Individual Submission
Internet-Draft
Intended status: Informational
Expires: September 15, 2011

C. Dannewitz
University of Paderborn
T. Rautio
VTT Technical Research Centre of Finland
O. Strandberg
Nokia Siemens Networks
B. Ohlman
Ericsson

March 14, 2011

Secure naming structure and p2p application interaction
draft-dannewitz-ppsp-secure-naming-02

Abstract

Today, each application typically uses its own way to identify data. The lack of a common naming scheme prevents applications from benefiting from available copies of the same data distributed via different P2P and CDN systems. The main proposal presented in this draft is idea that there should be a secure and application independent way of naming information objects that are transported over the Internet. The draft defines a set of requirements for such a naming structure. It also presents a proposal for such a naming structure that could relevant for a number of work groups (existing and potential), e.g. PPSP, DECADE and CDNI. In addition, today's P2P naming schemes lack important security aspects that would allow the user to check the data integrity and build trust in data and data publishers. This is especially important in P2P applications as data is received from untrusted peers. Providing a generic naming scheme for P2P systems so that multiple P2P systems can use the same data regardless of data location and P2P system increases the efficiency and data availability of the overall data dissemination process. The secure naming scheme is providing self-certification such that the receiver can verify the data integrity, i.e., that the correct data has been received, without requiring a trusted third party. It also enables owner authentication to build up trust in (potentially anonymous) data publishers. The secure naming structure should be beneficial as potential design principle in defining the two protocols identified as objectives in the PPSP charter. This document enumerates a number of design considerations to impact the design and implementation of the tracker-peer signaling and peer-peer streaming signaling protocols.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",

"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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 September 15, 2011.

Copyright Notice

Table of Contents

1. Introduction

Today's dominating naming schemes in the Internet, i.e., IP addresses and URLs, are rather host-centric with respect to the fact that they are bound to a location. This kind of naming scheme is not optimal for many of the predominant users of todays Internet like P2P and CDN systems as they are based on an information-centric thinking, i.e., putting the information itself in focus. In these system the source of the information is secondary and can constantly change, e.g. new caches or P2P peers becomes available. It also common to retrieve information from more than one source at once.

For any type of caching solution (network based or P2P) and network based storage, e.g. DECADE, a common application independent naming scheme is essential to be able to identify cached copies of information/data objects.

Many applications, in particular P2P applications, use their own data model and protocol for keeping track of data and locations. This poses a challenge for use of the same information for several applications. A common naming scheme for information objects is important to enable interconnectivity between different application systems, such as P2P and CDN. To be able to build a common P2P infrastructure that can serve a multitude of applications there is a need for a common application independent naming scheme. With such a naming scheme different applications can use and refer to the same information/data objects.

It is possible to introduce false data into P2P systems, only detectable when the content is played out in the user application. The false data copies can be identified and sorted out if the P2P system can verify the reference used in the tracker protocol towards data received at the peer. One option to address this can be to secure the naming structure i.e. make the data reference be dependent on the data and related metadata.

An additional reason to introduce a common naming scheme for information objects is caching. When data are named in a host-centric way, as is done today, it is not always identify that copies of the same information object are available in multiple hosts. With location independent identifiers for information objects this becomes much easier.

This document enumerates and explains the rationale for why a common naming structure for information/data objects should be defined and used by a wide range of applications and network protocols. Examples of WGs (and potential WGs) where we think a new standard for naming of information/data objects should be valuable includes PPS, DECADE

and CDNI. For P2P systems the main advantage is probably in the definition of a protocol for signaling and control between trackers and peers (the PPSP "tracker protocol") but also a signaling and control protocol for communication among the peers (the PPSP "peer protocol") might have benefits from a common and secure naming scheme. In DECADE one key feature would be that different applications can easily share the same cache entries. It should also be valuable for cooperative caching, e.g. CDNI.

2. Naming requirements

In the following, we discuss the requirements that a common naming scheme has to fulfill.

To enable efficient, large scale data dissemination that can make use of any available data copy, identifiers (IDs) have to be location-independent. Thereby, identical data can be identified by the same ID independently of its storage location and improved data dissemination can then benefit from all available copies. This should be possible without compromising trust in data regardless of its network source.

Security in an information-centric network (including P2P, caching, and network-based solutions) needs to be implemented differently than in host-centric networks. In the latter, most security mechanisms are based on host authentication and then trusting the data that the host delivers. In e.g. a P2P system, host authentication cannot be relied upon, or one of the main advantages of a P2P system, i.e., benefiting from any available copy, is defeated. Host authentication of a random, untrusted host that happens to have a copy does not establish the needed trust. Instead, the security has to be directly attached to the data which can be done via the scheme used to name the data.

Therefore, self-certification is a main requirement for the naming scheme. Self-certification ensures the integrity of data and securely binds this data to its ID. More precisely, this property means that any unauthorized change of data with a given ID is detectable without requiring a third party for verification. Beforehand, secure retrieval of IDs (e.g., via search, embedded in a Web page as link, etc.) is required to ensure that the user has the right ID in the first place. Secure ID retrieval can be achieved by using recommendations, past experience, and specialized ID authentication services and mechanisms that are out of the scope of this discussion.

Another important requirement is name persistence, not only with

respect to storage location changes as discussed above, but also with respect to changes of owner and/or owner's organizational structure, and content changes producing a new version of the information. Information should always be identifiable with the same ID as long as it remains essentially equivalent. Spreading of persistent naming schemes like the Digital Object Identifier (DOI) [Paskin2010] also emphasizes the need for a persistent naming scheme. However, name persistence and self-certification are partly contradictory and achieving both simultaneously for dynamic content is not trivial.

From a user's perspective, persistent IDs ensure that links and bookmarks remain valid as long as the respective information exists somewhere in the network, reducing today's problem of "404 - file not found" errors triggered by renamed or moved content. From a content provider's perspective, name persistence simplifies data management as content can, e.g., be moved between folders and different servers as desired. Name persistence with respect to content changes makes it possible to identify different versions of the same information by the same consistent ID. If it is important to differentiate between multiple versions, a dedicated versioning mechanism is required, and version numbers may be included as a special part of the ID.

The requirement of building trust in an information-centric system combined with the desire for anonymous publication as well as accountability (at least for some content) can be translated into two related naming requirements. The first is owner authentication, where the owner is recognized as the same entity, which repeatedly acts as the object owner, but may remain anonymous. The second is owner identification, where the owner is also identified by a physically verifiable identifier, such as a personal name. This separation is important to allow for anonymous publication of content, e.g., to support free speech, while at the same time building up trust in a (potentially anonymous) owner.

In general, the naming scheme should be able to adapt to future needs. Therefore, the naming scheme should be extensible, i.e., it should be able to add new information (e.g., a chunk number for BitTorrent-like protocols) to the naming scheme. The need for such extensions is stressed by today's variety of naming schemes (e.g., DOI or PermaLink) added on top of the original Internet architecture that fulfill specialized needs which cannot be met by the common Internet naming schemes, i.e., IP addresses and URLs.

3. Basic Concepts for an Application-independent Naming Scheme

In this section, we introduce an exemplary naming scheme that illustrates a possible way to fulfill the requirements posed upon an

application-independent naming scheme for information-centric networks. The naming scheme integrates security deeply into the system architecture. Trust is based on the data's ID in combination with additional security metadata. Section 3.1 gives an overview of the naming scheme in general with details about the ID structure, and Section 3.2 describes the security metadata in more detail.

3.1. Overview

Building on an identifier/locator split, each data element, e.g., file, is given a unique ID with cryptographic properties. Together with the additional security metadata, the ID can be used to verify data integrity, owner authentication, and owner identification. The security metadata contains information needed for the security functions of the naming scheme, e.g., public keys, content hashes, certificates, and a data signature authenticating the content. In comparison with the security model in today's host-centric networks, this approach minimizes the need for trust in the infrastructure, especially in the host(s) providing the data.

In an information-centric network, multiple copies of the same data element typically exist at different locations. Thanks to the ID/ locator split and the application-independent naming scheme, those identical copies have the same ID and, hence, each application can benefit from all available copies.

Data elements are manipulated (e.g., generated, modified, registered, and retrieved) by physical entities such as nodes (clients or hosts), persons, and companies. Physical entities able of generating, i.e., creating or modifying data elements are called owners here. Several security properties of this naming scheme are based on the fact that each ID contains the hash of a public key that is part of a public/ secret key pair PK/SK. This PK/SK pair is conceptually bound to the data element itself and not directly to the owner as in other systems like DONA [Koponen]. If desired, the PK/SK pair can be bound to the owner only indirectly, via a certificate chain. This is important to note because it enables owner change while keeping persistent IDs. The key pair bound to the data is thus denoted as PK_D/SK_D.

Making the (hash of the) public key part of ID enables self-certification of dynamic content while keeping persistent IDs. Self-certification of static content can be achieved by simply including the hash of content in the ID, but this would obviously result in non-persistent IDs for dynamic content. For dynamic content, the public key in the ID can be used to securely bind the hash of content to the ID, by signing it with the corresponding secret key, while not making it part of ID.

The owner's PK as part of the ID inherently provides owner authentication. If the public key is bound to the owner's identity (i.e., to its real-world name) via a trusted third party certificate, this also allows owner identification. Without this additional certificate, the owner can remain anonymous.

To support the potentially diverse requirements of certain groups of applications and adapt to future changes, the naming scheme can enable flexibility and extensibility by supporting different name structures, differentiated via a Type field in the ID.

3.2. ID Structure

The naming scheme uses flat IDs to support self-certification and name persistence. In addition, flat IDs are advantageous when it comes to mobility and they can be allocated without an administrative authority by relying on statistical uniqueness in a large namespace, with the rare case of ID collisions being handled by the applications. Although IDs are not hierarchical, they have a specified basic ID structure. The ID structure given as ID = (Type field | A = hash(PK) | L) is described subsequently.

The Authenticator field A=Hash(PK_D) binds the ID to a public key PK_D. The hash function Hash is a cryptographic hash function, which is required to be one-way and collision-resistant. The hash function serves only to reduce the bit length of PK_D. PK_D is generated in accordance with a chosen public-key cryptosystem. The corresponding secret key SK_D should only be known to a legitimate owner. In consequence, an owner of the data is defined as any entity who (legitimately) knows SK_D.

The pair (A, L) has to be globally unique. Hence, the Label field L provides global uniqueness if PK_D is repeatedly used for different data.

To build a flexible and extensible naming scheme, e.g., to adapt the naming scheme to future changes, different types of IDs are supported by the naming scheme and differentiated via a mandatory and globally standardized Type field in each ID. For example, the Type field specifies the hash functions used to generate the ID. If a used hash function becomes insecure, the Type field can be exploited by the P2P system in order to automatically mark the IDs using this hash function as invalid.

3.3. Security Metadata Structure

The security metadata is extensible and contains all information required to perform the security functions embedded in the naming

scheme. The metadata (or selected parts of it) will be signed by SK_D corresponding to PK_D. This securely binds the metadata to the ID, i.e., to the Hash(PK_D) which is part of the ID. For example, the security metadata may include:

- specification of the hash function h and the algorithm DSAlg used for the digital signature
- complete PK_D (not only Hash(PK_D))
- specification of the parts of data that are self-certified, i.e., authenticated via the signature
- hash of the self-certified data
- signature of the self-certified data signed by SK_D
- all data required for owner authentication and identification

A detailed description and security analysis of this naming scheme and its security properties, especially self-certification, name persistence, owner authentication, and owner identification can be found in the GIS paper Secure Naming for a Network of Information [Dannewitz_10].

4. Examples of application use of secure naming structure

This section contains a number of examples how a secure naming system, as outlined in this draft, could be used by different types of applications.

4.1. Secure naming for P2P applications

From an P2P application perspective the main advantage of a secure naming structure for a P2P infrastructure is that multiple P2P applications can have common access to the same data elements. Another benefit of application-independent naming is that locally available and cached copies can easily be located. The secure naming also enables that data can be verified even if it is received from an untrusted host.

For example, when an application like BitTorrent [WWWbittorrent] uses self-certifying names, the user is guaranteed that the data received is actually the data that has been requested, without having to trust any servers in the network (e.g., the tracker) or the peers that provide the data.

This means that BitTorrent's validation of the data integrity can be improved significantly using the presented secure naming structure. Currently, a standard BitTorrent system has no means to verify the integrity of the torrent file and consequently of the data. The torrent file (see Figure 1) contains the SHA1 hashes of the content pieces (pieces in Figure 2). However, anyone can modify a torrent file to bind different content to this file. If the torrent file gets modified, the user has no means any more to verify the integrity of the data. Modification of the torrent affects only to info_hash value, which is SHA1 hash calculated from the torrent's info field (see figure). The info_hash is respectively used for torrent session identification in different software entities (e.g. in trackers). After changes in the torrent's info field, the torrent is referring to different torrent session that is carrying a forged content. Additionally if, the tracker allows insertion of several torrents with the same name - delivers forged data (consistent with the forged torrent file), a user could effectively be tricked into downloading forged content which would falsely be identified as being correct by the BitTorrent client. On the other hand, the torrent referring to a forged content can be also modified to point to, another, "convenient" tracker by modifying the announce field in the torrent, and the outcome would be the same from user perspective. I.e., in the current BitTorrent system, a user has no guarantee that the downloaded content actually matches the expected/correct content.

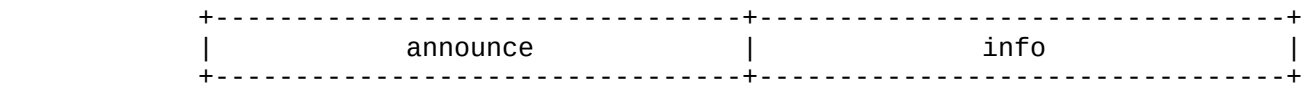


Figure 1: Basic structure of the BitTorrent torrent file

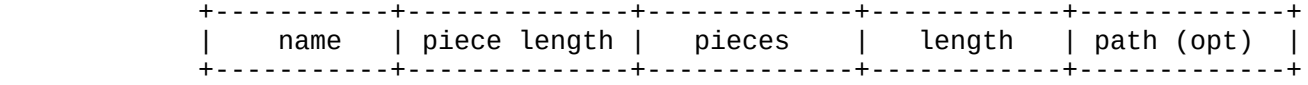


Figure 2: Structure of info field in torrent file

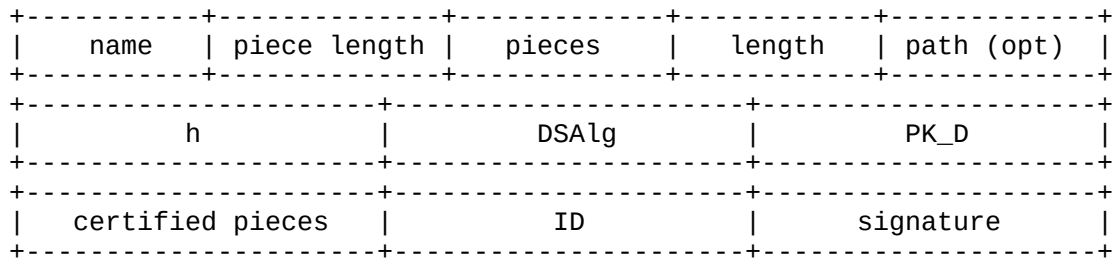


Figure 3: Structure of Secure naming enabled info field in torrent

The secure naming structure presented in this draft can provide a simple solution for this problem by securely binding the content of the torrent file to the name/ID of the torrent file. This can be done by extending the torrent file to include the above described security metadata information, as it is seen in Figure 3. In practice, during the torrent file creation, an object owner would store information about utilized algorithms (h - hash function and DSAlg - digital signature algorithm), the public key (PK_D), specification of signed data and ID into the torrent's info field, and will sign the combination of the secure metadata and the piece hash values (pieces in the torrent's info field) with the private key (SK_D). The generated signature will also be included in the extension part of the info field (signature).

After the content of the extended torrent is created, the respective torrent file ID would be generated according to the rules described in Section 3. As defined in that section, ID contains three different fields, namely Type, A and L. In the case of BitTorrent, Type field would carry on information about used hash function to generate field A from PK_D, and also structure of the field L. If, for example, L has name and version of the distributed file, Type field should tell that by including strings "Name" and "Version" in it. The next one, field A, includes hash values of the used PK_D (method defined in Type). And finally the proposed BitTorrents ID field L, can take in name and version of the distributed file. According to the description and by using separators - (within one field) and _ (between fields) the torrent file name could look, for example, like: HashMethod-Name-Version_HashofPK_Filename-Fileversion.torrent.

Consequently, whenever a user knows the ID of the content/torrent file and retrieves the torrent file, she/he can now open the torrent with the secure naming supported BitTorrent client. The client verifies the integrity of the torrent file by comparing PK_D in secure metadata and field A in the ID, in addition, conformance of ID in the torrent name and ID in the metadata is verified. With respect

to the secure metadata the signature and actual data is compared also. Once these three are verified, the client can download the data pieces, and can use the BitTorrent's included (and now secured) hash(es) to verify the integrity of the received data. As a result, the user can be sure that the correct content was retrieved.

4.2. Secure naming use in DECADE

DECADE WG is specifying requirements for a protocol concerning accessing data in a network storage and resource control of said data. A key aspect in accessing data in a network storage is the way the data is referenced. This naming draft has outlined a naming structure that can be utilized to enhance the reference to include additional features. The secure naming structure tries to fulfill several design targets and requirements, however not all are necessarily the priority requirements for the DECADE scope.

The DECADE storage is used by individual and uncoordinated entities, thus the naming of the data must be collision free. Also when a user accesses data the name should point to the correct data. With no entity to keep track of used names for data, one potential approach is to use large enough identifier designed with statistically collision free random properties. One obvious identifier alternative is to base it on the hash of the content.

The basic requirement for naming in DECADE is that the data identifier is tied to the hash of the content and that it is taken from a large enough flat namespace. In this way, wherever the same data is stored, the same name identifier can be used. Someone accessing the data can verify that the content is correct based on relationship between data and identifier. Other requirements can be included to further enhance the meaning and capability of the data reference identifier. Additional naming requirements could be:

- self-certified name for verifying content owner (owner of Pk/Sk keys), the self-certification can be used for building trust about data publisher
- solution for persistent identifier names for dynamic (changing) data
- potentially way to identify content owner, this typically requires trusted third party certifier.

This draft specifies several requirements that would be useful for the DECADE protocol, the main requirement is hashing of data into the identifier name. Depending of data use (like enhanced security properties) other secondary requirements will be beneficial for

additional functionality.

4.3. Secure naming for CDNs

The use of common naming within a CDN is not the challenge, it is the common naming between end users and the CDN or between CDNs that can be feasible. A common naming enables use of numerous caching points and CDNs as the same data can be referenced in the same way. The same data can depending of popularity be available in multiple location like caches and CDNs. The population of the resources (caches and CDNs) can be efficient if a common naming of data is used. The interaction between CDNs to 'negotiate' data population of caching resources would benefit from a common data reference model. The security features of the naming scheme also helps the CDN provider trust the data it is accessing and providing. The CDN interaction is potentially in the scope of the CDNI WG BOF.

5. Conclusion

The main proposal presented in this draft is idea that there should be a secure and application independent way of naming information objects that are transported over the Internet. The draft defines a set of requirements for such a naming structure. It also presents a proposal for such a naming structure that could relevant for a number of work groups (existing and potential), e.g. PPSP, DECADE and CDNI.

Specifically for the PPSP WG the secure naming structure is proposed for consideration as common reference ID structure. For any P2P streaming application to have fair and multitude of data access, it is essential to have a common naming structure that is suitable for many different needs. The common naming is probably best displayed in the tracker protocol case but potential benefit in the actual streaming protocol case has to still be identified. The secure binding of reference ID to the actual content is manifested in the end user peer possibility to check correct data reception in regard to the used ID.

The naming structure has been implemented in the 4WARD project prototypes and has been released as open source (www.netinf.org). The naming structure is also available through a public NetInf registration service at www.netinf.org. Three NetInf-enabled applications have also been published, the InFox (Firefox plugin), InBird (Thunderbird plugin), and a NetInf Information Object Management Tool, all available at the www.netinf.org site.

Continued work on defining a common naming structure for information objects is carried out in the SAIL project. More information is

available at the www.sail-project.eu site.

6. IANA Considerations

This document has no requests to IANA.

7. Security Considerations

There are considerations about what private/public key and hash algorithms to utilize when designing the naming structure in a secure way.

8. Acknowledgements

We would like to thank all the persons participating in the Network of Information work packages in the EU FP7 projects 4WARD and SAIL and the Finnish ICT SHOK Future Internet 2 project for contributions and feedback to this document.

9. Informative References

[Dannewitz_10] Dannewitz, C., Golic, J., Ohlman, B., and B. Ahlgren, "Secure Naming for a Network of Information", 13th IEEE Global Internet Symposium , 2010.

[Koponen] Koponen, T., Chawla, M., Chun, B., Ermolinskiy, A., Kim, K., Shenker, S., and I. Stoica, "A Data-Oriented (and beyond) Network Architecture", Proc. ACM SIGCOMM , 2007.

[Paskin2010] Paskin, N., "Digital Object Identifier ({DOI}?) System", Encyclopedia of Library and Information Sciences , 2010.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[WWWbittorrent] Cohen, B., "The BitTorrent Protocol Specification", http://www.bittorrent.org/beps/bep_0003.html , 2008.

Authors' Addresses

Christian Dannewitz
University of Paderborn
Paderborn
Germany

Email: cdannewitz@upb.de

Teemu Rautio
VTT Technical Research Centre of Finland
Oulu
Finland

Email: teemu.rautio@vtt.fi

Ove Strandberg
Nokia Siemens Networks
Espoo
Finland

Email: ove.strandberg@nsn.com

Borje Ohlman
Ericsson
Stockholm
Sweden

Email: Borje.Ohlman@ericsson.com

PPSP
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2012

Y. Gu
J. Xia
Huawei
R. Cruz
M. Nunes
IST/INESC-ID/INOV
David A. Bryan
Polycom
J. Taveira
ID/INOV

Oct 31, 2011

Peer Protocol
draft-gu-ppsp-peer-protocol-03

Abstract

This document presents the architecture of the PPSP Peer protocol outlining the functional entities, message flows and message processing instructions, with the respective parameters. The PPSP Peer Protocol proposed in this document extends the capabilities of PPSP to support adaptive and scalable video and 3D video, for Video On Demand (VoD) and Live video services. The protocol messages formal syntax and semantics, methods, and formats are presented for both Binary and HTTP/XML encoded formats.

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 May 3, 2012.

Table of Contents

1. Introduction

The P2P Streaming Protocol (PPSP) is composed of two protocols: the PPSP Tracker Protocol and the PPSP Peer Protocol [I-D.ietf-ppsp-problem-statement].

The PPSP architecture requires PPSP peers able to communicate with a Tracker in order to participate in a particular swarm. This centralized Tracker service is used for peer and content registration and location. Content indexes (Media Presentation Descriptions) are also stored in the Tracker system allowing the association of content location information to the active peers in the swarm sharing the content.

The PPSP Tracker Protocol provides communication between Trackers and Peers and outlines how a peer is able to communicate with a tracker in order to exchange meta information about the location of other peers contributing with a specific stream (swarm) the peer interested in, as well as to report streaming status. The Peer can also apply to be a contributor for several streams (swarms), periodically reporting its status to the Tracker, allow it to estimate whether the peer is a competent contributor.

The PPSP Peer protocol outlines how a peer is able to communicate with other peers in order to control the advertising and exchange of media data, directly between peers, for a specific stream (swarm), as described in [I-D.ietf-ppsp-problem-statement].

The process used for media streaming distribution assumes a segment transfer scheme whereby the original content (that can be encoded using adaptive or scalable techniques) is chopped into small segments (and subsegments). For simplicity, in this document the segments (and subsegments) of media are named Chunks. The media streaming process has the following representations:

1. Adaptive - alternate representations with different qualities and bitrates; a single represention is non-adaptive;

2. Scalable description levels - multiple additive descriptions (i.e., addition of descriptions refine the quality of the video);

3. Scalable layered levels - nested dependent layers corresponding to several hierarchical levels of quality, i.e., higher enhancement layers refine the quality of the video of lower layers.

4. Scalable multiple views - views correspond to mono and stereoscopic 3D videos, with several hierarchical levels of

quality.

These streaming distribution techniques support dynamic variations in video streaming quality while ensuring support for a plethora of end user devices and network connections.

The information that should be exchanged between peers using this Peer Protocol includes:

1. ChunkMap indicating which chunks a peer possesses.

2. Required ChunkIDs

3. Peer preferences and status information

4. Signalling and Data Transport protocol negotiation

5. Information that can help improve the performance of PPSP.

In this document, a set of concrete information that needs to be exchanged between peers is introduced, together with the messages to convey such information.

This documents describes the PPSP Peer protocol and how it satisfies the requirements for the IETF Peer-to-Peer Streaming Protocol (PPSP), in order to derive the implications for the standardization of the PPSP streaming protocols and to identify open issues and promote further discussion.

This PPSP Peer Protocol proposal presents an early sketch for an extensible protocol that extends the capabilities of PPSP to support adaptive and scalable video.

2. Document Conventions

2.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.2. Terminology

The draft uses the terms defined in [I-D.ietf-ppsp-problem-statement], [I-D.gu-ppsp-tracker-protocol] and [I-D.cruz-ppsp-http-peer-protocol]. Additionally, This document uses the following acronyms and definitions frequently in itself:

Peer-Peer Messages

The Peer Protocol messages enable each Peer to exchange content availability with other Peers and request other Peers for content.

Tracker-Peer Messages

The Tracker Protocol messages provide communication between Peers and Trackers, by which Peers provide content availability, report streaming status and request candidate Peer lists from Trackers.

Connection Tracker

The Tracker Node to which the PPSP Peer will connect when it wants to join the PPSP system.

Sender Peer

A peer that contains the corresponding chunk files requested by leech peer is the Sender peer. Many peers can contain the content, but only one who is contributing the content to the leech peer can be named as Sender peer.

Leech Peer

A peer that requests the specific media content from other peers. Note that the leech peer can also contribute the downloaded media content (i.e., chunks) even the swarm is not completed, in such case, the leech peer will take on the role of sender peer for downloaded chunks.

Chunk Map

A peer list that indicates which chunks can be available for leech peer to playback smoothly.

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.

Video-on-demand (VoD)

The scenario where all clients are allowed to select and watch video content on demand.

Adaptive Streaming

Multiple alternate versions (different qualities and bitrates) of the same media content co-exist for the same streaming session; each alternate version corresponds to a different media quality level; peers can choose among the alternate versions for decode and playback.

Scalable Streaming

With Multiple Description Coding (MDC), multiple additive descriptions (that can be independently played-out) to refine the quality of the video when combined together. With Scalable Video Coding (SVC), nested dependent enhancement layers (hierarchical levels of quality), refine the quality of lower layers, from the lowest level (the playable Base Layer). With Multiple View Coding (MVC), multiple views allow the video to be played in 3D when the views are combined together.

Base Layer

The playable level in Scalable Video Coding (SVC) required by all upper level Enhancements Layers for proper decoding of the video.

Enhancement Layer

Enhancement differential quality level in Scalable Video Coding (SVC) used to produce a higher quality, higher definition video in terms of space (i.e., image resolution), time (i.e., frame rate) or Signal-to-Noise Ratio (SNR) when combined with the playable Base Layer.

Continuous Media

Media with an inherent notion of time, for example, speech, audio, video, timed text or timed metadata.

Media Component

An encoded version of one individual media type such as audio, video or timed text with specific attributes, e.g., bandwidth, language, or resolution.

3. Protocol Overview

3.1. Protocol Architecture

The functional entities involved in the PPSP Peer Protocol are Peers, which may support different capabilities.

Peers correspond to devices that actually participate in sharing a media content and are organized in (various) swarms corresponding each swarm to the group of peers streaming that content at any given time.

Each peer contacts a Tracker to advertise which information it has available. When a peer wishes to obtain information about the swarm, it contacts the Tracker to find other peers participating in that specific swarm.

The tracker is a logical entity that maintains the lists of peers storing/exchanging chunks for a specific Live media channel or VoD media streaming content, answers queries from peers and collects information on the activity of peers. A simplified network diagram showing this interaction of tracker and peers is depicted in Figure 1.

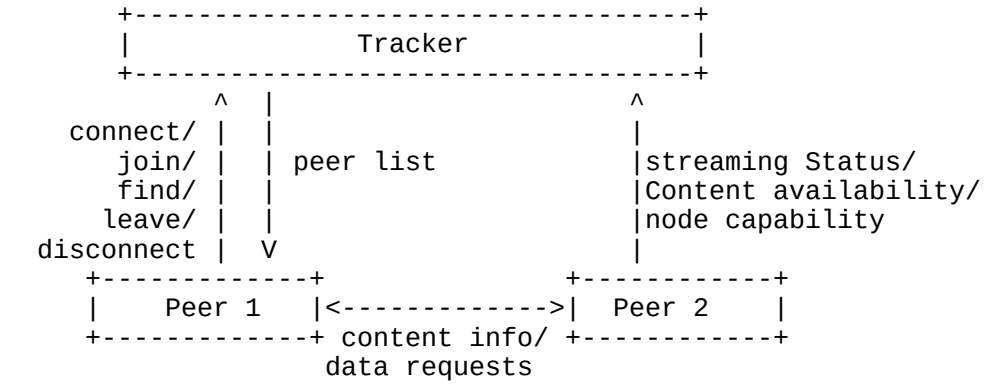


Figure 1: A PPSP streaming process

The signaling between PPSP Peers and trackers is done using a request/reply mechanism as defined in PPSP Tracker protocol [I-D.gu-ppsp-tracker-protocol].

This protocol can be used to connect peers that are sharing real-time streams of video or offline video, segmented in chunks. As for the streams of video, they can correspond to Live or Video on Demand streaming modes.

There are some significant differences between the details of these

scenarios, i.e., Live streaming, VoD and offline video. From a high level perspective the overall structure is quite similar. The optimal signaling flow for the different scenarios could also be different, but it depends on the real situation and on the implementer's choice

This draft defines a PULL based streaming signaling, as mandatory. However, a PUSH based or hybrid streaming signaling can optionally be considered.

For a PULL based Peer Protocol, the steps of signaling for a peer wishing to participate either in a Live streaming or a VoD or offline video is as follows (assuming the leech peer has already obtained from the Tracker a list of peers) and that, in case of traversing a NAT, performed ICE connectivity checks [I-D.li-ppsp-nat-traversal] with candidate peers using PPSP's own authentication method, as described in [I-D.gu-ppsp-tracker-protocol]:

1. The leech peer using PPSP Peer Protocol messages, establishes a connection to at least one of the peers in the Peerlist, based on the known PeerID and Peer IP address.

2. The peer sends request to candidate peers and the request could include one or more of the information described in below:

    * Request for the data availability of the candidate peer;

    * Notify its data availability to the candidate peer;

    * Request for the peer status of the candidate peer;

    * Notify its peer status to the candidate peer;

    * Request for additional peerlist;

    * Transport negotiation, wherein the requesting peer can have two choices:

        + Only support Mandatory Tranport Protocol;

        + Providing a list of supported Transport protocol.

3. Finally, the peers exchange the actual chunks of data, using the mechanism/protocol negotiated in the previous step.

In terms of Data Transport protocol negotiation, the leech peer can either inform the candidate that it supports a Mandatory Tranport Protocol or provides a list of supported Transport protocols. That

there are several options here to negotiate the connection model. The PPSP Peer Protocol may include new mechanisms to negotiate the protocol used to exchange data, or the offer-answer mechanism in SIP [RFC3261] (the IETF protocol for session establishment) along with SDP [RFC4566].

Note also that these mechanisms are not new protocols defined in PPSP, but existing protocols, and would eventually differ between an offline and a Live streaming scenario. Mechanisms such as flow control are handled in the negotiated Data Transport mechanism, not in the Peer Protocol itself.

3.2. Example Call Flow

This is a very high-level example of a session in which a leech peer joins a swarm, and retrieves some data (either via blocks or by streaming). The protocol used is indicated for each transaction. Note that not all of the communication shown in this figure are in scope of Peer Protocol, only those request/response followed by Peer Protocol are in scope.

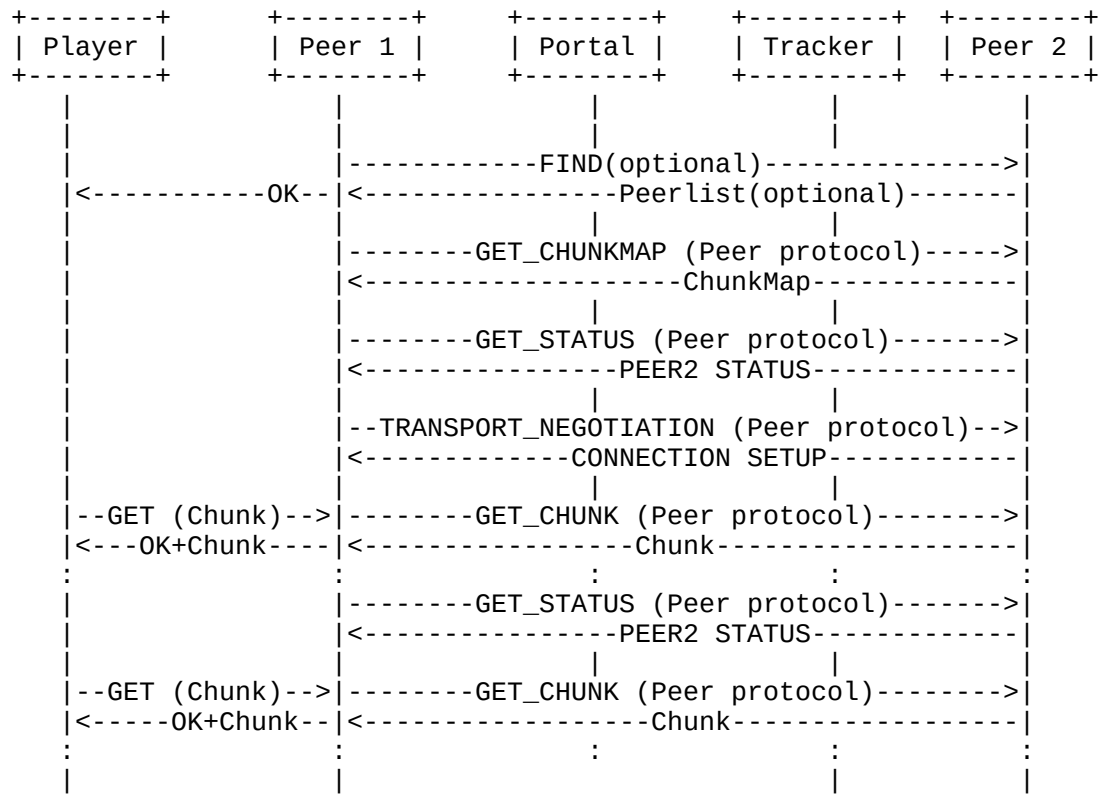


Figure 2: Example Call Flow

3.3. Chunk Scheduling

The goal of chunk trading is receiving the stream smoothly (and with small delay) and to cooperate in the distribution procedure. Peers need to exchange information about their current status to enable scheduling decisions. The information exchanged refers to the state of the peer with respect to the flow, i.e., a map of which chunks are needed by a peer to smoothly playback the stream.

This task means:

1. sending chunk maps to other nodes with the proper timing,

2. receiving chunk maps from other nodes and merging the information in the local buffer map.

3. besides chunk map exchange, the signaling includes Status/ Request/Select primitives used to trade chunks.

The core of the scheduler, not described in this specification, is the algorithm used to choose the chunks to be exchanged and the peers to communicate with.

4. Protocol Architecture

The PPSP Peer Protocol is a request-response protocol. Requests are sent, and responses returned to these requests. A single request generates a single response (neglecting fragmentation of messages).

As shown in example call flow depicted in Figure 2, the Peer protocol only provides signaling messages for obtaining additional peerlist (optionally), query for content availability and negotiation for transfer protocol. Peer protocol may also provide communication for peers to exchange information that can improve system performance.

The encoding for the signaling messages is not yet decided. Two encodings are proposed, a Text-based (HTTP/XML) and a Binary-based, described in Appendixes A and B. The authors will raise more discussion on the encoding, and will move the one that gets rough consensus of the PPSP WG to the draft text. In the Appendixes, some considerations are provided on each encoding based on the Mail List discussions.

The specific PPSP signaling messages are listed as following:

GET_PEERLIST:

The GET_PEERLIST message is sent from a leech peer to one or more remote peers in order for a peer to refresh/update the list of active peers in the swarm.

When receiving the GET_PEERLIST message, and if the message is well formed and accepted, the peer will search for the requested data and will respond to the leech peer with the peer list with PeerIDs (and respective IP Addresses) of sender peers that can provide the specific content.

GET_CHUNKMAP:

The GET_CHUNKMAP message is sent from a leech peer to one or more remote peers in order to receive the map of chunks of a content (of a swarm identified by SwarmID) the other peer presently stores. The chunk map returned by the other peer lists ranges of chunks.

When receiving the GET_CHUNKMAP message, and if the message is well formed and accepted, the peer will search for the requested data and will respond to the leech peer with the map of chunks it currently stores of the specific content.

GET_CHUNK:

The GET_CHUNK message is sent from a leech peer to sender peer in order to request the delivery of media content chunks.

When receiving the GET_CHUNK message, and if the message is well formed and accepted, the peer will search for the requested data and will respond to the leech peer with the specific chunks the leech peer requested.

GET_STATUS:

The GET_ STATUS message is sent from a leech peer to one or more remote peers in order to request the corresponding properties of the sender peers. The corresponding properties are enumerated in [draft-gu-ppsp-tracker-protocol], e.g., Caching Size, Bandwidth etc.

When receiving the GET_STATUS message, and if the message is well formed and accepted, the peer will search for the requested data and will respond to the leech peer with the specific parameters to the properties the leech peer requested.

TRANSPORT_NEGOTIATION:

The TRANSPORT_NEGOTIATION message is sent from a leech peer to a sender peer in order to negotiate the underlying transport protocol. Leech peer provide a set of transport protocols it supported to sender peer, and leave send peer to choose its preference. Reusing existing transport protocol to transfer data is recommended.

When receiving the TRANSPORT_NEGOTIATION message, and if the message is well formed and accepted, the sender peer will decide the transport protocol and will respond to the leech peer with the specific transport protocol the sender peer preferred.

5. Security Consideration

P2P streaming systems are subject to attacks by malicious/unfriendly peers/trackers that may eavesdrop on signaling, forge/deny information/knowledge about streaming content and/or its availability, impersonating to be another valid participant, or launch DoS attacks to a chosen victim.

No security system can guarantees complete security in an open P2P streaming system where participants may be malicious or uncooperative. The goal of security considerations described here is to provide sufficient protection for maintaining some security properties during the peer-peer communication even in the face of a large number of malicious peers.

As in typical Peer to Peer network, the most significant security issue is that the peers are untrusted. A peer may announce that it has a specific content, but the content might be just noise or it could be poisoned. A peer could also download a large number of chunks but upload very few of them. This problem can be alleviated by incentive mechanism, the goal of which is to reward honest peers and degrade dishonest peers.

5.1. Authentication

To protect the PPSP signaling from attackers pretending to be valid peers (or peers other than themselves) all messages received in the Tracker are required to be received from authorized peers.

For that purpose a peer must enroll in the system via a centralized enrollment server. The enrollment server is expected to provide a proper PeerID for the peer and information about the authentication mechanisms. The specification of the enrollment method and the provision of identifiers and authentication tokens is out of scope of this draft.

The authetication mechanism MUST allow the means for negotiating data security layer mechanisms to provide data integrity, data confidentiality, and other services, subject to local policies and security requirements.

5.2. Content Integrity Protection Against Polluting Peers/Trackers

Malicious peers may declaim ownership of popular content to the Tracker but serve polluted (i.e., decoy content or even virus/trojan infected contents) to other peers. This kind of pollution can be detected by incorporating a checksum distribution scheme for published sharing content. As content chunks of the same content are

transferred independently and concurrently, correspondent chunk-level checksums MUST be distributed from an authentic origin.

5.3. Residual Attacks and Mitigation

To mitigate the impact of sybil attackers, impersonating a large number of valid participants by repeatedly acquiring different peer identities, the enrollment server SHOULD carefully regulate the rate of peer/tracker admission.

There is no guarantee that a peer honestly report its status to the Tracker, or server authentic content to other peers as it claims to the Tracker. It is expected that a global trust mechanism, where the credit of each peer is accumulated from evaluations for previous transactions, may be taken into account by other peers when selecting partner for future transactions, helping to mitigate the impact of such malicious behaviors. A globally trusted Tracker MAY also take part of the trust mechanism by collecting evaluations, computing credit values and providing them to joining peers.

5.4. Pro-incentive Parameter Trustfulness

Properties for PEER_STATUS messages will consider pro-incentive parameters, which can enable the improvement of the performance of the whole P2P streaming system. Trustworthiness of these pro-incentive parameters is critical to the effectiveness of the incentive mechanisms. For example, ChunkMap is essential, and needs to be accurate. The P2P system should be designed in a way such that a peer will have the incentive to report truthfully its ChunkMap (otherwise it may penalize itself).

Furthermore, both the amount of upload and download should be reported to the Tracker to allow checking if there is any inconsistency between the upload and download report, and establish an appropriate credit/trust system.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[draft-ietf-p2psip-base] Jennings, C., Lowekamp, B., Ed., Rescorla, E., and H. Schulzrinne, "draft-ietf-p2psip-base-07", February 2010, <draft-ietf-p2psip-base>.

6.2. Informative References

[I-D.ietf-ppsp-problem-statement] Zhang, Y., Zong, N., Camarillo, V., Seng, J., and R. Yang, "Problem Statement of P2P Streaming Protocol (PPSP)", Januray 2011, <I-D.ietf-ppsp-problem-statement>.

[I-D.ietf-ppsp-reqs] Zong, N., Zhang, Y., Pascual, V., and C. Williams, "P2P Streaming Protocol (PPSP) Requirements", February 2011, <I-D.ietf-ppsp-reqs>.

[I-D.ietf-ppsp-survey] Gu, Y., Zong, N., Zhang, H., Zhang, Y., Camarillo, G., and Y. Liu, "Survey of P2P Streaming Applications", March 2011, <I-D.ietf-ppsp-survey>.

[I-D.gu-ppsp-tracker-protocol] Cruz, R., Nunes, M., Gu, Y., Xia, J., Bryan, D., Taveira, J., and D. Deng, "PPSP Tracker Protocol", March 2011, <I-D.gu-ppsp-tracker-protocol>.

[I-D.cruz-ppsp-http-peer-protocol] Gu, Y., Xia, J., Cruz, R., Nunes, M., Bryan, D., and J. Taveira, "PPSP HTTP-Based Peer Protocol", March 2011, <I-D.cruz-ppsp-http-peer-protocol>.

[I-D.li-ppsp-nat-traversal] Li , L., Wang , J., and W. Chen , "PPSP NAT Traversal", March 2011, <I-D.li-ppsp-nat-traversal>.

[BittorrentSpecification] "Bittorrent Protocol Specification v1.0", February 2010, <Bittorrent Specification>.

Appendix A. Binary Encoding

Binary Encoding is an encoding of data in plain text. More precisely, it is an encoding of binary data in a sequence of ASCII-printable characters. Binary Encoding is necessary for transmission of data when the channel or the protocol only allows ASCII-printable characters.

The PPSP Peer protocol can be carried on top of IP, UDP, RTP or TCP. But using which layer to carry peer protocol is out of scope in current stage.

The peer message header has the following format:

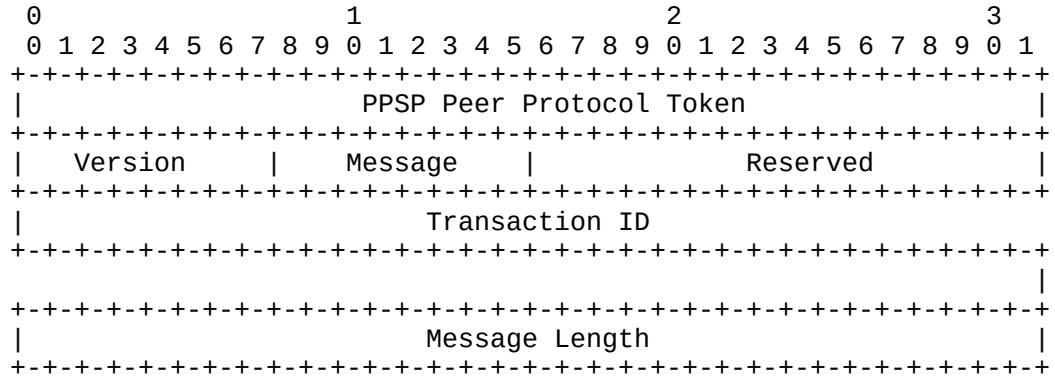


Figure 3: PPSP Peer message header

The fields have the following meaning:

PPSP Peer Protocol Token: 32 bits

A fixed token indicating to the receiver this message is a PPSP Peer Protocol message. The token field is four bytes long. This value MUST be set to 0x50505350, the string "PPSP".

