PPSP
Internet-Draft
Intended status: Informational
Expires: April 30, 2015

Y. Gu
Unaffiliated
N. Zong, Ed.
Huawei
Y. Zhang
Coolpad
China Mobile
F. Piccolo
Cisco
S. Duan
CATR

October 27, 2014

Survey of P2P Streaming Applications
draft-ietf-ppsp-survey-09

Abstract

This document presents a survey of some of the most popular Peer-to-Peer (P2P) streaming applications on the Internet. The main selection criteria have been popularity and availability of information on operation details at writing time. In doing this, selected applications are not reviewed as a whole, but they are reviewed with main focus on the signaling and control protocol used to establish and maintain overlay connections among peers and to advertise and download streaming content.

Table of Contents

1. Introduction

An ever-increasing number of multimedia streaming systems have been adopting Peer-to-Peer (P2P) paradigm to stream multimedia audio and video contents from a source to a large number of end users. This is the reference scenario of this document, which presents a survey of some of the most popular P2P streaming applications available on the nowadays Internet.

The presented survey does not aim at being exhaustive. Reviewed applications have indeed been selected mainly based on their popularity and on the information publicly available on P2P operation details at writing time. In addition, the provided descriptions may sometimes appear inhomogeneous from the detail level point of view, but this always depends on the amount of available information at writing time.

In addition, the selected applications are not reviewed as a whole, but they are reviewed with main focus on signaling and control protocols used to construct and maintain the overlay connections among peers and to advertise and download multimedia content. More precisely, we assume throughout the document the high level system model reported in Figure 1.

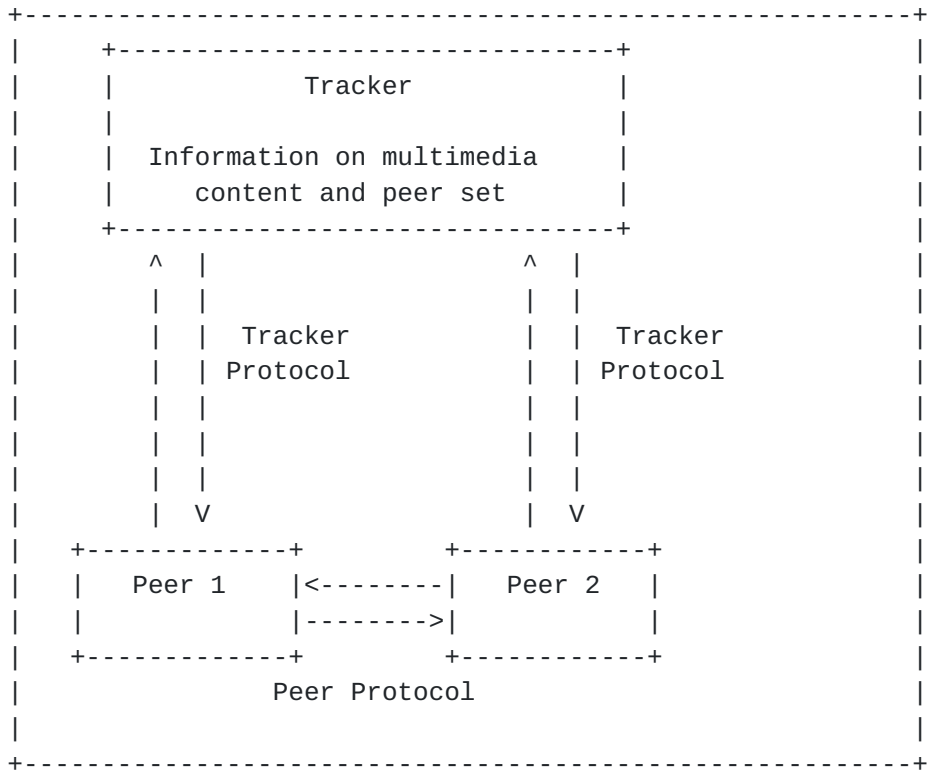

Figure 1, High level architecture of P2P streaming systems assumed as reference model througout the document

As Figure 1 shows, it is possible to identify in every P2P streaming system two main types of entity: peers and trackers. Peers represent end users, which join the system dynamically to send and receive streamed media content, whereas trackers represent well-known nodes, which are stably connected to the system and provide peers with metadata information about the streamed content and the set of active

peers. According to this model, it is possible to distinguish between two different control/signaling protocols:

- -the "tracker protocol" for the interaction between trackers and peer;

- -the "peer protocol" for the interaction between peers.

Hence, whenever possible, we always try to identify tracker and peer protocols and we provide the corresponding details.

This document is organized as follows. Section 2 introduces terminology and concepts used throughout the current survey. Since overlay topology built on connections among peers impacts some aspects of tracker and peer protocols, Section 3 classifies P2P streaming applications according to the overlay topology: mesh-based, tree-based and hybrid. Then, Section 4 presents some of the most popular mesh-based P2P streaming applications: Octoshape, PPLive, Zattoo, PPStream, Tribler, QQLive. Likewise, Section 5 presents End System Multicast as example of tree-based P2P streaming applications, whereas Section 6 presents New Coolstreaming as example of hybrid-topology P2P streaming application. Finally, Section 7 provides some security considerations.

