

PPSP
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2014

L. Deng
J. Peng
China Mobile
Y. Zhang
CoolPad
Y. Huang
Huawei
February 12, 2014

Efficient Chunk Availability Compression for PPSP
draft-deng-ppsp-bfbitmap-05.txt

Abstract

Bloom filters are proposed to be used in compressing chunk availability information, periodically exchanged between peers and the tracker through PPSP-TP and PPSP protocols, to reduce relevant cost and enhance scalability.

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 August 16, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Background on Bloom Filter	3
3. BF-based Chunk Availability Exchange	5
3.1. A non-BF PPSP session	5
3.2. A PPSP Session with BF-bitmaps	7
3.3. Summary	8
3.3.1. Base TP protocol	9
3.3.2. Extended TP protocol	9
3.4. Peer protocol	10
4. Open Issues	12
5. Security Considerations	12
6. IANA Considerations	12
7. References	13
7.1. Normative References	13
7.2. Informative References	13
Authors' Addresses	14

1. Introduction

As it is pointed out by [I-D.ietf-ppsp-problem-statement], current P2P streaming practices often use a "bitmap" message to exchange chunk availability. The message is of kilobytes in size and exchanged frequently, e.g., an interval of several seconds or less.

To begin with, 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.

Even in a wire-line P2P streaming application, frequent exchange of large volume of bitmap information, is among the key factors that set a limit to the system's efficiency and scalability [P2P-limit].

Therefore, the following requirements for PPSP protocols in terms of chunk availability exchange are stated in [I-D.ietf-ppsp-problem-statement] :

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.

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

In this draft, we propose an efficient bitmap compression scheme for chunk availability information in PPSP protocols. Given the Bloom Filters' wide applications in Internet and demonstrated efficiency with highly compacted data structure and low complexity and cost in terms of information storage, transportation and computation, it is expected to relieve a PPSP implementer from the dilemma between "the frequency of messages" (i.e. the timely exchange of information that contributes to better user experience) and "efficient use of bandwidth" (i.e. the limit of a single node/peer that holds the system's overall scalability by throat).

2. Background on Bloom Filter

Bloom Filter (or BF for short) was first introduced in 1970s [BF-bloom], which makes use of multiple hashing functions to build a mapping from a set of elements to a compact binary array, to realize highly efficient member queries with a tolerably low error rate of wrongly reported hits. Despite their extraordinary efficiency in terms of storage reduction and query acceleration, BFs suffer from the fact that there is possibility that a non-member of the set be wrongly taken as a member after the query. However, research [BF-analysis] shows that the odds that a BF-based membership query makes an erroneous hit can be suppressed to near zero, by a tactful configuration of various system parameters, including the hash functions used, the number of hash functions to be used, and the length of the bit array.

```

-----
BF(set S, integer m, hash set H)
1 filter=allocate m bits initialized to 0;
2 for each element xi in S do
3   for each hash functions hi in H do
4     filter[hi(xi)]=1;
5 return filter;
-----

MT(element elm, BF filter, integer m, hash set H)
1 for each hash functions hi in H do
2   if (filter[hi(elm)]!=1)
3     return false;
4 return true;
-----

ST(BF query, BF filter)
1 temp=query OR filter;
2 if (temp!=filter)
3   return false;
4 return true;
-----

```

Figure 1: Basic algorithms for BF-bitmap

As shown in Figure 1, the BF(S,m) algorithm takes a n-membered subset $S=\{x_1, x_2, \dots, x_n\}$ from a universal set U as input, and outputs a m-bit binary array B as a compacted representation of S. In order to do that, it makes use of k independent random hash functions, each of which maps a member to a marked bit in B (i.e $h_j: U \rightarrow [1, m]$, $j=1 \dots k$). The BF algorithm is highly efficient in the following aspects:

- o It is quite simple and straightforward to generate the BF representation of a set S, $B=BF(S)$: initially, all the bits in B is set to 0; then, for each member x of the set S, mark each bit in B, to which a hash function maps x (as shown in Figure 1 as the BF algorithm).
- o It is highly efficient to check whether or not a given element x is in any BF-represented set $B=BF(S)$: for each hash function h_j , check the value of $B[h_j(x)]$ against 1. It is always safe to exclude the element x out of set S, once there is a zero-valued hash bit. Otherwise it is assumed that x is a member of S (the MT algorithm in Figure 1).
- o It is also highly efficient to check whether or not a given element set S is contained by another set S' if they are both represented as BF-bitmaps, say $query(=BF(S))$ and $filter(=BF(S'))$ for instance. It is always safe to return "false" to the requestor if there's any marked bits in $BF(S')$ and not marked in

BF(S), which can be realized by two simple bitwise operations (as shown by the ST algorithm in Figure 1).

For instance, given a 2GB movie file, the original bitmap for a sharing peer would be 1024-bit (if the system is using 2MB-sized segments). By simply using 4 uniform random hash functions and a 128-bit BF-bitmap, the possibility of erroneous hits by MT algorithm would be lower than 3%.

As for a simple illustration, the 4 hash functions may be established through the MD5 message-digest algorithm [RFC1321], a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value from an arbitrary binary input. MD5 has been utilized in a wide variety of security applications, and is also commonly used to check data integrity.

Specifically, the processing of 4 hash functions is as follows: use the MD5 algorithm to turn a given chunk_ID into a 128-bit binary array, further separate the 128 bits into 4 arrays (32-bit each), and finally divide each of them using 128 to yield 4 integers in the range of [1,m].

3. BF-based Chunk Availability Exchange

We first construct a general message flow (shown in Figure 2) from PPSP protocols, and then discuss how to integrate BF-bitmap algorithm with it.

3.1. A non-BF PPSP session

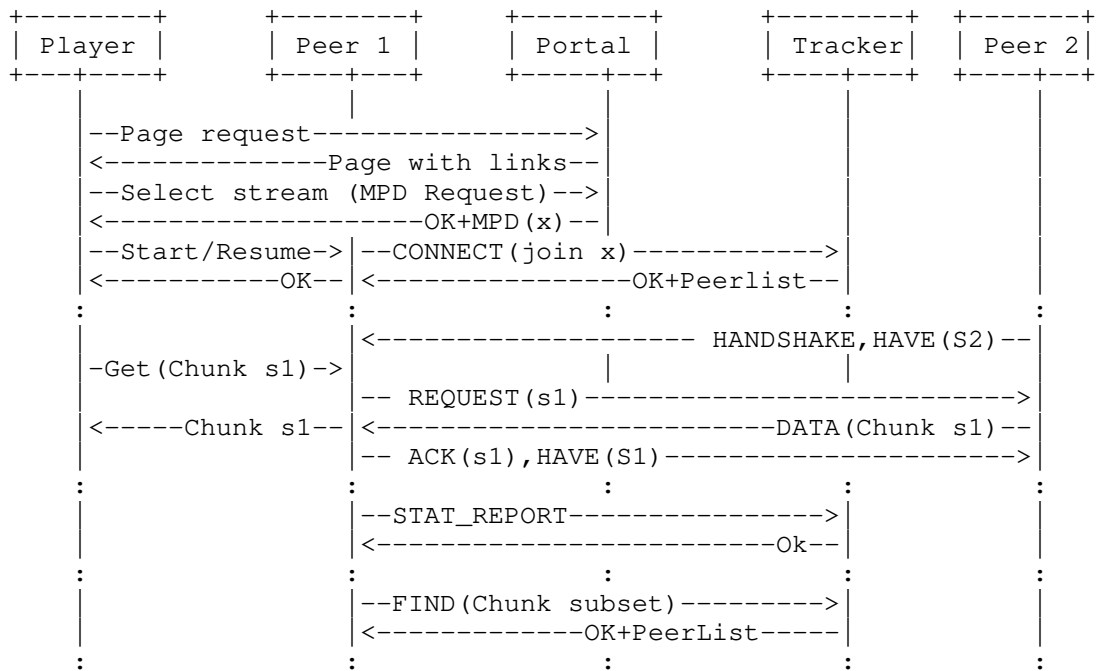


Figure 2: A typical PPSP session for watching a streaming content.

When a peer wants to receive streaming of a selected content (Leech mode):

1. Peer connects to a connection tracker (which may be located through a web portal) and joins a swarm.
2. Peer acquires a list of other peers in the swarm from the connection tracker (via the tracker protocol) through the CONNECT message.
3. Peer exchanges its content availability with the peers on the obtained peer list (via peer protocol) through the HAVE message.
4. Peer requests content from the identified peers (via peer protocol) through the REQUEST-DATA messages.
5. Peer periodically reports its status and chunk availability with the tracker (via the tracker protocol) through the STAT_REPORT message.
6. Peer acquires a list of other peers for a specific subset of media chunks in the swarm from the connection tracer (via the tracker protocol) through the FIND message.

3.2. A PPSP Session with BF-bitmaps

This document proposes to employ bloom filter algorithms in compressing chunk availability information exchanged and stored between peers and the tracker through the PPSP-TP protocols and PPSP protocol. Relevant extensions to the current protocols are summarized as follows: (as shown in Figure 3)

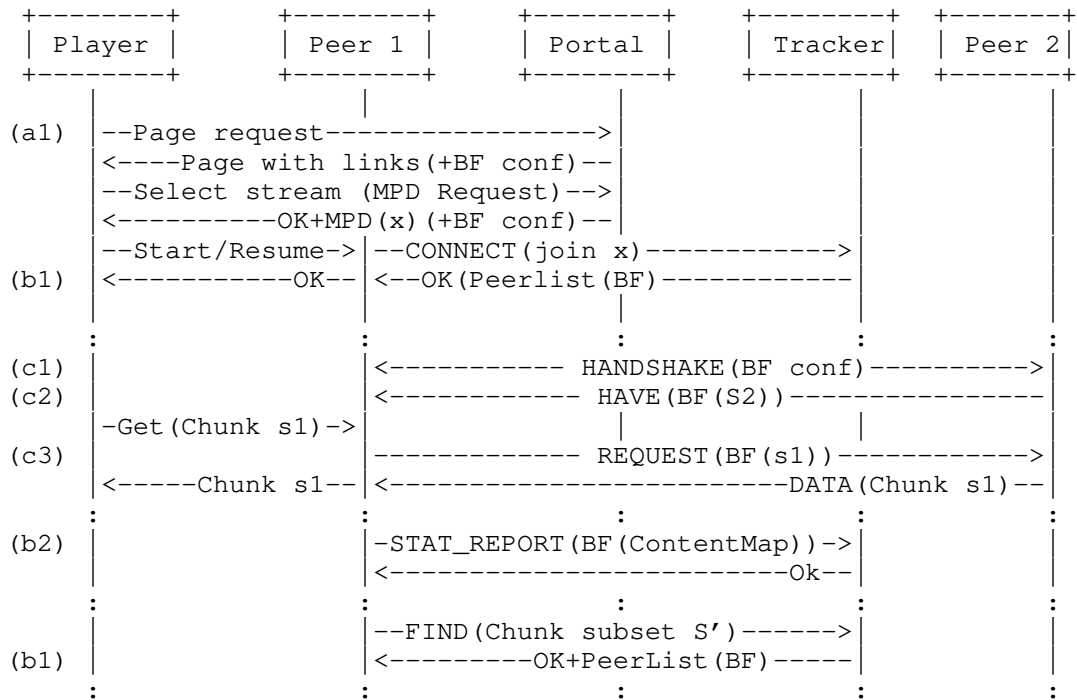


Figure 3: A typical PPSP session with BF-bitmaps.

a. Configuration Setup:

- * (a1) Configuration Setup: m, The length of the output bit array and H, the hash functions in use, are system level parameters that should be configured globally: (a1) the BF configurations (or BF conf for short) be stored at the web portal and published to a requesting peer through the web page or MPD file transaction, which could be incorporated to the "Installation and Initial Setup" in PPSP TP protocol [I-D.ietf-ppsp-base-tracker-protocol].

