                Peer-to-Peer Streaming Tracker Protocol (PPSTP)

Abstract

   This document specifies the base Peer-to-Peer Streaming Tracker
   Protocol (PPSTP) version 1, 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; it also describes message flows, message
   processing instructions, message formats, formal syntax, and
   semantics. The PPSTP enables cooperating peers to form content-
   streaming overlay networks to support near real-time delivery of
   structured media content (audio, video, and 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 Notice

Table of Contents

1. Introduction

   The Peer-to-Peer Streaming Protocol (PPSP) is composed of two
   protocols: the Tracker Protocol (defined in this document) and the
   Peer Protocol (defined in [RFC7574]). [RFC6972] specifies that the
   Tracker Protocol should standardize the messages between PPSP peers
   and PPSP trackers and also defines the requirements.

   The Peer-to-Peer Streaming Tracker Protocol (PPSTP) 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 to be 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 acquisition of peer lists.

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

   This document introduces a base Peer-to-Peer Streaming Tracker
   Protocol (PPSTP) that satisfies the requirements in [RFC6972].

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 [RFC2119].

   absolute time: Expressed as ISO 8601 timestamps, using zero UTC
      offset. Fractions of a second may be indicated, for example,
      December 25, 2010 at 14h56 and 20.25 seconds in basic format is
      20101225T145620.25Z and in extended format is
      2010-12-25T14:56:20.25Z.

   chunk: An uniformly atomic subset of the resource that constitutes
      the basic unit of data organized in P2P streaming for storage,
      scheduling, advertisement, and exchange among peers.

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

   LEECH: The peers in a swarm that download content from other peers as
      well as contribute downloaded content with others. A LEECH should
      join the swarm with uncompleted media content.

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

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

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

peer list: A list of peers that are in the same swarm maintained by
    the tracker. A peer can fetch the peer list of a swarm from the
    tracker.

PPSP: The abbreviation of Peer-to-Peer Streaming Protocol.

PPSTP: The abbreviation of Peer-to-Peer Streaming Tracker Protocol.

SEEDER: The peers in a swarm that only contribute the content they
    have to others. A SEEDER should join the swarm with complete
    media content.

service portal: A logical entity typically used for client enrollment
    and for publishing, searching, and retrieving content information.
    It is usually located in a server of a content provider.

swarm: A group of peers that exchange data to distribute chunks of
    the same content (e.g., video/audio program, digital file, etc.)
    at a given time.

swarm ID: The identifier of a swarm containing a group of peers
    sharing common streaming content. The swarm ID may use a
    universally unique identifier (UUID), e.g., a 64- or 128-bit datum
    to refer to the content resource being shared among peers.

tracker: A directory service that maintains a list of peers
    participating in a specific audio/video channel or in the
    distribution of a streaming file. It is a logical component that
    can be deployed in a centralized or distributed way.

   transaction ID: The identifier of a request from the peer to the
      tracker. It is used to disambiguate responses that may arrive in
      a different order than the corresponding requests.

1.2. Design Overview

   The functional entities related to peer-to-peer streaming protocols
   are the Client Media Player, the service portal, the tracker, and the
   peers. The complete description of Client Media Player and service
   portal is not discussed here, as they are not in the scope of the
   specification. The functional entities directly involved in PPSTP
   are trackers and peers (which may support different capabilities).

   The Client Media Player is a logical entity providing direct
   interface to the end user at the client device and includes the
   functions to select, request, decode, and render content. The Client
   Media Player may interface with the local peer application using the
   standard format for HTTP request and response messages [RFC7230].

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

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

   A tracker is a logical entity that maintains the lists of peers
   storing chunks 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; in this document, it will be treated as a single logical
   entity. Communication between these physical nodes to present them
   as a single tracker to peers is not considered in PPSTP, which is a
   protocol between a tracker and a peer.

   PPSTP 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. PPSTP is not designed for applications for
   which in-sync reception is needed.

1.2.1. Typical PPSP Session

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

   1. Peer connects to a tracker and joins a swarm.

   2. Peer acquires a list of other peers in the swarm from the 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 content from the identified peers.

   When a peer wants to share streaming content (SEEDER mode) with other
   peers:

   1. Peer connects to a tracker.

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

   3. Peer waits for other peers in LEECH mode to connect with it (see
      steps 3-5 in the previous list).

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

1.2.2. Example of a PPSP Session

   In order to be able to bootstrap in the P2P network, a peer must
   first obtain a peer ID and any required security certificates or
   authorization tokens from an enrollment service (end-user
   registration). The peer ID MUST be unique (see the definition of
   "peer ID" in Section 1.1); however, the representation of the peer ID
   is not considered in this document.

```
    +--------+      +--------+      +--------+     +---------+  +--------+
    | 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--|              |
        :               :               :              :           :
        |               |--FIND---------------------->|             |
        |               |<---------------OK+Peerlist--|             |
        :               :               :              :           :
        |--Get(chunk)--->|<---------- (Peer protocol) -------------->|
        |<-------chunk--|<-----------------------------------chunks--|
        :               :               :              :           :
```
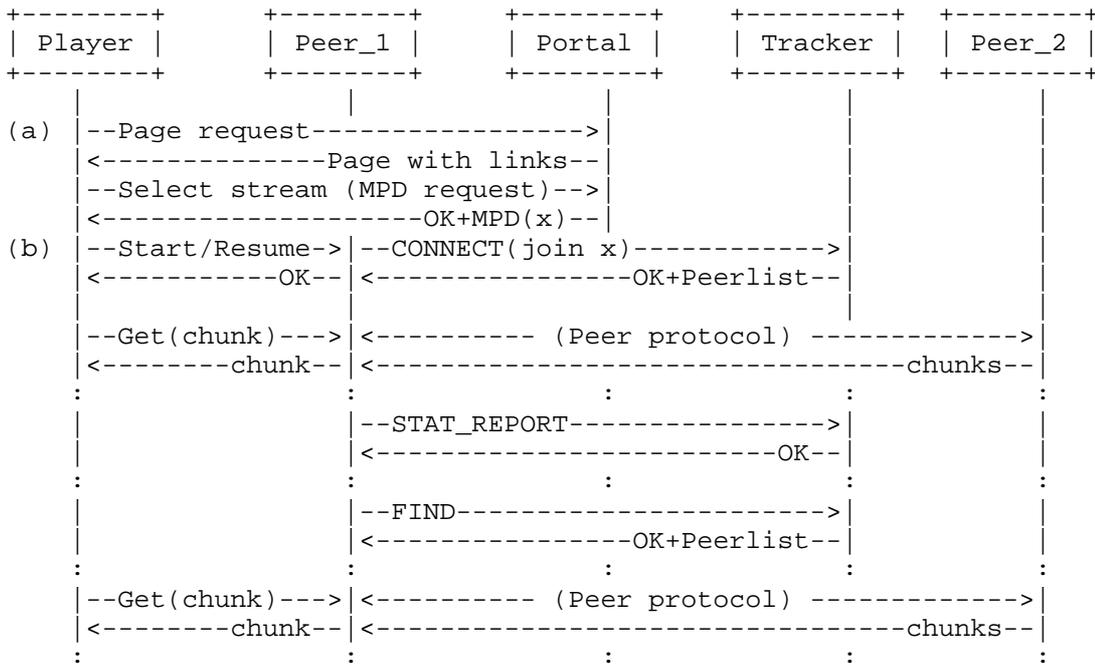
             Figure 1: A Typical PPSP Session for Streaming Content

   To join an existing P2P streaming service and to participate in
   content sharing, a peer must first locate a tracker.

   As illustrated in Figure 1, a P2P streaming session may be initiated
   starting at point (a), with the Client Media Player browsing for the
   desired content in order to request it (to the local Peer_1 in the
   figure), or resume a previously initiated stream, but starting at
   point (b). For this example, the Peer_1 is in mode LEECH.

   At point (a) in Figure 1, the Client Media Player accesses the portal
   and selects the content of interest. The portal returns the Media
   Presentation Description (MPD) file that includes information about
   the address of one or more trackers (which can be grouped by tiers of
   priority) that control the swarm x for that media content (e.g.,
   content x).

   With the information from the MPD, the Client Media Player is able to
   trigger the start of the streaming session, requesting to the local
   Peer_1 the chunks of interest.