2. Terminologies and concepts

Reader is referred to RFC 6972 [RFC6972] for concepts such as chunk, live streaming, video-on-demand (VOD), peer, tracker, swarm, which will be extensively used throughout the document.

In addition, reader can refer to this section for the following concepts.

CHANNEL: A CHANNEL denotes a TV channel from which live streaming content is transmitted in a P2P streaming application.

PEER PROTOCOL: PEER PROTOCOL denotes the control and signaling protocol for the interaction among peers.

PULL: PULL denotes the transmission of multimedia content that is initiated by receiving peers.

PUSH: PUSH denotes the transmission of multimedia content that is not initiated by receiving peers.

TRACKER PROTOCOL: TRACKER PROTOCOL denotes the control and signaling protocol for the interaction among peers and trackers.

3. Classification of P2P Streaming Applications Based on Overlay Topology

Depending on the topology of overlay connections among peers, it is possible to distinguish among the following general types of P2P streaming applications:

-mesh-based: peers are organized in a randomly connected overlay network, and multimedia content delivery is pull-based. This is the reason why these systems are also referred to as "data-driven". Due to their unstructured nature, mesh-based P2P streaming applications are very resilient with respect to peer churn and guarantee high network resource utilization. On the other side, the cost to maintain overlay topology may limit performance in terms of delay, and pull-based data delivery calls for large size buffers to store chunks;

-tree-based: peers are organized to form a tree-shape overlay network rooted at the streaming source, and multimedia content delivery is push-based. Peers that forward data are called parent nodes, and peers that receive it are called children nodes. Due to their structured nature, tree-based P2P streaming applications guarantee both topology maintenance at very low cost and good delay performance. On the other side, they are not very resilient to peer churn, that may be very high in a P2P environment;

-hybrid: this category includes all the P2P applications that cannot be classified as simply mesh-based or tree-based and present characteristics of both mesh-based and tree-based categories.

4. Mesh-based P2P Streaming Applications

In mesh-based P2P streaming application peers self-organize in a randomly connected overlay graph where each peer interacts with a limited subset of other peers (neighbors) and explicitly requests chunks it needs (pull-based or data-driven delivery). This type of content delivery may be associated with high overhead, not only because peers formulate requests in order to download chunks they need, but also because in some applications peers exchange chunk availability information in form of buffer-maps (a sort of bit maps with a bit "1" in correspondence of chunks stored in the local buffer). On the one side, the main advantage of this kind of applications lies in that a peer does not rely on a single peer for retrieving multimedia content. Hence, these applications are very resilient to peer churn. On the other side, overlay connections are highly dynamic and not persistent (being driven by content availability), and this makes content distribution efficiency

unpredictable. In fact, different chunks may be retrieved via different network paths, and this may imply for end users playback quality degradation ranging from low bit rates to long start-up delays, to frequent playback freezes. Moreover, peers have to maintain large buffers to increase the probability of satisfying chunk requests received by neighbors.

4.1. Octoshape

Octoshape [Octoshape] is a P2P plug-in that has been realized by the homonym Danish company and has become popular for being used by CNN [CNN] to broadcast live streaming content. Octoshape helps indeed CNN serve a peak of more than a million simultaneous viewers thanks not only to the P2P content distribution paradigm, but also to 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.

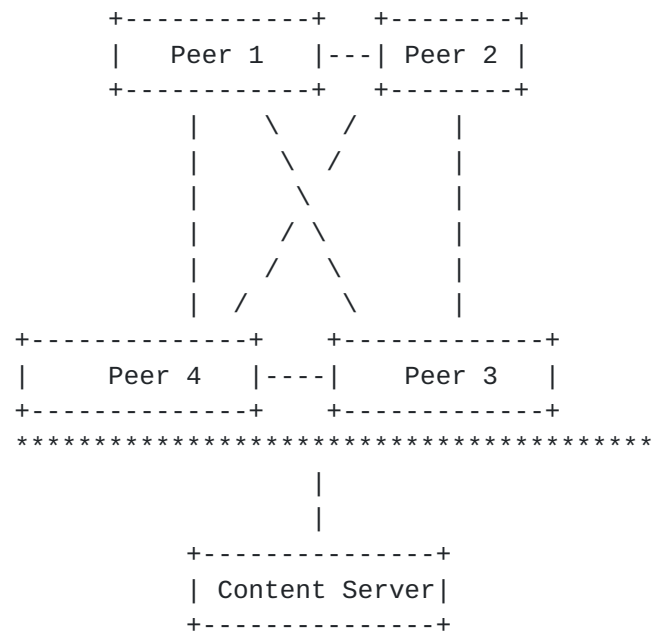

Figure 2, Architecture of Octoshape system

As it can be seen from the picture, there are no trackers and consequently no tracker protocol is necessary. The content server plays indeed the role of tracker and transmits the information on peers that already joined the channel in form of metadata when streaming the live content.

