GLORIA is funded by the European Union 7th Framework Programme (FP7/2007-2013) under grant agreement n° 283783

GLObal Robotic telescopes Intelligent Array for e-Science

Peer-to-Peer live broadcasting software specification and source code

CODE: DEL-062

VERSION: 01.D

DATE: March 28th, 2013

Authors:	Urko SERRANO (UPM) Mercedes GRIJALVO (UPM)
Collaborators:	-
Revised by:	Fernando SERENA (UPM)
Approved by:	Francisco M. SANCHEZ (UPM)

Distribution List:

Name	Affiliation	Date
Urko SERRANO	UPM	March 28st, 2013
Fernando SERENA	UPM	March 28st, 2013
Francisco M. SANCHEZ	UPM	March 28st, 2013

Change Control

Issue	Date	Section	Page	Change Description
01.A	06/03/2012	All	All	Creation
01.B	30/03/2012	All	All	Review style and format
01.C	29/09/2012	1.5	All	Add algorithm design section as part of D9.6
01.D	28/03/2013	1.6	All	Update transport protocol algorithm

Reference Documents

Nº	Document Name	Code	Version

Index

Figures Index

1. Introduction

This document covers the design and implementation of a collaborative broadcast infrastructure that provides a media content solution for astronomical live events within the GLORIA system. All the content has been elaborated by the partners of the GLORIA project, which means "GLObal Robotic-telescopes Intelligent Array for e-Science". This project is funded by the European Union 7th Framework Programme FP7/2007-2013) under grant agreement no. 283783.

1.1. Purpose

The main topic of this document is the specification of a cost-effective media platform that enables GLORIA system to broadcast astronomical events to a mass audience with the collaboration of the GLORIA community. The platform is organized into two main software solutions: media production and content delivery.

1.2. Scope

This document is primarily aimed at developers of the GLORIA system. It includes the design and the deployment of the broadcasting platform for astronomical live events.

1.3. Responsibilities

The content of this document is the responsibility of UPM.

1.4. Overview

The outline of this document proceeds as follows. The System chapter summarizes the concept of the media platform along with an explanation of both media production and content delivery solutions. The Media production section presents the components which are necessary to generate the final video stream from several media sources. Then, the Content delivery section explores two main approaches to deliver live content to the GLORIA users, where both are adopted in the final infrastructure.

1.5. System

The system embraces the content delivery infrastructure in charge of broadcasting astronomical events for the GLORIA community. The system covers the technology behind the media lifecycle, from media production to live content dissemination.

The aim of the system is two-fold. First, it provides a procedure to ensure live media content for astronomical events regardless of geographical locations and limited Internet access. Second, the system integrates a media platform solution to support online video streaming to a large audience. The underlying infrastructure is designed to combine several approaches for live content distribution, where the end-user is able to share the current media stream in real-time among other viewers within the GLORIA community.

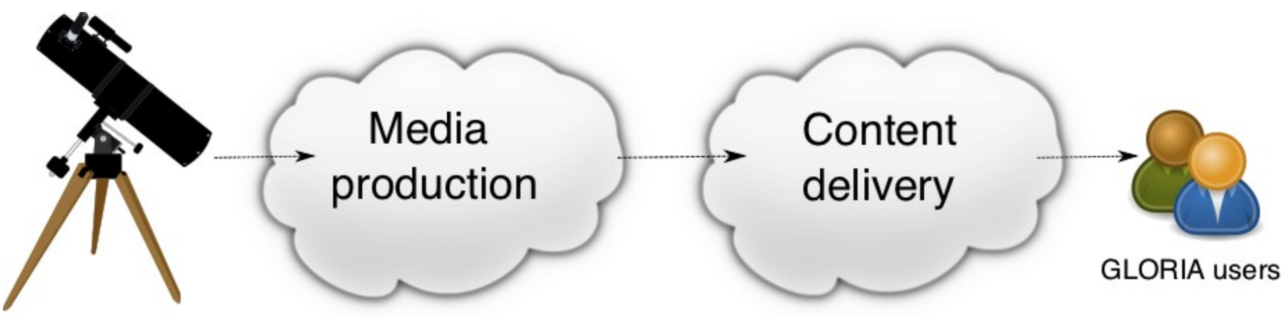

Figure 1: Flow diagram of the overall system.

The Figure 1 represents the overall schema of the system. The media production is the technical solution to broadcast live streaming content for each astronomical event. This solution allows to coordinate several media input streams from local or remote sources in order to generate a final output media stream, achieving smooth video playback. Moreover, the solution abstracts away the complexity of dealing with media sources scattered geographically, presenting a common user interface for the media producer.

On the other hand, the content delivery is focused on the media streaming solution that provides content to the GLORIA users. The architecture that implements this solution is able to scale gradually according to the number of current viewers. It is intended to save computational and network resources by offering the GLORIA users the possibility to contribute and share the live media content.

1.5.1. Media production

This section analyses the steps needed to deploy the media production solution for each astronomical event. The first subsection defines the media sources, either local or remote, that are supported by the system. Then, the broadcasting software is explained in order to combine all media sources, along with the necessary encoding tool to transmit the final output stream. The following subsection presents the equipment that is mainly used to get access to Internet during the expeditions and hence, generates the live streaming. Finally, the last subsection covers the media infrastructure.