- b. Integration to the extended TP protocol
[I-D.ietf-huang-extended-tracker-protocol]: In the extended TP protocol, a new "ContentGroup" element is added in requests, i.e., CONNECT and FIND, if the request includes a content scope. In addition, this new element is also added to the "StatisticsGroup" element for containing chunkmap information in STAT_REPORT messages. To enable the BF-based schemes to be used, the relevant interactions are described as follows:
- * (b1) In response to a CONNECT(join)/FIND request from a peer, the tracker may accompany the returned peer list with each recommended peer's BF-formed chunk availability bitmap, as the initial guidance for the requestor to start looking for neighbors in the same swarm. The additional cost for bearing the chunk-level availability information is constant ($O(m)$) for each peer in the returned peer list.
 - * (b2) STAT_REPORT: Peers use the BF(S, m, H) algorithm for compressing the subset of locally stored and integrity verified chunks (set S) in terms of a given swarm-ID, whenever reporting or updating its chunk availability information with the tracker. As the length of each BF-bitmap is constant ($O(m)$), this will greatly reduce the tracker's resource expenditure in communicating and storing such information for a large peer population.
- c. Integration to the peer protocol
[I-D.ietf-ietf-ppsp-peer-protocol]:
- * (c1) HANDSHAKE: Peers exchange their local settings for the chunk compression schemes via HANDSHAKE messages.
 - * (c2) HAVE: Peers use the BF(S, m, H) algorithm for compressing the subset of locally stored and integrity verified chunks (e.g. set S_2 for Peer 2 in Figure 3) in terms of a given swarm-ID, whenever sharing its chunk availability information with another peer. The length of each BF-bitmap is constant ($O(m)$).
 - * (c3) REQUEST: For a downloading peer to decide which neighbor to request for a given chunk_ID s , it uses the member query algorithm MT(s, bf, m, H) on each neighbor's BF-bitmap bf . The computation cost for this member check is constant ($O(m)$). It is also optional for a requesting peer to use BF-bitmap to indicate its data request to another peer, if needed.

3.3. Summary

3.3.1. Base TP protocol

In view of the wish that the "configuration processes for the PPSP Tracking facility, the service Portal and content sources are not standardized, enabling all the flexibility for implementers. But as there could have been different options to realize the chunk availability information (i.e. the chunk bitmap), it should be systematically configured so as to be mutually understandable to both parties. Therefore, we propose to add the following sentence to [I-D.ietf-ppsp-base-tracker-protocol] Section 5.1.1 "Installation and Initial Setup": "In case of a peer or the tracker wishes to exchange further information about the available peers in a flexible way, e.g. the chunk availability information of a specific peer in the same swarm could be represented in a various ways, there should be a way of negotiation/indication about the specific method/parameters in use, e.g. in the MPD file downloaded by the requesting peer from the web portal."

3.3.2. Extended TP protocol

The "ContentGroup" element in STAT_REPORT message has been extended to contain contentmap information, as shown in Figure 4).

Element Name or Attribute Name	Use	Description
ContentGroup	0...1	Provides information on content.
CAM	1	Describes the chunk addressing method of this content. The value is identical with the value of Table 6 of [I-D.ietf-ppsp-peer-protocol]
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 @		

Figure 4: Semantics of ContentGroup.

In order to incorporate BF-based representation, we propose to

- o use the "SegmentInfo" field for the BF-formatted bitmap; and
- o the three attributes of "startIndex", "endIndex" and "chunkmapSize" are meaningless in this case and are allowed to be absent.

3.4. Peer protocol

In peer protocols, the bitmap representation schemes are called "chunk addressing schemes", whose choice and configuration is included in the protocol options information exchanged between peers in communication initiation via HANDSHAKE messages. As shown by Figure 5), there are currently five options defined for bitmap representations.

Method	Description
0	32-bit bins
1	64-bit byte ranges
2	32-bit chunk ranges
3	64-bit bins
4	64-bit chunk ranges
5-255	Unassigned

Figure 5: PPSP Peer Chunk Addressing Methods.

After downloading and verifying a chunk, a peer updates its local chunk availability information to each neighboring peer in the same swarm using HAVE message, which consists of a single chunk specification that states that the sending peer has those chunks and successfully checked their integrity. In order to incorporate BF-based representation, we propose to

- o add a defined value (e.g. 5) from "unassigned" value range of the PPSP peer chunk addressing methods for the BF-formatted bitmap; and
- o use HAVE/REQUEST message to convey the BF-format array for the overall local chunk bitmap in the way as shown in Figure 6.

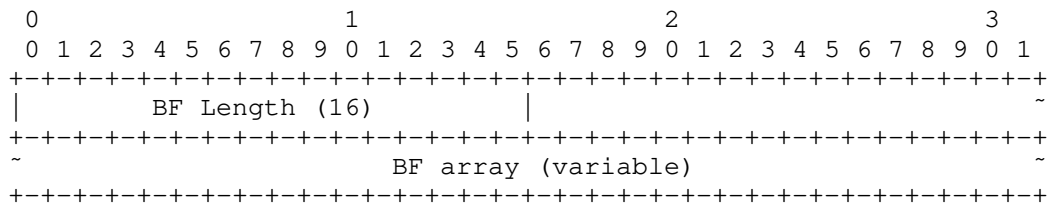


Figure 6: BF formatted bitmap.

The BF Length field contains the length of the BF-formatted bitmap that follows in bytes. The Length field is 16 bits wide to allow for flexible configuration to allow for less erroneous BF compression for large chunk groups by using longer binary arrays.

4. Open Issues

As the BF-based scheme is highly controllable and stable in terms of resource efficiency, it is viable to incorporate the contentmap information about each returned peer in the Peerlist without introducing considerable extra overhead. The current definition of PeerGroup element in the extended TP protocol needs to be extended to include "SegmentInfo" field as defined in ContentGroup element and used accordingly. However, the necessity of introducing other peer specific information into the TP protocol via Peerlist is still subject to discussion. In particular, these info exchange is far less frequent and accurate as compared to the peer protocols's HANDSHAKE and HAVE messages.

For the sake of simplicity, the above integration with tracker protocol does not consider the case of employing more than one bitmap compression algorithms for a single swarm. Meanwhile, it is noted that there are considerable content in the peer protocol dealing with chunk addressing schemes inter-working between peers employing different schemes. However, in BF-based scheme, as for the irreversible nature of hash functions, it is not feasible to translate a BF-formatted array back into an original bitmap. Therefore, in order to keep compatible and open to peers using other addressing schemes, it is required that there be an original bitmap maintained by the local peer to allow the ability of communicating with peers using another addressing scheme.

Even if the chunk addressing scheme can be uniquely specified in a MPD file for a give swarm, there is still possibility that a peer who cannot support the relevant addressing scheme comes along and contact the tracker for joining intention. Therefore, there is need for such configuration exchange channel via TP protocol to allow mismatch detection and error handling. A possible solution could be to add an algorithm identifier attribute in the "SwarmId" field, and keep the value definition for different schemes consistent with the peer protocol's "Chunking Addressing Method" field.

5. Security Considerations

TBA

6. IANA Considerations

None.

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

7.2. Informative References

- [BF-analysis] Broder, A. and M. Mitzenmacher, "Network applications of Bloom Filters: a survey", Internet Mathematics Vol. 1, No. 4, pp. 485-509, 2004.
- [BF-bloom] Bloom, B., "Space/time trade-offs in hash coding with allowable errors.", Communications of ACM Vol. 13, No. 7, pp. 422-426, 1970.
- [I-D.ietf-huang-extended-tracker-protocol] Huang, R., Zong, N., Cruz, R., Nunes, , and J. Taveira, "PPSP Tracker Protocol-Extended Protocol", draft-ietf-huang-extended-tracker-protocol-02 (work in progress), February 2013.
- [I-D.ietf-ietf-ppsp-peer-protocol] Bakker, A., Petrocco, R., and V. Grishchenko, "Peer-to-Peer Streaming Peer Protocol (PPSPP)", draft-ietf-ppsp-peer-protocol-06 (work in progress), February 2013.
- [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-26 (work in progress), February 2013.
- [I-D.ietf-ppsp-base-tracker-protocol] Cruz, R., Nunes, M., Gu, Y., Xia, J., and J. Taveira, "PPSP Tracker Protocol-Base Protocol (PPSP-TP/1.0)", draft-ietf-ppsp-base-tracker-protocol-00 (work in progress), February 2013.

[I-D.ietf-ppsp-problem-statement]

Zhang, Y. and N. Zong, "Problem Statement and Requirements of Peer-to-Peer Streaming Protocol (PPSP)", draft-ietf-ppsp-problem-statement-12 (work in progress), January 2013.

[P2P-limit]

Feng, C., Li, B., and B. Li, "Understanding the performance gap between pull-based mesh streaming protocols and fundamental limits", in Proc. of IEEE INFOCOM , 2009.

[RFC1321] Rivest, and Newport, "RFC 1321: The MD5 message-digest algorithm", April 1992.

Authors' Addresses

Lingli Deng
China Mobile

Email: denglingli@chinamobile.com

Jin Peng
China Mobile

Email: pengjin@chinamobile.com

Yunfei Zhang
CoolPad

Email: hishigh@gmail.com

Yihong Huang
Huawei

Email: rachel.huang@huawei.com

INTERNET-DRAFT
Intended Status: Informational
Expires: August 18, 2014

R. Huang
Huawei
Y. Zhang
CoolPad
February 14, 2014

Survey of WebRTC based P2P Streaming
draft-huang-ppsp-p2p-webrtc-survey-00

Abstract

This document presents a survey of current emerging WebRTC based P2P streaming applications available on the nowadays Internet.

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/lid-abstracts.html>

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

Copyright and License Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	3
2	Terminology	3
3	Traditional P2P Streaming Applications on Browsers	4
4	WebRTC based P2P Applications	5
4.1	Swarm CDN	5
4.2	PeerCDN	6
4.3	Sharefest	7
4.4	P2P Media Streaming with HTML5 and WebRTC	8
5	Discussion	9
5.1	Incorporating P2P Streaming in RTCWEB Scenarios	9
5.2	Open Issues	9
6	Security Considerations	10
7	IANA Considerations	10
8	Acknowledgments	10
9	References	10
9.1	Informative References	10
	Authors' Addresses	11

1 Introduction

P2P streaming applications usually use standard or private P2P protocol to deliver content between their end users. Some of them uses specific desktop software as the client which end users have to download and install in their OS. The others uses browsers as clients, but users still need to download corresponding plug-ins to enable the P2P transmission.

WebRTC is a new software architecture which enable web browsers with Real Time Communication (RTC) capabilities via simple JavaScript APIs. It provides direct interactive communications using audio, video, etc. between two peers' web-browsers without any additional plug-ins. Currently, WebRTC only focuses on conversational applications such as VoIP and video conferencing, multimedia streaming systems are not considered. However, the interests towards WebRTC is strongly increasing. An ever-increasing number of P2P applications have adopted WebRTC to distribute multimedia content or file sharing from a source to a large number of endpoints only via browsers without installing any plug-ins. This document presents a survey of current emerging WebRTC based P2P streaming applications available on the nowadays Internet.

the remainder of this document is organized as follows: Section 2 introduces terminology used throughout the survey; Section 3 provide the analysis of how traditional P2P streaming applications on browsers work; Section 4 presents current emerging WebRTC based P2P streaming applications.

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].

Plug-in: a piece of software which enhances another software application and usually cannot be run independently.

API: Application Programming Interface - a specification of a set of calls and events, usually tied to a programming language or an abstract formal specification such as WebIDL, with its defined semantics.