As regards the peer protocol, each peer maintains a sort of Address Book with the information necessary to contact other peers who are watching the same channel.

Regarding the data distribution strategy, in the Octoshape solution the original stream is split into a number K of smaller equal-sized data streams, but a number N > K of unique data streams are actually constructed, in such a way that a peer receiving any K of the N available data streams is able to play the original stream. For instance, if the original live stream is a 400 kbit/sec signal, for K=4 and N=12, 12 unique data streams are constructed, and a peer that downloads any 4 of the 12 data streams is able to play the live stream. In this way, each peer sends requests of data streams to some selected peers, and it receives positive/negative answers depending on availability of upload capacity at requested peers. In case of negative answers, a peer continues sending requests until it finds K peers willing to upload the minimum number of data streams needed to display the original live stream. This allows a flexible use of bandwidth at end users. In fact, since the original stream is split into smaller data streams, a peer that does not have enough upload capacity to transmit the original whole stream can transmit a number of smaller data streams that fits its actual upload capacity.

In order to mitigate the impact of peer loss, the address book is also used at each peer to derive the so called Standby List, which Octoshape peers use to probe other peers and be sure that they are ready to take over if one of the current senders leaves or gets congested.

Finally, in order to optimize bandwidth utilization, Octoshape leverages peers within a network to minimize external bandwidth usage and to select the most reliable and "closest" source to each viewer. It also chooses the best matching available codecs and players, and it scales bit rate up and down according to the available Internet connection.

4.2. PPLive

PPLive [PPLive] was first developed in Huazhong University of Science and Technology in 2004, and it is one of the earliest and most popular P2P streaming software in China. To give an idea, PPLive website served 50 millions visitors during the Beijing 2008 Olympics opening ceremony, and the dedicated Olympics channel attracted 221 millions of viewers in two weeks.

Even though PPLive was renamed to PPTV in 2010, we continue using the old name PPLive throughout this document.

PPLive system includes the following main components:

-video streaming server, that plays the role of source of video content and copes with content coding issues;

-peer, also called node or client, that is PPLive entity downloading video content from other peers and uploading video content to other peers

-channel server, that provides the list of available channels (live TV or VoD content) to a PPLive peer, as soon as the peer joins the system;

-tracker server, that provides a PPLive peer with the list of online peers that are watching the same channel as the one the joining peer is interested in.

Figure 3 illustrates the high level diagram of PPLive system.

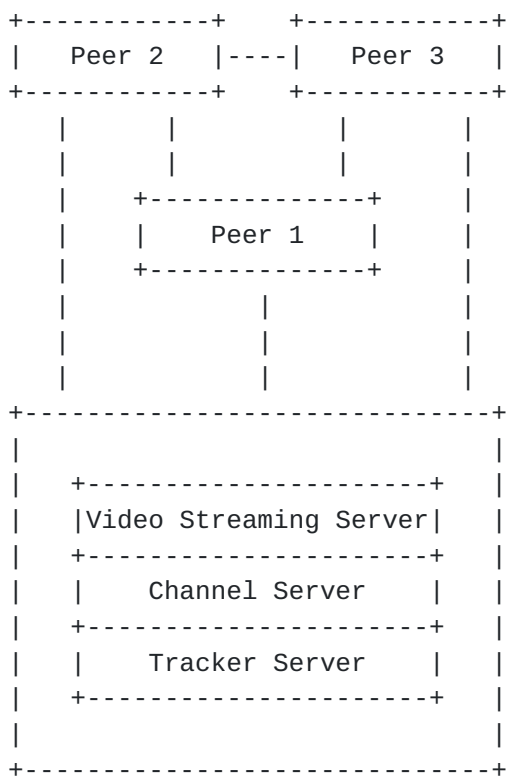

Figure 3, High level overview of PPLive system architecture

As regards the tracker protocol, as soon as a PPLive peer joins the systems and selects the channel to watch, it retrieves from the tracker server a list of peers that are watching the same channel.

As regards the peer protocol, it controls both peer discovery and chunk distribution process. More specifically, peer discovery is implemented by a kind of gossip-like mechanism. After retrieving the list of active peers watching a specific channel from tracker server, a PPLive peer sends out probes to establish active peer connections, and some of those peers may return also their own list of active peers to help the new peer discover more peers in the initial phase. Chunk distribution process is mainly based on buffer map exchange to advertise the availability of cached chunks. In more detail, PPLive software client exploits two local buffers to cache chunks: the PPLive TV engine buffer and media player buffer. The main reason behind the double buffer structure is to address the download rate variations when downloading chunks from PPLive network. In fact, received chunks are first buffered and reassembled into the PPLive TV engine buffer; as soon as the number of consecutive chunks in PPLive TV engine buffer overcomes a predefined threshold, the media player buffer downloads chunks from the PPLive TV engine buffer; finally, when the media player buffer fills up to the required level, the actual video playback starts.