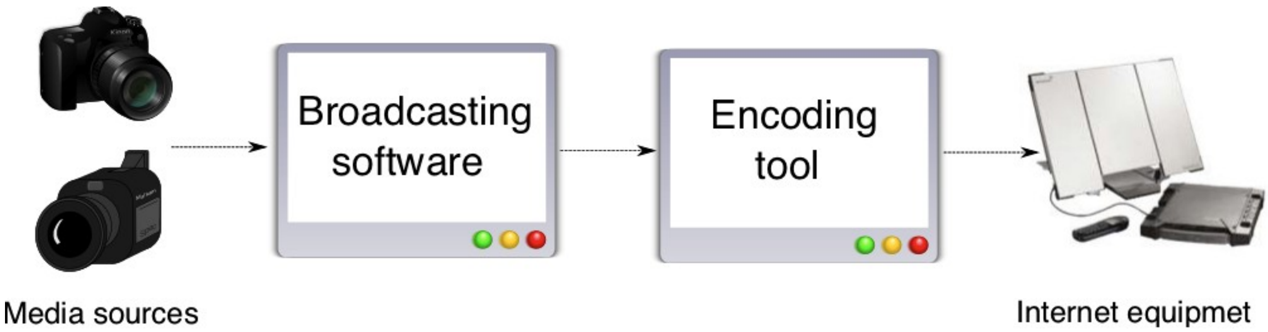

Figure 2: Media production overview.

1.5.1.1. Media sources

On each broadcasting there are several media input sources that can be added to the final video stream. Apart from live telescope images and recorded clips, several audio and video sources are used in live production such as cameras attached to telescopes. These media sources can be generated from different locations and all of them have to be reached from the server hosting the broadcasting software.

The main issue regarding multiple remote media sources is the delay time that is inherent to each source, mainly caused by network overhead. In addition, the broadcasting software has to verify the correct delivery of all remote media sources to avoid errors and therefore, a choppy video playback in the output stream. This situation is targeted by the broadcasting infrastructure mentioned below.

1.5.1.2. Broadcasting software

Media sources have to be managed properly to generate the final output stream. The broadcasting software is essentially a live video switcher that allows the media producer to control in real time the current output stream while adding effects, transitions, and other external resources. A common scenario is a set of cameras recording the astronomical event while professional astronomers give an explanation helped by graphical resources.

The broadcasting tool used for media production is a free software application called [CamTwist Studio]. This software is a multi-camera switcher that provides a wide range of features for several media sources in high definition. It supports professional transition effects and it is compatible with all major encoding tools. In addition, it is designed to customize predefined settings and apply several transitions among different video sources at once.

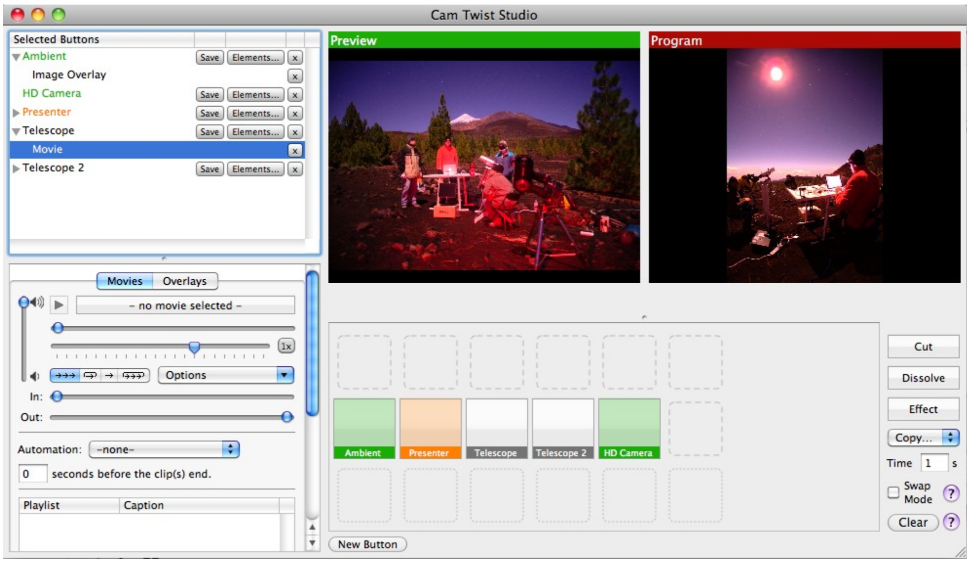

Figure 3: CamTwist Studio - broadcast live video switcher.

1.5.1.3. Encoding tool

During a live streaming, the output stream generated from the broadcasting software has to be encoded along with the audio channels. This implies that the current stream has to be adapted and transmitted through Internet in a specific media container. This media container includes all the metadata that is needed for the user to play the streaming in the GLORIA website. The quality of the video stream is selected according to the current available bandwidth and the network congestion.

The media production solution contemplates two main encoding tools: [VideoLAN] and [Flash Media Live Encoder]. VideoLAN is a free open source project that is further beyond the features of a normal media player. It provides you with a set of plugins to broadcast your own video streams in a variety of media formats in a flexible way. This software is mainly used to encode the remote video sources for the broadcasting software. On the other hand, Flash Media Live Encoder is a free encoding tool with professional settings for adaptive video quality, such as dynamic bit-rate or auto frames dropping. This tool is in charge of generating the final video output stream and it is compatible with the streaming servers used in the content delivery network solution.

