

PPSP  
Internet-Draft  
Intended status: Standards Track  
Expires: April 28, 2011

Y. Gu  
N. Zong  
Huawei  
Hui. Zhang  
NEC Labs America.  
Yunfei. Zhang  
China Mobile  
J. Lei  
University of Goettingen  
Gonzalo. Camarillo  
Ericsson  
Yong. Liu  
Polytechnic University  
October 25, 2010

**Survey of P2P Streaming Applications**  
**draft-gu-ppsp-survey-02**

Abstract

This document presents a survey of popular Peer-to-Peer streaming applications on the Internet. We focus on the Architecture and Peer Protocol/Tracker Signaling Protocol description in the presentation, and study a selection of well-known P2P streaming systems, including Joost, PPlive, and other popular existing systems. Through the survey, we summarize a common P2P streaming process model and the correspondent signaling process for P2P Streaming Protocol standardization.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 28, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction . . . . .</a>	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">Terminologies and concepts . . . . .</a>	<a href="#">3</a>
<a href="#">3.</a>	<a href="#">Survey of P2P streaming system . . . . .</a>	<a href="#">4</a>
<a href="#">3.1.</a>	<a href="#">Mesh-based P2P streaming systems . . . . .</a>	<a href="#">4</a>
<a href="#">3.1.1.</a>	<a href="#">Joost . . . . .</a>	<a href="#">4</a>
<a href="#">3.1.2.</a>	<a href="#">Octoshape . . . . .</a>	<a href="#">6</a>
<a href="#">3.1.3.</a>	<a href="#">PPLive . . . . .</a>	<a href="#">8</a>
<a href="#">3.1.4.</a>	<a href="#">Zattoo . . . . .</a>	<a href="#">9</a>
<a href="#">3.1.5.</a>	<a href="#">PPStream . . . . .</a>	<a href="#">10</a>
<a href="#">3.1.6.</a>	<a href="#">SopCast . . . . .</a>	<a href="#">11</a>
<a href="#">3.1.7.</a>	<a href="#">TVants . . . . .</a>	<a href="#">11</a>
<a href="#">3.2.</a>	<a href="#">Tree-based P2P streaming systems . . . . .</a>	<a href="#">12</a>
<a href="#">3.2.1.</a>	<a href="#">PeerCast . . . . .</a>	<a href="#">12</a>
<a href="#">3.2.2.</a>	<a href="#">Conviva . . . . .</a>	<a href="#">14</a>
<a href="#">4.</a>	<a href="#">A common P2P Streaming Process Model . . . . .</a>	<a href="#">15</a>
<a href="#">5.</a>	<a href="#">Security Considerations . . . . .</a>	<a href="#">16</a>
<a href="#">6.</a>	<a href="#">Acknowledgments . . . . .</a>	<a href="#">16</a>
<a href="#">7.</a>	<a href="#">Informative References . . . . .</a>	<a href="#">16</a>
	<a href="#">Authors' Addresses . . . . .</a>	<a href="#">18</a>



## **1. Introduction**

Toward standardizing the signaling protocols used in today's Peer-to-Peer (P2P) streaming applications, we surveyed several popular P2P streaming systems regarding their architectures and signaling protocols between peers, as well as, between peers and trackers. The studied P2P streaming systems, running worldwide or domestically, include such as PPLive, Joost, Cybersky-TV, and Octoshape. This document does not intend to cover all design options of P2P streaming applications. Instead, we choose a representative set of applications and focus on the respective signaling characteristics of each kind. Through the survey, we generalize a common streaming process model from those P2P streaming systems, and summarize the companion signaling process as the base for P2P Streaming Protocol (PPSP) standardization.

## **2. Terminologies and concepts**

**Chunk:** A chunk is a basic unit of partitioned streaming media, which is used by a peer for the purpose of storage, advertisement and exchange among peers [Sigcomm:P2P streaming].

**Content Distribution Network (CDN) node:** A CDN node refers to a network entity that usually is deployed at the network edge to store content provided by the original servers, and serves content to the clients located nearby topologically.

**Live streaming:** The scenario where all clients receive streaming content for the same ongoing event. The lags between the play points of the clients and that of the streaming source are small..

**P2P cache:** A P2P cache refers to a network entity that caches P2P traffic in the network, and either transparently or explicitly distributes content to other peers.

