PPSP
INTERNET-DRAFT
Intended Status: Informational
Expires: December 22, 2011

Rui S. Cruz
IST/INESC-ID/INOV
Mario S. Nunes
IST/INESC-ID/INOV
Joao P. Taveira
IST/INOV

June 20, 2011

HTTP-based PPSP Tracker Protocol
draft-cruz-ppsp-http-tracker-protocol-01

Abstract

This document presents a proposal for an HTTP-based P2P streaming Tracker Protocol, outlining the requirements, functional entities, message flows, formal syntax and semantics, with detailed message processing instructions using an HTTP/XML encoding, the respective parameters, methods, and message formats. The PPSP Peer Protocol proposed in this document extends the capabilities of PPSP to support adaptive and scalable video and 3D video, for Video On Demand (VoD) and Live video services.

Status of this Memo

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

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

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

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

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

Copyright and License Notice

Table of Contents

1 Introduction

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

The PPSP Peer protocol controls the advertising and exchange of media data directly between peers.

The PPSP Tracker Protocol provides communication between Trackers and Peers, by which Peers exchange meta information with trackers, report streaming status and request candidate lists from trackers.

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

The EU research project SARACEN (Socially Aware, collaboRative, scAlable Coding mEdia distributioN) is developing an innovative P2P architecture that targets the optimization of the Quality of Experience in personalized media streaming, through the integration of scalable media coding and multi view coding techniques, and with respect to user privacy [refs.saracenwebpage].

The process used for streaming distribution in SARACEN relies on a chunk transfer scheme whereby the original content is re-encoded using adaptive or scalable techniques and then chopped into small video chunks with a short duration:

1. (adaptive) - alternate versions with different qualities and bitrates;
2. (scalable description levels) - multiple additive descriptions (i.e., addition of descriptions refine the quality of the video);
3. (scalable layered levels) - nested dependent layers corresponding to several hierarchical levels of quality, i.e., higher enhancement layers refine the quality of the video of lower layers.
4. (scalable multi views) - views correspond to 2D and to stereoscopic 3D videos, with several hierarchical levels of quality.

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

The signaling and the media transfer between PPSP peers in SARACEN is done using a request/reply mechanism as defined in HTTP-based PPSP Peer Protocol [I-D.cruz-ppsp-http-peer-protocol].

The HTTP-based PPSP Tracker Protocol used in SARACEN and presented in this draft follows the design defined in PPSP Tracker protocol [I-D.gu-ppsp-tracker-protocol], but extending the capabilities of PPSP to support adaptive and scalable video.

The goal of this draft is to derive from the work being developed in the SARACEN Project the implications for the standardization of the PPSP streaming protocols, believed as important inputs to the IETF PPSP working group, in order to identify open issues and promote further discussion. [I-D.ietf-ppsp-survey].

2 Terminology

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

This draft uses the terms defined in [I-D.ietf-ppsp-problem-statement] and in [I-D.gu-ppsp-tracker-protocol].

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

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

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

Chunk: A chunk is a basic unit of partitioned streaming media, which is used by a peer for the purpose of storage, advertisement and exchange among peers.

Enhancement Layer: enhancement differential quality level in Scalable Video Coding (SVC) used to produce a higher quality, higher definition video in terms of space (i.e., image resolution), time

(i.e., frame rate) or Signal-to-Noise Ratio (SNR) when combined with the playable Base Layer.

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

Manifest: The manifest file holds information about the content, i.e., describes the structure of the media, namely, the codecs used, the chunks, and the corresponding mapping within a container file system.

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

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

Peer-Peer Messages (i.e., Peer Protocol): The Peer Protocol messages enable each Peer to exchange content availability with other Peers and request other Peers for content.

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

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

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

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

Tracker: A tracker refers to a directory service which maintains the lists of PPSP peers storing chunks for a specific channel or streaming file, and answers queries from PPSP peers.

Tracker-Peer Messages (i.e., Tracker Protocol): The Tracker Protocol messages provide communication between Peers and Trackers, by which Peers provide content availability, report streaming status and

request candidate Peer lists from Trackers.

UserID: Unique identifier for the peer client user. The UserID and any required security certificates or Tokens are obtained from an offline enrollment server.

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

3. Protocol Overview

The function entities involved in the PPSP Tracker Protocol are Trackers and Peers (which may support different capabilities).

Peers are organized in (various) swarms corresponding each swarm to the group of peers sharing a content at any given time.

The Tracker is a logical entity that maintains the lists of peers storing chunks for a specific channel or streaming file, answers queries from peers and collects information on the activity of peers.

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

A P2P streaming process is summarized in Figure 1.

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

1. Peer connects to a tracker and joins a swarm.
2. Peer acquires a list of peers 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 for the content from the identified peers.

