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

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

February 24, 2012

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

Abstract

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

Status of this Memo

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

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

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

material or to cite them other than as "work in progress."

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

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

Copyright and License Notice

Table of Contents

1. Introduction

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

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

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

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

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

1.1. Use Scenarios and Streaming Modes

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

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

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

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

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

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

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

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

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

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

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

1.2. Assumptions

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

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

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

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

1.2.1. Enrollment and Bootstrap

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

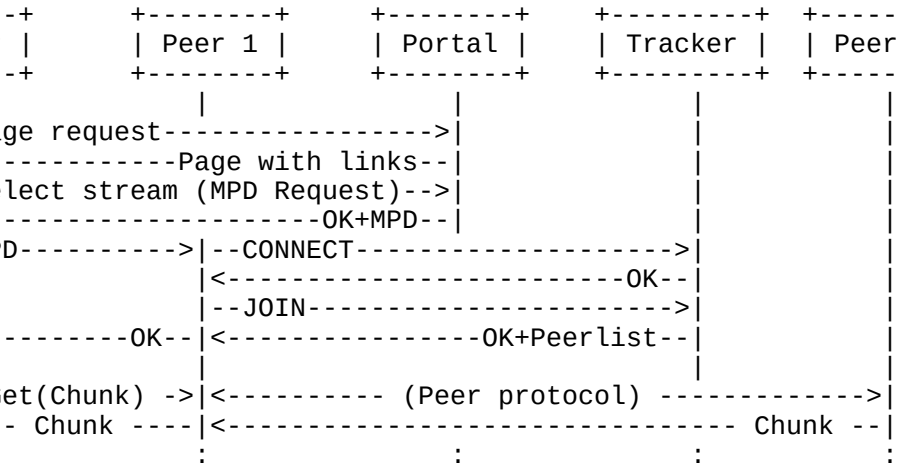

Figure 1: A typical PPSP session

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

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

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

1.2.2. NAT Traversal

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

1.2.3. Content Information Metadata

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

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

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

1.2.4. Authentication, Confidentiality, Integrity

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

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

2. Terminology

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

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

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

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

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

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

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

and presentation.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Representation: Structured collection of one or more media components.

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

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

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

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

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

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

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

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

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

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

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

3. Architectural and Functional View

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

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

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

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

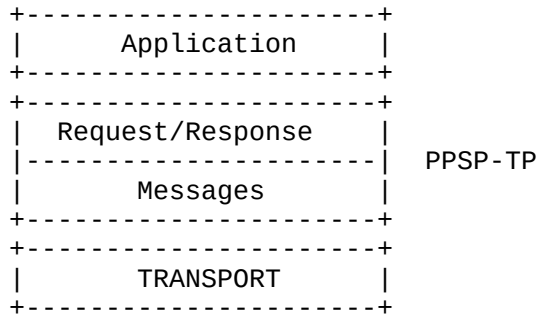

Figure 2: Abstract layering of PPSP-TP

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

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

peers participating in that specific swarm.

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

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

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

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

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

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

A P2P streaming process is summarized in Figure 3.

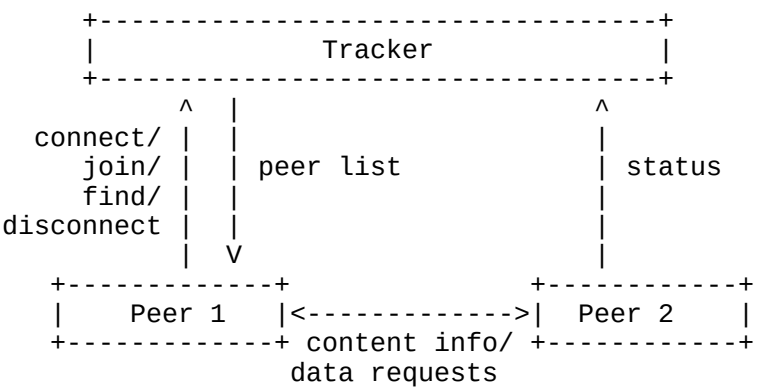

Figure 3: A PPSP streaming process

4. Messaging Model

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

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

5. Request/Response model

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

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

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

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

PPSP Tracker Req. Messages
CONNECT
DISCONNECT
JOIN
FIND
STAT_REPORT

Table 1: Request Messages

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

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

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

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

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

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

6. The Tracker State Machine

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

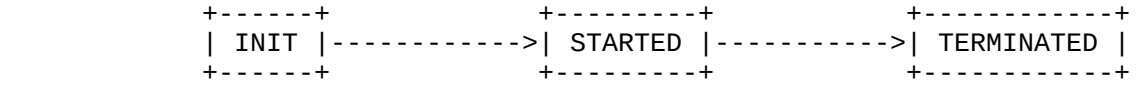

Figure 4: Tracker State Machine

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

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

that, the TERMINATED state here is transient.

