layers, also taking into account the nature (and resource availability) of the underlying network.

I. INTRODUCTION

MOST of services exploiting GEO satellite were primarily meant to support long-distance data delivery for passive receivers. Real-time services can be included in that category, even though the possibility to access on-demand channels has become real and satellite return channels have been made largely available with DVB-RCS technology [1][2]. The advent of bidirectional satellite links is fundamental since the interaction of sender and receivers can be directly exploited in order to guarantee a high degree of QoS and a strong robustness with respect to data integrity and security.

In this paper we aim at defining a concrete framework and an appropriate GEO satellite system architecture with reference to interactive real time multicasting systems. The satellite return channel is proposed for transmissions requests that can be of interest for multiple users, i.e. a multicast group. Thus, we propose to exploit applicative protocols for broadcast/multicast of real-time transmissions, and we define a new procedure for the interaction of multicast users and the satellite diffusion point. It is worth noting that the proposed solution can use also other types of return channel, if the DVB-RCS is disadvantageous depending on the specific scenario considered.

According to the spirit proposed by the IETF Working Group on "Reliable Multicast Transport" [3], the discussion is not centered on a general robust multicast solution, but here we discuss on the robustness of the multicast protocols for a given class of applications, i.e. the class of multimedia real time applications, able to supply great amount of data for multiple wideband users. Thus we build up a particular framework and propose how particular building blocks could be adopted to properly design the system (see [4] for an analogous approach).

The paper is structured as follows. Section II introduces the concept of delayed real time services and the possibility to use a GEO satellite network to distribute contents. Section III introduces an innovative scenario for supporting delayed real time services over GEO satellite for multicast groups. Section IV describes a reliable multicast procedure – which exploits AIMD transport facilities - for the scenario of section III. System design principles and preliminary results are presented in section V. Eventually, section VI concludes the paper.

II. DELAYED REAL-TIME SERVICES OVER GEO SATELLITE

This paper aims at describing the support for an interesting family of interactive real-time satellite services: the delayed real time services (DRT). Within this class of services, real-time streams are buffered by the receiver or by a proxy before being distributed to the application that plays the stream. DRT services have been preliminary addressed in [5-7], jointly with multimedia on-demand services, and considering a specialized interactive distribution network scenario. In those works, authors investigated a fluidic model for the distribution system accessed by mobile vehicles, both in the case of GEO and wireless terrestrial networks. However, the effect of the actual transport protocol adopted in the communication is not taken into account.

Conversely, we here specifically deal with the transport protocol optimization. Furthermore, we propose a cross-layer approach to allow recovery procedures and transport

This work has been co-funded by the European Commission under the FP6 IST Program, in the framework of the SatNEx Network of Excellence.

V. Mancuso was with the Department of Electronic Engineering, Universita' di Roma "Tor Vergata", Rome, Italy. He is now is with the Department of Electric, Electronic and Telecommunication Engineering, Universita' of Palermo, Italy (e-mail: vincenzo.mancuso@ieee.org).

A. Panto' was with CNIT Research Unit of Catania, University of Catania, Italy (e-mail: apanto@diit.unict.it).

protocols to cooperate on the basis of the available resources and the knowledge of the priority, or better the *hurry*, of each retransmission.

In a DRT service, the hurry of a retransmission request is strongly correlated to the length of the playout buffer inserted at the end of the transmission chain. In fact, when some data is lost, a recovery procedure can be started using additional resources, if any is available, while delivering delayed data to the user's application. In any case, the recovery system has to react within a time shorter than the playout delay. It is worth noting that multiple retransmissions could be requested at the same time, and different retransmissions could partially overlap. Thus, in order to avoid a waste of bandwidth for multiple retransmissions of the same object, multicast should be used. Summarizing, a legacy satellite broadcast service should be endowed with:

i) playout buffers (possibly managed by a proxy function) that decouple the reception of data from the delivering to the final user's application;

ii) a multicast recovery protocol with a policy function, able to manage available resources and to mitigate the impact of network/satellite disruptions, as in the case of link failures due to the user mobility and shadowing effects.

III. THE REFERENCE SCENARIO

The general scenario is represented in figure 1. It is possible to recognize the elements listed below.

- The **Content Providers**: these are the primary sources for video applications, i.e. they generate the real-time data to be accessed via satellite.
- The **Recovery Service Manager (RSM)**: it consists of a streaming proxy, that has access to satellite resources and manages the retransmissions priority.
- The **User**: it is a DRT customer located behind the satellite link (or, equivalently, a proxy in a fixed or mobile subnet where at least a DRT customer is present). Actually, users can be logically grouped in accordance with the selected stream.