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

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

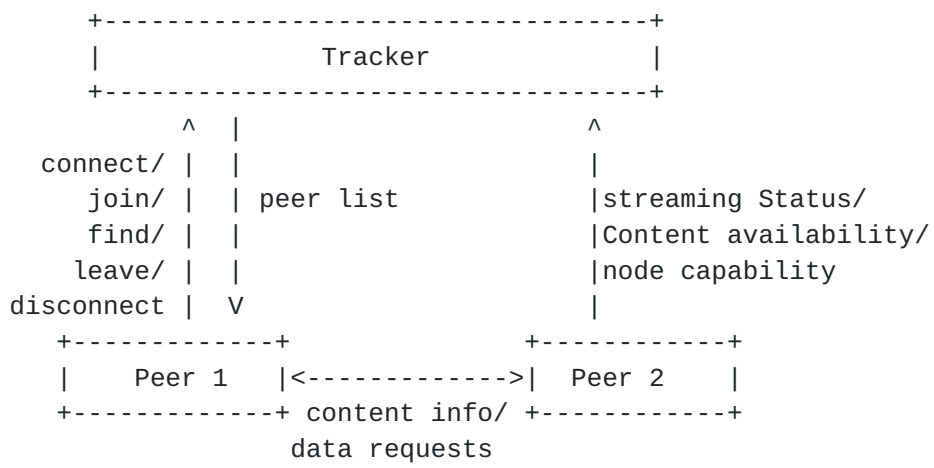

Figure 1: A PPSP streaming process

The PPSP Tracker Protocol is a request-response protocol. Requests are sent, and responses returned to these requests. A single request generates a single response (neglecting fragmentation of messages). The specific operations of the protocol at present, are (names correspond to Method strings):

1. CONNECT
2. DISCONNECT
3. JOIN
4. LEAVE
5. FIND_PEER
6. FIND_CHUNK
7. STAT_REPORT
8. KEEPALIVE

CONNECT: This method is used when a Peer connects to the system. The Service Tracker records the Peer-ID, connect-time (referenced to the absolute time), peer IP address and link status.

DISCONNECT: This method is used when the Peer intends to leave the system and no longer participate in any swarm. The Service Tracker deletes the corresponding activity records related to the peer (including its status and all content status for all swarms) updating the information on the historical profile record of that peer.

JOIN: This method is used for Peers to notify the Service Tracker that they wish to participate in a particular swarm.

LEAVE: This method is used when Peers want to indicate to the Service Tracker that they no longer wish to participate in a particular swarm.

FIND_PEER: This method allows Peers to request to the Service Tracker the Peer list for a specific content or a particular swarm.

FIND_CHUNK: This method allows Peers to request to the Service Tracker the Peer list for a specific Chunk of a particular swarm.

STAT_REPORT: This method allows the exchange of statistic and status data between Peers and Service Tracker to improve system performance. The method is initiated by the peer, periodically.

KEEPALIVE: These messages are periodically sent from Peers to the Service Tracker to notify it that the Peer is still alive, to prevent the Service Tracker from disconnecting the Peer, after some pre-configured time (assume that the Peer is no longer available).

The Response messages correspond to a number of logical responses common to the Tracker or the Peer protocols request messages. Incorrectly formatted XML request bodies are handled by the HTTP protocol itself and reported in an HTTP message.

At present, the minimum set of response messages for the PPSP Streaming Protocols is the following, re-using the error codes from HTTP conveyed in the actual HTTP message:

SUCCESSFUL (200 OK): a message has been processed properly and the desired operation has completed. If the message is a request for information, the body of the message will also include the requested information.

INVALID SYNTAX (400 Bad Request): Indicates an error in the format of the message/message body. These responses correspond to errors within the XML/PPSP protocol messages, not HTTP.

VERSION NOT SUPPORTED (400 Bad Request): Invalid version of the protocol or message bodies. These responses correspond to errors within the XML/PPSP protocol messages, not HTTP.

AUTHENTICATION REQUIRED (401 UNAUTHORISED): Authentication is required to access this information.

MESSAGE FORBIDDEN (403 FORBIDDEN): The requester is not allowed to make this request.

OBJECT NOT FOUND (404 NOT FOUND): The requested object or a swarm that is being searched for cannot be found.

INTERNAL ERROR (500 INTERNAL SERVER ERROR): The server was unable to process the request due to an internal error.

TEMPORARILY OVERLOADED (503 SERVICE UNAVAILABLE): The server is unable to process this message at this time.

3.1 Assumptions

The function entities related to PPSP protocols are the Client Media Player, the service Portal, the Service Tracker and Peers. Their complete description is not discussed in this document (as not in the scope of this specification).