**P2P streaming protocols:** P2P streaming protocols refer to multiple protocols such as streaming control, resource discovery, streaming data transport, etc. which are needed to build a P2P streaming system.

**Peer/PPSP peer:** A peer/PPSP peer refers to a participant in a P2P streaming system. The participant not only receives streaming content, but also stores and uploads streaming content to other participants.

**PPSP protocols:** PPSP protocols refer to the key signaling protocols among various P2P streaming system components, including the tracker



and peers.

Swarm: A swarm refers to a group of clients (i.e. peers) sharing the same content (e.g. video/audio program, digital file, etc) at a given time.

Tracker/PPSP tracker: A tracker/PPSP tracker refers to a directory service which maintains the lists of peers/PPSP peers storing chunks for a specific channel or streaming file, and answers queries from peers/PPSP peers.

Video-on-demand (VoD): A kind of application that allows users to select and watch video content on demand

### **3. Survey of P2P streaming system**

In this section, we summarize some existing P2P streaming systems. The construction techniques used in these systems can be largely classified into two categories: tree-based and mesh-based structures.

Tree-based structure: Group members self-organize into a tree structure, based on which group management and data delivery is performed. Such structure has small maintenance cost and good scalability and can be easily implemented. However, it may result in low bandwidth usage and less reliability.

Mesh-based structure: In contrast to tree-based structure, a mesh uses multiple links between any two nodes. Thus, the reliability of data transmission is relatively high. Nevertheless, the cost of maintaining such mesh is much larger than that of a tree.

#### **3.1. Mesh-based P2P streaming systems**

##### **3.1.1. Joost**

Joost announced to give up P2P technology on its desktop version last year, though it introduced a flash version for browsers and iPhone application. The key reason why Joost shut down its desktop version is probably the legal issues of provided media content. However, as one of the most popular P2P VoD application in the past years, it's worthwhile to understand how Joost works. The peer management and data transmission in Joost mainly relies on mesh-based structure.

The three key components of Joost are servers, super nodes and peers. There are five types of servers: Tracker server, Version server, Backend server, Content server and Graphics server. The architecture of Joost system is shown in Figure 1.



First, we introduce the functionalities of Joost's key components through three basic phases. Then we will discuss the Peer protocol and Tracker protocol of Joost.

**Installation:** Backend server is involved in the installation phase. Backend server provides peer with an initial channel list in a SQLite file. No other parameters, such as local cache, node ID, or listening port, are configured in this file.

**Bootstrapping:** In case of a newcomer, Tracker server provides several super node addresses and possibly some content server addresses. Then the peer connects Version server for the latest software version. Later, the peer starts to connect some super nodes to obtain the list of other available peers and begins streaming video contents. Different from Skype [skype], super nodes in Joost only deal with control and peer management traffic. They do not relay/forward any media data.

**Channel switching:** Super nodes are responsible for redirecting clients to content server or peers.

Peers communicate with servers over HTTP/HTTPS and with super nodes/other peers over UDP.

**Tracker Protocol:** Because super nodes here are responsible for providing the peerlist/content servers to peers, protocol used between tracker server and peers is rather simple. Peers get the addresses of super nodes and content servers from Tracker Server over HTTP. After that, Tracker server will not appear in any stage, e.g. channel switching, VoD interaction. In fact, the protocol spoken between peers and super nodes is more like what we normally called "Tracker Protocol". It enables super nodes to check peer status, maintain peer lists for several, if not all, channels. It provides peer list/content servers to peers. Thus, in the rest of this section, when we mention Tracker Protocol, we mean the one used between peers and super nodes.

Peers will communicate with super nodes in some scenarios using Tracker Protocol.

1. When a peer starts Joost software, after the installation and bootstrapping, the peer will communicate with one or several super nodes to get a list of available peers/content servers.
2. For on-demand video functions, super nodes periodically exchange small UDP packets for peer management purpose.
3. When switching between channels, peers contact super nodes and





the latter help the peers find available peers to fetch the requested media data.

Peer Protocol: The following investigations are mainly motivated from [Joost- experiment ], in which a data-driven reverse-engineer experiments are performed. We omitted the analysis process and directly show the conclusion. Media data in Joost is split into chunks and then encrypted. Each chunk is packetized with about 5-10 seconds of video data. After receiving peer list from super nodes, a peer negotiates with some or, if necessary, all of the peers in the list to find out what chunks they have. Then the peer makes decision about from which peers to get the chunks. No peer capability information is exchanged in the Peer Protocol.

