LARGE-SCALE TERRESTRIAL RELAYING OF SATELLITE BROADCASTED REAL-TIME MULTIMEDIA STREAMS

Franco Tommasi and Catiuscia Melle

Department of Innovation Engineering, University of Salento, Lecce, Italy
franco.tommasi@unisalento.it
catiuscia.melle@unisalento.it

ABSTRACT

This paper describes an architecture to relay on demand a real-time IP multicast audio-video stream broadcasted by a satellite on a terrestrial link. The stream is received by suitably equipped sites and then relayed to other sites that are not equipped with satellite receiving hardware but are nonetheless willing to receive the stream. By exploiting the properties of satellite transmission and adopting a hybrid satellite/terrestrial, multicast/unicast approach, the described architecture allows to overcome the restrictions suffered by multicast traffic in the global Internet, allowing it to scale easily across autonomous systems. All things considered, the proposed architecture outlines a large-scale interactive audio-video distribution system similar to those based on Content Distribution Networks (CDN) and it compares favorably with them when performances, costs and scalability are examined.

KEYWORDS

Satellite Terrestrial Hybrid Systems, Multicast Unicast relaying, large-scale real-time video distribution on the Internet, CDN, Internet TV

1. INTRODUCTION

IP Multicast would be the ideal technique to distribute real-time video on the Internet but for many well-known reasons it is not globally supported on the Internet [1]. There are indeed many ISPs supporting IP multicast in their AS (Autonomous Systems) and multicast peering agreements are frequent among ISPs but even then the common user isn't left the faculty to send multicast traffic to other users in the same AS. Clearly this ability is regarded as a primary asset within an ISP network and acquiring it (when available) can be subjected to substantial fees. Many methods to overcome this limitation have been proposed [2][3][4] but none of them has proved very successful until now.

A natural way to transmit IP multicast over a large geographical area is satellite broadcasting. The cost of such medium used to be rather high in the past but has now become quite reasonable, especially when satellites different from those preferred by main video broadcasters are used. Unfortunately such satellites are not the ones most domestic parabolas are aimed at.

To overcome this problem, while still profiting from the satellite's power, a hybrid architecture is proposed in the following. Sites equipped to receive a satellite transmission become (main or primary) relayers of the multicast flows on the terrestrial Internet to the benefit of the sites that are not equipped with a satellite receiver. Also, this is done recursively, that is, the sites receiving the relayed flows will help to propagate them further (acting as secondary relayers). A central server is needed to keep track of who is getting each flow and who is available to relay it. The server will also help newcomers to locate the most convenient relayer to contact.

The proposed architecture has been inspired from the routine operations of an infrastructure called *Campus Satellitare del Salento*, CSS [5][6][7][8][9].

The CSS is a large-scale real-time interactive distance-learning system based on satellite. An essential part of the infrastructure is a good number of classrooms equipped to receive IP-based satellite transmissions. Frequent requests from sites willing to join the CSS for some particular event are received from the infrastructure's managers. Very often it is not possible to set them up to receive the streams by satellite just in time for the event. Also, requests are submitted for temporary receiving locations, where it makes no sense to install a permanent parabola. Sometime the request comes from someone who can't afford (or simply doesn't want to pay) the cost of the set-up, no matter how reasonable. All these routine situations have prompted us with a solution whose span could easily exceed the above reported needs and even become a feasible alternative to the lease of expensive CDNs (Content Distribution Network) [10][11], sharing some of the characteristics of P2P distribution networks while retaining an essential property when live contents are managed: a short time delay.

In Figure 1 we present the topology and the elements that constitute the background of the CHARMS architecture. CHARMS stands for "*Cooperative Hybrid Architecture for Relaying Multicast Satellite Streams over terrestrial links*", and its main purpose is the delivery of multicast real-time multimedia streams broadcasted by satellites to sites not equipped with a satellite receiver.

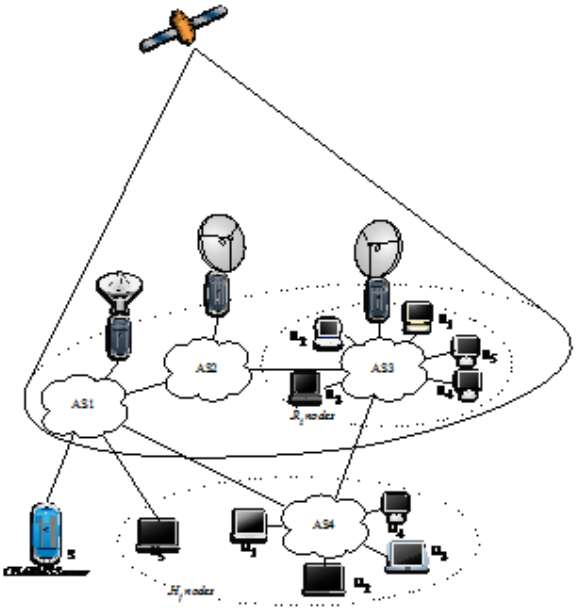


Figure 1. CHARMS architecture background

There are 3 types of CHARMS nodes: R, H and S. R_1, …, R_n are hosts located in sites equipped to receive a stream directly from the satellite. H_1, …, H_m are normal hosts at sites with no satellite receiving equipments, yet wishing to receive satellite broadcasted streams. S is a central server ("*the CHARMS Server*") whose main responsibility is matching requests from a host H_j with offers from hosts R_i. Out of the picture is a video server sending out an IP multicast audio-video stream to the satellite for transmission. The following is a description of the main parts and operations of the architecture.

An operation timeline is showed in Figure 2, where R_i and H_j, the generic CHARMS relayer and CHARMS host, are performing the preliminaryoperations needed for the start-up of the stream

relaying process. Only main activities are represented in the figure, in order to clarify the logic underlying the architecture.

This paper aims at describing the proposed architecture and at giving the first implementation details. A forthcoming work will provide measures and statistics about its actual utilization.

In the next section a number of related works are considered. Section 3 will present an overview and a description of the architecture. It also contains a discussion of how its main problems have been addressed and resolved. In particular, Section 3.2 will detail how R_i and H_j can exchange traffic in the frequent case they are both behind NAT boxes. Section 3.3 will discuss the way a particular relayer R_i is selected among others to match an incoming request from a host H. Section 3.4 will detail the way the multicast traffic each R_i receives from satellite is relayed on demand to some host H_j. Section 4 will summarize CHARMS architecture and Section 5 will draw conclusions and outline future work.