Version: 8 bits The version of the PPSP peer protocol being used in the form of a fixed point integer between 0.1 and 25.4. For the version of the protocol described in this document, this field MUST be set to 0.1. The version field is one byte long.

Message Types: 8 bits

Message types currently have two kinds of value: Request and Response.

Reserved: 16 bits

Not to be assigned. Reserved values are held for special uses, such as to extend the namespace when it becomes exhausted. Reserved values are not available for general assignment.

Transaction ID: 64 bits

Identifies the transaction and also allows receivers to disambiguate transactions which are otherwise identical. Responses use the same Transaction ID as the request they correspond to. Transaction IDs are also used for fragment reassembly.

Message Length: 32 bits:

The length of the message, including header, in bytes. Note if the message is fragmented, this is the length of this message, not the total length of all assembled fragments.

A.1. Methods in Peer messages

To improve the compatibility of the peer methods, the method fields in message extension MUST be encoded as TLV elements as described below and sketched in Figure 4:

To improve the compatibility of the peer methods, the method fields in message extension MUST be encoded as TLV elements as described below and sketched in Figure 4:

- Type: A single-octet identifier that defines the type of the parameter represented in this TLV element.
- Length: A two-octet field that indicates the length (in octets) of the TLV element excluding the Type and Length fields, and the 8-bit Reserved field between them. Note that this length does not include any padding that is required for alignment.

o Value: Variable-size set of octets that contains the specific value for the parameter.

In the extensions, the Reserved field SHALL be set to zero and ignored. If a TLV element does not fall on a 32-bit boundary, the last word MUST be padded to the boundary using further bits set to zero.

In a peer message, any method extension MUST be placed after the mandatory message header. The extensions MAY be placed in any order.

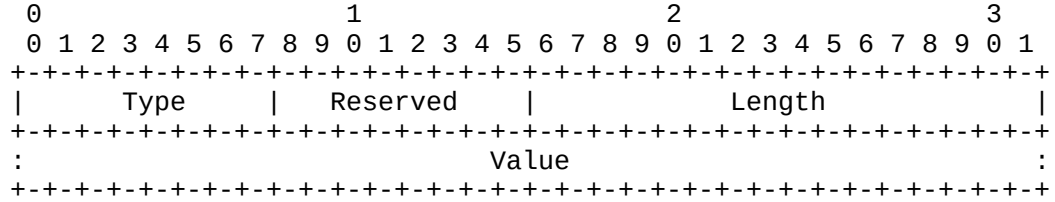


Figure 4: Structure of a TLV element

Method Type: 8 bits

Indicates the method type for the message. There are five method types: GET_PEERLIST, GET_CHUNKMAP, GET_CHUNK, GET_ PROPERTY and TRANSPORT_NEGOTIATION. They are counted from 1 to 5.

Method Body Length: 24 bits

The length of the method body in bytes.

A.1.1. GET_PEERLIST

Peerlist is composed of several pairs of Peer ID and Peer IP. Peer ID is a 128 bit integer that is unique in the P2P streaming system. That’s no matter there is a centralized tracker or several distributed trackers in the streaming system, a peer ID should be unique.

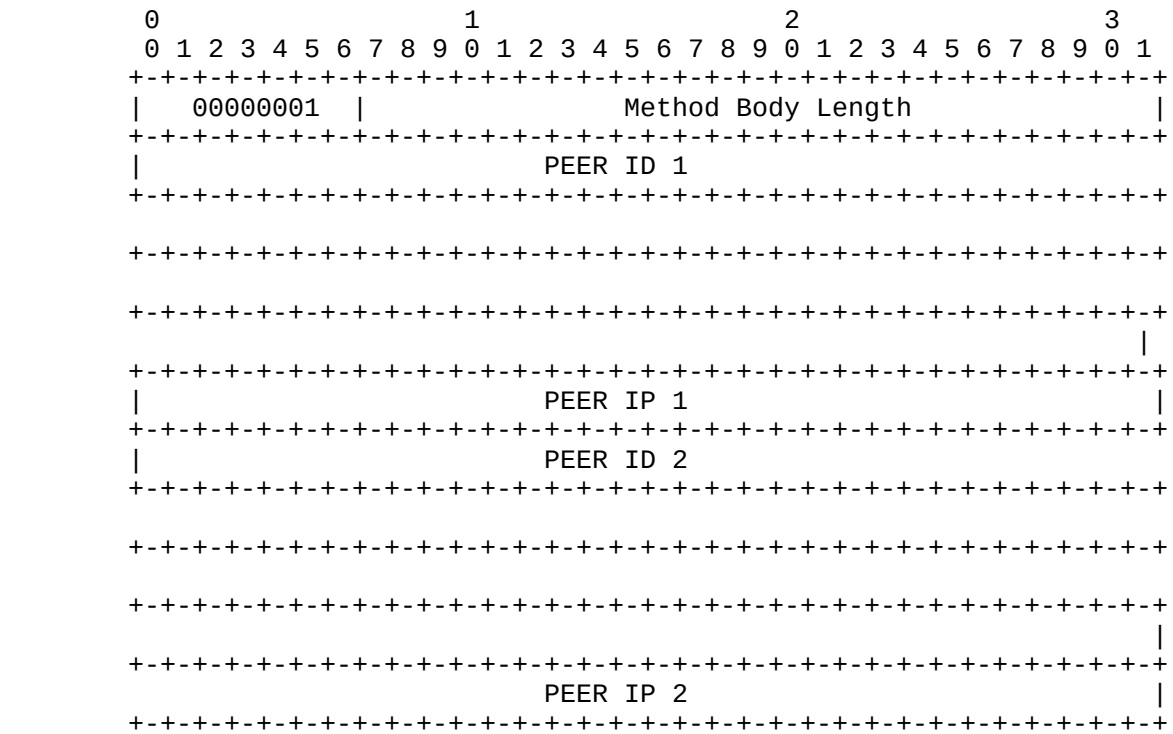


Figure 5: GET_PEERLIST Method Body

A.1.2. GET_CHUNKMAP

Chunkmap of a content (a swarm identified by SwarmID)

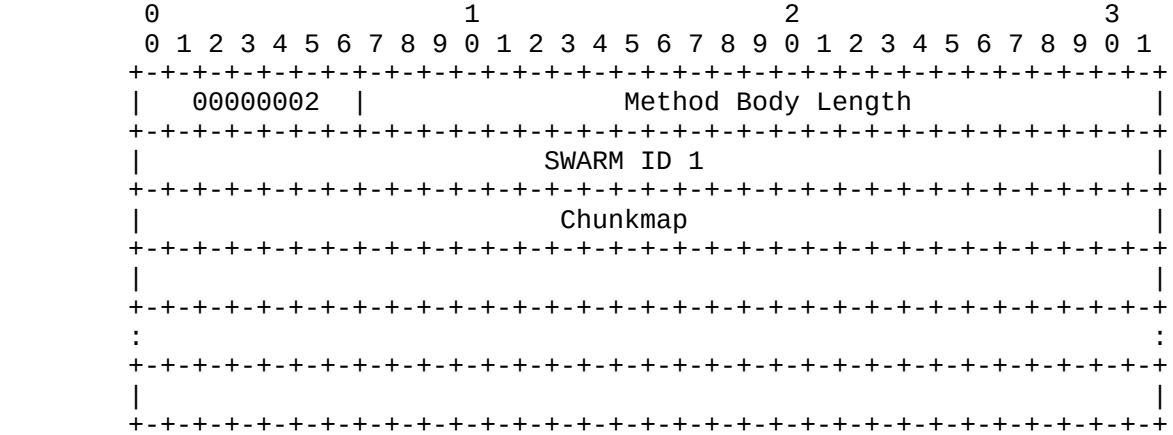


Figure 6: GET_CHUNKMAP Method Body

A.1.3. GET_CHUNK

[TBD]

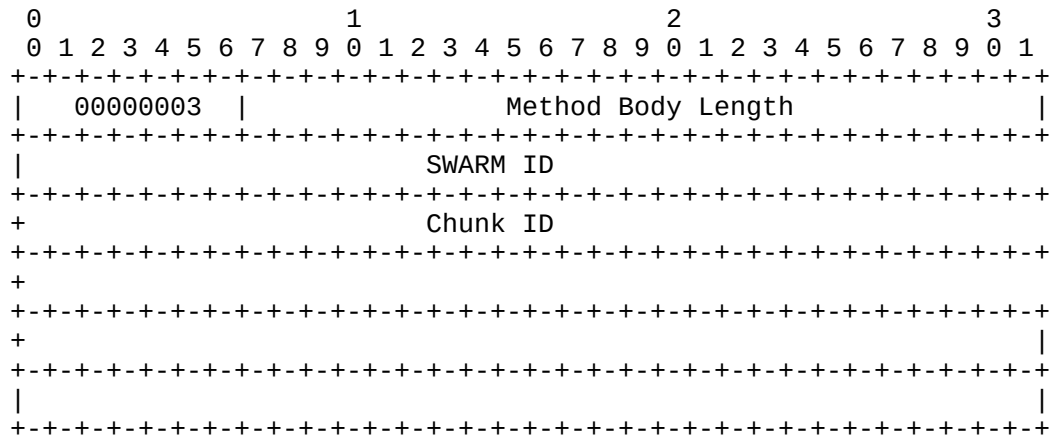


Figure 6: GET_CHUNKMAP Method Body

A.1.4. GET_STATUS

Several property types are defined in I-D.gu-ppsp-tracker-protocol. But not all of the property types are reasonable to be used in peer protocol. So we just list the following property types. New types can be easily added.

PROPERTY	Description	Code
CachingSize	Caching size: available size for caching	0x01
Bandwidth	Bandwidth: available bandwidth	0x02
LinkNumber	Link number: acceptable links for remote peer	0x03
Certificate	Certificate: certificate of the peer	0x04

Table 1: Status changed between peers

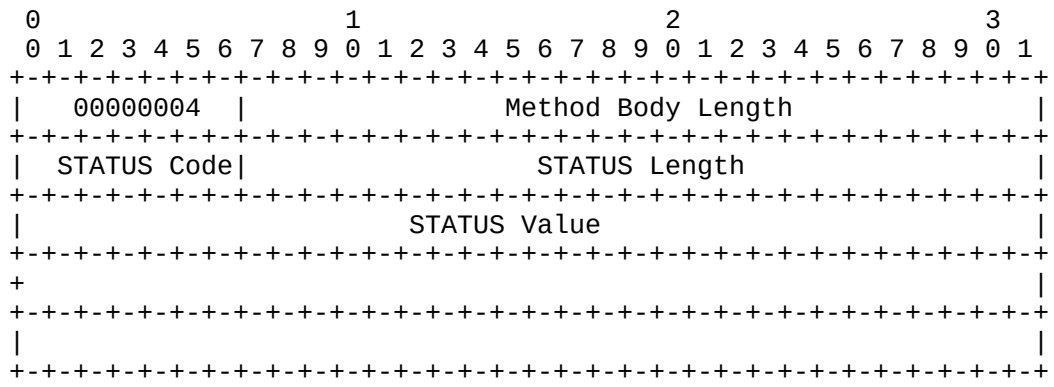


Figure 6: GET_STATUS Method Body

A.1.5. TRANSPORT_NEGOTIATION

To Do: Define mandatory transport protocol and some optional transport protocol.

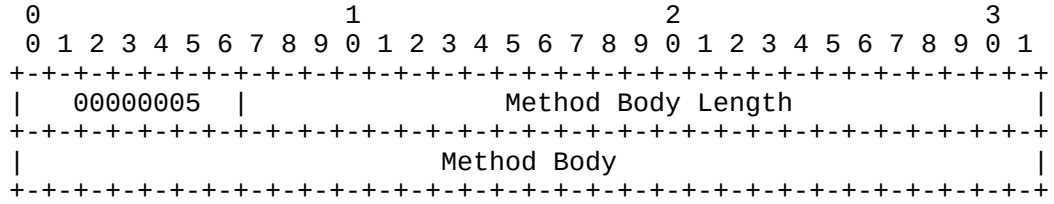


Figure 7: TRANSPORT_NEGOTIATION Method Body

Appendix B. HTTP/XML Encoding

The PPSP Peer Protocol HTTP/XML encoding messages follow the request and response standard formats for HTTP Request and Response messages [RFC2616].

B.1. HTTP/XML Encoding

A Request message is a standard HTTP Request generated by the HTTP Client Peer with the following syntax:

```
<Method> /<Resource> HTTP/1.1
Host: <Host>
```

The HTTP Method and URI path (the Resource) indicates the operation requested. The current proposal uses only HTTP POST as a mechanism for the request messages.

The Host header follows the standard rules for the HTTP 1.1 Host Header.

The Response message is also a standard HTTP Response generated by the HTTP Serving Peer with the following syntax:

```
HTTP/1.1 <StatusCode> <StatusMsg>
Content-Lenght: <ContentLenght>
Content-Type: <ContentType>
Content-Encoding: <ContentCoding>
<Response_Body>
```

The body for both Request and Response messages are encoded in XML for all the PPSP Peer Protocols messages, with the following schema (the XML information being method specific):

```
<?xml version="1.0" encoding="utf-8"?>
<ProtocolName version="#.#">
   <Method>***</Method>     <!-- for the Request method -->
   <Response>***</Response> <!-- for the Response method -->
   <TransactionID>###</TransactionID>
     ...XML information specific of the Method...
</ProtocolName>
```

In the XML body, the *** represents alphanumeric data and ### represents numeric data to be inserted. The <Method> corresponds to the method type for the message, the <Response> corresponds to the response method type of the message and the element <TransactionID> uniquely identifies the transaction.

The Response message MAY use Content-Encoding entity-header with "gzip" compression scheme [RFC2616] for faster transmission times and less network bandwidth usage.

B.2. Method Fields

Table B 1 and Table B 2 define the valid string representations for the requests and responses, respectively. These values MUST be treated as case-insensitive.

PPSP Request	XML Request Value String
GET_PEERLIST	GET_PEERLIST
GET_CHUNKMAP	GET_CHUNKMAP
GET_CHUNK	GET_CHUNK
PEER_STATUS	PEER_STATUS
TRANSPORT_NEGOTIATION	TRANSP_NEGO

Table B 1: Valid Strings for Requests

Response Method Name	HTTP Response Mechanism	XML Response Value
SUCCESSFUL (OK)	200 OK	OK
INVALID SYNTAX	400 Bad Request	INVALID SYNTAX
VERSION NOT SUPPORTED	400 Bad Request	VERSION NOT SUPPORTED
AUTHENTICATION REQUIRED	401 Unauthorized	AUTHENTICATION REQUIRED
MESSAGE FORBIDDEN	403 Forbidden	MESSAGE FORBIDDEN
OBJECT NOT FOUND	404 Not Found	OBJECT NOT FOUND
INTERNAL ERROR	500 Internal Server Error	INTERNAL ERROR
TEMPORARILY OVERLOADED	503 Service Unavailable	TEMPORARILY OVERLOADED

Table B 2: Valid Strings for Responses

B.3. Message Processing

When a PPSP Peer Protocol message is received in a peer, some basic processing is performed, regardless of the message type. Upon reception, a message is examined to ensure that it is properly formed. The receiver MUST check that the HTTP message itself is properly formed, and if not appropriate standard HTTP errors MUST be generated. The receiver must also verify that the XML body is properly formed. If the message is found to be incorrectly formed or the length does not match the length encoded in the header, the receiver MUST reply with an HTTP 400 response with a PPSP XML body with the Response method set to INVALID SYNTAX.

B.4. GET_PEERLIST Message

The GET_PEERLIST message is sent from a client peer to a selected serving peer in order for a peer to refresh/update the list of active peers in the swarm.

The Request message uses a HTTP POST method with the following body:

```
<?xml version="1.0" encoding="utf-8"?>
<PPSPPeerProtocol version="#.#">
   <Method>GET_PEERLIST</Method>
   <PeerID>***</PeerID>
   <SwarmID>***</SwarmID>
   <TransactionID>###</TransactionID>
</PPSPPeerProtocol>
```

The sender MUST properly form the XML body, MUST set the Method string to GET_PEERLIST, MUST set the PeerID to the PeerID of the peer, MUST set the SwarmID to the joined swarm identifier and randomly generate and set the TransactionID value.

When receiving the GET_PEERLIST message, and if the message is well formed and accepted, the peer will search for the requested data and will respond to the requesting peer with an HTTP 200 OK message response with a PPSP XML payload SUCCESSFUL, as well as the peer list with PeerIDs (and respective IP Addresses) of peers that can provide the specific content.

The response MUST have the same TransactionID value as the request.

An example of the Response message structure is the following:

```
<?xml version="1.0" encoding="utf-8"?>
<PPSPPeerProtocol version="#.#">
   <Response>OK</Response>
   <SwarmID>***</SwarmID>
   <TransactionID>###</TransactionID>
   <PeerInfoList>
      <PeerInfo>
         <PeerID>***</PeerID>
         <PeerType>***</PeerType>
         <PeerAddresses>
            <PeerAddress ip="##.##.##.##"
                         port="###" />
            <PeerAddress ip="hh:hh:hh:hh:hh:hh:hh:hh"
                         port="###" />
         </PeerAddresses>
         <PeerLocation>****</PeerLocation>
         <ConnectionType>***</ConnectionType>
         <EndPointRankCost>###</EndPointRankCost>
      </PeerInfo>
      <PeerInfo>
         <PeerID>***</PeerID>
         <PeerType>***</PeerType>
         <PeerAddresses>
            <PeerAddress ip="##.##.##.##"
                         port="###" />
            <PeerAddress ip="hh:hh:hh:hh:hh:hh:hh:hh"
                         port="###" />
         </PeerAddresses>
         <PeerLocation>****</PeerLocation>
         <ConnectionType>***</ConnectionType>
         <EndPointRankCost>###</EndPointRankCost>
      </PeerInfo>
   </PeerInfoList>
</PPSPPeerProtocol>
```

The element <PeerInfoList> MAY contain multiple <PeerInfo> child elements.

The element <PeerAddresses> MAY contain multiple <PeerAddress> child elements with attributes "ip" and "port" corresponding to each of the network interfaces of the peer. The "ip" attribute can be expressed in dotted decimal format for IPv4 or 16-bit hexadecimal values (hh) separated by colons (:) for IPv6.

The elements <PeerLocation> and <ConnectionType> have a string format, and together with the element <EndPointRankCost> of numerical integer format, form a set of information related to peer location.

B.5. GET_CHUNKMAP Message

The GET_CHUNKMAP message is sent from a client peer to a selected serving peer in order to receive the map of chunks of a content (of a swarm identified by SwarmID) the other peer presently stores. The chunk map returned by the other peer lists ranges of chunks. The Request message uses a HTTP POST method with the following body:

```
<?xml version="1.0" encoding="utf-8"?>
<PPSPPeerProtocol version="#.#">
   <Method>GET_CHUNKMAP</Method>
   <PeerID>***</PeerID>
   <SwarmID>***</SwarmID>
   <TransactionID>###</TransactionID>
</PPSPPeerProtocol>
```

The sender MUST properly form the XML body, MUST set the Method string to GET_CHUNKMAP, MUST set the PeerID to the PeerID of the peer, MUST set the SwarmID to the joined swarm identifier and randomly generate and set the TransactionID value.

When receiving the GET_CHUNKMAP message, and if the message is well formed and accepted, the peer will search for the requested data and will respond to the requesting peer with an HTTP 200 OK message response with a PPSP XML payload SUCCESSFUL, as well as the map of chunks it currently stores of the specific content.

The response MUST have the same TransactionID value as the request.

The Response message is an HTTP 200 OK message with the following body, exemplified for a video component of a media clip:

```
<?xml version="1.0" encoding="utf-8"?>
<PPSPPeerProtocol version="#.#">
   <Response>OK</Response>
   <TransactionID>###</TransactionID>
   <StreamInfo>
      <SwarmID>***</SwarmID>
      <Clip>
         <Name>***</Name>
         <ChunkSegments type="video/audio/etc">
            <ChunkSegment from="###" to="###"
                          bitmapSize="###">
               ...(base64 string)...
           </ChunkSegment>
         </ChunkSegments>
      </Clip>
   </StreamInfo>
```

```
      </PPSPPeerProtocol>
```

The element <StreamInfo> MAY contain multiple <Clip> child elements.

The element <ChunkSegments> has an attribute "type" that indicates the type of media of the corresponding chunks.

A <ChunkSegments> element MAY contain multiple <ChunkSegment> child elements with attributes "from" and "to" corresponding to ranges of contiguous chunks. The "from", "to", and "bitmapSize" attributes are expressed as integer number string format. The <ChunkSegment> content corresponds to the chunk map, and is represented as base64 encoded string.

B.6. GET_CHUNK Message

The GET_CHUNK message is sent from a client peer to a serving peer in order to request the delivery of media content chunks. The Request message uses a HTTP POST method with the following body:

```
      <?xml version="1.0" encoding="utf-8"?>
      <PPSPPeerProtocol version="#.#">
         <Method>GET_CHUNK</Method>
         <PeerID>***</PeerID>
         <SwarmID>***</SwarmID>
         <TransactionID>###</TransactionID>
      </PPSPPeerProtocol>
```

The sender MUST properly for the HTTP request for a POST method including the URI path (the Resource) of the chunk. The sender MUST also properly form the XML body, MUST set the Method string to GET_CHUNK, MUST set the PeerID to the PeerID of the peer, MUST set the SwarmID to the joined swarm identifier and randomly generate and set the TransactionID value.

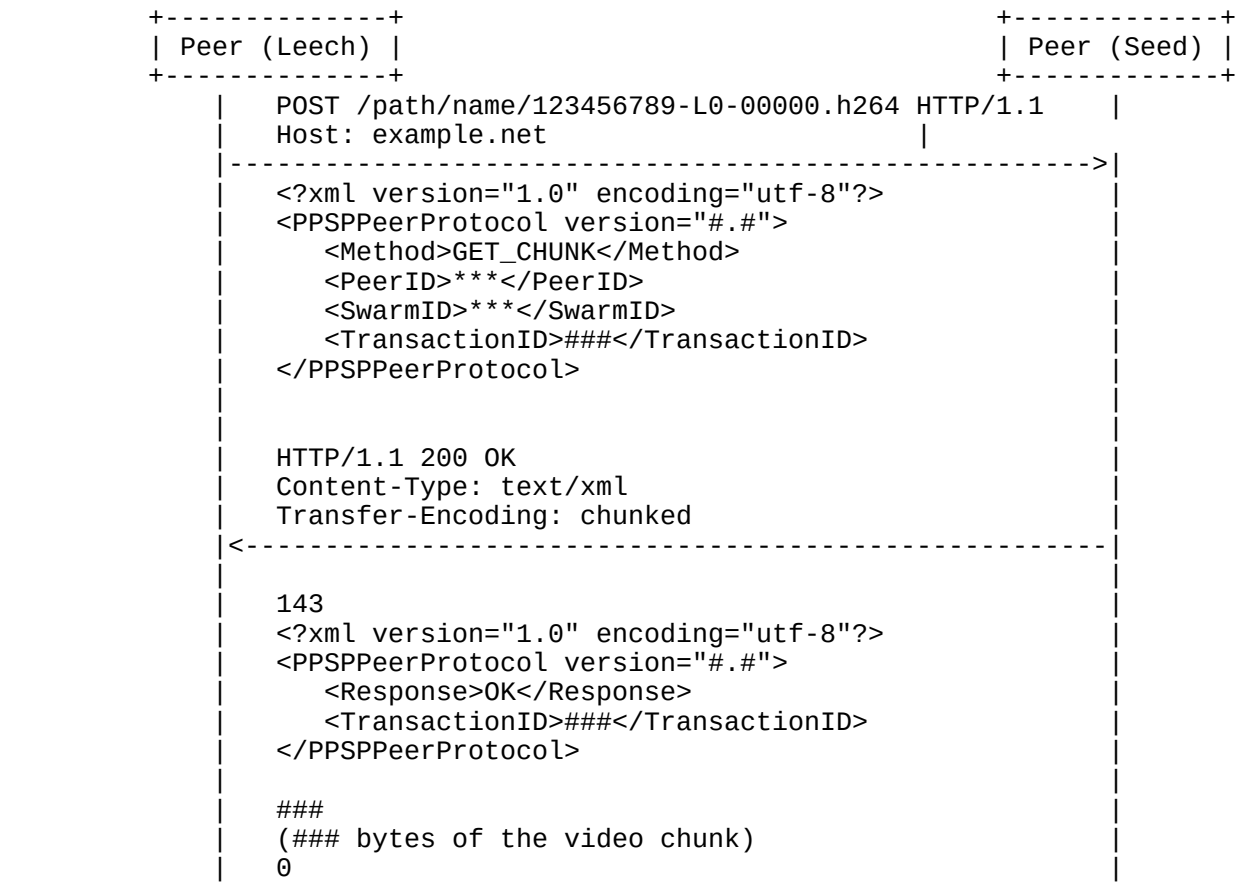


Figure B 1: Example of GET_CHUNK message sequence (simplified)

When receiving the GET_CHUNK message, and if the message is well formed and accepted, the peer will search for the requested data and will respond to the requesting peer with an HTTP 200 OK message response with a PPSP XML payload SUCCESSFUL.

The Response message is an HTTP 200 OK message. If The Data Transport Protocol negotiated is also HTTP/XML, the body of the response to GET_CHUNK can be immediately followed by the chunk data transfer, as shown in Figure B 1.

The response MUST have the same TransactionID as the request.

B.7. PEER_STATUS Message

The PEER_STATUS message is sent from a serving peer to a client peer to indicate its participation status. The information conveyed may include information related to chunk trading like "choke" (to inform the other peer of the intention to stop sending data to it) and "unchoke" (to inform the other peer of the intention to start/re-start sending data to it).

The Request message uses a HTTP POST method with the following body:

```
<?xml version="1.0" encoding="utf-8"?>
<PPSPPeerProtocol version="#.#">
   <Method>PEER_STATUS</Method>
   <PeerID>***</PeerID>
   <SwarmID>***</SwarmID>
   <TransactionID>###</TransactionID>
   <Status>(choke/unchoke)</Status>
</PPSPPeerProtocol>
```

The sender MUST properly form the XML body, MUST set the Method string to PEER_STATUS, MUST set the PeerID to the PeerID of the peer, MUST set the SwarmID to the joined swarm identifier and randomly generate and set the TransactionID value.

When receiving the PEER_STATUS message, and if the message is well formed and accepted, the peer will respond to the requesting peer with an HTTP 200 OK message response with a PPSP XML payload SUCCESSFUL.

If the status signal received is "choke" the peer will stop requesting chunks from the other peer until receiving an "unchoke" status signal.

The only element currently defined in the request message is <Status>, assuming values of "choke" and "unchoke", but, in future, other values may be added.

The Response message is an HTTP 200 OK message with the following body.

```
<?xml version="1.0" encoding="utf-8"?>
<PPSPPeerProtocol version="#.#">
   <Response>OK</Response>
   <TransactionID>###</TransactionID>
</PPSPPeerProtocol>
```

The response MUST have the same TransactionID value as the request.

The only element currently defined in the response message is the <TransactionID>, but, in future, other elements may be added, for example, containing statistical data or other primitives for chunk trading negotiation.

B.8. TRANSPORT_NEGOTIATION Message

To Do: Define message format, mandatory transport protocol and some optional transport protocols.

Authors' Addresses

Yingjie Gu
Huawei
No.101 Software Avenue
Nanjing, Jiangsu Province 210012
P.R.China

Phone: +86-25-56622638
Email: guyingjie@huawei.com

Jinwei Xia
Huawei
Software No.101
Nanjing, Yuhuatai District 210012
China

Phone: +86-025-86622310
Email: xiajinwei@huawei.com

Rui Santos Cruz
IST/INESC-ID/INOV

Phone: +351.939060939
Email: rui.cruz@ieee.org

Mario Serafim Nunes
IST/INESC-ID/INOV
Rua Alves Redol, n.9
1000-029 LISBOA
Portugal

Phone: +351.213100256
Email: mario.nunes@inov.pt

David A. Bryan
Polycom
San Jose, CA, USA,
USA

Phone:
Email: dbryan@ethernot.org

Joao P. Taveira
ID/INOV

Email: joao.silva@inov.pt

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, andother 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

Table of Contents

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 sever 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.

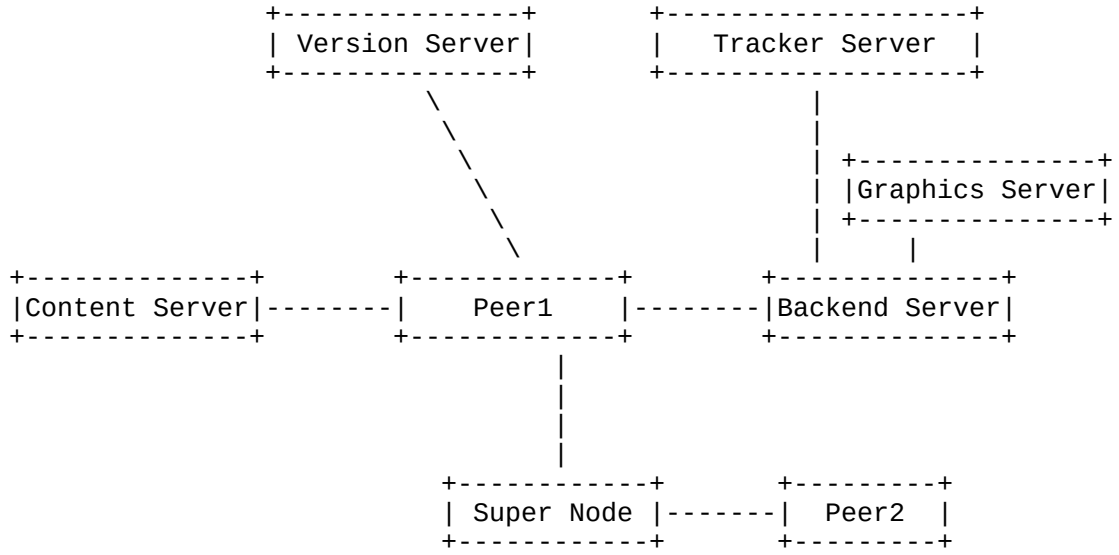


Figure 1, Architecture of Joost system

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.

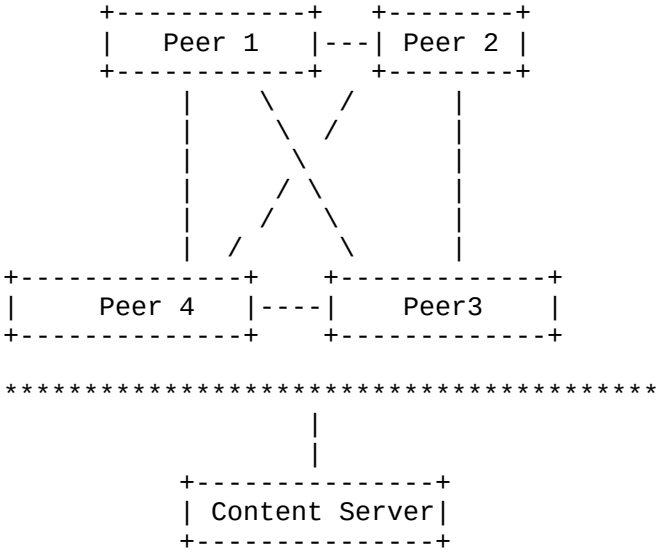


Figure 2, Architecture of Octoshape system

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]

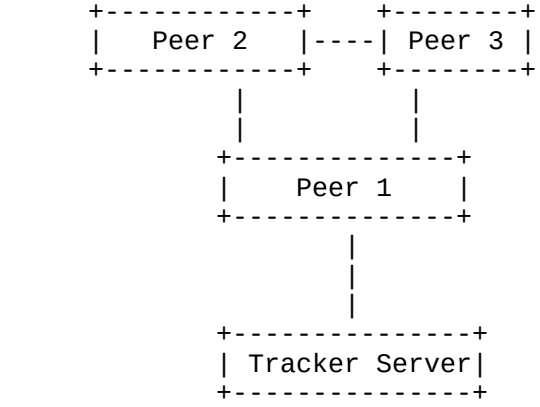


Figure 3, Architecture of PPlive system

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 reliabily streams media among peers using the mesh structure.

Figure 4 depcits 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.

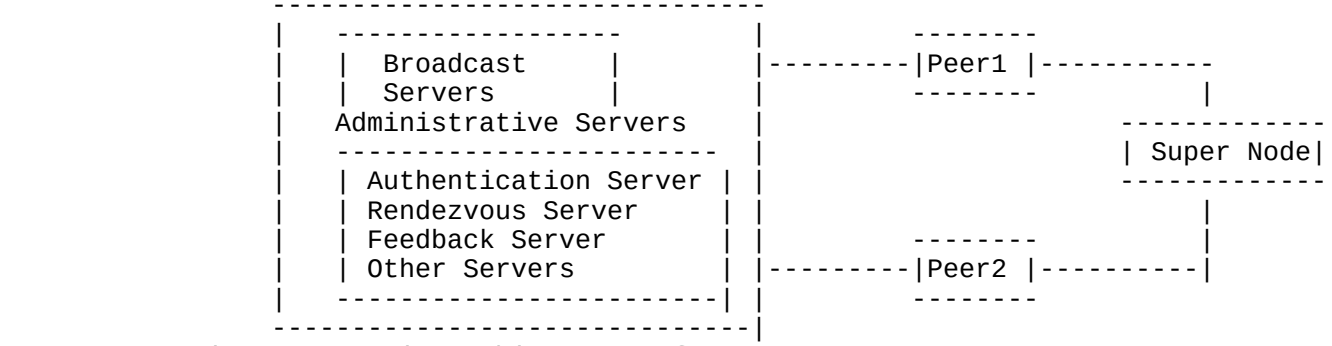


Figure 4, Basic architecture of Zattoo system

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 Bandwdith Estimation Server to initially estimate the amount of available uplink bandwith 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.

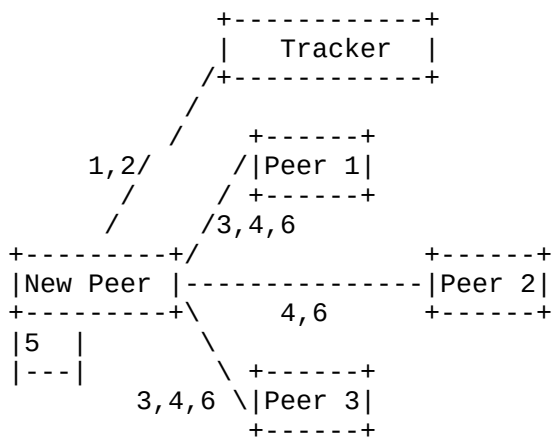


Figure 5, Main components and steps of PPLive, PPStream, SopCast and Tvants

The main steps are:

(1) A new peer registers with tracker / distributed hash table (DHT) to join the peer group which shares a same channel / media content;

(2) Tracker / DHT returns an initial peer list to the new peer;

(3) The new peer harvests peer lists by gossiping (i.e. exchange peer list) with the peers in the initial peer list to aggregate more peers sharing the channel / media content;

(4) The new peer randomly (or with some guide) selects some peers from its peer list to connect and exchange peer information (e.g. buffer map, peer status, etc) with connected peers to know where to get what data;

(5) The new peer decides what data should be requested in which order / priority using some scheduling algorithm and the peer information obtained in Step (4);

(6) The new peer requests the data from some connected peers.

3.2. Tree-based P2P streaming systems

3.2.1. PeerCast

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.

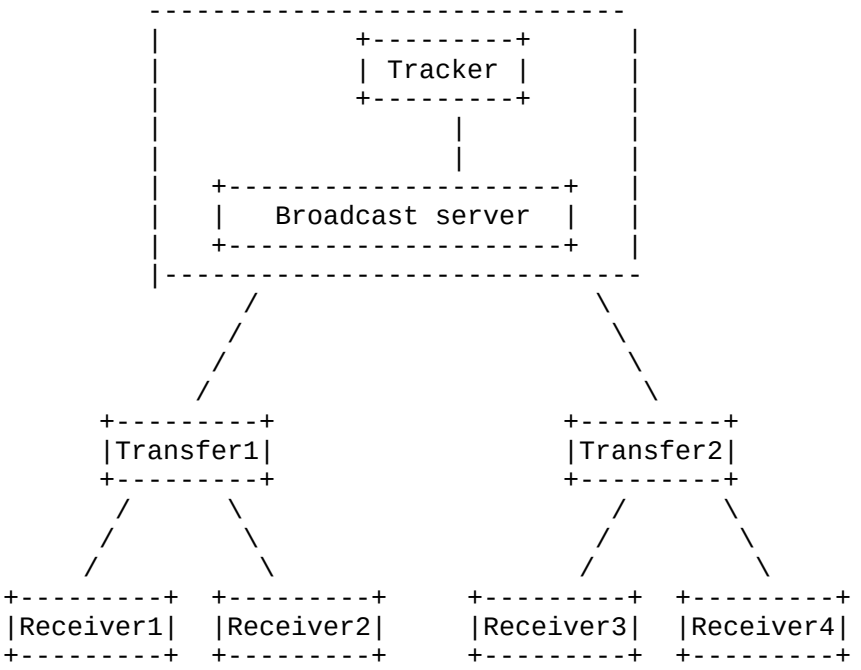


Figure 6, Architecture of PeerCast system

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.

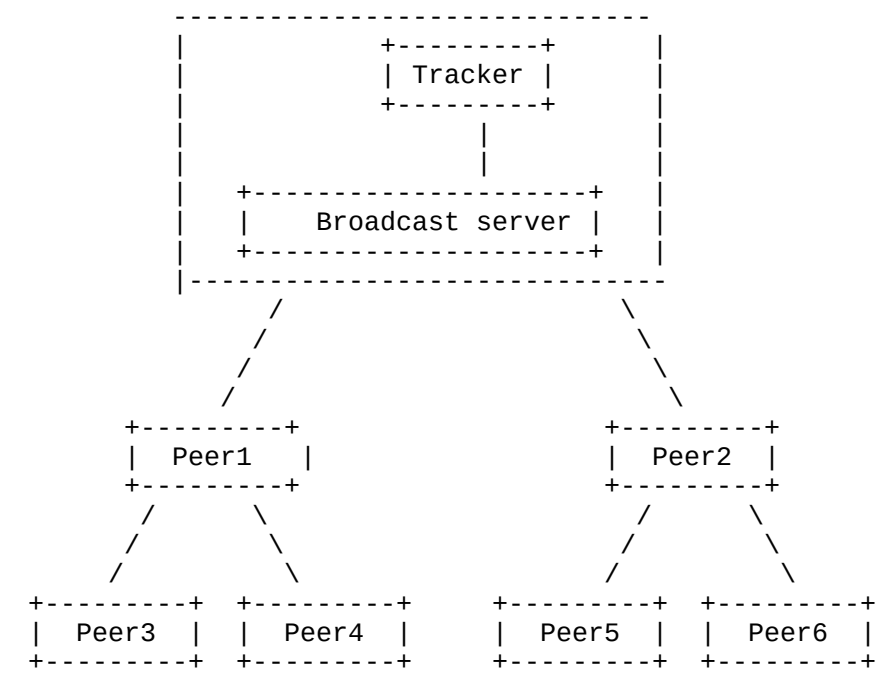


Figure 7, Architecture of ESM system

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.

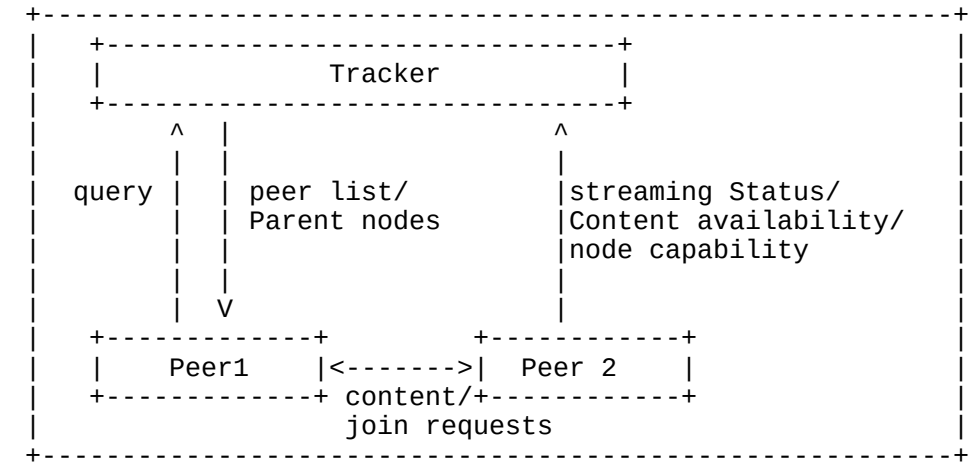


Figure 8, A common P2P streaming process model

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]
"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

PPSP
INTERNET-DRAFT
Intended Status: Standards Track
Expires: August 27, 2012

Rui S. Cruz
Mario S. Nunes
IST/INESC-ID/INOV
Yingjie Gu
Jinwei Xia
Huawei
David A. Bryan
Polycom
Joao P. Taveira
IST/INOV
Deng Lingli
China Mobile

February 24, 2012

PPSP Tracker Protocol (PPSP-TP)
draft-gu-ppsp-tracker-protocol-07

Abstract

This document specifies the Peer-to-Peer Streaming Protocol--Tracker Protocol (PPSP-TP), an application-layer control (signaling) protocol for the exchange of meta information between trackers and peers, such as initial offer/request of participation in multimedia content streaming, content information, peer lists and reports of activity and status. The specification outlines the architecture of the protocol and its functionality, and describes message flows, message processing instructions, message formats, formal syntax and semantics. The PPSP Tracker Protocol enables cooperating peers to form content streaming overlay networks to support near real-time Structured Media content (audio, video, associated text/metadata) delivery, such as adaptive multi-rate, layered (scalable) and multi-view (3D), in live, time-shifted and on-demand modes.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright and License Notice

Table of Contents

1. Introduction

The Peer-to-Peer Streaming Protocol (PPSP) is composed of two protocols: the PPSP Tracker Protocol and the PPSP Peer Protocol. [I-D.ietf-ppsp-problem-statement] specifies that the Tracker Protocol should standardize format/encoding of information and messages between PPSP peers and PPSP trackers and [I-D.ietf-ppsp-reqs] defines the requirements.

The PPSP Tracker Protocol provides communication between trackers and peers, by which peers send meta information to trackers, report streaming status and obtain peer lists from trackers.

The PPSP architecture requires PPSP peers able to communicate with a tracker in order to participate in a particular streaming content swarm. This centralized tracker service is used by PPSP peers for content registration and location. Content information metafiles (Media Presentation Descriptions) are also stored in the tracker system allowing active peers in the swarm to interpret content structure.

The signaling and the media data transfer between PPSP peers is not in the scope of this specification.

This document describes the PPSP Tracker protocol and how it satisfies the requirements for the IETF Peer-to-Peer Streaming Protocol (PPSP-TP), in order to derive the implications for the standardization of the PPSP streaming protocols and to identify open issues and promote further discussion.

1.1. Use Scenarios and Streaming Modes

This section is tutorial in nature and does not contain any normative statements.

This section describes some aspects of the use of PPSP-TP. The examples were chosen to illustrate the basic operation, but not to limit what PPSP-TP may be used for.

The functional entities related to PPSP protocols are the Client Media Player, the service Portal, the tracker and the peers. The complete description of these entities is not discussed here, as not in the scope the specification.

The Client Media Player is a logical entity providing direct interface to the end user at the client device, and includes the functions to select, request, decode and render contents. The Client Media Player may interface with the local peer application using

request and response standard formats for HTTP Request and Response messages [RFC2616].

The service Portal is a logical entity typically used for client enrollment and content information publishing, searching and retrieval.

The tracker is a logical entity that maintains the lists of PPSP active peers storing and exchanging a specific media content. The tracker also stores the status of active peers in swarms, to help in the selection of appropriate peers for a requesting peer. The tracker can be realized by geographically distributed tracker nodes or multiple server nodes in a data center, increasing the content availability, the service robustness and the network scalability or reliability. The management and locating of content index information are totally internal behaviors of the tracker system, which is invisible to the PPSP Peer [I-D.xiao-ppsp-reload-distributed-tracker].

The peer is also a logical entity embedding the P2P core engine, with a client serving side interface to respond to Client Media Player requests and a network side interface to exchange data and PPSP signaling with trackers and other peers.

The streaming technique is chunk-based, i.e., client peers obtain media chunks from serving peers and handle the buffering that is necessary for the playback processes during the download of the media chunks.

In Live streaming, all end users are interested in a specific media coming from an ongoing program, which means that all respective peers share nearly the same streaming content at a given point of time. Peers may store the live media for further distribution (known as time-shift TV), where the stored media is distributed in a VoD-like manner.