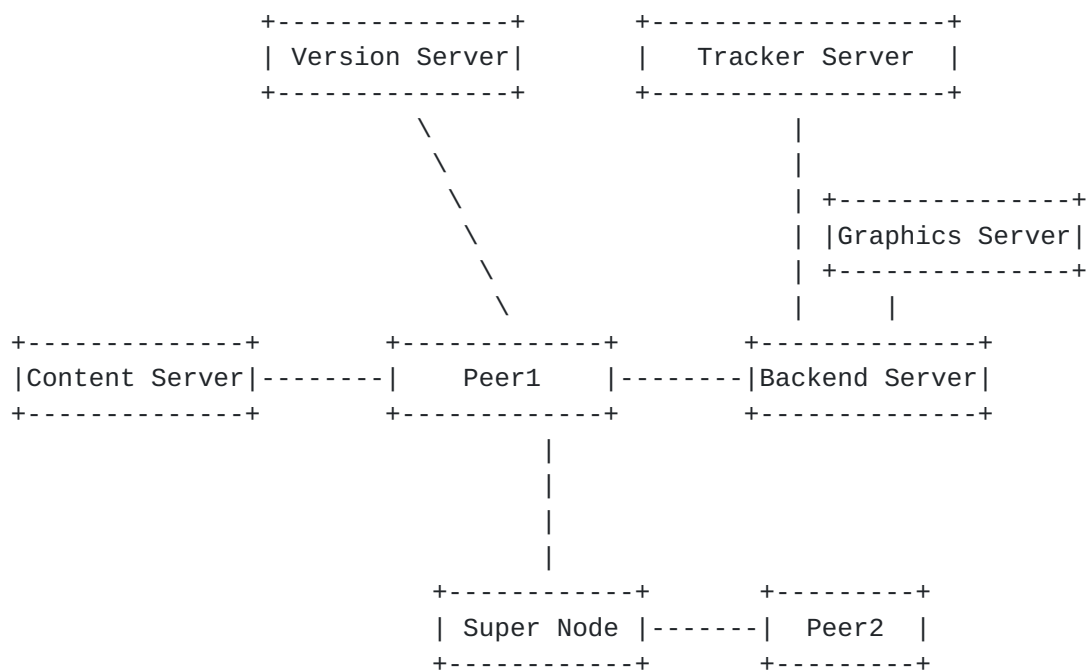


Figure 1, Architecture of Joost system

### 3.1.2. Octoshape

CNN has been working with a P2P Plug-in, from a Denmark-based company Octoshape, to broadcast its living streaming. Octoshape helps CNN serve a peak of more than a million simultaneous viewers. It has also provided several innovative delivery technologies such as loss resilient transport, adaptive bit rate, adaptive path optimization and adaptive proximity delivery. Figure 2 depicts the architecture of the Octoshape system.

Octoshape maintains a mesh overlay topology. Its overlay topology maintenance scheme is similar to that of P2P file-sharing applications, such as BitTorrent. There is no Tracker server in



Octoshape, thus no Tracker Protocol is required. Peers obtain live streaming from content servers and peers over Octoshape Protocol. Several data streams are constructed from live stream. No data streams are identical and any number  $K$  of data streams can reconstruct the original live stream. The number  $K$  is based on the original media playback rate and the playback rate of each data stream. For example, a 400Kbit/s media is split into four 100Kbit/s data streams, and then  $k = 4$ . Data streams are constructed in peers, instead of Broadcast server, which release server from large burden. The number of data streams constructed in a particular peer equals the number of peers downloading data from the particular peer, which is constrained by the upload capacity of the particular peer. To get the best performance, the upload capacity of a peer should be larger than the playback rate of the live stream. If not, an artificial peer may be added to deliver extra bandwidth.

Each single peer has an address book of other peers who is watching the same channel. A Standby list is set up based on the address book. The peer periodically probes/asks the peers in the standby list to be sure that they are ready to take over if one of the current senders stops or gets congested. [[Octoshape](#)]

Peer Protocol: The live stream is firstly sent to a few peers in the network and then be spread to the rest. When a peer joins a channel, it notifies all the other peers about its presence over Peer Protocol, which will drive the others to add it into their address books. Although [[Octoshape](#)] declares that each peer records all the peers joining the channel, we suspect that not all the peers are recorded, considering the notification traffic will be large and peers will be busy with recording when a popular program starts in a channel and lots of peers switch to this channel. Maybe some geographic or topological neighbors are notified and the peer gets its address book from these neighbors.

