PPSP
INTERNET-DRAFT
Intended Status: Standards Track
Expires: August 18, 2013

Rui S. Cruz
Mario S. Nunes
IST/INESC-ID/INOV
Yingjie Gu
Jinwei Xia
Huawei
Joao P. Taveira
IST/INOV
Deng Lingli
China Mobile
February 14, 2013

PPSP Tracker Protocol-Base Protocol (PPSP-TP/1.0)
draft-ietf-ppsp-base-tracker-protocol-00

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

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 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.

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 basic unit of data block organized in P2P streaming for storage, scheduling, advertisement and exchange among peers. A chunk therefore refers to a segment of partitioned streaming media.

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.

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.

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.

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.

Leecher: A Peer that has not yet completed the transfer of all chunks of the media content.

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 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 stores and uploads streaming content to other participants.

Peer-ID: Unique identifier for the peer. 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 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-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.

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

Segment: A segment is a resource that can be identified, by an ID or an HTTP-URL and possibly a byte-range, and is described 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.

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. 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 centralized (or distributed) 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 can 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.

2 Operation and Protocol Architecture Overview

2.1 Operation

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

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 (via peer protocol).
4. Peer identifies the peers with desired content.
5. Peer requests content from the identified peers (peer protocol).

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 for example, the following methods (illustrated in Figures 1 and 2):

```
+--------+      +--------+     +--------+    +---------+  +--------+
| Player |      | Peer 1 |     | Portal |    | Tracker |  | Peer 2 |
+--------+      +--------+     +--------+    +---------+  +--------+
    |                |               |              |            |
(a) |--Page request----------------->|              |            |
    |<--------------Page with links--|              |            |
    |--Select stream (MPD Request)-->|              |            |
    |<--------------------OK+MPD(x)--|              |            |
(b) |--Start/Resume->|--CONNECT(join x)------------>|            |
    |<-----------OK--|<----------------OK+Peerlist--|            |
    |                |                              |            |
    |--Get(Chunk)--->|<---------- (Peer protocol) ------------->|
    |<--------Chunk--|<---------------------------------Chunks--|
    :                :               :              :            :
    |                |--STAT_REPORT---------------->|            |
    |                |<-------------------------Ok--|            |
    :                :               :              :            :
    |--Get(Chunk)--->|<---------- (Peer protocol) ------------->|
    |<--------Chunk--|<---------------------------------Chunks--|
    :                :               :              :            :
```

Figure 1: A typical PPSP session for watching a streaming content.

```
             +---------+                      +---------+
             |  Seeder |                      | Tracker |
             +---------+                      +---------+
                  |                                |
           Start->|--CONNECT (join x,y,z)--------->|
                  |<--------------------------OK--|
                  :                                :
                  |                                |
                  |--STAT_REPORT------------------>|
                  |<--------------------------Ok--|
                  :                                :
                  |                                |
                  |--STAT_REPORT------------------>|
                  |<--------------------------Ok--|
                  :                                :
```

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

1. From a content service provider provisioning mechanism: this is the typical case used for the provider Seeders (edge caches and/or Media Servers), where some "Start" activation signal triggers the process for contents x, y, z (Figure 2).
2. From a web page: a Publishing and Searching Portal may provide tracker location information to end users.
3. From the Media Presentation Description (MPD) file of a content: this meta-info 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, as illustrated in Figure 1 for a content x.

As illustrated in Figure 1, a peer may initiate a stream using the above method 3 starting at point (a), or resume a previously initiated stream, using also method 3 but starting at point (b).

In order to be able to bootstrap, a peer 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). The specification of the format of the Peer-ID 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).

```
                 +----------------------+
                 |      Application     |
                 +----------------------+
                 |  Request/Response    |  PPSP-TP
                 |----------------------|
                 |   (HTTP) Message     |
                 +----------------------+
                 |       TRANSPORT      |
                 +----------------------+
```

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

PPSP-TP 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
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 link status. 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)

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.

```
           --------------------------------------------
          /                                            \
         |  +------------+    +=========+    +======+   |
          \-| TERMINATED |<---| STARTED |<---| INIT |<-/
            +------------+    +=========+    +======+
             (Transient)                         \- (start tracker)
```