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

Tracker: A tracker refers to a directory service that maintains a

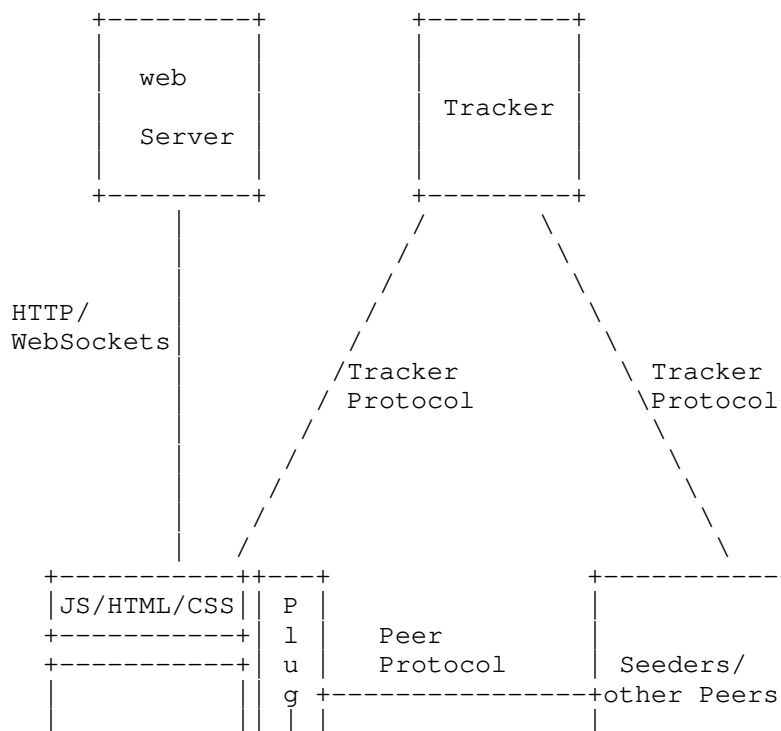
list of peers participating in a specific audio/video source or in the distribution of a streaming file. A tracker answers queries received from peers for the information of other peers sharing the same streaming content. A tracker is a logical component which could be centralized or distributed.

Tracker protocol: A signaling protocol between trackers and peers.

Peer protocol: A signaling and data protocol among the peers.

3 Traditional P2P Streaming Applications on Browsers

Some of traditional P2P streaming applications also allow their users to obtain their content via browsers. The browsers are required to install some plug-ins provided by these content providers firstly. After that, the browsers of end users download and run specific JavaScript application from content server. The JavaScript application will act as a peer of P2P network to connect to a tracker and get a list of available peers via tracker protocol. With the help of plug-ins, the JavaScript application will then be able to communicate with other peers exchanging content via standard or private peer protocol.



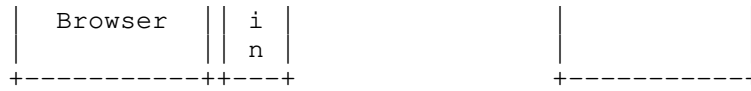


Figure 1 Traditional P2P Streaming Applications on Browsers

Adobe's Flash Player is a typical example of such usage. It has cooperated with many content providers, e.g. Sohu video, to support P2P video streaming. It also provides APIs to help developers to be able to use flash player to build various other P2P applications right within the browsers. But the browsers of these users will be required to install or update the plug-ins of Flash Player. Flash Player has provided the ability of peer-to-peer communication since version 10.1.

4 WebRTC based P2P Applications

WebRTC is a collection of technologies, including a media connection protocol (which is RTP) that can be used to initiate sessions like video conversation, a data exchange protocol that will enable users to transfer data, e.g. chat or files, without the need for a third-party server, and a way to access local resources, e.g. user's microphone and video camera. WebRTC is created to facilitate conversational communications from web browsers. However, it is not the only usage WebRTC turns out to be. Recently, some innovative applications and projects are using WebRTC technology on peer-to-peer connections to accomplish tasks like streaming and sharing, instead of relying on third-party plug-ins or proprietary software. They all rely on WebRTC's data channel protocol, which is designed to allow browsers to exchange data other than audio or video. Some of the typical WebRTC based P2P applications are presented as follow, but not exhaustive. Currently, the data channel protocol has been implemented in Chromes as well as Firefox, which means that current WebRTC based P2P applications are only supported in these two types of browsers.

4.1 Swarm CDN

Swarm CDN is a new peer-to-peer browser based content delivery network using WebRTC technology to save web site owners' money, reduce their bandwidth and reduce server costs. Instead of hosting the content on servers in data centers, Swarm CDN provides a way for users to deliver content to other users on the same site without installing any plug-ins.

Figure 2 demonstrates the rough architecture of this system.

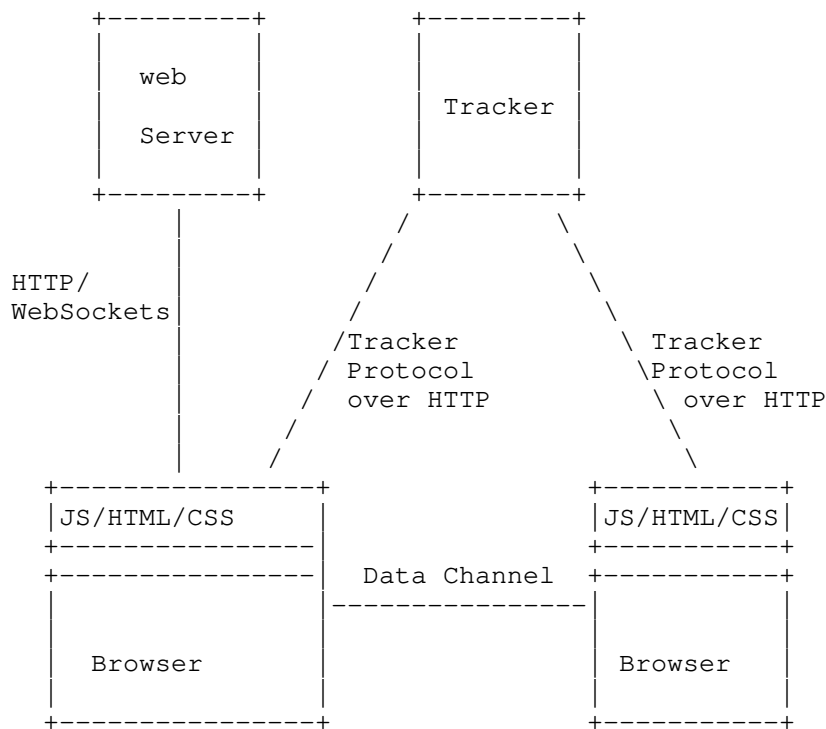


Figure 2 WebRTC based P2P Applications

Swarm CDN provides plug-ins to content providers or modifies their sites' HTML code so that peers made up of the users's browsers will be redirected to a centralized server which is called tracker in this memo. As depicted from the above figure, after the web page is loaded from the original web server, the peer will join the P2P network by establishing a HTTP connection to the centralized server which holds connections to a certain number of peers that have recently joined the network. Afterwards, this HTTP connection is used to signal the WebRTC handshake, resulting in a direct WebRTC data channel between the newly joined client and one of the other clients that has the content. For those users on a non compatible WebRTC browsers, the tracker will detect this and default them to retrieve the requested content from the original web servers.

Swarm CDN could provide service for video and images. Currently, it's free and able to swarm video and images for 20 simultaneous users.

4.2 PeerCDN

PeerCDN helps web applications build a peer-to-peer content delivery

network (CDN) that will make the web faster, more reliable, and help sites to reduce bandwidth costs. The technology of PeerCDN is similar to Swarm CDN, which uses WebRTC encrypted data channel for data transmission. It also requires users to insert a single line of HTML code to enable the usage without any browser plug-ins. Afterwards, a web browser triggers a notification to the peerCDN's tracking server that they are visiting the page. If someone else visits this page at the same time, peerCDN automatically sets up a connection and quietly streams the data between the peers. PeerCDN serves a site's static assets, such as images, streaming videos and file downloads. It only works on all WebRTC browsers.

PeerCDN was founded by Stanford engineering graduate in early 2013. And Yahoo has bought this startup company in December of 2013.

4.3 Sharefest

Sharefest is a file sharing application developed by a video streaming company called Peer5. Different from tools like Dropbox transferring files to the remote server then back again, Sharefest creates peer-to-peer communication between browsers directly using WebRTC.

Figure 3 depicts the architecture of Sharefest.

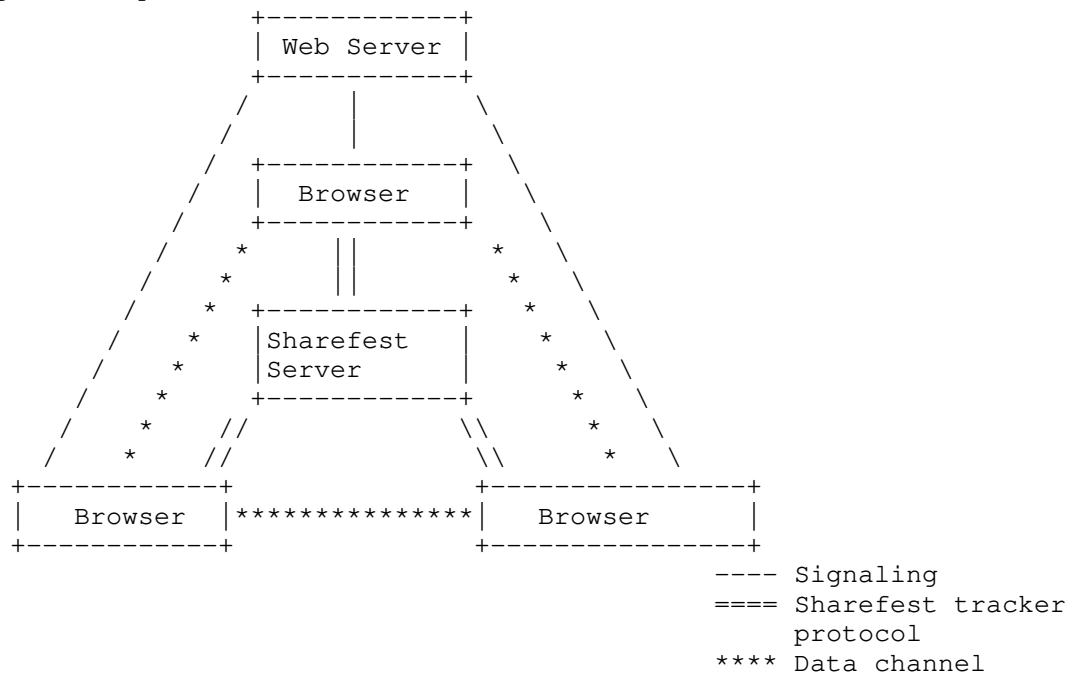


Figure 3 Architecture of Sharefest

The Sharefest server presents an auto-generated short-URL for each file shared by the client wishing to share. When a client browser visits the URL, the server connects it to the sharing client's browser and the sharing client's browser transfers the file over an WebRTC data channel. If multiple clients connect from different locations, the server attempts to pair nearby clients to each other to maximize the overall throughput. Sharefest utilizes some simple p2p technologies like slicing. It does its best to maximize efficiency by dividing each file into slices that the downloading peers exchange with each other.

Currently the size limit for shared files is around 1 GB and only the recent versions of Chrome and Firefox are supported.

4.4 P2P Media Streaming with HTML5 and WebRTC

Some students from Aalto university have proposed an experimental system for P2P VoD service applying WebRTC standards. The experiment mirrors the BitTorrent architecture and the same entities - tracker, seeder and peers - are used. Peers and seeders could be WebRTC browsers. It can be seen in Figure 4.