Every content provider sends a multimedia stream over the satellite link using a guaranteed bandwidth. According to the figure 1, there are N content providers and, therefore, N statically allocated channels, considering that the i^{th} group enjoys the multimedia stream sent by the i^{th} content provider.

A group receives the multimedia stream delivered at a constant speed, but for retransmission attempts which use as fast as possible speed[1]. Data is used by the streaming application at user-side after a playout delay (say D_k seconds for the k^{th} stream). Thus, each receiver needs a local proxy buffer to store at most D_k seconds of streaming data. This playout buffer that empties at constant rate and fills at variable rate, permits to continue the playout during the satellite channel outage, as long as sufficient information has been previously stored in the buffer.

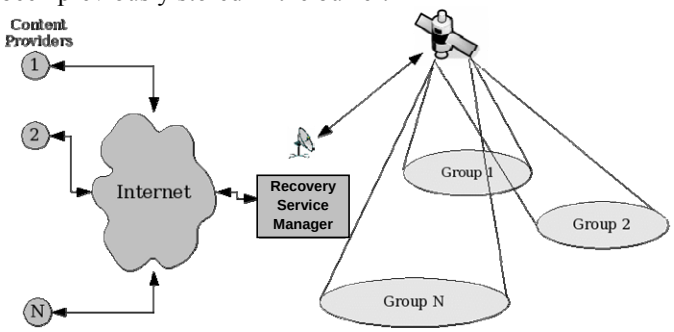

Figure 1 - Delayed Real-Time Services Architecture

When a channel outage happens, i.e. a specific satellite-user link fails, the receiver leaves a blank space in the application buffer and, upon the channel re-establishes, sends a retransmission request to the RSM. All retransmissions will share the same channel, say the $(N+1)^{th}$ connection in a DVB-RCS downlink [2][2].

A policy function runs on RSM to manage recovery procedures based on a metric quantifying the importance of a data segment for each requesting customer. Note that, due to the adoption of distinct playout delays and to the random nature of data loss, a simple FIFO or LIFO scheduling of retransmissions is far to be optimal in terms of fairness, throughput and user's satisfaction degree. For the same reason, a fair bandwidth distribution (i.e. proportional to the demand) cannot meet the requirement of fairness in the QoS experienced by user applications.

IV. THE RECOVERY PROCEDURE

A. The Recovery Service Manager (RSM)

The recovery procedures are carried out by the Recovery Service Manager (RSM) that manages the available bandwidth resources according to the whole set of the current retransmission requests. To this end, the RMS uses an adaptive algorithm in order to give more bandwidth to more urgent requests and locally evaluates those requests that cannot be fulfilled. The RSM has also a complete knowledge of all existing receivers and the way they are grouped.

B. The Multicast Transport Protocol

The retransmission requests are sent by the receivers without any distributed procedure, even if, when produced by receivers of the same group, they are likely to be correlated. When one or more users request retransmissions of overlapping data segments, the RSM creates a multicast group and sends all requested data only once. If an additional user requests the retransmission of part of those data, the RSM simply adds that user to the multicast retransmission group, and possibly includes new data segments in the section. In the described scenario we can

[1] For sake of simplicity, here we consider CBR streams. However, the analysis can be easily extended to the more general case of VBR sources by distinguishing the "real-time" rate of the source from the "recovery" rate.

[2] WiFi extensions with wireless multi-hop [8] and WiMax [9] could be suitable solutions due to their broadcast capabilities over large areas.

observe that the multicast solution fulfills the system requirements better than the unicast one.

A lot of TCP-like congestion control schemes for multicast application have been studied and proposed, e.g., [10-15]. In this paper we have chosen a single rate congestion control scheme. The two most interesting schemes are PGMCC, [10-11], MTCP [13], proposed for a generic network, and TCP-Peachtree [13] that has been specifically designed for satellite networks [13].

The primary objective of the adopted multicast transport protocol is providing high performance in the particular satellite scenario while featuring design and implementation simplicity. The requirements for such multicast protocol include congestion control, scalability, robustness and security. Moreover, upon request, the multicast transport protocol should provide packet ordering and a certain grade of reliability. In our work we focus on the major challenges related to reliability and congestion control issues.