The PPSP streaming session is then started (or resumed) at Peer_1 by
sending a PPSTP CONNECT message to the tracker in order to join swarm
x. The tracker will then return the OK response message containing a
peer list, if the CONNECT message is successfully accepted. From
that point, every chunk request is addressed by Peer_1 to its
neighbors (Peer_2 in Figure 1) using a peer protocol, e.g.,
[RFC7574], returning the received chunks to the Client Media Player.

Once connected, Peer_1 needs to periodically report its status and
statistics data to the tracker using a STAT_REPORT message.

If Peer_1 needs to refresh its neighborhood (for example, due to
churn), it will send a PPSTP FIND message (with the desired scope) to
the tracker.

Peers that are only SEEDERs (i.e., serving content to other peers),
as are the typical cases of service provider P2P edge caches and/or
media servers, trigger their P2P streaming sessions for content x, y,
z... (Figure 2), not from Media Player signals, but from some
"Start" activation signal received from the service provider
provisioning mechanism. In this particular case, the peer starts or
resumes all its streaming sessions just by sending a PPSTP CONNECT
message to the tracker (Figure 2), in order to "join" all the
requested swarms.

Periodically, the peer also reports its status and statistics data to
the tracker using a PPSTP STAT_REPORT message.

```
           +---------+                   +---------+
           | SEEDER  |                   | Tracker |
           +---------+                   +---------+
                |                             |
       Start->|--CONNECT (join x,y,z)-------->|
                |<------------------------OK--|
                :                             :
                |                             |
                |--STAT_REPORT---------------->|
                |<------------------------Ok--|
                :                             :
                |                             |
                |--STAT_REPORT---------------->|
                |<------------------------Ok--|
                :                             :
```
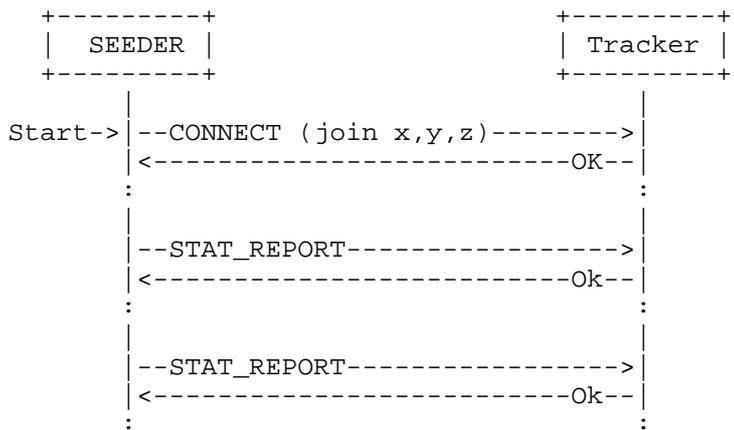
   Figure 2: A Typical PPSP Session for a Streaming SEEDER

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

2. Protocol Architecture and Functional View

   PPSTP is designed with a layered approach i.e., a PPSTP
   Request/Response layer, a Message layer, and a Transport layer (see
   Figure 3).

```
                  +-----------------------+
                  |      Application       |
                  +-----------------------+
                  |(PPSTP) Request/Response|
                  |-----------------------|
                  |    (HTTP) Message      |
                  +-----------------------+
                  |      Transport         |
                  +-----------------------+
```
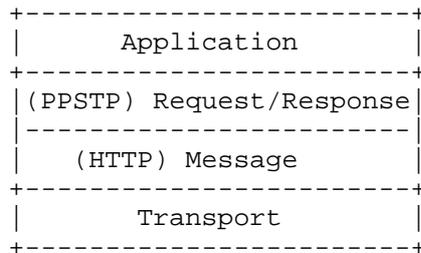
              Figure 3: Abstract Layering of PPSTP

   The PPSTP Request/Response layer deals with the interactions between
   tracker and peers using request and response messages.

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

   The Transport layer is responsible for the actual transmission of
   requests and responses over network transports, including the
   determination of the connection to use for a request or response
   message when using TCP or Transport Layer Security (TLS) [RFC5246]
   over it.

2.1. Messaging Model

   The messaging model of PPSTP aligns with HTTP, which is currently in
   version 1.1 [RFC7230], and the semantics of its messages. PPSTP is
   intended to also support future versions of HTTP.

2.2. Request/Response Model

   PPSTP uses a design like REST (Representational State Transfer) with
   the goal of leveraging current HTTP implementations and
   infrastructure, as well as familiarity with existing REST-like

services in popular use. PPSTP messages use the UTF-8 character set
[RFC3629] and are either requests from peers to a tracker service or
responses from a tracker service to peers. The request and response
semantics are carried as entities (header and body) in messages that
correspond to either HTTP request methods or HTTP response codes,
respectively.

PPSTP uses the HTTP POST method to send parameters in requests.
PPSTP messages use JavaScript Object Notation (JSON) [RFC7159] to
encode message bodies.

Peers send requests to trackers. Trackers send a single response for
each request though both requests and responses can be subject to
fragmentation of messages in transport.

The request messages of the base protocol are listed in Table 1:

```
          +------------------------------+
          | PPSTP Request Messages       |
          +------------------------------+
          | CONNECT                      |
          | FIND                         |
          | STAT_REPORT                  |
          +------------------------------+
```

                  Table 1: Request Messages

CONNECT:
   This request message is used when a peer registers in the tracker
   to notify it about participation in the named swarm(s). If the
   peer is already registered in the tracker, this request message
   simply notifies the tracker about participation in the named
   swarm(s). The tracker records the peer ID, connect-time
   (referenced to the absolute time), peer IP addresses (and
   associated location information), link status, and peer mode for
   the named swarm(s). The tracker also changes the content
   availability of the valid named swarm(s), i.e., changes the peer's
   lists of the corresponding swarm(s) for the requesting peer ID.
   On receiving a CONNECT message, the tracker first checks the peer
   mode type (SEEDER/LEECH) for the specified swarm(s) and then
   decides the next steps (see Section 4.1 for more details).

FIND:
   This request message is used by peers to request a list of peers
   active in the named swarm from the tracker whenever needed. On
   receiving a FIND message, the tracker finds the peers listed in
   the content status of the specified swarm that can satisfy the
   requesting peer's requirements and returns the list to the

      requesting peer. To create the peer list, the tracker may take
      peer status, capabilities, and peer priority into consideration.
      Peer priority may be determined by network topology preference,
      operator policy preference, etc.

   STAT_REPORT:
      This request message is used to allow 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.