2. RELATED WORKS

Among the works published on the subject of Satellite-Terrestrial Hybrid Networks we believe the closest to our researches are [7-9][12][13][14][15][16][17][18][19][20][21]. They try to simultaneously exploit the benefits of the two types of networks (i.e. terrestrial and satellite) by focusing on different applications.

Reference [14] describes an architecture where a receiver can obtain data through a satellite link only after its request, sent through the terrestrial link, has been received by a satellite transmission enabled node, the *feed*. Reference [15] extends the previous solution for a multibeams-satellite equipped with on-board software. In this work, data traffic is forwarded only through pertinent set of beams (those covering the receivers that have sent an explicit request). Papers [16][17] describe an architecture aimed at minimizing the satellite/terrestrial links' costs. Data are forwarded through a satellite only if the number of expected receivers does justifies it, otherwise they are forwarded through the terrestrial network. Reference [18] proposes an architecture where satellites send data to aircrafts. The path of data will switch from the satellite link to a ground wireless link when the latter is available.

The focus of our research is set on large-scale distance learning applications. The intrinsic multicast/broadcast nature of satellite and its ability to cover wide areas are balanced by a few drawbacks such as technological complexity and the potentially high costs related to the lease of such medium and to the receiving locations set-up. A hybrid architecture can take advantage from both terrestrial and satellite telecommunication systems, optimizing the usage of the network resources: the satellite system is used to distribute the traffic to all the classrooms (this is a reference to our CSS scenario) or locations, by a multicast transmission, while the terrestrial Internet can be used to perform a unicast delivery of the traffic, thus improving the utilization of the available network resources. To an end-user is left the choice between the two delivery modes: terrestrial (cheaper) or satellite (more reliable).

Compared to CDNs, we have here a much better scalability and an easier and better synchronization of the distributed replicas. From an economical point of view, the advantage can be on the satellite's side, especially when the scale of the distribution grows and if satellites different from those preferred by main video broadcasters are used. We are not forgetting, of course, the cost overhead implied at the main relayers' sites to aim parabolas at such less popular satellites but it can be shown to be quite modest.

In our scenario, the ideal location for the primary relayer nodes is the AS domain to which belongs also the host nodes, that is those not equipped to receive streams directly from satellite. In such case, the traffic generated between the nodes of the CHARMS system doesn't cross the Internet core and it is confined within each AS. Obviously the more Autonomous Systems are served by a dedicated relayer the better the traffic is managed and the architecture performs.

Some correspondences to our work can be found in Zattoo Network [22]: there a standard broadcaster satellite channel is received by an encoder machine which relays it on the overlay after encoding. No use is made of a plurality of satellite receivers as in our case.

Also similarities can be found in an hybrid P2P-CDN architecture, LiveSky [23], which makes no use of the satellite for making the streams available to seeder nodes (that part is played by a CDN network), but still accomplish content distribution among peers by P2P.

3. ARCHITECTURE OVERVIEW AND DESCRIPTION

In our architecture we have a certain number of nodes R, that we call *primary relayers*, that are entities able to receive IP multicast real-time streams, directly from satellite. Our purpose is to use such capability in order to provide the same contents to another group of nodes H, that we call *hosts*, which are nodes able to receive only terrestrial Internet traffic, in unicast mode. Once the nodes H receive the streams, we want themselves to be able to relay the same streams to other hosts H and so on recursively. Once they are available to relay the streams, all the hosts H can be called *secondary relayers*.

In order to accomplish these tasks, a supervisor entity is needed: the central server S. S has the main responsibility to help nodes, relayers and hosts, to get in touch with each other, to meet reciprocal requirements and to support dynamic changes in the overlay.

3.1. Server S and communication between S and CHARMS nodes

The server S, after a setup phase, is able to receive communication messages from all the nodes in the CHARMS architecture. Both relayers and hosts must perform an initial activity in order to inform S about their state and operating context. A user at the host H_j can visit a web page, hosted on a Web server dedicated to the presentation of the real-time events, to get the list of the available transmissions at a given time and choose among them a stream he wants to receive. On the other side, whenever an R_i host receives a satellite stream, it will inform the server S, in order to allow it to update the list of the available streams and that of the relayers available for each of them.

The communication among the three entities H_j, R_i and S consists of three main elements:

1. as soon as a relayer R_i starts receiving a stream, it may perform a registration with the server sending to it information about itself and the stream and about its availability to relay the stream;

2. when a host H_j requests a stream to the server, the server will perform the operations needed to select a particular relayer R_i among the available ones and to allow the forwarding of the stream from R_i to H_j (see next sections for details about the relaying);

3. the server S will perform continuous monitoring of the states of the involved entities in order to keep the stored information about them consistent and to take proper actions when changes are detected in the state of the hosts and in that of the network.

Figure 3 shows the steady state of the delivery system. The chance for H nodes to act as secondary relayers is also shown.

3.2. Nodes Visibility and Communication Setup

It is very common for one or both of the hosts involved in the relaying of a stream (i.e. R_i and H_j) to be behind a NAT box. The presence of NATs in a given network topology is a critical factor when deploying a peer-to-peer application [24] and it requires a careful analysis before defining the architecture operations. In the following we'll propose a solution to the establishment of a direct contact between peers in presence of NATs. We introduce a first level

of abstraction, considering every NAT like a "black-box": we don't know how each NAT works, and clients aren't aware of a NAT being performed. This is a common approach, since NAT boxes' behaviour may change over time, depending on the traffic handled, conflicts inside a local domain, etc. [25][26].