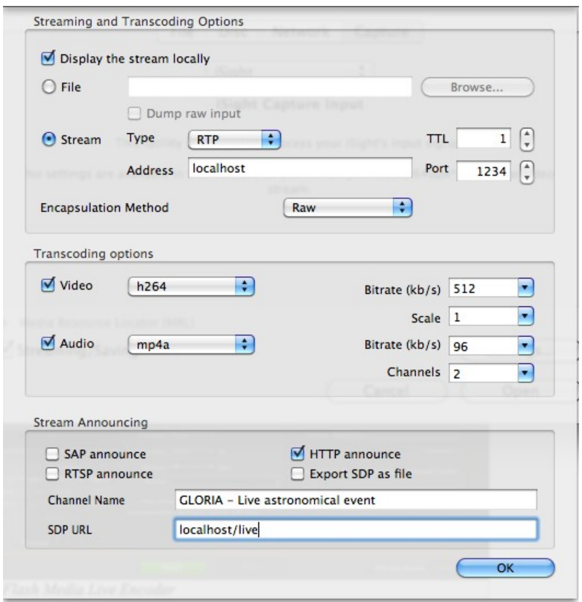

Figure 4: VideoLAN streaming settings.

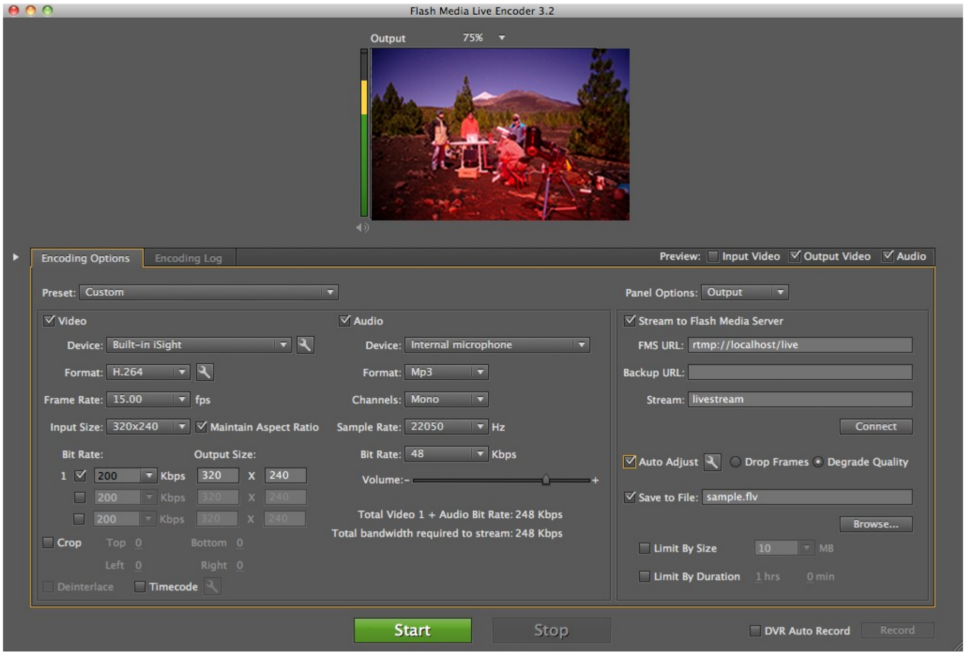

Figure 5: Flash Media Live Encoder.

1.5.1.4. Internet equipment

In order to publish media content for the GLORIA community, the Internet access connection represents a key part of a live broadcasting on each expedition. Given that these expeditions can be located in isolated places with poor Internet coverage, a common solution has to be deployed to ensure enough bandwidth once media content is generated. The Internet connection provided by this solution will be also used to monitor the status of the content delivery network in real time and assure broadcasting availability for the GLORIA community.

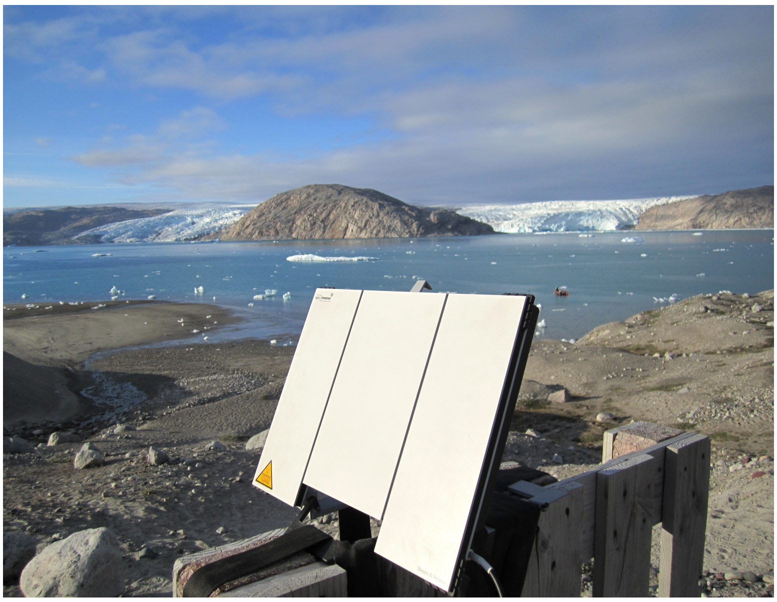
Figure 6: BGAN terminal.

The adopted solution to overcome live stream disruptions is the [BGAN] Broadband Satellite Internet. BGAN is a combined voice and broadband data mobile communications service delivered via a highly portable device. Based on IP technology, BGAN makes it possible for the media production to connect the live stream to the Internet at speeds up to 492Kbps.