2.3. 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)
```
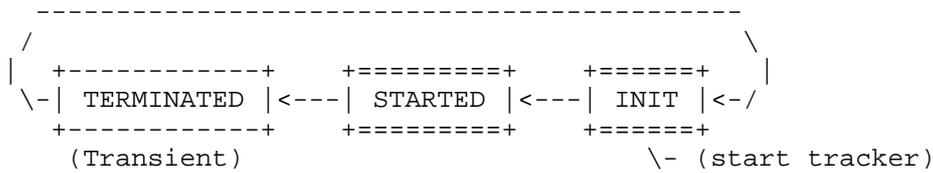
                 Figure 4: Tracker State Machine

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

   When the first peer connects to register 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
   STARTED state. When the last peer ID is removed, the state machine
   transitions to TERMINATED. From there, it immediately transitions
   back to INIT state. Because of this, TERMINATED state is transient.

   Once in STARTED state, each peer is instantiated (per peer ID) in the
   tracker state machine with a dedicated transaction state machine
   (Figure 5), which is deleted when the peer ID is removed.

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

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

   Unlike the tracker state machine, which exists even when no peer IDs
   are registered, the "per-Peer-ID" 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" 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 instantiated "per-Peer-ID" State Machine progresses from one state to another in response to various events. The events that may potentially advance the state include:

o Reception of CONNECT, FIND, and STAT_REPORT messages

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.

2.3.1. Normal Operation

For normal operation, the process consists of the following steps:

1) When a peer wants to access the system, it needs to register with a tracker by sending a CONNECT message asking for the swarm(s) it wants to join. This request from a new peer ID triggers the instantiation in the tracker of a "per-Peer-ID" State Machine. In the START state of the new "per-Peer-ID" State Machine, the tracker registers the peer ID and associated information (IP addresses), starts the "init timer", and moves to PEER REGISTERED state.

2) In PEER REGISTERED state, if the peer ID is valid, the tracker either:

   a) processes the requested action(s) for the valid swarm information contained in the CONNECT requests, and if successful, the tracker stops the "init timer", starts the "track timer", and sends the response to the peer (the response may contain the appropriate list of peers for the joining swarm(s), as detailed in Section 4.1), or

   b) moves the valid FIND request to TRACKING state.

3) In TRACKING state, STAT_REPORT or FIND messages received from that peer ID will reset the "track timer", and the tracker responds to the requests with the following, respectively:

   a) a successful condition, or

   b) a successful condition containing the appropriate list of peers
      for the named swarm (Section 4.2).

4) While in TRACKING state, a CONNECT message received from that peer
   ID with valid swarm action information (Section 4.1.1) resets the
   "track timer", and the tracker responds to the request with a
   successful condition.

2.3.2. Error Conditions

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

A) In PEER REGISTERED state, when a CONNECT request only contains
   invalid swarm actions (Section 4.1.1), the tracker responds with a
   PPSTP error code as specified in Section 4.3, deletes the
   registration, and transitions to TERMINATE state for that peer ID.
   The state machine is destroyed.

B) In PEER REGISTERED state, if the peer ID is considered invalid (in
   the case of a CONNECT request or in the case of FIND or
   STAT_REPORT requests received from an unregistered peer ID), the
   tracker responds with either a 06 (Authentication Required)
   error_code or a 03 (Forbidden Action) error_code as described in
   Section 4.3 and transitions to TERMINATE state for that peer ID.
   The state machine is destroyed.

C) In TRACKING state (while the "track timer" has not expired),
   receiving a CONNECT message from a peer ID with invalid swarm
   actions (Section 4.1.1) or receiving a FIND/STAT_REPORT message
   from a peer ID with an invalid swarm ID is considered an error
   condition. The tracker responds with the corresponding error code
   (described in Section 4.3).

D) In TRACKING state, without receiving messages from the peer on
   timeout (the "track timer" has expired), 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 state machine is destroyed.

   NOTE: These situations may correspond to malfunctions at the peer or
   to malicious conditions. As a preventive measure, the tracker
   proceeds to TERMINATE state for that peer ID.

3. Protocol Specification

3.1. Presentation Language

   PPSTP uses a REST-like design, encoding the requests and responses
   using JSON [RFC7159]. For a generalization of the definition of
   protocol elements and fields, as well as their types and structures,
   this document uses a C-style notation, similar to the presentation
   language used to define TLS [RFC5246].

   A JSON object consists of name/value pairs with the grammar specified
   in [RFC7159]. In this document, comments begin with "//", and the
   "ppsp_tp_string_t" and "ppsp_tp_integer_t" types are used to indicate
   the JSON string and number, respectively. Optional fields are
   enclosed in "[ ]" brackets. An array is indicated by two numbers in
   angle brackets, <min..max>, where "min" indicates the minimal number
   of values and "max" the maximum. An "*" is used to denote a no
   upper-bound value for "max".

3.2. Resource Element Types

   This section details the format of PPSTP resource element types.

3.2.1. Version

   For both requests and responses, the version of PPSTP being used MUST
   be indicated by the attribute version, defined as follows:

      ppsp_tp_integer_t ppsp_tp_version_t = 1

   The defined value for ppsp_tp_version_t is listed in Table 2.

      +--+
      | ppsp_tp_version_t | Description |
      +--+
      | 0 | Reserved |
      | 1 | PPSTP version 1 |
      | 2-255 | Unassigned |
      +--+

                    Table 2: PPSTP Version Numbers

3.2.2. Peer Number Element

   The peer number element is a scope selector optionally present in
   CONNECT and FIND requests.

   This element contains the attribute peer_count to indicate the
   maximum number of peers in the returned peer list. peer_count should
   be less than 30 in this specification. The other 4 attributes, i.e.,
   ability_nat, concurrent_links, online_time, and upload_bandwidth may
   also be contained in this element to inform the tracker the status of
   the peer so that the tracker could return some eligible peers based
   on the implementing rules set by the service providers:

   o ability_nat is used to indicate the preferred NAT traversal
      situation of the requesting peer.

   o concurrent_links means the number of P2P links the peer currently
      has.

   o online_time represents online duration time of the peer. The unit
      is second.

   o upload_bandwidth is the maximum upload bandwidth capability of the
      peer. The unit is Kbps.

   The scope selector element and its attributes are defined as follows:

```
      Object {
              ppsp_tp_integer_t    peer_count;
              [ppsp_tp_string_t    ability_nat = "NO_NAT"
                                               | "STUN"
                                               | "TURN";]
              [ppsp_tp_integer_t    concurrent_links;]
              [ppsp_tp_integer_t    online_time;]
              [ppsp_tp_integer_t    upload_bandwidth;]
      } ppsp_tp_peer_num_t;
```

3.2.3. Swarm Action Element

   The swarm action element identifies the action(s) to be taken in the
   named swarm(s) as well as the corresponding peer mode (if the peer is
   LEECH or SEEDER in that swarm).

```
      Object {
              ppsp_tp_string_t   swarm_id;   //swarm ID
              ppsp_tp_string_t   action = "JOIN"
                                      |"LEAVE"; // Action type of
                                                // the CONNECT
                                                // message

              ppsp_tp_string_t   peer_mode = "SEEDER"
                                      | "LEECH"; // Mode of the peer
                                                 // participating
                                                 // in this swarm
      } ppsp_tp_swarm_action_t;
```

3.2.4. Peer Information Elements

   The peer information elements provide network identification
   information of peers. A peer information element consists of a peer
   identifier and the IP-related addressing information.

```
      Object {
              ppsp_tp_string_t    peer_id;
              ppsp_tp_peer_addr_t peer_addr;
      } ppsp_tp_peer_info_t;
```

   The ppsp_tp_peer_addr_t element includes the IP address and port,
   with a few optional attributes related to connection type and network
   location (in terms of ASN) as well as, optionally, the identifier of
   the peer protocol being used.

```
      Object {
              ppsp_tp_ip_address        ip_address;
              ppsp_tp_integer_t         port;
              ppsp_tp_integer_t         priority;
              ppsp_tp_string_t          type = "HOST"
                                             | "REFLEXIVE"
                                             | "PROXY";
              [ppsp_tp_string_t         connection = "wireless"
                                              | "wired";]
              [ppsp_tp_string_t         asn;]
              [ppsp_tp_string_t         peer_protocol;]
      } ppsp_tp_peer_addr_t;
```

The semantics of ppsp_tp_peer_addr_t attributes are listed in
Table 3:

```
+---------------------+-------------------------------+
| Element or Attribute | Description                   |
+---------------------+-------------------------------+
|      ip_address     | IP address information        |
|      port           | IP service port value         |
|      priority       | The priority of this interface. |
|                     | It may be determined by network |
|                     | topology preference, operator |
|                     | policy preference, etc.  How to |
|                     | create a priority is outside of |
|                     | the scope.  The larger the value, |
|                     | the higher the priority.      |
|      type           | Describes the address for NAT |
|                     | traversal, which can be HOST  |
|                     | REFLEXIVE or PROXY            |
|      connection     | Access type (wireless or wired) |
|      asn            | Autonomous System Number      |
|      peer_protocol  | Peer-to-Peer Streaming Peer   |
|                     | Protocol (PPSPP) supported    |
+---------------------+-------------------------------+
```

            Table 3: Semantics of ppsp_tp_peer_addr_t

   In this document, IP address is specified as ppsp_tp_addr_value. The
   exact characters and format depend on address_type:

   o The IPv4 address is encoded as specified by the "IPv4address" rule
      in Section 3.2.2 of [RFC3986].

   o The IPv6 address is encoded as specified in Section 4 of
      [RFC5952].

      Object {
              ppsp_tp_string_t   address_type;
              ppsp_tp_addr_value address;
      } ppsp_tp_ip_address;

   The peer information in responses is grouped in a
   ppsp_tp_peer_group_t element:

      Object {
              ppsp_tp_peer_info_t peer_info<1..*>;
      } ppsp_tp_peer_group_t;

3.2.5. Statistics and Status Information Element

   The statistics element (stat) is used to describe several properties
   relevant to the P2P network. These properties can be related to
   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. This specification
   only defines the property type "STREAM_STATS".

   The definition of the statistic element and attributes is as follows:

```
      Object {
              ppsp_tp_string_t   swarm_id;
              ppsp_tp_integer_t  uploaded_bytes;
              ppsp_tp_integer_t  downloaded_bytes;
              ppsp_tp_integer_t  available_bandwidth;
              ppsp_tp_integer_t  concurrent_links;
      } stream_stats;
```

   The semantics of stream_stats attributes are listed in Table 4:

   +----------------------+---------------------------------+
   | Element or Attribute | Description |
   +----------------------+---------------------------------+
swarm_id	Swarm ID
uploaded_bytes	Bytes sent to swarm
downloaded_bytes	Bytes received from swarm
available_bandwidth	Available instantaneous upload
	bandwidth
concurrent_links	Number of concurrent links
   +----------------------+---------------------------------+

                 Table 4: Semantics of stream_stats

   The stat information is grouped in the ppsp_tp_stat_group_t element:

```
      Object {
         ppsp_tp_string_t      type = "STREAM_STATS"; // property type
         stream_stats          stat<1..*>;
      } ppsp_tp_stat_group_t
```

   Other properties may be defined, related, for example, to 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 specific information for new properties, elements, or
attributes (see the guidelines in Section 7).

3.3. Requests and Responses

   This section defines the structure of PPSTP requests and responses.

3.3.1. Request Types

   The request type includes CONNECT, FIND, and STAT_REPORT, defined as
   follows:

```
ppsp_tp_string_t ppsp_tp_request_type_t = "CONNECT"
                                         | "FIND"
                                         | "STAT_REPORT";
```

3.3.2. Response Types

   Response type corresponds to the response method type of the message,
   defined as follows:

```
JSONValue ppsp_tp_response_type_t = 0x00    // SUCCESSFUL
                                  | 0x01;   // FAILED
```

3.3.3. Request Element

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

   The generic definition of a request element is as follows:

```
   Object {
           [ppsp_tp_peer_num_t        peer_num;]
           [ppsp_tp_peer_addr_t       peer_addr<1..*>;]
           ppsp_tp_swarm_action_t    swarm_action<1..*>;
   } ppsp_tp_request_connect;

   Object {
           ppsp_tp_string_t          swarm_id;
           [ppsp_tp_peer_num_t       peer_num;]
   } ppsp_tp_request_find;

   Object {
           ppsp_tp_version_t          version;
           ppsp_tp_request_type_t    request_type;
           ppsp_tp_string_t          transaction_id;
           ppsp_tp_string_t          peer_id;
           JSONValue request_data = ppsp_tp_req_connect connect
                                   | ppsp_tp_req_find     find
                                   | ppsp_tp_stat_group_t stat_report;
   } ppsp_tp_request;
```

   A request element consists of the version of PPSTP, the request type,
   a transaction ID, the requesting peer ID, and requesting body (i.e.,
   request_data). The request_data MUST be correctly set to the
   corresponding element based on the request type (see Table 5).

```
        +---------------------+---------------------+
        | request_type        | request_data        |
        +---------------------+---------------------+
        | "CONNECT"           | "connect"           |
        | "FIND"              | "find"              |
        | "STAT_REPORT"       | "stat_report"       |
        +---------------------+---------------------+
```

   Table 5: The Relationship between request_type and request_data

3.3.4. Response Element

   The generic definition of a response element is as follows:

```
   Object {
            ppsp_tp_version_t                version;
            ppsp_tp_response_type_t          response_type;
            ppsp_tp_integer_t                error_code;
            ppsp_tp_string_t                 transaction_id;
            [ppsp_tp_peer_addr_t             peer_addr;]
            [ppsp_tp_swarm_action_result_t swarm_result<1..*>;]
   } ppsp_tp_response;
```

   A response element consists of the version of PPSTP, the response
   type, the error code, a transaction ID, and optionally the public
   address of the requesting peer and one or multiple swarm action
   result elements. Normally, swarm action result elements SHOULD be
   present and error_code MUST be set to 00 (No Error) when
   response_type is 0x00. Swarm action result elements SHOULD NOT be
   set when error_code is 01 (Bad Request). Detailed selection of
   error_code is introduced in Section 4.3.

```
   Object {
       ppsp_tp_string_t           swarm_id;
       ppsp_tp_response_type_t     result;
       [ppsp_tp_peer_group_t       peer_group;]
   } ppsp_tp_swarm_action_result_t;
```

   A swarm action result element represents the result of an action
   requested by the peer. It contains a swarm identifier that globally
   indicates the swarm, the result for the peer of this action (which
   could be CONNECT ("JOIN" or "LEAVE"), FIND, or STAT_REPORT), and
   optionally one peer group element. The attribute result indicates
   the operation result of the corresponding request. When the response
   element corresponds to the STAT_REPORT request or the result
   attribute is set to 0x01, the peer group element SHOULD NOT be set.

3.4. PPSTP Message Element

   PPSTP messages (requests or responses) are designed to have a similar
   structure with a root field named "PPSPTrackerProtocol" containing
   meta information and data pertaining to a request or a response.

   The base type of a PPSTP message is defined as follows:

```
      Object {
              JSONValue PPSPTrackerProtocol = ppsp_tp_request  Request
                                            | ppsp_tp_response Response;
      } ppsp_tp_message_root;
```

4. Protocol Specification: Encoding and Operation

   PPSTP is a message-oriented request/response protocol. PPSTP
   messages use a text type encoding in JSON [RFC7159], which MUST be
   indicated in the Content-Type field in HTTP/1.1 [RFC7231], specifying
   the "application/ppsp-tracker+json" media type for all PPSTP request
   parameters and responses.

   Implementations MUST support the "https" URI scheme [RFC2818] and
   Transport Layer Security (TLS) [RFC5246].

   For deployment scenarios where peer (client) authentication is
   desired at the tracker, HTTP Digest Access Authentication [RFC7616]
   MUST be supported, with TLS Client Authentication as the preferred
   mechanism, if available.

   PPSTP uses the HTTP POST method to send parameters in requests to
   provide information resources that are the function of one or more of
   those input parameters. Input parameters are encoded in JSON in the
   HTTP entity body of the request.

   The section describes the operation of the three types of requests of
   PPSTP and provides some examples of usage.

4.1. Requests and Responses

4.1.1. CONNECT Request

   This method is used when a peer registers to the system and/or
   requests some swarm actions (join/leave). The peer MUST properly set
   the request type to CONNECT, generate and set the transaction_ids,
   set the peer_id, and include swarms the peer is interested in,
   followed by the corresponding action type and peer mode.

   o When a peer already possesses content and agrees to share it with
      others, it should set the action type to the value JOIN, as well
      as set the peer mode to SEEDER during its start (or re-start)
      period.

   o When a peer makes a request to join a swarm to consume content, it
      should set the action type to the value JOIN, as well as set the
      peer mode to LEECH during its start (or re-start) period.

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

   When a peer plans to leave a previously joined swarm, it should set
   action type to LEAVE, regardless of the peer mode.

   When receiving a well-formed CONNECT request message, the tracker
   starts 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. If successful, a response message with a
   corresponding response value of SUCCESSFUL will be generated.

   The valid sets of the number of swarms whose action type is combined
   with peer mode for the CONNECT request logic are enumerated in
   Table 6 (referring to the "per-Peer-ID" State Machine in
   Section 2.3).

Swarm Number	peer_mode 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	SEEDER	JOIN	START	TRACKING	Valid
N	SEEDER	JOIN	TRACKING	TERMINATE	Invalid
N	SEEDER	LEAVE	TRACKING	TERMINATE	Valid

Table 6: Validity of Action Combinations in CONNECT Requests

In the CONNECT request message, multiple swarm action elements ppsp_tp_swarm_action_t could be contained. Each of them contains the request action and the peer_mode of the peer. The peer_mode attribute MUST be set to the type of participation of the peer in the swarm (SEEDER or LEECH).

The CONNECT message may contain multiple peer_addr elements with attributes ip_address, port, priority, and type (if Interactive Connectivity Establishment (ICE) [RFC5245] NAT traversal techniques are used), and optionally connection, asn, and peer_protocol corresponding to each of the network interfaces the peer wants to advertise.

The element peer_num indicates the maximum number of peers to be returned in a list from the tracker. The returned peer list can be optionally filtered by some indicated properties, such as ability_nat for NAT traversal, and concurrent_links, online_time and upload_bandwidth for the preferred capabilities.

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

The response may include peer_addr data of the requesting peer public
IP address. Peers can use Session Traversal Utilities for NAT (STUN)
[RFC5389] and Traversal Using Relays around NAT (TURN) [RFC5766] to
gather their candidates, in which case peer_addr SHOULD NOT present
in the response. If no STUN is used and the tracker is able to work
as a "STUN-like" server that can inspect the public address of a
peer, the tracker can return the address back with a "REFLEXIVE"
attribute type. The swarm_result may also include peer_addr 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.

If the peer_mode is SEEDER, the tracker responds with a SUCCESSFUL
response and enters the peer information into the corresponding swarm
activity. If the peer_mode is LEECH (or if a SEEDER includes a
peer_num 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 Application-Layer Traffic Optimization
(ALTO) [RFC7285].

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

4.1.1.1. Example

   The following example of a CONNECT request corresponds to a peer that
   wants to start (or re-start) sharing its previously streamed content
   (peer_mode is SEEDER).