In VoD, different end users watch different parts of the recorded media content during a past event. In this case, each respective peer obtains from other peers the information on media chunks they store and then gets the required media from a selected set of those peers. While watching VoD, an end user can also switch to any place of the content, e.g., skip the credits part, or skip the part that it is not interested in. In this case the respective participating peer may not store all the content segments. From the whole swarm point of view, the participating peers typically store different parts of content.

1.2. Assumptions

This section is tutorial in nature and does not contain any normative statements.

The process used for media streaming distribution assumes a segment (chunk) transfer scheme whereby the original content (that can be encoded using adaptive or scalable techniques) is chopped into small segments (and subsegments) having the following representations:

1. Adaptive - alternate representations with different qualities and bitrates; a single represention is non-adaptive;
2. Scalable description levels - multiple additive descriptions (i.e., addition of descriptions refine the quality of the video);
3. Scalable layered levels - nested dependent layers corresponding to several hierarchical levels of quality, i.e., higher enhancement layers refine the quality of the video of lower layers.
4. Scalable multiple views - views correspond to mono (2D) and stereoscopic (3D) videos, with several hierarchical levels of quality.

These streaming distribution techniques support dynamic variations in video streaming quality while ensuring support for a plethora of end user devices and network connections.

1.2.1. Enrollment and Bootstrap

In order to join an existing P2P streaming service and to participate in content sharing, any peer must first locate a tracker, using for example, the following method (as illustrated in Figure 1):

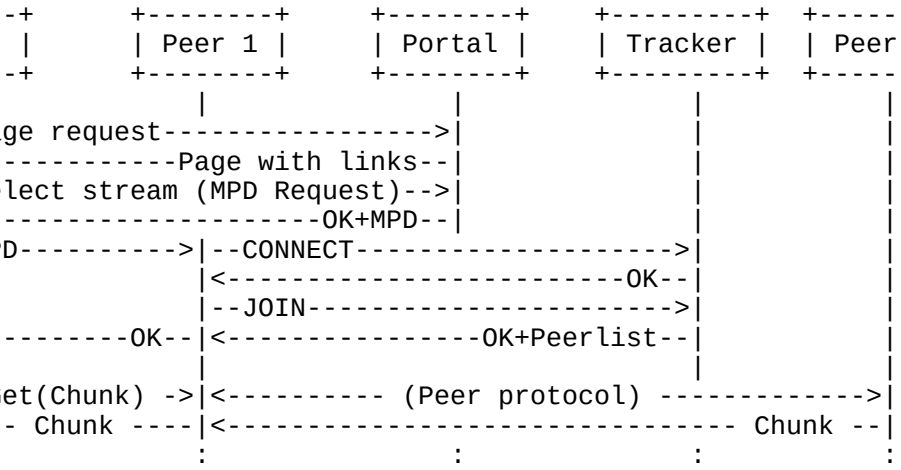


Figure 1: A typical PPSP session

1. From a service provider provisioning mechanism: this is a typical case used on the provider Super-Seeders (edge caches and/or Media Servers).
2. From a web page: a Publishing and Searching Portal may provide tracker location information to end users.
3. From the MPD file of a content: this metainfo file must contain information about the address of one or more trackers (that can be grouped by tiers of priority) which are controlling the swarm for that media content.

In order to be able to bootstrap, a peer must first obtain a Peer-ID (identity associated with the end user authentication credentials) and any required security certificates or authorization tokens from an enrollment service (end user registration).

The specification of the mechanism used to obtain a Peer-ID, certificates or tokens is not in the scope of this document.

1.2.2. NAT Traversal

It is assumed that all trackers must be in the public Internet and have been placed there deliberately. This document will not describe NAT Traversal mechanisms but the protocol facilitates flexible NAT Traversal techniques, such as those based on ICE [RFC5245], considering that the tracker node may provide NAT traversal services, as a STUN-like tracker. Being a STUN-like tracker, it can discover the reflexive candidate addresses of a peer and make them available in responses to requesting peers, a mechanism named PPSP-ICE in [I-D.li-ppsp-nat-traversal-02].

1.2.3. Content Information Metadata

Multimedia contents may consist of several media components (for example, audio, video, and text), each of which might have different characteristics.

The representations of a media content correspond to encoded alternative of the same media component, varying from other representations by bitrate, resolution, number of channels, or other characteristics. Each representation consists of one or multiple segments. Segments are the media stream chunks in temporal sequence.

These characteristics may be described in a Media Presentation Description (MPD). Examples of MPD for on-demand and Live programs are illustrated in Appendix B. It is envisioned that the content information metadata used in PPSP may align with the MPD format of ISO/IEC 23009-1 [ISO.IEC.23009-1].

1.2.4. Authentication, Confidentiality, Integrity

Channel-oriented security should be used in the communication between peers and tracker, such as the Transport Layer Security (TLS) to provide privacy and data integrity. HTTP/1.1 over TLS (HTTPS) [RFC2818] is the preferred approach for preventing disclosure of peer critical information via the communication channel.

Due to the transactional nature of the communication between peers and tracker a method for adding authentication and data security services via replaceable mechanisms should be employed. One such method is the OAuth 2.0 Authorization [I-D.ietf-oauth-v2] with bearer token, providing the peer with the information required to successfully utilize the access token to make protected requests to the tracker [I-D.ietf-oauth-v2-bearer].

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This draft uses terms defined in [I-D.ietf-ppsp-problem-statement] and in [I-D.xiao-ppsp-reload-distributed-tracker].

Absolute Time: Absolute time is expressed as ISO 8601 [ISO.8601.2004] timestamps, using zero UTC offset (GMT). Fractions of a second may be indicated. Example for December 25, 2010 at 14h56 and 20.25 seconds: basic format 20101225T145620.25Z or extended format 2010-12-25T14:56:20.25Z.

Adaptive Streaming: Multiple alternate representations (different qualities and bitrates) of the same media content co-exist for the same streaming session; each alternate representation corresponds to a different media quality level; peers can choose among the alternate representations for decode and playback.

Base Layer: The playable representation level in Scalable Video Coding (SVC) required by all upper level Enhancements Layers for proper decoding of the video.

Chunk: A chunk is a generic term used whenever no ambiguity is raised, to refer to a segment or a subsegment of partitioned streaming media.

Complementary Representation: Representation in a set of representations which have inter-representation dependencies and which when combined result in a single representation for decoding

and presentation.

Connection Tracker: The tracker node to which the PPSP peer will connect when it wants to get registered and join the PPSP system.

Continuous media: Media with an inherent notion of time, for example, speech, audio, video, timed text or timed metadata.

Enhancement Layer: Enhancement differential quality level (complementary representation) in Scalable Video Coding (SVC) used to produce a higher quality, higher definition video in terms of space (i.e., image resolution), time (i.e., frame rate) or Signal-to-Noise Ratio (SNR) when combined with the playable Base Layer [ITU-T.H.264].

Join Time: Join time is the absolute time when a peer registers on a tracker. This value is recorded by the tracker and is used to calculate Online Time.

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.

Media Component: An encoded version of one individual media type such as audio, video or timed text with specific attributes, e.g., bandwidth, language, or resolution.

Media Presentation Description (MPD): Formalized description for a media presentation, i.e., describes the structure of the media, namely, the representations, the codecs used, the segments (chunks), and the corresponding addressing scheme.

Method: The method is the primary function that a request from a peer is meant to invoke on a tracker. The method is carried in the request message itself.

Online Time: Online Time shows how long the peer has been in the P2P streaming system since it joins. This value indicates the stability of a peer, and it is calculated by tracker when necessary.

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

Peer-ID: Unique identifier for the peer. The Peer-ID and any required security certificates are obtained from an offline enrollment server.

Peer-Peer Messages (i.e., Peer Protocol): The Peer Protocol messages

enable each peer to exchange content availability with other peers and request other peers for content.

PPSP: The abbreviation of Peer-to-Peer Streaming Protocols. PPSP protocols refer to the key signaling protocols among various P2P streaming system components, including the tracker and peers.

Representation: Structured collection of one or more media components.

Request: A message sent from a peer to a tracker, for the purpose of invoking a particular operation.

Response: A message sent from a tracker to a peer, for indicating the status of a request sent from the peer to the tracker.

Scalable Streaming: With Multiple Description Coding (MDC), multiple additive descriptions (that can be independently played-out) to refine the quality of the video when combined together. With Scalable Video Coding (SVC), nested dependent enhancement layers (hierarchical levels of quality), refine the quality of lower layers, from the lowest level (the playable Base Layer). With Multiple View Coding (MVC), multiple views allow the video to be played in 3D when the views are combined together.

Segment: A segment is a resource that can be identified, by an ID or an HTTP-URL and possibly a byte-range, and is included in the MPD. The segment is a basic unit of partitioned streaming media, which is used by a peer for the purpose of storage, advertisement and exchange among peers.

Subsegment: Smallest unit within segments which may be indexed at the segment level.

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

Swarm-ID: Unique identifier for a swarm. It is used to describe a specific resource shared among peers.

Tracker: A tracker refers to a centralized logical directory service used to communicate with PPSP Peers for content registration and location, which maintains the lists of PPSP peers storing segments for a specific live content channel or streaming media and answers queries from PPSP peers.

Tracker-Peer Messages (i.e., Tracker Protocol): The Tracker Protocol messages provide communication between peers and trackers, by which

peers provide content availability, report streaming status and request peer lists from trackers.

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

3. Architectural and Functional View

The PPSP Tracker Protocol architecture uses a two-layer approach i.e., a PPSP-TP messaging layer and a PPSP-TP request/response layer.

The PPSP-TP messaging layer deals with the underlying transport protocol and the asynchronous nature of the interactions between tracker and peers.

The PPSP-TP request/response layer deals with the interactions between tracker and peers using Method and Response codes (see Figure 2).

The transport layer is responsible for the actual transmission of requests and responses over network transports, including the determination of the connection to use for a request or response when using a connection-oriented transport like TCP, or TLS [RFC5246] over it.

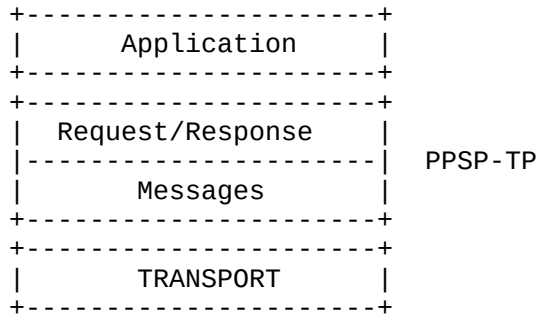


Figure 2: Abstract layering of PPSP-TP

The functional entities involved in the PPSP Tracker Protocol are trackers and peers (which may support different capabilities).

Peers correspond to devices that actually participate in sharing a media content and are organized in (various) swarms corresponding each swarm to the group of peers streaming that content at any given time. Each peer stores chunks of the media content, called segments (or subsegments), and contacts the tracker to advertise which information it has available. When a peer wishes to obtain information about the swarm, it contacts the tracker to find other

peers participating in that specific swarm.

The tracker is a logical entity that maintains the lists of peers storing chunks for a specific Live media channel or media streaming content, answers queries from peers and collects information on the activity of peers. While a tracker may have an underlying implementation consisting of more than one physical node, logically the tracker can most simply be thought of as a single element, and in this document, it will be treated as a single logical entity.

The Tracker Protocol is not used to exchange actual content data (either VoD or Live streaming) with peers, but information about which peers can provide which pieces of content.

When a peer wants to receive streaming of a selected content:

1. Peer connects to a local connection tracker and joins a swarm.
2. Peer acquires a list of peers from the connection tracker.
3. Peer exchanges its content availability with the peers on the obtained peer list.
4. Peer identifies the peers with desired content.
5. Peer requests for the content from the identified peers.

When a peer wants to share streaming of certain content with other peers:

1. Peer connects to the connection tracker.
2. Peer sends information to the connection tracker about the swarm it belongs to (joins), plus streaming status and/or content availability.

A P2P streaming process is summarized in Figure 3.

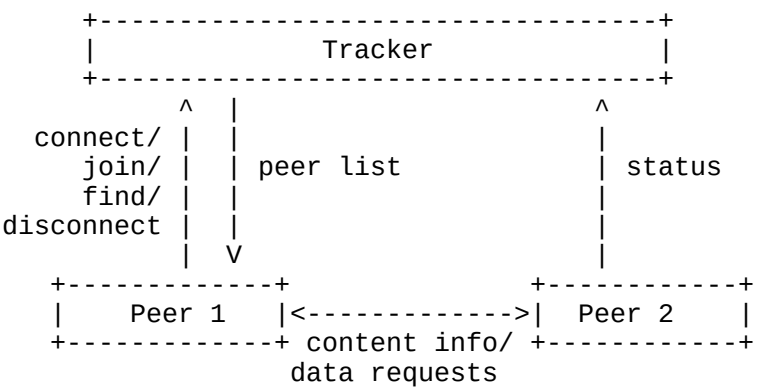


Figure 3: A PPSP streaming process

4. Messaging Model

The messaging model of PPSP-TP is based on the exchange of messages that follow the syntax and semantics of the current HTTP/1.1 specification [RFC2616]. The exchange of messages is envisioned to be performed over a stream-oriented reliable transport protocol, like TCP.

PPSP-TP is a text-based protocol, uses the UTF-8 character set [RFC3629] and the protocol messages are either requests from client peers to a tracker server, or responses from a tracker server to client peers.

5. Request/Response model

PPSP-TP request and response semantics are carried as entities (header and body) in PPSP-TP messages which correspond to either HTTP/1.1 request methods or HTTP/1.1 response codes, respectively.

Requests are sent, and responses returned to these requests. A single request generates a single response (neglecting fragmentation of messages in transport).

The response codes are consistent with HTTP/1.1 response codes, however, not all HTTP/1.1 response codes are used for the PPSP-TP (section 7).

The Request Messages of the protocol, are listed in Table 1:

PPSP Tracker Req. Messages
CONNECT
DISCONNECT
JOIN
FIND
STAT_REPORT

Table 1: Request Messages

CONNECT: This request message is used when a peer registers in the tracker. The tracker records the Peer-ID, connect-time (referenced to the absolute time), peer IP addresses and link status.

DISCONNECT: This request message is used when the peer intends to no longer participate in a specific swarm, or in all swarms. The

tracker deletes the corresponding activity records related to the peer (including its status and all content status for the corresponding swarms).

JOIN: This request message is used for a peer to notify the tracker that it wishes to participate in a swarm. The tracker records the content availability, i.e., adds the peer to the peers list for the swarm. On receiving a JOIN message, the tracker first checks the PeerMode type and then decides the next step (more details are referred in section 8.3).

FIND: This request message allows a peer to request to the tracker the peer list for a specific content representation or specific chunks of a media component in a swarm, before it can request the content from the peers. On receiving a FIND message, the tracker finds the peers, listed in content status of the specified swarm, that can satisfy the requesting peer's requirements, returning the list to the requesting peer. To create the peer list, the tracker may take peer status, capabilities and peers priority into consideration. Peer priority may be determined by network topology preference, operator policy preference, etc.

STAT_REPORT: This request message allows the exchange of statistic and status data between an active peer and a tracker to improve system performance. This request message is sent periodically to the tracker.

6. The Tracker State Machine

The state machine for the tracker is very simple, as shown in Figure 4.
Peer-ID registrations represent a dynamic piece of state maintained by the network.

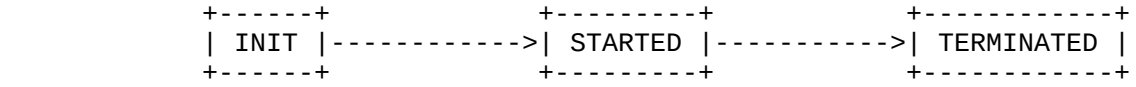


Figure 4: Tracker State Machine

When there are no peers registered in the tracker, the state machine is in the INIT state. When the first peer is registered with its Peer-ID, the state machine moves from INIT to STARTED.

As long as there is at least one active registration of a Peer-ID, the state machine remains in the STARTED state. When the last Peer-ID is removed, the state machine transitions to TERMINATED. From there, it immediately transitions back to the INIT state. Because of

that, the TERMINATED state here is transient.

In addition to this state machine, each registered Peer-ID is modeled with its own transaction state machine (Figure 5), instantiated per Peer-ID registered in the tracker, and deleted when it is removed. Unlike the state machine for the Peer-ID registration, which exists even when no Peer-IDs are registered, the per-Peer-ID transaction state machine is instantiated when the Peer-ID is registered, and deleted when the Peer-ID is removed.

This allows for an implementation optimization whereby the tracker can destroy the objects associated with the per-Peer-ID transaction state machine once it enters the TERMINATE state (Figure 5).

Figure 5: Per-Peer-ID Transaction State Machine

When a new Peer-ID is added, the per-Peer-ID state machine for it is

instantiated, and it moves into the PEER REGISTERED state. Because of that, the START state here is transient.

When the Peer-ID is no longer bound to a registration, the per-Peer-ID state machine moves to the TERMINATE state, and the state machine is destroyed.

During the life time of streaming activity of a peer, the per-Peer-ID transaction state machine progresses from one state to another in response to various events. The events that may potentially advance the state include:

- Reception of CONNECT, JOIN, FIND, DISCONNECT and STAT_REPORT messages, or
- Timeout events.

The state diagram in Figure 5 illustrates state changes, together with the causing events and resulting actions. Specific error conditions are not shown in the state diagram.

6.1. Normal Operation

On normal operation the process consists of the following steps:

1) When a CONNECT message is received from a peer, if successfully authenticated and validated, the tracker registers the Peer-ID and associated information (IP addresses), sends the response of successful registration to peer and starts the "init timer" waiting for a new message from the peer.

2) While PEER REGISTERED, when a JOIN message is received with valid swarm information, the tracker stops the "init timer", starts the "track timer" and sends the response of successful join to the peer. The response MAY contain the appropriate list of peers in the swarm, depending on PeerMode (section 8.3). A successful first JOIN starts the TRACKING state associated with the peer-ID for the requested swarm.

3) While TRACKING, a JOIN or FIND message received with valid swarm information from the peer resets the "track timer" and is responded with a successful condition, either for the JOIN to (an additional) swarm or for including the appropriate list of peers for the scope in the FIND request.

4) While TRACKING, a DISCONNECT(x) message received from the peer, containing a valid x=Swarm-ID resets the "track timer" and is responded with a successful condition. The tracker cleans the information associated with the participation of the Peer-ID in

the specified swarm(s).

In TRACKING state a STAT_REPORT message received from the peer resets the "track timer" and is responded with a successful condition. The STAT_REPORT message MAY contain information related with Swarm-IDs to which the peer is joined.

5) From either PEER REGISTERED or TRACKING states a DISCONNECT(x) message received from the peer, where x=nil, the tracker stops the "track timer", cleans the information associated with the participation of the Peer-ID in the the swarm(s) joined, responds with a successful condition, deletes the registration of the Peer-ID and transitions to TERMINATED state for that Peer-ID.

6) From TRACKING state a DISCONNECT(x) message received from the peer, where x=ALL or x=Swarm-ID is the last swarm, the tracker stops the "track timer", cleans the information associated with the participation of the Peer-ID in the the swarm(s) joined, responds with a successful condition and transitions to PEER REGISTERED state.

6.2. Error Conditions

Peers MUST NOT generate protocol elements that are invalid. However, several situations of a peer may lead to abnormal conditions in the interaction with the tracker. The situations may be related with peer malfunction or communications errors. The tracker reacts to the abnormal situations depending on its current state related to a peer-ID, as follows:

A) At the PEER REGISTERED state (while the "init timer" has not expired) receiving FIND, CONNECT or STAT_REPORT messages from the peer is considered an error condition. The tracker responds with error code 403 Forbidden (described in section 7), and resets the "init timer" one last time.

B) At the TRACKING state (while the "track timer" has not expired) receiving a CONNECT message from the peer is considered an error condition. The tracker responds with error code 403 Forbidden (described in section 7), and resets the "track timer".

NOTE: This situation may correspond to a malfunction at the peer or to malicious conditions. A preventive measure would be to reset the "track timer" one last time and if no valid message is received proceed to TERMINATE state for the Peer-ID by de-registering the peer and cleaning all peer information.

C) Without receiving messages from the peer, either from PEER

REGISTERED sate (init timer) or TRACKING state (track timer), on timeout the tracker cleans all the information associated with the Peer-ID in all swarms it was joined, deletes the registration, and transitions to TERMINATE state for that Peer-ID. The same action is taken if no valid message is received at the PEER REGISTERED state after the last "init timer" expires.

7. Protocol Specification

7.1. Messages Syntax

PPSP-TP messages use the generic message format of RFC 5322 [RFC5322] for transferring the payload of the message (Requests and Responses).

PPSP-TP messages consist of a start-line, one or more header fields, an empty line indicating the end of the header fields, and, when applicable, a message-body.

The start-line, each message-header line, and the empty line MUST be terminated by a carriage-return line-feed sequence (CRLF). Note that the empty line MUST be present even if the message-body is not.

The PPSP-TP message and header field syntax is identical to HTTP/1.1 [RFC2616].

A Request message is a standard HTTP/1.1 message starting with a Request-Line generated by the HTTP client peer. The Request-Line contains a method name, a Request-URI, and the protocol version separated by a single space (SP) character.

```
Request-Line =
    Method SP Request-URI SP HTTP-Version CRLF
```

A Request message example is the following:

```
<Method> /<Resource> HTTP/1.1
Host: <Host>
Content-Lenght: <ContentLenght>
Content-Type: <ContentType>
Authorization: <AuthToken>

[Request_Body]
```

The HTTP Method token and Request-URI (the Resource) identifies the resource upon which to apply the operation requested.

The Response message is also a standard HTTP/1.1 message starting with a Status-Line generated by the tracker. The Status-Line consists of the protocol version followed by a numeric Status-Code and its associated Reason-Phrase, with each element separated by a single SP character.

Status-Line =
HTTP-Version SP Status-Code SP Reason-Phrase CRLF

A Response message example is the following:

```
HTTP/1.1 <Status-Code> <Reason-Phrase>
Content-Lenght: <ContentLenght>
Content-Type: <ContentType>
Content-Encoding: <ContentCoding>

[Response_Body]
```

The Status-Code element is a 3-digit integer result code that indicates the outcome of an attempt to understand and satisfy a request.

The Reason-Phrase element is intended to give a short textual description of the Status-Code.

7.1.1. Header Fields

The header fields are identical to HTTP/1.1 header fields in both syntax and semantics.

Some header fields only make sense in requests or responses. If a header field appears in a message not matching its category (such as a request header field in a response), it MUST be ignored.

The Host request-header field in the request message follows the standard rules for the HTTP/1.1 Host header.

The Content-Type entity-header field MUST be used in requests and responses containing message-bodies to define the Internet media type of the message-body.

The Content-Encoding entity-header field MAY be used in response messages with "gzip" compression scheme [RFC2616] for faster transmission times and less network bandwidth usage.

The Content-Length entity-header field MUST be used in messages containing message-bodies to locate the end of each message in a stream.

The Authorization header field in the request message allows a peer to authenticate itself with a tracker, containing authentication information.

7.1.2. Methods

PPSP-TP uses HTTP/1.1 POST method token for all request messages.

7.1.3. Message Bodies

PPSP-TP requests MUST contain message-bodies.

PPSP-TP responses MAY include a message-body.

If the message-body has undergone any encoding such as compression, then this MUST be indicated by the Content-Encoding header field; otherwise, Content-Encoding MUST be omitted.

If applicable, the character set of the message body is indicated as part of the Content-Type header-field, and the default value for PPSP-TP messages is "UTF-8".

7.1.4. Message Response Codes

The response codes in PPSP-TP response messages are consistent with HTTP/1.1 response status-codes. However, not all HTTP/1.1 response status-codes are appropriate for PPSP-TP, and only those that are appropriate are given here. Other HTTP/1.1 response codes SHOULD NOT be used in PPSP-TP.

The class of the response is defined by the first digit of the Status-Code. The last two digits do not have any categorization role. For this reason, any response with a Status-Code between 200 and 299 is referred to as a "2xx response", and similarly to the other supported classes:

2xx: Success -- the action was successfully received, understood, and accepted;
4xx: Peer Error -- the request contains bad syntax or cannot be fulfilled at this tracker;
5xx: Tracker Error -- the tracker failed to fulfill an apparently valid request;

The valid response codes are the following (Status-Code Reason-Phrase):

200 OK -- The request has succeeded. The information returned with the response describes or contains the result of the action;

400 Bad Request -- The request could not be understood due to malformed syntax.

401 Unauthorized -- The request requires authentication.

403 Forbidden -- The tracker understood the request, but is refusing to fulfill it. The request SHOULD NOT be repeated.

404 Not Found -- This status is returned if the tracker did not find anything matching the Request-URI.

408 Request Timeout -- The peer did not produce a request within the time that the tracker was prepared to wait.

411 Length Required -- The tracker refuses to accept the request without a defined Content-Length. The peer MAY repeat the request if it adds a valid Content-Length header field containing the length of the message-body in the request message.

414 Request-URI Too Long -- The tracker is refusing to service the request because the Request-URI is longer than the tracker is willing to interpret. This rare condition is likely to occur when the tracker is under attack by a client attempting to exploit security holes.

500 Internal Server Error -- The tracker encountered an unexpected condition which prevented it from fulfilling the request.

503 Service Unavailable -- The tracker is currently unable to handle the request due to a temporary overloading or maintenance condition.

7.2. Request/Response Syntax and Format

The message-body for Requests and Responses requiring it, is encoded in XML.

The XML message-body MUST begin with an XML declaration line specifying the version of XML being used and indicating the character encoding, that SHOULD be "UTF-8". The root element MUST be PPSPTrackerProtocol.

The generic format of a Request is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
    <PPSPTrackerProtocol version="1.0">
        <Request></Request>
        <TransactionID></TransactionID>
        <PeerID></PeerID>
        <SwarmID></SwarmID>
        <PeerNum></PeerNum>
        <PeerMode></PeerMode>
        <PeerGroup></PeerGroup>
        <ContentGroup></ContentGroup>
        <StatisticsGroup></StatisticsGroup>
    </PPSPTrackerProtocol>
```

The generic format of a Response is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
    <PPSPTrackerProtocol version="1.0">
        <Response></Response>
        <TransactionID></TransactionID>
        <SwarmID></SwarmID>
        <PeerGroup></PeerGroup>
    </PPSPTrackerProtocol>
```

The Request element MUST be present in requests and corresponds to the request method type for the message.

The Response element MUST be present in responses and corresponds to the response method type of the message.

The element TransactionID MUST be present in requests to uniquely identify the transaction. Responses to completed transactions use the same TransactionID as the request they correspond to.

The version of PPSP-TP being used is indicated by the attribute @version of the root element.

All Request messages MUST contain a PeerID element to uniquely identify the peer (Peer-ID) in the network.

The PeerID information may be present on the following levels:

- On PPSPTrackerProtocol level in PPSPTrackerProtocol.PeerID element. For details refer to 7.2.1 Table 2.

- On PeerGroup level in PeerGroup.PeerInfo.PeerID element. For details refer to 7.2.1 Table 3.

The SwarmID element MUST be be present in JOIN, FIND and DISCONNECT requests. The SwarmID element MUST be present in JOIN and FIND responses. Details of usage in 8.2, 8.3 and 8.4.

The SwarmID information may be present on the following levels:

- On PPSPTrackerProtocol level in PPSPTrackerProtocol.SwarmID element. For details refer to 7.2.1 Table 2.

- On StatisticsGroup level in StatisticsGroup.Stat.SwarmID element. For details refer to 7.2.1 Table 5.

The PeerMode element MUST be present in JOIN requests. Details of usage in 8.3.

The PeerMode information may be present on the following levels:

- On PPSPTrackerProtocol level in PPSPTrackerProtocol.PeerMode element. For details refer to 7.2.1 Table 2.

- On PeerGroup level in PeerGroup.PeerMode element. For details refer to 7.2.1 Table 5.

The PeerNum element MUST be present in JOIN requests and MAY contain the attribute @abilityNAT to inform the tracker on the preferred type of peers, in what concerns their NAT traversal situation, to be returned in a peer list. Details of usage in 8.2, 8.3 and 8.4.

The PeerGroup element MUST be present in CONNECT requests and responses and MAY be present in responses to JOIN and FIND requests if the corresponding response returns information about peers. Details of usage in 8.1, 8.3 and 8.4.

The ContentGroup element MAY be present in requests referencing content, i.e., JOIN and FIND, if the request includes a content scope. Details of usage in 8.3 and 8.4.

The StatisticsGroup element MAY be present in STAT_REPORT requests. Details of usage in 8.5.

The semantics of the attributes and elements within a PPSPTrackerProtocol root element is described in subsection 7.2.1.

Request and Response processing is provided in section 8 for each message.

The XML-syntax of the of PPSP-TP XML elements for Requests and Responses is provided in the XML-Schema of Appendix A.

7.2.1. Semantics of PPSPTrackerProtocol elements

The semantics of PPSPTrackerProtocol elements and attributes are described in the following tables.

Element Name or Attribute Name	Use	Description
PPSPTrackerProtocol	1	The root element.
@version	M	Provides the version of PPSP-TP.
Request	0...1	Provides the request method and MUST be present in Request.
Response	0...1	Provides the response method and MUST be present in Response.
PeerID	0...1	Peer Identification. MUST be present in Request.
SwarmID	0...1	Swarm Identification. Details in 8.2/8.3/8.4/8.5.
PeerMode	0...1	Mode of Peer participation in a swarm, which can be "LEECH" or "SEED". Details in 8.3/8.4.
PeerNUM	0...1	Maximum peers to be received in with capabilities indicated.
@abilityNAT	CM	Type of NAT traversal peers, as "NoNAT", "STUN","TURN" or "PROXY"
@concurrentLinks	CM	Concurrent connectivity level of peers, "HIGH", "LOW" or "NORMAL"
@onlineTime	CM	Availability or online duration of peers, "HIGH" or "NORNMAL"
@uploadBWlevel	CM	Upload bandwidth capability of peers, "HIGH" or "NORMAL"
PeerGroup	0...1	Provides information on peers. More details in Table 3
ContentGroup	0...1	Provides information on content. More details in Table 4
StatisticsGroup	0...1	Provides statistic data of peer and content. Details in Table 5

Legend:
Use for attributes: M=Mandatory, OP=Optional, CM=Conditionally Mandatory
Use for elements: minOccurs...maxOccurs (N=unbounded)
Elements are represented by their name (case-sensitive)
Attribute names (case-sensitive) are preceded with an @

Table 2: Semantics of PPSPTrackerProtocol.

Element Name or Attribute Name	Use	Description
PeerGroup	0...1	Contains description of peers.
PeerInfo	1...N	Provides information on a peer.
PeerID	0...1	Peer Identification. MAY be present in JOIN and FIND responses. Details in 8.3/8.4.
PeerMode	0...1	Mode of Peer participation in a swarm, which can be "LEECH" or "SEED". MAY be present in JOIN and FIND responses. Details in 8.3/8.4.
PeerAddress	1...N	IP Address information.
@addrType	M	Type of IP address, which can be "ipv4" or "ipv6"
@priority	CM	The priority of this interface. Used for NAT traversal.
@type	CM	Describes the address for NAT traversal, which can be "HOST" "REFLEXIVE" or "PROXY".
@connection	OP	Access type ("3G", "ADSL", etc.)
@asn	OP	Autonomous System number.
@ip	M	IP address value.
@port	M	IP service port value.

Legend:
Use for attributes: M=Mandatory, OP=Optional,
CM=Conditionally Mandatory
Use for elements: minOccurs...maxOccurs (N=unbounded)
Elements are represented by their name (case-sensitive)
Attribute names (case-sensitive) are preceded with an @

Table 3: Semantics of PeerGroup.

If STUN-like functions are enabled in the tracker and a PPSP-ICE method is used, as described in [I-D.li-ppsp-nat-traversal-02], the attributes @type and @priority MUST be returned with the transport address candidates in responses to CONNECT, JOIN or FIND requests.

The @asn attribute MAY be used to inform about the network location, in terms of Autonomous System, for each of the active public network interfaces of the peer.

The @connection attribute is informative on the type of access network of the respective interface.

Element Name or Attribute Name	Use	Description
ContentGroup	0...1	Provides information on content.
Representation	1...N	Describes a component of content.
@id	M	Unique identifier for this Representation.
SegmentInfo	1	Provides segment information.
@startIndex	M	The index of the first media segment in the request scope for this Representation.
@endIndex	OP	The index of the last media segment in the request scope for this Representation.

Legend:
Use for attributes: M=Mandatory, OP=Optional, CM=Conditionally Mandatory
Use for elements: minOccurs...maxOccurs (N=unbounded)
Elements are represented by their name (case-sensitive)
Attribute names (case-sensitive) are preceded with an @

Table 4: Semantics of ContentGroup.

The Representation element describes a component of a content identified by its attribute @id in the MPD. This element MAY be present for each component desired in the scope of the JOIN or FIND request. The scope of each Representation is indicated in the SegmentInfo element by the attribute @startIndex and, optionally, @endIndex.

The peer may use this information in JOIN or FIND requests, for example, to join a swarm starting from a specific point (as is the case of a live program, by specifying the adequate @startIndex) and/or find adequate peers in the swarm for that content scope.

An example of on-demand usage is the case of an end-user that previously watched a content with a certain audio language, then interrupted for a while (having disconnected) and later continued by re-joining from that point onwards but selecting a different available audio language. In this case the JOIN request would specify the required Representations and the @startIndex for each, i.e., all the adequate video components and the selected audio component. An example is illustrated in subsection 8.3.

Element Name or Attribute Name	Use	Description
StatisticsGroup	0...1	Provides statistic data on peer and content.
Stat	1...N	Groups statistics property data.
@property	M	The property to be reported. Property values in Table 6.
SwarmID	0...1	Swarm Identification.
UploadedBytes	0...1	Bytes sent to swarm.
DownloadedBytes	0...1	Bytes received from swarm.
AvailBandwidth	0...1	Upstream Bandwidth available.
Representation	0...N	Describes a component of content.
@id	CM	Unique identifier for this Representation.
SegmentInfo	1...N	Provides segment information by segment range. The chunkmap can be encoded in Base64 [RFC4648].
@startIndex	CM	The index of the first media segment in the chunkmap report for this Representation.
@endIndex	CM	The index of the last media segment in the chunkmap report for this Representation.
@chunkmapSize	CM	Size of chunkmap reported.

Legend:
Use for attributes: M=Mandatory, OP=Optional, CM=Conditionally Mandatory
Use for elements: minOccurs...maxOccurs (N=unbounded)
Elements are represented by their name (case-sensitive)
Attribute names (case-sensitive) are preceded with an @

Table 5: Semantics of StatisticsGroup.

The Stat element is used to describe several properties relevant to the P2P network. These properties can be related with stream statistics, peer status information and content data information, like chunkmaps. Each Stat element will correspond to a @property type and several Stat blocks can be reported in a single STAT_REPORT message, corresponding to some or all the swarms the peer is actively involved.

Other properties may be defined, related, for example, with incentives and reputation mechanisms, like peer online time, or connectivity conditions, like physical link status, etc.

For that purpose, the Stat element may be extended to provide additional scheme specific information for new @property groups, new elements and new attributes.

@property	Description
StreamStatistics	Stream statistic values per SwarmID
ContentMap	Reports map of chunks the peer has per Representation of the content

Table 6: StatisticsGroup default Stat @property values.

An example of a STAT_REPORT for multiple properties is illustrated in subsection 8.5.

7.2.2. Request element in request Messages

Table 7 defines the valid string representations for the requests. These values MUST be treated as case-sensitive.

XML Request Methods String Values
CONNECT
DISCONNECT
JOIN
FIND
STAT_REPORT

Table 7: Valid Strings for Request element of requests.

7.2.3. Response element in response Messages

Table 8 defines the valid string representations for Response messages that require message-body. These values MUST be treated as case-sensitive.

Response messages not requiring message-body only use the standard HTTP/1.1 Status-Code and Reason-Phrase (appended, if appropriate, with detail phrase, as described in section 8.6).

XML Response Method String Values	HTTP Status-Code and Reason-Phrase
SUCCESSFUL	200 OK
AUTHENTICATION REQUIRED	401 Unauthorized

Table 8: Valid Strings for Response element of responses.

SUCCESSFUL: indicates that the request has been processed properly and the desired operation has completed. The body of the response message includes the requested information and MUST include the same TransactionID of the corresponding request.

CONNECT: returns information about the successful registration of the peer.

DISCONNECT and STAT_REPORT: confirms the success of the requested operation.

JOIN and FIND: MAY return the list of peers meeting the desired criteria.

AUTHENTICATION REQUIRED: Authentication is required for the peer to make the request.

8. Request/Response Processing

When a PPSP-TP message is received some basic processing is performed, regardless of the message type.

Upon reception, a message is examined to ensure that it is properly formed. The receiver MUST check that the HTTP message itself is properly formed, and if not, appropriate standard HTTP errors MUST be generated. The receiver must also verify that the XML body is properly formed. In case of error due to malformed messages appropriate responses MUST be returned, as described in 8.6.

8.1. CONNECT Request

This method is used when a peer registers to the system. The tracker records the Peer-ID, connect-time, IP addresses and link status.

The peer MUST properly form the XML message-body, set the Request method to CONNECT, generate and set the TransactionID, and set the PeerID with the identifier of the peer. The peer SHOULD also include

the IP addresses of its network interfaces in the CONNECT message.

An example of the message-body of a CONNECT Request is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol version="1.0">
  <Request>CONNECT</Request>
  <PeerID>656164657221</PeerID>
  <TransactionID>12345</TransactionID>
  <PeerGroup>
    <PeerInfo>
      <PeerAddress addrType="ipv4" ip="192.0.2.1" port="80"
                   priority="1" />
      <PeerAddress addrType="ipv6" ip="2001:db8::1" port="80"
                   priority="2"
                   type="HOST"
                   connection="3G" />
    </PeerInfo>
  </PeerGroup>
</PPSPTrackerProtocol>
```

When receiving a well-formed CONNECT Request message, the tracker will first processes the peer authentication information (provided as Authorization scheme and token in the HTTP message) to check whether it is valid and that it can connect to the service, and then proceed to register the peer in the service. In case of success a Response message with a corresponding response value of SUCCESSFULL will be generated.

The element PeerInfo MAY contain multiple PeerAddress child elements with attributes @addrType, @ip, and @port, and optionally @priority and @type (if PPSP-ICE NAT traversal techniques are used) corresponding to each of the network interfaces of the peer.

If STUN-like function is enabled in the tracker, the response MAY include the peer reflexive address [I-D.li-ppsp-nat-traversal-02].

The response MUST have the same TransactionID value as the request.

An example of a Response message for the CONNECT Request is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol version="1.0">
  <Response>SUCCESSFUL</Response>
  <TransactionID>12345</TransactionID>
  <PeerGroup>
    <PeerInfo>
      <PeerAddress addrType="ipv4" ip="198.51.100.1" port="80"
                   priority="1"
                   type="REFLEXIVE"
                   connection="ADSL"
                   asn="64496" />
    </PeerInfo>
  </PeerGroup>
</PPSPTrackerProtocol>
```

The Response MUST include a PeerGroup with PeerInfo data that includes the peer public IP address. If STUN-like function is enabled in the tracker, the PeerAddress includes the attribute @type with a value of REFLEXIVE, corresponding to the transport address "candidate" of the peer.

The tracker MAY also include the attribute @asn with network location information of the transport address, corresponding to the Autonomous System Number of the access network provider.

8.2. DISCONNECT Request

This method is used when the peer intends to leave a specific swarm, or the system, and no longer participate.

The tracker SHOULD delete the corresponding activity records related with the peer in the corresponding swarms (including its status and all content status).

The peer MUST properly form the XML message-body, set the Request method to DISCONNECT, set the PeerID with the identifier of the peer, randomly generate and set the TransactionID and include the SwarmID information.

The SwarmID value MUST be either a specific Swarm-ID the peer had previously joined, the value "ALL" to designate all joined swarms, or the value "nil" to completely disconnect from the system.

An example of the message-body of a DISCONNECT Request is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol version="1.0">
  <Request>DISCONNECT</Request>
  <PeerID>656164657221</PeerID>
  <SwarmID>ALL</SwarmID>
  <TransactionID>12345</TransactionID>
</PPSPTrackerProtocol>
```

In case of success a Response message with a corresponding response value of SUCCESSFULL will be generated. The response MUST have the same TransactionID value as the request.

Upon receiving a DISCONNECT message, the tracker cleans the information associated with the participation of the Peer-ID in the specified swarm (or in all swarms).

An example of a Response message for the DISCONNECT Request is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol version="1.0">
  <Response>SUCCESSFUL</Response>
  <TransactionID>12345</TransactionID>
</PPSPTrackerProtocol>
```

If the scope of SwarmID in the DISCONNECT request is "nil" the tracker will also delete the registration of the Peer-ID.

8.3. JOIN Request

This method is used for peers to notify the tracker that they wish to participate in a particular swarm.

The JOIN message is used when the peer has none or just some chunks (LEECH), or has all the chunks (SEED) of a content. The JOIN is used for both on-demand or Live streaming modes.

The peer MUST properly form the XML message-body, set the Request method to JOIN, set the PeerID with the identifier of the peer, set the SwarmID with the identifier of the swarm it is interested in, and randomly generate and set the TransactionID.

An example of the message-body of a JOIN Request is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol version="1.0">
  <Request>JOIN</Request>
  <PeerID>656164657221</PeerID>
  <SwarmID>1111</SwarmID>
  <TransactionID>12345</TransactionID>
  <PeerNum abilityNAT="STUN"
           concurrentLinks="HIGH"
           onlineTime="NORMAL"
           uploadBWlevel="NORMAL">5</PeerNum>
  <PeerMode>LEECH</PeerMode>
  <ContentGroup>
    <Representation id="tag0">
      <SegmentInfo startIndex="20" />
    </Representation>
    <Representation id="tag6">
      <SegmentInfo startIndex="20" />
    </Representation>
  </ContentGroup>
</PPSPTrackerProtocol>
```

The JOIN request MAY include a PeerNum element to indicate to the tracker the number of peers to be returned in a list corresponding to the indicated properties, being @abilityNAT for NAT traversal (considering that PPSP-ICE NAT traversal techniques may be used), and optionally @concurrentLinks, @onlineTime and @uploadBWlevel for the preferred capabilities.

The PeerMode element SHOULD be set to the type of participation of the peer in the swarm (SEED or LEECH).

In the case of a JOIN to a specific point in a stream the request SHOULD include a ContentGroup to specify the joining point in terms of content Representations. The above example of a JOIN request would be for the case of an end-user that previously watched a content with a certain audio language, then interrupted for a while (having disconnected) and later continued by re-joining from that point onwards but selecting a different available audio language (Representation with @id="tag6" in the MPD of Appendix B).

When receiving a well-formed JOIN Request the tracker processes the information to check if it is valid and if the peer can join the swarm of interest. In case of success a response message with a Response value of SUCCESSFULL will be generated and the tracker enters the peer information into the corresponding swarm activity.

In case the PeerMode is SEED, the tracker just responds with a SUCCESSFUL response and enters the peer information into the corresponding swarm activity.

In case the PeerMode is LEECH the tracker will search and select an appropriate list of peers satisfying the conditions requested. The peer list MUST contain the Peer-IDs and the corresponding IP Addresses. To create the peer list, the tracker may take peer status and network location information into consideration, to express network topology preferences or Operators' policy preferences, with regard to the possibility of connecting with other IETF efforts such as ALTO [I.D.ietf-alto-protocol].

The response MUST have the same TransactionID value as the request.

An example of a Response message for the JOIN Request is:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol version="1.0">
  <Response>SUCCESSFUL</Response>
  <TransactionID>12345</TransactionID>
  <PeerGroup>
    <PeerInfo>
      <PeerID>956264622298</PeerID>
      <PeerAddress addrType="ipv4" ip="198.51.100.22" port="80"
                   asn="64496" />
    </PeerInfo>
    <PeerInfo>
      <PeerID>3332001256741</PeerID>
      <PeerAddress addrType="ipv4" ip="198.51.100.201" port="80"
                   asn="64496" />
    </PeerInfo>
  </PeerGroup>
</PPSPTrackerProtocol>
```

The Response MUST include a PeerGroup with PeerInfo data that includes the public IP address of the selected active peers in the swarm.

The tracker MAY also include the attribute @asn with network location information of the transport addresses of the peers, corresponding to the Autonomous System Numbers of the access network provider of each peer in the list.

8.4. FIND Request

This method allows peers to request to the tracker, whenever needed and after being joined to a swarm, a new peer list for the swarm or

for specific scope of chunks of a media content Representation of that swarm.

The peer MUST properly form the XML message-body, set the Request method to FIND, set the PeerID with the identifier of the peer, set the SwarmID with the identifier of the swarm the peer is interested, and optionally, in order to find peers having the specific chunks, include information about the content.

The peer MUST also generate and set the TransactionID for the request.