Since the protocols and algorithm of PPLive are proprietary, most of known details have been derived from measurement studies. Specifically, it seems that:

-number of peers from which a PPLive node downloads live TV chunks from is constant and relatively low, and the top-ten peers contribute to a major part of the download traffic, as shown in [P2PIPTVMEA];

-PPLive can provide satisfactory performance for popular live TV and VoD channels. For unpopular live TV channels, performance may severely degrade, whereas for unpopular VoD channels this problem rarely happens, as it shown in [CNSR]. Authors of [CNSR] also demonstrate that the workload in most VoD channels is well balanced, whereas for live TV channels the workload distribution is unbalanced, and a small number of peers provide most video data.

4.3. Zattoo

Zattoo [Zattoo] is P2P live streaming system that was launched in Switzerland in 2006 in coincidence with the EUFA European Football Championship and in few years was able to attract almost 10 million registered users in several European countries.

Figure 4 depicts the high level architecture of Zattoo system. The main reference for the information provided in this document is [IMC09].

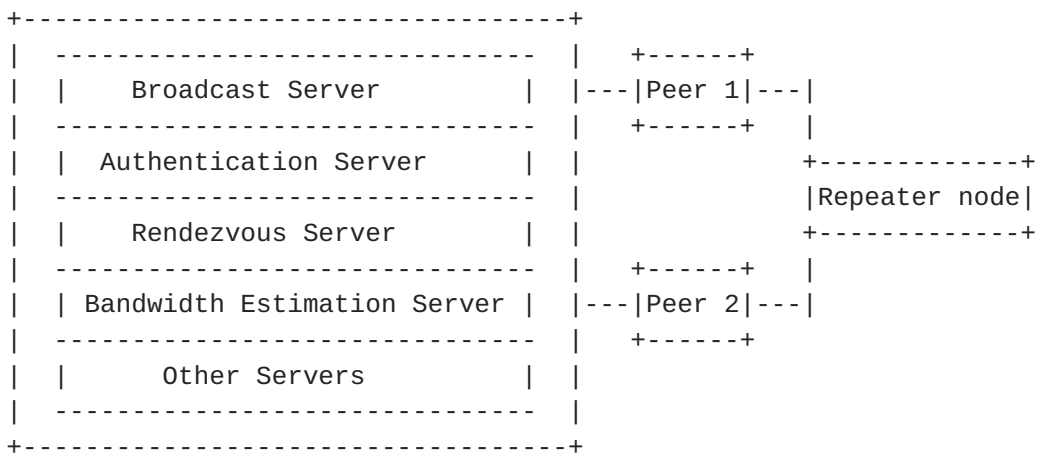

Figure 4, High level overview of Zattoo system architecture

Broadcast server is in charge of capturing, encoding, encrypting and sending the TV channel to the Zattoo network. A number N of logical sub-streams is derived from the original stream, and packets of the same order in the sub-streams are grouped together into the so-called segments. Each segment is then coded via a Reed-Salomon error correcting code in such a way that any number k < N of received packets in the segment is enough to reconstruct the whole segment.

Authentication server is the first point of contact for a peer that joins the system, and it authenticates Zattoo users. Then, a user contacts the Rendezvous server and specifies the TV channel of interest. The rendezvous server returns a list of Zattoo peers that have already joined the requested channel. Hence, rendezvous server plays the role of tracker. At this point the direct interaction between peers starts using the peer protocol.

A new Zattoo user contacts the peers returned by the rendezvous server in order to identify a set of neighboring peers covering the full set of sub-streams in the TV channel. This process is denoted in Zattoo jargon as Peer Division Multiplexing (PDM). To ease the identification of neighboring peers, each contacted peer provides also the list of its own known peers, in such a way that a new Zattoo user, if needed, can contact new peers besides the ones indicated by the rendezvous server. In selecting which peers to establish connections with, a peer adopts the criterion of topological closeness. The topological location of a peer is defined in Zattoo as (in order of preference) its subset number, its autonomous system number and its country code, and it is provided to each peer by the authentication server.

Zattoo peer protocol provides also a mechanism to make PDM process adaptive with respect to bandwidth fluctuations. First of all, a

peer controls the admission of new connections based on the available uplink bandwidth. This is estimated i) at beginning with each peer sending probe messages to the Bandwidth Estimation server, and ii) while forwarding sub-streams to other peers based on the quality-of-service feedback received by those peers. A quality-of-service feedback is sent from the receiver to the sender only when the quality of the received sub-stream is below a given threshold. So if a quality-of-service feedback is received, a Zattoo peer decrements the estimation of available uplink bandwidth, and if this drops below the amount needed to supports the current connections, a proper number of connections is closed. On the other side, if no quality-of-service feedback is received for a given time interval, a Zattoo peer increments the estimation of available uplink bandwidth according to a mechanism very similar to the one of TCP congestion window (a mechanism very similar to the one of TCP congestion window (double increase or linear increase depending on whether the estimate is below or above a given threshold).

Figure 4 also shows that there exist two classes of Zattoo nodes: simple peers, whose behavior has already been presented, and repeater nodes, that implement the same peer protocol as simple peers and in addition are high-bandwidth peers and are able to forward any sub-stream. In such a way repeater nodes serve as bandwidth multiplier.

4.4. PPStream

PPStream [PPStream] is a very popular P2P streaming software in China and in many other countries of East Asia.