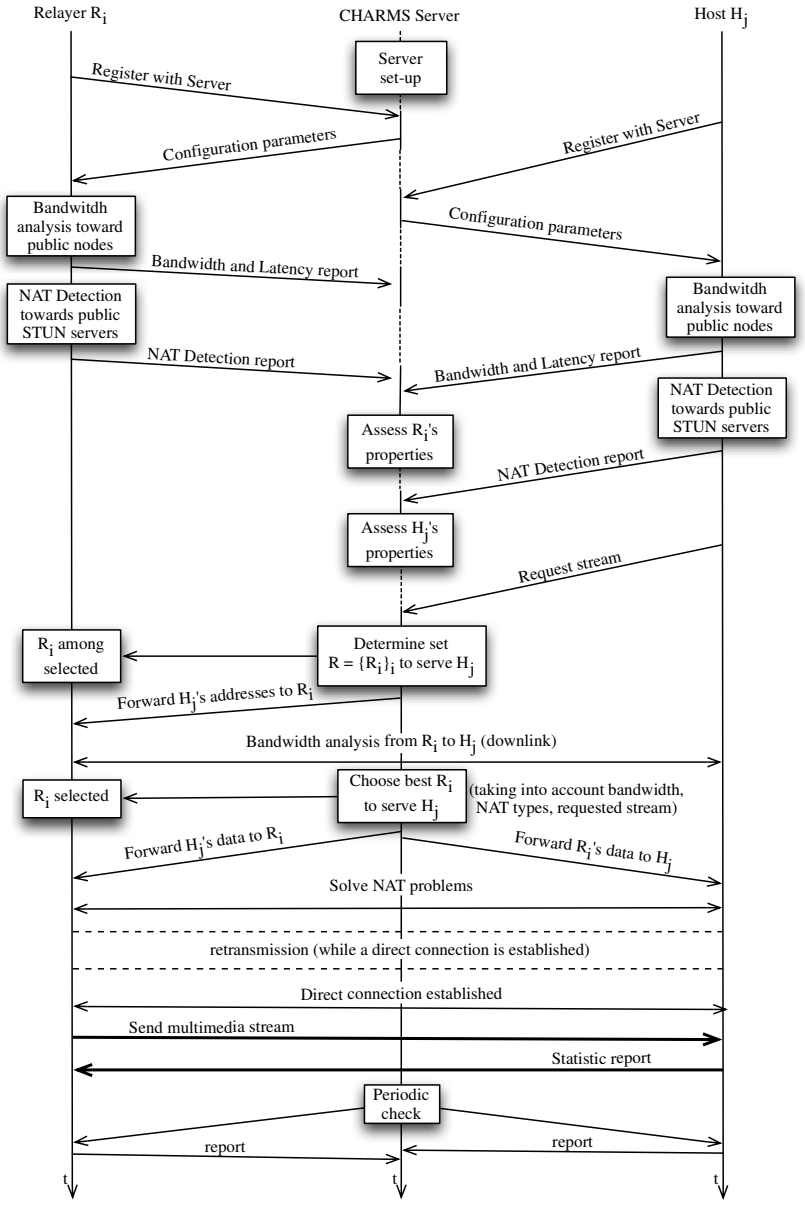


Figure 2. Overview of interactions timeline in CHARMS architecture

Thanks to the STUN protocol [27][28], a dynamic discovery of the type of NAT crossed by a particular host can be performed [29]. The mapping and filtering behaviours [30] can be modelled. These are the two most important features affecting the possibility of achieving a direct end-to-end communication between peers. In the following we will generalize, by assuming that every end-system is behind a NAT: in the case of a host with a normal public IP address, its NAT will be said of an "Open" type [27][29].

Various solutions have been proposed and implemented in the past for allowing peer-to-peer applications to work in presence of NATs [24][31][32].

In the CHARMS architecture we focused our attention on the UDP Hole Punch technique [32] for its ability to achieve a direct end-to-end communication, without the aid of the relaying technique (i.e. relaying packets through a dedicated server) that will increase the latency of the packet delivery, like TURN [33]. Such type of relaying solution isn't suitable for our purposes, both for the latency added to the delivery of packets to peers and for the obvious bandwidth problems introduced: the relayer nodes in our architecture receive a real-time streaming and they have to forward it to the requesting hosts with the least possible delay. To avoid confusion of terms, in the following we stress the fact that in our architecture, relaying is not performed centrally by a server as in the above definition but it is shared among all the receivers R_i.

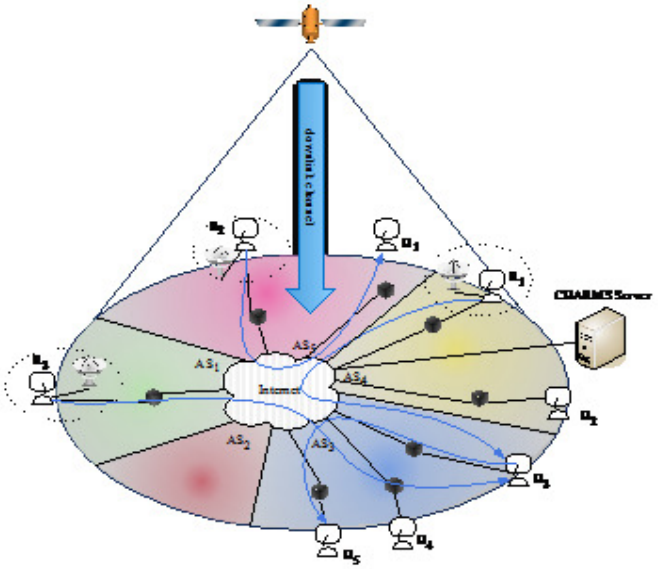


Figure 3. CHARMS architecture detail

While the Hole Punch works fine only in presence of NATs with an Endpoint Independent Mapping (EIM) [30], in our scenario, it is reasonable to hypothesize the presence of both kinds of NATs, i.e. EIM and Endpoint Dependent Mapping (or "symmetric NATs" as defined in [27]). In order to extend the use of the Hole-Punching technique to hosts behind symmetric NATs, we propose here to modify the role of the Rendezvous server (implemented within the CHARMS server), which is normally used to gather information about other peers in the network, with the introduction of a new feature: the ability to create pairs of hosts on the basis of their NATs behaviour. We implemented an UDP Hole Punch technique, based on the use of the STUN protocol, to set up direct communication channels between peers. For this reason, every host in the system has to discover the way its NAT box operates and to report the resulting policies (i.e. mapping and filtering behaviours) to the server, responsible for the execution of the matching function f_{NAT}, for every couple of hosts.