An example of the message-body of a FIND Request is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol version="1.0">
  <Request>FIND</Request>
  <PeerID>656164657221</PeerID>
  <SwarmID>1111</SwarmID>
  <TransactionID>12345</TransactionID>
  <PeerNum abilityNAT="STUN"
           concurrentLinks="HIGH"
           onlineTime="NORMAL"
           uploadBWlevel="NORMAL">5</PeerNum>
  <ContentGroup>
    <Representation id="tag4">
      <SegmentInfo startIndex="110" endIndex="150" />
    </Representation>
  </ContentGroup>
</PPSPTrackerProtocol>
```

The FIND request MAY include a PeerNum element to indicate to the tracker the number of peers to be returned in a list corresponding to the indicated properties, being @abilityNAT for NAT traversal (considering that PPSP-ICE NAT traversal techniques may be used), and optionally @concurrentLinks, @onlineTime and @uploadBWlevel for the preferred capabilities.

In the case of a FIND with a specific scope of a stream content the request SHOULD include a ContentGroup to specify the content Representations segment range of interest.

When receiving a well-formed FIND Request the tracker processes the information to check if it is valid. In case of success a response message with a Response value of SUCCESSFULL will be generated and the tracker will include the appropriate list of peers satisfying the conditions requested. The peer list returned MUST contain the Peer-IDs and the corresponding IP Addresses.

The tracker may take peer status and network location information into consideration when selecting the peer list to return, to express network topology preferences or Operators' policy preferences, with regard to the possibility of connecting with other IETF efforts such as ALTO [I.D.ietf-alto-protocol].

The response MUST have the same TransactionID value as the request.

An example of a Response message for the FIND Request is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol version="1.0">
  <Response>SUCCESSFUL</Response>
  <TransactionID>12345</TransactionID>
  <PeerGroup>
    <PeerInfo>
      <PeerID>956264622298</PeerID>
      <PeerAddress addrType="ipv4" ip="198.51.100.22" port="80"
                   asn="64496" />
    </PeerInfo>
    <PeerInfo>
      <PeerID>3332001256741</PeerID>
      <PeerAddress addrType="ipv4" ip="198.51.100.201" port="80"
                   asn="64496" />
    </PeerInfo>
  </PeerGroup>
</PPSPTrackerProtocol>
```

The Response MUST include a PeerGroup with PeerInfo data that includes the public IP address of the selected active peers in the swarm.

The tracker MAY also include the attribute @asn with network location information of the transport addresses of the peers, corresponding to the Autonomous System Numbers of the access network provider of each peer in the list.

8.5. STAT_REPORT Request

This method allows the exchange of statistic and status data between peers and trackers to improve system performance. The method is initiated by the peer, periodically while active.

The peer MUST properly form the XML message-body, set the Request method to STAT_REPORT, set the PeerID with the identifier of the peer, and generate and set the TransactionID.

The report MAY include a StatisticsGroup containing multiple Stat elements describing several properties relevant to the P2P network. These properties can be related with stream statistics, peer status information and content data information, like chunkmaps.

Other properties may be defined, related for example, with incentives and reputation mechanisms.

In case no StatisticsGroup is included, the STAT_REPORT may be used as a "keep-alive" message, to prevent the Tracker from de-registering the peer when timer expired.

An example of the message-body of a STAT_REPORT Request is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol version="1.0">
  <Request>STAT_REPORT</Request>
  <PeerID>656164657221</PeerID>
  <TransactionID>12345</TransactionID>
  <StatisticsGroup>
    <Stat property="StreamStatistics">
      <SwarmID>1111</SwarmID>
      <UploadedBytes>512</UploadedBytes>
      <DownloadedBytes>768</DownloadedBytes>
      <AvailBandwidth>1024000</AvailBandwidth>
    </Stat>
    <Stat property="StreamStatistics">
      <SwarmID>2222</SwarmID>
      <UploadedBytes>1024</UploadedBytes>
      <DownloadedBytes>2048</DownloadedBytes>
      <AvailBandwidth>512000</AvailBandwidth>
    </Stat>
    <Stat property="ContentMap">
      <SwarmID>1111</SwarmID>
      <Representation id="tag0">
        <SegmentInfo startIndex="0" endIndex="24"
                     chunkmapSize="25">
          A/8D/wP/A/8D/wP/A/8D/wP/A/8D/wP/....
        </SegmentInfo>
      </Representation>
      <Representation id="tag1">
        <SegmentInfo startIndex="0" endIndex="14"
                     chunkmapSize="15">
          A/8D/wP/A/8D/wP/A/8D/wP/A/8D/wP/....
        </SegmentInfo>
        <SegmentInfo startIndex="20" endIndex="24"
                     chunkmapSize="5">
```

```
            A/8D/wP/A/8D/wP/A/8D/wP/A/8D/wP/....
          </SegmentInfo>
        </Representation>
      </Stat>
      <Stat property="ContentMap">
        <SwarmID>2222</SwarmID>
        <Representation id="tag5">
          <SegmentInfo startIndex="0" endIndex="4"
                     chunkmapSize="5">
            A/8D/wP/A/8D/wP/A/8D/wP/A/8D/wP/....
          </SegmentInfo>
        </Representation>
        <Representation id="tag6">
          <SegmentInfo startIndex="0" endIndex="4"
                      chunkmapSize="5">
            A/8D/wP/A/8D/wP/A/8D/wP/A/8D/wP/....
          </SegmentInfo>
        </Representation>
      </Stat>
    </StatisticsGroup>
  </PPSPTrackerProtocol>
```

If the request is valid the tracker process the received information for future use, and generates a response message with a Response value of SUCCESSFULL.

The response MUST have the same TransactionID value as the request.

An example of a Response message for the START_REPORT Request is the following:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <PPSPTrackerProtocol version="1.0">
    <Response>SUCCESSFUL</Response>
    <TransactionID>12345</TransactionID>
  </PPSPTrackerProtocol>
```

8.6. Error and Recovery conditions

If the peer fails to read the tracker response, the same Request with identical content, including the same TransactionID, SHOULD be repeated, if the condition is transient.

The TransactionID on a Request can be reused if and only if all of the content is identical, including eventual Date/Time information. Details of the retry process (including time intervals to pause, number of retries to attempt, and timeouts for retrying) are implementation dependent.

The tracker SHOULD be prepared to receive a Request with a repeated TransactionID.

Error situations resulting from the Normal Operation or from abnormal conditions (section 6.2) MUST be responded with the adequate response codes, as described here:

If the message is found to be incorrectly formed, the receiver MUST respond with a 400 (Bad Request) response with an empty message-body. The Reason-Phrase SHOULD identify the syntax problem in more detail, for example, "Missing Content-Type header field".

If the version number of the protocol is for a version the receiver does not supports, the receiver MUST respond with a 400 (Bad Request) with an empty message-body. Additional information SHOULD be provided in the Reason-Phrase, for example, "PPSP Version #.#".

If the length of the message does not matches the Content-Length specified in the message header, or the message is received without a defined Content-Length, the receiver MUST respond with a 411 (Length Required) response with an empty message-body.

If the Request-URI in a Request message is longer than the tracker is willing to interpret, the tracker MUST respond with a 414 (Request-URI Too Long) response with an empty message-body.

In the PEER REGISTERED and TRACKING states of the tracker, certain requests are not allowed (section 6.2). The tracker MUST respond with a 403 (Forbidden) response with an empty message-body. The Reason-Phrase SHOULD identify the error condition in more detail, for example, "Already Connected".

If the tracker is unable to process a Request message due to unexpected condition, it SHOULD respond with a 500 (Internal Server Error) response with an empty message-body.

If the tracker is unable to process a Request message for being in an overloaded state, it SHOULD respond with a 503 (Service Unavailable) response with an empty message-body.

9. Security Considerations

P2P streaming systems are subject to attacks by malicious/unfriendly peers/trackers that may eavesdrop on signaling, forge/deny information/knowledge about streaming content and/or its availability, impersonating to be another valid participant, or launch DoS attacks to a chosen victim.

No security system can guarantees complete security in an open P2P streaming system where participants may be malicious or uncooperative. The goal of security considerations described here is to provide sufficient protection for maintaining some security properties during the tracker-peer communication even in the face of a large number of malicious peers and/or eventual distrustful trackers (under the distributed tracker deployment scenario).

Since the protocol uses HTTP to transfer signaling most of the same security considerations described in RFC 2616 also apply [RFC2616].

9.1. Authentication between Tracker and Peers

To protect the PPSP-TP signaling from attackers pretending to be valid peers (or peers other than themselves) all messages received in the tracker are required to be received from authorized peers.

For that purpose a peer must enroll in the system via a centralized enrollment server. The enrollment server is expected to provide a proper Peer-ID for the peer and information about the authentication mechanisms. The specification of the enrollment method and the provision of identifiers and authentication tokens is out of scope of this specification.

A Channel-oriented security mechanism should be used in the communication between peers and tracker, such as the Transport Layer Security (TLS) to provide privacy and data integrity.

Due to the transactional nature of the communication between peers and tracker the method for adding authentication and data security services can be the OAuth 2.0 Authorization [I-D.ietf-oauth-v2] with bearer token, which provides the peer with the information required to successfully utilize an access token to make protected requests to the tracker [I-D.ietf-oauth-v2-bearer].

9.2. Content Integrity protection against polluting peers/trackers

Malicious peers may declaim ownership of popular content to the tracker but try to serve polluted (i.e., decoy content or even virus/trojan infected contents) to other peers.

This kind of pollution can be detected by incorporating integrity verification schemes for published shared contents. As content chunks are transferred independently and concurrently, a correspondent chunk-level integrity verification MUST be used, checked with signed fingerprints received from authentic origin.

9.3. Residual attacks and mitigation

To mitigate the impact of sybil attackers, impersonating a large number of valid participants by repeatedly acquiring different peer identities, the enrollment server SHOULD carefully regulate the rate of peer/tracker admission.

There is no guarantee that peers honestly report their status to the tracker, or serve authentic content to other peers as they claim to the tracker. It is expected that a global trust mechanism, where the credit of each peer is accumulated from evaluations for previous transactions, may be taken into account by other peers when selecting partners for future transactions, helping to mitigate the impact of such malicious behaviors. A globally trusted tracker MAY also take part of the trust mechanism by collecting evaluations, computing credit values and providing them to joining peers.

9.4. Pro-incentive parameter trustfulness

Property types for STAT_REPORT messages may consider pro-incentive parameters, which can enable the tracker to improve the performance of the whole P2P streaming system.

Trustworthiness of these pro-incentive parameters is critical to the effectiveness of the incentive mechanisms. For example, ChunkMaps are essential, and need to be accurate. The P2P system should be designed in a way such that a peer will have the incentive to report truthfully its ChunkMaps (otherwise it may penalize itself, as in the case of under-reporting addressed in [prTorrent]).

Furthermore, both the amount of uploaded and downloaded data should be reported to the tracker to allow checking if there is any inconsistency between the upload and download report, and establish an appropriate credit/trust system. Alternatively, exchange of cryptographic receipts signed by receiving peers can be used to attest to the upload contribution of a peer to the swarm, as

suggested in [Contracts].

10. IANA Considerations

There are presently no IANA considerations with this document.

11. Acknowledgments

The authors would like to thank many people for for their help and comments, particularly: Zhang Yunfei, Liao Hongluan, Roni Even, Bhumip Khasnabish, Wu Yichuan, Peng Jin, Chi Jing, Zong Ning, Song Haibin, Chen Wei, Zhijia Chen, Christian Schmidt, Lars Eggert, David Harrington, Henning Schulzrinne, Kangheng Wu, Martin Stiemerling, Jianyin Zhang, Johan Pouwelse and Arno Bakker.

The authors would also like to thank the people participating in the EU FP7 project SARACEN (contract no. ICT-248474) [refs.saracenwebpage] for contributions and feedback to this document.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the SARACEN project or the European Commission.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.

[ISO.8601.2004] International Organization for Standardization, "Data elements and interchange formats - Information interchange - Representation of dates and times", ISO Standard 8601, December 2004.

12.2. Informative References

[RFC1952] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, May 1996.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[I-D.ietf-ppsp-reqs] Zong, N., Zhang, Y., Avila, V., Williams, C., and L. Xiao, "P2P Streaming Protocol (PPSP) Requirements", draft-ietf-ppsp-reqs-05 (work in progress), October 2011.

[I-D.ietf-ppsp-problem-statement] Zhang, Y., Zong, N., Camarillo, G., Seng, J., and Y. Yang, "Problem Statement of P2P Streaming Protocol (PPSP)", draft-ietf-ppsp-problem-statement-07

(work in progress), November 2011.

[I-D.li-ppsp-nat-traversal-02] Li, L., Wang, J., Chen, W., "PPSP NAT Traversal", draft-li-ppsp-nat-traversal-02 (work in progress), July 2011.

[I-D.xiao-ppsp-reload-distributed-tracker] Xiao, L., Bryan, D., Gu, Y., Tai, X., "A PPSP Tracker Usage for Reload", draft-xiao-ppsp-reload-distributed-tracker-03 (work in progress), October 2011.

[I.D.ietf-alto-protocol] Alimi, R., Penno, R., Yang, Y., "ALTO Protocol", draft-ietf-alto-protocol-10, (work in progress), October 2011.

[I-D.ietf-oauth-v2] Hammer-Lahav, E., Recordon, D., and D. Hardt, "The OAuth 2.0 Authorization Protocol," draft-ietf-oauth-v2-23 (work in progress), January 2012.

[I-D.ietf-oauth-v2-bearer] Jones, M., Hardt, D., and D. Recordon, "The OAuth 2.0 Authorization Protocol: Bearer Tokens," draft-ietf-oauth-v2-bearer-17 (work in progress), February 2012.

[MP4REG] MP4REG, The MPEG-4 Registration Authority, URL: <http://www.mp4ra.org>.

[ISO.IEC.23009-1] ISO/IEC, "Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats", ISO/IEC DIS 23009-1, Aug. 2011.

[ITU-T.H.264] ITU-T, "Advanced video coding for generic audiovisual services," Recommendation H.264 (03/2010), International Telecommunication Union - Telecommunication Standardization Sector, Mar. 2010.

[refs.saracenwebpage] "SARACEN Project Website", http://www.saracen-p2p.eu/.

[prTorrent] Roy, S., Zeng, W., "prTorrent: On Establishment of Piece Rarity in the BitTorrent Unchoking Algorithm", in IEEE Ninth International Conference on Peer-to-Peer Computing, September 2009.

[Contracts] Piatek, M., Venkataramani, A., Yang, R., Zhang, D., Jaffe, A., "Contracts: Practical Contribution Incentives for P2P Live Streaming", in NSDI ’10: USENIX Symposium on

Networked Systems Design and Implementation, April 2010.

Appendix A. PPSP Tracker Protocol XML-Schema

TO BE ADDED.

Appendix B. Media Presentation Description (MPD)

The MPD file describes a Media Presentation, i.e., a bounded or unbounded presentation of media content. In particular, it defines formats to announce resource identifiers for segments and subsegments (layers in case of SVC, descriptions in case of MDC, or views in case of 3D) and to provide the context for these identified resources within a Media Presentation, i.e., describes the structure of the media, the codecs used (as registered with the MP4 registration authority [MP4REG]), the segments and the corresponding mapping within a container file system.

The MPD contains information about the preferred Connection Trackers, than can be classified in tiers of priority (attribute @tier).

The MPD is a Well-Formed XML Document, encoded as double-byte Unicode. The XML-Schema of the MPD aligns with ISO/IEC 23009-1 [ISO.IEC.23009-1].

The following example of MPD is for an on-demand media program encoded in SVC with two alternative SVC streams, two audio streams and a text stream. The example SVC stream has one base layer representation with two complementary enhancement layers for one video resolution and another SVC stream with a base layer and one complementary enhancement representation for a higher video resolution, an audio stream in English and another in Portuguese, and a timed subtitle file in Portuguese. The contents have protection schemes and include the root fingerprints (attribute @hash of element RootFP) in each video and audio groups (for integrity verification purposes).

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD type="OnDemand">
  <ProgramInformation>
    <Title>Movie in SVC</Title>
  </ProgramInformation>
  <Trackers>
    <Tracker url="http://example.com:80" tier="1" />
    <Tracker url="http://example.net:80" tier="2" />
```

```
    </Trackers>
    <SwarmID>1234</SwarmID>
    <Period>
      <BaseURL>Program01</BaseURL>
      <Group mimeType="video; codecs=h264/SVC" lang="en">
        <Representation frameRate="15" width="1280" height="720"
                        id="tag0" bandwidth="32000">
          <ContentProtection schemeIdUri="urn:uuid:706D6953-656C....">
            <RootFP hash="57438tgfkv...." />
          </ContentProtection>
          <SegmentInfo startIndex="0" endIndex="150"
                       duration="PT2.00S" levels="3" />
        </Representation>
        <Representation frameRate="30" width="1920" height="1080"
                        id="tag3" bandwidth="256000">
          <ContentProtection schemeIdUri="urn:uuid:706D6953-656C....">
            <RootFP hash="95448trf6v...." />
          </ContentProtection>
          <SegmentInfo startIndex="0" endIndex="150"
                       duration="PT2.00S" levels="2" />
        </Representation>
      </Group>
      <Group mimeType="video; codecs=h264/SVC" lang="en">
        <Representation frameRate="30" width="1280" height="720"
                        id="tag1" bandwidth="64000"
                        dependencyId="tag0">
          <ContentProtection schemeIdUri="urn:uuid:706D6953-656C....">
            <RootFP hash="2356ac468k...." />
          </ContentProtection>
          <SegmentInfo startIndex="0" endIndex="150"
                       duration="PT2.00S" />
        </Representation>
        <Representation frameRate="60" width="1920" height="1080"
                        id="tag4" bandwidth="512000"
                        dependencyId="tag3">
          <ContentProtection schemeIdUri="urn:uuid:706D6953-656C....">
            <RootFP hash="98216d99ab...." />
          </ContentProtection>
          <SegmentInfo startIndex="0" endIndex="150"
                       duration="PT2.00S" />
        </Representation>
      </Group>
      <Group mimeType="video; codecs=h264/SVC" lang="en">
        <ContentProtection schemeIdUri="urn:uuid:706D6953-656C....">
          <RootFP hash="364t96au9d...." />
        </ContentProtection>
        <Representation frameRate="60" width="1280" height="720"
                        id="tag2" bandwidth="256000"
```

```
                        dependencyId="tag0 tag1">
          <SegmentInfo startIndex="0" endIndex="150"
                       duration="PT2.00S" />
        </Representation>
      </Group>
      <Group mimeType="audio/mp4; codecs=mp4a" lang="en"
             bandwidth="64000">
        <ContentProtection schemeIdUri="http://example.net/drm">
          <RootFP hash="26ft54zd9a...." />
        </ContentProtection>
        <Representation id="tag5">
          <SegmentInfo startIndex="0" endIndex="150"
                       duration="PT2.00S" />
        </Representation>
      </Group>
      <Group mimeType="audio/mp4; codecs=mp4a" lang="pt"
             bandwidth="64000">
        <ContentProtection schemeIdUri="http://example.net/drm">
          <RootFP hash="64fg53zn53...." />
        </ContentProtection>
        <Representation id="tag6">
          <SegmentInfo startIndex="0" endIndex="150"
                       duration="PT2.00S" />
        </Representation>
      </Group>
      <Group mimeType="application/ttml+xml" lang="pt">
        <Representation id="tag7">
          <SegmentInfo>subtitles/Program01-pt.xml</SegmentInfo>
        </Representation>
      </Group>
    </Period>
  </MPD>
```

The MPD file for P2P Streaming contains tracker information and can be compressed with GZIP file format [RFC1952] in order to be used with HTTP compression [RFC2616] for faster transmission times and less network bandwidth usage.

The Client Media Player parses the downloaded MPD file and, if it includes information for P2P Streaming, sends the information to the peer and waits for the response in order to start requesting media chunks to decode and play-out.

The MPD file for Live Streaming has a similar structure but describes a sliding window of a small range in the SegmentInfo element from the live program stream timeline (typically, 10 seconds of video). The sliding window is updated for every new encoded segments (a range of chunks defined by the attributes @startIndex and @endIndex) of the

program stream.

The following excerpt of MPD is for a Live scalable video content. The MPD is updated every 10 seconds while the content is being encoded in real-time. Note that each segment set defined in the Live MPD is self-contained and the necessary information related to eventual content protection and integrity verification keys for the set is provided:

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD type="Live"
     availabilityStartTime="2001-12-17T09:40Z"
     availabilityEndTime="2001-12-17T09:50Z"
     minBufferTime="PT10.00S"
     minimumUpdatePeriodMPD="PT10S">
  <SwarmID>654321xyz</SwarmID>
  <Period start="PT11S">
    <Group mimeType="video; codecs=h264/SVC" lang="en">
      <Representation frameRate="15" width="1280" height="720"
                      id="tag0" bandwidth="32000">
        <ContentProtection schemeIdUri="urn:uuid:706D6953-656C....">
          <RootFP hash="57438tgfkv...." />
        </ContentProtection>
        <SegmentInfo startIndex="5" endIndex="9"
                     duration="PT2.00S" levels="3" />
      </Representation>
      .... more descriptions ....
    </Group>
      .... more descriptions ....
  </Period>
</MPD>
```

Appendix C. PPSP Requirements Compliance

C.1. PPSP Basic Requirements

PPSP.REQ-1: The design of the Tracker protocol in this document allows the Peer Protocol to be similar in terms of design, message formats and flows.

PPSP.REQ-2: The design of the Tracker protocol in this document enables peers to receive streaming content within required time constraints.

PPSP.REQ-3: Each peer has a unique ID (i.e., Peer-ID) that identifies the peer in all swarms joined.

PPSP.REQ-4: Each streaming content is uniquely identified by a Swarm-ID.

PPSP.REQ-5: The streaming content is partitioned into chunks individually addressable.

PPSP.REQ-6: Each chunk has an unique ID in the swarm and is individually addressable.

PPSP.REQ-7: The Tracker Protocol is carried over TCP.

PPSP.REQ-8: The Tracker Protocol is designed to facilitate acceptable QoS, supporting, without special algorithms, adaptive and scalable video and 3D video, for both Video On Demand (VoD) and Live video services, allowing additionally complementary mechanisms like super peers, in-network storage, alternative peer addresses and usage of QoS information for advanced peer selection.

C.2. PPSP Tracker Protocol Requirements

PPSP.TP.REQ-1: The Tracker Protocol implements the reception of queries from peers, such as those in JOIN and FIND messages and periodical peer status reports (STAT_REPORT), as well as the corresponding replies.

PPSP.TP.REQ-2: The peer MUST implement the Tracker Protocol designed in this draft.

PPSP.TP.REQ-3: The tracker request messages JOIN and FIND allow the requesting of peer list from the tracker with respect to a specific Swarm-ID and include preferred number of peers in the peer list as well as peer properties which enable appropriate candidate peer selections by the tracker.

PPSP.TP.REQ-4: The tracker responses from JOIN and FIND messages allow the tracker to offer the peer list to the requesting peer with respect to a specific Swarm-ID.

PPSP.TP.REQ-5: The Tracker supports generating the peer lists with the help of traffic optimization services like ALTO.

PPSP.TP.REQ-6: The STATUS_REPORT message informs the Tracker about the peer’s activity in the swarm.

PPSP.TP.REQ-7: The chunk availability information (ChunkMaps) of the Peer (for all joined swarms) is reported to the tracker in STATUS_REPORT messages.

PPSP.TP.REQ-8: The ChunkMaps exchanged between peer and tracker can be expressed as compact encoded strings.

PPSP.TP.REQ-9: The STATUS_REPORT message informs the tracker about the peer status and capabilities.

C.3. PPSP Security Considerations

PPSP.SEC.REQ-1: The Tracker Protocol supports closed swarms, where the peers are required to be authenticated.

PPSP.SEC.REQ-2: Confidentiality of the streaming content can be supported, and the corresponding key management mechanisms can be negotiated in the authentication and authorization phase (via CONNECT message) before the peer JOINs the swarm.

PPSP.SEC.REQ-3: The Tracker Protocol uses security layers to encrypt the data exchanged among the PPSP entities.

PPSP.SEC.REQ-4: The Tracker Protocol security layer mechanisms help to limit potential damages caused by malfunctioning and badly behaving peers in the P2P streaming system. The streaming mechanisms considered in the PPSP-TP model prevent pollution of contents.

PPSP.SEC.REQ-6: The use of trusted trackers and peer authentication and authorization mechanisms capable to provide additional security and confidentiality, allow to mitigate and prevent peers from DoS attacks.

PPSP.SEC.REQ-7: The Tracker Protocol design supports distributed tracker architectures, providing robustness to the streaming service in case of centralized tracker failure.

PPSP.SEC.REQ-8: The Tracker Protocol use of Transport Layer Security mechanisms avoids the need for developing new security mechanisms.

PPSP.SEC.REQ-9: The Tracker Protocol together with the Media Presentation Description (MPD) allow the use of streaming content integrity mechanisms.

Authors' Addresses

Rui Santos Cruz
IST/INESC-ID/INOV
Phone: +351.939060939
Email: rui.cruz@ieee.org

Gu Yingjie
Huawei
Phone: +86-25-56624760
Fax: +86-25-56624702
Email: guyingjie@huawei.com

Mario Serafim Nunes
IST/INESC-ID/INOV
Rua Alves Redol, n.9
1000-029 LISBOA, Portugal
Phone: +351.213100256
Email: mario.nunes@inov.pt

David A. Bryan
Polycom
P.O. Box 6741
Williamsburg, Virginia 23188
United States of America
Phone: +1.571.314.0256
Email: dbryan@ethernot.org

Jinwei Xia
Huawei
Nanjing, Baixia District 210001
China
Phone: +86-025-86622310
Email: xiajinwei@huawei.com

Joao P. Taveira
IST/INOV
Email: joao.silva@inov.pt

Deng Lingli
China Mobile

PPSP
Internet-Draft
Intended status: Informational
Expires: November 15, 2013

Y. Zhang
Unaffiliated
N. Zong
Huawei Technologies
May 14, 2013

Problem Statement and Requirements of Peer-to-Peer Streaming Protocol (PPSP)
draft-ietf-ppsp-problem-statement-15

Abstract

Peer-to-Peer(P2P for short) streaming systems show more and more popularity in current Internet with proprietary protocols. This document identifies problems of the proprietary protocols, proposes the development of Peer to Peer Streaming Protocol(PPSP) including the tracker and peer protocol, and discusses the scope, requirements and use cases of PPSP.

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 November 15, 2013.

Copyright Notice

Table of Contents

1. Introduction

1.1. Backgrounds

Streaming traffic is among the largest and fastest growing traffic on the Internet [Cisco], where peer-to-peer (P2P) streaming contributes substantially. With the advantage of high scalability and fault tolerance against single point of failure, P2P streaming applications are able to distribute large-scale, live and video on demand (VoD) streaming programs to a large audience with only a handful of servers. What's more, along with the players like CDN providers joining in the effort of using P2P technologies in distributing their serving streaming content, there are more and more various players in P2P streaming ecosystem.

Given the increasing integration of P2P streaming into the global content delivery infrastructure, the lack of an open, standard P2P streaming signaling protocol suite becomes a major missing component. Almost all of existing systems use their proprietary protocols. Multiple, similar but proprietary protocols result in repetitious development efforts for new systems, and the lock-in effects lead to substantial difficulties in their integration with other players like CDN. For example, in the enhancement of existing caches and CDN systems to support P2P streaming, proprietary protocols may increase the complexity of the interaction with different P2P streaming applications.

In this document we propose the development of an open P2P Streaming Protocol, which is abbreviated as PPSP, to standardize signaling operations in P2P streaming systems to solve the above problems.

1.2. Requirements Language

The key words "MUST" and "MUST NOT" in this document are to be interpreted as described in RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

2. Terminology and concepts

CHUNK: A CHUNK is a basic unit of data organized in P2P streaming for storage, scheduling, advertisement and exchange among peers [VoD]. A CHUNK size varies from several KBs to several MBs in different systems. In case of MBs size CHUNK scenario, a sub-CHUNK structure named piece is often defined to fit in a single transmitted packet. A streaming system may use different granularities for different usage, e.g., using CHUNKs during data exchange, and using a larger unit such as a set of CHUNKs during advertisement.

CHUNK ID: The identifier of a CHUNK in a content stream.

CLIENT: A CLIENT refers to a participant in a P2P streaming system that only receives streaming content. In some cases, a node not having enough computing and storage capabilities will act as a CLIENT. Such node can be viewed as a specific type of PEER.

CONTENT DISTRIBUTION NETWORK (CDN): A CDN is a collection of nodes that are deployed, in general, at the network edge like Points of Presence (POP) or Data Centers (DC) and that store content provided by the original content servers. Typically, CDN nodes serve content to the users located nearby topologically.

LIVE STREAMING: It refers to a scenario where all the audiences receive streaming content for the same ongoing event. It is desired that the lags between the play points of the audiences and streaming source be small.

P2P CACHE: A P2P CACHE refers to a network entity that caches P2P traffic in the network and, either transparently or explicitly, streams content to other PEERs.

PEER: A PEER refers to a participant in a P2P streaming system that not only receives streaming content, but also caches and streams streaming content to other participants.

PEER LIST: A list of PEERs which are in a same SWARM maintained by the TRACKER. A PEER can fetch the PEER LIST of a SWARM from the TRACKER or from other PEERs in order to know which PEERs have the required streaming content.

PEER ID: The identifier of a PEER such that other PEERs, or the TRACKER, can refer to the PEER by using its ID.

PPSP: The abbreviation of Peer-to-Peer Streaming Protocols. PPSP refer to the primary signaling protocols among various P2P streaming system components, including the TRACKER and the PEER.

TRACKER: A TRACKER refers to a directory service that maintains a list of PEERs participating in a specific audio/video channel or in the distribution of a streaming file. Also, the TRACKER answers PEER LIST queries received from PEERs. The TRACKER is a logical component which can be centralized or distributed.

VIDEO-ON-DEMAND (VoD): It refers to a scenario where different audiences may watch different parts of the same recorded streaming with downloaded content.

SWARM: A SWARM refers to a group of PEERs who exchange data to distribute CHUNKs of the same content (e.g. video/audio program, digital file, etc.) at a given time.

SWARM ID: The identifier of a SWARM containing a group of PEERs sharing a common streaming content.

SUPER-NODE: A SUPER-NODE is a special kind of PEER deployed by ISPs. This kind of PEER is more stable with higher computing, storage and bandwidth capabilities than normal PEERs.

3. Problem statement

The problems caused by proprietary protocols for P2P streaming applications are listed as follows.

3.1. Heterogeneous P2P traffic and P2P cache deployment

ISPs are faced with different P2P streaming application introducing substantial traffic into their infrastructure, including their backbone and their exchange/interconnection points. P2P caches are used by ISPs in order to locally store content and hence reduce the P2P traffic. P2P caches usually operate at the chunk or file granularity.

However, unlike web traffic that is represented by HTTP requests and responses and therefore allows any caching device to be served (as long as it supports HTTP), P2P traffic is originated by multiple P2P applications which require the ISPs to deploy different type of caches for the different types of P2P streams.

This increases both engineering and operational costs dramatically.

3.2. QoS issue and CDN deployment

P2P streaming is often criticized due to its worse QoS performance compared to client/server streaming (e.g., longer startup delay, longer seek delay and channel switch delay). Hybrid CDN/P2P is a good approach in order to address this problem [Hybrid CDN P2P].

In order to form the hybrid P2P+CDN architecture, the CDN must be aware of the specific P2P streaming protocol in the collaboration. Similarly to what is described in section 3.1, proprietary P2P protocols introduce complexity and deployment cost of CDN.

3.3. Extended applicability in mobile and wireless environment

Mobility and wireless are becoming increasingly important in today's Internet, where streaming service is a major usage. It's reported that the average volume of video traffic on mobile networks has risen up to 50% in the early of 2012 [ByteMobile]. There are multiple prior studies exploring P2P streaming in mobile and wireless networks [Mobile Streaming1] [Mobile Streaming2].

However it's difficult to directly apply current P2P streaming protocols (even assuming we can re-use some of the proprietary ones) in mobile and wireless networks.

Following are some illustrative problems:

First, P2P streaming assumes a stable Internet connection in downlink and uplink direction, with decent capacity and peers that can run for hours. This isn't the typical setting for mobile terminals. Usually the connections are unstable and expensive in terms of energy consumption and transmission (especially in uplink direction). To enable mobile/wireless P2P streaming feasible, trackers may need more information on peers like packet loss rate, peer battery status and processing capability during peer selection compared to fixed peers. Unfortunately current protocols don't convey this kind of information.

Second, current practices often use a "bitmap" message in order to exchange chunk availability. The message is of kilobytes in size and exchanged frequently, e.g., an interval of several seconds or less. In a mobile environment with scarce bandwidth, the message size may need to be shortened or it may require more efficient methods for expressing and distributing chunk availability information, which is different from wire-line P2P streaming.

Third, for a resource constraint peer like mobile handsets or set-top boxes (STB), there are severe contentions on limited resource when using proprietary protocols. The terminal has to install different streaming client software for different usages, e.g., some for movies and others for sports. Each of these applications will compete for the same set of resources even when it is sometimes running in background mode. PPSP can alleviate this problem with the basic idea that the "one common client software with PPSP and different scheduling plug-ins" is better than "different client software running at the same time" in memory and disk consumption.

4. Tasks of PPSP: Standard peer to peer streaming protocols

PPSP is targeted to standardize signaling protocols to solve the above problems that support either live or VoD streaming. PPSP

supports both centralized tracker and distributed trackers. In distributed trackers, the tracker functionality is distributed in decentralized peers. In the following part of this section, the tracker is a logic conception, which can be implemented in a dedicated tracker server or in peers.

The PPSP design includes a signaling protocol between trackers and peers (the PPSP "tracker protocol") and a signaling protocol among the peers (the PPSP "peer protocol") as shown in Figure 1. The two protocols enable peers to receive streaming content within the time constraints.

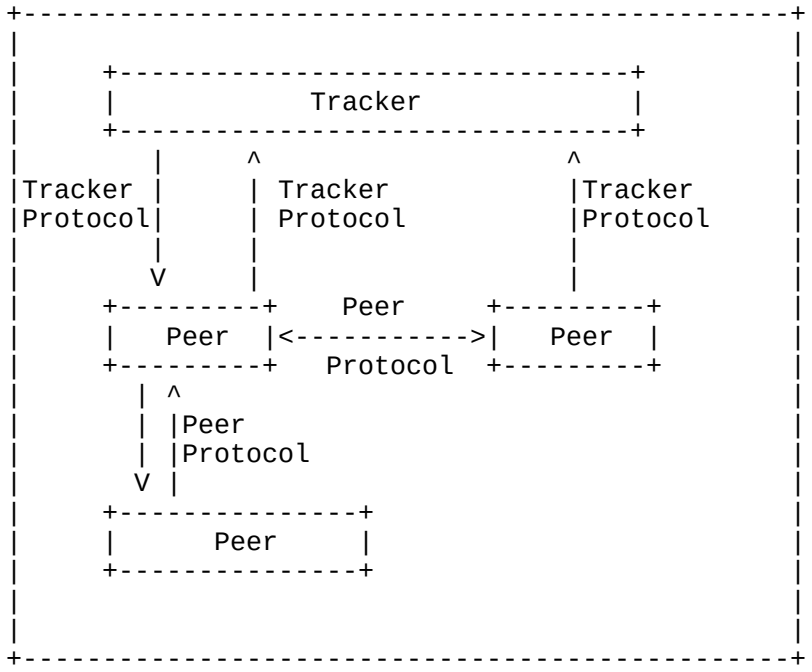


Figure 1 PPSP System Architecture

PPSP design in general needs to solve the following challenges, e.g.

1) When joining a swarm, how does a peer know which peers it should contact for content?

2) After knowing a set of peers, how does a peer contact with these peers? In which manner?

3) How to choose peers with better service capabilities, and how to collect such information from peers?

4) How to improve the efficiency of the communication, e.g. compact on-the-wire message format and suitable underlying transport mechanism (UDP or TCP)?

5) How to improve the robustness of the system using PPSP, e.g. when the tracker fails? How to make the tracker protocol and the peer protocol loose coupled?

4.1. Tasks and design issues of Tracker protocol

The tracker protocol handles the initial and periodic exchange of meta-information between trackers and peers, such as peer list and content information.

Therefore tracker protocol is best modeled as a request/response protocol between peers and trackers, and will carry information needed for the selection of peers suitable for real-time/VoD streaming.

Special tasks for the design of the tracker protocol are listed as follows. This is a high-level task-list. The detailed requirements on the design of the tracker protocol are explicated in section 6.

1) How should a peer be globally identified? This is related to the peer ID definition, but irrelevant to how the peer ID is generated.

2) How to identify different peers, e.g. peers with public or private IP address, peers behind or not behind NAT, peers with IPV4 or IPV6 addresses, peers with different property?

3) The tracker protocol must be light-weight, since a tracker may need to server large amount of peers. This is related to the encoding issue (e.g., Binary based or Text based) and keep-alive mechanism.

4) How can the tracker be able to report optimized peer list to serve a particular content. This is related to status statistic, with which the tracker can be aware of peer status and content status.

PPSP tracker protocol will consider all these issues in the design according to the requirements from both peer and tracker perspective and also taking into consideration deployment and operation perspectives.

4.2. Tasks and design issues of Peer protocol

The peer protocol controls the advertising and exchange of content between the peers.

Therefore peer protocol is modeled as a gossip-like protocol with periodic exchanges of neighbor and chunk availability information.

Special tasks for the design of the peer protocol are listed as follows. This is a high-level task-list. The detailed requirements on the design of the peer protocol are explicated in section 6.

1) How does the certain content be globally identified and verified? Since the content can be retrieved from everywhere, how to ensure the exchanged content between the peers is authentic?

2) How to identify the chunk availability in the certain content? This is related to the chunk addressing and chunk state maintenance. Considering the large amount of chunks in the certain content, light-weight expression is necessary.

3) How to ensure the peer protocol efficiency? As we mentioned in section 3, the chunk availability information exchange is quite frequent. How to balance the information exchange size and amount is a big challenge. What kind of encoding and underlying transport mechanism (UDP or TCP) is used in the messages?

PPSP peer protocol will consider all the above issues in the design according to the requirements from the peer perspective.

5. Use cases of PPSP

This section is not the to-do list for the WG, but for the explanatory effect to show how PPSP could be used in practice.

5.1. Worldwide provision of live/VoD streaming

The content provider can increase live streaming coverage by introducing PPSP in between different providers. This is quite similar to the case described in CDNI [RFC6707][RFC6770].

We suppose a scenario that there is only provider A (e.g., in China) providing the live streaming service in provider B (e.g., in USA) and C (e.g., in Europe)’s coverage. Without PPSP, when a user(e.g. a Chinese American) in USA requests the program to the tracker (which is located in A’s coverage), the tracker may generally return to the user with a peer list including most of peers in China, because generally most users are in China and there are only few users in USA. This may affect the user experience. But if we can use the PPSP tracker protocol to involve B and C in the cooperative

provision, as shown in Figure 2, even when the streaming is not hot to attract many users in USA and Europe to view, the tracker in A can optimally return the user with a peer list including B's Super-nodes (SN for short) and C's SN to provide a better user performance. Furthermore User@B and User@C can exchange data (availability) with these local SNs using the peer protocol.

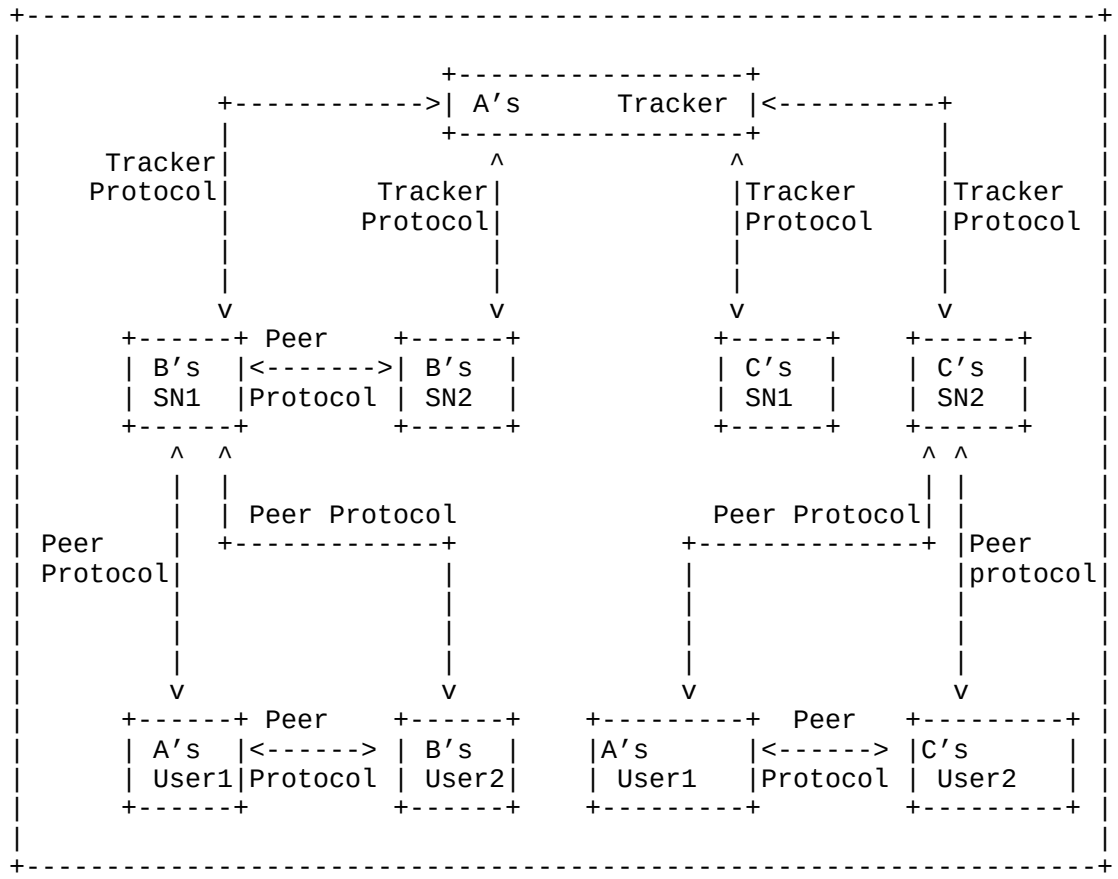


Figure 2 Cooperative Vendors Interaction

5.2. Enabling CDN for P2P VoD streaming

Figure 3 shows the case of enabling CDN to support P2P VoD streaming from different content providers by introducing PPSP inside CDN overlays. It is similar to Figure 2 except that the intermediate SNs are replaced by 3rd party CDN surrogates. The CDN nodes talk with the different streaming systems (including trackers and peers) with the same PPSP protocols.

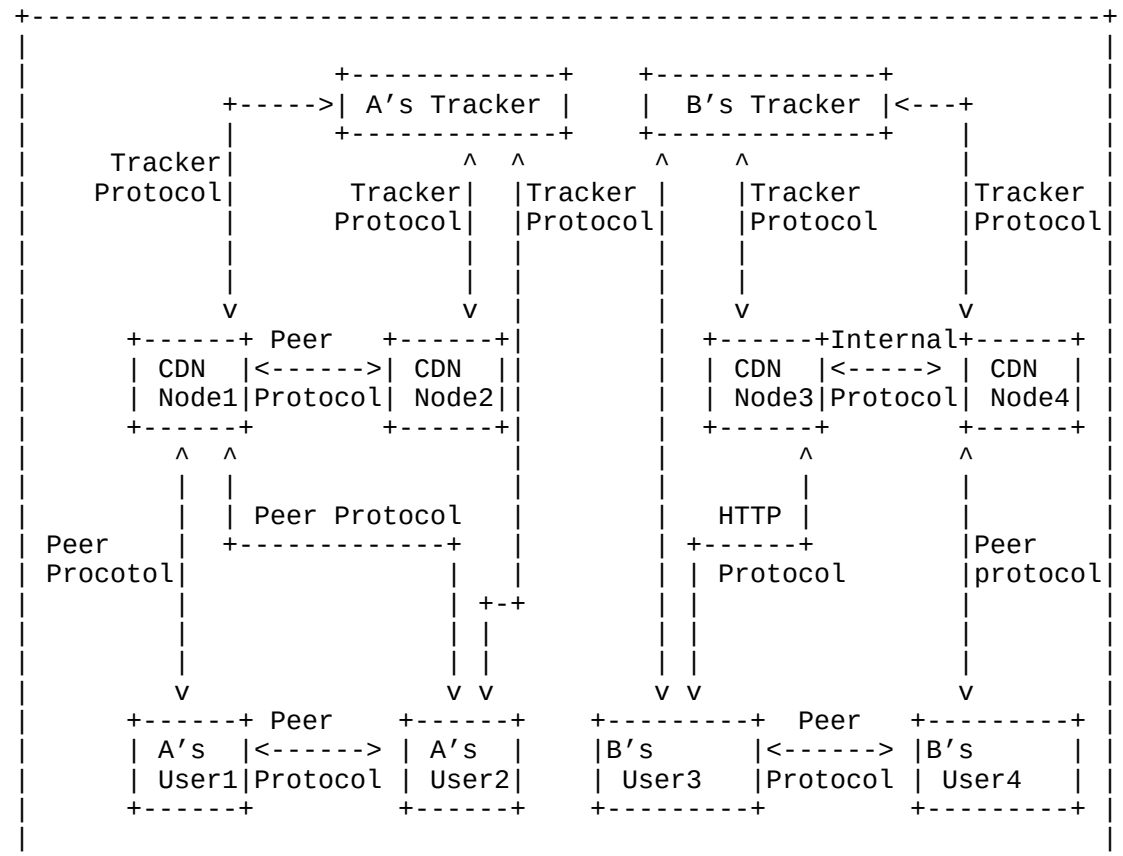


Figure 3 CDN Supporting P2P Streaming

Furthermore the interaction between the CDN nodes can be executed by either existing (maybe proprietary) protocols or the PPSP peer protocol. The peer protocol is useful for building new CDN systems (e.g., operator CDN) supporting streaming in a low cost.