The system architecture of PPStream is very similar to the one of PPLive. When a PPStream peer joins the system, it retrieves the list of channels from the channel list server. After selecting the channel to watch, a PPStream peer retrieves from the peer list server the identifiers of peers that are watching the selected channel, and it establishes connections that are used first of all to exchange buffer-maps. In more detail, a PPStream chunk is identified by the play time offset which is encoded by the streaming source and it is subdivided into sub-chunks. So buffer-maps in PPStream carry the play time offset information and are strings of bits that indicate the availability of sub-chunks. After receiving the buffer-maps from the connected peers, a PPStream peer selects peers to download sub-chunks according to a rate-based algorithm, which maximizes the utility of uplink and downlink bandwidth.

4.5. Tribler

Tribler [Tribler] is a BitTorrent [Bittorrent] client that was able to go very much beyond BitTorrent model also thanks to the support for video streaming. Initially developed by a team of researchers at Delft University of Technology, Tribler was able to both i) attract attention from other universities and media companies and ii) receive European Union research funding (P2P-Next and QLectives projects).

Differently from BitTorrent, where a tracker server centrally coordinates peers in uploads/downloads of chunks and peers directly interact with each other only when they actually upload/download chunks to/from each other, there is no tracker server in Tribler and, as a consequence, there is no need of tracker protocol.

This is illustrated also in Figure 5, which depicts the high level architecture of Tribler.

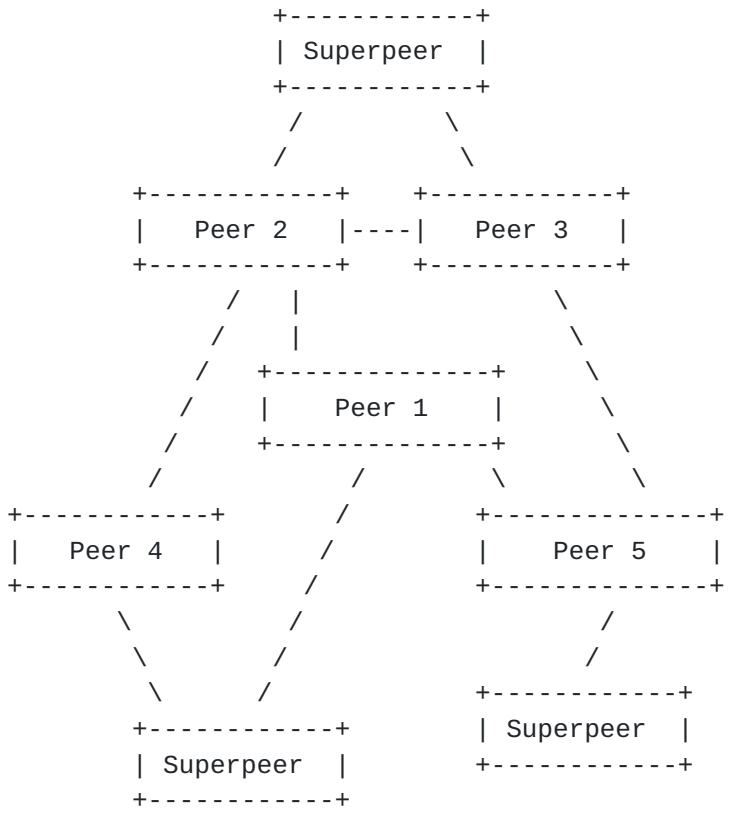

Figure 5, High level overview of Tribler system architecture

Regarding peer protocol and the organization of overlay mesh, Tribler bootstrap process consists in preloading well known superpeer addresses into peer local cache, in such a way that a joining peer randomly selects a superpeer to retrieve a random list of already

active peers to establish overlay connections with. A gossip-like mechanism called BuddyCast allows Tribler peers to exchange their preference list, that is their downloaded files, and to build the so called Preference Cache. This cache is used to calculate similarity levels among peers and to identify the so called "taste buddies" as the peers with highest similarity. Thanks to this mechanism each peer maintains two lists of peers: i) a list of its top-N taste buddies along with their current preference lists, and ii) a list of random peers. So a peer alternatively selects a peer from one of the lists and sends it its preference list, taste-buddy list and a selection of random peers. The goal behind the propagation of this kind of information is the support for the remote search function, a completely decentralized search service that consists in querying Preference Cache of taste buddies in order to find the torrent file associated with an interest file. If no torrent is found in this way, Tribler users may alternatively resort to a web-based torrent collector server available for BitTorrent clients.

Tribler supports video streaming in two different forms: video on demand and live streaming.