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

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

Figure 5: Per-Peer-ID Transaction State Machine

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

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

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

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

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

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

6.1. Normal Operation

On normal operation the process consists of the following steps:

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

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

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

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

the specified swarm(s).

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

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

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

6.2. Error Conditions

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

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

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

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

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

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

7. Protocol Specification

7.1. Messages Syntax

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

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

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

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

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

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

A Request message example is the following:

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

[Request_Body]
```

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

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

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

A Response message example is the following:

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

[Response_Body]
```

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

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

7.1.1. Header Fields

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

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

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

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

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

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

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

7.1.2. Methods

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

7.1.3. Message Bodies

PPSP-TP requests MUST contain message-bodies.

PPSP-TP responses MAY include a message-body.

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

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

7.1.4. Message Response Codes

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

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

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

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

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

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

401 Unauthorized -- The request requires authentication.

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

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

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

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

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

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

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

7.2. Request/Response Syntax and Format

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

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

The generic format of a Request is the following:

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

The generic format of a Response is the following:

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

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

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

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

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

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

The PeerID information may be present on the following levels:

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

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

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

The SwarmID information may be present on the following levels:

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

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

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

The PeerMode information may be present on the following levels:

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

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

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

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

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

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

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

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

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

7.2.1. Semantics of PPSPTrackerProtocol elements

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

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

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

Table 2: Semantics of PPSPTrackerProtocol.

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

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

Table 3: Semantics of PeerGroup.

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

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

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

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

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

Table 4: Semantics of ContentGroup.

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

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

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

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

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

Table 5: Semantics of StatisticsGroup.

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

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

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

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

Table 6: StatisticsGroup default Stat @property values.

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

7.2.2. Request element in request Messages

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

XML Request Methods String Values
CONNECT
DISCONNECT
JOIN
FIND
STAT_REPORT

Table 7: Valid Strings for Request element of requests.

7.2.3. Response element in response Messages

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

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

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

Table 8: Valid Strings for Response element of responses.

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

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

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

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

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

8. Request/Response Processing

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

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

8.1. CONNECT Request

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

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

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

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

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

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

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

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

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

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

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

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

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

8.2. DISCONNECT Request

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

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

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

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

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

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

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

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

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

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

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

8.3. JOIN Request

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

8.4. FIND Request

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

8.5. STAT_REPORT Request

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

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

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

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

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

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

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

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

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

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

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

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

8.6. Error and Recovery conditions

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

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

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

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

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

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

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

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

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

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

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

9. Security Considerations

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

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

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

9.1. Authentication between Tracker and Peers

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

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

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

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

9.2. Content Integrity protection against polluting peers/trackers

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

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

9.3. Residual attacks and mitigation

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

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

9.4. Pro-incentive parameter trustfulness

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

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

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

suggested in [Contracts].

10. IANA Considerations

There are presently no IANA considerations with this document.

11. Acknowledgments

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

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

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

12. References

12.1. Normative References

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

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

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

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

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

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

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

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

12.2. Informative References

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

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

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

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

(work in progress), November 2011.

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

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

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

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

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

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

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

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

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

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

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

Networked Systems Design and Implementation, April 2010.

Appendix A. PPSP Tracker Protocol XML-Schema

TO BE ADDED.

Appendix B. Media Presentation Description (MPD)

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

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

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

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

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

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

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

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

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

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

program stream.

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

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

Appendix C. PPSP Requirements Compliance

C.1. PPSP Basic Requirements

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

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

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

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

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

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

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

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

C.2. PPSP Tracker Protocol Requirements

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

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

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

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

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

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

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

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

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

C.3. PPSP Security Considerations

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

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

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

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

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

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

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

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

Authors' Addresses

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

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

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

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

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

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

Deng Lingli
China Mobile