Note that for compatibility reason both HTTP streaming and P2P streaming can be supported by CDN from the users' perspective.

5.3. Cross-screen streaming

In this scenario PC, STB/TV and mobile terminals from both fixed network and mobile/wireless network share the streaming content. With PPSP, peers can identify the types of access networks, average load, peer abilities and get to know what content other peers have even in different networks(potentially with the conversion of the

content availability expression in different networks) as shown in Figure 4.

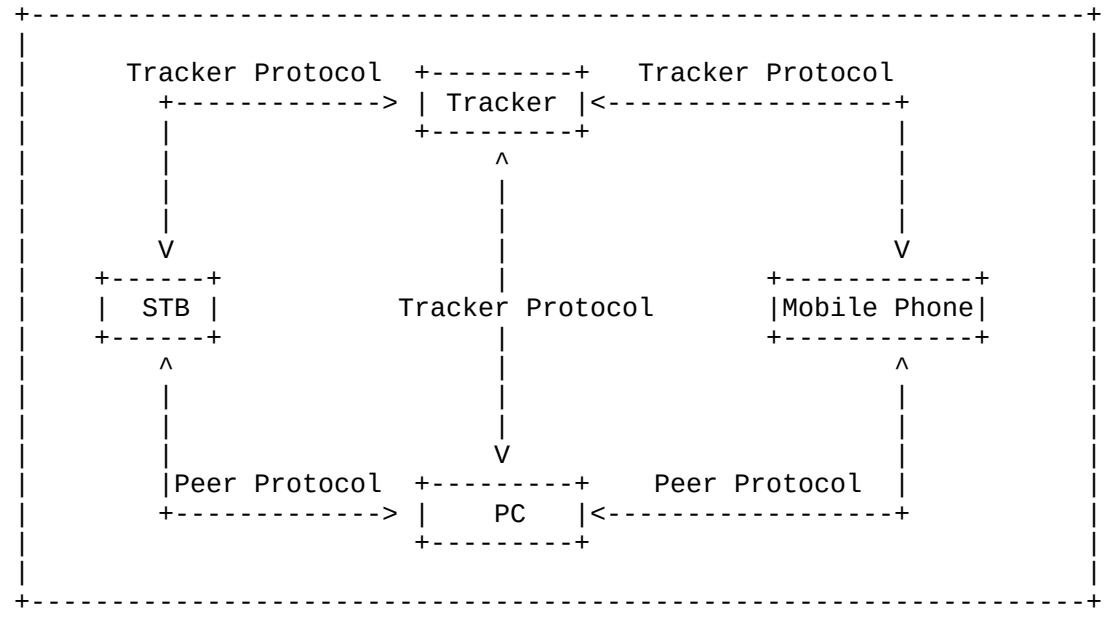
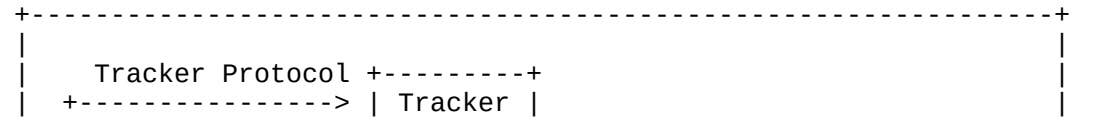


Figure 4 Heterogeneous P2P Streaming with PPSP

Such information will play an important role on selecting suitable peers, e.g., a PC or STB is more likely to provide stable content and a mobile peer within a high-load cell is unlikely to be selected, which may otherwise lead to higher load on the base station.

5.4. Cache service supporting P2P streaming

In Figure 5, when peers request the P2P streaming data, the cache nodes intercept the requests and ask for the frequently visited content (or part of) on behalf of the peers. To do this, it asks the tracker for the peer list and the tracker replies with external peers in the peer list. After the cache nodes exchange data with these peers, it can also act as a peer and report what it caches to the tracker and serve inside requesting peers afterward. This operation greatly decreases the inter-network traffic in many conditions and increases user experience.



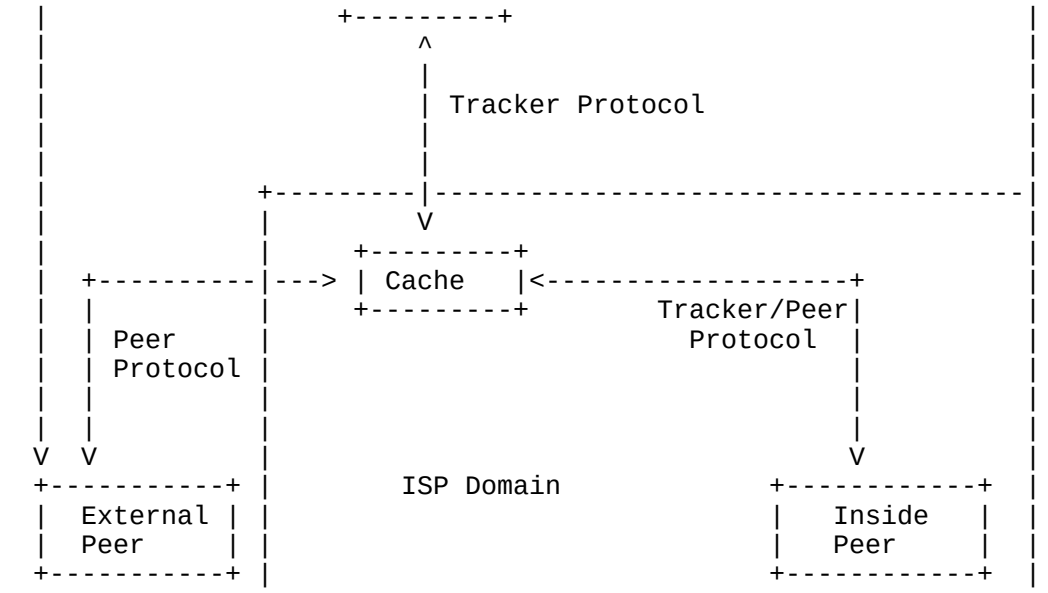


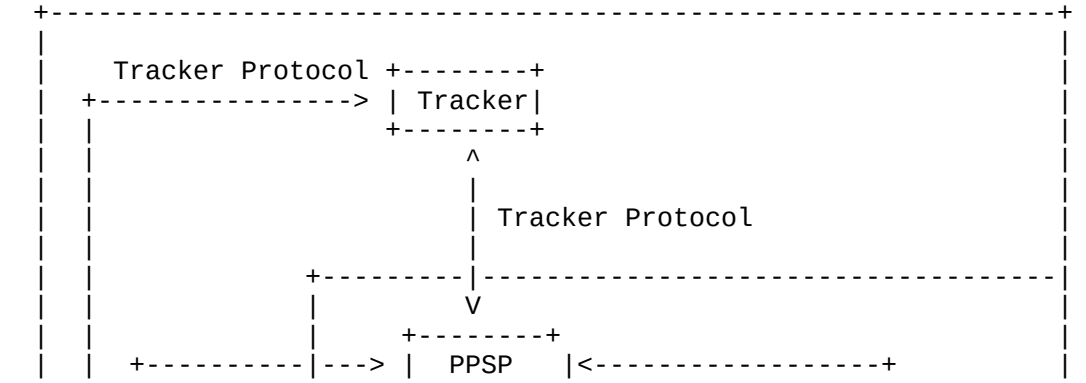
Figure 5 Cache Service Supporting Streaming with PPSP

The cache nodes do not need to update their library when new applications supporting PPSP are introduced, which reduces the cost.

5.5. Proxy service supporting P2P streaming

5.5.1. Home Networking Scenario

For applications where the peer is not co-located with the Media Player in the same device (e.g. the peer is located in a Home Media Gateway), we can use a PPSP Proxy, as shown in figure 6.



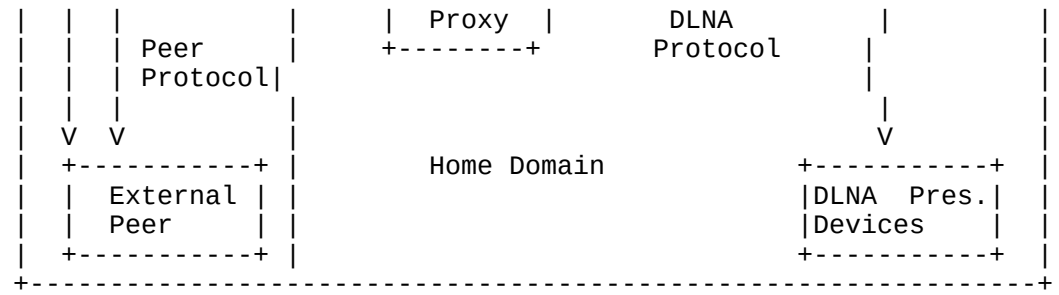


Figure 6 Proxy service Supporting P2P Streaming

As shown in figure 6, the PPSP Proxy terminates both the tracker and peer protocol allowing the legacy presentation devices to access P2P streaming content. In figure 6 the DLNA protocol [DLNA] is used in order to communicate with the presentation devices thanks to its wide deployment. Obviously, other protocols can also be used.

5.5.2. Browser-based HTTP Streaming

P2P Plug-ins are often used in browser-based environment in order to stream content. With P2P plug-ins, HTTP streaming can be turned into a de facto P2P streaming. From the browser (and hence the user) perspective, it's just HTTP based streaming but the PPSP capable plug-in can actually accelerate the process by leveraging streams from multiple sources/peers [P2PYoutube]. In this case the plug-ins behave just like the proxy.

6. Requirements of PPSP

This section enumerates the requirements that should be considered when designing PPSP.

6.1. Basic Requirements

PPSP.REQ-1: Each peer MUST have a unique ID (i.e., peer ID).

It's a basic requirement for a peer to be uniquely identified in a P2P streaming system so that other peers or tracker can refer to the peer by ID.

Note that a peer can join multiple swarms with a unique ID, or change swarm without changing its ID.

PPSP.REQ-2: The streaming content MUST be uniquely identified by a swarm ID.

A swarm refers to a group of peers sharing the same streaming content. A swarm ID uniquely identifies a swarm. The swarm ID can be used in two cases: 1) a peer requests the tracker for the peer list indexed by a swarm ID; 2) a peer tells the tracker about the swarms it belongs to.

PPSP.REQ-3: The streaming content MUST be partitioned into chunks.

PPSP.REQ-4: Each chunk MUST have a unique ID (i.e. chunk ID) in the swarm.

Each chunk must have a unique ID in the swarm so that the peer can understand which chunks are stored in which peers and which chunks are requested by other peers.

6.2. Operation and Management Requirements

This section lists some operation and management requirements following the checklist presented by Appendix A in [RFC5706].

6.2.1. Operation Considerations

PPSP.OAM.REQ-1: PPSP MUST be sufficiently configurable.

According to basic requirements, when setting up PPSP, content provider should generate chunk IDs and swarm ID for each streaming content. Original content server and tracker are configured and setup. Content provider then should publish this information typically by creating web links.

The configuration should allow the proxy-based and end-client scenarios.

PPSP.OAM.REQ-2: PPSP MUST implement a set of configuration parameters with default values.

PPSP.OAM.REQ-3: PPSP MUST support diagnostic operations.

Mechanisms must be supported by PPSP to verify correct operation. The PPSP tracker should collect the status of the peers including peer’s activity, whether it obtained chunks in time, etc. Such information can be used to monitor the streaming behavior of PPSP.

PPSP.OAM.REQ-4: PPSP MUST facilitate achieving quality acceptable to the streaming application.

There are basic quality requirements for streaming systems. Setup time to receive a new streaming channel or to switch between

channels should be reasonably small. End to end delay, which consists of the time between content generation (e.g., a camera) and content consumption (e.g., a monitor), will become critical in case of live streaming especially in provisioning of sport events where end to end delay of 1 minute and more are not acceptable.

For instance, the tracker and peer protocol can carry quality related parameters (e.g. video quality and delay requirements) together with the priorities of these parameters in addition to the measured QoS situation (e.g., performance, available uplink bandwidth) of content providing peers.

PPSP implementations may use techniques such as scalable streaming to handle bandwidth shortages without disrupting playback.

6.2.2. Management Considerations

PPSP.OAM.REQ-5: When management purpose needs to be supported in implementation, PPSP MUST support remote management using standard interface, as well as a basic set of management information.

Due to large-scale peer network, PPSP tracker service or seeders should remotely collect information from peers and expose the information via standard interface for management purpose. Peer information can be collected via PPSP tracker protocol or peer protocol.

The minimum set of management objects should include swarm information such as content characteristics, rate limits, tracking information such as swarm list, log events, peer information such as peer activity, chunk statistics, log event.

PPSP.OAM.REQ-6: PPSP MUST support fault monitoring including peer and server health, as well as streaming behavior of peers.

Peer and server health will at least include node activity and connectivity especially for peers behind NAT. As mentioned in OAM.REQ-4, streaming behavior of peer can be learnt from chunk distribution information.

PPSP.OAM.REQ-7: PPSP MUST support configuration management to define the configuration parameters.

A set of configurable parameters related to chunk generation in PPSP setup stage can be defined by content providers via a management interface to content servers.

PPSP.OAM.REQ-8: PPSP MUST support performance management with respect to streaming performance based on chunk distribution statistics, network load, tracker and peer monitoring.

PPSP.OAM.REQ-9: PPSP MUST support security management. See section of "Security Considerations" in this document.

6.3. PPSP Tracker Protocol Requirements

PPSP.TP.REQ-1: The tracker protocol MUST allow the peer to solicit a peer list in a swarm generated and possibly tailored by the tracker in a query and response manner.

The tracker request message may include the requesting peer's preference parameter (e.g. preferred number of peers in the peerlist) or preferred downloading bandwidth. The tracker will then be able to select an appropriate set of peers for the requesting peer according to the preference.

The tracker may also generate the peer list with the help of traffic optimization services, e.g. ALTO [I-D.ietf-alto-protocol].

PPSP.TP.REQ-2: The tracker protocol MUST report the peer's activity in the swarm to the tracker.

PPSP.TP.REQ-3: The tracker protocol MUST take the frequency of messages and efficient use of bandwidth into consideration, when communicating chunk availability information.

For example, the chunk availability information between peer and tracker can be presented in a compact method, e.g., to express a sequence of continuous "1" more efficiently.

PPSP.TP.REQ-4: The tracker protocol MUST have a provision for tracker to authenticate the peer.

This ensures that only the authenticated users can access the original content in the P2P streaming system.

6.4. PPSP Peer Protocol Requirements

PPSP.PP.REQ-1: The peer protocol MUST allow the peer to solicit the chunk information from other peers in a query and response manner.

PPSP.PP.REQ-2: The chunk information exchanged between a pair of peers MUST NOT be passed to other peers, unless the chunk information is validated (e.g. preventing hearsay and DoS attack).

PPSP.PP.REQ-3: The peer protocol MUST allow the peer to solicit an additional list of peers to that received from the tracker.

It is possible that a peer may need additional peers for certain streaming content. Therefore, it is allowed that the peer communicates with other peers in the current peer list to obtain an additional list of peers in the same swarm.

PPSP.PP.REQ-4: When used for soliciting additional list of peers, the peer protocol MUST contain swarm-membership information of the peers that have explicitly indicated they are part of the swarm, verifiable by the receiver.

PPSP.PP.REQ-5: The additional list of peers MUST only contain peers which have been checked to be valid and online recently (e.g., preventing hearsay and DoS attack).

PPSP.PP.REQ-6: The peer protocol MUST report the peer's chunk availability update.

Due to the dynamic change of the buffered streaming content in each peer and the frequent join/leave of peers in the swarm, the streaming content availability among a peer's neighbors (i.e. the peers known to a peer by getting the peer list from either tracker or peers) always changes and thus requires being updated on time. This update should be done at least on demand. For example, when a peer requires finding more peers with certain chunks, it sends a message to some other peers in the swarm for streaming content availability update. Alternatively, each peer in the swarm can advertise its streaming content availability to some other peers periodically. However, the detailed mechanisms for this update such as how far to spread the update message, how often to send this update message, etc. should leave to the algorithms, rather than protocol concerns.

PPSP.PP.REQ-7: The peer protocol MUST take the frequency of messages and efficient use of bandwidth into consideration, when communicating chunk information.

For example, the chunk availability information between peers can be presented in a compact method.

PPSP.PP.REQ-8: The peer protocol MUST exchange additional information, e.g., status about the peers.

This information can be, for instance, information about the access link or information about whether a peer is running on battery or is connected to a power supply. With such information, a peer can select more appropriate peers for streaming.

7. Security Considerations

This document discusses the problem statement and requirements around P2P streaming protocols without specifying the protocols. However we believe it is important for the reader to understand areas of security introduced by the P2P nature of the proposed solution. The main issue is the usage of un-trusted entities (peers) for service provisioning. For example, malicious peers/trackers may:

Originate denial of service (DOS) attacks to the trackers by sending large amount of requests with the tracker protocol;

Originate fake information on behalf of other peers;

Originate fake information about chunk availability;

Originate reply instead of the regular tracker (man in the middle attack);

leak private information about other peers or trackers.

We list some important security requirements for PPSP protocols as below:

PPSP.SEC.REQ-1: PPSP MUST support closed swarms, where the peers are authenticated or in a private network.

This ensures that only the trusted peers can access the original content in the P2P streaming system. This can be achieved by security mechanisms such as peer authentication and/or key management scheme.

Another aspect is that confidentiality of the streaming content in PPSP need to be supported. In order to achieve this, PPSP should provide mechanisms to encrypt the data exchange among the peers.

PPSP.SEC.REQ-2: Integrity of the streaming content in PPSP MUST be supported to provide a peer with the possibility to identify unauthentic content (undesirable modified by other entities rather than its genuine source).

In a P2P live streaming system a polluter can introduce corrupted chunks. Each receiver integrates into its playback stream the

polluted chunks it receives from its neighbors. Since the peers forwards chunks to other peers, the polluted content can potentially spread through the P2P streaming network.

The PPSP protocol specifications will document the expected threats (and how they will be mitigated by each protocol) and also considerations on threats and mitigations when combining both protocols in an application. This will include privacy of the users and protection of the content distribution.

PPSP.SEC.REQ-3: The security mechanisms in PPSP, such as key management and checksum distribution MUST scale well in P2P streaming systems.

8. IANA Considerations

This document has no actions for IANA.

9. Acknowledgements

Thanks to J.Seng, G. Camarillo, R. Yang,C. Schmidt, R. Cruz, Y. Gu, A.Bakker and S. Previdi for contribution to many sections of this draft. Thank you to C. Williams, V. Pascual and L. Xiao for contributions to PPSP requirements section.

We would like to acknowledge the following people who provided review, feedback and suggestions to this document:M. Stiemerling,D. Bryan, E. Marocco, V. Gurbani, R. Even, H. Zhang, D. Zhang, J. Lei, H.Song, X.Jiang, J.Seedorf, D.Saumitra, A.Rahman, J.Pouwelse, W.Eddy, B. Claise, D. Harrington, J. Arkko and all the AD reviewers.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6707] B. Niven-Jenkins, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, Sep 2012.

[RFC6770] G. Bertrand, "Use Cases for Content Delivery Network Interconnection", RFC6770, Nov 2012.

[RFC5706] D. Harrington, "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC5706, Nov 2009.

10.2. Informative References

[Cisco] Cisco Visual Networking Index: Forecast and Methodology, 2009-2014, http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html

[VoD] Y. Huang et al,Challenges,"Design and Analysis of a Large-scale P2P-VoD System", Sigcomm08.

[ByteMobile]http://www.bytemobile.com/news- events/2012/archive_230212.html

[Mobile Streaming1] Streaming to Mobile Users in a Peer-to-Peer Network,J. Noh etal,MOBIMEDIA ’09.

[Mobile Streaming2] J.Peltotaloetal.,"A real-time Peer-to-Peer streaming system for mobile networking environment",in Proceedings of the INFOCOM and Workshop on Mobile Video Delivery (MoVID ’09).

[Hybrid CDN P2P]D. Xu et al, "Analysis of a CDN-P2Phybrid architecture for cost-effective streaming mediadistribution," SpringerMultimediaSystems, vol.11, no.4, pp.383-399, 2006.

[PPTV] http://www.pptv.com

[PPStream] http://www.ppstream.com

[DLNA] http://www.dlna.org

[P2PYoutube] https://addons.opera.com/en/extensions/details/p2p-youtube/

[I-D.ietf-alto-protocol] R.Alimi et al, "ALTO Protocol", draft-ietf-alto-protocol-13 (work in progress), Sep. 2012.

11. References

Authors’ Addresses

Yunfei Zhang
Unaffiliated

Email: hishigh@gmail.com

Ning Zong
Huawei Technologies

Email: zongning@huawei.com

PPSP
Internet-Draft
Intended status: Informational
Expires: April 10, 2012

N. Zong, Ed.
Huawei Technologies
Y. Zhang
China Mobile Communication Corporation
V. Pascual
Acme Packet
C. Williams
Consultant
L. Xiao
Nokia Siemens Networks

October 08, 2011

P2P Streaming Protocol (PPSP) Requirements
draft-ietf-ppsp-reqs-05

Abstract

The objective of the PPSP work is to standardize the key signaling protocols that apply to tracker and peers in a Peer-to-Peer (P2P) streaming system. These protocols are called PPSP. This document enumerates the requirements for the PPSP, which should be considered when designing PPSP.

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 10, 2012.

Copyright Notice

Table of Contents

1. Introduction

Peer to Peer (P2P) computing has been successfully used in many fields, from one-to-one communication like Voice over IP (VoIP) and Instance Messaging (IM), to one-to-many communication like streaming, file sharing and gaming. In the streaming area, the popularity of P2P real-time and video on demand (VoD) streaming technology has been demonstrated by PPLive [PPLive], PPStream [PPStream], UUSee [UUSee], Pando [Pando] etc. Take PPLive for example, it has over 5 million online users at the same time for real-time streaming. P2P streaming applications account for more and more Internet traffic. According to statistics in a major Chinese Internet Service Provider (ISP), the traffic generated by P2P streaming applications exceeded 50% of the total backbone traffic during peak time in 2008 [I-D.ietf-ppsp-problem-statement].

Given the increasing integration of P2P streaming into the global content delivery infrastructure, the lack of an open, standard P2P streaming protocol has become a major missing component in the Internet protocol stack. Multiple similar but proprietary P2P streaming protocols result in repetitious development efforts and lock-in effects. More importantly, it leads to substantial difficulties when integrating P2P streaming as a component of a global content delivery infrastructure. For example, proprietary P2P streaming protocols do not integrate well with infrastructure devices such as caches and other edge devices [I-D.ietf-ppsp-problem-statement].

The objective of the PPSP work is to standardize the key signaling protocols that apply to tracker and peers in a P2P streaming system. These protocols are called PPSP. PPSP will serve as an enabling technology, building on the development experiences of existing P2P streaming systems. Its design will allow it to integrate with IETF efforts on distributed resource location, traffic localization, and streaming control mechanisms. It allows effective integration with edge infrastructures such as cache and mobile edge equipment [I-D.ietf-ppsp-problem-statement].

This document enumerates the requirements for the PPSP, which should be considered when designing PPSP.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

This document uses the following PPSP-related terms, which are defined in [I-D.ietf-ppsp-problem-statement], including:

Chunk, Live streaming, Peer/PPSP peer, PPSP, Swarm, Tracker/PPSP tracker, Video-on-demand (VoD).

Furthermore, the following additional terms will be used:

Peer list: A list of peers which are in a same swarm maintained by the tracker. A peer can fetch the peer list of a swarm from either tracker or other peers to know which peers have the required streaming content.

Peer ID: An identifier of a peer such that other peers or tracker can refer the ID for the peer.

Swarm ID: An identifier of a swarm containing a group of peers sharing a same streaming content.

Chunk ID: An identifier of a chunk in a streaming content.

3. Overview of PPSP

As described in [I-D.ietf-ppsp-problem-statement], the following components are considered in the scope of PPSP:

1) Tracker communication. Tracker communication is a component that enables each peer to get peer list from the tracker and/or provide content availability to the tracker.

2) Peer communication. Peer communication is a component that enables each peer to exchange content availability and request content from other peers.

3) Report. Report is a component that enables peers to report streaming status to the tracker. The information may include swarm IDs to show swarms that the peer is taking active part in, chunk list for each swarm to show the current content availability in the peer, inbound/outbound traffic capacity, amount of neighbor peers, peer health degree, total amount of bytes uploaded/downloaded to neighbour peers, and other streaming parameters.

Therefore, PPSP includes the PPSP tracker protocol - a signaling protocol between PPSP trackers and PPSP peers, and the PPSP peer protocol - a signaling protocol among PPSP peers.

PPSP tracker protocol will define:

1) Standard format/encoding of information between PPSP peers and PPSP tracker. Some of this exchanged information may be explicitly marked as optional. Exchanged information may include peer list, swarm ID, chunk information, content availability, streaming status including online time, link status, node capability and other streaming parameters.

2) Standard messages between PPSP peers and PPSP trackers defining how PPSP peers report streaming status and request to PPSP trackers, as well as how PPSP trackers reply to the requests.

PPSP peer protocol will define:

1) Standard format/encoding of information among PPSP peers, such as chunk description.

2) Standard messages among PPSP peers defining how PPSP peers advertise chunk availability to each other, as well as the signaling for requesting the chunks among PPSP peers.

This document itemizes requirements for the following aspects of PPSP:

1) Basic requirements to PPSP protocols (peer and tracker protocols), entities (peer and tracker), streaming content, and QoS issues.

2) General requirements to the tracker protocol.

3) General requirements to the peer protocol.

4) Security requirements.

4. PPSP Requirements

4.1. Basic Requirements

PPSP.REQ-1: The tracker and the peer protocols SHOULD be as similar as possible, in terms of design, message formats and flows.

It is desirable that the peer protocol would be an extension to the tracker protocol by adding a few message types, or vice versa.

PPSP.REQ-2: The tracker protocol and the peer protocol SHOULD enable peers to receive streaming content within the required time constraints, i.e., fulfill streaming feature.

PPSP.REQ-3: Each peer MUST have a unique ID (i.e. peer ID) in a

swarm.

It’s a basic requirement for a peer to be uniquely identified in a swarm that other peers or tracker can refer to the peer by ID.

PPSP.REQ-4: The streaming content MUST be uniquely identified by a swarm ID.

A swarm refers to a group of peers sharing the same streaming content. A swarm ID uniquely identifies a swarm. The swarm ID can be used in two cases: 1) a peer requests the tracker for the peer list indexed by a swarm ID; 2) a peer tells the tracker about the swarms it belongs to.

PPSP.REQ-5: The streaming content MUST allow to be partitioned into chunks.

A key characteristic of P2P streaming system is allowing the data fetching from different peers concurrently. Therefore, the whole streaming content must allow to be partitioned into small pieces or chunks for transmission between peers.

PPSP.REQ-6: Each chunk MUST have an unique ID (i.e. chunk ID) in the swarm.

Each chunk must have an unique ID in the swarm such as the peer can understand which chunks are stored in which peers and which chunks are requested by other peers. An example for generating the chunk ID is the buffer map approach [I-D.ietf-ppsp-survey].

PPSP.REQ-7: The tracker protocol and peer protocol are Recommended to be carried over TCP (or UDP, when delivery requirements cannot be met by TCP).

PPSP.REQ-8: The tracker and peer protocol together MUST facilitate acceptable QoS (e.g. low startup delay, low channel/content switching time and minimal end-to-end delay) for both on-demand and live streaming, even for very popular content. The tracker and peer protocol do not include the algorithm required for scalable streaming. However, the tracker and peer protocol SHALL NOT restrict or place limits on any such algorithm.

There are basic QoS requirements for streaming system. Setup time to receive a new streaming channel or to switch between channels should be reasonable small. End to end delay (time between content generation, e.g. camera and content consumption, e.g. user side monitor) will become critical in case of live streaming. Especially in provisioning of sports events, end to end delay of 1 minute and

more are not acceptable.

For instance, the tracker and peer protocols can support carrying QoS related parameters (e.g. video quality, delay requirements) together with the priorities of these parameters, and QoS situation (e.g. performance, available uplink bandwidth) of content providing peers.

There are also some other possible mechanisms, e.g. addition of super peers, in-network storage, request of alternative peer addresses, and the usage of QoS information for an advanced peer selection.

4.2. PPSP Tracker Protocol Requirements

The tracker protocol defines how the peers report and request information to/from the tracker and how the tracker replies to the requests. The tracker discovery and the possible communication between trackers are out of the scope of tracker protocol.

PPSP.TP.REQ-1: The tracker MUST implement the tracker protocol for receiving queries and periodical peer status reports/updates from the peers and for sending the corresponding replies.

PPSP.TP.REQ-2: The peer MUST implement the tracker protocol for sending queries and periodical peer status reports/updates to the tracker and receiving the corresponding replies.

PPSP.TP.REQ-3: The tracker request message MUST allow the requesting peer to solicit the peer list from the tracker with respect to a specific swarm ID.

The tracker request message may also include the requesting peer’s preference parameter, e.g. preferred number of peers in the peer list, or preferred downloading bandwidth. The track will then be able to select an appropriate set of peers for the requesting peer according to the preference.

PPSP.TP.REQ-4: The tracker reply message MUST allow the tracker to offer the peer list to the requesting peer with respect of a specific swarm ID.

PPSP.TP.REQ-5: The tracker SHOULD support generating the peer list with the help of traffic optimization services, e.g. ALTO [I-D.ietf-alto-protocol].

PPSP.TP.REQ-6: The peer status report/update MUST have the ability to inform the tracker about the peer’s activity in the swarm.

PPSP.TP.REQ-7: The chunk availability information of the peer SHOULD

be reported to tracker when tracker needs such information to steer peer selection. The chunk information MUST at least contain the chunk ID.

PPSP.TP.REQ-8: The chunk availability information between peer and tracker MUST be as expressed as compactly as possible.

The peers may report CHUNK AVAILABILTY DIGEST information (i.e. compact expression of chunk availability) to the tracker when possible to decrease the bandwidth consumption for messages in bandwidth constraint environment like mobile network. For example, if a peer has a bitmap like 111111...1(100 continuous 1)xxx..., the 100 continuous "1" can be expressed by one byte with seven bits representing 100 and one bit representing "1". In this example, 100-8=92 bits are saved. Considering the frequency of exchange of CHUNK AVAILBILITY and the fact that many bitmaps have quite a long length of continuous "1" or "0", such compression makes sense.

PPSP.TP.REQ-9: The status of the peer SHOULD be reported to the tracker when tracker needs such information to steer peer selection.

For example, peer status can be online time, physical link status including DSL/WIFI/etc, battery status, processing capability, and other capabilities of the peer. Therefore, the tracker is able to select better candidate peers for streaming.

4.3. PPSP Peer Protocol Requirements

The peer protocol defines how the peers advertise streaming content availability and exchange status with each other. The peer protocol also defines the requests and responses of the chunks among the peers. The first task for this WG will be to decide which signaling and media transfer protocols will be used. The WG will consider existing protocols and, if needed, identify potential extensions to these protocols.

PPSP.PP.REQ-1: The streaming content availability request message MUST allow the peer to solicit the chunk information from other peers in the peer list. The chunk information MUST at least contain the chunk ID. This chunk availability information MUST NOT be passed on to other peer, unless validated (e.g. prevent hearsay and DoS).

PPSP.PP.REQ-2: The streaming content availability reply message MUST allow the peer to offer the information of the chunks in its content buffer. The chunk information MUST at least contain the chunk ID.

PPSP.PP.REQ-3: The streaming content availability request message SHOULD allow the peer to solicit an additional list of peers to that

received from the tracker - with the same swarm ID. The reply message MUST contain swarm-membership information of the peers that have explicitly indicated they are part of the swarm, verifiable by the receiver. This additional list of peers MUST only contain peers which have been checked to be valid and online recently (e.g. prevent hearsay and DoS).

It is possible that a peer may need additional peers for certain streaming content. Therefore, it is allowed that the peer communicates with the peers in the current peer list to obtain an additional list of peers in the same swarm.

PPSP.PP.REQ-4: Streaming content availability update message among the peers MUST be supported by peer protocol. In the push based model, where peers advocate their own chunk availability proactively, the content availability request message described in PP.REQ-1 is not needed. The peer protocol MUST implement either pull-based, push-based or both.

Due to the dynamic change of the buffered streaming content in each peer and the frequent join/leave of peers in the swarm, the streaming content availability among a peer's neighbours (i.e. the peers known to a peer by getting the peer lists from either tracker or peers) always changes and thus requires being updated on time. This update should be done at least on demand. For example, when a peer requires finding more peers with certain chunks, it sends a message to some other peers in the swarm for streaming content availability update. Alternatively, each peer in the swarm can advertise its streaming content availability to some other peers periodically. However, the detailed mechanisms for this update such as how far to spread such update message, how often to send this update message, etc should leave to peer algorithms, rather than protocol concerns.

PPSP.PP.REQ-5: The chunk availability information between peers MUST be as expressed as compactly as possible.

In PP.REQ-1/2/4, the peers may exchange CHUNK AVAILABILTY DIGEST information (i.e. compact expression of chunk availability) to with other peers when possible to decrease the bandwidth consumption for messages in bandwidth constraint environment like mobile network.

PPSP.PP.REQ-6: The peer status report/update SHOULD be advertised among the peers to reflect the status of the peer.

Peer status information should be advertised among the peers via the peer status report/update message. For example, peer status can be online time, physical link status including DSL/WIFI/etc, battery status, processing capability, and other capabilities of the peer.

With this information, a peer can select more appropriate peers for streaming.

PPSP.PP.REQ-7: The peers MUST implement the peer protocol for chunk data (not availability information) requests and responses among the peers before the streaming content is transmitted.

5. Security Considerations

The scope of this section is to analyze the security threats and provide the requirements for PPSP.

PPSP.SEC.REQ-1: PPSP MUST support closed swarms, where the peers are authenticated.

This ensures that only the authenticated users can access the original media in the P2P streaming system. This can be achieved by security mechanisms such as user authentication and/or key management scheme.

PPSP.SEC.REQ-2: Confidentiality of the streaming content in PPSP SHOULD be supported and the corresponding key management scheme SHOULD scale well in P2P streaming system.

PPSP.SEC.REQ-3: PPSP MUST provide an option to encrypt the data exchange among the PPSP entities.

PPSP.SEC.REQ-4: PPSP MUST have mechanisms to limit potential damage caused by malfunctioning and badly behaving peers in the P2P streaming system.

Such an attack will degrade the quality of the rendered media at the receiver. For example, in a P2P live video streaming system a polluter can introduce corrupted chunks. Each receiver integrates into its playback stream the polluted chunks it receives from its other neighbors. Since the peers forwards chunks to other peers, the polluted content can potentially spread through much of the P2P streaming network.

PPSP.SEC.REQ-5: PPSP SHOULD support identifying badly behaving peers, and exclude or reject them from the P2P streaming system.

PPSP.SEC.REQ-6: PPSP MUST prevent peers from DoS attacks which will exhaust the P2P streaming system’s available resource.

Given the prevalence of DoS attacks in the Internet, it is important to realize that a similar threat could exist in a large-scale

streaming system where attackers are capable of consuming a lot of resources with just a small amount of effort.

PPSP.SEC.REQ-7: PPSP SHOULD be robust, i.e., when centralized tracker fails the P2P streaming system SHOULD still work by supporting distributed trackers.

PPSP.SEC.REQ-8: Existing P2P security mechanisms SHOULD be re-used as much as possible in PPSP, to avoid developing new security mechanisms.

PPSP.SEC.REQ-9: Integrity of the streaming content in PPSP MUST be supported to provide a peer with the possibility to identify inauthentic media content (undesirable modified by other entities rather than its genuine source). The corresponding checksum distribution and verification scheme SHOULD scale well in P2P streaming system and be robust against distrustful trackers/peers.

6. IANA Considerations

This document presently raises no IANA considerations.

7. Acknowledgements

The authors would like to thank many people for discussing P2P streaming. We would particularly like to thank: Yingjie Gu, Haibin Song, Xingfeng Jiang from Huawei, Hui Zhang, Jan Seedorf, Martin Stiemerling from NEC Labs, Jun Lei from University of Goettingen, James Seng from PPLive, Das Saumitra from Qualcomm, Christian Schmidt from NSN, Akbar Rahman from Interdigital, Lingli Deng from China Mobile, Johan Pouwelse, Arno Bakker and Wesley Eddy.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[PPLive] "www.pplive.com".

[PPStream] "www.ppstream.com".

[UUSee] "www.uusee.com".

[Pando] "www.pando.com".

[I-D.ietf-ppsp-survey]
Gu, Y., Zong, N., Zhang, H., Zhang, Y., Lei, J., Camarillo, G., Liu, Y., Montuno, D., and X. Lei, "Survey of P2P Streaming Applications", draft-ietf-ppsp-survey-02 (work in progress), July 2011.

[I-D.ietf-alto-protocol]
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-09 (work in progress), June 2011.

[I-D.ietf-ppsp-problem-statement]
Zhang, Y., Zong, N., Camarillo, G., Seng, J., and R. Yang, "Problem Statement of P2P Streaming Protocol (PPSP)", draft-ietf-ppsp-problem-statement-05 (work in progress), September 2011.

Authors' Addresses

Ning Zong (editor)
Huawei Technologies
Huawei Base, No.101 Software Avenue, Nanjing, China

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

Yunfei Zhang
China Mobile Communication Corporation

Phone: +86 13601032119
Email: zhangyunfei@chinamobile.com

Victor Pascual
Acme Packet
Anabel Segura 10, Madrid 28108, Spain

Email: VPascual@acmepacket.com

Carl Williams
Consultant
Palo Alto, California 94306

Email: carlw@mcsr-labs.org

Lin Xiao
Nokia Siemens Networks

Phone: +86 10 84358977
Email: lin.xiao@nsn.com

PPSP Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 11, 2012

L. Li
J. Wang
ZTE Corporation
W. Chen
China Mobile

July 10, 2011

PPSP NAT Traversal
draft-li-ppsp-nat-traversal-02

Abstract

This document discusses the necessity and solutions of PPSP NAT traversal. Two NAT traversal solutions based are described in this document: PPSP-ICE and RELOAD-ICE solution. PPSP-ICE and RELOAD-ICE solutions both use ICE. PPSP-ICE solution uses PPSP messages to convey ICE parameters, while RELOAD-ICE solution proposes to form a RELOAD overlay with PPSP peers and use RELOAD messages to exchange ICE parameters. Extensions to PPSP are also proposed to support these solutions.

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 January 11, 2012.

Copyright Notice

carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction

NAT is widely deployed in the Internet. This document focuses on PPSP NAT traversal issues, and proposes extensions to [I-D.gu-ppsp-tracker-protocol] and [I-D.gu-ppsp-peer-protocol] to support NAT traversal. It discusses the necessity and solutions of PPSP NAT traversal. Two NAT traversal solutions are described in this document: PPSP-ICE solution and RELOAD-ICE solution. PPSP-ICE and RELOAD-ICE solutions both use ICE [RFC5245]. This document also discusses the implementation decisions of NAT traversal.

The major change of this version is adding the No-ICE solution and detailing extensions to PPSP.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

We use the terminology and definitions from "Problem Statement of P2P Streaming Protocol" [I-D.zhang-ppsp-problem-statement] and ICE [RFC5245] and extensively in this document. Other terms used in this document are defined below.

STUN peer. A STUN peer is a peer functioning as a STUN [RFC5389] server and providing STUN services to other peers.

Relay peer. A relay peer is a peer providing relay service to other peers. The relay service may be provided in PPSP layer or TURN layer or both.

TURN peer. A TURN peer is a relay peer providing relay service in TURN [RFC5766] layer. In another word, a TURN peer is a peer functioning as a TURN server and providing TURN services to other peers.

Proxy peer. A proxy peer is a relay peer providing relay service in PPSP layer. In another word, a proxy peer is a peer functioning as a proxy server and providing PPSP proxy services to other peers. Unlike TURN peer, proxy peer only relays PPSP messages.

Proxy candidate. As the defined in ICE, a candidate is a transport address, which is potential for communication. A peer’s proxy candidate is the address of a proxy peer serving for the peer. Through a peer’s proxy candidate, the peer can be contacted.

3. The Necessity of NAT Traversal

Without adopting NAT traversal method, the existence of NATs prevents some PPSP peers from connecting to some other peers. Without NAT traversal, peers MAY not be able to download needed chunks, or MAY take long time to download needed chunks. This probably happens when the ratio of NATed peer is high in a P2P streaming system or a swarm.

When there are NATed peers, adopting NAT traversal allows peers to download contents from more peers, which can increase download speed, avoid NAT-caused download failure and high download latency.

If there is no NAT or the QoE is satisfied without NAT traversal solution, there is no need to apply NAT traversal solution.

Therefore, NAT traversal is necessary at least in some P2P streaming systems. Some commercial P2P streaming systems like UUSEE are using NAT traversal measures.

4. NAT Traversal Solution Overview

This document describes two NAT traversal solutions: PPSP-ICE and RELOAD-ICE. These two solutions both use ICE because ICE is an IETF standard with following advantage. ICE can use any combination of following NAT traversal methods: NAT assisting (e.g. UPNP-IGD), STUN/STUN-like, TURN, connection reversal and hole punching. To use ICE, ICE parameters must be conveyed by application protocol. PPSP-ICE solution conveys ICE parameters with PPSP messages, while RELOAD-ICE solution conveys ICE parameters with RELOAD [I-D.ietf-p2psip-base] messages. These two solutions require discovering NAT traversal service and gathering candidates.

4.1. Candidates and NAT Traversal Service

As the defined in ICE, a candidate is a transport address, which is potential for communication. ICE defines host candidate, reflexive candidate, relayed candidate and NAT-assisted candidate. This document defines one more candidate type called proxy candidate for PPSP-ICE solution.

Among above candidates, host candidate is created by peer itself, NAT-assisted candidate is provided by NAT device, while other candidates can be obtained from nodes providing NAT traversal service. The types of nodes that might provide NAT traversal services in PPSP are listed below. Among below types, Proxy peer can only be used in PPSP-ICE solution. Other types may be used in both NAT traversal solutions in document.

- Dedicated STUN/TURN server. STUN/TURN servers provided by P2P streaming service provider or third party may be utilized for NAT traversal. Dedicated servers are powerful and stable, but costly compared with STUN/TURN peer.

- STUN/TURN peer. In a P2P system, peers may acts as STUN/TURN servers. These peers are called STUN/TURN peers in this document. Utilizing STUN/TURN peers increase the system scalability. Please note that some STUN/TURN peers can also be servers deployed streaming service provider. User nodes acting as STUN/TURN peers make NAT traversal service highly scalable.

- Proxy peer. Publicly accessible PPSP peer can act as proxy of NATed peer. Proxy peer receives PPSP messages destined to NATed peer and forward to the NATed peer. Compared with TURN server/ peer, proxy peer relay only PPSP messages and uses PPSP own authentication method. Please note that a proxy peer can be a user node or a server deployed streaming service provider.

- o STUN-like tracker. Tracker may provide STUN-like service to peers using PPSP protocol. Compared with STUN, providing STUN-like services with PPSP protocol can reduce message number. For example, tracker can discover peer's reflexive address in PPSP JOIN request, and return the reflexive address to peer in PPSP JOIN response.

Reflexive candidate can be discovered with the help of dedicated STUN server, STUN peer or STUN-like tracker. Relayed candidate can be obtained from dedicated TURN server or TURN peer. Proxy candidate is obtained from proxy peer.

Proxy candidate and proxy peer can only be used in PPSP-ICE solution. Other candidates and NAT traversal service node may be used in any NAT traversal solution in document.

4.2. NAT Traversal Service Discovery

Possible methods to discover NAT traversal services are listed below.

- o Traditional methods including DNS, DHCP, manual configuration, etc. The traditional ways are suitable to discover stable and handful service nodes. Dedicated STUN/TURN servers can be discovered using traditional ways.

- o Tracker method. As illustrated in Figure 1, STUN/TURN peers and proxy peers can report their ability to tracker. Then peers can discover them through tracker.