As regards video on demand, a peer first of all keeps informed its neighbors about the chunks it has. Then, on the one side it applies suitable chunk-picking policy in order to establish the order according to which to request the chunks he wants to download. This policy aims to assure that chunks come to the media player in order and in the same time that overall chunk availability is maximized. To this end, the chunk-picking policy differentiates among high, mid and low priority chunks depending on their closeness with the playback position. High priority chunks are requested first and in strict order. When there are no more high priority chunks to request, mid priority chunks are requested according to a rarest-first policy. Finally, when there are no more mid priority chunks to request, low priority chunks are requested according to a rarest-first policy as well. On the other side, Tribler peers follow the give-to-get policy in order to establish which peer neighbors are allowed to request chunks (according to BitTorrent jargon to be unchoked). In more detail, time is subdivided in periods and after each period Tribler peers first sort their neighbors according to the decreasing numbers of chunks they have forwarded to other peers, counting only the chunks they originally received from them. In case of tie, Tribler sorts their neighbors according to the decreasing total number of chunks they have forwarded to other peers. In this way, Tribler peer unchokes the three highest-ranked neighbours and, in order to saturate upload bandwidth and in the same time not decrease the performance of individual connections, it further unchokes a limited number of neighbors. Moreover, in order to search for better neighbors, Tribler peers randomly select a new peer in the

rest of the neighbours and optimistically unchoke it every two periods.

As regards live streaming, differently from video on demand scenario, the number of chunks cannot be known in advance. As a consequence a sliding window of fixed width is used to identify chunks of interest: every chunk that falls out the sliding window is considered outdated, is locally deleted and is considered as deleted by peer neighbors as well. In this way, when a peer joins the network, it learns about chunks its neighbors possess and identify the most recent one. This is assumed as beginning of the sliding window at the joining peer, which starts downloading and uploading chunks according to the description provided for video on demand scenario.

4.6. QQLive

QQLive [QQLive] is large-scale video broadcast software including streaming media encoding, distribution and broadcasting. Its client can apply for web, desktop program or other environments and provides abundant interactive function in order to meet the watching requirements of different kinds of users.

QQLive adopts Content Delivery Network (CDN) [CDN] and P2P architecture for video distribution and is different from other popular P2P streaming applications. QQLive provides video by source servers and CDN, and the video content can be push to every region by CDN throughout China. In each region, QQLive adopts P2P technology for video content distribution.

One of the main aims for QQLive is to use the simplest architecture to provide the best user experience. So QQLive takes some servers to implement P2P file distribution. There are two servers in QQLive: Stun Server [RFC5389] and Tracker Server. Stun Server is responsible for NAT traversing. Tracker Server is responsible for providing content address information. There are a group of these two Servers for providing services. There is no Super Peer in QQLive.

Working flow of QQLive includes startup stage and play stage.

-Startup stage includes only interactions between peers and Tracker servers. There is a built-in URL in QQLive client software. When the client startups and connects to the network, the client gets the Tracker's address through DNS and tells the Tracker the information of its owned video contents.

-Play stage includes interactions between peers and peers or peers and CDN. Generally, the client will download the video content from CDN during the first 30 seconds and then gets contents from

other peers. If unfortunately there is no peer which owns the content, the client will get the content from CDN again.

As the client watches the video, the client will store the video to the hard disk. The default storage space is one Gbyte. If the storage space is full, the client will delete the oldest content. When the client does VCR operation, if the video content is stored in hard disk, the client will not do interactions with other peers or CDN. If there are messages or video content missing, the client will take retransmission and the retransmission interval is decided by the network condition. The QQLive does not take care of the strategy of transmission and chunk selection, which is simple and not similar with BT because of the CDN support.

5. Tree-based P2P Streaming Systems

In tree-based P2P streaming applications peers self-organize in a tree-shape overlay network, where peers do not ask for a specific chunk, but simply receive it from their so called "parent" node. Such content delivery model is denoted as push-based. Receiving peers are denoted as children, whereas sending nodes are denoted as parents. Overhead to maintain overlay topology is usually lower for tree-based streaming applications than for mesh-based streaming applications, whereas performance in terms of delay is usually better. On the other side, the greatest drawback of this type of application lies in that each node depends on one single node, its parent in overlay tree, to receive streamed content. Thus, tree-based streaming applications suffer from peer churn phenomenon more than mesh-based ones.

5.1. End System Multicast (ESM)

Even though End System Multicast (ESM) project is ended by now and ESM infrastructure is not being currently implemented anywhere, we decided to include it in this survey for a twofold reason. First of all, it was probably the first and most significant research work proposing the possibility of implementing multicast functionality at end hosts in a P2P way. Secondly, ESM research group at Carnegie Mellon University developed the first P2P live streaming system of the world, and some members founded later Conviva [conviva] live platform.

The main property of ESM is that it constructs the multicast tree in a two-step process. The first step aims at the construction of a mesh among participating peers, whereas the second step aims at the construction of data delivery trees rooted at the stream source. Therefore a peer participates in two types of topology management structures: a control structure that guarantees peers are always

connected in a mesh, and a data delivery structure that guarantees data gets delivered in an overlay multicast tree.

There exist two versions of ESM.