Peer Protocol: The live stream is firstly sent to a few peers in the network and then spread to the rest of the network. When a peer joins a channel, it notifies all the other peers about its presence using Peer Protocol, which will drive the others to add it into their address books. Although [[Octoshape](#)] declares that each peer records all the peers joining the channel, we suspect that not all the peers are recorded, considering the notification traffic will be large and peers will be busy with recording when a popular program starts in a channel and lots of peers switch to this channel. Maybe some geographic or topological neighbors are notified and the peer gets its address book from these nearby neighbors.

The peer sends requests to some selected peers for the live stream and the receivers answers OK or not according to their upload



capacity. The peer continues sending requests to peers until it finds enough peers to provide the needed data streams to redisplay the original live stream. The details of Octoshape are (not?) disclosed yet, we hope someone else can provide much specific information.

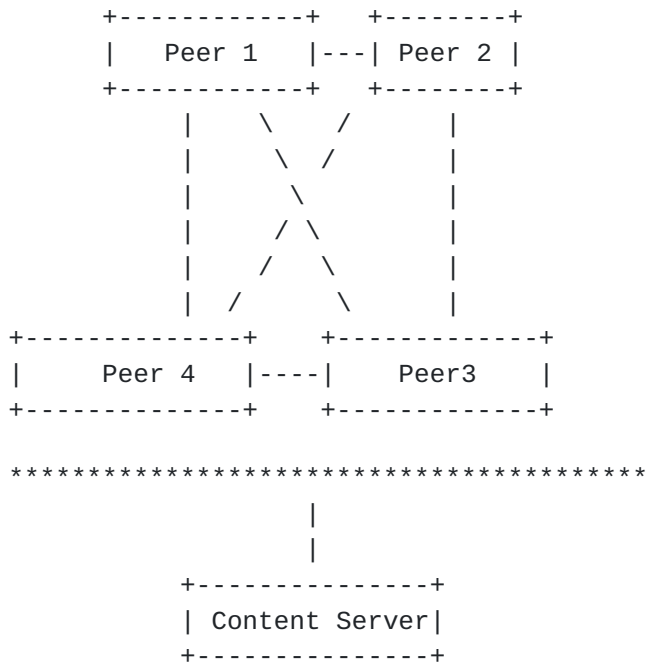


Figure 2, Architecture of Octoshape system

### 3.1.3. PPLive

PPLive is one of the most popular P2P streaming software in China. It has two major communication protocols. One is Registration and peer discovery protocol, i.e. Tracker Protocol, and the other is P2P chunk distribution protocol, i.e. Peer Protocol. Figure 3 shows the architecture of PPLive.

**Tracker Protocol:** First, a peer gets the channel list from the Channel server, in a way similar to that of Joost. Then the peer chooses a channel and asks the Tracker server for the peerlist of this channel.

**Peer Protocol:** The peer contacts the peers in its peerlist to get additional peerlists, which are aggregated with its existing list. Through this list, peers can maintain a mesh for peer management and data delivery.

For the video-on-demand (VoD) operation, because different peers watch different parts of the channel, a peer buffers up to a few minutes worth of chunks within a sliding window to share with each



others. Some of these chunks may be chunks that have been recently played; the remaining chunks are chunks scheduled to be played in the next few minutes. Peers upload chunks to each other. To this end, peers send to each other "buffer-map" messages; a buffer-map message indicates which chunks a peer currently has buffered and can share. The buffer-map message includes the offset (the ID of the first chunk), the length of the buffer map, and a string of zeroes and ones indicating which chunks are available (starting with the chunk designated by the offset). PPLive transfer Data over UDP.