The Client Media Player is the entity providing a direct interface to the end user at the client device, and includes the functions to select, request, decode and render contents. In PPSP the Client Media Player interfaces with the peer using request and response standard formats for HTTP Request and Response messages [RFC2616].

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

The Service Tracker is a logical entity that maintains the lists of PPSP active peers storing and exchanging chunks for a specific content. The tracker also stores the status of peers, to help in the selection of appropriate candidate peers for a requesting peer.

The Peer is also a logical entity embedding the P2P core engine, with a client serving side interface (a proxy HTTP server) to respond to Client Media Player requests and a network side interface (also an HTTP server) to exchange data and PPSP signaling with peers and trackers.

3.2 Bootstrapping

In order to join an existing P2P streaming service and to participate in content sharing, any peer must first locate a Tracker service, using for example, the following methods (as illustrated in Figure 2):

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

In order to be able to bootstrap, a peer must first obtain a PeerID

and any required security certificates from an enrollment service (user registration).

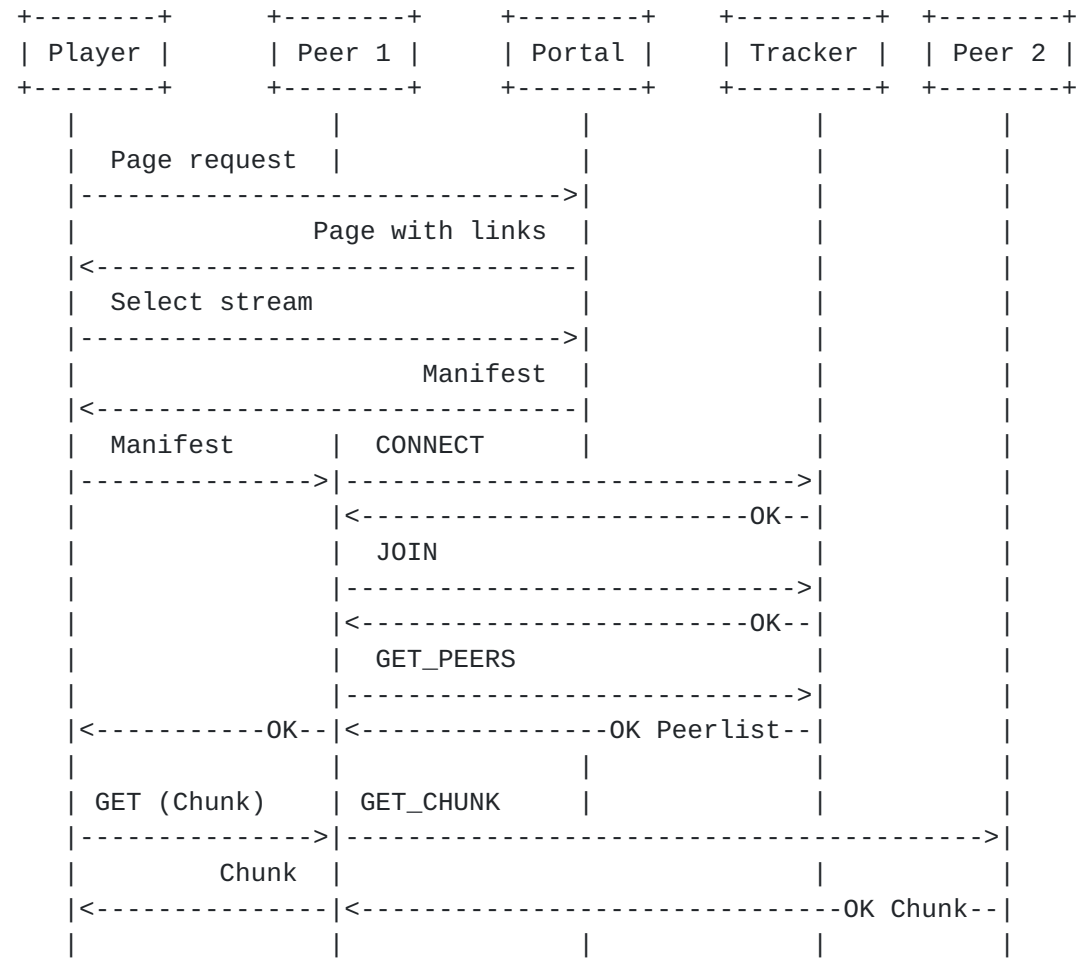

Figure 2: A typical PPSP bootstrap procedure

3.3 Streaming Modes

The streaming technique is pull-based, i.e., the client peer requests the media chunks from serving peers and is responsible for handling the buffering that is necessary for the playback processes during the download of the media chunked files, turning this technique more robust to peer churn but still with acceptable latency for a smooth play-out.