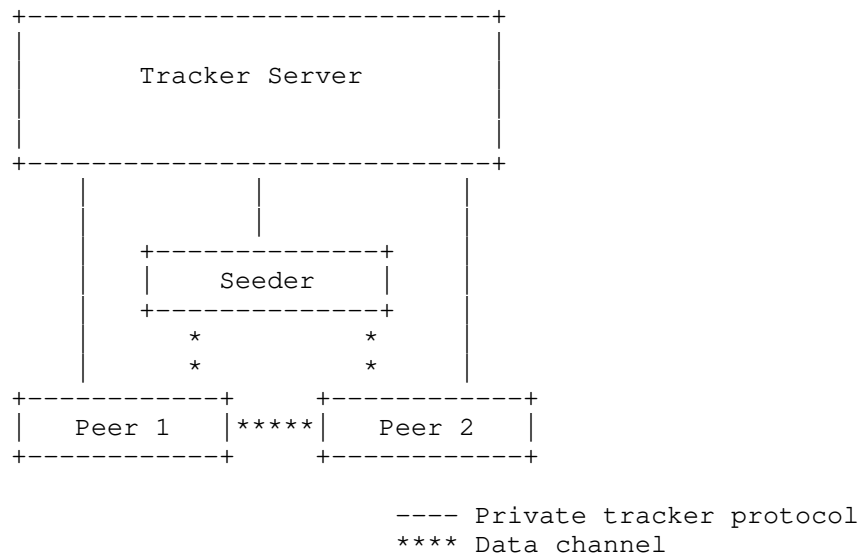


Figure 4 Network Architecture for the P2P VoD service with HTML5 and WebRTC

Initially users or content providers create a torrent file containing the metadata of the video, hashes of each piece of the video and uploads the torrent file to the tracker. Media content is uploaded to

the seeder simultaneously. When a peer requests this content, it will obtain the metadata from the tracker and start to download the content slices from the seeder or original peer uploading the content or both. As the content becomes popular, subsequent peers could get the content from the nearby peers, which decrease the load on the seeder and original peer uploading the content.

The experiment raises some considerations on this kind of implementations. The main concern is that the performance of MD5 on JavaScript and how much load could be handle in current solution generated. It points out that the resources available through browsers are much more limited than for native applications. The calculation of MD5 hashes use too much CPU consumption which will adversely affect the speed of getting content to play, especially for mobile devices. This experiment also indicates that handling two streams, incoming and outgoing, is manageable with current desktop and mobile devices. Increasing peers number will severely degrade performance and result in a bad user experience.

As the capability of computing for both desktop and mobile devices is rapidly increasing, it can be anticipated that the above problems will be eliminated soon.

5 Discussion

5.1 Incorporating P2P Streaming in RTCWEB Scenarios

As we have discussed in the above sections, there are more and more efforts on enabling p2p streaming using WebRTC technologies. According to the statistics of LTE usage in Verizon [Verizon], video streaming traffic is accounting the largest part of the LTE traffic and it slows down the wireless download speed by 20 percentages. Using p2p streaming is an efficient means to solve the traffic pressure not only in the core network, backbone but also in the air interface especially when Device to Device mode (D2D) is deployed between the browsers of end nodes, which is in the standardization process in 3GPP. Although RTCWEB is created to facilitate JavaScript APIs for real-time communication in the browser without implementing any plug-ins, the current scope of RTCWEB only includes conversational applications not streaming systems. We can see that web applications supporting p2p streaming without plug-ins are more and more appealing. So is it possible to incorporate P2P streaming in RTCWEB scenarios so as to implement JavaScript APIs for P2P algorithms as part of the WebRTC standardization process?

5.2 Open Issues

Several open issues need to be addressed for the purpose described in Section 5.1:

1. Do we need WebRTC to support P2P algorithms? If yes, how to support?
2. Do we need WebRTC to support the tracker protocol? If yes, how to support?
3. Do we need WebRTC to support the peer protocol? If yes, how to support?
4. Is there any security problems?

6 Security Considerations

This document has no security concern.

7 IANA Considerations

This document has no actions for IANA.

8 Acknowledgments

TBD.

9 References

9.1 Informative References

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

[SwarmCDN] Swarm CDN web site, <http://swarmcdn.com/>

[PeerCDN] Peer CDN web site, <https://peercdn.com/?>

[Sharefest] Sharefest web site, <http://www.sharefest.me/?>

[PPMSHW] J. Nurminen, A. Meyn, et al., "P2P Media Streaming with HTML5 and WebRTC", February 2013

[Verizon] <http://www.fiercewireless.com/tech/story/verizon-spending-millions-network-upgrades-counter-growth-video-traffic/2013-12-16>

Authors' Addresses

Rachel Huang
Huawei
Phone: +86-25-56623633

EMail: rachel.huang@huawei.com

Yunfei Zhang
Coolpad

EMail: hishigh@gmail.com

PPSP
INTERNET-DRAFT
Intended Status: Standards Track
Expires: July 4, 2014

Rui S. Cruz
Mario S. Nunes
IST/INESC-ID/INOV
Yingjie Gu
Jinwei Xia
Huawei
Joao P. Taveira
IST/INOV
Deng Lingli
China Mobile
December 31, 2013

PPSP Tracker Protocol-Base Protocol (PPSP-TP/1.0)
draft-ietf-ppsp-base-tracker-protocol-03

Abstract

This document specifies the base Peer-to-Peer Streaming Protocol-Tracker Protocol (PPSP-TP/1.0), an application-layer control (signaling) protocol for the exchange of meta information between trackers and peers. 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 delivery (audio, video, associated timed text and metadata), such as adaptive multi-rate, layered (scalable) and multi-view (3D) videos, 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/lid-abstracts.html>

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

Copyright and License Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1	Introduction	5
1.1	Terminology	5
2	Operation and Protocol Architecture Overview	8
2.1	Operation	8
2.2	Enrollment and Bootstrap	9
2.3	Architectural and Functional View	11
2.3.1	Messaging Model	11
2.3.2	Request/Response model	11
2.4	State Machines and Flows of the Protocol	13
2.4.1	Normal Operation	15
2.4.2	Error Conditions	15
3	Protocol Specification	17
3.1	Request/Response Syntax and Format	17
3.2	Request element in request Messages	22
3.3		

Table of Contents

1	Introduction	6
1.1	Terminology	6
2	Operation and Protocol Architecture Overview	9
2.1	Operation	9
2.2	Enrollment and Bootstrap	10
2.3	Architectural and Functional View	12
2.3.1	Messaging Model	12
2.3.2	Request/Response model	13
2.4	State Machines and Flows of the Protocol	14
2.4.1	Normal Operation	16
2.4.2	Error Conditions	17
3	Protocol Specification	18
3.1	Request/Response Syntax and Semantics	18
3.2	Request element in request Messages	22
3.3	Response element in response Messages	23
4	Request/Response Processing	23
4.1	CONNECT Request	24
4.2	FIND Request	26
4.3	STAT_REPORT Request	27
4.4	Error and Recovery conditions	27
5	Operations and Manageability	29
5.1	Operational Considerations	29
5.1.1	Installation and Initial Setup	29
5.1.2	Migration Path	30
5.1.3	Requirements on Other Protocols and Functional Components	30
5.1.4	Impact on Network Operation	30
5.1.5	Verifying Correct Operation	30
5.2	Management Considerations	30

5.2.1	Interoperability	30
5.2.2	Management Information	31
5.2.3	Fault Management	31
5.2.4	Configuration Management	31
5.2.5	Accounting Management	31
5.2.6	Performance Management	32
5.2.7	Security Management	32
6	Security Considerations	33
6.1	Authentication between Tracker and Peers	33
6.2	Content Integrity protection against polluting peers/trackers	33
6.3	Residual attacks and mitigation	34
6.4	Pro-incentive parameter trustfulness	34
7	Guidelines for Extending PPSP-TP	35
7.1	Forms of PPSP-TP Extension	35
7.2	Issues to Be Addressed in PPSP-TP Extensions	36
8	IANA Considerations	37
9	Acknowledgments	37
10	References	38
10.1	Normative References	38
10.2	Informative References	39
	Appendix A. Revision History	42
	Appendix B. PPSP-TP Message Syntax for HTTP/1.1	43
B.1	Header Fields	44
B.2	Methods	44
B.3	Message Bodies	45
B.4	Message Response Codes	45
	Appendix C. Use Scenarios	47
C.1	Additional Terminology	47
C.2	Functional Entities	48
C.3	Streaming Capabilities	49
C.4	NAT Traversal	49
C.5	Content Information Metadata	49
C.6	Authentication, Confidentiality, Integrity	50
	Appendix D. Implementation Options	50
	Authors' Addresses	51
	Response element in response Messages	22 4
	Request/Response Processing	24
4.1	CONNECT Request	24
4.2	FIND Request	28
4.3	STAT_REPORT Request	30
4.4	Error and Recovery conditions	31 5
	Operations and Manageability	33
5.1	Operational Considerations	33
5.1.1	Installation and Initial Setup	33
5.1.2	Migration Path	34
5.1.3	Requirements on Other Protocols and Functional	

Components	34	5.1.4
Impact on Network Operation	34	5.1.5
Verifying Correct Operation	34	5.2
Management Considerations	34	
5.2.1 Interoperability	34	
5.2.2 Management Information	35	
5.2.3 Fault Management	35	
5.2.4 Configuration Management	35	
5.2.5 Accounting Management	35	
5.2.6 Performance Management	36	
5.2.7 Security Management	36	6
Security Considerations	37	
6.1 Authentication between Tracker and Peers	37	
6.2 Content Integrity protection against polluting peers/trackers	37	6.3
Residual attacks and mitigation	38	6.4
Pro-incentive parameter trustfulness	38	7
Guidelines for Extending PPSP-TP	39	
7.1 Forms of PPSP-TP Extension	39	
7.2 Issues to Be Addressed in PPSP-TP Extensions	41	
8 IANA Considerations	43	
9 Acknowledgments	43	
10 References	44	
10.1 Normative References	44	
10.2 Informative References	44	
Appendix A. Revision History	47	
Appendix B. PPSP-TP Message Syntax for HTTP/1.1	48	
B.1 Header Fields	49	
B.2 Methods	49	
B.3 Message Bodies	50	
B.4 Message Response Codes	50	
Appendix C. Use Scenarios	52	
C.1 Additional Terminology	52	
C.2 Functional Entities	53	
C.3 Streaming Capabilities	54	
C.4 NAT Traversal	54	
C.5 Content Information Metadata	54	
C.6 Authentication, Confidentiality, Integrity	55	
Appendix D. Implementation Options	55	
Authors' Addresses	56	

1 Introduction

The Peer-to-Peer Streaming Protocol (PPSP) is composed of two protocols: the PPSP Tracker Protocol and the PPSP Peer Protocol [RFC6972] specifies that the Tracker Protocol should standardize the messages between PPSP peers and PPSP trackers and also 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.

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

This document describes the base PPSP Tracker protocol and how it satisfies the requirements for the IETF Peer-to-Peer Streaming Protocol, 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 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].

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.

CHUNK: A Chunk is a basic unit of data organized in P2P streaming for storage, scheduling, advertisement and exchange among peers.

CHUNK ID: A unique resource identifier for a SEGMENT CHUNK. The identifier type depends on the addressing scheme used, i.e., an integer, an HTTP-URL and possibly a byte-range, and is described in the MPD.

CONNECTION TRACKER: The node running the tracker service to which the PPSP peer will connect when it wants to get registered and join

the PPSP system.

LEECHER: A Peer that has not yet completed the transfer of all Chunks of the media content.

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.

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 joined. This value indicates the stability of a peer, and can be 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 caches and streams streaming content to other participants.

PEER ID: The identifier of a PEER such that other PEERs, or the TRACKER, can refer to the PEER by using its ID. The Peer ID is mandatory, can take the form of a universal unique identifier (UUID), defined in [RFC4122], and can be bound to a network address of the peer, i.e., an IP address, or a uniform resource identifier/locator (URI/URL) that uniquely identifies the corresponding peer in the network. The Peer ID and any required security certificates are obtained from an offline enrollment server.

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.

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.

PPSP-TP: The abbreviation of Peer-to-Peer Streaming Protocols - Tracker Protocol.

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.

SEEDER: A Peer that holds and shares the complete media content.

SEGMENT: 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.

SEGMENT CHUNK: For Structured Media contents, a segment may correspond to a set of nested and dependent CHUNKs, a set of correlated additive CHUNKs or a set of alternate REPRESENTATION CHUNKs.

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. The Swarm-ID may use a universal unique identifier (UUID), e.g., a 64 or 128 bit datum to refer to the content resource being shared among peers.

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.

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.

TRANSACTION ID: The identifier of a REQUEST from the PEER to the TRACKER. Used to disambiguate RESPONSES that may arrive in a different order of the corresponding REQUESTs.