This solution offers seamless and continuous coverage in areas where terrestrial telecom networks are non-existent. In addition, this mobile communications service provides guaranteed data rates on demand. This is ideal for the purpose of astronomical live events where quality of streaming is paramount.

1.5.1.5. Infrastructure

The infrastructure of the media production is composed by four main entities:

- *Streaming node* (N): entity that generates a media source from a specific geographical location.
- *Coordinator* (C): streaming node that aside from generating a media source, it controls the broadcasting server remotely and monitors the current output stream.
- *Intermediary server* (I): entity used as a backup system in case the media sources can not be reached directly by the broadcasting server.
- *Broadcasting server* (B): component of the infrastructure that includes the necessary broadcasting and encoding software. It collects the media sources, supports remote access and generates the output stream.

The are two main approaches in order to deploy the media production solution according to how the media sources are captured by the coordinator:

- *Pull approach*: the broadcasting server behaves as an active entity, requesting all media sources from the streaming nodes. Using this topology, the overall performance increases since the broadcasting server gets the input streams directly from the streaming nodes, without any other entity involved. This configuration reduces the time delay among the media sources generated by the streaming node. In addition, it allows the coordinator to get a faster response while controlling the live media production.

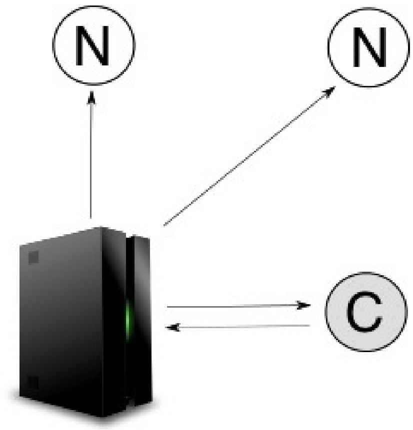

Figure 7: Pull approach.

  On the other hand, the main drawback of this approach resides in the need of a predefined network configuration in the broadcasting server. This is because the streaming server has to know in advance the public IP address of each streaming node. This situation narrows down the possibility of adding new nodes on demand and exposes the overall system to a network failure if the IP addresses change dynamically.

- *Push approach*: this scenario is depicted in Figure 8. Using this configuration, each streaming node sends the media source to the intermediary server. Then, the broadcasting server accesses these media sources through the intermediary server, which is controlled remotely by the coordinator node. In this scenario, the infrastructure is more flexible to network changes as well as it allows streaming nodes without prior notification to the broadcasting server.

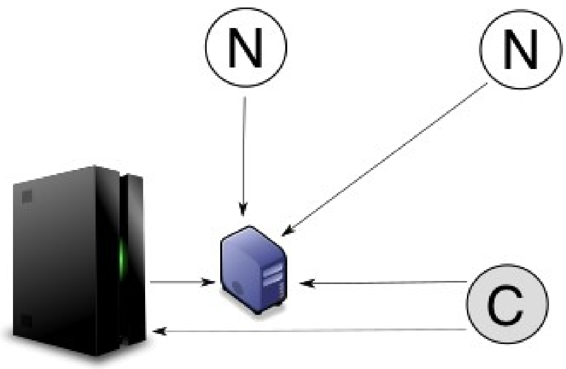

Figure 8: Push approach.

> However, the use of an intermediary server brings worse performance than the previous approach. For instance, the coordinator could have the media sources out of sync, which turns out into a wrong output stream. Also, the delay of the final output stream increases drastically.

In essence, the media production is based on a hybrid push-pull infrastructure where the system behaves as a pull approach with a push fallback support.

1.6. Content delivery

The output stream generated by the media production solution has to be delivered to the users of the GLORIA community. To do so, the content delivery solution takes advantage of the resources donated by the audience watching an astronomical live event, along with a cloud computing infrastructure.

First, the term cloud computing and its features will be explained. Then, the following section presents the peer-to-peer technology that enables GLORIA users to collaborate in the system. The last section defines the infrastructure of the content delivery solution and its behaviour in a live scenario.

1.6.1. Cloud computing

Broadly speaking, the definition of cloud computing refers to services available over the Internet. These hosted services are offered from plenty of online virtual servers spread all over the world, which collectively are referred to as a cloud. These services are provided on demand, typically in terms of time and bandwidth consumption. These solutions are elastic, so the system can grow dynamically at any given time.

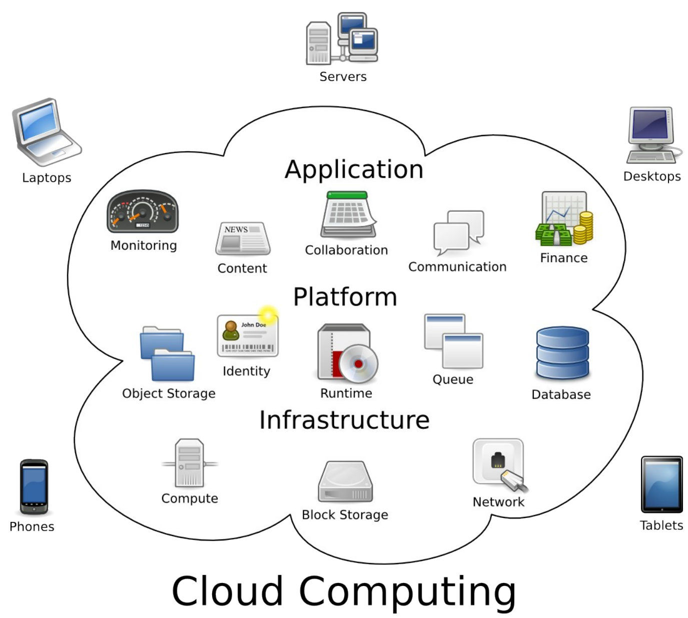

Figure 9: Logical diagram of a cloud computing architecture.