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

```
               --------------------------------------------
              /                                            \
             |  +------------+    +=========+    +======+   |
              \-| TERMINATED |<---| STARTED |<---| INIT |<-/
                +------------+    +=========+    +======+
                 (Transient)           | (1)        \- (start tracker)
                                       V
                   +-----------+   +-------+  rcv CONNECT
       (Transient) | TERMINATE |   | START |  --------------- (1)
                   +-----------+   +-------+  strt init timer
                         ^             |
   rcv STAT_REPORT       |             |
   on registration error |             v
   on action error       |      +------------+
   ---------------- (A)  +<-----| PEER       | (Transient)
   stop init timer       |      | REGISTERED |
   snd error             |      +------------+
                         |            |
                         |            |   process swarm actions
                         |            |   --------------------- (2)
   on CONNECT Error (B)  |            |   snd OK (PeerList)
   on timeout       (C)  |           /    stop init timer
   ----------------      |          /     strt track timer
   stop track timer      |         /
   clean peer info       |        |
   del registration      |        |
   snd error (B)          \       |     ----
                    ----   \      |   /      \
                  /      \  \     |  |        |
   rcv CONNECT   |  (4)   |  |    |  |        |
   -----------   |        v  |    v  v        | rcv STAT_REPORT
   snd OK         \     +==============+     /  --------------- (3)
   rst track timer  ----|   TRACKING   |----    snd OK response
                        +==============+        rst track timer
```

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

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.

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 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 processes the requested action(s) for the valid swarm information contained in the CONNECT request. 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), depending on peer mode (section 4.1).

3) In TRACKING state, a STAT_REPORT message received from that Peer-ID resets the "track timer" and is responded with a successful condition.

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 (or a STAT_REPORT is received from an unregistered Peer-ID), the tracker responds with either error codes 401 Unauthorized or 403 Forbidden (described in section 4.3), 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 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 Format

The message-body for Requests and Responses requiring it, correspond to a XML document [XML], optionally represented in a binary form using, for example, Efficient XML Interchange (EXI) Format [EXI].

The XML message-body MUST begin with an XML declaration line specifying the version of XML being used and indicating the character encoding, which SHOULD be "UTF-8". The root element MUST begin with an <PPSPTrackerProtocol> element. This element identifies the start of an PPSP-TP protocol element, the namespace used within the protocol, and the location of the protocol schema.

The generic format of a Request is the following:

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

The generic format of a Response is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol xmlns="TBD" schemaLocation="TBD" version="1.0">
  <Response></Response>
  <TransactionID></TransactionID>
  <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 element SwarmID MUST be present in requests to identify the

actions to be taken in the specified swarms.

The version of PPSP-TP being used is indicated by the attribute @version of the root element, specified in a XML Schema ([XMLSchema.1] and [XMLSchema.2]) with XML namespaces [XMLNameSpace] to identify protocol grammars.

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 subsection 3.2 Table 2.

- On PeerGroup level in PeerGroup.PeerInfo.PeerID element (attribute @swarID). For details refer to subsection 3.2 Table 3.

The SwarmID element MUST be be present in CONNECT Requests and SHOULD be present in STAT_REPORT Requests on the following levels:

- On PPSPTrackerProtocol level in PPSPTrackerProtocol.SwarmID element. For details refer to subsection 3.2 Table 2.

- On StatisticsGroup level in StatisticsGroup.Stat.SwarmID element. For details refer to subsection 3.2 Table 4.

The PeerNum element MAY be present in CONNECT 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.

The StatisticsGroup element MAY be present in STAT_REPORT requests. Details of usage in subsection 4.2.

The semantics of the attributes and elements within a PPSPTrackerProtocol root element is described in subsection subsection 3.2.

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

3.2 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
TransactionID	M	Root transaction Identification
Result	0...N	Result of @action MUST be present in Responses
@transactionID	CM	Identifier of the @action
PeerID	0...1	Peer Identifier MUST be present in Request
SwarmID	0...N	Swarm Identifier MUST be present in Requests
@action	CM	Must be set to JOIN or LEAVE
@peerMode	CM	Mode of Peer participation in the swarm, LEECH or SEED
@transactionID	CM	Identifier for the @action
PeerNUM	0...1	Maximum peers in PeeerList
@abilityNAT	OP	Type of NAT traversal peers, as No-NAT, STUN, TURN or PROXY
@concurrentLinks	OP	Concurrent connectivity level of peers, HIGH, LOW or NORMAL
@onlineTime	OP	Availability or online duration of peers, HIGH or NORMAL
@uploadBWlevel	OP	Upload bandwidth capability of peers, HIGH or NORMAL
PeerGroup	0...1	Information on peers (Table 3)
StatisticsGroup	0...1	Statistic data of peer (Table 4)

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 the base PPSPTrackerProtocol.

Element Name or Attribute Name	Use	Description
PeerGroup	0...1	Contains description of peers
PeerInfo	1...N	Provides information on a peer
@swarmID	0...1	Swarm Identifier
PeerID	0...1	Peer Identifier
PeerAddress	0...N	IP Address information
@addrType	M	Type of IP address, which can be ipv4 or ipv6
@priority	CM	The priority of this interface
@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
@peerProtocol	OP	PPSP Peer Protocol supported

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 [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.