The main idea is when a host A, behind a NAT with an EIM policy tries to contact another peer, B, its public address will remain unchanged, regardless of the destination address. So, if peer B is behind another NAT with an EIM policy, the UDP Hole Punch technique works fine. But, if peer B is behind of what is known as symmetric NAT, host A doesn't know B's public address. So, B can send STUN Binding Requests to A, but A cannot. If A's NAT implements an

Endpoint Independent Filtering policy (EIF), A will receive the B's STUN Binding Requests: in this way A will learn the B's public address, and will also discover if they are on the same private network. So A too can start sending STUN Binding Requests to B, executing NAT Traversal operations based on STUN Binding Requests retransmissions, until all STUN Binding Responses are received.

Based on this idea, every node in the network that join the overlay has to cooperate with the server in order to support the decision process: a client must not only detect the model of its NAT box, but also collect information about private and public addresses and forward them to the server. With the use of a STUN client and a STUN server in every host, the various addresses gathered and announced by the server will be tested in the NAT Traversal process. Our Rendezvous server will manage a list of nodes and, for each node, it will collect the following items: stream received, network bandwidth capability, NAT type. When a new node H_j sends its data, the server has to choose from its list a node R_i possessing the requested stream and an adequate outgoing channel, and whose NAT box is compatible with that of H_j, from the point of view of success of the Hole Punch technique deployed, as mentioned above. In the initial phase, only relayer nodes will be known at the server. As soon as new clients, from terrestrial links, request a stream, the server will analyze the visibility of the clients, that is the characteristics of the NATs involved, before deciding the match.

This approach may look like the Connection Reversal technique [32], but it's different in that we relax the need of a peer with a public IP address, accepting to deal with hosts behind NATs having a fixed behaviour. Clearly, if such a kind of NAT is not present in the list managed by the server, and no peers with public IPs are available, the only way left for clients behind symmetric NATs to receive a stream is by server based relaying.

Our approach is different from that used in the ICE protocol [34]: we avoid both the need of a TURN server and that of nodes with a public IP address in the system. Our solution relies only on the matching logic implemented in the server and on the success of the NAT Traversal process between couples of nodes.

3.3. Nodes Identity and Uplink Estimation

When a host H_j sends a request to the server S for a given real-time media stream, the server must decide which node R_i is the most suitable to meet its demand, i.e. which R_i has the better uplink capacity needed to support the forward rates. This implies the need to estimate the available bandwidth on the link R_i-H_j [35] and for a careful analysis of the factors impacting the performances of interactive applications like the ones we are interested in, such as loss rate, delay and jitter.

In our scenario, it is reasonable to assume that H nodes might join and leave the overlay at any time, while R nodes are a more stable set of entities, along with the server. As a result, we can make the assumption that the operative context of an R node will not change very frequently, in particular we can assume the type of access of R to the Internet will not change over time (if R is not a mobile node). Under these assumptions, we consider it useful to identify each node in the architecture, not by using its Internet address, which is subject to change, but by the assignment of a *cookie* the first time the node joins the overlay: this value is unique within the domain. Thanks to this procedure, the server S can better manage the overlay and take advantage of prior results to improve its matching logics.

In order to decide which R node is the best candidate to relay the real-time stream to an H node and in view of the implementation of a load-sharing policy, the server S may find it useful to know not only the end-to-end bandwidth (from R to H), but also the outgoing bandwidth of R. In fact, S imposes an upper limit on the percentage of available outgoing bandwidth each R can commit to relaying. As a consequence, each candidate node R has to estimate and report to S, before any stream forwarding, two values:

- the total available outgoing bandwidth, to help estimate whether it has sufficient resources to proceed with the relaying of real-time stream to other nodes (such estimation may take place at start-up);

- the end-to-end bandwidth from each node R_i to a given destination H_j, to state which one is the best candidate to forward multimedia traffic nodes (such estimation has to take place after H_j's request).

The estimation of available bandwidth, along a given network path, can be performed by different algorithms which may be more or less invasive and require different processing times [35]. Because of constraints posed on the global latency in CHARMS, and not to generate excessive disturbance to other traffic, the candidate algorithms should be able to produce an acceptable approximation of requested results in a very short time. The potential inaccuracy of a fast estimate must be balanced by a periodic execution of the test, so to take into account the variability of network. From this point of view, the tools designed to produce results on a per-hop basis, like in [36][37], are too slow and intrusive, therefore not suitable for our architecture.

These considerations led us to the adoption of a SLOPS-like algorithm (Self Loading Periodic Streams) [38], in which a source (our client node) sends over UDP (the same transport used for the stream relaying) k packets of equal size m, at frequency r to a destination (a generic server node). The frequency r is increased until it saturates the available bandwidth, thus giving a measure of its value. Since in our system the amount of bandwidth needed to transmit an audio/video stream is typically around 500 kbps, the time needed for the test can be reduced if the measurement stream is increased in steps of the same size. These estimates are constantly updated to the current value, periodically running the algorithm that will take into account previous measurements, so to assess the extent of the differences between a measure and the previous one. Also the knowledge of the portion of bandwidth already used in active sessions is used to save time (by avoiding values of frequency r which are already known to cause congestions).

To perform this test between two endpoints, two modules are needed, one active in the R node and the other in the H node (or another fixed destination of the system), acting like a client and server respectively. In order to find the available outgoing bandwidth of a node R_i, regardless of the destination of output data, we make use of dedicated servers, S_B, connected to the Internet with links at high capacity, so that we can assume that any bottleneck or delay in transmission along the path R_i-S_B is to be ascribed to the network segment that connects R_i to the Internet: the result will be a lower bound for the value of available bandwidth of R_i's uplink.

After S has invited a set of R nodes to perform the test versus the H_j node, it waits for the results. All the collected measures are used (as specified below in the description of the matching logic of S) to designate the relayer R_{best} that will provide the stream to H_j. Once R_{best} is chosen, the transmission of the flow can start from it to H_j.

3.4. Relaying Streams from R_i to H_j

Let a relayer R_i receive a multicast stream generated from a Streaming Server and transmitted by the satellite. The description of the stream is provided through an SDP [39] file, which is sent beforehand to the relayer (or, alternatively, it is already owned by it). Once a particular relayer R_i is selected with the above procedure to satisfy the needs of a new request from a newcomer H_j, it must forward the received stream to H_j, via a terrestrial unicast IP transmission.

If there where no NATs in the overlay network, this task would be accomplished using the RTSP [40] protocol between R and H and a modified copy of the original SDP file possessed by R (the modification is needed because of the multicast/unicast variation). This modified SDP file will be sent from the relayer R_i to the host H_j, and it will contain all the information about the unicast multimedia sessions, obtained from the original stream. Because of excessive

complications arising when trying to utilize RTSP in conjunction with NATs, this strategy is abandoned.