- o RELOAD method. PPSP peers may found a RELOAD overlay and uses RELOAD's TURN discovery method to locate TURN peers. There are two TURN discovery methods defined by RELOAD. One is defined in RELOAD-base draft [I-D.ietf-p2psip-base] for discovering TURN service only. The other is defined in [I-D.ietf-p2psip-service-discovery] for discovering any service including TURN service.

- o Gossip method. PPSP peers gossip to exchange peer list and status. As illustrated in Figure 1, STUN/TURN peer list and proxy peer list may also be exchanged using gossip method. Gossip method can be used as complement to tracker or RELOAD method.

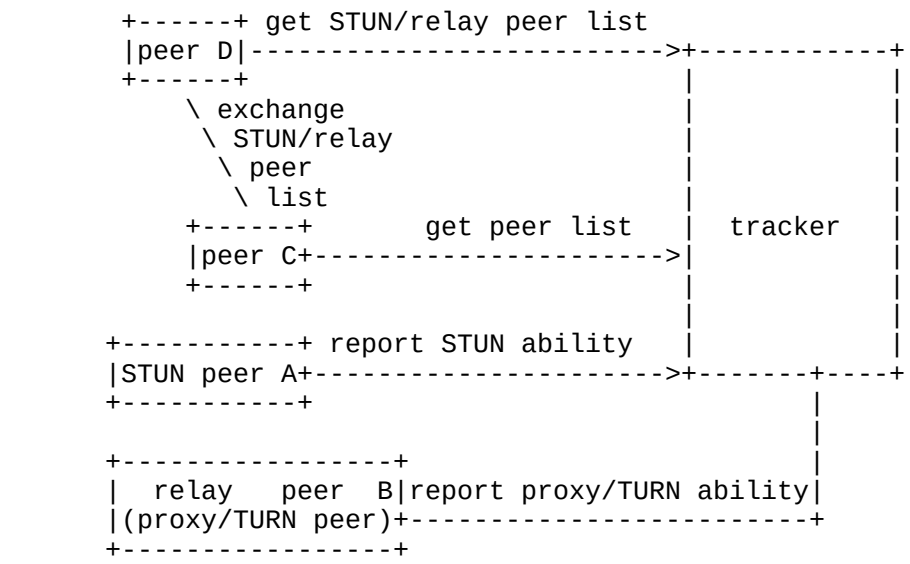


Figure 1: NAT traversal discovery with tracker method and gossip method

RELOAD-ICE solution can use all NAT traversal service discovery methods above, while PPSP-ICE can use all methods except RELOAD method.

5. NAT Traversal Solutions

5.1. PPSP-ICE Solution

5.1.1. PPSP Signal Traversal

The process of PPSP signal traversal is shown in Figure 2. As shown in Figure 2, the process of a peer (peer A) establishing PPSP connection to another peer (peer B) includes following seven steps (step 5 to step 7 is optional). 1, both peer A and B gather their candidates, and report their candidates to tracker when joining swarm. 2, peer A gets peer list from tracker or other peer(s) and learns peer A's candidates from peer list. 3, peer A does one-direction ICE or PPSP connectivity checks from peer A to peer B. 4, peer A chooses candidate pair based the result of one-direction ICE or PPSP connectivity checks. 5, peer B and peer A exchange ICE parameters with PPSP messages. 6, peer A and B performs two-direction ICE connectivity checks or one-direction ICE connectivity checks from peer B to peer A. 7, peer A or peer B chooses candidate pair.

After step 4, peer A has established a communication path to peer B. But the communication path may not be the optimal one, because the path is discovered based on one-direction connectivity checks from peer A to peer B. So step 5 to step 7 is used to find a better communication path. Step 5 to step 7 a standard ICE process. This ICE process is optional because the best communication path may have been found or may not be necessary.

Following sections will describe the key steps of PPSP signal traversal in details.

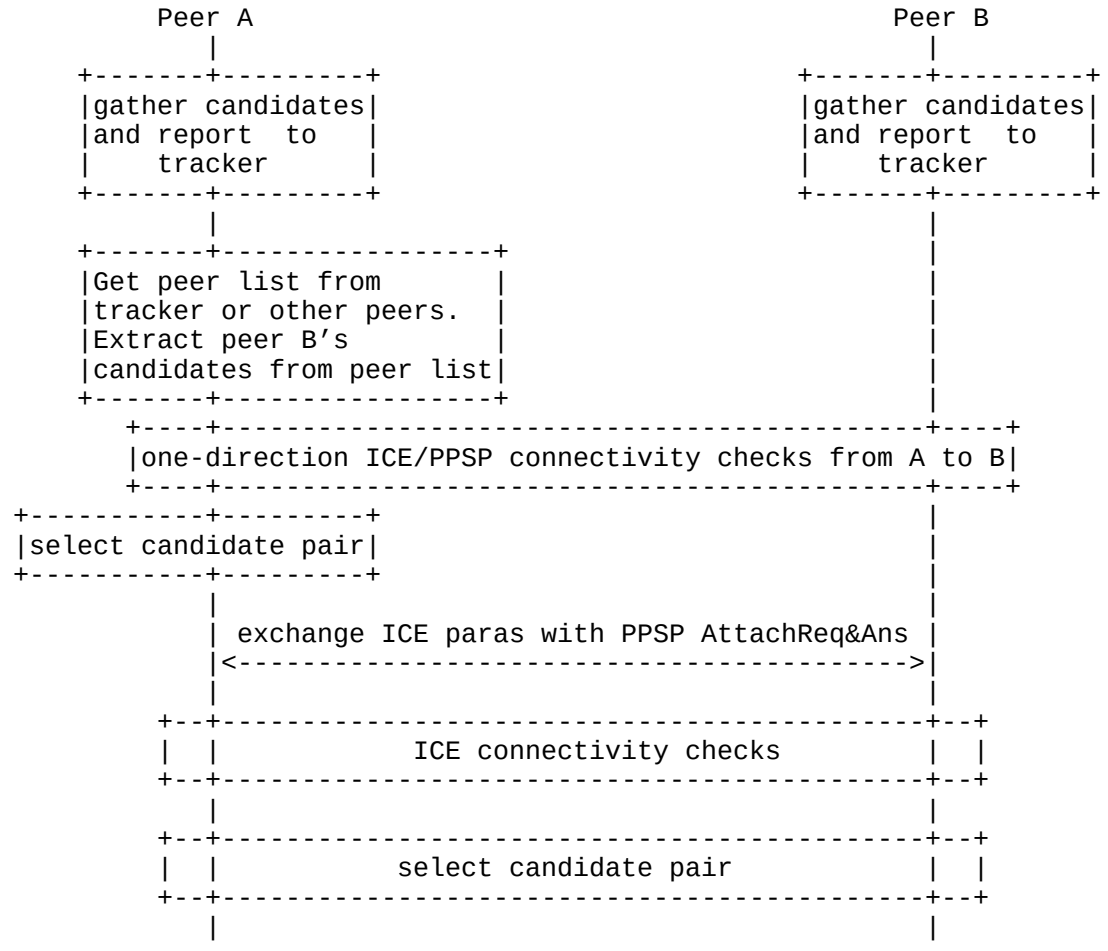


Figure 2: PPSP signal traversal

5.1.1.1. Gathering Candidates

Every peer MUST gather candidates for communication. PPSP-ICE solution may use host candidate, NAT-assisted candidate, reflexive candidate, proxy candidate and relayed candidate.

Every peer MUST gather its candidates according to its accessibility and the type of connectivity check. Proxy candidate is used for the connectivity check in PPSP layer. Relayed candidate is used for the connectivity check in ICE layer. Host candidate, NAT-assisted candidate and reflexive candidate can be used for the connectivity check in both ICE layer and PPSP layer.

5.1.1.2. Conveying Candidates

Candidates are conveyed by PPSP messages. When joining a swarm, a peer puts its candidates and peer ID in the JOIN message sent to tracker. Peer list in the PPSP message contains each peer's candidates and peer ID in the list.

5.1.1.3. One-direction Connectivity Checks

Because peer A doesn't know peer B's candidates, the connectivity checks are one-direction checks. The one-direction connectivity checks can be performed in ICE layer or PPSP layer. This document recommends PPSP layer check because this step's ICE layer check requires modifications to ICE and TURN standard.

If ICE layer check is used in this step, some modifications to ICE and TURN authentication are required because there is no ICE parameter exchanging before.

ICE uses STUN binding request and response to check connectivity. Standard ICE [RFC5245] uses offer/answer exchange to exchange STUN username fragment and password. In standard ICE [RFC5245], the username part of STUN credential is formed by concatenating a username fragment from each ICE agent, separated by a colon. However, there is no offer/answer exchange for the one-direction connectivity checks here. A possible solution is to put STUN username and password in peer list. Then in the example shown in Figure 2, peer A can extract peer B's STUN username and password from peer list.

According to standard ICE [RFC5245], before certain connectivity checks, an ICE agent MUST create permissions in its TURN server for the IP addresses learned from its peer in the offer/answer exchange. However, in the example shown in Figure 2, peer B can't create permissions because peer B doesn't know peer A's IP addresses due to the absence of offer/answer exchange. To address this issue, it needs a new TURN authentication method or another way to create permissions in peer B's TURN server.

If the connectivity checks are performed in PPSP layer, PPSP message is used to test connectivity. In the example shown in Figure 2, peer A simply tries to reach peer B using different candidate pair until it got response from peer B. Connectivity check in PPSP layer can use PPSP's own authentication method.

5.1.1.4. Using ICE to Optimize Connection

After step 4 (selecting candidate pair), connection between peer A and peer B is established based on one-direction connectivity checks. Because the checks are one-direction, the established connection may not be optimal. A better connection may be established by performing a standard ICE with two-direction connectivity checks or one-direction connectivity checks of reverse direction. This document proposes to define new PPSP messages called Attach for candidates exchanging.

5.1.2. PPSP Media Traversal

PPSP-ICE solution uses ICE for media traversal. The way to use ICE here is almost the same as the way defined in [RFC5245]. In ICE as defined by [RFC5245], SDP is used to carry the ICE parameters. This document proposes to define a PPSP message called MediaAttach for exchanging the ICE parameters including candidates and authentication data.

The use of MediaAttach with ICE for NAT traversal is shown in Figure 3 and Figure 4. As shown in these figures, a peer (say peer B) wants to establish media connection to another peer (say peer A). Peer A and peer B already have established PPSP signal connection directly or via a third-party peer (the method to establish signal connection please refers to the above section). So peer A can send a PPSP MediaAttach request to peer B directly or via a third peer. Then Peer B responses to peer A with MediaAttach response message. Through MediaAttach request and response, peer A and peer B exchange ICE parameters. After that, the following NAT traversal process complies with standard ICE [RFC5245].

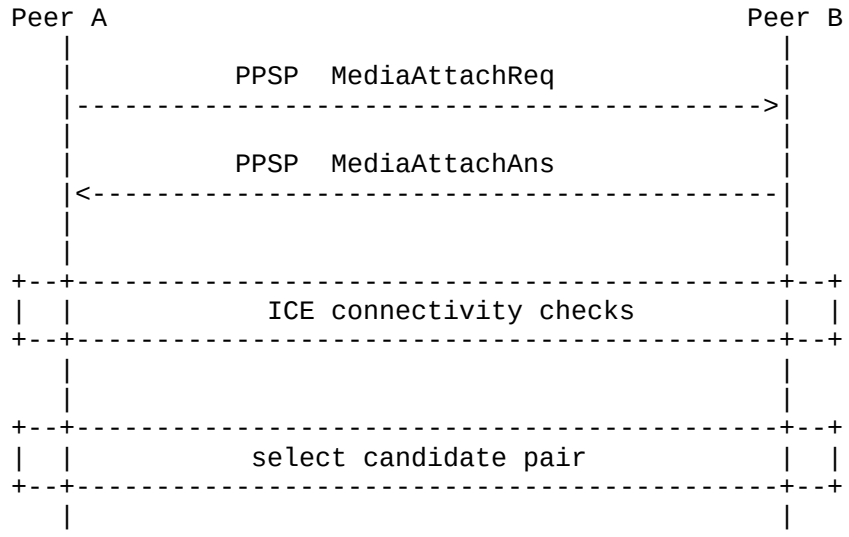


Figure 3: PPSP Media Traversal 1

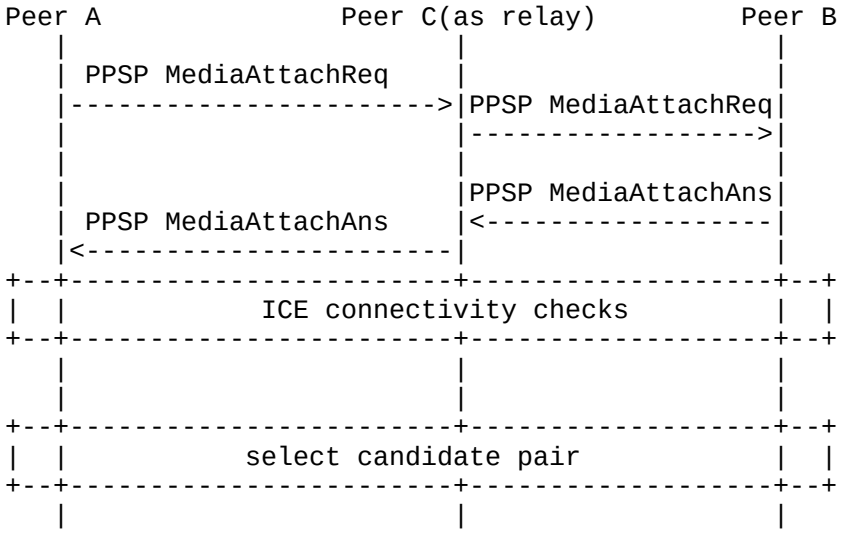


Figure 4: PPSP Media Traversal 2

5.2. RELOAD-ICE Solution

RELOAD-ICE solution uses ICE for PPSP signal and media traversal. RELOAD [I-D.ietf-p2psip-base] defines AppAttach message for exchanging the ICE parameters including candidates and authentication data. Candidates MAY include host candidate, NAT-assisted candidate, reflexive candidate and relayed candidate. NAT traversal service nodes used by RELOAD-ICE solution MAY include dedicated STUN/TURN server, STUN/TURN peer and STUN-like tracker. RELOAD defines two ways to discover TURN service. RELOAD-ICE solution MAY use any other NAT traversal discovery methods described in section 3.2 as well.

The use of AppAttach with ICE for NAT traversal is shown in Figure 5. As shown in this figure, a peer (say peer B) wants to establish PPSP signal or media connection to another peer (say peer A). Peer A can send a RELOAD AppAttach request to peer B through RELOAD overlay routing. Then Peer B responses to peer A with AppAttach response message through RELOAD overlay routing. Through MediaAttach request and response, peer A and peer B exchange ICE parameters. After that, the following NAT traversal process complies with standard ICE [RFC5245].

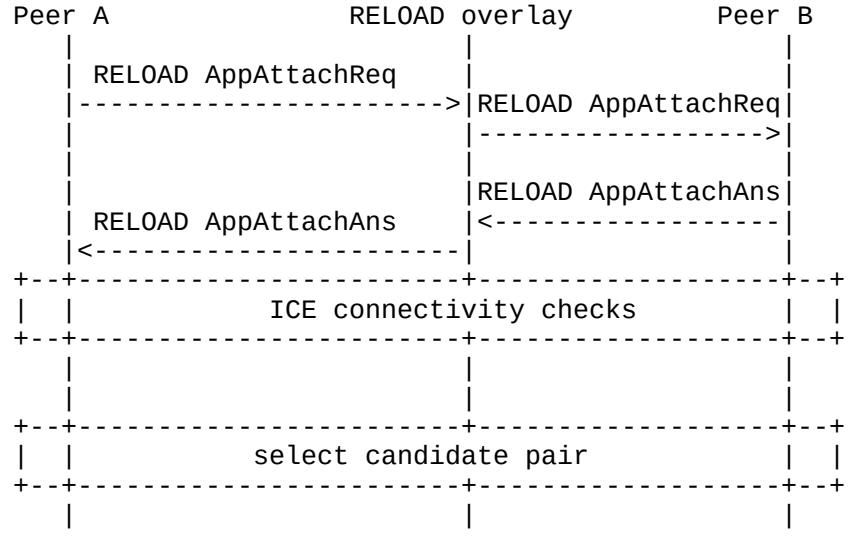


Figure 5: NAT Traversal with RELOAD

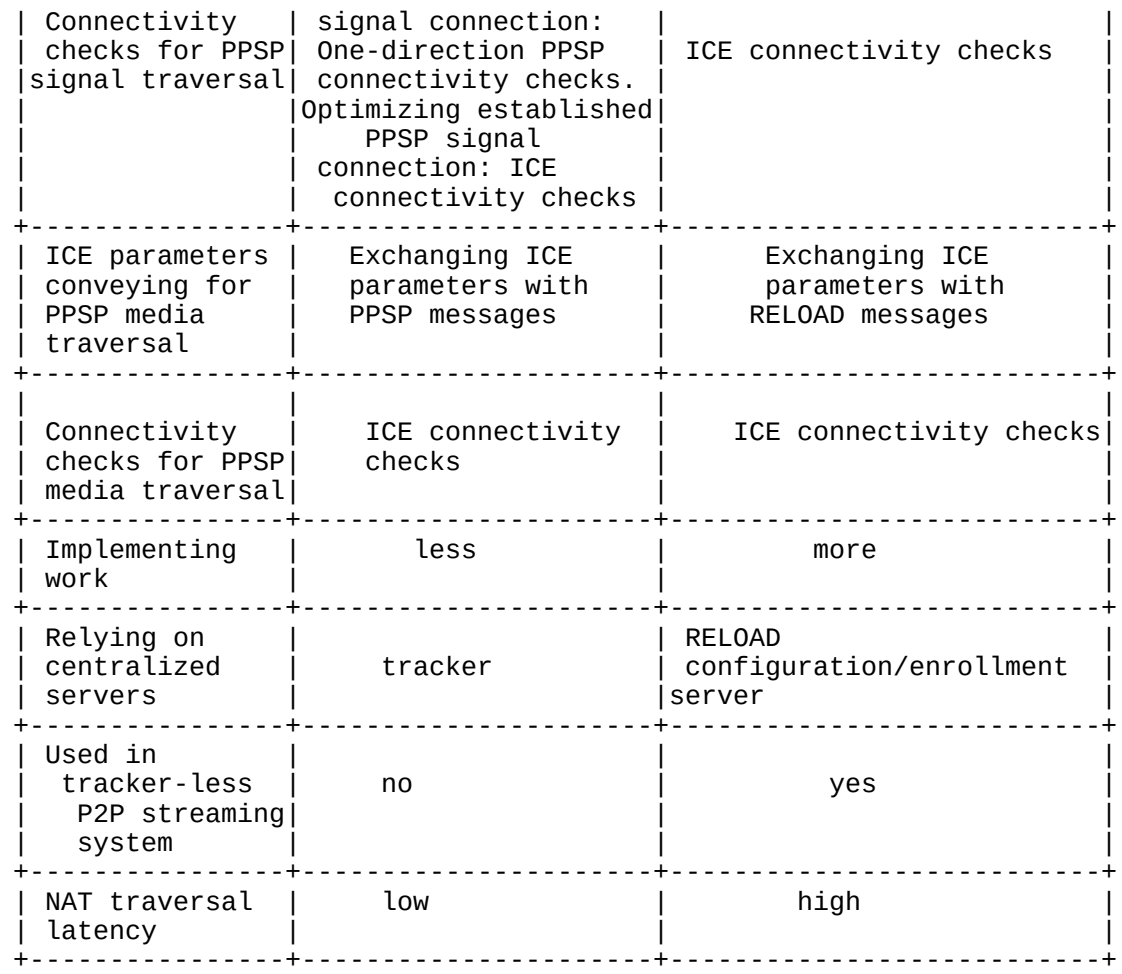
5.3. Solution Comparison

NAT traversal solutions in this document all use ICE. PPSP-ICE solution uses modified ICE or ICE-like method to establish PPSP signal connection, and optionally uses ICE to optimize connection path. PPSP-ICE solution uses ICE for PPSP media traversal of NAT. RELOAD solution uses ICE for both PPSP signal and media traversal of NAT.

Compared with RELOAD-ICE solution, PPSP-ICE solution increases tracker's workload. But the workload is acceptable considering tracker's work of maintaining and providing peer status and content location. Compared with PPSP-ICE solution, RELOAD-ICE solution requires much more time to traverse NAT, and is more complicated to implement. RELOAD solution can be used in both tracker-based and tracker-less P2P streaming systems.

The major differences between PPSP-ICE solution and RELOAD-ICE solution are listed in the table below.

	PPSP-ICE	RELOAD-ICE
Using proxy candidate or relayed candidate	Using proxy candidate in PPSP connectivity checks; using relayed candidate in ICE checks	Using relayed candidate
NAT traversal service discovery	All methods except RELOAD method	All methods
candidates conveying for PPSP signal traversal	Establishing PPSP signal connection: one-direction conveying candidates with PPSP messages. Optimizing established PPSP signal connection: exchanging ICE parameters with PPSP messages.	Exchanging ICE parameters with RELOAD messages
	Establishing PPSP	


Connectivity checks for PPSP signal traversal
signal connection: One-direction PPSP connectivity checks. Optimizing established PPSP signal connection: ICE connectivity checks
ICE connectivity checks
ICE parameters conveying for PPSP media traversal
Exchanging ICE parameters with PPSP messages
Exchanging ICE parameters with RELOAD messages
Connectivity checks for PPSP media traversal
ICE connectivity checks
ICE connectivity checks
Implementing work
less
more
Relying on centralized servers
tracker
RELOAD configuration/enrollment server
Used in tracker-less P2P streaming system
no
yes
NAT traversal latency
low
high

6. Decisions to Implement NAT Traversal

NAT traversal is not a mandatory requirement for PPSP operations, and if NAT traversal needs to be implemented there are several possible implementation options. The decision of supporting NAT traversal or not and choosing which NAT traversal solution should be left to implementation. If a NAT traversal solution is chosen, there are still decisions to make on using which NAT traversal method and NAT traversal service node.

These decisions could be made with following considerations.

- First, the success rate of connection. Unlike P2P VoIP service, P2P streaming service doesn't require that any two peers can establish connection. Instead, it only requires that the download of streaming media succeed and the download speed is satisfied.

- Second, NAT type and its ratio. For example, in an environment that all or most NATs are full cone NATs, a P2P streaming system only needs STUN/STUN-like method.

- Third, the implementation and maintenance overheads of NAT traversal solution/method/service. For example, a P2P streaming system may choose not to use RELOAD-ICE solution due to implementing overhead.

- Fourth, NAT traversal solution/method/service's performance, reliability, etc. For example, a P2P streaming system may choose not to use media relay or use media relay only as the last resort because media relay consumes much more resources on relay node.

7. PPSP Extension for NAT Traversal

To enable NAT traversal, this section proposes extension to the tracker protocol and peer protocol.

7.1. Tracker's STUN-like Function

This extension is optional. Tracker performs STUN-like function by putting peer's address it observes in CONNECT response.

If enabling STUN-like function, this draft proposes to extent tracker protocol by adding following tag in CONNECT response.

<ReflexiveAddr>

IP and port

</ReflexiveAddr>

7.2. Proxy Peer

This extension is mandatory for PPSP-ICE solution.

7.2.1. Relayed by Proxy

Proxy peer relaying messages requires PPSP messages between peers containing destination peer's ID. As shown below, a tag named "DestPeerID" containing the destination peer's ID can be used.

<DestPeerID>***</DestPeerID>

7.2.2. Connecting and Disconnecting Proxy

To receive messages via proxy peer, a NATed peer MUST connect to proxy peer. If the NATed leaves the PPSP network, it MUST disconnect from its proxy peer. CONNECT and DISTCONNECT methods can be reused to connect and disconnect proxy peer separately. The messages don't need to be changed except for adding DestPeerID in the messages.

7.3. STUN/TURN/proxy Ability Report and Querying STUN/TURN/proxy Peer List

This extension is mandatory for PPSP-ICE solution. PPSP tracker protocol already supports STUN/TURN ability report with STAT messages. To support proxy ability report, a STAT type name "proxy" can be added. The value of proxy STAT is Boolean value.

Peer can query STUN/TURN/proxy peer list from tracker or other peer

using extended FIND messages. The extension uses <Stat> tag to indicate the type of peer list, and removes <SwarmID> and <ChunkID> tags.

The method specific XML of the extended FIND message takes the form shown below:

```
<PeerID>***</PeerID>

<Peernum>***</Peernum>

<Stats>

<Stat property="STUN">true</Stat>

... more stats ...

</Stats>
```

The method specific XML of the extended FIND response takes the form shown below:

```
<Peers> Peer list </Peers>
```

7.4. Carrying Candidates in PPSP Message

This extension is mandatory for PPSP-ICE solution. A peer MAY have multiple IP addresses with different properties. This document proposes to extend the PeerAddresses tag defined by [I-D.cruz-ppsp-http-tracker-protocol]. An example of the extended PeerAddresses is shown below:

```
<PeerAddresses>

<PeerAddress ip="***" port="***" priority="***" type="host"/>

<PeerAddress ip="***" port="***" priority="***" type="reflexive"/>

<PeerAddress ip="***" port="***" priority="***" type="proxy"/>

</PeerAddresses>
```

To use PPSP-ICE solution, all PPSP messages that containing peer address MUST use the tag.

7.5. Exchanging ICE Parameters

This extension is mandatory for PPSP-ICE solution. This document proposes Attach and MediaAttach methods to exanging ICE parameters for building PPSP connection and media connection separately. These two methods have different method name, but share the same method body as shown below:

<DestPeerID>***</DestPeerID>

<PeerID>***</PeerID>

<SDP>

...

</SDP>

The ICE parameters are encoded in SDP.

8. Security Considerations

Todo: The content of this section need further input.

9. IANA Considerations

TBD

10. References

10.1. Normative References

[I-D.cruz-ppsp-http-tracker-protocol]
Cruz, Rui S., Nunes, Mario S., and Joao P. Taveira, "HTTP-based PPSP Tracker Protocol", draft-cruz-ppsp-http-tracker-protocol-01 (work in progress), June 2011.

[I-D.gu-ppsp-peer-protocol]
Gu, Y. and David A. Bryan, "Peer Protocol", draft-gu-ppsp-peer-protocol-02 (work in progress), June 2011.

[I-D.gu-ppsp-tracker-protocol]
Gu, Y., Bryan, David A., and L. Deng, "PPSP Tracker Protocol", draft-gu-ppsp-peer-protocol-04 (work in progress), May 2011.

[I-D.zhang-ppsp-problem-statement]
Zhang, Y., Zong, N., Camarillo, G., Seng, J., and R. Yang, "Problem Statement of P2P Streaming Protocol (PPSP)", draft-zhang-ppsp-problem-statement-06 (work in progress), July 2010.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

10.2. Informative References

[I-D.ietf-p2psip-base]
Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-12 work in progress, November 2010.

[I-D.ietf-p2psip-service-discovery]
Maenpaa, J. and G. Camarillo, "Service Discovery Usage for REsource LOcation And Discovery (RELOAD)", draft-maenpaa-p2psip-service-discovery-02 work in progress, January 2011.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5766] Matthews, P., Rosenberg, J., and R. Mahy, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC RFC5766, April 2010.

Authors' Addresses

Lichun Li
ZTE Corporation
RD Building 1,Zijinghua Road No.68
Yuhuatai District,Nanjing 210012
P.R.China

Phone:
Email: lilichun@gmail.com

Jun Wang
ZTE Corporation
RD Building 1,Zijinghua Road No.68
Yuhuatai District,Nanjing 210012
P.R.China

Phone:
Email: wang.jun17@zte.com.cn

Wei Chen
China Mobile
Unit 2, 28 Xuanwumenxi Ave, Xuanwu District,
Beijing 100053
P.R.China

Phone:
Email: chenweiyj@chinamobile.com

PPSP Group
Internet Draft
Intended status: Informational
Expires: March 21, 2011

G.Lu
JC.Zuniga
A.Rahman
InterDigital Communications, LLC

September 21, 2010

P2P Streaming for Mobile Nodes: Scenarios and Related Issues
draft-lu-ppsp-mobile-04.txt

Abstract

The scenarios where a Peer-to-Peer Streaming Protocol (PPSP) contains mobile nodes need special considerations. An analysis of all the scenarios that involve mobile nodes is necessary to provide the guidelines to PPSP protocol design and applicability. This document describes some key issues for a PPSP network with mobile nodes, and proposes some additional requirements for PPSP to handle these scenarios.

Table of Contents

1. Introduction

The PPSP Working Group is developing protocols for Peer-to-Peer (P2P) streaming systems [I-D.zong-ppsp-reqs]. In the past P2P solutions

have mostly targeted wired or fixed connections. Mobile P2P communications are expected to grow rapidly and the nature of mobile nodes and mobile environments cause specific challenges to P2P communications, specifically for streaming scenarios. This draft discusses some key mobility specific issues.

2. Conventions and Terminology

This document uses the same terminologies as [I-D.zong-ppsp-reqs]. For simplicity, this document illustrates scenarios showing a centralized Tracker architecture. However, it should be understood that all the scenarios also apply to the distributed architecture, e.g. using a Distributed Hash Table (DHT).

3. Mobile Node Issues

Mobile nodes are constrained by nature due to their limited battery, screen size, computational capability, etc. Also mobile nodes operate in variable and unpredictable environments. These attributes bring about the following problems that may adversely affect the P2P Streaming sessions.

3.1. Uplink vs. Downlink Bandwidth

Often mobile nodes have asymmetrical bandwidth capabilities. For instance, most mobile nodes are capable of handling higher bit rates in the downlink (to the mobile) than in the uplink (from the mobile). In addition, many mobile networks also have policies to assign bandwidth in this asymmetrical manner regardless of the capabilities of the mobile node. Since peer-to-peer streaming sessions can be either generated or terminated on a mobile node, this bandwidth asymmetry should be considered for the Tracker-Peer protocol (e.g. as part of Peer status parameters reported to the Tracker), and may also affect Peer-Peer protocol in the peer information negotiation.

3.2. Battery Power

By definition, a mobile node is often disconnected from the electrical grid and runs on its own battery power. In this

scenario, the user of the mobile node may want to restrict the types of P2P sessions that the mobile node should participate in because of battery drain issues. For example, the user may be willing to participate in a P2P session if the user herself is watching the content. However, the user may not want to participate in uploading large amounts of content to other peers.

Therefore, battery power (or battery status) of a mobile node should be considered in both the Peer-Peer and the Tracker-Peer protocols (e.g. as part of Peer status parameters reported to the Tracker and other peers).

3.3. Multiple Interfaces

Simple IP refers to the scenario where there is no IP layer mobility protocol such as Mobile IP or Proxy Mobile IP, and a peer needs to obtain a new IP address through a standard method like DHCP after losing the previous IP address.

As illustrated in Figure 1, when Peer 1 moves from AN1 to AN2, its IP address changes from IP1 to IP2. This will impact both the Peer-Peer connection and the Tracker-Peer connection. For example, Peer-Peer communication maybe lost (e.g. Peer 2 incorrectly sends chunks to IP1 even though Peer 1 has now changed address to IP2). Also the Tracker-Peer communication may be compromised (e.g. Tracker has corrupted Peer lists containing incorrect IP Address for Peer 1).

These effects may be somewhat mitigated by having the mobile node update the tracker and corresponding peers with its new IP address. The key question then is the trade-off between signaling required to provide notification of the IP address change and the load this causes on the system. Also race conditions must be carefully considered.

Therefore, reporting of change of the IP address of a mobile node should be considered in both the Peer-Peer and the Tracker-Peer protocols.

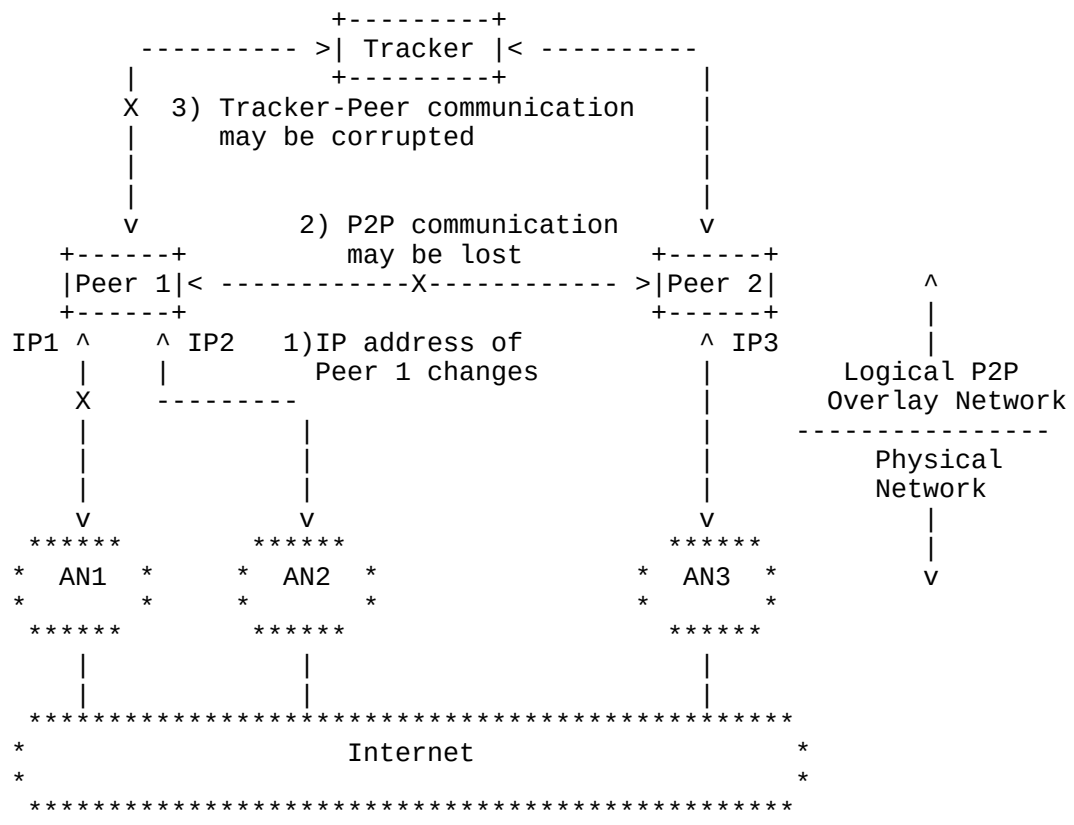


Figure 1 P2P Streaming with Device with Multiple Interfaces

3.4. Geo-Targeting

Geo-targeting is a technique used to determine the physical location (i.e. geo-location) of a user. The geo-location is based on geographical and other personal information provided by the requester peer or a third party. Techniques to determine geo-location of a user can rely on civic location, GPS geographical coordinates, cellular base station ID, or most commonly IP address. The primary source for IP address geographical data is the regional Internet registries.

Depending on the location, different regulations and rules may apply. For instance, some content may not be distributed on certain locations or can only be distributed on some other locations.

Current content distribution policies can apply certain rules to fixed P2P Streaming clients. However, device mobility may hide the peer movement from one region to another where possibly different content distribution rules may apply hence rendering the set forth policies un-enforceable. This may also be the case where the peer is connecting through a Virtual Private Network (VPN).

Therefore, geo-location reporting of a mobile node should be considered in both the Peer-Peer and the Tracker-Peer protocols.

4. Conclusion and Recommendations

The PPSP Working Group should consider the impacts of various aspects of mobility discussed in this draft. In particular, PPSP should consider how these issues can be mitigated in a mobile P2P streaming environment when designing both the PPSP Peer-Peer and the Tracker-Peer protocols. Therefore, it is recommended that the following requirements be added to the "Basic Requirements to PPSP Node" section of [I-D.zong-ppsp-reqs]:

PPSP.REQ-1: Change in IP address of a Peer device MUST immediately be reported via the Tracker Protocol and Peer Protocol

PPSP.REQ-2: Available uplink and downlink bandwidth of a Peer device MAY be reported via the Tracker Protocol and Peer Protocol

PPSP.REQ-3: Battery status of a Peer device SHOULD be reported via the Tracker Protocol and Peer Protocol

PPSP.REQ-4: Location of a Peer device SHOULD be reported via the Tracker Protocol and Peer Protocol

5. Security Considerations

This draft does not introduce new threats to security.

6. IANA Considerations

This document makes no request of IANA.

7. References

7.1. Normative References

[RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in Ipv6", RFC 3775, June 2004.

[RFC5213] Gundavelli, S., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, August 2008.

7.2. Informative References

[I-D.zong-ppsp-reqs]
Zong, N., Zhang, Y., Pascual, V., and C. Williams, "P2P Streaming Protocol (PPSP) Requirements", draft-zong-ppsp-reqs-04 (Work in progress), July 7, 2010.

[I-D.ietf-p2psip-base]
Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-10 (Work in progress), August 3, 2010.

8. Acknowledgments

The authors would like to thank Serhad Doken and Milan Patel for their thorough review and valuable inputs to this draft.

9. Appendix A - Other Mobility Considerations

This Appendix summarizes some other mobility considerations that were analyzed. However, these considerations are outside the scope of the current PPSP Working Group scope and thus are recorded here for purely informational purposes.

9.1. Processing Power

Some devices are more capable than others in terms of computational performance or processing power. Similarly, devices can have different performance for generating a session (e.g. video recording) or terminating it (e.g. video display). Taking these differences into account is important for maintaining a good quality of the P2P streaming session.

9.2. Link Layer Mobility

PPSP uses a P2P based overlay network on top of the transport network. Mobility or link quality at link layers is not visible to the peers.

As illustrated in Figure 1, if Peer 1 is connected to a poor quality link via mobile Access Network 1 (AN1), then the overall P2P streaming session quality can suffer from high error rate and low throughput due to poor link layer conditions. This will impact both the Peer-Peer connection and the Tracker-Peer connection. For example, on the Peer-Peer connection frame loss, audio/video synch loss, or streaming stalls are likely to be seen on the media transfer protocols.

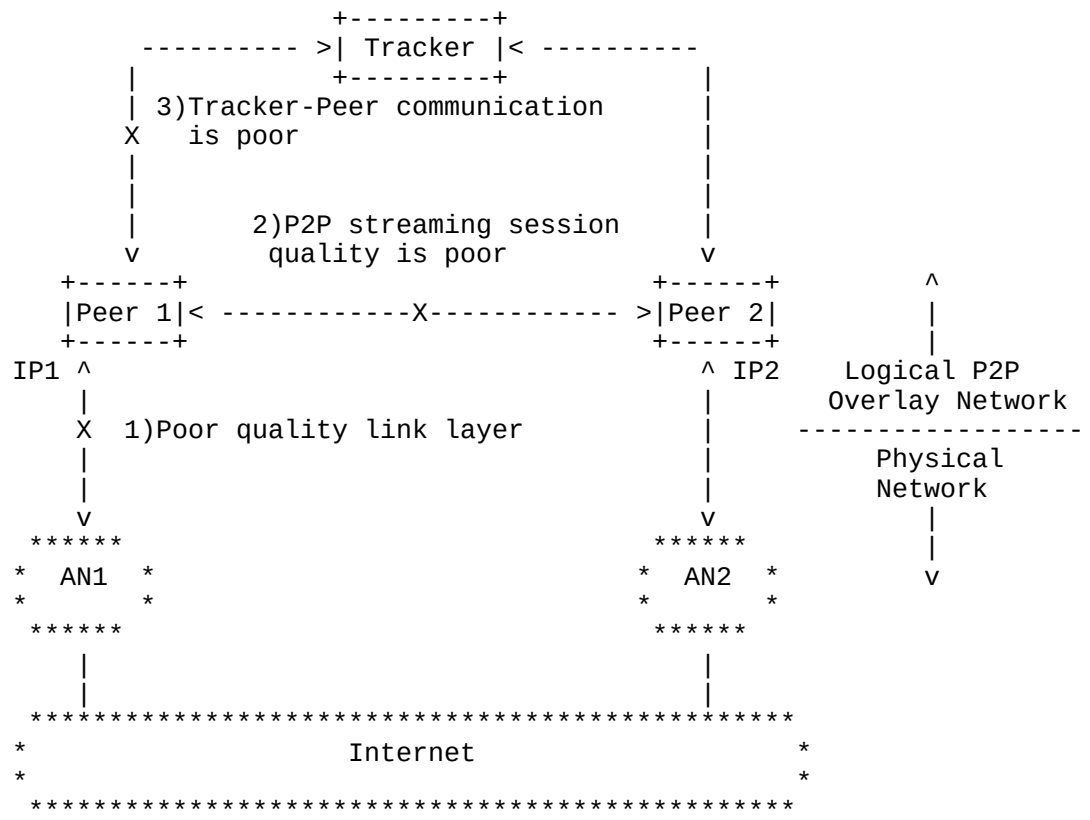


Figure 2 P2P Streaming with Link Layer Mobility

9.3. Mobile IP

Mobile IP (MIP) provides IP mobility and hides the mobile's movement from the Correspondent Node (CN) [RFC3775].

Figure 3 illustrates the case when Peer 1 moves from AN1 to AN1'. Because of Mobile IP, neither the Tracker nor Peer 2 are aware of the change of network for peer 1. However, due to the inherent tunneling and triangular routing of the Mobile IP protocol (through the Home Agent) the P2P session may in some scenarios experience extra latency. This may adversely affect the user experience of the P2P

streaming session. As seen above, Mobile IP will impact primarily the Peer-Peer connection (and the Tracker-Peer connection is not significantly affected).

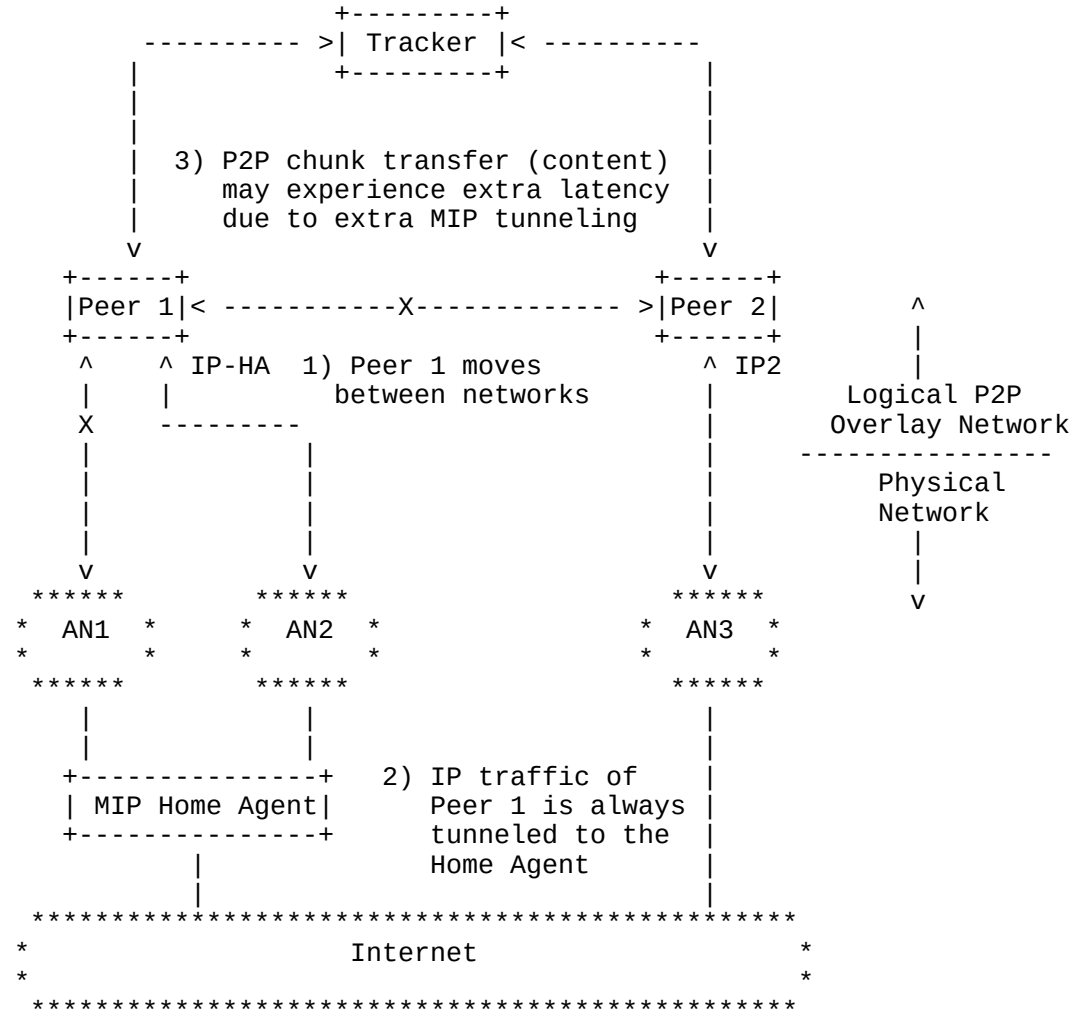


Figure 3 P2P Streaming with Mobile IP

9.4. Proxy Mobile IP

The use of Proxy Mobile IP [RFC5213] causes similar issues as the ones mentioned for Mobile IP in the above section. On top of these, Proxy Mobile IP also introduces a new issue for P2P streaming sessions. Since Proxy Mobile IP is a network based solution, the mobile node (peer) is not aware of its IP mobility so it cannot inform the Tracker, P2P Cache, CDNs or other peers of the IP level mobility. Therefore IP mobility is totally invisible to the P2P Streaming session entities and harder to detect and respond accordingly. Thus Proxy Mobile IP will impact both the Peer-Peer connection and the Tracker-Peer connection.