```
POST https://tracker.example.com/video_1 HTTP/1.1
Host: tracker.example.com
Content-Length: 494
Content-Type: application/ppsp-tracker+json
Accept: application/ppsp-tracker+json

{
  "PPSPTrackerProtocol": {
    "version":               1,
    "request_type":          "CONNECT",
    "transaction_id":        "12345",
    "peer_id":               "656164657220",
    "connect":{
        "peer_addr": {
              "ip_address": {
                    "address_type":     "ipv4",
                    "address":          "192.0.2.2"
              },
              "port":        80,
              "priority":    1,
              "type":        "HOST",
              "connection":  "wired",
              "asn":         "45645"
        },
        "swarm_action": [{
            "swarm_id":        "1111",
            "action":          "JOIN",
            "peer_mode":       "SEEDER"
        },
        {
            "swarm_id":        "2222",
            "action":          "JOIN",
            "peer_mode":       "SEEDER"
        }]
    }
  }
}
```

   Another example of the message-body of a CONNECT request corresponds
   to a peer (peer_mode is LEECH, meaning that the peer is not in
   possession of the content) requesting join to a swarm, in order to

start receiving the stream and providing optional information on the
addresses of its network interface(s):

```
      {
        "PPSPTrackerProtocol": {
          "version":                1,
          "request_type":           "CONNECT",
          "transaction_id":         "12345.0",
          "peer_id":                "656164657221",
          "connect":{
              "peer_num": {
                  "peer_count":         5,
                  "ability_nat":        "STUN",
                  "concurrent_links":   "5",
                  "online_time":        "200",
                  "upload_bandwidth":   "600"
              },
              "peer_addr": [{
                  "ip_address": {
                      "address_type":       "ipv4",
                      "address":            "192.0.2.2"
                  },
                  "port":         80,
                  "priority":     1,
                  "type":         "HOST",
                  "connection":   "wired",
                  "asn":          "3256546"
              },
              {
                  "ip_address":{
                      "address_type":       "ipv6",
                      "address":            "2001:db8::2"
                  },
                  "port":         80,
                  "priority":     2,
                  "type":         "HOST",
                  "connection":   "wireless",
                  "asn":          "34563456",
                  "peer_protocol": "PPSP-PP"
              }],
              "swarm_action": {
                 "swarm_id":        "1111",
                 "action":          "JOIN",
                 "peer_mode":       "LEECH"
              }
          }
        }
      }
```

The next example of a CONNECT request corresponds to a peer leaving a
previously joined swarm and requesting to join a new swarm. This is
the typical example of a user watching a live channel but then
deciding to switch to a different one:

```
{
   "PPSPTrackerProtocol": {
      "version":                1,
      "request_type":           "CONNECT",
      "transaction_id":         "12345",
      "peer_id":                "656164657221",
      "connect":{
          "peer_num": {
              "peer_count":         5,
              "ability_nat":        "STUN",
              "concurrent_links":   "5",
              "online_time":        "200",
              "upload_bandwidth":   "600"
          },
          "swarm_action": [{
              "swarm_id":           "1111",
              "action":             "LEAVE",
              "peer_mode":          "LEECH"
          },
          {
              "swarm_id":           "2222",
              "action":             "JOIN",
              "peer_mode":          "LEECH"
          }]
      }
   }
}
```

The next example illustrates the response for the previous example of
a CONNECT request where the peer requested two swarm actions and not
more than 5 other peers, receiving from the tracker a peer list with
only two other peers in the swarm "2222":

```
HTTP/1.1 200 OK
Content-Length: 1342
Content-Type: application/ppsp-tracker+json

{
   "PPSPTrackerProtocol": {
      "version":                1,
      "response_type":          0,
      "error_code":             0,
      "transaction_id":         "12345",
```

```
       "peer_addr": {
           "ip_address": {
               "address_type":      "ipv4",
               "address":           "198.51.100.1"
           },
           "port":           80,
           "priority":       1,
           "asn":            "64496"
       },
       "swarm_result": {
           "swarm_id":          "2222",
           "result":            0,
           "peer_group": {
               "peer_info": [{
                   "peer_id":    "956264622298",
                   "peer_addr": {
                       "ip_address": {
                           "address_type":      "ipv4",
                           "address":           "198.51.100.22"
                       },
                       "port":           80,
                       "priority":       2,
                       "type":           "REFLEXIVE",
                       "connection":     "wired",
                       "asn":            "64496",
                       "peer_protocol": "PPSP-PP"
                   }
               },
               {
                   "peer_id":    "3332001256741",
                   "peer_addr": {
                       "ip_address": {
                           "address_type":      "ipv4",
                           "address":           "198.51.100.201"
                       },
                       "port":           80,
                       "priority":       2,
                       "type":           "REFLEXIVE",

                       "connection":     "wired",
                       "asn":            "64496",
                       "peer_protocol": "PPSP-PP"
                   }
               }]
           }
       }
   }
 }
```

4.1.2. FIND Request

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

   The FIND request may include a peer_number element to indicate to the
   tracker the maximum number of peers to be returned in a list
   corresponding to the indicated conditions set by the requesting peer,
   being ability_nat for NAT traversal (considering that PPSP-ICE NAT
   traversal techniques may be used), and optionally concurrent_links,
   online_time, and upload_bandwidth for the preferred capabilities.

   When receiving a well-formed FIND request, the tracker processes the
   information to check if it is valid. If successful, a response
   message with a response value of SUCCESSFUL will be generated, and
   the tracker will search out the list of peers for the swarm and
   select an appropriate peer list satisfying the conditions set by the
   requesting peer. The peer list returned MUST contain the peer IDs
   and the corresponding IP addresses.

   The tracker may take the ability of peers and popularity of the
   requested content into consideration. For example, the tracker could
   select peers with higher ability than the current peers that provide
   the content if the content is relatively popular (see Section 5.1.1);
   the tracker could also select peers with lower ability than the
   current peers that provide the content when the content is relatively
   uncommon. The tracker may take network location information into
   consideration as well, to express network topology preferences or
   operators' policy preferences. It can implement other IETF efforts
   like ALTO [RFC7285], which is out of the scope of this document.

   The response MUST include a peer_group element that contains the peer
   IDs and the corresponding IP addresses; it 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.

   The response may also include a peer_addr element that includes the
   requesting peer public IP address. If no STUN is used and the
   tracker is able to work as a "STUN-like" server that can inspect the
   public address of a peer, the tracker can return the address back
   with a "REFLEXIVE" attribute type.

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

4.1.2.1. Example

   An example of the message-body of a FIND request, where the peer
   requests from the tracker a list of not more than 5 peers in the
   swarm "1111" conforming to the characteristics expressed (concurrent
   links, online time, and upload bandwidth level) is as follows:

```
     {
       "PPSPTrackerProtocol": {
           "version":              1,
           "request_type":         "FIND",
           "transaction_id":       "12345",
           "peer_id":              "656164657221",
           "swarm_id":             "1111",
           "peer_num": {
               "peer_count":        5,
               "ability_nat":       "STUN",
               "concurrent_links":  "5",
               "online_time":       "200",
               "upload_bandwidth":  "600"
           }
       }
     }
```

   An example of the message-body of a response for the above FIND
   request, including the requesting peer public IP address information,
   is as follows:

```
     {
       "PPSPTrackerProtocol": {
           "version":              1,
           "response_type":        0,
           "error_code":           0,
           "transaction_id":       "12345",
           "swarm_result": {
               "swarm_id":         "1111",
               "result":           0,
               "peer_group": {
                   "peer_info": [{

                       "peer_id":     "656164657221",
                       "peer_addr": {
                           "ip_address": {
                               "address_type":     "ipv4",
                               "address":          "198.51.100.1"
                           },
                           "port":          80,
                           "priority":      1,
```