The first version of ESM architecture [ESM1] was conceived for small scale multi-source conferencing applications. Regarding the mesh construction phase, when a new member wants to join the group, an out-of-bandwidth bootstrap mechanism provides the new member with a list of some group members. The new member randomly selects a few group members as peer neighbors. The number of selected neighbors never exceeds a given bound, which reflects the bandwidth of the peer's connection to the Internet. Each peer periodically emits a refresh message with monotonically increasing sequence number, which is propagated across the mesh in such a way that each peer can maintain a list of all the other peers in the system. When a peer leaves, either it notifies its neighbors and the information is propagated across the mesh to all the participating peers, or peer neighbors detect the condition of abrupt departure and propagate it through the mesh. To improve mesh/tree quality, on the one side peers constantly and randomly probe each other to add new links; on the other side, peers continually monitor existing links in order to drop the ones that are not perceived as good-quality links. This is done thanks to the evaluation of a utility function and a cost function, which are conceived to guarantee that the shortest overlay delay between any pair of peers is comparable to the unicast delay among them. Regarding multicast tree construction phase, peers run a distance-vector protocol on top of the tree and use latency as routing metric. In this way, data delivery trees may be constructed from the reverse shortest path between source and recipients.

The second and subsequent version of ESM architecture [ESM2] was conceived for an operational large scale single-source Internet broadcast system. As regards the mesh construction phase, a node joins the system by contacting the source and retrieving a random list of already connected nodes. Information on active participating peers is maintained thanks to a gossip protocol: each peer periodically advertises to a randomly selected neighbor a subset of nodes he knows and the last timestamps it has heard for each known node. The main difference with the first version is that the second version constructs and maintains the data delivery tree in a completely distributed manner according to the following criteria: i) each node maintains a degree bound on the maximum number of children it can accept depending on its uplink bandwidth, ii) tree is optimized mainly for bandwidth and secondarily for delay. To this end, a parent selection algorithm allows identifying among the neighbors the one that guarantees the best performance in terms of throughput and delay. The same algorithm is also applied either if a

parent leaves the system or if a node is experiencing poor performance (in terms of both bandwidth and packet loss). As loop prevention mechanism, each node keeps also the information about the hosts in the path between the source and its parent node.

This second ESM prototype is also able to cope with receiver heterogeneity and presence of NAT/firewalls. In more detail, audio stream is kept separated from video stream and multiple bit-rate video streams are encoded at source and broadcast in parallel though the overlay tree. Audio is always prioritized over video streams, and lower quality video is always prioritized over high quality video. In this way, system can dynamically select the most suitable video stream according to receiver bandwidth and network congestion level. Moreover, in order to take presence of hosts behind NAT/ firewalls, tree is structured in such a way that public hosts use hosts behind NAT/firewalls as parents.

6. Hybrid P2P streaming applications

This type of applications aims at integrating the main advantages of mesh-based and tree-based approaches. To this end, overlay topology is mixed mesh-tree, and content delivery model is push-pull.

6.1. New Coolstreaming

Coolstreaming, first released in summer 2004 with a mesh-based structure, arguably represented the first successful large-scale P2P live streaming. Nevertheless, it suffers poor delay performance and high overhead associated with each video block transmission. In the attempt of overcoming such a limitation, New Coolstreaming [NEWCOOLStreaming] adopts a hybrid mesh-tree overlay structure and a hybrid pull-push content delivery mechanism.

Like in the old Coolstreaming, a newly joined node contacts a special bootstrap node and retrieves a partial list of active nodes in the system.

The interaction with bootstrap node is the only one related to the tracker protocol. The rest of New Coolstreaming interactions are related to peer protocol.

The newly joined node then establishes a partnership with few active nodes by periodically exchanging information on content availability. Streaming content is divided in New Coolstreaming in equal-size blocks or chunks, which are unambiguously associated with sequence numbers that represent the playback order. Chunks are then grouped to form multiple sub-streams.

Like in most of P2P streaming applications information on content availability is exchanged in form of buffer-maps. However, New Coolstreaming buffer-maps differ from the usual format of strings of bits where each bit represents the availability of a chunk. Two vectors represent indeed buffer-maps in New Coolstreaming. The first vector reports the sequence numbers of the last chunk received for a given sub-stream. The second vector is used to explicitly request chunks from partner peers. In more details, the second vector has as many bits as sub-streams, and a peer receiving a bit "1" in correspondence of a given sub-stream is being requested from the sending peer to upload chunks belonging to that sub-streams. Since chunks are explicitly requested, data delivery may be regarded as pull-based. However, data delivery is push-based as well, since every time a node is requested to upload chunks, it uploads all chunks for that sub-stream starting from the one indicated in the first vector of received buffer-map. Hence, the overall overlay topology is mesh-based, but it is also possible to identify as many overlay trees as sub-streams.

In order to improve quality of mesh-tree overlay, each node continuously monitors the quality of active connections in terms of mutual delay between sub-streams. If such quality drops below a predefined threshold, a New Coolstreaming node selects a new partner among its partners. Parent re-selection is also triggered for a peer when its previous parent leaves.

7. Security Considerations

Security in P2P streaming applications may be addressed at two different levels: on the one side, at the control protocol level, on the other side, at streamed multimedia content level.

In PPLive and PPStream control protocol messages are sent over HTTP, UDP and TCP mostly in plain text, and this can allow malicious users to interfere with the normal operation of the system and can lead to malicious attacks that can make key components of the system ineffective.