There are several categories of cloud computing, but the solution is focused on Infrastructure-as-a-Service. In this sense, a cloud server provider offers virtual server instances to start, stop, access and configure their virtual servers. These server instances can be managed remotely, installing a common set of software applications regardless of the hardware underneath. Eventually, the solution consists of a network of servers running the same software and hence, minimizing maintenance and deployment costs.

The main features of a cloud computing system are as follows:

- *Cross platform*: different hardware and operating systems are supported, behaving as a whole.
- *Scalable*: the system's capacity increases or decreases depending on the current users.
- *Reliable*: virtual servers can be replaced in real time without affecting the overall performance.

1.6.1.1. Media streaming platform

The cloud computing system forms the basis of the content delivery solution. However, virtual servers need a software application to deliver the output stream from the media production to the GLORIA users. In this regard, a media streaming platform is deployed on top of the cloud computing system. This platform includes several live streaming servers installed on each virtual server, which are hosted by the cloud-based system. They interact with each other to bring a robust solution for a massive audience during the each live astronomical event in GLORIA.

The following benefits of this media streaming platform can be sum up hereafter:

- Fast response and good performance in scenarios with high overhead.
- Multi-streaming server deployment to scale up to hundreds of virtual server instances.
- The output stream from the media production can be transcoded to different media formats.
- Multi-screen support for all major devices, from desktop to mobile devices.

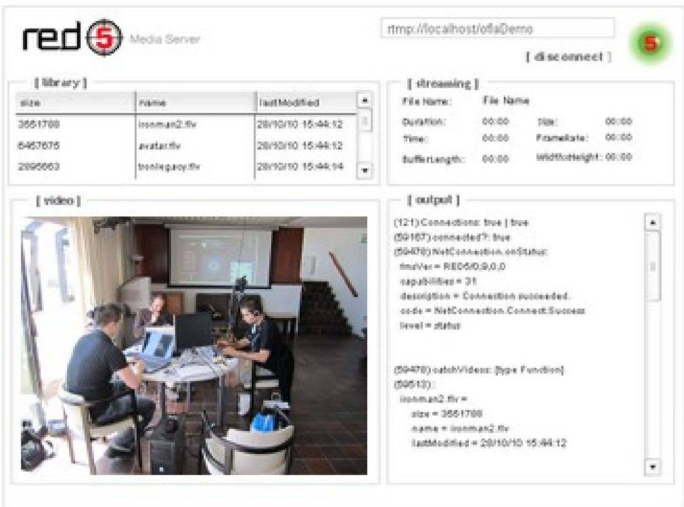

Figure 10: Red5 streaming media server.

Two different live streaming servers can be used interchangeably within the platform. [Red5] is an open source media server based on the RTMP protocol and with additional features such as multi-user interaction among viewers. On the other hand, [Wowza Media Server] offers a mature and modular solution with several addOns, providing a throughput up to 10Gbps per virtual server, approximately 20.000 current GLORIA users.

1.6.2. Peer-to-Peer overlay network

Currently, all major content delivery solutions are based on a set of streaming servers which are deployed in private or public clouds. The main drawback of this approach is the cost of bandwidth that is needed to support a large-scale infrastructure for a wide audience.

A peer-to-peer approach takes advantage of resources available from the GLORIA users or *peers* to collaborate with each other. These peers form an overlay network of multiple nodes, which is built on top of the IP layer. This overlay network tends to be decentralized, with no predefined hierarchy.

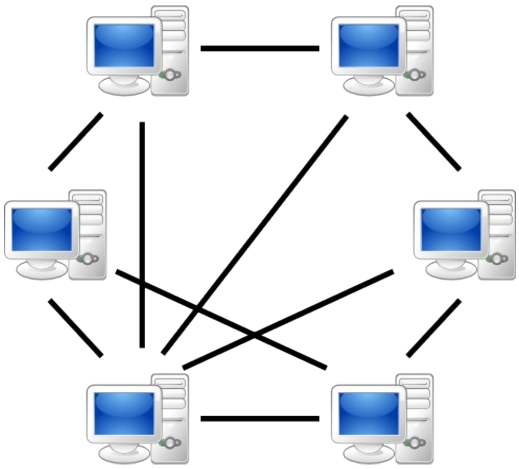

Figure 11: Peer-to-peer network.

A node in a peer-to-peer network can be a seed or leech. A seed is used to refer to a node who has all the content available. A seed usually accepts incoming connections from leeches who request the content, since they do not have the complete playable stream.

A peer-to-peer network has several features:

- *Cost-effective*: no streaming servers are needed since the output stream is transmitted among the nodes.
- *Fully distributed*: nodes in the network are self-dependent and there is no unique point of failure.
- *Scalable*: the size of the network grows according to the amount of current live users.
- *Maintenance*: the network is self-organized with the help of a membership management algorithm.