#### Video Download Policy of PPLive

1 Top ten peers contribute to a major part of the download traffic. Meanwhile, the top peer session is quite short compared with the video session duration. This would suggest that PPLive gets video from only a few peers at any given time, and switches periodically from one peer to another;

2 PPLive can send multiple chunk requests for different chunks to one peer at one time;

PPLive maintains a constant peer list with relatively small number of peers. [P2PIPTV-measuring]

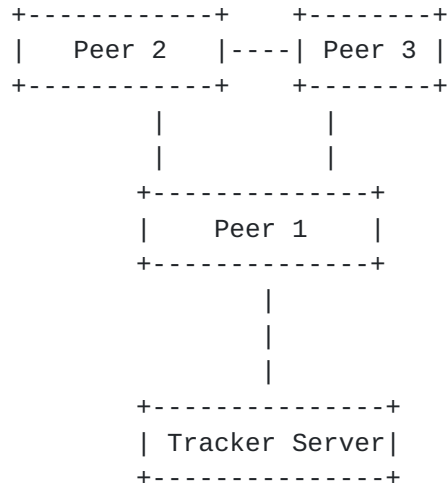


Figure 3, Architecture of PPLive system

#### 3.1.4. Zattoo

Zattoo is P2P live streaming system which serves over 3 million registered users over European countries [Zattoo]. The system delivers live streaming using a receiver-based, peer-division multiplexing scheme. Zattoo reliably streams media among peers using the mesh structure.





Figure 4 depicts a typical procedure of single TV channel carried over Zattoo network. First, Zattoo system broadcasts live TV, captured from satellites, onto the Internet. Each TV channel is delivered through a separate P2P network.

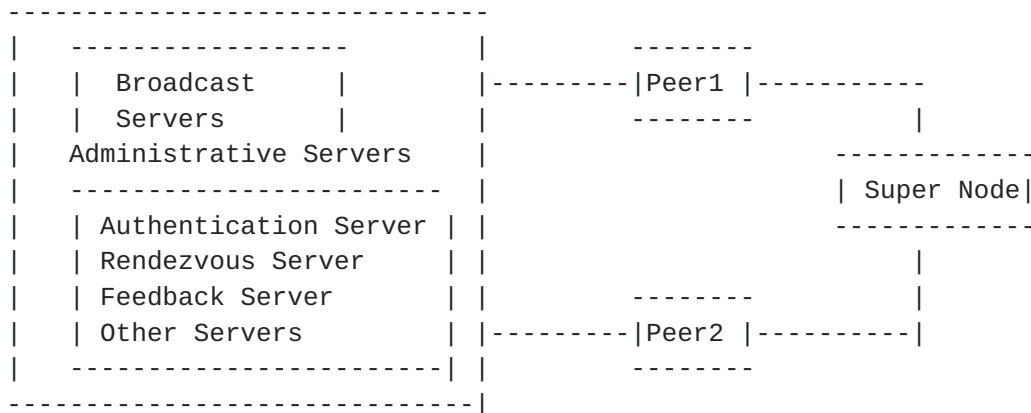


Figure 4, Basic architecture of Zattoo system

Tracker(Rendezvous Server) Protocol: In order to receive the signal the requested channel, registered users are required to be authenticated through Zattoo Authentication Server. Upon authentication, users obtain a ticket with specific lifetime. Then, users contact Rendezvous Server with the ticket and identify of interested TV channel. In return, the Rendezvous Server sends back a list joined peers carrying the channel.

Peer Protocol: Similar to aforementioned procedures in Joost, PPLive, a new Zattoo peer requests to join an existing peer among the peer list. Upon the availability of bandwidth, requested peer decides how to multiplex a stream onto its set of neighboring peers. When packets arrive at the peer, sub-streams are stored for reassembly constructing the full stream.

Note Zattoo relies on Bandwidth Estimation Server to initially estimate the amount of available uplink bandwidth at a peer. Once a peer starts to forward substream to other peers, it receives QoS feedback from other receivers if the quality of sub-stream drops below a threshold.

### **3.1.5. PPStream**

The system architecture and working flows of PPStream is similar to PPLive. PPStream transfers data using mostly TCP, only occasionally UDP.

Video Download Policy of PPStream



1 Top ten peers do not contribute to a large part of the download traffic. This would suggest that PPStream gets the video from many peers simultaneously, and its peers have long session duration;

2 PPStream does not send multiple chunk requests for different chunks to one peer at one time;

PPStream maintains a constant peer list with relatively large number of peers. [P2PIPTV-measuring]

#### **3.1.6. SopCast**

The system architecture and working flows of SopCast is similar to PPLive. SOPCast transfer data mainly using UDP, occasionally TCP;

Top ten peers contribute to about half of the total download traffic. SOPCast's download policy is similar to PPLive's policy in that it switches periodically between provider peers. However, SOPCast seems to always need more than one peer to get the video, while in PPLive a single peer could be the only video provider;

SOPCast's peer list can be as large as PPStream's peer list. But SOPCast's peer list varies over time. [P2PIPTV-measuring]

#### **3.1.7. TVants**

The system architecture and working flows of TVants is similar to PPLive. TVants is more balanced between TCP and UDP in data transmission;

The system architecture and working flows of TVants is similar to PPLive. TVants is more balanced between TCP and UDP in data transmission;

TVants' peer list is also large and varies over time. [P2PIPTV-measuring]

We extract the common Main components and steps of PPLive, PPStream, SopCast and TVants, which is shown in Figure 5.