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 and elements 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 4: 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 and peer status information. 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.

"Property Name" Element Name or Attribute Name	Use	Description
"StreamStatistics"		Property for swarm statistics
SwarmID	0...1	Swarm Identifier
UploadedBytes	0...1	Bytes sent to swarm
DownloadedBytes	0...1	Bytes received from swarm
AvailBandwidth	0...1	Upstream Bandwidth available
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: Basic StatisticsGroup property blocks.

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).

3.3 Request element in request Messages

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

XML Request Methods String Values
CONNECT
STAT_REPORT

Table 6: Valid Strings for Request element of requests.

3.4 Response element in response Messages

Table 7 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 Status-Code and Reason-Phrase (appended, if appropriate, with detail phrase, as described in section 4.3).

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

Table 7: 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.

In CONNECT Request: returns information about the successful registration of the peer and/or of each swarm @action requested. MAY additionally return the list of peers corresponding to the join @action requested.

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

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

4 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 section 4.3.

4.1 CONNECT Request

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

The peer MUST properly form the XML message-body, set the Request method to CONNECT, generate and set the TransactionIDs, set the PeerID with the identifier of the peer and include SwarmID action(s) to be performed. The peer MAY also include the IP addresses of its network interfaces in the CONNECT message.

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:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol xmlns="TBD" schemaLocation="TBD" version="1.0">
  <Request>CONNECT</Request>
  <PeerID>656164657220</PeerID>
  <PeerNum abilityNAT="STUN">15</PeerNum>
  <SwarmID action="JOIN" peerMode="SEED"
           transactionID="12345.1">1111</SwarmID>
  <SwarmID action="JOIN" peerMode="SEED"
           transactionID="12345.2">2222</SwarmID>
  <TransactionID>12345.0</TransactionID>
</PPSPTrackerProtocol>
```

Another example of the message-body of a CONNECT Request corresponds to a peer Leecher requesting join to a swarm, in order to start receiving the stream, and providing optional information on the addresses of its network interface(s):

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol xmlns="TBD" schemaLocation="TBD" version="1.0">
  <Request>CONNECT</Request>
  <PeerID>656164657221</PeerID>
  <PeerNum abilityNAT="STUN">5</PeerNum>
  <SwarmID action="JOIN" peerMode="LEECH"
           transactionID="12345.1">1111</SwarmID>
  <TransactionID>12345.0</TransactionID>
  <PeerGroup>
    <PeerInfo>
      <PeerAddress addrType="ipv4" ip="192.0.2.2" port="80"
                   priority="1" peerProtocol="PPSP-PP" />
      <PeerAddress addrType="ipv6" ip="2001:db8::2" port="80"
                   priority="2" peerProtocol="PPSP-PP" />
    </PeerInfo>
  </PeerGroup>
</PPSPTrackerProtocol>
```

The next example of the message-body of a CONNECT Request corresponds to a peer Leecher "leaving" a previously joined swarm and requesting join to a new swarm:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol xmlns="TBD" schemaLocation="TBD" version="1.0">
  <Request>CONNECT</Request>
  <PeerID>656164657221</PeerID>
  <PeerNum abilityNAT="STUN">5</PeerNum>
  <SwarmID action="LEAVE" peerMode="LEECH"
           transactionID="12345.1">1111</SwarmID>
  <SwarmID action="JOIN" peerMode="LEECH"
           transactionID="12345.2">2222</SwarmID>
  <TransactionID>12345.0</TransactionID>
</PPSPTrackerProtocol>
```

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 element SwarmID MUST be present with cardinality 1 to N, each containing the request for @action, the @peerMode of the peer and the child @transactionID for that swarm. The @peerMode element MUST be set to the type of participation of the peer in the swarm (SEED or LEECH).

The valid sets of SwarmID @action combinations for the CONNECT Request logic in PPSP-TP base protocol are summarized 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 combinations in CONNECT Request.

The element PeerInfo, if present, MAY contain multiple PeerAddress child elements with attributes @addrType, @ip, @port and @peerProtocol, 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 PeerNum 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.

The next example illustrates a Response for the CONNECT Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol xmlns="TBD" schemaLocation="TBD" version="1.0">
  <Response>SUCCESSFUL</Response>
  <TransactionID>
    <Result transactionID="12345.0">200 OK</Result>
    <Result transactionID="12345.1">200 OK</Result>
    <Result transactionID="12345.2">200 OK</Result>
  </TransactionID>
  <PeerGroup>
    <PeerInfo>
      <PeerID>656164657221</PeerID>
      <PeerAddress addrType="ipv4" ip="198.51.100.1" port="80"
                   priority="1"
                   type="REFLEXIVE"
                   connection="3G"
                   asn="64496" />
    </PeerInfo>
    <PeerInfo swarmID="2222">
       <PeerID>956264622298</PeerID>
       <PeerAddress addrType="ipv4" ip="198.51.100.22" port="80"
                    asn="64496" peerProtocol="PPSP-PP" />
    </PeerInfo>
    <PeerInfo swarmID="2222">
       <PeerID>3332001256741</PeerID>
       <PeerAddress addrType="ipv4" ip="198.51.100.201" port="80"
                    asn="64496" peerProtocol="PPSP-PP" />
    </PeerInfo>
  </PeerGroup>
</PPSPTrackerProtocol>
```