The proposed multicast transport protocol uses a single-rate, window-based congestion control scheme. We adopt a new congestion control scheme defined in [16], for satellite networks that is based on the TCP-Peach approach (see [17-18]), and therefore on the use of *dummy segments*. The sender uses the dummy segments to probe the availability of network resources and to reduce the time interval required to recover from packet losses. It is worth noting that this paper uses the same algorithm adopted in [16] but for a different reason: if in [16] the TCP-Peach approach is needed to avoid congestion events, in the framework here proposed the same approach allows a faster and better utilization of the available bandwidth.

Adopting the AIMD values of a classic TCP window-based congestion control scheme both the reliable and the unreliable protocol proposed in [16] have a TCP-friendly behaviour. Starting from this result we propose to calibrate the congestion control algorithm to obtain the controlled intra-protocol unfriendliness that we need to fulfil the different time constraints of the different multicast flows sharing the satellite channel.

C. *The TCP unfriendly AIMD scheme*

The stability and the robustness of the Internet are due to the TCP congestion control mechanism, but, if the TCP fits well with application as data transfers, a real-time application would find halving the sending rate of a flow too drastic as the solution for a congestion event. As described in [19] and [20], the different requirements of different types of Internet applications need transport protocols with flexible congestion control schemes. In window-based congestion control schemes, the transmission rate is increased to probe available bandwidth, and decreased to react to a congestion event. An application is said to be *TCP-friendly* if it shares the bandwidth with a concurrent TCP connection fairly. Accordingly we can classify applications as *TCP-friendly* and *non TCP-friendly*. However, in a shared network such as the Internet all traffic flows - reliable and unreliable as well as unicast and multicast - are expected to be *TCP-friendly* [27].

It is worth noting that fairness is only one of several desirable properties of the considered congestion control scheme. Other characteristics to be considered are: the responsiveness, when there is an event of network congestion and aggressiveness, when there is a step increase of available bandwidth. Several AIMD congestion control schemes have been studied and proposed (e.g., [21], [22], and [23]), to support applications in hybrid wired/wireless networks. In our work we introduce a novel AIMD scheme optimized for reliable multicast connections over satellite networks. By run-time monitoring the performance of AIMD-controlled flows over a shared channel of fixed bandwidth, we propose a cross-flow procedure to select the AIMD protocol parameters considering the hurry of each flow as its main requirement.

A number of studies dealing with AIMD scheme customized for particular scenarios or targeted toward a specific type of application have been conducted over the past years, i.e. [24-26]. We consider the most general one, [24], that simply considers the increase value α and the decrease ratio β as parameters, thus in response to a single acknowledgement the AIMD increases the congestion window, W, in segments as follows:

$$W = W + \alpha / W \qquad \textbf{(1)}$$

In response to a congestion event, the AIMD decreases W as follows:

$$W = W - \beta W \qquad \textbf{(2)}$$

In the TCP scheme α = 1 and β = ½. As in [24] we refer to the general window adjustment strategy as General Additive Increase Multiplicative Decrease (GAIMD). In [24], a formula is shown that correlates the GAIMD mean sending rate with the control parameters, α and β, the loss rate p, the mean round-trip time RTT, the mean timeout value T_0 and the number of packets each ACK acknowledges b:

$$T_{\alpha,\beta}(p, RTT, T_0, b) = \frac{1}{TD_{\alpha,\beta} + TO_{\alpha,\beta}} \qquad \textbf{(3)}$$

where:

$$TD_{\alpha,\beta} = RTT \sqrt{\frac{2b(1-\beta)}{\alpha(1+\beta)}\, p} \qquad \textbf{(4)}$$

$$TO_{\alpha,\beta} = T_0 \min\left(1, 3\sqrt{\frac{(1-\beta^2)b}{2\alpha}\, p}\right) p(1+32p^2) \qquad \textbf{(5)}$$

These formulas are very important for our purposes, together with another two formulas that point out the relationship between α and β and the flow general stability and fairness:

$$\begin{aligned} 0 &< \alpha \\ 0 &< \beta < 1 \end{aligned} \qquad \textbf{(6)}$$

and its TCP-friendliness:

$$\alpha = \frac{4(1-\beta^2)}{3} \qquad (7)$$

The proposed AIMD algorithm is designed in order to have a manageable TCP unfriendliness, that is, every flow that uses the shared channel must use a bandwidth proportional to its hurry, that is computed by the RSM and on which depends the run-time values of AIMD parameters of the single flow. Moreover the AIMD parameters of each flow are calculated considering the current ranking of the hurry of each retransmission request that run-time varies depending on the system conditions.