Our solution is based on the relay of UDP traffic between R and H, over the direct channels tested in the NAT Traversal procedure.

A real-time multimedia stream is made up of sub-session streams, each belonging to a specific media and having a specific encoding. In our context, the multimedia stream incorporates audio and video sessions, which require 4 distinct UDP channels: RTP/RTCP [41] audio sub-sessions, and RTP/RTCP video sub-sessions. Due to NAT problems, we can't make assumptions on a specific range of UDP ports for the relaying process. In our application, the UDP ports are freely chosen in the range of available ports.

When a node R_i has to forward the stream packets received by satellite, it allocates the resources needed for the relaying and simply begin to send its buffered packets to the new destination over unicast UDP channels. The host H_j receives the stream from the relayer node in the form of RTP/RTCP packets that are buffered and then forwarded to a local player in order to allow the end-user to view the requested content.

The host H_j can, in its turn, become a secondary relayer, in the sense that it can forward the received packets over unicast terrestrial links to other H nodes of the system. So, the operations performed by an R node, must be implemented in an H node too.

The RTCP signalling allow the nodes involved in the relaying process to periodically report some QoS statistics to the server S, thanks to the RTCP SR and RTCP RR messages exchanged between R and H nodes. As long as no relevant loss rate nor jitter variability is experienced by the H node, the QoS level is good and no countermeasures have to be taken from the server S. On the other extreme, the failed delivery of the stream must be dealt with: this can be due to a significant congestion over the R-H link or, more simply, to a failure of the node (a crash or the loss of the Internet connection). In this situation, the node H will stop receiving the stream packets, and the local player will go "dark". In order to avoid this situation, and to guarantee a minimum level of QoS and QoE for the end user, it's necessary to consider the worst scenario and to make provisions beforehand.

Our strategy is based on the assignment of a fall-back node, R_{bkup}, for every H node. Due to the NAT problems, H and R_{bkup} have to perform NAT Traversal procedures as soon as possible, that is, after the relaying process between R_i and H is established. In order to avoid wasting uplink bandwidth of R_{bkup} and not to affect the outgoing traffic of the R_{bkup} node, the stream is not duplicated from R_{bkup}. Still it has to keep direct UDP connections between R_{bkup} and H alive (considering the limited NAT binding lifetimes). Therefore R_{bkup} maintains a periodic signalling over the 4 UDP channel. When H experiments a QoS degradation in the transmission from R_i, it must report it to the server S, which is responsible for activating the appropriate procedures; however, in case of failure of R_i, H has the chance to immediately activate the relaying from R_{bkup} on the already opened channels. All these changes must be immediately reported to the server, in order to let it maintain a consistent view of the dynamic overlay.

4. CHARMS ARCHITECTURE SUMMARY

1. Server set-up: the server S initializes the system's database and starts to listen clients (that is request from Hs and offers from Rs) on TCP and UDP ports, in order to let nodes refreshing their UDP channels and activate a second way to communicate with it.

2. Client set-up and registration: whenever it tries to join the CHARMS overlay network, a client must establish a TCP connection with the server S, as depicted in Figure 4. The server S records the client's request and it sends to the client a list of public addresses towards which the client will perform both bandwidth/latency tests, and the NAT Detection tests. Thus, the client returns to the server S the results of these tests, which are needed to characterize the outgoing

channel. All dialogs in the system (i.e. the dialogs between server and clients and between couples of clients) are exchanged by an ad hoc protocol, the *CHARMS protocol* (see Section 4.1). The last operation, performed by the client during this phase, is needed to discover the type of NAT box it is connected to. This test is performed with a combined NAT Detection test, which is able to identify the mapping and filtering properties of the NAT chain, from the client up to STUN servers used to perform the test. Also these information are formatted in a specific CHARMS message and sent to the server. At the end of each client's set-up and registration phase, the server S is informed about the bandwidth and the latency of the client's outgoing channel and also about the type of its NAT.

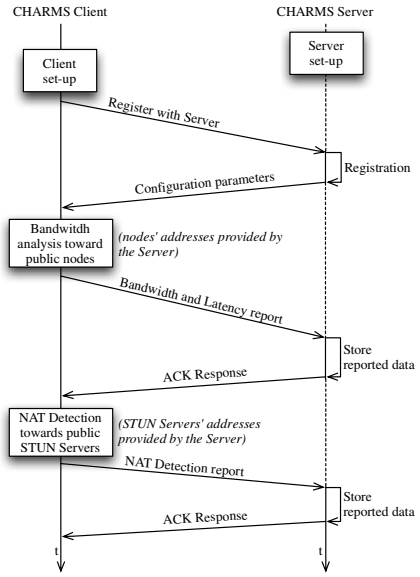


Figure 4. Client set-up and registration with the CHARMS Server

3. Stream sharing or request: any client able to receive an audio-video stream from the satellite (i.e. the potential primary relayer) can decide to share the stream with other clients ("the hosts") over the terrestrial network through the CHARMS architecture. If it agrees to share the stream, the relayer, identified as R_i, must inform the server S about such choice by sending it a message which contains information about the stream. In a similar way an host, identified as H_j, wishing to receive a stream through the terrestrial network, advances its request to the server S. The server must store and update the lists of the available relayers and that of the clients requesting streams. Let's note that, once it receives a stream, a client H_j could decide to become a relayer on its turn (a secondary relayer), as outlined in Figure 5.

4. Calculation of the candidate relayers subset: for each H_j the server S selects the subset of R_i which are NAT-compatible with H_j evaluating the possibility of a direct UDP connection between the two clients. The set of candidate relayers is defined as $R'=\{R_i \mid R_i$ *is receiving* $stream_k$ *and* R_i*'s NAT is compatible with* H_j*'s NAT}*. The cardinality of R' will be further reduced considering only the nodes that have sufficient available bandwidth to support another outgoing relaying, as reported by the periodic checks of available bandwidth toward fixed destinations, and utilizing load-sharing policies among the remaining ones. At the end of this step, the server S has a small set of candidate relayers for H_j relative to the $stream_k$, R_k^*: among these relayers, the specific one for H_j will be choosed as result of the relayer selection process.