9.5. Mobility support with RELOAD

It has already been identified in the proposed WG charter that any PPSP developed protocol should be analyzed for interactions with the RELOAD protocol. The RELOAD protocol provides a signaling and routing mechanism for P2P overlay networks over the general Internet. The latest RELOAD draft [I-D.ietf-p2psip-base] also has a future consideration section for support of HIP (section 5.6.1.1). HIP is an experimental mobility protocol with good security properties.

In addition to HIP, the following mobility protocols should also be considered for PPSP-RELOAD interactions:

- Mobile IP
- Proxy Mobile IP

9.6. Tracker Mobility

Normally Trackers are assumed to be fixed nodes. However, in a mobile environment mobile nodes can also become Trackers. In this sense, similar considerations to the ones described above for mobile peers should be applied to mobile Trackers.

Authors' Addresses

Guang Lu
InterDigital Communications, LLC
Email: Guang.Lu@InterDigital.com

Juan Carlos Zuniga
InterDigital Communications, LLC
Email: JuanCarlos.Zuniga@InterDigital.com

Akbar Rahman
InterDigital Communications, LLC
Email: Akbar.Rahman@InterDigital.com

PPSP
Internet-Draft
Intended status: Informational
Expires: October 22, 2011

J. Seedorf
M. Stiemerling
NEC
M. Mellia
Politecnico di Torino
R. Lo Cigno
C. Kiraly
University of Trento

April 20, 2011

Design Considerations for a Peer-to-Peer Streaming Protocol
draft-seedorf-ppsp-design-considerations-02

Abstract

Streaming video on P2P overlays puts extremely high demands and stress on the underlying network, especially in case of TV and live streaming. The EU research project NAPA-WINE has devised an overall architecture for live video streaming that supports the needs of the users and content providers, while being respective of network-level needs, as reducing inter-AS traffic using ALTO-like services. In this document, we describe generic elements of this software architecture for P2P live streaming and derive the corresponding implications for standardizing a Peer-to-Peer streaming protocol.

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 October 22, 2011.

Copyright Notice

Table of Contents

1. Introduction

In recent years, Peer-to-peer (P2P) technology became increasingly popular for video streaming applications, including TV services (P2P-TV). Examples of commercial P2P-TV include SopCast, TVAnts, PPLive, UUSee, and TVUplayer. The interest of the research and standardization communities, content providers and network operators is also on the rise. Content providers see a novel opportunity to reach users, but at the same time they are concerned about the threats posed to their standard business models. Network operators are mainly worried by the stress posed by such a bandwidth-hungry and delay sensitive application on their infrastructure. The research community is stimulated by the opportunities offered by live P2P distribution and broadcasting, looking both for novel technical solutions and innovative business models.

NAPA-WINE (Network Aware P2P-TV Application over WIse NEtworks) is a three years project (STREP) within the 7-th Research Framework of the European Commission whose goal is finding innovative solutions for P2P live streaming to meet opportunities envisaged by content providers while soothing worries of network operators [refs.napawebpage].

In a P2P-TV system, a source divides the video stream into chunks of data, which are exchanged among nodes to distribute them to all participating peers. Peers form an overlay topology at the application layer, where neighbor peers are connected and exchange chunks using the underlying IP network. The IP and overlay layer are both "network" layers in that they both perform routing and forwarding of the data: packets in the IP layer and chunks (normally a sequence of packets) in the overlay. In this context, the NAPA-WINE project has developed an innovative, network cooperative P2P architecture that explicitly targets the optimization of the quality perceived by the users while minimizing the impact on the underlying transport network. Our architecture does not impose any structure on the overlay topology, which can be any type of generic mesh. The video distribution is chunk-based, but chunk construction is free enough to accommodate anything from a single video frame (even less if required) to large swaths of a video in case of nearly on-demand applications. The focus is on the design of a system empowering future P2P High Quality TV, a system where a source peer produces the video stream, chops it into chunks, and injects them in the overlay where peers cooperate to distribute them, without the need for the source to have enormous resources and bandwidth to support the service.

In this document, we summarize our architecture for P2P live streaming (a more detailed description can be found in

[refs.napa-architecture]). The goal of this draft is to derive the implications for standardizing a P2P-based streaming protocol from the aforemetioned architecture. We thus highlight key design considerations for a P2P-based streaming protocol which we believe are important input to the IETF PPSP working group.

An open-source implementation of this P2P live streaming architecture has been released, entitled "WINESTREAMER". The latest release of the WINESTREAMER software can be retrieved at [refs.winestreamer].

2. An Architecture for a P2P-based Live Streaming Application

The architecture we developed is based on five main building blocks:

- Application Layer
- Topology Management
- Chunk Scheduling
- Monitoring Layer
- Messaging Layer

In addition, our architecture contains an external ALTO interface that can support the topology management providing information that cannot be measured at the application level. In the following we briefly outline the role and key features of each building block.

2.1. Application Layer

The Application Layer (or User Layer) is mainly responsible for of video encoding and its packaging into chunks (at the source) and de-chunkization and decoding at receivers. Standard encoding tools like ffmpeg can be used by the User Layer, whose goal is not implementing a proprietary video encoder, but supporting as many as possible standard formats (MPEG1/2/4, H.264, etc.). Depending on the type of video source this may include analog/digital conversion, encoding and any other video manipulation that the content provider whishes to do, like advertising introduction and similar.

The standardization of the application layer is out-of-scope for IETF PPSP work other than considerations on how to express and transport metadata information about the content.

2.2. Topology Management

A P2P-TV client needs to communicate efficiently with other peers to receive and redistribute the huge amount of information embedded in a video stream. Information must arrive in nearly real-time and with small delay variation. The application goal is then to deliver all the video information to all peers in the system in the smallest possible amount of time. One of the key enabling factors is who are the peers to communicate with, i.e. the neighborhood selection.

The overlay network in P2P systems is the result of a distributed algorithm that builds and maintains the neighborhood at each peer. When a peer joins the system, it selects an initial set of neighbors,

then the set of neighbors of every node in the system is dynamically optimized over time. The bootstrapping phase can be helped by the source or a web server where the user selects a channel, which can behave as a bit-torrent like tracker. The maintenance of the topology is based on a gossiping protocol that enables discovery of peers in the overlay. Once peers are discovered, the optimal topology management is obtained through an topology manager which chooses which peers to connect to.

IMPLICATIONS ON STANDARDIZATION:

- o A tracker protocol is required that supports the goals of the topology management
- o The topology management algorithm can be standardized, but this would probably be beyond the scope of the PPSP WG
- o The definition of information about the employed topology management and the exchange as part of the PPSP WG can be standardized

2.3. Chunk Scheduling

Strictly related to topology management is the problem of chunk trading. The goal of chunk trading is receiving the stream smoothly (and with small delay) and to cooperate in the distribution procedure.

Chunks transferring is the operation that most affects performance and network friendliness. It includes protocol and algorithmic problems. First of all, peers need to exchange information about their current status to enable scheduling decisions. The information exchanged refers to the state of the peer with respect to the flow, i.e., a map of which chunks are needed by a peer to smoothly playout the stream. This task means i) sending buffer maps to other nodes with the proper timing, ii) receiving them from other nodes and merging the information in the local buffer map data base, iii) negotiating if this and other information should be spread by gossiping protocols or not, and to which depth it should spread in the topology.

Besides the buffer map exchange, the signaling includes Offer/ Request/Select primitives used to trade chunks. These messages can be piggybacked on chunks for efficiency. Another key protocol decision is about "Pushing" or "Pulling" information. A chunk is pushed when the peer owning the chunk decides to offer it to some other peer, while it is pulled when a peer needing the chunk requests it from another peer. From a theoretical point of view, as shown in

[refs.opt-scheduling], pushing is more effective. Regardless of the protocol and the signaling strategy used, the core of a scheduler is the algorithm to choose the chunks to be exchanged and the peers to communicate with. Many different strategies have been studied, including both fundamental algorithms and their adaptation to heterogeneous scenarios, multiple sub-streams etc.

IMPLICATIONS ON STANDARDIZATION:

- The PPSP protocol design should allow to operate either with a push or pull regime
- The PPSP protocol design should allow to select if push or pull is used in the PPSP system during runtime
- The PPSP protocol design should allow to employ multiple chunk scheduling algorithms with the same protocol

2.4. Monitoring Layer

Beside the information provided by e.g. an ALTO Server, both the chunk scheduler and the overlay manager can exploit timely information about the quality of the connectivity between peers collected in real time by monitoring modules. This includes, but is not limited to, the distance and the available bandwidth between two peers, or the presence of Network Address Translation (NAT). The Monitoring Layer may employ passive or active measurement. Passive measurements are performed by observing the messages that are exchanged between peers. Active measurements craft special probe messages which are sent to other peers at the discretion of the Monitoring Layer. Monitoring the network conditions is important for the peer-to-peer streaming application in order to judge the current state of the surrounding network environment

IMPLICATIONS ON STANDARDIZATION:

- The PPSP protocol should allow the exchange of monitoring status information (e.g., in the spirit of "Do you see what I see?" [refs.dywis])
- The PPSP protocol should support active and passive measurements between peers

2.5. Messaging Layer

The Messaging Layer offers primitives to other modules for sending and receiving data to/from other peers. It provides an abstract interface to transport protocols (UDP/TCP) and the corresponding

service access points offered by the operating system by extending their capabilities and providing an application level addressing, i.e., assigning a unique identifier to each peer. For example, it provides the ability to send a chunk to another peer, which has to be segmented and then reassembled to fit into UDP datagrams. The messaging layer also provides an interface to the monitoring module invoked for passive measurements: whenever a message is sent or received an indication will be given to the monitoring module, which can update its statistics. The last important feature of the messaging layer is mechanisms for the traversal of NAT boxes.

IMPLICATIONS ON STANDARDIZATION:

- The PPSP protocol should allow to negotiate or select different transport protocols, e.g., between plain TCP and LEDBAT.

- The PPSP protocol or framework should support peers in NAT traversal.

2.6. Interaction with ALTO

Application Layer Traffic Information (ALTO) [refs.alto] is an important means for improving the resource provider selection in P2P applications. The goal of an ALTO service is to provide applications with useful information (e.g. for P2P neighbor selection) which these applications cannot measure themselves [RFC5693]. Simulations have shown that the usage of information provided by an ALTO service can reduce operational costs associated with the transmission of P2P live streaming traffic [refs.etm2010]. The topology management in the NAPA-WINE architecture therefore fully supports ALTO guidance through the integration of an ALTO client within the topology manager. Further, the information retrieved via the integrated ALTO client can be used in neighbor selection, i.e. peers select the links to other peers in their neighbor list (partially) based on ALTO information.

IMPLICATIONS ON STANDARDIZATION:

- The PPSP protocol should allow peers to interact with an ALTO server and to retrieve ALTO information.

- The PPSP protocol should enable the use of ALTO information in peer selection.

3. Summary of Considerations for PPSP Protocol Design

In this section we summarize the design considerations we derived from our experience in designing a P2P live streaming system and which we motivated in the previous section.

DESIGN CONSIDERATIONS FOR A PPSP PROTOCOL:

- A tracker protocol is required that supports the goals of the topology management
- The topology management algorithm can be standardized, but this would probably be beyond the scope of the PPSP WG
- The definition of information about the employed topology management and the exchange as part of the PPSP WG can be standardized
- The PPSP protocol design should allow to operate either with a push or pull regime
- The PPSP protocol design should allow to select if push or pull is used in the PPSP system during runtime
- The PPSP protocol design should allow to employ multiple chunk scheduling algorithms with the same protocol
- The PPSP protocol should allow the exchange of monitoring status information (e.g., in the spirit of "Do you see what I see?" [refs.dywis])
- The PPSP protocol should support active and passive measurements between peers
- The PPSP protocol should allow to negotiate or select different transport protocols, e.g., between plain TCP and LEDBAT.
- The PPSP protocol or framework should support peers in NAT traversal.
- The PPSP protocol should allow peers to interact with an ALTO server and to retrieve ALTO information.
- The PPSP protocol should enable the use of ALTO information in peer selection.

4. Conclusions

Video Streaming applications exploiting the P2P communication paradigm are a commercial reality, but their overall architecture is still biased by file-sharing applications and they operate without any coordination with the IP network, often resulting in poor, even wasteful resource usage. This will prevent them to support High Quality TV in the future, or to make the transition to High Definition, which will require several Mbit/s per peer.

This document presented the NAPA-WINE architecture for a P2P-TV system, which has been developed with the goal of efficiency and cooperation between the application and both the network operators and the content providers. Prototypes of the peers and system are already running on the Internet and are demonstrated at major venues [refs.demo-iptcomm2010] [refs.demo-p2p2010] [refs.demo-infocom2011].

In addition, an open-source implementation of the P2P live streaming architecture presented in this document has been released, entitled "WINESTREAMER". The latest release of the WINESTREAMER software can be retrieved at [refs.winestreamer].

The overlay topology management and the chunk scheduling of information have been identified as important features for the application to be network-friendly. The first function enables building efficient and rational overlay topologies that are correctly mapped on top of the transport network structure (e.g., considering minimal number of hops between neighbors, locality w.r.t. Autonomous Systems, etc.). The second function guarantees that the network capacity is exploited without waste (e.g., by minimizing retransmissions and pursuing an efficient distribution of chunks, etc.).

Based on our experience in designing and implementing a P2P live streaming system, we highlighted key implications for standardization. We believe that these key design considerations we derived based on our architecture will be important input to the IETF PPSP working group for standardizing a P2P live streaming protocol.

5. Security Considerations

Security considerations will be detailed in future versions of this draft.

6. Acknowledgements

The authors would like to thank all the people participating in NAPA-WINE and contributing to the project success with their work and research.

Jan Seedorf, Marco Mellia, Renato Lo Cigno, and Csaba Kiraly are partially supported by the NAPA-WINE project (Network-Aware P2P-TV Application over Wise Networks, http://www.napa-wine.org), a research project supported by the European Commission under its 7th Framework Program (contract no. 214412). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NAPA-WINE project or the European Commission.

Martin Stiemerling is partially supported by the COAST project (COntent Aware Searching, retrieval and sTreaming, http://www.coast-fp7.eu), a research project supported by the European Commission under its 7th Framework Program (contract no. 248036). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the COAST project or the European Commission.

7. Informative References

[refs.napa-architecture]
Birke, R., Leonardi, E., Mellia, M., Bakay, A., Szemethy, T., Kiraly, C., Lo Cigno, R., Mathieu, F., Muscariello, L., Niccolini, S., Seedorf, J., and G. Tropea, "Architecture of a Network-Aware P2P-TV Application: the NAPA-WINE Approach", IEEE Communication Magazine, to appear.

[refs.opt-scheduling]
Abeni, L., Kiraly, C., and R. Lo Cigno, "On the Optimal Scheduling of Streaming Applications in Unstructured Meshes", IFIP Networking 2009.

[refs.demo-iptcomm2010]
Seedorf, J., Niccolini, S., Lo Cigno, R., and C. Kiraly, "Prototypical Implementation of ALTO Client and ALTO Server and Integration into a P2P Live Streaming Software", Demonstration, IPTComm 2010.

[refs.demo-p2p2010]
Abeni, L., Bakay, A., Biazzini, M., Birke, R., Leonardi, E., Lo Cigno, R., Kiraly, C., Mellia, M., Niccolini, S., Seedorf, J., Szemethy, T., and G. Tropea, "Network Friendly P2P-TV: The Napa-Wine Approach", Live Demonstration and Extended Abstract, IEEE P2P 2010.

[refs.demo-infocom2011]
Abeni, L., Bakay, A., Birke, R., Birke, R., Leonardi, E., Lo Cigno, R., Kiraly, C., Mellia, M., Niccolini, S., Seedorf, J., Szemethy, T., and G. Tropea, "WineStreamer(s): Flexible P2P-TV Streaming Applications", Live Demonstration and Extended Abstract, IEEE INFOCOM 2011.

[refs.dywis]
Miao, X., Schulzrinne, H., Singh, V., and Q. Deng, "Distributed Self Fault-Diagnosis for SIP Multimedia Applications", Springer Real-Time Mobile Multimedia Services, 2007.

[refs.etm2010]
Seedorf, J., Niccolini, S., Stiemerling, M., Ferranti, E., and R. Winter, "Quantifying Operational Cost-Savings through ALTO-Guidance for P2P Live Streaming", 3rd Workshop on Economic Traffic Management (ETM 2010), LNCS 6236.

[refs.alto]
Peterson, J., Gurbani, V., Marocco, E., and et. al., "IETF ALTO WG charter", https://datatracker.ietf.org/wg/alto/charter/.

[refs.winestreamer]
"Napa-Wine Winestreamer latest release", http://www.napa-wine.eu/cgi-bin/twiki/view/Public/NapaWineShowRoom.

[refs.napawebpage]
"Napa-Wine Project Website", http://www.napa-wine.eu.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.

Authors' Addresses

Jan Seedorf
NEC Laboratories Europe, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Phone: +49 (0) 6221 4342 221
Email: jan.seedorf@neclab.eu
URI: http://www.nw.neclab.eu

Martin Stiemerling
NEC Laboratories Europe / University of Goettingen
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Phone: +49 (0) 6221 4342 113
Email: martin.stiemerling@neclab.eu
URI: http://ietf.stiemerling.org

Marco Mellia
Politecnico di Torino, Italy
Corso Duca degli Abruzzi 24
Torino 10129
ITALY

Phone:
Email: mellia@tlc.polito.it
URI:

Renato Lo Cigno
University of Trento
Via Sommarive 14
Povo 38123
ITALY

Phone:
Email: locigno@disi.unitn.it
URI:

Csaba Kiraly
University of Trento
Via Sommarive 14
Povo 38123
ITALY

Phone:
Email: kiraly@disi.unitn.it
URI:

PPSP
Internet Draft
Intended status: Informational
Expires: April 2011

Xin.Wang
Fudan University
Jin.Zhao
Fudan University
Ming.Rong
Fudan University
Linjie.Yu
Fudan University
Shihui Duan
China CATR

October 25, 2010

Implementation of a P2P-VoD System with video-segmentation and Network Coding
draft-wang-ppsp-vod-system-01.txt

Status of this Memo

Copyright Notice

Abstract

This document introduces a practical work for a Peer-to-Peer (P2P) VoD system with video segmentation (VS) and network coding (NC). In this document, it introduces some key problem of P2P-VoD system. Then it gives a brief introduction on our P2P-VoD system and especially discusses some key technologies.

Table of Contents

1. Introduction

P2P-VoD System with VS and NC [1][2] is a P2P based VoD system which uses network coding and video segmentation to promote user experience when watching a video.

In this system, the granularity of random seek is the scene information in a video. And scenes of a video are defined by video segmentation which can be automatically or manually obtained. Thus, the less but efficient seek point can reduce bandwidth consumption and workload of the whole P2P network. Furthermore, the utilization of network coding is a great help to improve the efficiency of content distribution [3]. Introducing network coding-based distribution scheme enables us to shorten buffering time before users start to view the clips. Meanwhile, it offers better adaptability for peer churn when they dynamically join or leave the system.

2. Overview of P2P-VoD System with VS and NC

In this document, P2P-VoD system is a VoD system based on P2P overlay, which uses network coding and video segmentation to promote user experience when watching a video.

2.1. System Architecture

According to the requirement of PPSP peer protocol and PPSP tracker protocol, our system takes the similar strategy. Otherwise, NC simplifies the scheduling of data transfer between peers. It also introduces a scene server to offer scene information. The architecture of P2P-VoD System with VS and NC is shown as figure 1.

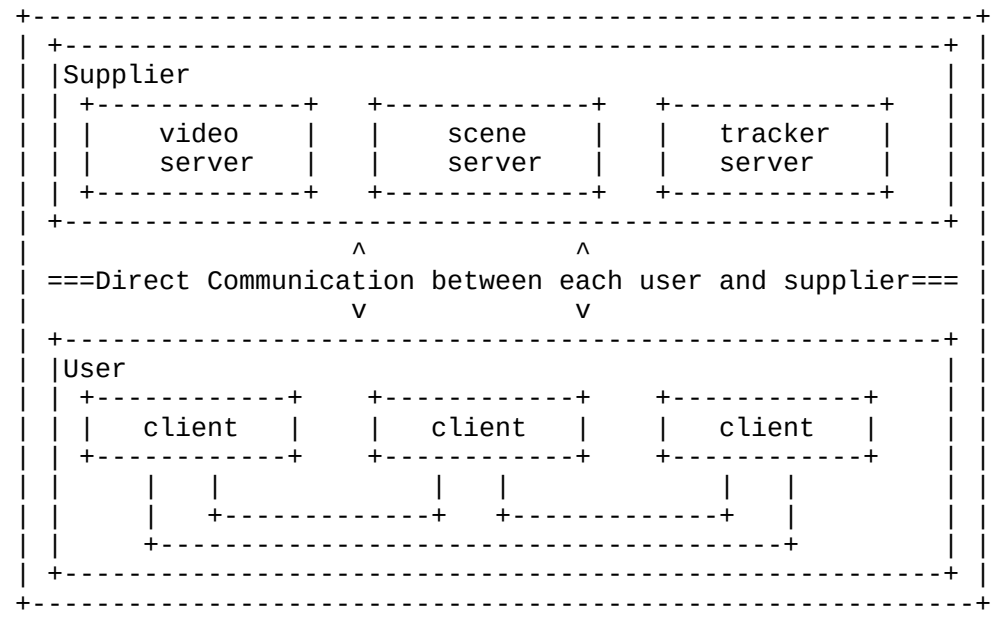


Figure 1 The architecture of P2P-VoD System with VS and NC

Our system has implemented the necessary functions required by PPSP protocol and some additional functions to make the system more practicability.

Video server: It supplies the original video stream. Once, enough clients request for the video data, the introduction of P2P will reduce the pressure of the video server on response to the client.

Tracker server: It holds the list of the owners who has the full video or part of the video. Once a user asks for a video, it will request tracker server for peer list.

Scene server: It provides the scene information when client chooses a video. The structure of the scene information will be mentioned in the following part.

Client: It let users have an extra functionality of random seek by scene information. Thus, when users want to skip, there are references. Therefore, unnecessary seek events can be avoided.

2.2. Some definitions and Key technologies

2.2.1. The structure of Scene information

P2P-VoD System with VS and NC offers the functionality of random seek by scene. As a result, the structure of scene information is defined below and the operation of a user is well supported by this kind of information.

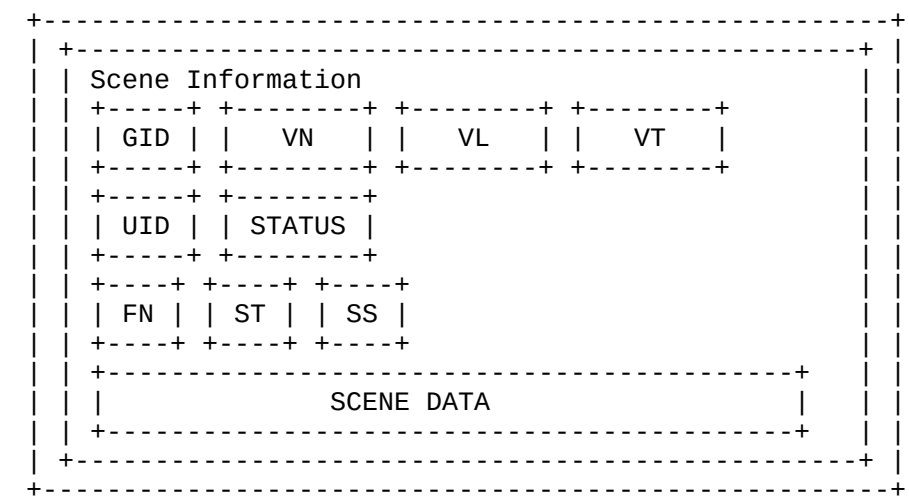


Figure 2 The structure of Scene information

GID It’s the global ID of a video. We use this ID to differ the video at server.

VN: It’s the name of a video.

VL: It’s the file length of a video.

VT: It’s the time length of a video.

UID: It’s the owner of a video. Sometimes, a user can publish a video so that the video may not stored by server.

STATUS: It’s the type of the video the user owns. It can be PART or FULL which means user has part of the video or full video.

FN: It’s the key-frame number of a scene.

ST: It’s the start time of a scene within a video.

SS: It’s the size of a scene picture which includes the height and width information.

SCENE DATA: It’s the data of a scene picture.

2.2.2. Structure of video-data buffer

A dynamic way is used in the buffer strategy. When a video is requested, a header is first been allocated instead of traditional way of allocating the same size of a video. Then, new coming video-data is appended to the previous video-data and the block offset is recorded in the header. Therefore, user will not pay for the non-coming video data.

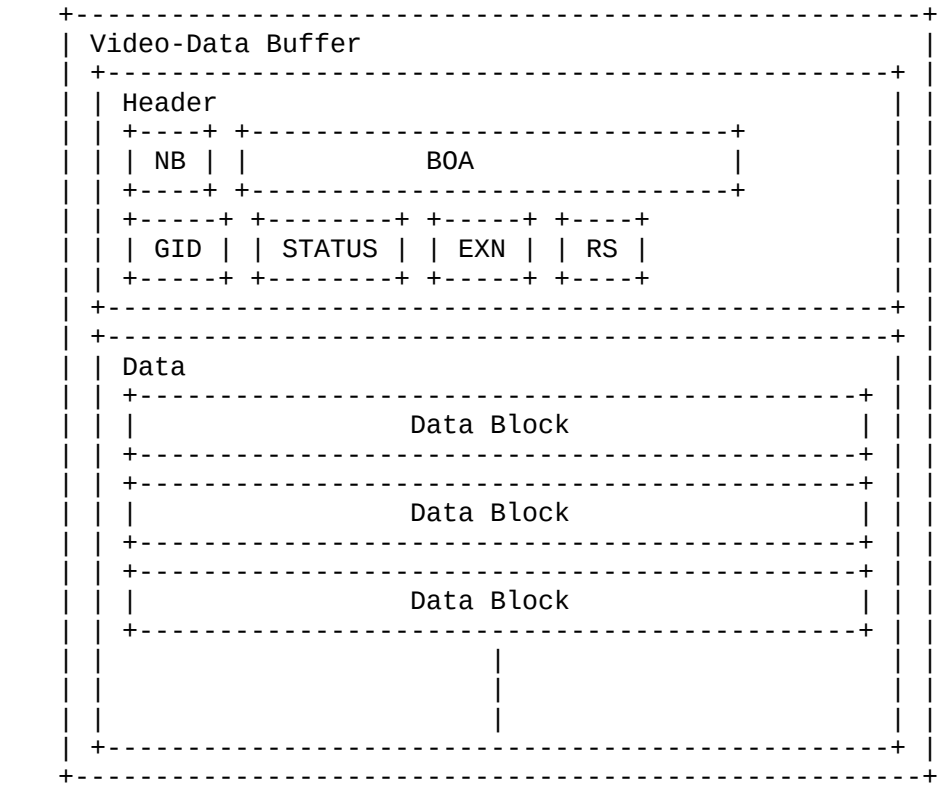


Figure 3 The structure of video-data buffer

NB: it’s the total count of data-blocks.

BOA: it’s the array of block offset. The length depends on the practical video size and the block size.

GID: it’s the global ID of the video file.

STATUS: reserved.

EXN: it’s the extension of the video file.

RS: it’s the reserved space for extension use.

Data Block: it’s the video data divided by block size.

2.2.3. Network Coding

Network coding (NC) is introduced into this system. Therefore, every peer sends and receives both video-clip and scene information data after filtered by the network coding codec module.

One block of data encoded by the module has some intersect information over the generation the block belongs to. In this way, each block within a generation acts equally to others and thus a method of Push-Based distributing mechanism is introduced which entitles a client with many downstream neighbors serving him concurrently. While some file-share P2P system requires a long time to do off-line decoding after the total download of the file, decoding of the video-clip data must be finished before it is played so as to sustain the playback rate. And a suitable size of generation should be determined to make the decoding-task more conveniently.

The advantage of network coding is to make every received block data useful, which takes the full advance of the bandwidth. It allows a more effective way to use the bandwidth and shows a more robust resilience to peer churn.

3. Validation of P2P-VoD System with VS and NC

We do some measurement and experiment to validate our system and its key technology.

In our experiment, we play a 20 minutes video-clip and the playback rate is set to 10 blocks (each block is 1K). We set Network coding operation within 40 blocks. Every peer has an average of five peers for both downstream and upstream. And the initial buffer time is 5 seconds. Then, we compare the system with NC or without NC.

3.1. Normal performance

In this experiment, we set the initial buffer time and set non-packet-drop. 200 peers sequentially join the system in an interval of five seconds. No peer joins or leaves the system during the process.

In the case of using NC or not using NC, We compare the average of 200 peer’s playback error / total playback time as the playback performance under 1.1 times, 1.5 times, 2.0 times, 2.5 times and 3.0 times bandwidth. And the basic bandwidth is equal to the playback rate.

Playback performance

Bandwidth:	1.1	1.5	2.0	2.5	3.0
With NC:	47%	29%	10%	2%	0%
Without NC:	43%	30%	19%	23%	10%

Re-request to the tracker is introduced in the system with NC. And it’s easily occurred in the situation of limited bandwidth. Therefore, our system leads to a worse result compared with non-NC system when the bandwidth is almost equal to the playback rate.

When the bandwidth is raised to the 1.5 times of playback rate, playback performance of the system with NC is much better. This is because the bandwidth is almost fully used to the most emergency.

The non-monotone phenomenon in the system without NC is because the peer selection algorithm is random pick. Therefore, the sequentially join of the peer may cause the much different performance among the 200 peers. Thus, the result of this experiment is not monotonic.

3.2. Performance with packet-drop

In this experiment, we introduce the packet-drop rate of 5% to make our results more practical. Generally, when a block is lost, the peer has to wait until next time slot to request the missing block. But with NC, a useful block is decoded by several encoded blocks. Therefore, one missing encoded block will not impact the performance in large extent.

Playback performance with packet drop

Bandwidth:	1.5	2.0	2.5	3.0

With NC:	28%	10%	0%	0%
Without NC:	33%	30%	12%	9%

The result of this experiment shows that in a more practical situation, our system behaves much better than the system without NC.

3.3. Performance under peer churn

In the actual environment, peers leave and join the system rapidly. To simulate this situation, we let over half of all the peers (around 100 peers) join the data distributing in sequence, and then inform peers to leave or join using random distribution.

Playback performance under peer churning

Bandwidth:	1.1	1.5	2.0	2.5	3.0
With NC:	45%	22%	10%	0%	0%
Without NC:	41%	37%	20%	17%	21%

Under the condition of peer churning, both systems' performance is declined, but system with NC is still better. The main reason is that in NC scheme, one block is served by several peers. Therefore, one peer's leaving will not affect the playback performance much.

4. PPSP Compatibility Considerations

The objective of the PPSP work is to standardize the key signaling protocols among various P2P streaming system components including the tracker and the peers.

Network coding can be well combined with the PPSP peer protocol. It uses buffermap to exchange the data availability and allows peers to exchange their property.

Using video-segmentation also promotes the performance of PPSP protocol since it can promote the VCR playback lag when a VCR operation occurs.

5. Security Considerations

This draft does not introduce any new security issues.

6. IANA Considerations

This memo includes no request to IANA.

7. Conclusions

Through the use of the P2P-VoD system with video segmentation and network coding, user will have a better experience when watching a video. Otherwise, the wasted bandwidth is reduced due to more precision seek operation. And the bandwidth is well used due to the introduction of network coding.

8. References

8.1. Normative References

8.2. Informative References

[1] M. Wang and B. Li, "Lava: A Reality Check of Network Coding in Peer-to-Peer Live St reaming," in Proc. of IEEE INFOCOM 2007, May 2007.

[2] Lingjie Yu, Linxiang Gao, Jin Zhao and Xin Wang, SonicVoD: A VCR-Supported P2P-VoD System with Network Coding, Consumer Electronics, IEEE Transactions, May 2009.

[3] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet : A Data-driven Overlay Network for Efficient Live Media St reaming," in Proc. of IEEE INFOCOM, 2005.

Authors' Addresses

Xin Wang
Fudan University
Shanghai 201203, China
Phone: 86-21-51355526
Email: xinw@fudan.edu.cn

Jin Zhao
Fudan University
Shanghai 201203, China
Phone: 86-21-51355526
Email: jzhao@fudan.edu.cn

Ming Rong
Fudan University
Shanghai 201203, China
Phone: 86-21-51355526
Email: 09210240072@fudan.edu.cn

Linjie Yu
Fudan University
Shanghai 201203, China
Phone: 86-21-51355526
Email: 082024067@fudan.edu.cn

Shihui Duan
CATR
Beijing 100045, China
Phone: 86-10-62300068
Email: duanshihui@mail.ritt.com.cn

PPSP
Internet Draft
Intended status: BCP
Expires: April 2011

K. Wu
Z. Lei
D. Chiu
ASTRI

October 18, 2010

P2P Layered Streaming for Heterogeneous Networks in PPSP
draft-wu-ppsp-p2p-layered-streaming-01.txt

Status of this Memo

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on April 18, 2009.

Abstract

The scenarios where a Peer-to-Peer Streaming Protocol (PPSP) contains heterogeneous nodes need special considerations. For example, mobile devices, PCs and set-top boxes may all need to access and provide service for the same content. To support heterogeneity and maximize the total network utilization, it may be necessary to have layered coding of content to achieve desired results. This document defines the Layered P2P Streaming in PPSP with support for heterogeneous peer nodes.

Table of Contents

1. Introduction

Single layer video streaming cannot satisfy heterogeneous customer requirement and heterogeneous download capacity. There are existing systems that use multiple versions of video content (each encoded at different resolution or visual quality) to minimize the overall transmission cost. For example, lower resolution video can be sent to

mobile devices while higher resolution or high quality video is sent to PC or STB players. However, in the P2P network, if there are too many independent video data transmitting, the users' inbound and outbound bandwidth will not be efficiently used [3]. Peers in different versions will not help each other. The overall video quality received will not be optimal.

The basic idea of layered encoding is that a video sequence is divided into multiple non-overlapped bit streams, or layers. The base layer contains the basic data representing the most important features of the video [2]. Additional layers, called enhancement layers, contain data that progressively refine the reconstructed video quality. According to the available bandwidth, participating nodes subscribe a subset of the layers to reconstruct the video with certain quality degradation [2].

Layered video has the advantage of bandwidth efficiency and at the same time meets the real-time streaming requirement of peer clients with wide range of variation in processing power, display capability and network conditions. In other words, although heterogeneity exists for peers in the P2P network, an optimal viewing experience can be achieved for each peer based on its own access bandwidth and capabilities.

An important requirement for Layered P2P streaming is that both base layer and enhancement layers can be decoded by standard compliant single layer decoders that have already been widely adopted, e.g. H.264/AVC, MPEG-4, etc. It requires minimum codec modifications because for many peer clients (e.g. STB or HMC), its decoder is hard-coded into the IC and cannot be changed at all.

Section 2 lists the terminology used.

In Section 3 we define the general architecture.

Section 4 defines the Layered P2P streaming control message flows.

Section 5 discusses the open issues.

Section 6 discusses the deployment issues.

Section 7 gives out the protocol details by expanding the PPSP streaming protocol to support the layered streaming.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [1].

Multi-layer P2P Streaming: multiple bit streams (layers) of same media content co-exist in the same P2P streaming group/session; each layer corresponds to a different media quality level; peers can choose different layers for decode and playback;

Base Layer (BL): unique Base Layer (required by all upper level layers for proper decoding);

Enhancement Layer (EL_i): i_th enhancement layer;

Heterogeneous peers: peers have different processing power, access bandwidth, or display constraints;

Low capacity peers, with low processing power, low access bandwidth or low display capability, e.g. mobile devices.

High capacity peers, with high processing power, high access bandwidth or high definition display, e.g. media center PC, or HD Set Top Box.

Layered coding: coding scheme that produces multiple layers of media streams for the same content. A higher layer can be decoded only if all the lower layers are available. Generally, layered coding does not lose much video coding efficiency compared with single layer coding streaming.

Decode ability: compatibility of the media bit stream that can be correctly decoded by the standard compliant decoders, e.g. H.264/AVC or MPEG-4.

Layer Information: It keeps the layer number and the information for each layer. For example, the information contains bitrate, resolution, frame rate, QP factor, etc. A peer can select the suitable layer according to its hardware configuration or network situation.

3. Architecture

Layered P2P can be considered a generalized version of single layer P2P. We describe how different components in single layer P2P need to be extended. Same as in single layer P2P, there is a meta file. In layered P2P, the meta file contains a layered content description.

A peer doing layered P2P has an important module that determines the number of layers it will subscribe to. This module may take manual input (e.g. the user decides to get base layer only, or to get HD no matter what). If there is no manual input, the module selects an appropriate number of layers to get, based on network conditions.

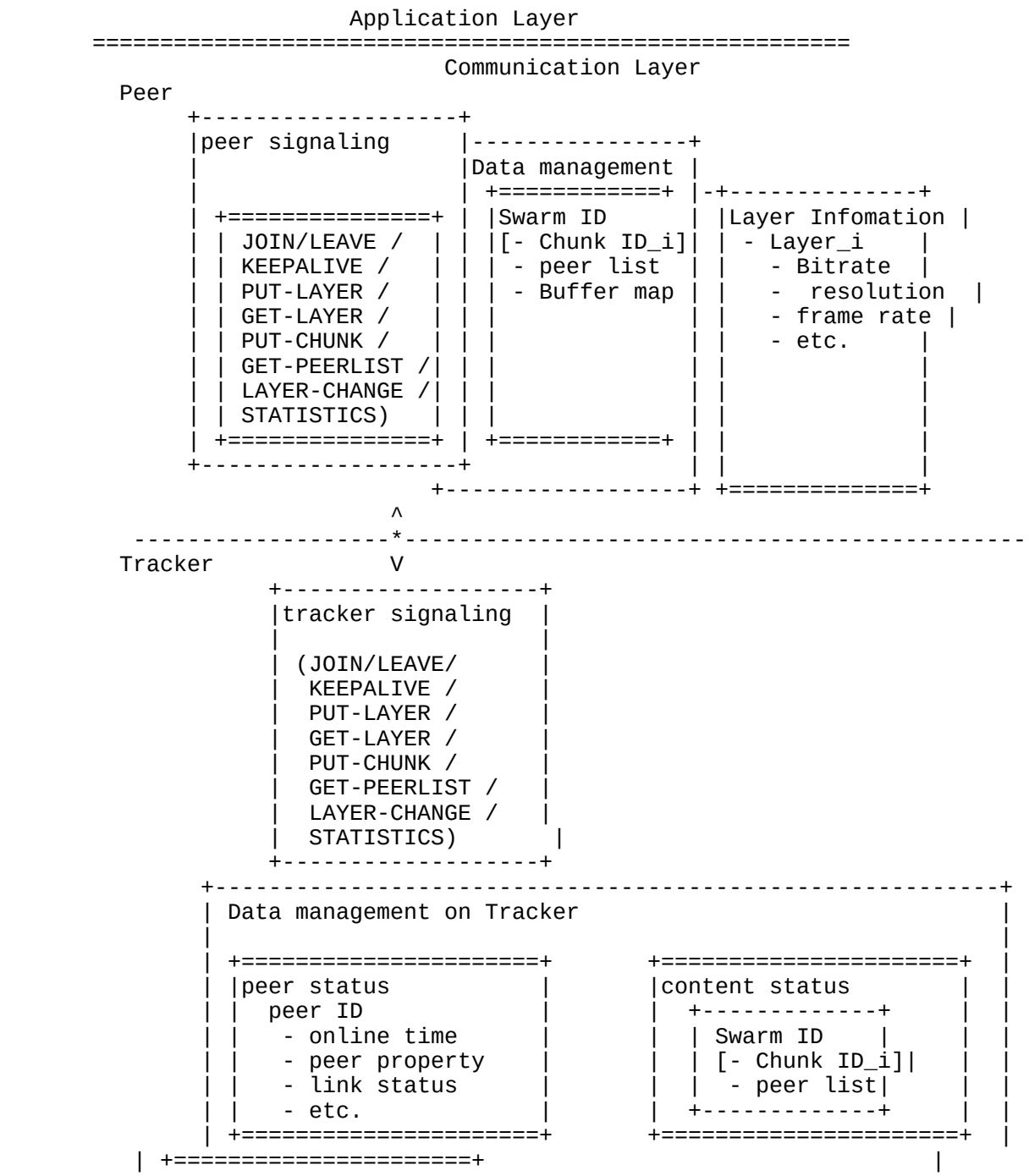
The peer-tracker protocol should allow a peer to indicate which layers are of interest. The tracker will then take that into consideration in returning peer-list. Optionally, the peers in peer-list may come with layer information. The management/status reports from peer to tracker should also indicate layer information about the local peer.

Each peer doing layered P2P has a more complicated scheduling module, figuring out which chunks in which layer have higher priority to get. There is no need to standardize the scheduling algorithm. The performance metrics related to this module (e.g. throughput, delay, layer completion ratio, waste ratio etc.) are discussed in later sections.

The peer-to-peer protocol for signaling (exchanging bitmap information) needs to have an extension to describe layered bitmaps and related data structures. The peer-to-peer data plane protocol can be almost the same as the single-layer case; the only extension is to add indicator for which layer the requested chunk belongs to.

3.1. Function Entities and Interfaces

Layered P2P architecture is extended from the single layer P2P streaming (PPSP) with modification of adding layer_i to meet the layered streaming requirement. Layer_i is the No. of active layer.


Application Layer
Communication Layer
Peer
peer signaling
JOIN/LEAVE /
KEEPALIVE /
PUT-LAYER /
GET-LAYER /
PUT-CHUNK /
GET-PEERLIST /
LAYER-CHANGE /
STATISTICS)
Data management
Swarm ID
[- Chunk ID_i]
- peer list
- Buffer map
Layer Infomation
- Layer_i
- Bitrate
- resolution
- frame rate
- etc.
Tracker
tracker signaling
(JOIN/LEAVE/
KEEPALIVE /
PUT-LAYER /
GET-LAYER /
PUT-CHUNK /
GET-PEERLIST /
LAYER-CHANGE /
STATISTICS)
Data management on Tracker
peer status
peer ID
- online time
- peer property
- link status
- etc.
content status
Swarm ID
[- Chunk ID_i]
- peer list

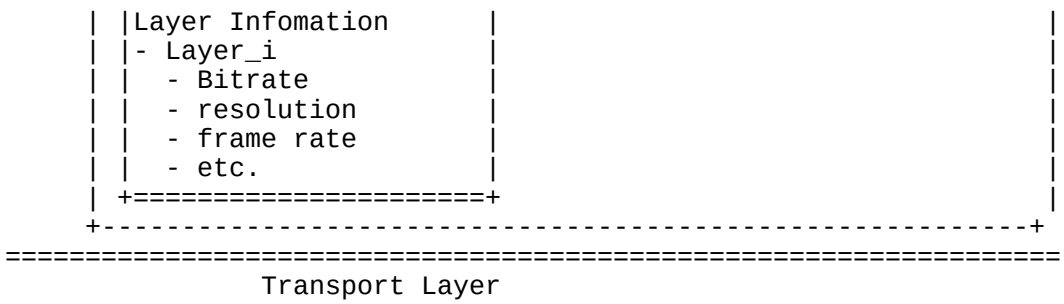


Fig 1 Function Entities of PPSP Tracker Communication

Different from the single layer P2P streaming, peer and tracker need to keep the layer information. The following components are considered:

1) Transmission of PUT-LAYER messages, by which source tells trackers the layer information the streaming is using.

2) Transmission of GET-LAYER messages, by which peers request what they want and get the layer information from trackers.

3) Transmission of PUT-CHUNK messages, by which peers tell trackers what they have. The bitfield can represent chunk_i.

4) Transmission of GET-PEERLIST messages, by which peers request what they want and get candidate peers list from trackers.

5) Transmission of STATISTICS requests and responses, by which trackers can get peers status, network performance, layer_i, etc.

3.2. Layered Dependency

The upper layers depend on the lower layers. An incomplete lower layer renders all chunks of upper layers useless. Hence, chunks of lower layers always carry high priority than chunks of higher layers. In reality, missing data (or delay of chunk arriving) is impossible to avoid. Certain error handling (e.g. error resilient decode or error concealment) mechanism in the players are needed to make sure errors in one layer won’t propagate and destroy the whole upper layers.