```
                          "type":           "REFLEXIVE",
                          "connection":     "wireless",
                          "asn":            "64496"
                      }
                  },
                  {
                      "peer_id":     "956264622298",
                      "peer_addr": {
                          "ip_address": {
                              "address_type":     "ipv4",
                              "address":          "198.51.100.22"
                          },
                          "port":           80,
                          "priority":       1,
                          "type":           "REFLEXIVE",
                          "connection":     "wireless",
                          "asn":            "64496"
                      }
                  },
                  {
                      "peer_id":     "3332001256741",
                      "peer_addr": {
                          "ip_address": {
                              "address_type":     "ipv4",
                              "address":          "198.51.100.201"
                          },
                          "port":           80,
                          "priority":       1,
                          "type":           "REFLEXIVE",

                          "connection":     "wireless",
                          "asn":            "64496"
                      }
                  }]
              }
          }
      }
  }
```

4.1.3. STAT_REPORT Request

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

   The peer MUST set the request_type to "STAT_REPORT", set the peer_id
   with the identifier of the peer, and generate and set the
   transaction_id.

   The report may include multiple statistics elements describing
   several properties relevant to a specific swarm. These properties
   can be related with stream statistics and peer status information,
   including uploaded_bytes, downloaded_bytes, available_bandwidth,
   concurrent_links, etc.

   Other properties may be defined (see the guidelines in Section 7.1),
   for example, those related to incentives and reputation mechanisms.
   If no Statistics Group is included, the STAT_REPORT is used as a
   "keep-alive" message to prevent the tracker from de-registering the
   peer when the "track timer" expires.

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

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

4.1.3.1. Example

   An example of the message-body of a STAT_REPORT request is:

```
   {
     "PPSPTrackerProtocol": {
         "version":             1,
         "request_type":        "STAT_REPORT",
         "transaction_id":      "12345",
         "peer_id":             "656164657221",
         "stat_report": {
             "type":  "STREAM_STATS",
             "Stat": {
                     "swarm_id":              "1111",
                     "uploaded_bytes":        512,
                     "downloaded_bytes":      768,
                     "available_bandwidth":   1024000,
                     "concurrent_links":      5
             }
         }
     }
   }
```

An example of the message-body of a response for the START_REPORT
request is:

```
{
  "PPSPTrackerProtocol": {
      "version":              1,
      "response_type":        0,
      "error_code":           0,
      "transaction_id":       "12345",
      "swarm_result": {
          "swarm_id":      "1111",
          "result":        0
      }
  }
}
```

4.2. Response Element in Response Messages

   Table 7 indicates the response type and corresponding semantics.