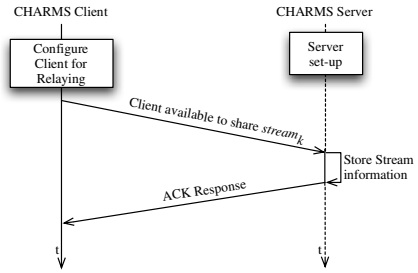


Figure 5. Sharing a stream

5. Relayer selection: if set R_k^* is empty, the requesting H_j enters a wait state. Periodically the server S checks for the presence of a new R_i, able to serve H_j with the requested stream and with proper NAT and outgoing channel capabilities. Otherwise, the server S will ask all the candidate relayers (belonging to the R_k^* set) to perform a bandwidth/latency test toward H. This test also requires a NAT Traversal operation prior to the uplink estimate. So, the selected R_i and H will create an UDP direct connection through their NATs, and if the channel is correctly open the bandwidth/latency test can be performed. The procedure is repeated sequentially by all the candidate nodes in the R_k^* set, as reported in Figure 6. After obtaining the tests' results, the server chooses the best relayer R_{best} comparing the results according to the metrics defined in Section 3.3. The bandwidth/latency tests are periodically refreshed to cope with the variations both of the connections and of the set of clients. At the end of this phase, the match is completed and the server S records the relaying couple (R_{best}, H_j) associated to the stream $stream_k$ as a session uniquely identified by the set: (R_{best}, H_j, $stream_k$). The set R_k^* is dynamically updated to reflects variations in the state of the R nodes.

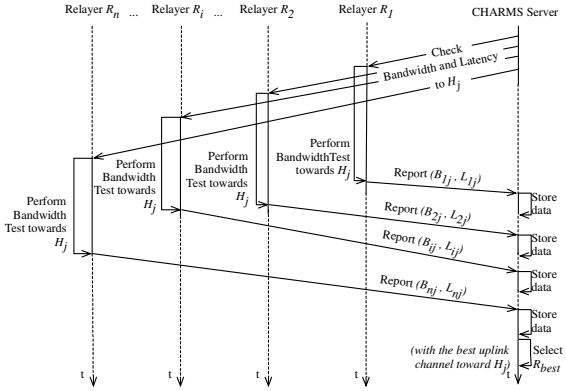


Figure 6. Process to choose R_{best}

6. Resource allocation and solving the NAT issue: first of all, the CHARMS server informs both R_{best} and H_j that the session related to a certain stream has been established, sending a message containing the relevant data. After receiving this message, both the clients R_{best} and H_j perform the following steps:

a. opening of 5 UDP sockets: 4 for the couple of RTP/RTCP channels and the last for the specific CHARMS signalling.

b. binding of every socket to a different local address, for which the corresponding public address must be determined.

c. requesting the mapped addresses, corresponding to the local ones, to a public STUN server.

d. coding such information about addresses in a message and sending it to the server S.

As soon as the server S gets from a client the couples of the local and public transport addresses for each channel, it shares these information with the client's peer. Now, the both clients are ready for the NAT traversal operation.

7. NAT Traversal: each client performs five NAT Traversal operations, one for each socket. The purpose is to create a 1:1 correspondence between a local socket and one of the peer's known addresses. In order to do this, each client implements a STUN client/server. The STUN client is responsible for sending a set of STUN Binding Request to the known addresses of the peer (the local one and the public one, extracted in the message received from the server). The sending is periodic, until a response is received. The STUN Server waits for a STUN Binding Request from the peer to which it will reply with a STUN Binding Response containing the mapped address as seen by the STUN server. In this way, each client discover the position of the peer relative to itself: they could be in the same local network, in which case the local address will be used to communicate directly, otherwise they learn the public address where all following messages will be sent. During this process, the receiving of a message from a peer means that the peer has punched a hole in the client's NAT box, and that a direct communication is now possible. At the end of the process, both clients will know, for each local address, the corresponding peer address to contact. The clients inform the server about the result of the NAT Traversal process and in case of success, the relay will start, as shown in Figure 7. Otherwise, the server will choose another R'_{best} for H_j, and the overall process will start again from the previous step. While the relaying is in progress, CHARMS signalling messages are also exchanged between the two peers on the dedicated channel.

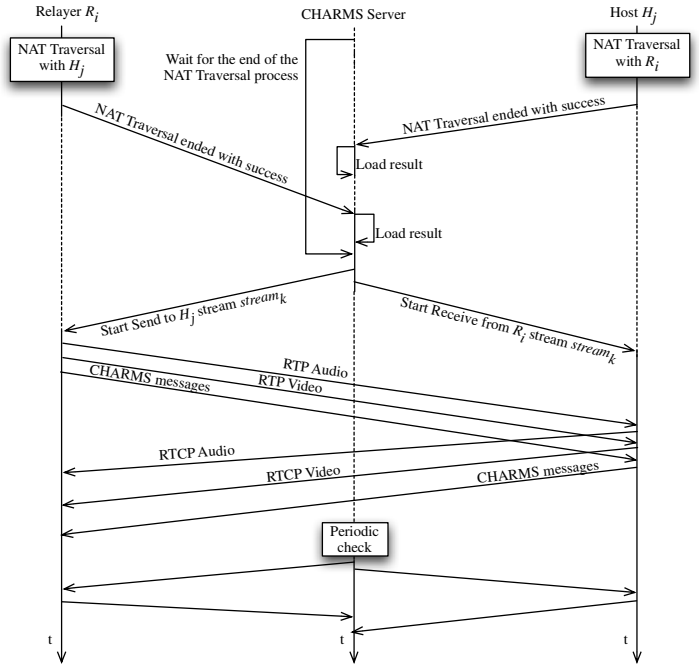


Figure 7. Starting the stream relaying between peers

8. Stream relay: the server S allows both the clients to activate the stream. By an operating system's point of view, the streaming process makes use of a POSIX thread which takes care of the UDP sending/receiving messages: regardless a client is the relayer or the receiving host, it loads the thread which will end only after the end of the streaming, leaving the main application free to manage other streaming sessions and the overlay management operations requested by

the server in the meanwhile. Each thread has the role to start the procedures that activate the relaying modules for forwarding, receiving and play the stream in each node.