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

2 Operation and Protocol Architecture Overview

2.1 Operation

The functional entities related to PPSP protocols are the Client Media Player, the service Portal, the tracker and the peers. The complete description of Client Media Player and service Portal is not discussed here, as not in the scope the specification. The functional entities directly involved in the PPSP Tracker Protocol are trackers and peers (which may support different capabilities).

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.

A Peer corresponds to a logical entity (typically in a user device) that actually participates in sharing a media content. Peers are organized in (various) swarms corresponding each swarm to the group of peers streaming a certain content at any given time.

The Tracker is a logical entity that maintains the lists of peers storing segments for a specific Live media channel or on-demand 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 on-demand or Live streaming) with peers, but information about which peers can provide the content.

When a peer wants to receive streaming of a selected content (Leech mode):

1. Peer connects to a connection tracker and joins a swarm.
2. Peer acquires a list of other peers in the swarm from the connection tracker.
3. Peer exchanges its content availability with the peers on the obtained peer list (the 3~5 steps are in the peer protocol).

4. Peer identifies the peers with desired content.
5. Peer requests content from the identified peers.

When a peer wants to share streaming contents (Seeder mode) with other peers:

1. Peer connects to the connection tracker.
2. Peer sends information to the connection tracker about the swarms it belongs to (joined swarms).

After having been disconnected due to some termination condition, a peer can resume previous activity by connecting and re-joining the corresponding swarm(s).

2.2 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 the following (not exhaustive) typical methods (illustrated in Figures 1 and 2):

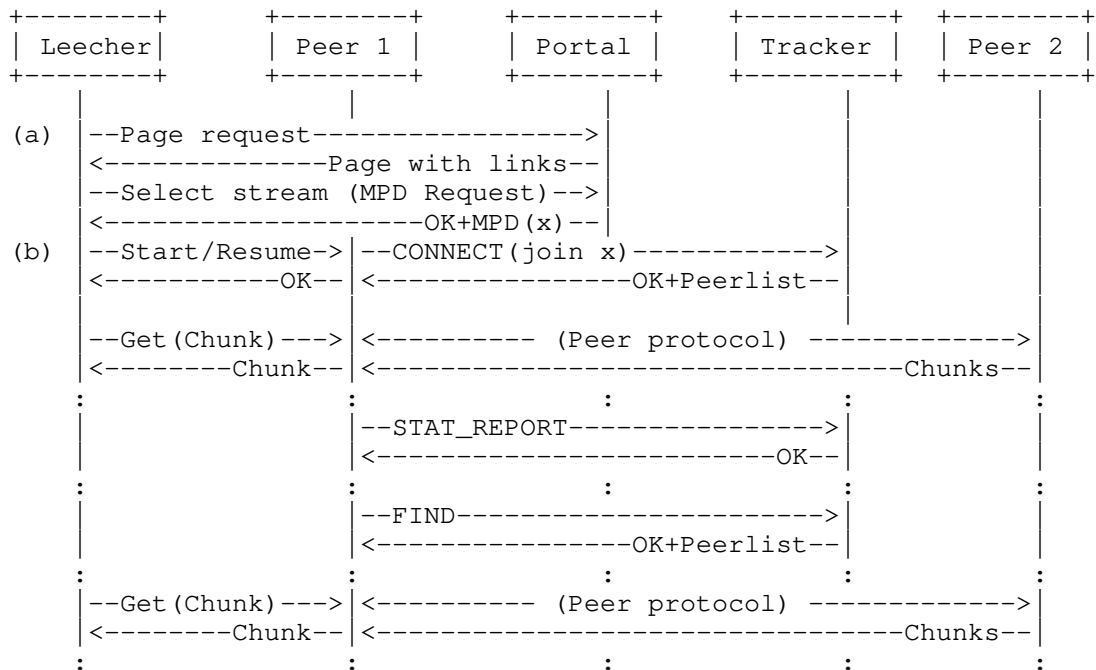


Figure 1: A typical PPSP session for a streaming Leecher.

A leecher logs on the portal to select the swarm it are interested, then the portal return the Media Presentation Description (MPD) file of the selected swarm to the leecher. The MPD includes the 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 (e.g., content x).

Then the leecher starts to initiate a PPSP-TP CONNECT message to the tracker, or resume a previously initiated swarm at point (b). Tracker will return OK code with the peer list to the leecher if the CONNECT message is successfully accepted.

Once the CONNECT is successfully accepted, the leecher needs to report its status and statisc data to the tracker periodically.

If the leecher plan to switch to another straw, it will initiate a FIND message to the tracker to ask the information of new swarm.

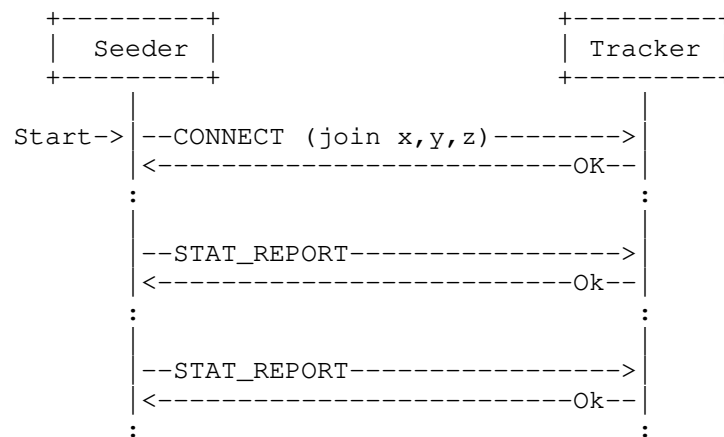


Figure 2: A typical PPSP session for a streaming Seeder.

For the seeder usually located at edge caches and/or Media Servers, the swarms it can provide are reported to tracker through CONNECT message.

In order to be able to bootstrap, either leecher or seeder must first obtain a Peer ID (identifier of the peer) and any required security certificates or authorization tokens from an enrollment service (end user registration). How the peer obtain above information is not in the scope of this document.

Peer may support different encoding type (e.g., text or binary), in which case, the portal needs to select an appropriate tracker supporting the same encoding type as the peer. How the portal learn the encoding type supported by the peer is not in the scope of this document.

The specification of the mechanisms used by the Client Media Player and the peer (to signal start/resume streams or request media chunks), obtain a Peer ID, security certificates or tokens are not in the scope of this document.

2.3 Architectural and Functional View

The PPSP Tracker Protocol architecture is intended to be compatible with the web infrastructure. PPSP-TP is designed with a layered approach i.e., a PPSP-TP Request/Response layer, a Message layer and a Transport layer. The PPSP-TP Request/Response layer deals with the interactions between tracker and peers using Request and Response codes (see Figure 3).

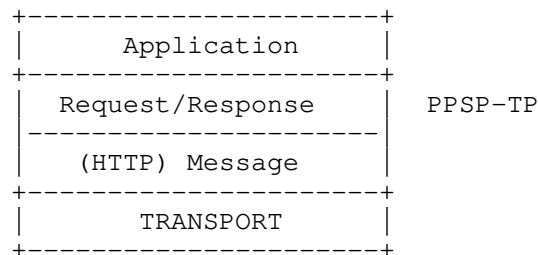


Figure 3: Abstract layering of PPSP-TP.

The Message layer deals with the framing format, for encoding and transmitting the data through the underlying transport protocol, as well as the asynchronous nature of the interactions between tracker and peers.

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 message when using a connection-oriented transport like TCP [RFC0793], or TLS [RFC5246] over it.

2.3.1 Messaging Model

The messaging model of PPSP-TP aligns with HTTP protocol and the semantics of its messages, currently in version 1.1 [RFC2616], but intended to support future versions of HTTP. The exchange of

messages of PPSP-TP is envisioned to be performed over a stream-oriented reliable transport protocol, like TCP [RFC0793].

Appendix B describes the message syntax when using HTTP/1.1.

2.3.2 Request/Response model

PPSP-TP messages are either requests from peers to a tracker service, or responses from a tracker service to peers. The Request and Response semantics are carried as entities (header and body) in messages which correspond to either HTTP request methods or HTTP 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 response codes, however, not all HTTP response codes are used for the PPSP-TP (section 3 and Appendix B.4).

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

PPSP-TP/1.0 Req. Messages
CONNECT
FIND
STAT_REPORT

Table 1: Request Messages

CONNECT: This Request message is an "action signal" used when a peer registers in the tracker (or if already registered) to notify it about the participation in named swarm(s). The tracker records the Peer ID, connect-time (referenced to the absolute time), peer IP addresses (and associated location information), link status and Peer Mode for the named swarm(s). The tracker also changes the content availability of the valid named swarm(s), i.e., changes the peers lists of the corresponding swarm(s) for the requester Peer ID. On receiving a CONNECT message, the tracker first checks the peer mode type (SEED/LEECH) for the specified swarm(s) and then decides the next steps (more details are referred in section 4.1)

FIND: This Request message is an "action signal" used by peers to request to the tracker, whenever needed, a list of peers active in the named swarm. 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 is an "information signal" that allows an active peer to send status (and optionally statistic data) to the tracker to signal continuing activity. This request message **MUST** be sent periodically to the tracker while the peer is active in the system.

2.4 State Machines and Flows of the Protocol

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.

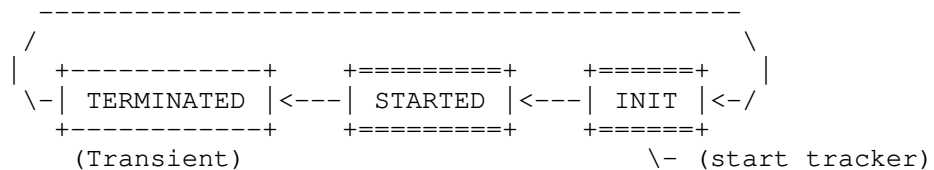


Figure 4: Tracker State Machine

When there are no peers connected in the tracker, the state machine is in the INIT state.

When the "first" peer connects for registration 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.

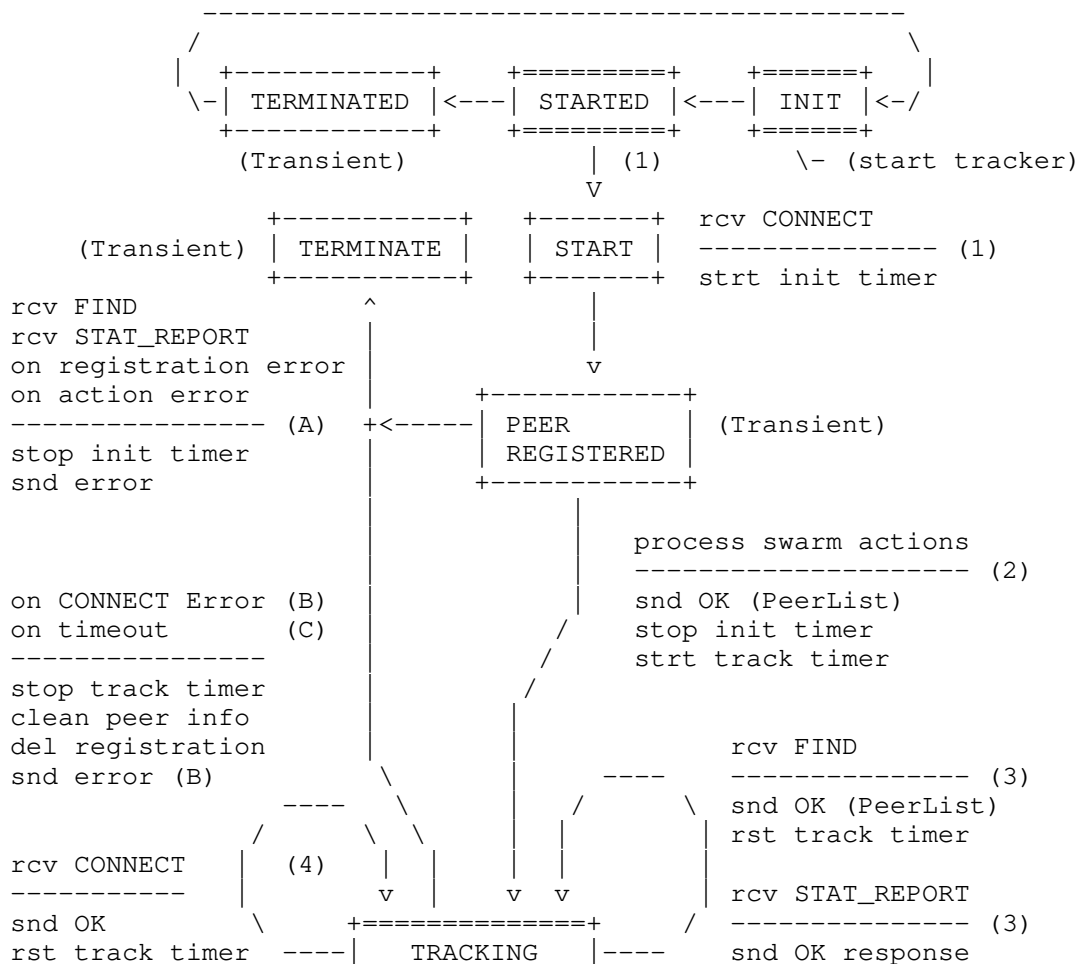
In addition to the tracker state machine, each peer is modeled with its own transaction state machine (Figure 5), instantiated per Peer ID in the tracker, and deleted when it is removed.