Specific techniques used for such error concealment tools are beyond the scope of this document. However, we will give some default mechanisms for handling such cases. A practical Layered P2P system can choose any way to implement it.

3.3. Layered Indication

Once a peer determines the highest layer (HL) it will subscribe to, only chunks belong to HL or lower layers are of interest to this local peer. When the peer sends the GET-PEERLIST request, the peer layer flag should be included in the request message. The neighbor peer and tracker will then take the layer information into consideration in returning peerlist. The peers in peerlist may come with layer info.

The management/status reports from peer to tracker should also indicate layer info about the local peer.

4. Message Flows

4.1. PUT-LAYER (Put Layer Information) into Tracker

When the source put the layer information, tracker will reply an ACK. The layer information keeps the layer number and the information for each layer.

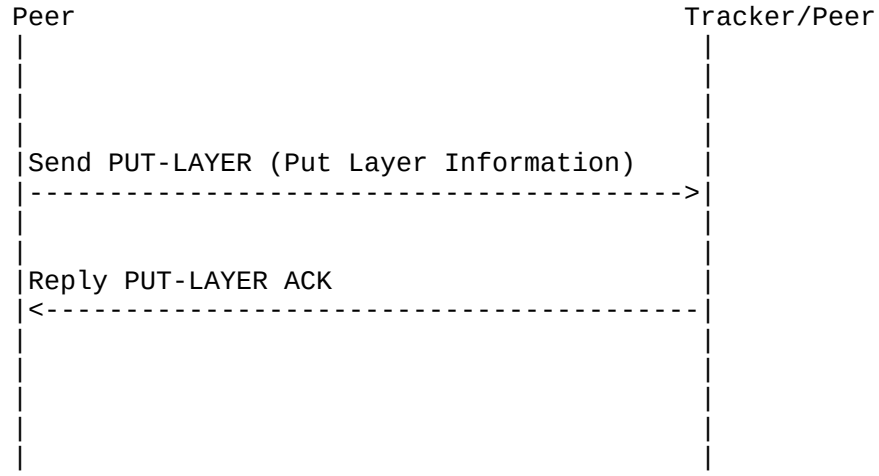


Figure 2: Get Channel Information from Tracker

4.2. GET-LAYER (Get Layer Information) from Tracker

When a peer gets the layer information, the tracker will reply the the layer information. It keeps the layer number and the information for each layer.

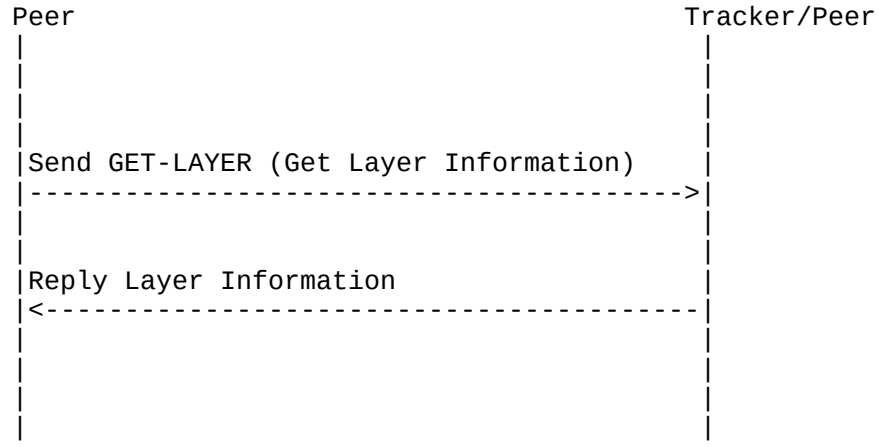


Figure 3: Get Channel Information from Tracker

4.3. PUT-CHUNK (Put Chunk Information) into Tracker

When a peer puts its chunk information, the tracker will reply an ACK. The chunk information keeps the layer number and the bitfield information for each layer in each chunk.

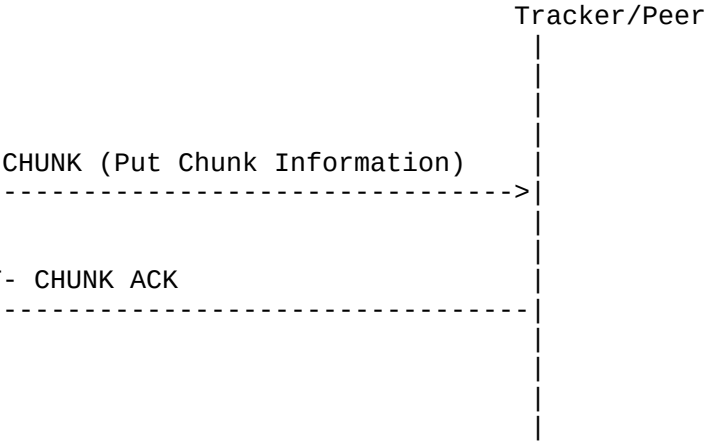


Figure 4: Get Channel Information from Tracker

4.4. GET-PEERLIST (Peer Selection)

The helpful neighbors of a peer should be those with equal or higher layer peers. When a peer sends the GET-PEERLIST request, the numbers of active layers of the peer should be included in the request message.

Each peer's peer selection strategy is based on its own criterion (layers for playback, tolerance for delays, jitter, and other factors, e.g. QoE parameters). The optimal peer selection algorithm is beyond the scope of this document. The document only specifies that there is a peer selection strategy such that Layer ID is part of the parameters, and can be used for data exchange (via buffer map).

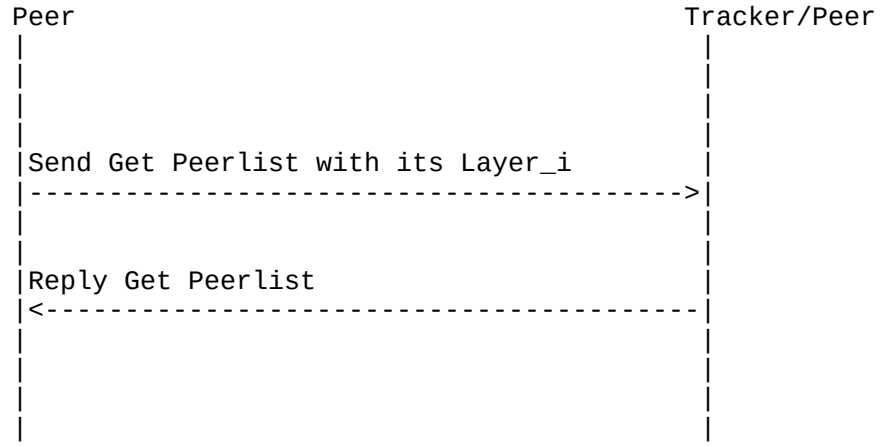


Figure 5: Peer Selection

4.5. LAYER-CHANGE (Layer Change)

Peer can select its suitable active layer according to it current network bandwidth. For example, when a peer's bandwidth is high, the peer can request all layer chunks. But when a peer's bandwidth is slow, the peer can request lower layers, or just base layer chunks. When the peer changes its layer state, it will send message to notify its peers and tracker for updating its information.

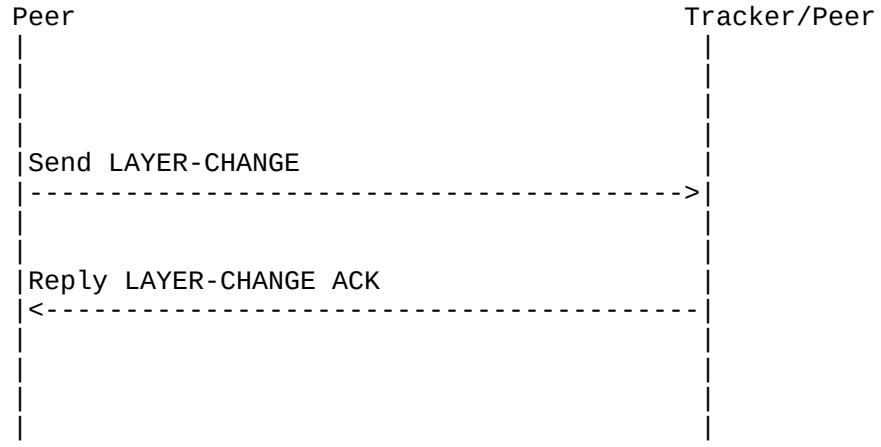


Figure 6: Layer Change

4.6. STATISTICS

Peer sends its statistics information (e.g., peers status, network performance, layer_i, etc) to tracker.

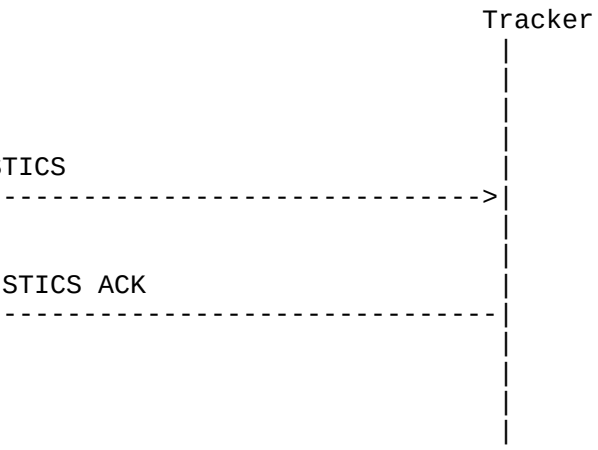


Figure 6: Statistic

5. Open issues

5.1. Data Scheduling

Different coding scheme has different optimal data scheduling strategy. This document does not specify which specific coding scheme to use, except that it has to be compliant with media codec standard (e.g. H.264/AVC or MPEG-4) at each and every layer to avoid a requirement for non-standard compliant decoder in the peer player.

A simple greedy data scheduling strategy is proposed in [3] and can be adopted here for layered P2P streaming. Each peer requests lower layers from lower bandwidth neighbors and higher layers from high bandwidth neighbors. This scheduling strategy is proven to be optimal for a given single peer within a certain time slot but may not be globally optimal.

5.2. System Performance Metrics

Although the layered encoding scheme brings more flexibility for participating peers to achieve adaptive video quality, it also causes challenges to the P2P protocol for layer streaming. In the paper [2], the following four performance metrics are mentioned.

5.2.1. Throughput and Delay

It is the basic requirement that the overall P2P network can maximize the overlay throughput and keep low packet delay. [2]

5.2.2. Layer delivery ratio

In single layered P2P streaming, maximizing the node delivery ratio is almost equal to maximizing the throughput. But it is not the case in layered streaming. In multiple layered P2P streaming, subscribing many layers but with low delivery ratio for each layer can result in high throughput. But, the video quality cannot be high because of the layer dependency. Therefore, it is a key point to ensure high delivery ratio for subscribed layers. [2]

5.2.3. Useless packets ratio

According to the layered encoding/decoding scheme, the decoding of upper layers depends on the availability of lower layers. If some lower layer packets are missed, the packets with the same sequence IDs in the upper layers cannot be correctly decoded. The upper layer packets become useless. Therefore, it is a key point that the useless packet ratio should be kept low. [2]

5.2.4. Jitter prevention

Because Internet is a dynamic environment, the bandwidth variation is common in P2P network. If the node subscribes more layers immediately after bandwidth increased, it may have to drop the high layers after a short while according to the bandwidth decreasing. This short-term subscribe-drop pair is called jitter. Jitter brings fluctuation in quality of service (QoS) and causes its buffer overflow or underflow.

Therefore, jitter should be considered for being prevented by the data scheduling design. [2]

5.3. User Performance Metrics

A set of QoE parameters, e.g. layers for playback, peer preference, tolerance of artifacts (e.g. delay, jitter, freeze, or error blocks effect) etc., may be defined and implemented as system tools inside or outside of the P2P layered streaming. The peer selection criterion can make best use of such information to make sure that it can obtain its optimal playback layer from the right peers.

One of the important goals for layered P2P streaming is to enhance the overall user quality of experience by designing the p2p streaming protocol to utilize the heterogeneous conditions of participating peers. For example, the start-up delay is the duration between a peer makes its request for a stream and the stream actually begins to play at the peer. Layered P2P streaming can reduce the start-up delay by serving the BL (Base Layer) first to the peer while enhancement layers may be delivered later with lower priority. The playback continuity and playback delay can be set as quality metrics as well by the P2P network to dynamically adjust for performance tuning or tradeoff between various quality metrics.

Several quality metrics may be considered in Layered PPSP for live streaming and progressing downloading.

5.3.1. Start-up Delay

The start-up delay is the duration between a peer makes a request for a stream and the stream actually begins to play at the peer.

5.3.2. Playback Continuity

Playback continuity is the percentage of the playing streaming successfully played at the correct time.

5.3.3. Playback Delay

Playback delay is the delay between a streaming chunk which is generated by the source and the streaming chunk being viewed by the peer.

6. Deployment Options

Todo: The content of this section need further input.

7. Protocol Detail

Todo: The content of this section need further input.

8. Security Considerations

Todo: The content of this section need further input.

9. IANA Considerations

Todo: The content of this section need further input.

10. Conclusions

According to the characteristics of P2P layered coding, we propose the P2P layered streaming protocol in the PPSP framework.

11. References

11.1. Normative References

[1] Yingjie Gu, et. al. Tracker Protocol, PPSP (drafting).

11.2. Informative References

[2] LayeredP2P: A New Data Scheduling Approach for Layered Streaming in Heterogeneous Networks. Xin Xiao, Yuanchun Shi, Yuan Gao and Qian Zhang. Infocom 2009

[3] PALS: Peer-to-Peer Adaptive Layered Streaming. Reza Rejaie, Antonio Ortega. NOSSDAV, 2003.

[4] Layered Peer-to-Peer Streaming. Yi Cui, Klara Nahrstedt. NOSSDAV, 2003

12. Acknowledgments

Authors' Addresses

Kent Kangheng Wu
Hong Kong Applied Science and Technology Research Institute Company Limited (ASTRI)
3/F, Building 6, 2 Science Park West Avenue, Hong Kong Science park, Shatin, New Territories, Hong Kong

Phone: 852-34062908
Email: khwu@astri.org

James Zhibin Lei
Hong Kong Applied Science and Technology Research Institute Company Limited (ASTRI)
3/F, Building 6, 2 Science Park West Avenue, Hong Kong Science Park, Shatin, New Territories, Hong Kong

Phone: 00852-34062748
Email: lei@astri.org

Dah Ming Chiu
Hong Kong Applied Science and Technology Research Institute Company Limited (ASTRI)
3/F, Building 6, 2 Science Park West Avenue, Hong Kong Science Park, Shatin, New Territories, Hong Kong

Phone: 00852-34062979
Email: dmchiu@ie.cuhk.edu.hk

PPSP
Internet Draft
Intended status: Informational
Expires: April 26, 2012

L.Xiao
Nokia Siemens Networks
D.Bryan
Cogent Force, LLC/Huawei
Y.Gu
Huawei
X.Tai
China Mobile/BUPT

October 25, 2011

A PPSP Tracker Usage for Reload
draft-xiao-ppsp-reload-distributed-tracker-03

Status of this Memo

Copyright Notice

Abstract

This document defines PPSP tracker usages for REsource LOcation And Discovery (RELOAD). Although PPSP assumes a centralized tracker from peer's point of view, the logical centralized tracker could be realized by a cluster of geographically distributed trackers. In this draft, we design distributed trackers system, which are organized by RELOAD. It provides lookup service for file/channel indexes and Peer Status among the distributed trackers.

Table of Contents

1. Introduction

PPSP assumes that a centralized 'tracker' is used to communicate with the PPSP Peers for content registration and location. The content index is stored in the tracker with location information that which peers have the content.

However, the logically centralized 'tracker' could be also realized by a cluster of geographically distributed trackers or deployed in multiple servers in a data center, which can increase the content availability, the service robustness and the network scalability or reliability. The management and locating of index information are totally internal behaviors of the tracker cluster, which is invisible

to PPSP Peers. The signaling between PPSP Peers and trackers is still the simple request/reply mechanism defined in PPSP tracker protocols. Therefore, the Tracker Nodes themselves construct and maintain a P2P overlay as shown in Figure 1. This document is to define the behaviors of the tracker overlay as a usage for RELOAD. Because all protocols this draft applies to - the PPSP tracker protocol, PPSP peer protocol and RELOAD - are still changing, this draft will certainly change in the future, and is very much a work in progress. The authors encourage list discussion of the draft and any suggestions are welcome.

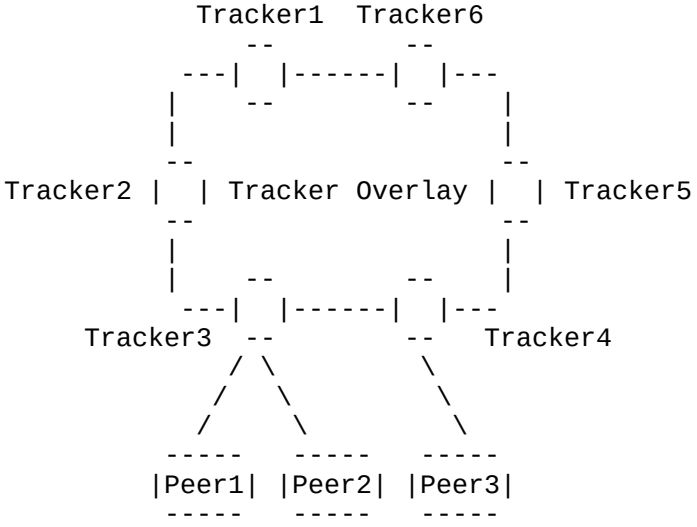


Figure 1 PPSP Peers connect with Tracker Overlay by Tracker Protocol

The document tries to handle the data maintenance for two kinds of information: content index and PPSP peer status, among a cluster of distributed trackers. Four basic functions in tracker overlay are defined as follows:

o Content/ channel index information registration: PPSP Peers registrar/update their contents/channels to a Connection Tracker.(How to find the initial tracker locally is out of scope.) This tracker takes the advantage of the RELOAD data storage functionality to store the index information to tracker nodes in the tracker overly accordingly. At the same time, the local Connection Tracker keeps a copy of local peer's content information for traffic localization.

o Look up a content/channel index: Once a PPSP Peer search for certain content/channel, it makes the request to a local Connection Tracker as defined in PPSP tracker protocol. If the swarm cannot be found or there is not enough peer records for such swarm in the Connection Tracker locally, the tracker will further locate the required index information in the tracker overlay on behalf of the requesting PPSP Peer. Once the full Peer List is fetched, the PPSP Peer will set up communications with the PPSP Peers in the Peer List as defined in PPSP Peer protocol;

o PPSP peer status registration: PPSP Peers registrar/update their status in the tracker overlay. All PPSP peers should firstly register their status to the local Connection Tracker. In order to enable this information being aware globally, the Connection Tracker should then store the position of the PPSP peer's status in the tracker overlay according to RELOAD scheme. The following peer status updates are only sent to the local Connection Tracker, the RELOAD based tracker overlay here only offers a way for remote nodes to find the location of requested peer status.

o Look up status of a certain peer: the tracker overlay can look up the status of a certain PPSP Peer. If the peer status cannot be found in the local Connection Tracker (that means it's not a local peer), the local tracker then searches the Status Position Tracker for the requested peer in the tracker overlay by RELOAD, which gives a route to access the status of the remote peer.

2. Terminology and Conventions

This document makes extensive use of the terminology and definitions from the RELOAD Base Protocol [I-D.ietf-p2psip-base], PPSP Requirements and Problem Statements [I-D.ietf-ppsp-problem-statement][I-D.ietf-ppsp-reqs] and the Gu PPSP Tracker Protocol proposal [I-D.gu-ppsp-tracker-protocol].

This document defines the following additional terminology:

PPSP Peer: The peer in PPSP protocol for content sharing and distribution among swarms.

Tracker Node: The RELOAD Node with PPSP tracker usage. Each Tracker Node takes the responsibility to store and maintain certain content/channel index.

Tracker Overlay: A RELOAD overlay constructed by Tracker Nodes. This Overlay is logically separated with overlay formed by PPSP Peers.

Connection Tracker: The Tracker Node to which the PPSP Peer will connect when it wants to join the PPSP system.

Swarm Tracker: The Tracker Node who is responsible for the swarm in the overlay, and stores the content information (e.g. Peerlist) of the swarm.

Status Position Tracker: A Tracker Node which is responsible to store the Position of certain peers' status of a particular list of Peers.

3. Content Information Registration and Update

To fulfill the functions of content information registration and update mentioned in Section 1, Tracker Node must maintain such resources related to peers;

Content Registration: Information about the content which belongs to a specific swarm. It can be stored in a data structure denoted as ContentRegistration, which primarily includes an identification of the swarm, a name of the content, and a Peer List.

3.1. Data structure of ContentRegistration

Structure The data structure of ContentRegistration uses the RELOAD dictionary kind whereas the DictionaryKey value is the Swarm ID of the content required. The data structure of type ContentRegistration is shown as follows:

```
struct{

       Uint32 index;

       ChunkID chunk_id;
```

```
}ArrayChunkListData;

struct{
       PeerID peer_id;
       ArrayChunkListData  chunklist_data;
}PeerListData;

struct{
       uint16 length;
       PeerListData peerlist_data;
}PeerList;

struct {
       uint16 length;
       opaque content_name<0..2^16-1>;
       PeerList peerlist <0..2^16-1>;
} ContentRegistration;
```

The content of the PeerList structure are as follows:

length

the length of the data structure

content_name

the name of the content

peerlist

the content of Peer List

3.2. Message flows

When a PPSP Peer wishes to share its contents to others, it will inform Tracker Overlay with the swarm information of the contents, then Swarm Tracker need to add this PPSP Peer into the corresponding Peer List to the swarm, or create a new swarm when there is no record of the swarm. A local record of the swarm may also be set up at the Connection tracker. Correspondingly, When a PPSP Peer deletes some old contents locally, it will inform Tracker Overlay that it would like to leave from a particular swarm, then both Connection Tracker and Swarm Tracker need to delete this PPSP Peer from the corresponding Peer List which is defined in the requirement of PPSP [I-D.ietf-ppsp-reqs].

An example is given as the figure has shown below:

1. PPSP Peer wants to join into a swarm to share the content, first it will send a PPSP message "Join" with a Swarm-ID to TrackerA, which is a connection tracker of the Tracker Overlay for PPSP Peer connects to;

2. TrackerA first handles the registration locally, then finds the Swarm Tracker by mapping the swarm ID to node ID of the Swarm Tracker, to forward the request. So TrackerA sends a RELOAD message "StoreReq" to TrackerB who is the Swarm Tracker for the content swarm;

3. When Swarm Tracker (TrackerB) receives the request (or if TrackerA is responsible for the Peer List of the swarm, TrackerB=TrackerA), it searches locally the Peer List of the swarm whose ID is the Swarm-ID, then add the Node-ID of the PPSP Peer into the Peer List or delete it from that, and send the result of the operation (e.g. successful or failed) in a RELOAD message "StoreReqAns" to TrackerA through Tracker Overlay;

4. Finally, TrackerA analyses the received message, and responds to the requesting Peer by a corresponding PPSP message: "Successful (OK)" or some error messages.

Note: When PPSP Peer is the first node of the swarm, which means it is the first one who stores this kind of content in the network, TrackerB doesn’t have records of the new swarm, TrackerB will create a new ContentRegistration for the swarm locally, and put the identification of PPSP Peer into Peer List of this new

ContentRegistration, then send the result of the operation (e.g. successful or failed) in a RELOAD message "StoreReqAns" to TrackerA through Tracker Overlay.

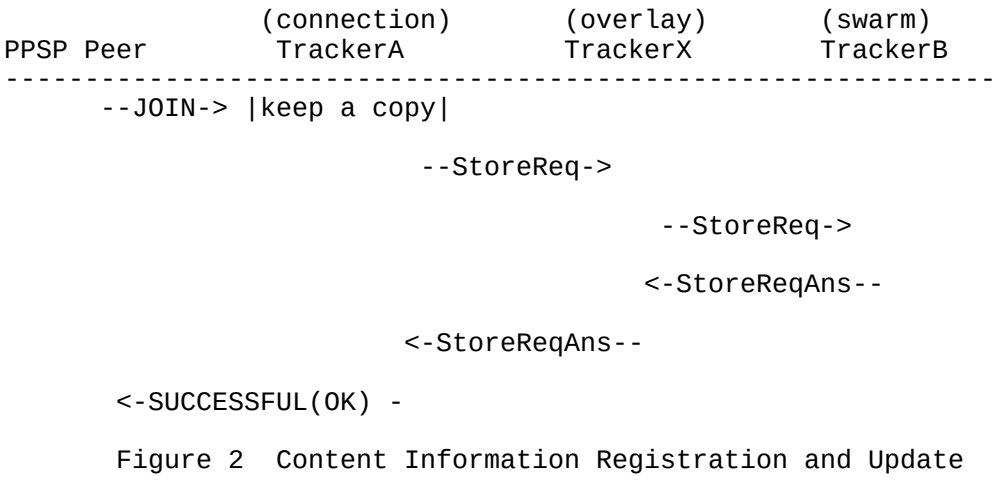


Figure 2 Content Information Registration and Update

If PPSP Peer wishes to update content information, for example, list of chunks it has, it sends a PPSP message "JOIN_CHUNK" to TrackerA. TrackerA makes update in its local table, and then sends the corresponding RELOAD message to TrackerB to update the detailed chunk-IDs in the Swarm according to the request message.

4. Lookup Content Index (a Swarm)

When a PPSP Peer wants to use some streaming service, which means it wants to download some interested contents from the system, it firstly needs to get related Peer List from Tracker Overlay. As the figure has shown below:

1) PPSP Peer wants to watch a video belonging to a swarm with a Swarm-ID, firstly it sends a PPSP message "Find" with the Swarm-ID to Connection TrackerA;

2) If TrackerA has enough local peer record for swarm, it can reply the request directly. Or it maps the Swarm-ID into a Node-ID to identify the Swarm Tracker, TrackerB, which stores the Peer List of the requested swarm. It then sends a RELOAD message "FetchReq" to TrackerB;

3) When Swarm TrackerB receives the request (or if TrackerA is responsible for the Peer List of the swarm, TrackerbB=TrackerA), it searches the Peer List of the swarm locally, then send the Peer List which is organized by the data structure of PeerList in a RELOAD message "FetchReqAns" to TrackerA through Tracker Overlay;

4) Finally, TrackerA analyses the received PeerList structure, and reconstructs it into a PPSP message "Successful(OK)", then forwards it to the PPSP Peer.

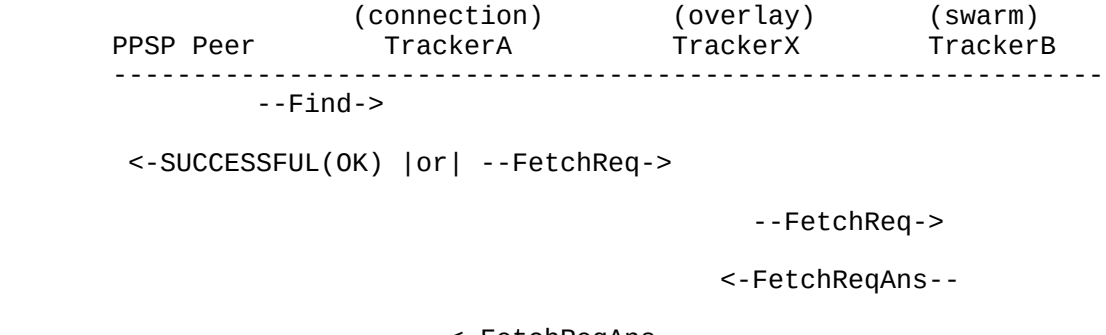


Figure 3 Content Information Lookup

5. Peer Status Registration, Update and Lookup

To fulfill the functions of peer status registration, update and lookup mentioned above, Tracker Node must maintain such resource related to peers:

Information about status of peers: the local Connection Tracker takes the responsibility to maintain the PPSP Peer status locally, including online time, link status, node capability and other streaming parameters, etc. It can be stored in a data structure denoted as PeerStatus.

Position of PPSP peer status: each PPSP Peer can be mapped to a Status Position Tracker in the tracker overlay. The status Position Tracker takes responsibility to only record the route (i.e., the address of the local Connection Tracker of the Peer) to access the PPSP Peer status.

5.1. Data Structure of PeerStatusIndex

The data structure of PeerStatusIndex uses the RELOAD dictionary kind whereas the DictionaryKey value is the Peer ID. The data structure of type PeerStatusIndex is shown as follows:

```
struct{
       TrackerID Connection_Tracker_ID;
}PeerStatusIndex;
```

The content of the PeerStatusIndex structure are as follows:

trackerID the ID of the Peer’s Connection Tracker;

6. Kind Definition

6.1. CONTENT-REGISTRATION Kind Definition

This section defines the CONTENT-REGISTRATION kind.

- Name: CONTENT-REGISTRATION

- Kind IDs: The Resource Name for the CONTENT-REGISTRATION Kind-ID is Swarm Name. The data stored is a CONTENT-REGISTRATION, which contains a identification of the swarm, a name of the content, and a list of PPSP Peer-IDs with or not a list of chunk-IDs for each PPSP Peer to show which chunks the PPSP Peer has.

- Data Model: The data model for the CONTENT-REGISTRATION Kind-ID is dictionary. The dictionary key is the Swarm-ID of the peer action as focus.

- Access Control: USER-NODE-MATCH.

6.2. PEER-STATUS Kind Definition

This section defines the PEER-STATUS kind.

- Name: PEER-STATUS

o Kind IDs: The Resource Name for the PEER-STATUS Kind-ID is Peer Status. The data stored is a PEER-STATUS, which contains a identification of the peer and a identification of the peer's connection tracker.

o Data Model: The data model for the PEER-STATUS Kind-ID is dictionary. The dictionary key is the Peer-ID.

o Access Control: USER-NODE-MATCH.

7. Security Considerations

This document does not currently introduce security considerations.

8. IANA Considerations

This document does not specify IANA considerations.

9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

References

9.1. Normative References

[1] [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] [I-D.ietf-ppsp-reqs] Zong, N., Zhang, Y., Avila, V., Williams, C., and L. Xiao, "P2P Streaming Protocol (PPSP) Requirements", draft-ietf-ppsp-reqs-03 (work in progress), July 2011.

[3] [I-D.ietf-ppsp-problem-statement] Zhang, Y., Zong, N., Camarillo, G., Seng, J., and Y. Yang, "Problem Statement of P2P Streaming Protocol (PPSP)", draft-ietf-ppsp-problem-statement-03 (work in progress), August 2011.

[4] [I-D.ietf-p2psip-base] Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-18 (work in progress), August 2011.

[5] [I-D.gu-ppsp-tracker-protocol] Yingjie, G., Bryan, D., Zhang, Y., and H. liao, "Tracker Protocol", draft-gu-ppsp-tracker-protocol-04 (work in progress), May 2011.

9.2. Informative References

Author's Addresses

Lin Xiao
Nokia Siemens Networks
No.14 Jiuxianqiao Road
Beijing, 100016
P.R.China

Phone: +86-13810361287
Email: lin.xiao@nsn.com

David A. Bryan
Cogent Force, LLC / Huawei

Email: dbryan@ethernot.org

Yingjie Gu
Huawei
No. 101 Software Avenue
Nanjing, Jiangsu Province 210012
P.R.China

Phone: +86-25-56624760
Email: guyingjie@huawei.com

Xuan Tai
China Mobile/BUPT

Phone: +86-13581762082
Email: taixuanyueshi@gmail.com

PPSP W. Zeng
Internet-Draft Y. Gu
Intended status: Informational Huawei Technologies
Expires: April 25, 2011 October 22, 2010

P2P Streaming Protocol Pro-incentive Parameters
draft-zeng-ppsp-protocol-pro-incentive-para-01

Abstract

This document analyzes the common parameters that are essential for deriving incentive mechanisms to promote peer cooperation and system robustness in a P2P system, and proposes to incorporate these pro-incentive parameters in information exchanges in the P2P streaming protocols to be specified, e.g., in the tracker protocol proposed in [draft-gu-ppsp-tracker-protocol], and in the future in the peer protocol proposed in [draft-gu-ppsp-peer-protocol].

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 25, 2011.

Table of Contents

1. Introduction

Lack of cooperation (free riding) is one of the key problems that confront today's P2P systems. What makes this problem particularly difficult is the unique set of challenges that P2P systems pose: large populations, high turnover, asymmetry of interest, collusion, zero-cost identities, and traitors [RobustIncentives].

Many incentive mechanisms have been proposed in the literature to promote cooperation and system robustness of a P2P system. The goal of this contribution however is not to choose a particular incentive mechanism to specify in the P2P streaming protocol, but instead is to analyze the common parameters that are essential for various incentive mechanisms, and propose to incorporate these pro-incentive parameters in information exchanges in the P2P streaming protocols.

2. Document Conventions

2.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.2. Terminology

The draft uses the terms defined in [draft-gu-ppsp-tracker-protocol].

The draft also uses some new terms, as defined below.

Piece: equivalent to Chunk, a basic unit of partitioned stream, which is used by a peer for the purpose of storage, advertisement and exchange among peers. As the term "Piece" has been widely used in the literature, Piece and Chunk are used interchangably in this draft.

Unchoking: uploading data to a selected peer in a P2P system.

Rarest First: When selecting which piece to start downloading next, peers generally download pieces which the fewest of their own peers have first.

Free Riding: downloading data from peers without uploading any data to peers.

Tit-for-Tat: peers exchange chunks preferentially with other peers with whom they have successfully exchanged chunks in the past at high

bandwidth.

Discount Parameter: the weight of the next move compared to the current one. It measures the degree to which the payoff of each move is discounted relative to the previous move.

Honest Piece Revelation: peers truthfully reveal which pieces they have.

Under-Reporting: peers not revealing some pieces they have in order to make profit.

3. Incentives in P2P Systems

Several works have demonstrated the limitations of P2P protocols in the presence of selfish or malicious users [RobustIncentives] [Free-riding]. Rewarding peer contributions has been suggested to overcome these limitations, e.g., in [ContributionAware][RobustIncentives]. A well known example is the tit-for-tat mechanism used in BitTorrent [BitTorrent].

3.1. Measurement of peer contribution

A typical metric for measuring peer’s contribution is the amount of upload a peer has contributed. For example, BitTorrent uses a bilateral mechanism called tit-for-tat. The amount of data a peer would upload to another peer depends on how much it has downloaded from that peer in the past.

Despite its pro-incentive approach, recent study [prTorrent] has empirically shown that BitTorrent is vulnerable to strategic manipulation by its constituent peers in a swarm. For example, the Discount Parameter (DP) attack is an incentive threat that exploits the core of BitTorrent cooperation - the tit-for-tat based unchoking. The weight (or importance) of the next move compared to the current is called Discount Parameter, because it measures the degree to which the payoff of each move is discounted relative to the previous move. If the DP is small, peers might defect and not worry about future consequences. In BitTorrent-like file sharing systems, piece rarity is a DP. This means if the pieces available for download to peer B from A are not lucrative enough (possibly because B can get them from somewhere else at cheaper uploads, i.e. the piece rarity of pieces that A holds are low), then B has sufficient incentive to defect. Defect would mean that B will not upload to A in the "tat" phase, denying A one round of legitimate download. Then B will break connection with A, leaving A with no opportunity to snub him.

Moreover, Honest Piece Revelation and Free-Riding is becoming an increasing concern. As explained in [BTAuction], Honest Piece Revelation is not enforced in BitTorrent and peers have sufficient incentive to stray from truthfulness. The results in [BTAuction] indicate that selfish BitTorrent clients can benefit from under-reporting pieces. Under reported pieces keep getting rarer in the swarm and therefore, the demand of peers holding those pieces increases due to the rarest first piece selection approach. It has been discovered in [prTorrent] that if a colluding group of peers are involved in under-reporting over a considerable period to improve their demand in the swarm, it might lead the whole swarm into premature starvation in which all peers have all the pieces of a file, except a few.

The findings in [prTorrent] indicate that it is the orthogonal treatment of piece rarity and unchoking that has encouraged strategic manipulation enabling unfair maximization of incentives in p2p systems. Note that BitTorrent uses rarest first approach for piece selection but not for peer selection (unchoking). The prTorrent protocol [prTorrent] unifies a Piece Rarity factor with the BitTorrent Unchoking Algorithm, i.e., peer selection for unchoking not only depends on the uploading bandwidth of the candidate peers, but also how valuable the pieces they have uploaded are. It is shown how strategic formulation of the Piece Rarity parameter can optimize incentives in a swarm, and help its constituent peers in achieving the equilibrium facilitating truly co-operative behavior.

The Piece Rarity factor depends on a number of other variables measured at the piece, peer and swarm levels, as summarized below.

- the global availability (in all swarms of the tracker): the high availability of a piece outside the swarm may result in peers leaving the swarm.

- local availability (in the target swarm) of a piece: a rarer downloaded piece has more value to the swarm

- number of upload slots a candidate peer has: long term benefit can be expected from a peer with more uploading potential

- the completion factor (i.e., the ratio of the number of pieces of the file that a peer has to total number of pieces of that file) of the candidate peer: a peer with high completion factor is a good one to maintain a good upload/download relation with.

- the contention in the swarm (i.e., the ratio of total number of peers to total number of seeds): high contention implies more strategic value of a piece.

Several studies (e.g., [RobustIncentives][Incentives][Peer-assisted]) however have pointed out the limitations of bilateral mechanisms (e.g., in the case of asymmetry of interest), and make the case for designing more global contribution-aware mechanisms. This is especially an issue in real-time streaming. Many systems that distribute content with the help of P2P overlays measure peer contribution and incentivize participation. Peers who contribute more are rewarded with better performance via different mechanisms such as higher priority in the distribution overlay (e.g., [ContributionAware][Collusion][Contracts]) or priority service through server- assisted downloads (e.g., [CooperativeCD]), or discount coupons [CooperativeCD]). Such systems are referred to as contribution aware peer-assisted content distribution systems. A typical metric to measure peer's contribution to the swarm is again the amount of upload a peer has contributed to the swarm (as opposed to individual peers). Approximate equilibria has also been proposed in [FlightPath] to guide how one designs systems to incentivize selfish (or rational) peers to obey protocols.

4. Pro-incentive Protocol Parameters

The above analysis suggests that it is important for a p2p streaming protocol to support the exchange/report of necessary information based on which a p2p streaming system can optimize incentives in a swarm to provide robust and improved services. Some of the basic pro-incentive parameters are listed below.

- no_upload_slots: a peer's upload bandwidth (i.e., number of upload slots a peer has).
- bytes_uploaded: total amount of data that a peer has uploaded
- bytes_downloaded: total amount of data that has been downloaded from a peer
- chunk_nos: total number of chunks of a file that a peer has.
- seed_nos: total number of seeds.
- peer_nos: total number of peers.
- chunk_copies_swarm: chunk availability, i.e., total number of copies of a chunk available in the swarm.

Exchanges of these parameters between tracker and peers SHALL be supported in the tracker protocol, e.g., through the methods defined in [draft-gu-ppsp-tracker-protocol]. A subset, e.g., a peer's

no_upload_slots and the chunk_nos of a file that a peer has, SHOULD be supported in the peer protocol. Some of these parameters (e.g.,no_upload_slots) have already been proposed in [draft-gu-ppsp-tracker-protocol] for the purpose of facilitating optimization of the system performance.

4.1. Suggested Changes in the Tracker Protocol

It is recommended that exchange of the following statistics be supported in the tracker protocol, e.g., by adding them to Table 3 (Section 4.1.10.1) in the proposed tracker protocol [draft-gu-ppsp-tracker-protocol].

XML Value	Definitions/Description
BytesUploaded	Total amount of data that a reporting peer has uploaded. This is to be reported by a peer to the tracker.
ChunkMAP	Indicates which chunks of a file that a peer has. This is to be reported by a peer to the tracker.
SeedNumber	Total number of seeds in the swarm. This is to be updated by the tracker to peers periodically or upon request by a peer.
PeerNumber	Total number of peers in the swarm. This is to be updated by the tracker to peers periodically or upon request by a peer.
ChunkCopies	Total number of copies of a chunk available in the swarm. Each ChunkCopies should be paired with a ChunkID. This is to be updated by the tracker to peers periodically or upon request by a peer.
BytesDownloaded	Total amount of data reported by a peer that has been downloaded from a different peer. Each BytesDownloaded should be paired with a PeerID. This is to be reported by a peer to the tracker.

Table 1: Additional Property Types to be supported in the tracker protocol

4.2. Suggested Changes in the Peer Protocol

It is recommended that exchange of the following statistics be supported in the peer protocol.

XML Value	Definitions/Description
UploadBW	A peer's upload bandwidth (i.e., number of upload slots a peer has).
ChunkNumber	Total number of chunks of a file that a peer has. This is necessary in case ChunkMAP reported is not a complete map.
ChunkMAP	Indicates which chunks of a file that a peer has.

Table 2: Additional Property Types to be supported in the peer protocol

5. Acknowledgements

The author would like to thank the following people for their help and comments: Robins George, Richard Yang, and Suman D. Roy.

6. IANA Considerations

This draft includes no request to IANA.

7. Security Considerations

Trustworthiness of these pro-incentive parameters is critical to the effectiveness of the incentive mechanisms. For example, ChunkMap defined above may be essential, and may need to be accurate. The P2P system should be designed in a way such that a peer will have the incentive to report truthfully its ChunkMap (otherwise it may penalize itself, as in the case of under-reporting addressed in [prTorrent]). Furthermore, both the amount of upload and download should be reported to the tracker to allow the tracker to check if there is any inconsistency between the upload and download report, and establish an appropriate credit/trust system. Alternatively, exchange of cryptographic receipts signed by receiving peers can be used to attest to the upload contribution of a peer to the swarm, as was suggested in [Contracts]. Security will be further considered in future versions of this draft.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[BTAuction]
Levin, D., LaCurts, K., Spring, N., and B. Bhattacharjee, "BitTorrent is an Auction: Analyzing and Improving BitTorrent's Incentives", ACM Special Interest Group on Data Communication , 2008.

[BitTorrent]
Cohen, B., "Incentives build robustness in BitTorrent", Proc. of P2P-Econ, Berkeley, California, USA , June 2003.

[Collusion]
Lian, Q., Zhang, Z., Yang, M., Zhao, B., Dai, Y., and X. Li, "An empirical study of collusion behavior in the Maze P2P file-sharing system", In Proc. ICDCS , 2007.

[Contracts]
Piatek, M., Krishnamurthy, A., Venkataramani, A., Yang, R., Zhang, D., and A. Jaffe, "Contracts: Practical Contribution Incentives for P2P Live Streaming", USENIX Symposium on Networked Systems Design and Implementation (NSDI) , April 2010.

[ContributionAware]
Sung, Y., Bishop, M., and S. Rao, "Enabling Contribution Awareness in an Overlay Broadcasting System", In Proc. ACM SIGCOMM , 2006.

[CooperativeCD]
Sirivianos, M., Park, J., Yang, X., and S. Jarecki, "Dandelion: Cooperative Content Distribution with Robust Incentives", In Proc. USENIX ATC , 2007.

[FlightPath]
Li, H., Clement, A., Marchetti, M., Kapritsos, M., Robison, L., Alvisi, L., and M. Dahlin, "FlightPath: Obedience vs. Choice in Cooperative Services", the USENIX Symposium on Operating System Design and Implementation (OSDI) , Dec. 2008.

[Free-riding]
Sirivianos, M., Park, J., Chen, R., and X. Yang, "Free-

riding in BitTorrent networks with the large view exploit", In Proc. IPTPS , 2007.

[Incentives]
Lai, K., Feldman, M., Stoica, I., and J. Chuang, "Incentives for cooperation in peer-to-peer networks", In Proc. P2P Econ , 2004.

[Peer-assisted]
Aperjis, C., Freedman, M., and R. Johari, "Peer-Assisted Content Distribution with Prices", In Proc. CoNeXT , 2008.

[RobustIncentives]
Feldman, M., Lai, K., Stoica, I., and J. Chuang, "Robust Incentive Techniques for Peer-to-Peer Networks", In Proc. ACM EC , 2004.

[draft-gu-ppsp-peer-protocol]
Gu, Y. and D. Bryan, "Peer Protocol", (IETF Work in Progress) http:// tools.ietf.org/html/draft-gu-ppsp-peer-protocol-00, July 2010.

[draft-gu-ppsp-tracker-protocol]
Gu, Y., Bryan, D., Zhang, Y., and H. Liao, "Tracker Protocol", (IETF Work in Progress) http:// tools.ietf.org/html/draft-gu-ppsp-tracker-protocol-01, July 2010.

[prTorrent]
Roy, S. and W. Zeng, "prTorrent: On Establishment of Piece Rarity in the BitTorrent Unchoking Algorithm", International Conference on Peer-to-Peer Computing (P2P2009) , September 2009.

Authors' Addresses

Wenjun (Kevin) Zeng
Huawei Technologies
USA

Email: zengw@huawei.com

Yingjie Gu
Huawei Technologies
No.101 Software Avenue
Nanjing, Jiangsu Province 210012
P.R. China

Email: guyingjie@huawei.com