AIMD parameters are computed as qualitatively described in what follows. After receiving a retransmission request, the RSM, which acts like a proxy for on-demand services, classifies the request according to the run-time estimated hurry of the request. The hurry of the retransmission is calculated from the requested information and the time available for recovery purposes, and can be stored in an array **p(t)** containing all hurry indicators for active retransmissions at time **t**. To this aim, every user communicates a time interval and two time values conveyed by the retransmission request:

$$\{\Delta t, t_0, t_1\}$$

where Δt the gap between packet timestamps after an outage event, (i.e. the length of the missing part of the stream, in terms of playout seconds), t_0 is the expected timestamp of the first missing packet, and t_1 is the time when the first missing packet should be used by the multimedia player (i.e. t_0 plus the specific playout delay **D**). Note that as for each quantity, related to the k^{th} recovery procedure, the index k should be added in the notation, but here we prefer to omit it for the sake of compactness. The RSM assigns a proper bandwidth to each retransmission, which is calculated from the corresponding hurry. The optimization of the policy that determines the hurry of a request is an objective of the ongoing research activity, however it will be based on:

1. the difference $(t_1 - t_{current})$, which is the time available before the first missing packet will be required;
2. the interval Δt.

Using both parameters it is also possible to compute the minimum required amount of bandwidth to be allotted to the retransmission, given the number S_{lost} of bits lost in the interval $[t_o, t_o+\Delta t]$ at the broadcast rate, which is known at the RSM. Thus, the minimum required amount of bandwidth to be allotted to the a retransmission is:

$$r_{\min} = \frac{S_{lost}}{t_1 + \Delta t - t_{current}} \qquad (8)$$

and represents the rate that should be adopted to complete the retransmission exactly at the end of the playout delay of the last lost bit. In equation (8), the delay due to signalling and elaboration is taken into account by subtracting the actual starting time of the recovery procedure, i.e. $t_{current}$, given that the retransmission begins at time $t_0+\Delta t < t_{current} < t_1+\Delta t$. it is worth noting that if the bust of lost packets exceeds the duration of the playout delay, i.e. $\Delta t > D = t_1 - t_0$, (i.e. $t_{current} > t_1$) it is only possible to recover at most **D** seconds, so that the RSM has to translate the request $\{\Delta t, t_0, t_1\}$ into $\{D, t_{current}-D, 0\}$, and $\Delta t - D$ seconds are surely lost.

D. Updating of bandwidth allocation

Bandwidth used is allotted to different retransmissions procedures in accordance with a given hurry-based policy. Whatever the chosen policy, note that the hurry of a retransmission will change during the retransmission itself, so that bandwidth assignments have to be dynamically adjusted. However, for the sake of simplicity, it is reasonable that the RSM updates each value of hurry indicators only after some significant events, let's say at time $t=t_k$, $k=0,1,\ldots$. Thus we can denote with **p(t)** the array of hurry indicators (or priorities) at time t, and with $q(k)=p(t_k)$ the array hurry indicators to be used in the time interval $[t_k;t_{k+1}[$. Instants t_k are taken in correspondence to the following events:

- a segment is requested which has not been yet retransmitted (outage event at user side: the playout buffer completely empties);
- a blank space is fulfilled (recovery completed);
- a new retransmission request arrives;
- two or more components of **p(t)** equals (e.g., for different values of **i** and **j**: $q_i(k-1)<q_j(k-1)$ but $p_i(t_k)=p_j(t_k)$) .

Last point in the above list is particularly important since it is related to time instants at which the order of retransmission priorities changes. So the RMS has to reschedule bandwidth allotting on the basis of a fluidic model for the array **p(t)** that accounts for the dynamic of all retransmission priorities.

Once the hurry indicators have been updated, the RMS should immediately change the amount of allotted bandwidth to each active retransmission, in accordance with the new array values **q(k+1)**.

E. Controlling AIMD bandwidth

At time t_k, when a new retransmission request occurs or an update of hurry indicators is performed, the bandwidth allotted for each of **N** active AIMD sessions is computed as a function of the **q(k)** array, and of $r_{i,\,min}$ values, **i = 1, 2, .., N**, computed at time $t_{current} = t_k$. Thus, the following formula can generically represent the bandwidth allotting process:

$$B_{i,AIMD}(\alpha, \beta) = B_{i,RES}(q(k), R_{min}(t_k)) \qquad (9)$$

for **i = 1, 2, .. , N**

where $R_{min}(t_k) = [r_{1,\,min}(t_k), r_{2,\,min}(t_k), \ldots, r_{N,\,min}(t_k)]^T$.