4.1. Implementation details and first experimental results

All the procedures needed to perform those tasks are implemented in C language using the Unix System API, as for the rest of the applications in the system. Concurrency is managed using threads in compliance with SUSv3 standard [42].

The CHARMS protocol is implemented using an XML [43] tree representation: each message is composed of a *Header*, a *Body* and a *Trailer*. The Header contains information needed to validate messages, perform message identification (*sequence number* field) and distinction in the type of message (*message type* field). The Body part contains the relevant information: every type of message has a unique data representation, specific for the set of value that have to be transmitted between endpoints. Finally, the Trailer part is needed in the messages because of the UDP transport: it contains an MD5 [44] digest of the entire message which is used at the destination to verify the integrity of the message.

The size of messages exchanged among the entities of the system is small, so no significant overhead is payed for the adoption of an XML encoding schema, compared to other encoding schemas such as TLV. Moreover the schema offers other advantages such as a very easy extensibility and human readability.

First tests in real-world scenario have been very encouraging and have shown the potential of the solution implemented. Work currently carried on is producing a statistical description of the performances achieved by the proposed solution.

5. CONCLUSIONS

A very simple, scalable and robust architecture for disseminating an IP multicast stream over a large area with no support for terrestrial multicast traffic has been described. By positioning satellite receivers in different ASs the CHARMS architecture can parallel the behaviour of expensive CDNs or the performance of closely guarded multicast enabled networks. In the case of CSS (Campus Satellitare del Salento) it can help appreciably to extend the reach of the distance learning infrastructure by allowing temporary users to join the lectures and motivating them to get the equipments to join the CSS permanently, consequently expanding the base for the functioning of the same CHARMS architecture.

A forthcoming paper will provide more implementation details and measures/statistics about the actual operations of the architecture. It will also try to prove its real-world effectiveness. Meanwhile, a promising direction for future work looks to be the crossing of the ideas here described with other efforts aimed at the same results. For example, P2P architectures [45][46] for the distribution of video could gain from the approach described here acceptable performances in terms of latency, which are not in sight for them at the moment. A better scalability could also be gained by MBONE-like architectures, by adopting some of the ideas here exposed.

ACKNOWLEDGEMENTS

The authors wish to express their gratitude to Simone Molendini, Elena Scialpi, Dario Pellegrino, Salvatore Totaro, Valerio De Luca, Antonio Tamborino, of the University of Salento, for many useful discussion and suggestions.

REFERENCES

[1] Diot, Cristophe & Levine, Brian N. & Lyles, Bryan & Kassem, Hassan & Balensiefen, Doug (2000) "Deployment Issues for the IP Multicast Service and Architecture", *IEEE Network*, Vol. 14, No. 1, pp78-88.

[2] Eriksson, Hans (1994) "MBONE: the multicast backbone", *Communications of the ACM*, Vol. 37, No. 8, pp54-60.

[3] *IETF MBONED Working Group*, http://datatracker.ietf.org/wg/mboned/charter/.

[4] Sardella, Alan (2005) "Video Distribution in a Hybrid Multicast-Unicast World", *Juniper networks*, http://www.juniper.net/solutions/literature/white_papers/200107.pdf.

[5] *Campus Satellitare del Salento*, http://www.campusdelsalento.it/campusen/index.html.

[6] Tommasi, Franco & Campa, Antonio & Molendini, Simone (2006) "The Campus Satellitare del Salento: a scalable and transparent e-learning application", *Proceedings of SoftCOM'06*, pp363-368.

[7] Tommasi, Franco & Campa, Antonio & Scialpi, Elena (2007) "The Campus Satellitare del Salento: a large-scale satellite distance e-learning trial", *Proceedings of EDEN'07*, European Distance and E-learning Network.

[8] Tommasi, Franco & Molendini, Simone & Scialpi, Elena & Melle, Catiuscia (2008) "A new approach for Hybrid Satellite-Terrestrial Networks: cooperative architecture to relay multicast satellite streams", *Proceedings of ICLAN'2008*, International Conference on the Latest Advances in Networks.

[9] Tommasi, Franco & Molendini, Simone & Scialpi, Elena & Melle, Catiuscia (2010) "CHARMS: Cooperative hybrid architecture for relaying multicast satellite streams to sites without a satellite receiver", *IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS)*, pp269-276

[10] Leighton, Tom (2009) "Improving performance on the Internet", *Communications of the ACM*, Vol. 52, No. 2, pp44-51.

[11] Vakali, Athena & Pallis, George (2003) "Content Delivery Networks: Status and Trends", *IEEE Internet Computing*, Vol. 7, No. 6, pp68-74.

[12] Weigle, Eric & Hiltunen, Matti & Schlichting, Rick & Vaishampayan, Vinay A. & Chien, Andrew A. (2006) "Peer-to-Peer Error Recovery for Hybrid Satellite-Terrestrial Networks", *Proceedings of Peer-to-Peer Computing'06*, Sixth IEEE international Conference on Peer-To-Peer Computing, pp.153-160.

[13] De Belleville, Florestan & Dairaine, Laurent & Lacan, Jerome & Fraboul, Christian (2004) "Reliable Multicast Transport by Satellite: A Hybrid Satellite/Terrestrial Solution with Erasure Codes", *HSNMC (High Speed Networks and Multimedia Communications) Lecture Notes in Computer Science*, Vol. 3079, pp436-445, Springer.

[14] Filali, Fethi & Dabbous, Walid & Kamoun, Farouk. (2001) "Efficient Planning of Satellite-Terrestrial Hybrid Networks for Multicast Applications", *Proceedings of ISCC'01*, IEEE Symposium on Computers and Communications, Hammamet, Tunisia.

[15] Filali, Fethi & Dabbous, Walid (2001) Issues on the IP "Multicast Service Behaviour over the Next-Generation of Satellite-Terrestrial Hybrid Networks", *Proceedings of ISCC'01*, IEEE Symposium on Computers and Communications, Hammamet, Tunisia.

[16] Wu, Hsiao Kuang & Chang, Chao Hsu (2001) "Adaptive Multicast Routing for Satellite-Terrestrial Network", *Proceedings of GLOBECOM'01*, IEEE Global Telecommunications Conference, San Antonio, USA.

[17] Chang, Chao Hsu & Wu, Hsiao Kuang & Jin, Ming Hui & Tseng, Yueh O (1999) "Intelligent Routing for Global Broadband Satellite Internet", *Proceedings of Europe HPCN'99*, Lecture Notes In Computer Science (LNCS), Vol. 1593, pp80-89.