Several peer-to-peer streaming networks can be found nowadays with similar characteristics. [P2P-Next] is a European research project focused on a free open source content delivery platform and it is supported by a consortium consisting of researches in academia and well-known commercial partners. [BitTorrent Live] is a cutting-edge cross-platform solution based on the popular BitTorrent file sharing open protocol. Also, [Octoshape Cloudmass] is a content delivery network hosted in a cloud streaming infrastructure with peer-to-peer capabilities.

1.6.3. PPSPP transport protocol

Traditional Internet protocols are based on a one-to-one communication approach. Typically, a client requests a service to a central server to fetch certain data. This communication provides the abstraction of conversation between both entities.

Despite the fact that these protocols are robust and well-defined, they do not scale properly in a large-scale live content scenario. Nowadays, the Internet is mostly used for content dissemination, where streaming content is becoming the dominant Internet traffic.

PPSPP [1] is a peer-to-peer based transport protocol for content dissemination, both for on-demand and live content. The protocol is defined by a network of users who participate by forwarding the content to each other via a mesh-like topology. PPSPP is intended to be a built-in cross-platform protocol in the TCP/IP stack, supporting all major operating systems and web browsers.

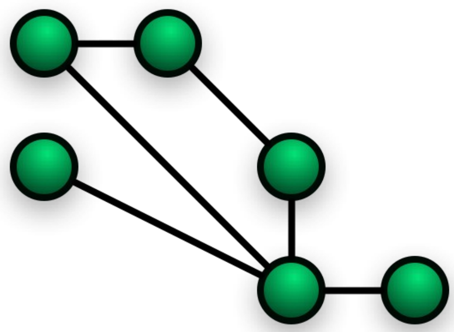

Figure 12: Mesh-based network topology.

Each content has a unique hash id in the overlay network that represents a self-certification mechanism where users only need this hash to receive the data from any available source, while data integrity is checked with Merkle hash trees. The protocol abstracts away the complex underlying connections among network entities providing a clean API layer for applications.

The transport protocol relies on the idea that bandwidth and memory resources have to be used in a proactive way. There is no benefit in keeping these resources unoccupied unless the power budget is critical. Therefore, the implementation is focused on a lightweight footprint, automatic disk space management and non-intrusive congestion protocol.

PPSPP features short start-up times and scalability since only a small amount of information is needed to start fetching and verifying incoming chunks of content. It can use different mechanisms to prevent users not sharing and only downloading the content (free riders). The transport protocol also works with different centralized or distributed peer discovery schemes, and it offers NAT traversal techniques for limited users.

1.6.3.1. Communication messages

The basic unit of communication in the network is the message. Multiple messages are combined into a single datagram for transmission. The following list represents all the message types available:

- HANDSHAKE: contains the initial information of a peer such as protocol options and network id.
- HAVE: includes which chunks of content a peer has available for download.
- ACK: contains the specification of a successfully checked chunk of content.
- DATA: contains the chunk data and its id.
- INTEGRITY: includes the information to verify the integrity of a chunk of content.
- SIGNED_INTEGRITY: includes signed information to verify the integrity of a chunk in live streaming.
- REQUEST: contains the specification of chunks that a peer wants to download.
- CHOKE / UNCHOKE: used to control the reply of REQUEST messages from a peer.
- CANCEL: includes a list of chunks that the peer no longer wants to request.

1 http://datatracker.ietf.org/doc/draft-ietf-ppsp-peer-protocol/

1.6.3.2. Joining the network

Given the scenario [13], user A wants to play video content from the network. To play the video, the user accesses a website containing a video element that has the PPSP URL. The web browser will parse the URL and extract the transport address of a PPSP tracker and swarm ID of the content.

The user A needs to have the hash id of the content and optionally, a list of trackers if there is no decentralized tracking mechanism. Then, user A communicates with a tracker (1) and receives a list of peers already in the network such as user B, C, and D. When user A registers itself in the tracker, it becomes a network peer ready to share content.

User A parses the user list and sends a HANDSHAKE message to all the new discovered users (2). This message includes information regarding protocol version and network id to verify that user A has joined the correct network. Users B and C reply back with a HANDSHAKE message and at least one HAVE message (3). The HAVE messages conveys the chunk availability of both users respectively. User D sends only a HANDSHAKE message in order to not choking A.

1.6.3.3. Data exchange

Following the current scenario [13], user A sends a REQUEST message to user B and C in response to their replies (4). Both messages are disjunct sets of chunks that user A wants to download. Users B and C reply (5) with three messages: INTEGRITY, DATA and HAVE. The INTEGRITY message conveys the hash ids of all the chunks sent in the DATA message in order to check if they are correct. The HAVE message is used by user A to update the chunk availability of users B and C.

When user A processes the messages, it sends back HAVE messages to users B and C with the information of received chunks from all the users (6). In addition, user A sends an ACK message for the received chunks and a REQUEST message for the ones to download. ACK messages are only sent in unreliable transport protocols.

User C discovers that user A received a chunk that user C does not have yet, so its chunk id will be included in a REQUEST message. User D does not send HAVE messages to user A until it will unchoke user A. In this case, user D sends a datagram with HAVE messages to user A to inform it about its chunk availability (*7*). If user B and C decide to choke A, they will stop sending HAVE and DATA messages.

1.6.3.4. Leaving the network

Depending on the underlying transport protocol, users can leave the network by sending leave messages or just stop replying all messages. If the user leaves normally, it should deregister itself from the original tracker.

1.6.3.5. Merkle hash tree

All content managed by PPSPP have a unique hash that is the root in a Merkle hash tree calculated from the content. The Merkle hash tree of a content is divided into N chunks of data with their corresponding hash ids. This self-certifying hash tree allows every peer to detect other peer that inject fake content in the system.