Unlike the tracker state machine, which exists even when no Peer IDs

are registered, the "per-Peer-ID" transaction state machine is instantiated only when the Peer ID starts registration in the tracker, and is deleted when the Peer ID is de-registered/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).

When a new Peer ID is added, the corresponding "per-Peer-ID" state machine 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.



+=====+ rst track timer

Figure 5: Per-Peer-ID Transaction State Machine and Flow Diagram

During the lifetime 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:

- o Reception of CONNECT, FIND and STAT_REPORT messages, or
- o 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.

2.4.1 Normal Operation

On normal operation the process consists of the following steps:

- 1) When a Peer wants to access the system it needs to register on a tracker by sending a CONNECT message asking for the swarm(s) it wants to join. This CONNECT request from a new Peer ID triggers the instantiation of a "per-Peer-ID" State Machine. In the START state of the new "per-Peer-ID" SM, the tracker initiates the registration of the Peer ID and associated information (IP addresses), starts the "init timer" and moves to PEER REGISTERED state.
- 2) In PEER REGISTERED state, if Peer ID is valid, the tracker either
 - a) processes the requested action(s) for the valid swarm information contained in the CONNECT request and in case of success the tracker stops the "init timer", starts the "track timer" and sends the response to the peer (the response MAY contain the appropriate list of peers for the joining swarm(s), as detailed in section 4.1, or
 - b) moves the valid FIND request to TRACKING state.
- 3) In TRACKING state, STAT_REPORT or FIND messages received from that Peer ID will reset the "track timer" and are respectively responded with a) a successful condition, b) a successful condition containing the appropriate list of peers for the named swarm (section 4.2).
- 4) While TRACKING, a CONNECT message received from that Peer ID with valid swarm actions information (section 4.1) resets the "track timer" and is responded with a successful condition.

2.4.2 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 PEER REGISTERED state, when a CONNECT Request only contains invalid swarm actions (section 4.1), the tracker responds with error code 403 Forbidden, deletes the registration, transition to TERMINATE state for that Peer ID and the SM is destroyed.

At the PEER REGISTERED state, if the Peer ID is considered invalid (in the case of a CONNECT request or in the case of FIND or STAT_REPORT requests received from an unregistered Peer ID), the tracker responds with either error codes 401 Unauthorized or 403 Forbidden (described in section 4.4), transitions to TERMINATE state for that Peer ID and the SM is destroyed.

- B) At the TRACKING state (while the "track timer" has not expired) receiving a CONNECT message from that Peer ID with invalid swarm actions (section 4.1) is considered an error condition. The tracker responds with error code 403 Forbidden (described in section 3), stops the "track timer", deletes the registration, transitions to TERMINATE state for that Peer ID and the SM is destroyed.
- C) In TRACKING state, without receiving messages from the peer, on timeout (track timer) the tracker cleans all the information associated with the Peer ID in all swarms it was joined, deletes the registration, transitions to TERMINATE state for that Peer ID and the SM is destroyed.

NOTE: These situations may correspond to a malfunction at the peer or to malicious conditions. Therefore, as preventive measure, the tracker proceeds to TERMINATE state for the Peer ID by de-registering the peer and cleaning all peer information.

3 Protocol Specification

3.1 Request/Response Syntax and Semantics

PPSP-TP is a message-oriented request/response protocol. The messages could be encoded using binary type or text type, which can be indicated in the Content-Type field in HTTP/1.1 [RFC2616], but is not in the scope of this draft. In this draft, only the syntax and format of the PPSP-TP messages are defined, at here, a C-like syntax based on the presentation language used to defined TLS [RFC5246] is recommended to present the message structures.

The generic format of a Request is the following:

```
struct {
    uint8          request_type;
    uint64         transaction_id;
    PeerInfo       peer_information;
    select (request_type){
        case CONNECT :
            uint16          swarm_length;
            SwarmInfo       swarm_information[swarm_length];
            uint16          peer_number;
            PeerAttr        peer_attribute;

        case FIND :
            uint16          swarm_length;
            SwarmInfo       swarm_information[swarm_length];
            uint16          peer_number;
            PeerAttr        peer_attribute;

        case STAT_REPORT :
            uint16          swarm_length;
            StatisticsGroup statistics_list[swarm_length];
    }
} PPSP_TP_Req;

struct {
    uint64          peer_id_low;
    uint64          peer_id_high;
    IPv4AddrPort    v4addr_port;
    IPv6AddrPort    v6addr_port;
} PeerInfo;

struct {
    uint32          addr;
    uint16          port;
    uint8           priority
}
```

```
        uint8      type; /* REFLEXIVE or PROXY */
        uint8      connection; /* 3G, ADSL etc */
        uint32     asn;
    } IPv4AddrPort;

    struct {
        uint128     addr;
        uint16      port;
        uint8       priority
        uint8       type; /* REFLEXIVE or PROXY */
        uint8       connection; /* 3G, ADSL etc */
        uint32     asn;
    } IPv6AddrPort;

    struct {
        uint64      swarm_id_low;
        uint64      swarm_id_high;
        uint8       action_type; /* JOIN or LEAVE */
        uint8       peer_mode; /* LEECH or SEED */
    } SwarmInfo;

    struct {
        uint8       abilityNAT;
        uint8       concurrentLinks;
        uint8       onlineTime;
        uint8       uploadBWlevel;
    } PeerAttr;

    struct {
        uint64      swarm_id_low;
        uint64      swarm_id_high;
        uint64      uploadedBytes;
        uint64      downloadedBytes;
        uint64      availBandwidth;
    } StatisticsGroup;
```

The Request element MUST be present in requests and corresponds to the request method type for the message. The request type includes CONNECT, FIND and STAT_REPORT.

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

All Request messages MUST contain a Peer_Info element to uniquely identify the requesting peer in the network. It includes the Peer_ID, IP addresses and ports, with a few of optional attributes.

The Peer_ID information may be present on the following levels:

- The identifier of the requesting peer on PPSP-TP level.
- The identifier of the candidate peer from tracker on PeerGroup level.

The peer also MAY include some attributes:

- Priority: the preference of IP address on which the requesting peer get the swarm.
- Type: Describes the address for NAT traversal, which can be HOST REFLEXIVE or PROXY.
- Connection: Access type (3G, ADSL, etc.).
- ASN: Autonomous System Number.

If STUN-like functions are enabled in the tracker and a PPSP-ICE method [RFC5245] is used the attributes type and priority MUST be returned with the transport address candidates in responses to CONNECT 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.

The Swarm_Info element includes the Swarm_ID and the actions on the specific swarm.

The element Swarm_ID MUST be present in requests to identify the actions to be taken to the specified swarms. It MUST be present in CONNECT and FIND Requests and SHOULD be present in STAT_REPORT Requests on the following levels:

- The identifier of the swarm requested by peer On PPSP-TP level.
- The identifier of the swarm whose statistics are be reported to tracker on StatisticsGroup level.

The swarm also include a couple of attributes:

- Action: This behavior MUST be set to JOIN or LEAVE.

- Peer_Mode: Mode of Peer participation in the swarm MUST be set to LEECH or SEED.

The Peer_Number element MAY be present in CONNECT and FIND requests and MAY contain the Peer_Attribute to inform the tracker the numbers of the candidate peers the requesting peer hope to get, followed by the optional attributes of the candidate peers:

- AbilityNAT: Type of NAT traversal peers, as No-NAT, STUN, TURN or PROXY.
- ConcurrentLinks: Concurrent connectivity level of peers, HIGH, LOW or NORMAL.
- OnlineTime: Availability or online duration of peers, HIGH or NORMAL.
- UploadBWlevel: Upload bandwidth capability of peers, HIGH or NORMAL.

The StatisticsGroup element MAY be present in STAT_REPORT requests. Several properties relevant to P2P network are presented:

- Swarm_ID: To identify the specified swarms whose properties need to be reported.
- UploadedBytes: Bytes sent to swarm.
- DownloadedBytes: Bytes received from swarm.
- AvailBandwidth: Upstream Bandwidth available.

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 sibling elements and new attributes (guidelines in section 7).

The generic format of a Response is the following:

```

struct {
    uint8      response_code;
    uint64     transaction_id;
    select (request_type){
        case CONNECT :
            PeerInfo      peer_information;
            PeerGroup     peer_group;

        case FIND :
            PeerInfo      peer_information;
            PeerGroup     peer_group;
    } PPSP_TP_Res;

    struct {
        uint16      swarm_length;
        SwarmGroup  swarm_group[swarm_length];
    } PeerGroup;

    struct {
        uint64      swarm_id_low;
        uint64      swarm_id_high;
        uint16      peer_list_length;
        PeerInfo    peer_list[peer_list_length];
    } SwarmGroup;

```

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

The element Transaction_ID 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 PeerGroup element MAY be present in Response to CONNECT and FIND requests, and MAY contain the peer list for the requested swarm.

Request and Response processing is provided in section 4 for each message.

3.2 Request element in request Messages

Table 6 defines the valid string representations for the requests. These values MUST be treated as case-sensitive.

```

+-----+
| Request Types |
+-----+

```

CONNECT	0
FIND	1
STAT_REPORT	2

Table 6: Request Type of PPSP-TP requests.

3.3 Response element in response Messages

Table 7 defines the valid code for Response messages that require message-body.

Response messages not requiring message-body only use the standard HTTP Status-Code and Reason-Phrase (appended, if appropriate, with detail phrase, as described in section 4.4).

Response Code	HTTP Status-Code and Reason-Phrase
SUCCESSFUL 0	200 OK
Error_AUTHENTICATION 1	401 Unauthorized

Table 7: Valid Codes 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.

To CONNECT Request: returns information about the successful registration of the peer and/or of each swarm to which Action element requested. **MAY** additionally return the list of peers corresponding to the JOIN attribute requested.

To FIND Request: returns the list of peers corresponding to the requested swarm.

To STAT_REPORT Request: confirms the success of the requested operation.

Error_AUTHENTICATION: request is denied by the tracker.

4 Request/Response Processing

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.

4.1 CONNECT Request

This method is used when a peer registers to the system and/or requests swarm actions.

The peer MUST properly set the Request type to CONNECT, generate and set the TransactionIDs, set the PeerInfo and MAY include the swarm the peer is interested in, followed by the corresponding action_type and peer_mode.

The following example of the message-body of a CONNECT Request corresponds to a peer that wants to start (or re-start) sharing its previously streamed contents (peerMode is of SEED). Note for this case that the peer also requests from the Tracker an appropriate list of peers (PeerNum element) already active in the swarm, i.e., a list of 15 peers having STUN capabilities in terms of NAT. In the case of a Super-Node peer of an ISP, the CONNECT request would be similar but, optionally not including the PeerNum element:

When a peer agrees to share its previous swarm to others, it should set action_type to JOIN to the swarm, as well set the peer_mode to SEED during its start (or re-start)period.