Omitting the index i, $B_{RES}(.)$ is the residual bandwidth function, i.e. a scalar function that takes into account the urgency of a request, plus some redundancy in the bandwidth allotting process and all other active

retransmissions. Depending on the estimated urgency of each request, $\mathbf{B_{RES}}(.)$ returns the bandwidth an AIMD transmitter should occupy without interfering with other retransmissions. Finally, $\mathbf{B_{AIMD}}(.)$ is the AIMD bandwidth function, i.e. a function of AIMD parameters α and β, which is a parametric expression, as shown in equations (3) to (5).

V. System design considerations and preliminary results

The main goal of the framework proposed in this paper is to find a trade-off between costs and QoS experienced by customers. To this aim, the system has to be designed to be robust to long lasting shadowing occurrences while minimizing the resources to be used by the RSM for retransmission. This is possible once the behaviour of the satellite link is statistically defined/estimated for any mobile user, and playout buffers at user side are fixed. Alternatively, given the maximum affordable recovery bandwidth, QoS and robustness can be optimized by selecting appropriate values for the length of playout buffers.

Possibly, the RSM might also act on the codec to be adopted for retransmissions, trying to accommodate multiple requests on the same channel. In any case, once the codec has been selected for a retransmission, the amount $\mathbf{S_{lost}}$ of data to be sent is determined and replaced in the previous expression to compute $\mathbf{r_{min}}$, and so it is possible to compute the bandwidth that the system can allot to the new request and, in turn, AIMD parameters to be used to meet that bandwidth.

In the following we show some preliminary work that illustrates how the usage of simple unresponsive transport protocol cannot meet QoS and robustness requirements for DRT. In particular we address issues related to the choice of bandwidth allocation methods and the incidence of transport protocols.

As for the bandwidth allocation methods, three are the proposed policies to be tested in the RMS to manage retransmitting applications. The first is a simple one, and it was thought for basic operation at RSM side: it consists of allotting a bandwidth which is proportional to the demand (Scheme 1 - Bandwidth proportional). The second one is purely based on the priority of each request, computed as the inverse of the time interval available for the recovery of the first missing byte (Scheme 2 - Inverse residual time). In that scheme, the hurry of each request is fundamental and it has to be frequently adjusted in order to avoid bandwidth stealing effects due to not updated estimates. The third scheme is based on the priority as defined before, but the allotting is reserved to retransmission with minimum priority only (Scheme 3 – "All to min"). In that case no urge of frequent bandwidth update occurs, since all the bandwidth has to be allotted only when the priority order of retransmissions changes. Furthermore, Schemes 2 and 3 can be slightly modified by weighting the priority of each retransmission with the bandwidth demand (Modified Inverse residual time scheme, and Modified All to min scheme, respectively). For any scheme, retransmissions are stopped if requested data are not sent within a timeout (i.e. the time at which data should be actually available at the receiver side).

Using one specific policy or another has a different impact, depending on the statistical transmission conditions, the load of the network, the available resources for retransmissions, the behavior of mobile users. Here we modeled the link with a good-bad process with exponential mean permanence time for both good and bad states. Real time broadcasting applications are always on, with a fixed bandwidth usage. Also the bandwidth available for retransmission is fixed and guaranteed by the distribution systems, and the playout delay of each receiving application is the same for all users. Furthermore, we represent each multicast group with a single user that acts as the worst case user, so that the good-bad process actually refers to the time distribution of periods in which link failure occurs or not, for a entire multicast group. This assumption simplifies the simulative analysis while preserving the correctness of results; in fact, in our system, overlapping retransmission requests sum and turn into a single multicast retransmission. Finally, no codec adaptation was considered so far.

As for the transport protocol, we have tested UDP-like retransmissions, while the evaluation of TCP and AIMD-like protocols is ongoing. However, preliminary results obtained with UDP, justify the study of connected transport protocols to enhance the system performance.

As a reference, let us consider a scenario with N=10 Content Providers generating an aggregate of 20.0 Mbps (each Content Providers generates at a fixed but different rate of about 2.0 Mbps, to avoid synchronization effects), and a 6.0 Mbps bandwidth is guaranteed for recovery. The playout delay of users is 20.0 seconds, and the transport protocol is UDP. Upon the average duration of the bad state of each link has been set to 5.0 seconds, we obtain results depicted in figures 2 to 5, by changing the average duration of the good state and by collecting simulation results over 200000 seconds.