PeerCast adopts a Tree structure. The architecture of PeerCast is shown in Figure 6.



Peers in one channel construct the Broadcast Tree and the Broadcast server is the root of the Tree. A Tracker can be implemented independently or merged in the Broadcast server. Tracker in Tree based P2P streaming application selects the parent nodes for those new peers who join in the Tree. A Transfer node in the Tree receives and transfers data simultaneously.

Peer Protocol: The peer joins a channel and gets the broadcast server address. First of all, the peer sends a request to the server, and the server answers OK or not according to its idle capability. If the broadcast server has enough idle capability, it will include the peer in its child-list. Otherwise, the broadcast server will choose at most eight nodes of its children and answer the peer. The peer records the nodes and contacts one of them, until it finds a node that can server it.

In stead of requesting the channel by the peer, a Transfer node pushes live stream to its children, which can be a transfer node or a receiver. A node in the tree will notify its status to its parent periodically, and the latter will update its child-list according to the received notifications.

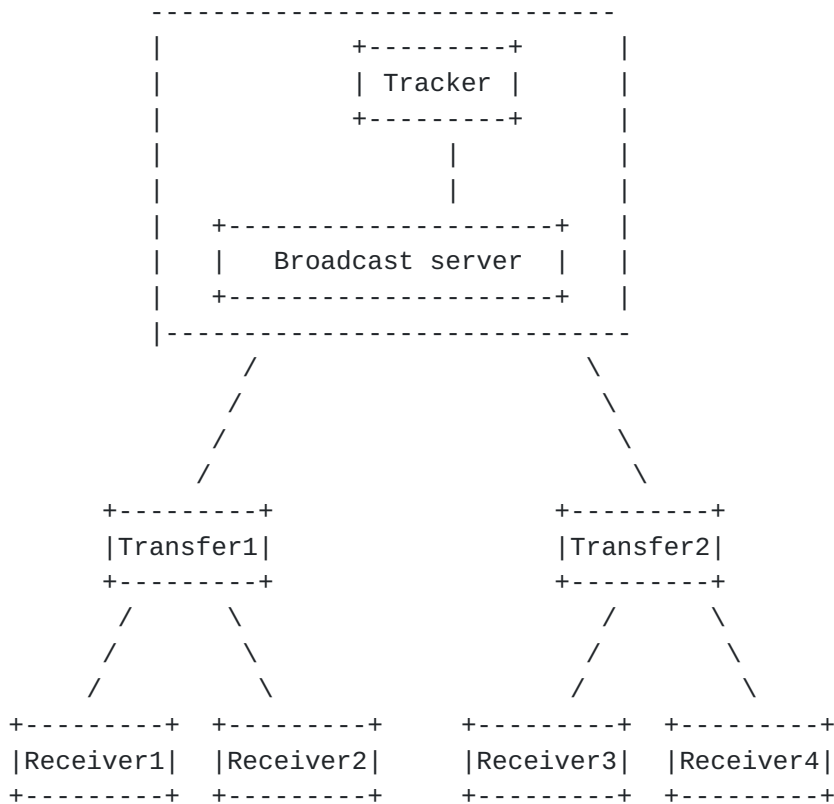


Figure 6, Architecture of PeerCast system





### **3.2.2. Conviva**

Conviva[TM][conviva] is a real-time media control platform for Internet multimedia broadcasting. For its early prototype, End System Multicast (ESM) [[ESM04](#)] is the underlying networking technology on organizing and maintaining an overlay broadcasting topology. Next we present the overview of ESM. ESM adopts a Tree structure. The architecture of ESM is shown in Figure 7.

ESM has two versions of protocols: one for smaller scale conferencing apps with multiple sources, and the other for larger scale broadcasting apps with Single source. We focus on the latter version in this survey.