```
+-------------------+-------------------+
| Response Type     | Semantics         |
|                   |                   |
+-------------------+-------------------+
| 0                 | SUCCESSFUL        |
| 1                 | FAILED            |
+-------------------+-------------------+
```

            Table 7: Semantics for the Value of Response Type

   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
   transaction_id as the corresponding request.

      CONNECT: 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 action attribute
      requested.

      FIND: Returns the list of peers corresponding to the requested
      scope.

      STAT_REPORT: Confirms the success of the requested operation.

   FAILED: Indicates that the request has not been processed properly.
   A corresponding error_code SHOULD be set according to the conditions
   described in Section 4.3.

4.3. Error and Recovery Conditions

   If the peer receives an invalid response, the same request with
   identical content including the same transaction_id MUST be repeated.

   The transaction_id 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 MUST be prepared to receive a request with a repeated
   transaction_id.

   Error situations resulting from normal operation or from abnormal
   conditions (Section 2.3.2) MUST be responded to with response_type
   set to 0x01 and with the adequate error_code, as described here:

   o If the message is found to be incorrectly formed, the receiver
      MUST respond with a 01 (Bad Request) error_code with an empty
      message-body (no peer_addr and swarm_result attributes).

   o If the version number of the protocol is for a version the
      receiver does not support, the receiver MUST respond with a 02
      (Unsupported Version Number) error_code with an empty message-body
      (no peer_addr and swarm_result attributes).

   o In the PEER REGISTERED and TRACKING states of the tracker, certain
      requests are not allowed (Section 2.3.2). The tracker MUST
      respond with a 03 (Forbidden Action) error_code with an empty
      message-body (no peer_addr and swarm_result attributes).

   o If the tracker is unable to process a request message due to an
      unexpected condition, it SHOULD respond with a 04 (Internal Server
      Error) error_code with an empty message-body (no peer_addr and
      swarm_result attributes).

   o If the tracker is unable to process a request message because it
      is in an overloaded state, it SHOULD respond with a 05 (Service
      Unavailable) error_code with an empty message-body (no peer_addr
      and swarm_result attributes).

   o If authentication is required for the peer to make the request,
      the tracker SHOULD respond with a 06 (Authentication Required)
      error_code with an empty message-body (no peer_addr and
      swarm_result attributes).

4.4. Parsing of Unknown Fields in message-body

   This document only details object members used by this specification.
   Extensions may include additional members within JSON objects defined
   in this document. PPSTP implementations MUST ignore unknown members
   when processing PPSTP messages.

5. Operations and Manageability

   This section provides the operational and management aspects that are
   required to be considered in implementations of PPSTP. These aspects
   follow the recommendations expressed in [RFC5706].

5.1. Operational Considerations

   PPSTP 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
   content.

   The "server" 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 content.

5.1.1. Installation and Initial Setup

   Content providers wishing to use PPSP for content distribution should
   set up at least a PPSP tracker and a service portal (public web
   server) to publish links of the content descriptions, for access to
   their on-demand or live original content sources. Content and
   service providers should also create conditions to generate peer IDs
   and any required security certificates, as well as chunk IDs and
   swarm IDs for each streaming content. The configuration processes
   for the PPSP tracking facility, the service portal, and content
   sources are not standardized, enabling flexibility for implementers.

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

   The available content could have different importance attribute
   values to indicate whether the content is popular or not. However,
   it is a totally implementation design and outside the scope of this

specification. For example, the importance attribute values of the
content could be set by content providers when distributing them or
could be determined by the tracker based on the statistics of the
requests from the peers that request the content. The tracker could
set an upper threshold to decide that the content is popular enough
when the importance attribute value is higher than the upper
threshold. The tracker could also set a lower threshold to decide
that the content is uncommon enough when the importance attribute
value is lower than the lower threshold.

End users browse and search for desired content in the service portal
and select by clicking the links of the corresponding MPDs. This
action typically requires security certificates or authorization
tokens from an enrollment service (end-user registration) and then
launches the Client Media Player (with PPSP awareness), which will
then, using PPSTP, contact the PPSP tracker to join the corresponding
swarm and obtain the transport addresses of other PPSP peers in order
to start streaming the content.

5.1.2. Migration Path

There is no previous standard protocol providing functionality
similar to PPSTP. However, some popular proprietary protocols, e.g.,
BitTorrent, are used in existing systems. There is no way for PPSTP
to migrate to proprietary protocols like the BitTorrent tracker
protocol. Because PPSTP is an application-level protocol, there is
no harm in PPSTP having no migration path. However, proprietary
protocols migrating to standard protocols like PPSTP can solve the
problems raised in [RFC6972]. It is also possible for systems to use
PPSTP as the management protocol to work with exiting propriety peer
protocols like the BitTorrent peer protocol.

5.1.3. Requirements on Other Protocols and Functional Components

For security reasons, when using the Peer-to-Peer Streaming Peer
Protocol (PPSPP) with PPSTP, the mechanisms described in Section 6.1
should be observed.

5.1.4. Impact on Network Operation

As the messaging model of PPSTP aligns with HTTP and the semantics of
its messages, the impact on network operation is similar to using
HTTP.

5.1.5. Verifying Correct Operation

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

5.2. Management Considerations

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

5.2.1. Interoperability

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

5.2.2. Management Information

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

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

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

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

   Logging is an important functionality for PPSTP trackers and peers;
   it is done via syslog [RFC5424].

5.2.3. Fault Management

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

5.2.4. Configuration Management

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

5.2.5. Accounting Management

   PPSTP implementations, primarily in content provider environments,
   can benefit from accounting standardization efforts as described in
   [RFC2975], which indicates that accounting management is "concerned
   with the collection of resource consumption data for the purposes of
   capacity and trend analysis, cost allocation, auditing, and billing".

5.2.6. Performance Management

   Because PPSTP is transaction oriented, its performance in terms of
   availability and responsiveness can be measured with the facilities
   of the APM-MIB [RFC3729] and the TPM-MIB [RFC4150].

5.2.7. Security Management

   Standard SNMP notifications for PPSP tracker management [RFC5590] and
   syslog messages [RFC5424] can be used to alert operators to the
   conditions identified in the security considerations (Section 6).

   The statistics collected about the operation of PPSTP can be used for
   detecting attacks (e.g., the receipt of malformed messages, messages
   out of order, or messages with invalid timestamps). However,
   collecting such endpoint properties may also raise some security
   issues. For example, the statistics collected by the tracker may be
   disclosed to an unauthorized third party that has malicious
   intentions. To address such risk, the provider of the tracker should

evaluate how much information is revealed and the associated risks.
A confidentiality mechanism must be provided by HTTP over TLS to
guarantee the confidentiality of PPSTP.

6. Security Considerations

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

   No security system can guarantee complete security in an open P2P
   streaming system where participants may be malicious or
   uncooperative. The goal of the security considerations described
   here is to provide sufficient protection for maintaining some
   security properties during 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
   security considerations described in [RFC7230] and [RFC7231] also
   apply. 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 a
   bearer token, which provides the peer with the information required
   to successfully utilize an access token to make protected requests to
   the tracker.

6.1. Authentication between Tracker and Peers

   To protect PPSTP 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 the scope of this
   specification.

   Transport Layer Security (TLS) [RFC5246] MUST be used in the
   communication between peers and tracker to provide privacy and data
   integrity. Software engineers developing and service providers
   deploying the tracker should make themselves familiar with the Best
   Current Practices (BCP) on configuring HTTP over TLS [RFC7525].

OAuth 2.0 Authorization [RFC6749] SHOULD also be considered when
digest authentication [RFC7616] and HTTPS client certificates are
required.

6.2. Content Integrity Protection against Polluting Peers/Trackers

Malicious peers may claim ownership of popular content to the tracker
and try to serve polluted (i.e., decoy content or even virus/trojan-
infected content) to other peers. Since trackers do not exchange
content information among peers, it is difficult to detect whether or
not a peer is polluting the content. Usually, this kind of pollution
can be detected by the Peer-to-Peer Streaming Peer Protocol (PPSPP)
[RFC7574] with requiring the use of Merkle Hash Tree scheme for
protecting the integrity of the content. More details can be seen in
Section 5 of [RFC7574].

Some attackers that disrupt P2P streaming on behalf of content
providers may provide false or modified content or peer list
information to achieve certain malicious goals. Peers connecting to
those portals or trackers provided by the attackers may be redirected
to some corrupted malicious content. However, there is no standard
way for peers to avoid this kind of situation completely. Peers can
have mechanisms to detect undesirable content or results themselves.
For example, if a peer finds that the portal returned some undesired
content information or the tracker returned some malicious peer
lists, the peer may choose to quit the swarm or switch to other P2P
streaming services provided by other content providers.

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 in 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 may consider additional pro-
   incentive parameters (see the guidelines for extension in Section 7),
   which can enable the tracker to improve the performance of the whole
   P2P streaming system. Trustworthiness of these pro-incentive
   parameters is critical to the effectiveness of the incentive
   mechanisms. Furthermore, the amount of both uploaded and downloaded
   data should be reported to the tracker to allow checking for
   inconsistencies between the upload and download report and to
   establish an appropriate credit/trust system.

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

   Alternatively, exchange of cryptographic receipts signed by receiving
   peers can be used to attest to the upload contribution of a peer to
   the swarm, as suggested in [Contracts].

6.5 Privacy for Peers

   PPSTP provides mechanisms in which the peers can send messages
   containing IP addresses, ports, and other information to the tracker.
   A tracker or a third party who is able to intercept such messages can
   store and process the obtained information in order to analyze peers'
   behaviors and communication patterns. Such analysis can lead to
   privacy risks. For example, an unauthorized party may snoop on the
   data transmission from the peer to a tracker in order to introduce
   some corrupted chunks.

   The Peer-to-Peer Streaming Peer Protocol (PPSPP) [RFC7574] has
   already introduced some mechanisms to protect streamed content; see
   Sections 12.3 and 12.4 of [RFC7574]. For PPSTP, peer implementations
   as well as tracker implementations MUST support the "https" URI
   scheme [RFC2818] and Transport Layer Security (TLS) [RFC5246]. In
   addition, a peer should be cognizant about potential trackers
   tracking through queries of peers, e.g., by using HTTP cookies.
   PPSTP as specified in this document does not rely on HTTP cookies.
   Thus, peers may decide not to return cookies received from the
   tracker, in order to make additional tracking more difficult.

7. Guidelines for Extending PPSTP

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

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

   In PPSTP, this 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 MUST be documented in Standards Track RFCs if there are
   requirements for coordination, interoperability, and broad
   distribution.

7.1. Forms of PPSTP Extension

   In PPSTP, 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 Statistics Group can be
      extended to include additional elements needed to express status
      information about the activity of the peer, such as online time
      for the stat element.

   o Protocol-Level Extension: If there is no existing element mapping
      that can be extended to meet the requirements and the existing
      PPSTP 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 LEECH, and then disconnect from the latter swarm but remain 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;SEEDER)---------->|
                     |<------------------------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 PPSTP Extended Version

7.2. Issues to Be Addressed in PPSTP Extensions

   There are several issues that all extensions should take into
   consideration.

   o Overview of the Extension: It is RECOMMENDED that extensions to
      PPSTP 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.

   o Backward Compatibility: The new extension MUST be backward
      compatible with the base PPSTP specified in this document.

   o 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 PPSTP. Method names SHOULD be
      shorter than 16 characters and SHOULD attempt to convey the
      general meaning of the request or response.

   o Semantic Issues: PPSTP 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.

   o Security Issues: As security is an important component of any
      protocol, designers of PPSTP extensions need to carefully consider
      security requirements, e.g., authorization requirements and
      requirements for end-to-end integrity.

   o 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

8.1. MIME Type Registry

   This document registers "application/ppsp-tracker+json" media types.

   Type name: application

   Subtype name: ppsp-tracker+json

   Required parameters: n/a

   Optional parameters: n/a

   Encoding considerations: Encoding considerations are identical to
   those specified for the "application/json" media type. See
   [RFC7159].

   Security considerations: See Section 6 of RFC 7846.

   Interoperability considerations: This document specifies the format
   of conforming messages and the interpretation thereof.

   Published specification: RFC 7846.

   Applications that use this media type: PPSP trackers and peers
   either stand alone or are embedded within other applications.

   Additional information:

      Magic number(s): n/a

      File extension(s): n/a

      Macintosh file type code(s): n/a

      Fragment identifier considerations: n/a

   Person & email address to contact for further information: See
   Authors' Addresses section.

   Intended usage: COMMON

   Restrictions on usage: none

   Author: See Authors' Addresses section of RFC 7846.

   Change controller: IESG (iesg@ietf.org)

8.2. PPSTP Version Number Registry

   IANA has created the "PPSTP Version Number Registry". Values are
   integers in the range 0-255, with initial assignments and
   reservations given in Table 2. New PPSTP version types are assigned
   after IETF Review [RFC5226] to ensure that proper documentation
   regarding the new version types and their usage has been provided.

8.3. PPSTP Request Type Registry

   IANA has created the "PPSTP Request Type Registry". Values are
   strings listed in Table 8. New PPSTP request types are assigned
   after IETF Review [RFC5226] to ensure that proper documentation
   regarding the new request types and their usage has been provided.

```
   +---------------------+-------------------------------------------+
   | request_type        | Description                               |
   +---------------------+-------------------------------------------+
   | "CONNECT"           | 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 action attribute                      |
   |                     | requested.                                |
   |                     |                                           |
   | "FIND"              | Returns the list of peers corresponding   |
   |                     | to the requested scope.                   |
   |                     |                                           |
   | "STAT_REPORT"       | Confirms the success of the requested     |
   |                     | operation.                                |
   +---------------------+-------------------------------------------+
```

          Table 8: The PPSTP Request Type Registry

8.4. PPSTP Error Code Registry

   IANA has created the "PPSTP Error Code Registry". Values are the
   strings listed in Table 9. New PPSTP error codes are assigned after
   IETF Review [RFC5226] to ensure that proper documentation regarding
   the new error codes and their usage has been provided.