Figure 2 shows the aggregate amount of retransmitted data when the Scheme 1 is adopted. Curves are normalized to the aggregate number of bytes requested by users. The lower curve in the figure represents data retransmitted for retransmissions that the system was able to complete. It is clear that a great number of retransmissions is stopped due to lack of resources as soon as the link error probability exceeds 0.2. Furthermore, for error probability greater that 0.1, the number of unrecoverable bytes increases (due to outage periods longer that the playout delay, which are now more frequent). For the same scenario, figure 3 depicts the aggregate delivered data and the amount of data lost due to link failures during the retransmission procedure. Lost data are normalized to retransmitted data and not to requested data, to give a neat measure of the need of a connected transport protocol during the recovery procedure. Note that performance of the system is not satisfactory even with values of the link failure probability as small as 0.1, which is not so much for mobile users.

Figures 4 and 5 present a comparison between different policy schemes. Figure 4 draws the case of retransmitted data, and figure 5 deals with data sent for recovery purposes but lost due to link failures only. Qualitatively, those figures show that hurry-based policies outperform legacy bandwidth proportional allotting methods only for limited values of the link failure probability. However, priorities seem to be an important metric up to reasonable values of the probability error (i.e., lower that 0.2). The lack of performance of hurry-based policies with high error probability, is due to the fact that they modulate the bandwidth allotting in order to equate all priorities in the retransmission list, so that successes of recovery procedure are strongly correlated with each other; thus, when available resources are not enough (e.g., due to very high retransmission requests, outreaching the capacity of the recovery service), the number of successes drastically decline for all connections. As a reference, consider that in the scenario described by our simulations, the recovery bandwidth corresponds to an average recovery request of 6.0 Mbps, that is the traffic requested by retransmission when the link failure probability is 0.3. Unfortunately, the bursty nature of requests, makes the assigned, recovery bandwidth not sufficient for failure probabilities greater than 0.1. The performance of that system can be traded-off with the cost for additional bandwidth, but the example reported here gives an idea of the minimal amount of resources to be guaranteed.

However, in order to obtain better performance, it should be considered the possibility to recover also lost retransmitted data, and this could be done by means of additional retransmission procedures, or, better, by exploiting the capability of multicast AIMD applications.

VI. Conclusions and Open Issues

The paper shows an architecture for supporting delayed real time services in GEO distribution systems by means of multicast retransmission procedures. Retransmissions are necessary due to shadowing effects experienced by mobile users, and that can be modelled as a good-bad behaviour of the connectivity. Multicast retransmissions also require the availability of additional resources, that can be sensibly less that those required by a unicast retransmission system. This is particularly suited for satellite distribution systems with very broad coverage capabilities and higher multiplexing factor for user requests, if compared with terrestrial wireless solutions as WiMax or WiFi. Note that, in the described framework, the DVB-RCS is not the only suitable satellite technology, since a satellite download method with a narrow-band upload service (e.g., via terrestrial modem, or terrestrial wireless systems, where available) could be enough and, mostly important, cheaper.

The proposed architecture exploits the presence of a Recovery Server Manager which is in charge of managing the bandwidth reserved for retransmission purposes, and different bandwidth management schemes was illustrated. The syntax of retransmission requests has been defined, but system time synchronization issues should be further addressed or, more likely, differential time values should be used (e.g., by referring to existing timestamps in the multimedia real time stream). However, system performance strongly depends on the adopted transport protocol, and unresponsive (UDP-like) approaches do not seem to be robust enough to support service QoS and continuity. Thus, an AIMD-like scheme has been proposed and the interaction of RSM and transport protocol has been discussed. The evaluation of AIMD scheme is in progress.

References

[1] ETSI: EN 301 192 V1.4.1 (2004-11) Digital Video Broadcasting (DVB); DVB specification for data broadcasting.

[2] ETSI EN 301 790 DVB; Interaction Channel for Satellite Distribution Systems, v1.3.1.

[3] M. Handley, S. Floyd, B. Whetten, R. Kermode, L. Vicisano, M. Luby, RFC2887, The Reliable Multicast Design Space for Bulk Data Transfer, August 2000.

[4] B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, M. Luby, RFC3048, Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer, January 2001

[5] V. Mancuso, G. Bianchi, "Streaming for vehicular users via elastic proxy buffer management", IEEE Communication Magazine, Volume: 42 , Issue: 11 , pp. 144-152, November 2004.