ESM maintains a single tree for its overlay topology. Its basic functional components include two parts: a bootstrap protocol, a parent selection algorithm, and a light-weight probing protocol for tree topology construction and maintenance; a separate control structure decoupled from tree, where a gossip-like algorithm is used for each member to know a small random subset of group members; members also maintain pathes from source.

Upon joining, a node gets a subset of group membership from the source (the root node); it then finds parent using a parent selection algorithm. The node uses light-weight probing heuristics to a subset of members it knows, and evaluates remote nodes and chooses a candidate parent. It also uses the parent selection algorithm to deal with performance degradation due to node and network churns.

ESM Supports for NATs. It allows NATs to be parents of public hosts, and public hosts can be parents of all hosts including NATs as children.



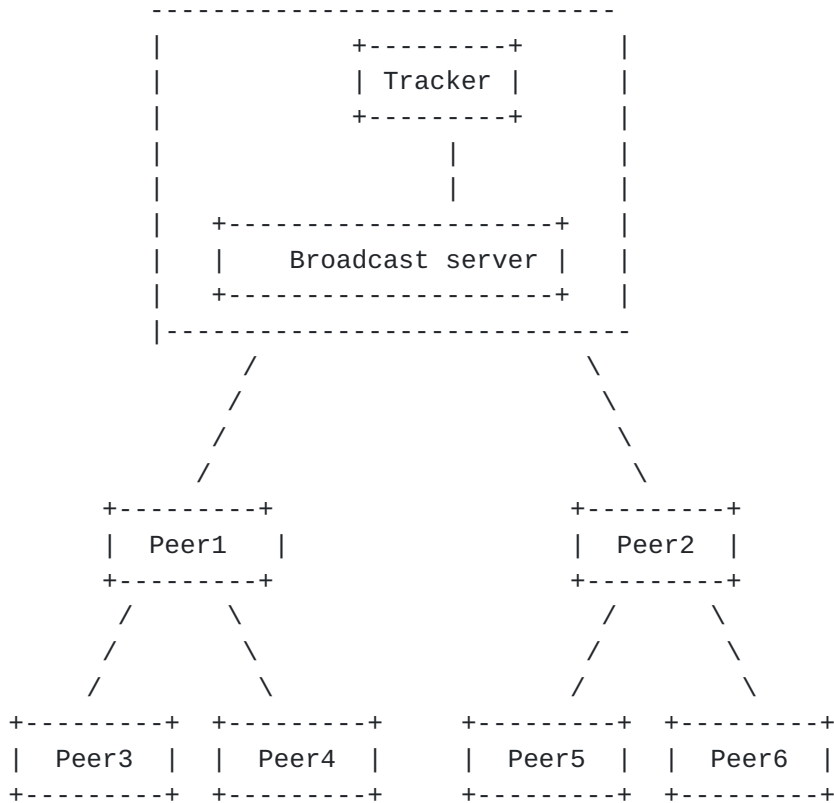


Figure 7, Architecture of ESM system

#### 4. A common P2P Streaming Process Model

As shown in Figure 8, a common P2P streaming process can be summarized based on [Section 3](#):

- 1) When a peer wants to receive streaming content:
  - 1.1) Peer acquires a list of peers/parent nodes from the tracker.
  - 1.2) Peer exchanges its content availability with the peers on the obtained peer list, or requests to be adopted by the parent nodes.
  - 1.3) Peer identifies the peers with desired content, or the available parent node.
  - 1.4) Peer requests for the content from the identified peers, or receives the content from its parent node.



2) When a peer wants to share streaming content with others:

2.1) Peer sends information to the tracker about the swarms it belongs to, plus streaming status and/or content availability.