```
+----------------+--------------------------------------------+
| error_code     | Description                                |
+----------------+--------------------------------------------+
| 00             | No Error                                   |
| 01             | Bad Request                                |
| 02             | Unsupported Version Number                 |
| 03             | Forbidden Action                           |
| 04             | Internal Server Error                      |
| 05             | Service Unavailable                        |
| 06             | Authentication Required                    |
+----------------+--------------------------------------------+
```

      Table 9: The PPSTP Error Code Registry

9. References

9.1. Normative References

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
               Requirement Levels", BCP 14, RFC 2119,
               DOI 10.17487/RFC2119, March 1997,
               <http://www.rfc-editor.org/info/rfc2119>.

   [RFC2818]   Rescorla, E., "HTTP Over TLS", RFC 2818,
               DOI 10.17487/RFC2818, May 2000,
               <http://www.rfc-editor.org/info/rfc2818>.

   [RFC3629]   Yergeau, F., "UTF-8, a transformation format of ISO
               10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November
               2003, <http://www.rfc-editor.org/info/rfc3629>.

   [RFC3986]   Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
               Resource Identifier (URI): Generic Syntax", STD 66, RFC
               3986, DOI 10.17487/RFC3986, January 2005,
               <http://www.rfc-editor.org/info/rfc3986>.

   [RFC5245]   Rosenberg, J., "Interactive Connectivity Establishment
               (ICE): A Protocol for Network Address Translator (NAT)
               Traversal for Offer/Answer Protocols", RFC 5245,
               DOI 10.17487/RFC5245, April 2010,
               <http://www.rfc-editor.org/info/rfc5245>.

   [RFC5246]   Dierks, T. and E. Rescorla, "The Transport Layer Security
               (TLS) Protocol Version 1.2", RFC 5246,
               DOI 10.17487/RFC5246, August 2008,
               <http://www.rfc-editor.org/info/rfc5246>.

   [RFC5389]   Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
               "Session Traversal Utilities for NAT (STUN)", RFC 5389,
               DOI 10.17487/RFC5389, October 2008,
               <http://www.rfc-editor.org/info/rfc5389>.

   [RFC5590]   Harrington, D. and J. Schoenwaelder, "Transport Subsystem
               for the Simple Network Management Protocol (SNMP)", STD
               78, RFC 5590, DOI 10.17487/RFC5590, June 2009,
               <http://www.rfc-editor.org/info/rfc5590>.

   [RFC5766]   Mahy, R., Matthews, P., and J. Rosenberg, "Traversal
               Using Relays around NAT (TURN): Relay Extensions to
               Session Traversal Utilities for NAT (STUN)", RFC 5766,
               DOI 10.17487/RFC5766, April 2010,
               <http://www.rfc-editor.org/info/rfc5766>.

   [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6
              Address Text Representation", RFC 5952,
              DOI 10.17487/RFC5952, August 2010,
              <http://www.rfc-editor.org/info/rfc5952>.

   [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J.,
              Ed., and A. Bierman, Ed., "Network Configuration Protocol
              (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
              <http://www.rfc-editor.org/info/rfc6241>.

   [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework",
              RFC 6749, DOI 10.17487/RFC6749, October 2012,
              <http://www.rfc-editor.org/info/rfc6749>.

   [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON)
              Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159,
              March 2014, <http://www.rfc-editor.org/info/rfc7159>.

   [RFC7230] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext
              Transfer Protocol (HTTP/1.1): Message Syntax and
              Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014,
              <http://www.rfc-editor.org/info/rfc7230>.

   [RFC7231] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext
              Transfer Protocol (HTTP/1.1): Semantics and Content", RFC
              7231, DOI 10.17487/RFC7231, June 2014,
              <http://www.rfc-editor.org/info/rfc7231>.

   [RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel,
              S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy,
              "Application-Layer Traffic Optimization (ALTO) Protocol",
              RFC 7285, DOI 10.17487/RFC7285, September 2014,
              <http://www.rfc-editor.org/info/rfc7285>.

   [RFC7574] Bakker, A., Petrocco, R., and V. Grishchenko, "Peer-to-
              Peer Streaming Peer Protocol (PPSPP)", RFC 7574,
              DOI 10.17487/RFC7574, July 2015,
              <http://www.rfc-editor.org/info/rfc7574>.

   [RFC7616] Shekh-Yusef, R., Ed., Ahrens, D., and S. Bremer, "HTTP
              Digest Access Authentication", RFC 7616,
              DOI 10.17487/RFC7616, September 2015,
              <http://www.rfc-editor.org/info/rfc7616>.

9.2. Informative References

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

   [RFC2564]   Kalbfleisch, C., Krupczak, C., Presuhn, R., and J.
               Saperia, "Application Management MIB", RFC 2564,
               DOI 10.17487/RFC2564, May 1999,
               <http://www.rfc-editor.org/info/rfc2564>.

   [RFC2790]   Waldbusser, S. and P. Grillo, "Host Resources MIB", RFC
               2790, DOI 10.17487/RFC2790, March 2000,
               <http://www.rfc-editor.org/info/rfc2790>.

   [RFC2975]   Aboba, B., Arkko, J., and D. Harrington, "Introduction to
               Accounting Management", RFC 2975, DOI 10.17487/RFC2975,
               October 2000, <http://www.rfc-editor.org/info/rfc2975>.

   [RFC3410]   Case, J., Mundy, R., Partain, D., and B. Stewart,
               "Introduction and Applicability Statements for Internet-
               Standard Management Framework", RFC 3410,
               DOI 10.17487/RFC3410, December 2002,
               <http://www.rfc-editor.org/info/rfc3410>.

   [RFC3729]   Waldbusser, S., "Application Performance Measurement
               MIB", RFC 3729, DOI 10.17487/RFC3729, March 2004,
               <http://www.rfc-editor.org/info/rfc3729>.

   [RFC4022]   Raghunarayan, R., Ed., "Management Information Base for
               the Transmission Control Protocol (TCP)", RFC 4022,
               DOI 10.17487/RFC4022, March 2005,
               <http://www.rfc-editor.org/info/rfc4022>.

   [RFC4122]   Leach, P., Mealling, M., and R. Salz, "A Universally
               Unique IDentifier (UUID) URN Namespace", RFC 4122,
               DOI 10.17487/RFC4122, July 2005,
               <http://www.rfc-editor.org/info/rfc4122>.

   [RFC4150]   Dietz, R. and R. Cole, "Transport Performance Metrics
               MIB", RFC 4150, DOI 10.17487/RFC4150, August 2005,
               <http://www.rfc-editor.org/info/rfc4150>.

   [RFC5226]   Narten, T. and H. Alvestrand, "Guidelines for Writing an
               IANA Considerations Section in RFCs", BCP 26, RFC 5226,
               DOI 10.17487/RFC5226, May 2008,
               <http://www.rfc-editor.org/info/rfc5226>.

   [RFC5424]   Gerhards, R., "The Syslog Protocol", RFC 5424, DOI
               10.17487/RFC5424, March 2009,
               <http://www.rfc-editor.org/info/rfc5424>.

   [RFC5706]   Harrington, D., "Guidelines for Considering Operations
               and Management of New Protocols and Protocol Extensions",
               RFC 5706, DOI 10.17487/RFC5706, November 2009,
               <http://www.rfc-editor.org/info/rfc5706>.

   [RFC6709]   Carpenter, B., Aboba, B., Ed., and S. Cheshire, "Design
               Considerations for Protocol Extensions", RFC 6709,
               DOI 10.17487/RFC6709, September 2012,
               <http://www.rfc-editor.org/info/rfc6709>.

   [RFC6972]   Zhang, Y. and N. Zong, "Problem Statement and
               Requirements of the Peer-to-Peer Streaming Protocol
               (PPSP)", RFC 6972, DOI 10.17487/RFC6972, July 2013,
               <http://www.rfc-editor.org/info/rfc6972>.

   [RFC7525]   Sheffer, Y., Holz, R., and P. Saint-Andre,
               "Recommendations for Secure Use of Transport Layer
               Security (TLS) and Datagram Transport Layer Security
               (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
               2015, <http://www.rfc-editor.org/info/rfc7525>.

   [SARACEN]   Sarecen P2P, <http://www.saracen-p2p.eu/>.

Acknowledgments

Authors' Addresses

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

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

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

   Rachel Huang (editor)
   Huawei
   Email: rachel.huang@huawei.com

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

   Deng Lingli
   China Mobile
   Email: denglingli@chinamobile.com