A binary tree is created with enough height that the lowest level in the tree has enough nodes to hold all chunk hashes in the set. Figure [14] shows the tree of a content with 7 chunks of data. The leaves of the tree represents sorted chunks of data starting from the left-most leaf. Remaining leaves not covered by the tree will be set as zero.

The hash values of upper levels in the tree are calculated by concatenating the hash value of the two children in a left-right order and generating the hash of that aggregate. This algorithm ends in a hash value for the root node called root hash.

1.6.3.6. Implementation

The ongoing prototype for the PPSPP transport protocol is available in the GLORIA repository through the official website.

Figure 13: PPSPP handshake protocol.

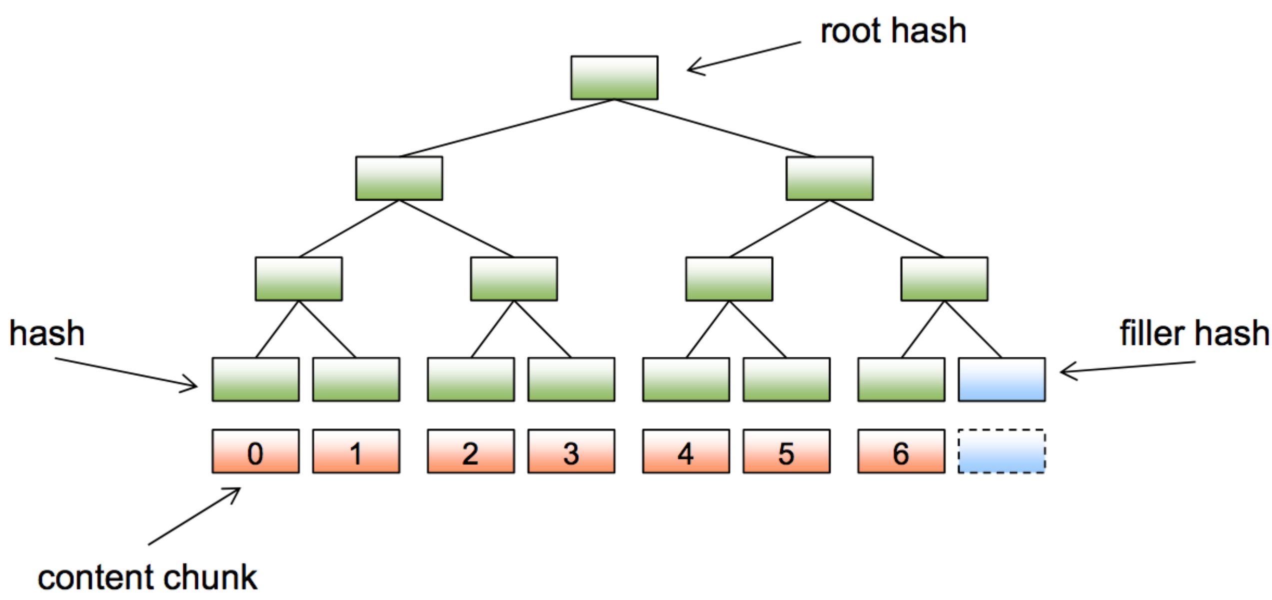

Figure 14: Merkle hash tree.

1.6.4. Infrastructure

The content delivery infrastructure consists of a hybrid solution that combines the benefits of a cloud-based media platform and a decentralized peer-to-peer network. The cloud computing platform forms the stable core of the infrastructure. It offers a flexible solution for rapid changes in the audience size, as well as support multiple configurations for GLORIA users having a variety of fixed and mobile devices on a daily basis. On the other hand, the peer-to-peer overlay network releases overhead in the system and allows GLORIA users to participate actively on each live astronomical event.

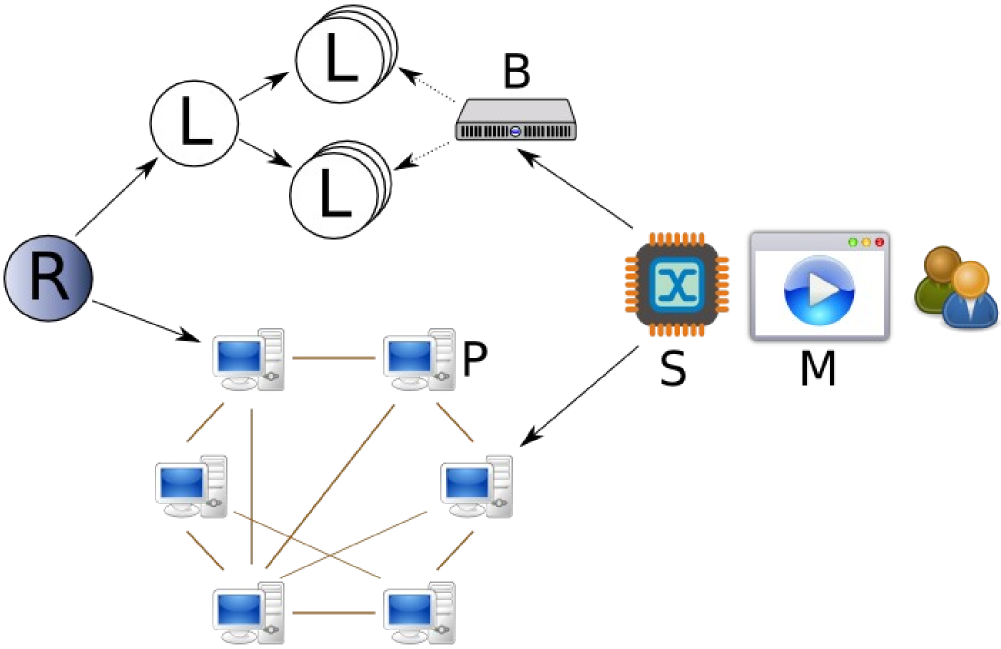

Figure 15: Content delivery infrastructure

The Figure 15 represents the design of the system. The elements of the diagram are defined as follows:

- *Root node* (R): cloud-based streaming server that receives the live output stream from the media production and redirects the stream to a layer of nodes called live repeaters, becoming a seeder.
- *Live repeater* (L): cloud-based streaming server that receives the live media content from the root node and waits for incoming connections from GLORIA users.
- *Load balancer* (B): network device which monitors the status of each live repeater in order to offer high availability to the overall system. Live repeaters send a value indicating its overhead, and the load balancer selects the most healthy live repeater available.
- *Peer* (P): node of the peer-to-peer overlay network. A peer is essentially a p2p-capable GLORIA user.
- *Switcher* (S): software application running on each GLORIA user that performs an algorithm to determine the most suitable option to fetch the live stream for the media player. Initially, the switcher tries to retrieve content from the cloud-based solution. If the cloud-based platform is saturated, the switcher will get the content from the peer-to-peer network. The algorithm is based on the information provided by both the load balancer and the peer-to-peer network.
- *Media player* (M): software application that renders the live output stream.

Finally, the system executes the following steps on each GLORIA user as shown in Figure 15:

1. The switcher and media player are loaded in the GLORIA user's device.
2. The switcher detects if the GLORIA user has a p2p-capable device. If not, go to 7.
3. The switcher checks availability and status of the cloud-based streaming platform.
4. If the status indicates overhead, go to 6.
5. If the switcher does not prioritize the p2p-network,. go to 7.
6. The switcher redirects the media player to the p2p-network. Go to 8.
7. The switcher points to the cloud-based load balancer.
8. The media player renders the live output stream from the selected source.

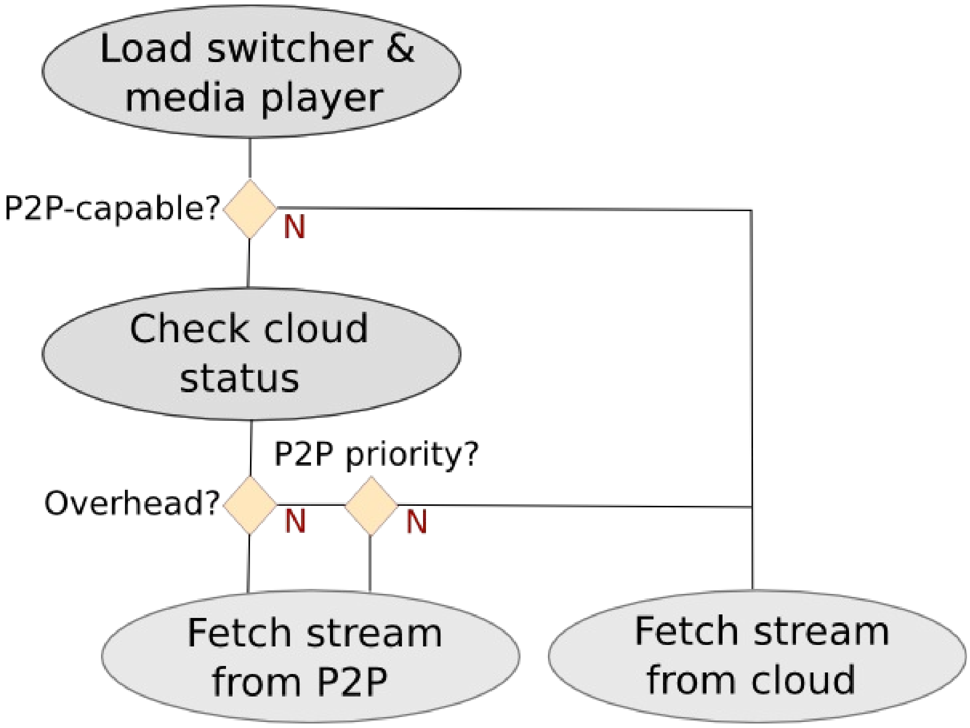

Figure 16: Content delivery flowchart

GLORIA Partners

UPM
Universidad Politécnica de Madrid
SPAIN

ASU-CAS
Astronomical Institute, Academy of Sciences of the Czech Republic
CZECH REPUBLIC

CSIC
Consejo Superior de Investigaciones Científicas
SPAIN

CTU
Czech Technical University in Prague
CZECH REPUBLIC

FZU-CAS
Institute of Physics of the Academy of Sciences of the Czech Republic
CZECH REPUBLIC

IAC
Instituto de Astrofisica de Canarias
SPAIN

INAF
Istituto Nazionale di Astrofisica
ITALY

SAO
Special Astrophysical Observatory of Russian Academy of Sciences
RUSSIA

UCD
University College Dublin
IRELAND

UCH
University of Chile
CHILE

UMA
University of Malaga
SPAIN

UOX
University of Oxford
UNITED KINGDOM

UWAR
Uniwersytet Warszawski
POLAND