[6] V. Mancuso, G. Bianchi, N. Blefari Melazzi "Streaming Support for Vehicular Networks Using Elastic Proxy Buffers", proceedings of IST Mobile Summit 2004, Lyon, France, June 2004.

[7] V.Mancuso, M.Gambardella, G.Bianchi, "Improved Support for Streaming Services in Vehicular Networks", proceedings of ICC'04 conference, Paris, France, June 2004.

[8] A. Raniwala and T. Chiueh. Architecture and Algorithms for an IEEE 802.11-based Multi-channel Wireless Mesh Network. In Proceedings of IEEE INFOCOM, April 2005.

[9] IEEE Std 802.16.2-2004, Coexistence of Fixed Broadband Wireless Access Systems.

[10] L. Rizzo. pgmcc: a TCP-friendly single-rate multicast congestion control scheme. *Proc. Of Sigcomm*'2000. August 2000.

[11] L. Rizzo, G. Iannaccone, L. Vicisano, and M. Handley. PGMCC Single Rate Multicast Congestion Control: Protocol Specification. *Internet Draft*: draft-ietf-rmt-bb-pgmcc-00.txt. February 2001.

[12] M. Hdley, S. Floyd, and B. Whetten. Strawman Specification for TCP Friendly (Reliable) Multicast Congestion Control. Technical Report – Reliable Multicast Research Group. December 1998.

[13] I. Rhee, N. Balaguru, and G. Rouskas. MTCP: Scalable TCP-Like Congestion Control for Reliable Multicast. Proc. Of IEEE Infocom'99. March 1999.

[14] G. Morabito and S. Palazzo. Modeling and Analysis of TCP-Like Multicast Congestion Control in Hybrid Terrestrial/Satellite IP Networks. *IEEE Journal on Selected Areas of Communications.* February 2004.

[15] F. Akyildiz and J. Fang. TCP-Peachtree: A Multicast Transport Protocol for Satellite IP Networks. *IEEE Journal of Selected Areas in Communications (JSAC)*, Vol. 22, Issue 2, pp. 388-400, February 2004.

[16] G. Morabito, S. Palazzo, A. Pantò. "A TCP-Friendly Multicast Protocol Suite for Satellite Networks" Proc. of 4th International Conference on Networking (ICN'05), Reunion Island, France, April 17-21, 2005.

[17] I. F. Akyildiz, G. Morabito, and S. Palazzo. TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks. IEEE/ACM Transactions on Networking. June 2001.

[18] G. Morabito, R. Narcisi, S. Palazzo, A. Pantò. TCP-Peach and FACK/SACK Options: Putting Pieces Together. IEEE Globecom 2002 – Satellite Communications Symposium, November 2002.

[19] Y. R. Yang, M. S. Kim, and S. S. Lam. Transient behavior of TCP-friendly congestion control protocols. In *Proceedings of IEEE INFOCOM*, April 2001.

[20] S. Jin, L. Guo, I. Matta, and A. Bestavros, "A spectrum of TCP-Friendly window-based congestion control algorithms," *IEEE/ACM Trans. Netw.*, vol. 11, no. 3, pp. 341-355, Jun. 2003.

[21] Sally Floyd. Highspeed TCP for large congestion windows. IETF, RFC 3649, December, 2003.
[22] Eitan Altman, Chadi Barakat, Victor Manuel Ramos Ramos: Analysis of AIMD protocols over paths with variable delay. INFOCOM 2004.
[23] L. Cai, X. Shen, J. W. Mark, and J. Pan, A QoS-aware AIMD protocol for time-sensitive applications in wireless/wired networks, Proc. IEEE Infocom'05, Miami, Florida, March 13-17, 2005.
[24] Y. R. Yang and S. S. Lam. General AIMD congestion control. University ofTexas, Tech. Rep. TR-2000-09, May 2000.
[25] L. Cai, X. Shen, J. W. Mark, and J. Pan, A QoS-aware AIMD protocol for time-sensitive applications in wireless/wired networks, *Proc. IEEE Infocom'05*, Miami, Florida, March 13--17, 2005.
[26] Eitan Altman, Chadi Barakat, Victor Manuel Ramos Ramos: Analysis of AIMD protocols over paths with variable delay. INFOCOM 2004.
[27] Sally Floyd, Kevin R. Fall: Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Transaction on Networking, Vol. 7, No. 4, pp. 458--472, 1999.