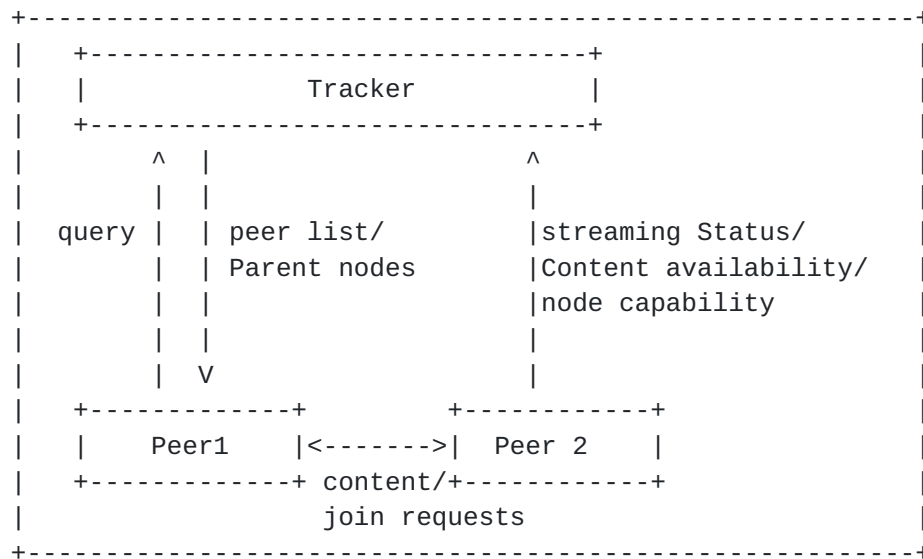


Figure 8, A common P2P streaming process model

The functionality of Tracker and data transfer in Mesh-based application and Tree-based is a little different. In the Mesh-based applications, such as Joost and PPLive, Tracker maintains the lists of peers storing chunks for a specific channel or streaming file. It provides peer list for peers to download from, as well as upload to, each other. In the Tree-based applications, such as PeerCast and Canviva, Tracker directs new peers to find parent nodes and the data flows from parent to child only.

## 5. Security Considerations

This document does not consider security issues. It follows the security consideration in [[draft-zhang-ppsp-problem-statement](#)].

## 6. Acknowledgments

We would like to acknowledge Jiang xingfeng for providing good ideas for this document.

## 7. Informative References

[PPLive] "www.pplive.com".



- [PPStream] "www.ppstream.com".
- [CNN] "www.cnn.com".
- [OctoshapeWeb] "www.octoshape.com".
- [Joost-Experiment] Lei, Jun, et al., "An Experimental Analysis of Joost Peer-to-Peer VoD Service".
- [Sigcomm\_P2P\_Streaming] Huang, Yan, et al., "Challenges, Design and Analysis of a Large-scale P2P-VoD System", 2008.
- [Octoshape] Alstrup, Stephen, et al., "Introducing Octoshape-a new technology for large-scale streaming over the Internet".
- [Zattoo] "http: //zattoo.com/".
- [Conviva] "http://www.rinera.com/".
- [ESM04] Zhang, Hui., "End System Multicast, <http://www.cs.cmu.edu/~hzhang/Talks/ESMPrinceton.pdf>", May .
- [Survey] Liu, Yong, et al., "A survey on peer-to-peer video streaming systems", 2008.
- [[draft-zhang-alto-traceroute-00](http://www.ietf.org/internet-draft/draft-zhang-alto-traceroute-00.txt)] "www.ietf.org/internet-draft/[draft-zhang-alto-traceroute-00.txt](http://www.ietf.org/internet-draft/draft-zhang-alto-traceroute-00.txt)".
- [P2PStreamingSurvey] Zong, Ning, et al., "Survey of P2P Streaming", Nov. 2008.
- [P2PIPTV\_measuring] Silverston, Thomas, et al., "Measuring P2P IPTV Systems".
- [Challenge] Li, Bo, et al., "Peer-to-Peer Live Video Streaming on the Internet: Issues, Existing Approaches, and Challenges", June 2007.





Authors' Addresses

Gu Yingjie  
Huawei  
Baixia Road No. 91  
Nanjing, Jiangsu Province 210001  
P.R.China

Phone: +86-25-84565868  
Fax: +86-25-84565888  
Email: guyingjie@huawei.com

Zong Ning  
Huawei  
Baixia Road No. 91  
Nanjing, Jiangsu Province 210001  
P.R.China

Phone: +86-25-84565866  
Fax: +86-25-84565888  
Email: zongning@huawei.com

Hui Zhang  
NEC Labs America.

Email: huizhang@nec-labs.com

Zhang Yunfei  
China Mobile

Email: zhangyunfei@chinamobile.com

Lei Jun  
University of Goettingen

Phone: +49 (551) 39172032  
Email: lei@cs.uni-goettingen.de

Gonzalo Camarillo  
Ericsson

Email: Gonzalo.Camarillo@ericsson.com



Liu Yong  
Polytechnic University

Email: [yongliu@poly.edu](mailto:yongliu@poly.edu)