The Response MUST include a PeerGroup with PeerInfo data of the requesting 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 PeerGroup MAY also include PeerInfo data corresponding to the Peer-IDs and public IP addresses of the selected active peers in the requested swarm. 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 of the referenced peer.

In case the @peerMode is SEED, the tracker responds with a SUCCESSFUL response and enters the peer information into the corresponding swarm activity. In case the @peerMode is LEECH (or if the peer Seeder includes a PeerNum element in the request) the tracker will search and select an appropriate list of peers satisfying the conditions set by the requesting peer. The peer list returned 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].

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

4.2 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 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 and peer status information. 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.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol xmlns="TBD" schemaLocation="TBD" 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>
  </StatisticsGroup>
</PPSPTrackerProtocol>
```

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.

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

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

4.3 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

TBD.

5.1.2 Migration Path

TBD.

5.1.3 Requirements on Other Protocols and Functional Components

TBD.

5.1.4 Impact on Network Operation

TBD.

5.1.5 Verifying Correct Operation

TBD.

5.2 Management Considerations

TBD.

5.2.1 Interoperability

TBD.

5.2.2 Management Information

TBD.

5.2.3 Fault Management

TBD.

5.2.4 Configuration Management

TBD.

5.2.5 Accounting Management

TBD.

5.2.6 Performance Management

TBD.

5.2.7 Security Management

TBD.

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 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].

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 (section 3.2) 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.

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].

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].

Since the beginning that extensibility has been a design goal of PPSP-TP, as the protocol is represented in the Extensible Markup Language [XML].

A powerful feature of XML is the extensibility, but this feature also provides multiple opportunities to make poor design decisions.

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

PPSP-TP extensions MUST be specified in XML. This ensures that parsers capable of processing PPSP-TP structures will also be capable of processing PPSP-TP extensions. Guidelines for the use of XML in IETF protocols can be found in RFC 3470 [RFC3470].

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 (Section 3.2, Table 5) 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:

```
<Stat property="StreamStatistics">
  <OnlineTime>3600</OnlineTime>
</Stat>
```

Another example also for the StatisticsGroup is for additional @property attributes and elements, such as those necessary to report content chunk availability information:

```
<Stat property="ContentMap">
  <SegmentInfo chunkmapSize="25">
      A/8D/wP/A/8D/wP/A/8D/wP/A/8D/wP/....
  </SegmentInfo>
</Stat>
```

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:

```
                 +--------+                     +---------+
                 |  Peer  |                     | Tracker |
                 +--------+                     +---------+
                     |                               |
                     |--CONNECT--------------------->|
                     |<--------------------------OK--|
                     |--JOIN(swarm_a;SEED)---------->|
                     |<--------------------------OK--|
                     :                               :
                     |--STAT_REPORT(activity)------->|
                     |<--------------------------Ok--|
                     :                               :
                     |--JOIN(swarm_b;LEECH)--------->|
                     |<-----------------OK+PeerList--|
                     :                               :
                     |--STAT_REPORT(ChunkMap_b)----->|
                     |<--------------------------Ok--|
                     :                               :
                     |--DISCONNECT(swarm_b)--------->|
                     |<--------------------------Ok--|
                     :                               :
                     |--STAT_REPORT(activity)------->|
                     |<--------------------------Ok--|
                     :                               :
                     |--DISCONNECT------------------>|
                     |<---------------------Ok(BYE)--|
```

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 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.

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.

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

[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.

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

[RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC 5706, November 2009.

[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.

[I-D.ietf-httpbis-http2] Belshe, M., Twist, Peon, R., Thomson, M., and A. Melnikov, "Hypertext Transfer Protocol version 2.0", draft-ietf-httpbis-http2-01 (work in progress), January 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-13, (work in progress), September 2012.

[I-D.pantos-http-live-streaming] Pantos, R. and W. May, "HTTP Live Streaming", draft-pantos-http-live-streaming-10 (work in progress), October 2012.

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

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 and Assumptions

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.

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.1 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.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 other requesting peers.

C.3 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.4 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].

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

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