When a peer requests to join a swarm, it should set action_type to JOIN to the swarm, as well set the peer_mode to LEECH during its start (or re-start)period.

In above two cases, the peers could provide optional information on the addresses of its network interface(s), for example, the priority, type, connection and ASN.

When a peer plans to leave a previously joined swarm, it should set action_type to LEAVE to the swarm regardless of the peer_mode.

When receiving a well-formed CONNECT Request message, the tracker MAY, when applicable, start by pre-processing 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, then proceed to register the peer in the service and perform the swarm actions requested. In case of success a Response message with a corresponding response value of SUCCESSFUL will be generated.

The valid sets of SwarmID whose action_type is combined with peer_mode for the CONNECT Request logic are enumerated in Table 8

(referring to the tracker "per-Peer-ID" state machine in section 2.4).

SwarmID Elements	@peerMode value	@action value	Initial State	Final State	Request validity
1	LEECH	JOIN	START	TRACKING	Valid
1	LEECH	LEAVE	START	TERMINATE	Invalid
1	LEECH	LEAVE	TRACKING	TERMINATE	Valid
1 1	LEECH LEECH	JOIN LEAVE	START	TERMINATE	Invalid
1 1	LEECH LEECH	JOIN LEAVE	TRACKING	TRACKING	Valid
N	SEED	JOIN	START	TRACKING	Valid
N	SEED	JOIN	TRACKING	TERMINATE	Invalid
N	SEED	LEAVE	TRACKING	TERMINATE	Valid

Table 8: Validity of action `_type` and peer_mode combinations

The element `PeerInfo`, if present, 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 the peer wants to advertise.

The element `peer_number` indicates 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. If STUN-like function is enabled in the tracker, the response MAY include the peer reflexive address.

The response MUST have the same `TransactionID` values as the corresponding request and actions.

When a peer's request to join a swarm as leecher was accepted by the tracker, the Response message tracker will search out the list of peers for the swarm and select an appropriate peer list satisfying

the conditions set by the requesting peer besides of the SUCCESSFUL response_code. The peer list returned MUST contain the Peer IDs and the corresponding IP Addresses, 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 of the referenced peer.

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].

When a peer's request to join a swarm as seeder was accepted by the tracker, the tracker responds with a SUCCESSFUL response_code and enters the peer information into the corresponding swarm activity.

IMPLEMENTATION NOTE: If no PeerNum attributes are present in the request the tracker MAY return a random sample from the peer population.

4.2 FIND Request

This method allows peers to request to the tracker, whenever needed, a new peer list for the swarm.

The FIND request MAY include a peer_number element to indicate to the tracker the maximum number of peers to be returned in a list corresponding to the indicated conditions set by the requesting peer, 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.

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 SUCCESSFUL will be generated and the tracker will search out the list of peers for the swarm and select an appropriate peer list satisfying the conditions set by the requesting peer. 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].

To provide more choices for the requesting peer, the tracker may

select a new peer list with lower priority from the list of peers and return it to the requesting peer later.

The peer list MUST contain the Peer IDs and the corresponding IP Addresses, 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 of the referenced peer.

IMPLEMENTATION NOTE: If no PeerNum attributes are present in the request the tracker MAY return a random sample from the peer population.

4.3 STAT_REPORT Request

This method allows peers to send status and statistic data to trackers. The method is initiated by the peer, periodically while active.

The report MAY include a StatisticsGroup containing multiple statistics elements describing several properties relevant to a specific swarm. These properties can be related with stream statistics and peer status information, including ploadedBytes, DownloadedBytes, AvailBandwidth and etc.

Other properties may be defined (guidelines in section 7.1) related for example, with incentives and reputation mechanisms. In case no StatisticsGroup is included, the STAT_REPORT is used as a "keep-alive" message to prevent the tracker from de-registering the peer when track timer expires.

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

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

4.4 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 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 2.4.2) 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 received 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 2.4.2). 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, "Action not allowed".

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.

5 Operations and Manageability

This section provides the operational and managements aspects that are required to be considered in implementations of the PPSP Tracker Protocol. These aspects follow the recommendations expressed in RFC 5706 [RFC5706].

5.1 Operational Considerations

The PPSP-TP provides communication between trackers and peers and is conceived as a "client-server" mechanism, allowing the exchange of information about the participant peers sharing multimedia streaming contents.

The "serving" component, i.e., the tracker, is a logical entity that can be envisioned as a centralized service (implemented in one or more physical nodes), or a fully distributed service.

The "client" component can be implemented at each peer participating in the streaming of contents.

5.1.1 Installation and Initial Setup

Content providers wishing to use PPSP for content distribution should setup at least a PPSP Tracker and a service Portal (public web server) to publish links of the content descriptions, for access to their on-demand or live original contents sources. Content/Service providers should also create conditions to generate PEER IDs and any required security certificates, as well as CHUNK IDs and SWARM IDs for each streaming content. The configuration processes for the PPSP Tracking facility, the service Portal and content sources are not standardized, enabling all the flexibility for implementers.

The SWARM IDs of available contents, as well as the addresses of the PPSP Tracking facility, can be distributed to end-users in various ways, but it is common practice to include both the SWARM ID and the corresponding PPSP Tracker addresses (as URLs) in the MPD of the content, which is obtainable (a link) from the service Portal.

End-users browse and search for the desired contents in the service Portal, selecting by clicking the links of the corresponding MPDs. This action typically launches the Client Media Player (with PPSP awareness) which will then, using PPSP-TP, contact the PPSP Tracker to join the corresponding swarm and obtain the transport addresses of other PPSP peers in order to start streaming the content.

5.1.2 Migration Path

Since there is no previous standard protocol providing similar functionality, this specification does not details a migration path.

5.1.3 Requirements on Other Protocols and Functional Components

For security reasons, when using PPSP Peer protocol with PPSP-TP, the mechanisms described in section 6.1 should be observed.

5.1.4 Impact on Network Operation

As the messaging model of PPSP-TP aligns with HTTP protocol and the semantics of its messages, the impact on Network Operation is similar to using HTTP.

5.1.5 Verifying Correct Operation

The correct operation of PPSP-TP can be verified both at the Tracker and at the peer by logging the behavior of PPSP-TP. Additionally, the PPSP Tracker collects the status of the peers including peer's activity, and such information can be used to monitor and obtain the global view of the operation.

5.2 Management Considerations

The management considerations for PPSP-TP are similar to other solutions using HTTP for large-scale content distribution. The PPSP Tracker can be realized by geographically distributed tracker nodes or multiple server nodes in a data center. As these nodes are akin to WWW nodes, their configuration procedures, detection of faults, measurement of performance, usage accounting and security measures can be achieved by standard solutions and facilities.

5.2.1 Interoperability

Interoperability refers to allowing information sharing and operations between multiple devices and multiple management applications. For PPSP-TP, distinct types of devices host PPSP-TP servers (Trackers) and clients (Peers). Therefore, support for multiple standard schema languages, management protocols and information models, suited to different purposes, was considered in the PPSP-TP design. Specifically, management functionalities for PPSP-TP devices can be achieved with Simple Network Management Protocol (SNMP) [RFC3410], syslog [RFC5424] and NETCONF [RFC6241].

5.2.2 Management Information

PPSP Trackers may implement SNMP management interfaces, namely the Application Management MIB [RFC2564] without the need to instrument the Tracker application itself. The channel, connections and transaction objects of the the Application Management MIB can be used to report the basic behavior of the PPSP Tracker service.

The Application Performance Measurement MIB (APM-MIB) [RFC3729] and the Transport Performance Metrics MIB (TPM-MIB) [RFC4150] can be used with PPSP-TP, providing adequate metrics for the analysis of performance for transaction flows in the network, in direct relationship to the transport of PPSP-TP.

The Host Resources MIB [RFC2790] can be used to supply information on the hardware, the operating system, and the installed and running software on a PPSP Tracker host.

The TCP-MIB [RFC4022] can additionally be considered for network monitoring.

Logging is an important functionality for PPSP-TP server (Tracker) and client (Peer), done via syslog [RFC5424].

5.2.3 Fault Management

As PPSP Tracker failures can be mainly attributed to host or network conditions, the facilities previously described for verifying the correct operation of PPSP-TP and the management of PPSP Tracker servers, appear sufficient for PPSP-TP fault monitoring.

5.2.4 Configuration Management

PPSP Tracker deployments, when realized by geographically distributed tracker nodes or multiple server nodes in a data center, may benefit from a standard way of replicating atomic configuration updates over a set of server nodes. This functionality can be provided via NETCONF [RFC6241].

5.2.5 Accounting Management

PPSP-TP implementations, namely for content provider environments, can benefit from accounting standardization efforts as defined in [RFC2975], in terms of resource consumption data, for the purposes of capacity and trend analysis, cost allocation, auditing, and billing.

5.2.6 Performance Management

Being transaction-oriented, PPSP-TP performance, in terms of availability and responsiveness, can be measured with the facilities of the APM-MIB [RFC3729] and the TPM-MIB [RFC4150].

5.2.7 Security Management

Standard SNMP notifications for PPSP Tracker management and syslog messages [RFC5424] can be used, to alert operators to the conditions identified in the security considerations (section 6).

The statistics collected about the operation of PPSP-TP can be used for detecting attacks, such as the receipt of malformed messages, messages out of order, or messages with invalid timestamps.

6 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 guarantee 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].

6.1 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 SHOULD be received from authorized peers. For that purpose a peer SHOULD 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 [RFC6749] with bearer token, which provides the peer with the information required to successfully utilize an access token to make protected requests to the tracker [RFC6750].

6.2 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.

6.3 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.

6.4 Pro-incentive parameter trustfulness

Property types for STAT_REPORT messages (such as those of property="StreamStatistics" in Table 5 of section 3.2) may consider additional pro-incentive parameters (guidelines for extension in section 7), 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. 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.

One such solution could be a reputation-incentive mechanism, based on the notions of reputation, social awareness and fairness. The mechanism would promote cooperation among participants (via each peer's reputation) based on the history of past transactions, such as, count of chunk requests (sent, received) in a swarm, contribution time of the peer, cumulative uploaded and downloaded content, JOIN and LEAVE timestamps, attainable rate, etc.

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].

7 Guidelines for Extending PPSP-TP

Extension mechanisms allow designers to add new features or to customize existing features of a protocol for different operating environments [RFC6709].

Extending a protocol implies either the addition of features without changing the protocol itself or the addition of new elements creating new versions of an existing schema and therefore new versions of the protocol.

In PPSP-TP it means that an extension **MUST NOT** alter an existing protocol schema as the changes would result in a new version of an existing schema, not an extension of an existing schema, typically non-backwards-compatible.

Additionally, a designer **MUST** remember that extensions themselves **MAY** also be extensible.

Extensions **MUST** adhere to the principles described in this section in order to be considered valid.

Extensions **MAY** be documented as Internet-Draft and RFC documents if there are requirements for coordination, interoperability, and broad distribution.

Extensions need not be published as Internet-Draft or RFC documents if they are intended for operation in a closed environment or are otherwise intended for a limited audience.

7.1 Forms of PPSP-TP Extension

In PPSP-TP two extension mechanisms can be used: a Request-Response Extension or a Protocol-level Extension.

- o Request-Response Extension: Adding elements or attributes to an existing element mapping in the schema is the simplest form of extension. This form should be explored before any other. This task can be accomplished by extending an existing element mapping.

For example, an element mapping for the StatisticsGroup can be extended to include additional elements needed to express status information about the activity of the peer, such as OnlineTime for the Stat element.

- o Protocol-level Extension: If there is no existing element mapping that can be extended to meet the requirements and the existing PPSP-TP Request and Response message structures are insufficient,

then extending the protocol should be considered in order to define new operational Requests and Responses.

For example, to enhance the level of control and the granularity of the operations, a new version of the protocol with new messages (JOIN, DISCONNECT), a retro-compatible change in semantics of an existing CONNECT Request/Response and an extension in STAT_REPORT could be considered.