[18] Kerczewski, Robert J. & Chomos, Gerald J. & Griner, James H. & Mainger, Steven W. & Martzaklis, Konstantinos S. (NASA, Glenn Research Center, Cleveland, OH), & Kachmar, Brian A. (Analex Corp., Brook Park, OH) (2000) "A hybrid satellite-terrestrial approach to aeronautical communication networks", *Collection of Technical Papers of AIAA-2000*, AIAA International Communications Satellite Systems Conference and Exhibit, Oakland, CA.

[19] Asai, Kikuo & Osawa, Noritaka & Kondo, Kimio & Ishihara, Kazuo (2004) "Performance estimation of asymmetric satellite-closed network for Inter-university Communication System", *Proceedings of IEEE Vehicular Technology Conference*.

[20] Corry, Jim (2008) "Interoperable Satellite Communications", *Proceedings of IEEE Conference on Technologies for Homeland Security*, Waltham, MA.

[21] Trajkovic, Ljiljana & Shao, Qing & Fraser, Simon (2004) "Measurement and Analysis of Traffic in a Hybrid Satellite-Terrestrial Network", *Proceedings of SPECTS'04*, International Symposium on Performance Evaluation of Computer and Telecommunication Systems, San Jose, California.

[22] Chang, Hyunseok & Jamin, Sugih & Wang, Wenjie (2009) "Live streaming performance of the Zattoo network", *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference (IMC '09)*, pp417-429.

[23] Yin, Hao & Liu, Xuening & Zhan, Tongyu & Sekar, Vyas & Qiu, Feng & Lin, Chuang & Zhang, Hui & Li, Bo (2009) "Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky", *Proceedings of the seventeen ACM international conference on Multimedia (MM '09)*, pp25-34.

[24] Holdrege, Matt & Srisuresh, Pyda (2001) "Protocol Complications with the IP Network Address Translator", *RFC 3027*.

[25] Huston, Geoff (2004) "Anatomy: A look inside Network Address Translators", *The Internet Protocol Journal*, Vol. 7, No. 3, pp1-15.

[26] Jennings, Cullen (2007) "NAT Classification Test Results", *draft-jennings-behave-test-results-04*.

[27] Rosenberg, Jonathan & Weinberger, Joel & Huitema, Christian & Mahy, Rohan (2003) "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", *RFC 3489*.

[28] Rosenberg, Jonathan & Mahy, Rohan & Matthews, Philip & Wing, Dan (2008) "Session Traversal Utilities for (NAT) (STUN)", *RFC 5389*.

[29] MacDonald, Derek & Lowekamp, Bruce. (2010) "NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)", *RFC 5780*.

[30] Audet, Francois & Jennings, Cullen (2007) "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", *RFC 4787*.

[31] Ford, Bryan & Srisuresh, Pyda & Kegel, Dan (2005) "Peer-to-peer communication across network address translators", *Proceedings of the USENIX'05 Annual Technical Conference Anaheim*, CA.

[32] Srisuresh, Pyda & Ford, Bryan & Kegel, Dan (2008) "State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)", *RFC 5128*.

[33] Mahy, Rohan & Matthews, Philip & Rosenberg, Jonathan (2010) "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", *RFC 5766*.

[34] Rosenberg, Jonathan (2010) "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", *RFC 5245*.

[35] Prasad, Ravi & Dovrolis, Constantinos & Murray, Margaret & Claffy, K. (2003), "Bandwidth estimation: metrics, measurement techniques, and tools", *IEEE Network*, Vol.17, No. 6, pp27-35.

[36] *Clink: a tool for estimating Internet link characteristics*, http://allendowney.com/research/clink/.

[37] Downey, Allen (1999) "Using pathchar to estimate internet link characteristics", *Proceedings of ACM SIGCOMM'99.*

[38] Jain, Manish & Dovrolis, Constantinos (2003) "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput", *IEEE/ACM Transaction on Networking (TON)*, Vol. 11, No. 4, pp537-549.

[39] Handley, Mark & Jacobson, Van (1998) "SDP: Session Description Protocol", *RFC 2327.*

[40] Schulzrinne, Henning & Rao, Anup & Lanphier, Robert (1998) "Real Time Streaming Protocol (RTSP)", *RFC 2326.*

[41] Schulzrinne, Henning. & Casner, Stephen & Frederick, Ron & Jacobson, Van (2003) "RTP: A Transport Protocol for Real-Time Applications", *RFC 3550.*

[42] *The Single UNIX Specification, Version 3*, http://www.unix.org/version3/.

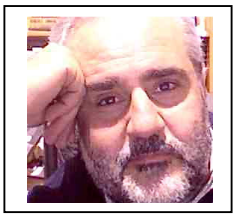
[43] *Extensible Markup Language (XML)*, http://www.w3.org/XML/.

[44] Rivest, Ron (1992) "Real Time Streaming Protocol (RTSP)", *RFC 1321.*

[45] Galán-Jiménez, Jaime & Gazo-Cervero, Alfonso (2010) "Overlay Networks: Overview, Applications and Challenges", *International Journal of Computer Science and Network Security*, Vol. 10, No. 12, pp40-49.

[46] Tang, Yun & Sun, Lifeng & Zhang, Meng & Yang, Shiqiang & Zhong, Yuzhuo (2006) "Live Video Streaming Service over Peer to Peer Network: Design, Implementation and Experience*", *International Journal of Computer Science and Network Security*, Vol. 6, No. 3, pp77-86.

Authors



Francesco Tommasi graduated from University of Pisa, Pisa, Italy, in Electronic Engineering in 1984. In the same year he joined CSELT, Turin, Italy, where he became involved in image processing. In 1988 he joined the Istituto per la Ricerca Scientifica e Tecnologica, Trento, Italy, where he made researches in image motion analysis. He has been head of software development at Cigraph, Venice, Italy. He is now associate professor at the Engineering Faculty of the University of Salento, Italy. His present research interests include networking and the applications of satellite to distance learning.

Catiuscia Melle graduated magna cum laude from University of Salento, Lecce, Italy, in Information Engineering in 2009. Currently she is a PhD student at LIIS laboratory at University of Salento. Her present research interests include networking and content distribution applications.