In Zattoo authentication server authenticates Zattoo users and assigns them with a limited lifetime ticket. Then, a user presents the tickets received by the authentication server to the rendezvous server. Provided that the presented ticket is valid, the rendezvous server returns a list of Zattoo peers that have already joined the requested channel and a signed channel ticket.

In Tribler authentication of peers is based on secure, permanent peer identifiers called PermIDs. PermID maps to a single IP address and port number and is initially used to identify users. The idea is to

have each Tribler user assigned with a public/private keypair based on Elliptic Curve Cryptography (ECC), where public key acts as the PermID for the user. Users distribute their PermID to their friends out-of-band to establish trusted friend relationships. When two peers connect as part of a download, they authenticate each other using the standard ISO/IEC 9798-3 [ISO/IEC 9798-3] challenge/response identification protocol. If the peer is successfully authenticated but not a friend of the user (i.e., does not appear in the list of friends' PermIDs), the Tribler client will allow it to request non-privileged operations, such as exchanging file preferences. If the peer is a friend, it may request privileged operations such as coordinating a friends-assisted download. Moreover, Tribler provides security at streamed content level too. In the video on demand scenario torrent files include a hash for each chunk in order to prevent malicious attackers from corrupting data. In live streaming scenario torrent files include the public key of the stream source. Each chunk is then assigned with absolute sequence number and timestamp and signed by source public key. Such a mechanism allows Tribler peers to use the public key included in torrent file and verify the integrity of each chunk.

In QQLive both tracker and peer protocol are fully private and encrypt the whole message. The tracker protocol uses UDP and the port for the tracker server is fixed. For the streamed content, if the client gets the streaming from CDN, the client use the HTTP with port 80 and no encryption. If the client gets the streaming from other peers, the client use UDP to transfer the encrypted media streaming and not RTP/RTCP.

8. IANA Considerations

This document has no actions for IANA.

9. Author List

Other authors of this document are listed as below.

Hui Zhang, NEC Labs America.

Jun Lei, University of Goettingen.

Gonzalo Camarillo, Ericsson.

Yong Liu, Polytechnic University.

Delfin Montuno, Huawei.

Lei Xie, Huawei.

10. Acknowledgments

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

11. Informative References

[RFC6972] RFC 6972, "Problem Statement and Requirements of the Peer-to-Peer Streaming Protocol (PPSP)".

[Octoshape] Alstrup, Stephen, et al., "Introducing Octoshape-a new technology for large-scale streaming over the Internet".

[CNN] CNN web site, http://www.cnn.com

[PPLive] PPLive web site, http://www.pplive.com

[P2PIPTVMEA] Silverston, Thomas, et al., "Measuring P2P IPTV Systems", June 2007.

[CNSR] Li, Ruixuan, et al., "Measurement Study on PPLive Based on Channel Popularity", May 2011.

[Zattoo] Zattoo web site, http://www.zattoo.com

[IMC09] Chang, Hyunseok, et al., "Live streaming performance of the Zattoo network", November 2009.

[PPStream] PPStream web site, http:// www.ppstream.com

[Tribler] Tribler Protocol Specification, January 2009, on line available at http://svn.tribler.org/bt2-design/proto-spec-unified/trunk/proto-spec-current.pdf

[Bittorrent] BitTorrent web site, http:// www.bittorrent.com

[QQLive] QQLive web site, http://v.qq.com

[CDN] CDN wiki, http://en.wikipedia.org/wiki/Content_delivery_network

[RFC5389] RFC5389, "Session Traversal Utilities for NAT (STUN)".

[conviva] Conviva web site, http://www.conviva.com

[ESM1] Chu, Yang-hua, et al., "A Case for End System Multicast", June 2000. (http://esm.cs.cmu.edu/technology/papers/Sigmetrics.CaseForESM.2000.pdf)

[ESM2] Chu, Yang-hua, et al., "Early Experience with an Internet Broadcast System Based on Overlay Multicast", June 2004. (http://static.usenix.org/events/usenix04/tech/general/full_papers/chu/chu.pdf)

[NEWCOOLStreaming] Li, Bo, et al., "Inside the New Coolstreaming: Principles,Measurements and Performance Implications", April 2008.

[ISO/IEC 9798-3] ISO web site, http://www.iso.org/iso/catalogue_detail.htm?csnumber=29062

Authors' Addresses

Yingjie Gu
Unaffiliated

Email: guyingjie@gmail.com

Ning Zong (editor)
Huawei
101 Software Avenue
Nanjing 210012
China

Phone: +86-25-56624760
Fax: +86-25-56624702
Email: zongning@huawei.com

Yunfei Zhang
Coolpad
China Mobile
Email: hishigh@gmail.com

Francesca Lo Piccolo
Cisco
Via del Serafico 200
Rome 00142
Italy

Phone: +39-06-51645136
Email: flopicco@cisco.com

Shihui Duan
CATR
No.52 HuaYuan BeiLu
Beijing 100191
P.R.China

Phone: +86-10-62300068
Email: duanshihui@catr.cn