As illustrated in Figure 6, the peer would use an enhanced CONNECT Request to perform the initial registration in the system. Then it would JOIN a first swarm as Seeder, later JOIN a second swarm as Leecher, and then DISCONNECT from the latter swarm but keeping as Seeder for the first one. When deciding to leave the system, the peer DISCONNECTs gracefully from it:

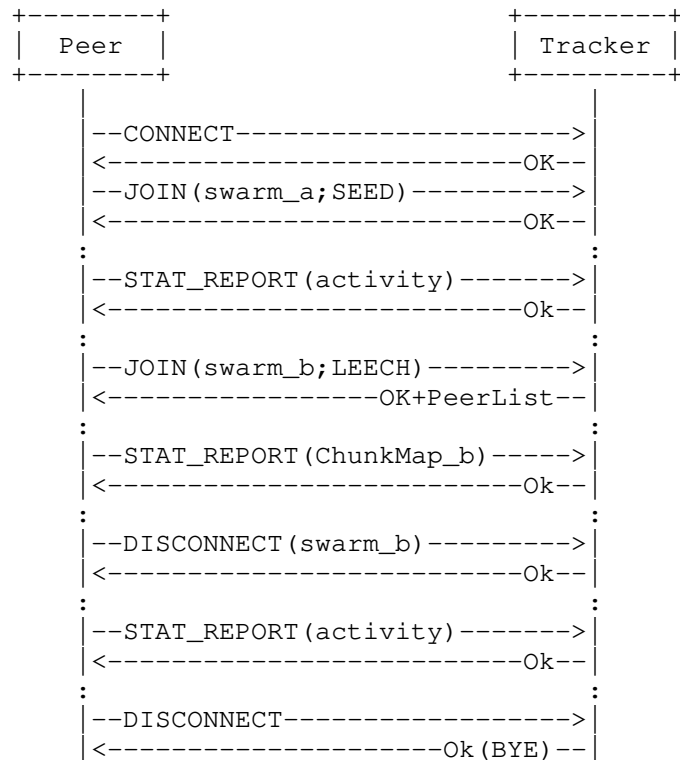


Figure 6: Example of a session for a PPSP-TP extended version.

7.2 Issues to Be Addressed in PPSP-TP Extensions

There are several issues that all extensions should take into

consideration.

- Overview of the Extension: It is RECOMMENDED that extensions to PPSP-TP have a protocol overview section that discusses the basic operation of the extension. The most important processing rules for the elements in the message flows SHOULD also be mentioned.
- Backward Compatibility: One of the most important issues to consider is whether the new extension is backward compatible with the base PPST-TP.
- Syntactic Issues: Extensions that define new Request/Response methods SHOULD use all capitals for the method name, keeping with a long-standing convention in many protocols, such as HTTP. Method names are case sensitive in PPSP-TP. Method names SHOULD be shorter than 16 characters and SHOULD attempt to convey the general meaning of the Request or Response.
- Semantic Issues: PPSP-TP extensions MUST clearly define the semantics of the extensions. Specifically, the extension MUST specify the behaviors expected from both the Peer and the Tracker in processing the extension, with the processing rules in temporal order of the common messaging scenario.

Processing rules generally specify actions to be taken on receipt of messages and expiration of timers.

The extension SHOULD specify procedures to be taken in exceptional conditions that are recoverable. Handling of unrecoverable errors does not require specification.

- Security Issues: Being security an important component of any protocol, designers of PPSP-TP extensions need to carefully consider security requirements, namely authorization requirements and requirements for end-to-end integrity.
- Examples of Usage: The specification of the extension SHOULD give examples of message flows and message formatting and include examples of messages containing new syntax. Examples of message flows should be given to cover common cases and at least one failure or unusual case.

8 IANA Considerations

There are presently no IANA considerations with this document.

9 Acknowledgments

The authors would like to thank many people 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 Stiernerling, Jianyin Zhang, Johan Pouwelse and Arno Bakker.

Rui Cruz, Mario Nunes and Joao Taveira are partially supported by the SARACEN project [SARACEN], a research project of the European Union 7th Framework Programme (contract no. ICT-248474).

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, the European Commission, Huawei or China Mobile.

This I-D document was prepared using NroffEdit version 2.08 (<http://aaa-sec.com/nroffedit/>).

10 References

10.1 Normative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [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.
- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.
- [XML] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E. and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", W3C xml, November 2008, <<http://www.w3.org/TR/xml/>>.
- [XMLSchema.1] Thompson, H., Beech, D., Maloney, M. and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", W3C xmlschema-1, October 2004, <<http://www.w3.org/TR/xmlschema-1/>>.

[XMLSchema.2] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", W3C xmlschema-2, October 2004, <<http://www.w3.org/TR/xmlschema-2/>>.

[XMLNamespace] Bray, T., Hollander, D., Layman, A., Tobin, R. and H. Thompson, "Namespaces in XML", W3C REC-xml-names, December 2009, <<http://www.w3.org/TR/REC-xml-names/>>.

[EXI] Schneider, J. and T. Kamiya, "Efficient XML Interchange (EXI) Format 1.0", W3C exi, March 2011, <<http://www.w3.org/TR/exi/>>.

10.2 Informative References

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

[RFC2564] Kalbfleisch, C., Krupczak, C., Presuhn, R., and J. Saperia, "Application Management MIB", RFC 2564, May 1999.

[RFC2790] Waldbusser, S. and P. Grillo, "Host Resources MIB", RFC 2790, March 2000.

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

[RFC2975] Aboba, B., Arkko, J., and D. Harrington, "Introduction to Accounting Management", RFC 2975, October 2000.

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

[RFC3470] Hollenbeck, S., Rose, M., and L. Masinter, "Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols", BCP 70, RFC 3470, January 2003.

[RFC3729] Waldbusser, S., "Application Performance Measurement MIB", RFC 3729, March 2004.

[RFC4022] Raghunathan, R., Ed., "Management Information Base for the Transmission Control Protocol (TCP)", RFC 4022, March 2005.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005.

- [RFC4150] Dietz, R. and R. Cole, "Transport Performance Metrics MIB", RFC 4150, August 2005.
- [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.
- [RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, March 2009.
- [RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC 5706, November 2009.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6709] Carpenter, B., Aboba, B., Ed., and S. Cheshire, "Design Considerations for Protocol Extensions", RFC 6709, September 2012.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.
- [RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, October 2012.
- [RFC6972] Zhang, Y. and N. Zong, "Problem Statement and Requirements of the Peer-to-Peer Streaming Protocol (PPSP)", RFC 6972, July 2013.
- [I-D.ietf-httpbis-http2] Belshe, M., Twist, Peon, R., Thomson, M., and A. Melnikov, "Hypertext Transfer Protocol version 2.0", draft-ietf-httpbis-http2-06 (work in progress), August 2013.
- [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.
- [I.D.ietf-alto-protocol] Alimi, R., Penno, R. and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-20, (work in progress)

progress), October 2013.

[I-D.pantos-http-live-streaming] Pantos, R. and W. May, "HTTP Live Streaming", draft-pantos-http-live-streaming-12 (work in progress), October 2013.

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

[Contracts] Piatek, M., Venkataramani, A., Yang, R., Zhang, D. and A. Jaffe, "Contracts: Practical Contribution Incentives for P2P Live Streaming", in NSDI '10: USENIX Symposium on Networked Systems Design and Implementation, April 2010.

Appendix A. Revision History

- 00 2013-02-14 Initial version.
- 01 2013-02-14 Minor revision.
 - * Terminology definitions related with streaming media contents moved to Appendix C. - Use Scenarios and Assumptions.
 - * Terminology updated to follow reviewed PPSP Problem Statement [RFC6972] definitions.
 - Definition of JOIN TIME, PEER-PEER MESSAGES and TRACKER-PEER MESSAGES removed.
 - + Definitions of PEER LIST, CHUNK ID, PPSP-TP, SEGMENT CHUNK, TRANSACTION ID added.
 - + Definitions of SEGMENT updated.
 - + Descriptions of Client Media Player and service Portal, related to PPSP protocols, were added to section 2.1.
 - + In section 2.2, "not exhaustive" and "typical" was added to first paragraph.
 - + In section 2.3.2, the description of CONNECT was updated to clarify the information recorded and maintained by the tracker related with Peer addressing, location and activity mode in the swarms.
 - + In section 3.1 and Table 2, PeerNum element justified to be limited to 30 entries.
 - + Section 6.4 enhanced with complementary information on incentive mechanisms.
 - + Section 5 completed.
- 02 2013-10-21 Minor revision.
 - + In section 2.2, Figure 1 was complemented to include the FIND request.
 - In section 2.3.1, references to HTTP/2.0 and framing formats were removed.
 - + In section 2.3.2, Table 1 was updated with FIND request message and the corresponding description added in text.
 - + In section 2.4, Figure 5 and the text were updated to reflect the FIND request message.
 - + In sections 2.4.1 and 2.4.2, the PEER REGISTERED and TRACKING state descriptions were modified to include the FIND request message.
 - In sections 3.1, references to EXI were removed and text was modified to reflect the FIND request message.
 - + In sections 3.3, Table 1 was updated with FIND request message.
 - + In sections 3.4, the text was updated to include the response to FIND requests.
 - + Section 4.2 renamed to 4.3, and new section 4.2 describes in detail the processing of FIND requests. All references to previous section 4.2 updated to 4.3.
 - + Appendix D added.

Appendix B. PPSP-TP Message Syntax for HTTP/1.1

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.

B.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 [RFC1952] 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.

B.2 Methods

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

B.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.

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".

B.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.

Appendix C. Use Scenarios

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.

C.1 Additional Terminology

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.

COMPLEMENTARY REPRESENTATION: Representation in a set of content representations which have inter-representation dependencies and which, when combined, result in a single representation for decoding and presentation.

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 (i.e., fidelity) when combined with the playable Base Layer.

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.

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 nested in dependent enhancement layers (levels of quality) allow the video to be played in compatible 3D rendering devices when the views are combined together.

C.2 Functional Entities

The functional entities related to PPSP protocols are the Client Media Player, the service Portal, the tracker and the peers.

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.

The peer is also a logical entity in the client device 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 get 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.

C.3 Streaming Capabilities

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 having the following representations:

1. Adaptive - alternate representations with different qualities and bitrates; a single representation 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.

C.4 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 other requesting peers.

C.5 Content Information Metadata

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

The representations of a media content correspond to encoded alternatives 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 transport chunks in temporal sequence.

These characteristics may be described in a Media Presentation Description (MPD) file. It is envisioned that the content information metadata used in PPSP may align with MPD formats, such as ISO/IEC 23009-1 [ISO.IEC.23009-1] and [I-D.pantos-http-live-streaming].

C.6 Authentication, Confidentiality, Integrity

Channel-oriented security can 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 may be employed. One such method is the OAuth 2.0 Authorization [RFC6749] with bearer token, providing the peer with the information required to successfully utilize the access token to make protected requests to the tracker [RFC6750].

Appendix D. Implementation Options

With newer versions of HTTP, the framing formats used for encoding and transmitting the data over the wire, e.g., the encapsulation proposed for HTTP/2.0 [I-D.ietf-httpbis-http2] may be used in PPSP-TP.

Similarly, for compact on the wire representation, PPSP-TP Requests and Responses can be represented in a binary form using, for example, Efficient XML Interchange (EXI) Format [EXI].

When using the above binary format for both the transport protocol and the PPSP messages the returned Peer List from a CONNECT or a FIND Request should be restricted to a number of entries that makes the total message size smaller than 1 KiB, allowing it to fit inside a single IP packet. The RECOMMENDED approach is to limit the list to 30 entries (less than 1KiB in size), as this size has a high likelihood of traveling across the Internet without fragmentation, or be transmitted as a single IP packet over an Ethernet network (1500 byte frame size).

Authors' Addresses

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

Gu Yingjie
Email: guyingjie@gmail.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

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
Email: denglingli@chinamobile.com