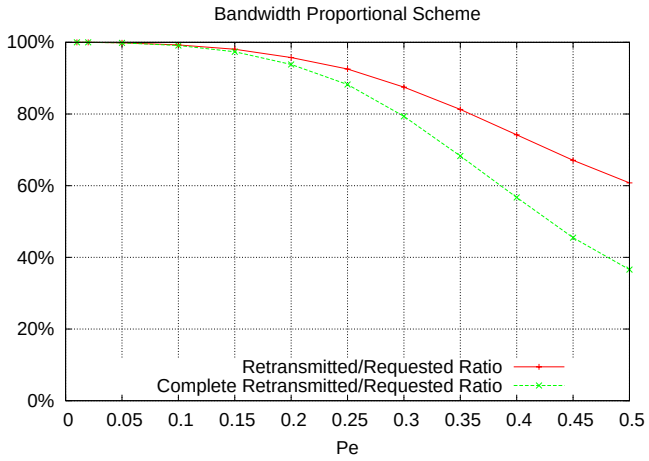

Figure 2 – Retransmitted data using Scheme 1

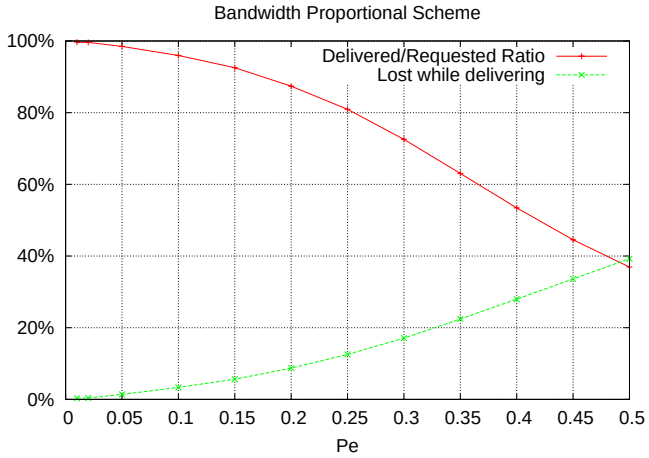

Figure 3 – Delivered and lost retransmitted data using Scheme 1

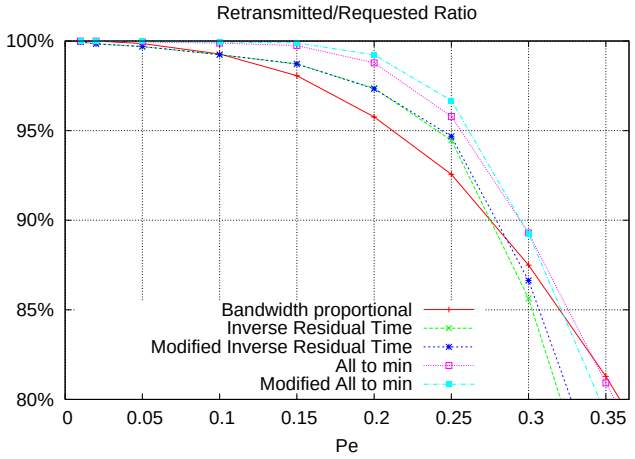

Figure 4 – Comparison of bandwidth allotting policies as to retransmitted data

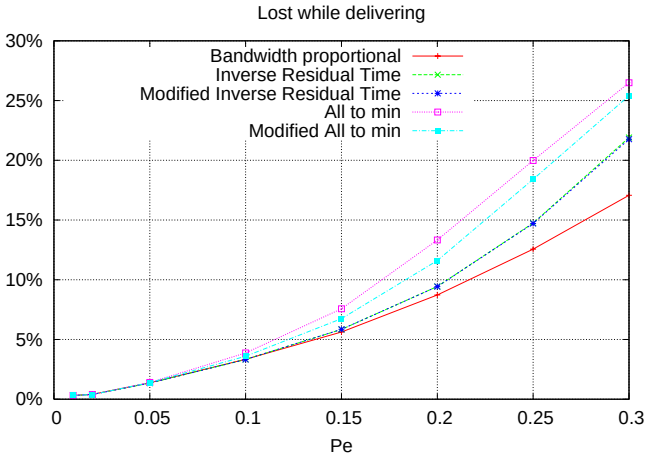

Figure 5 – Comparison of bandwidth allotting policies as to data lost in the retransmission procedure