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

In VoD, different peers watch different parts of the recorded media content during a past event. In this case, each peer keeps asking other peers which media chunks are stored in which peers, and then gets the required media from certain/selected peers.

3.5 NAT Traversal

It is assumed that all trackers must be in the public Internet. This document will not describe NAT Traversal mechanisms but the proposed protocol tries to enable flexible NAT Traversal. Future versions will consider the requirements raised by NAT.

3.6 Manifest File

The Manifest file holds information about the content, i.e., describes the structure of the media, namely, the codecs used (as registered with the MP4 registration authority [MP4-Reg]), the chunks, the number of layers (in case of SVC), number of descriptions (in case of MDC) or views (in case of 3D) and the corresponding mapping within a container file system.

The Manifest is a Well-Formed XML Document, encoded as double-byte Unicode.

```
<?xml version="1.0" encoding="utf-8"?>
<StreamInfo>
   <Trackers>
      <Tracker url="http://maintracker:80" tier="1" />
      <Tracker url="http://othertracker:80" tier="2" />
   </Trackers>
   <SwarmID>123456abcd</SwarmID>
   <Clip>
      <Name>path/name</Name>
      <Description>some description</Description>
      <Video codec="SVC" fps="25">
         <VideoQuality width="720" height="1280" level="0" />
         <VideoQuality width="1920" height="1080" level="3" />
      </Video>
      <ChunkSegments type="video">
         <Segment chunks="150" levels="8" />
      </ChunkSegments>
   </Clip>
</StreamInfo>
```

Figure 3: XML Manifest for a VoD scalable video

The Manifest is a composite schema that includes, as root Element, a

StreamInfo field that encapsulates all the metadata required for the play-out of the media in groups of <Clip> fields with <ChunkIndex> elements corresponding to each component of the media (video, audio) streams, as well as other associated metadata (subtitles, text descriptions, etc.). The different <Clip> elements may include specific descriptor elements, like Video and Audio and respective attributes for each of the media types. An example of a XML manifest for a VoD scalable video content is the following (Figure 3):

The Manifest file for P2P Streaming MUST contain Tracker information prior to its upload to the Service Tracker during the publishing procedure and can be compressed with GZIP file format [RFC1952] in order to be used with HTTP compression [RFC2616] for faster transmission times and less network bandwidth usage.

The Client Media Player parses the downloaded Manifest file and, if it includes information for P2P Streaming, sends the file to the Peer via a HTTP POST and waits for the response in order to start requesting media chunks to decode and play-out. For applications where the Peer is not co-located with the Media Player in the same device (e.g., the Peer is in a Home Media Gateway), more than one Media Player MAY use the same Peer.

The Manifest file can also be used for direct download/stream (HTTP Streaming) from a specific server, and in this case, the file is not appended with the P2P Tracker information. The Client Media Player, in this case, just starts requesting media chunks from the HTTP Streaming server to decode and play-out.

The Manifest file for Live Streaming has a similar structure but describes a sliding window of a small range of <ChunkSegments> from the live program stream timeline (typically, 10 seconds of video). The sliding window is updated for every new encoded (a range of chunks defined by the attributes from="##" and to="##") of the program stream. An example of a XML manifest for a Live scalable video content is the following (Figure 4):

```
<?xml version="1.0" encoding="utf-8"?>
<StreamInfo>
   <Trackers>
      <Tracker url="http://maintracker:80" tier="1" />
      <Tracker url="http://othertracker:80" tier="2" />
   </Trackers>
   <SwarmID>654321xyz</SwarmID>
   <LiveStream>TRUE</LiveStream>
   <UpdateTime>06:56:18</UpdateTime>
   <Clip>
      <Duration>1258</Duration>
      <Name>path/name</Name>
      <Description>some description</Description>
      <Video codec="SVC" fps="25">
         <VideoQuality width="432" height="240" level="0" />
         <VideoQuality width="842" height="480" level="6" />
      </Video>
      <ChunkSegments type="video">
         <Segment from="624" to="628" levels="10" />
      </ChunkSegments>
   </Clip>
</StreamInfo>
```

Figure 4: XML Manifest for a Live scalable video

The naming convention used to identify the media chunks considers a sequential numbering for the chunks, concatenated with the identifier of the content, typically an alphanumeric string. For SVC and MDC contents, a "leveltype" and a "level_id" are added. For SVC in 3D, a "view" number is also added, immediately after the content identifier. The resulting media chunks become named as follows:

SVC/MDC: <content_id><-><leveltype><level_id><-><chunk_id>.<ext>
SVC/3D:
<content_id><-><view#>-<leveltype><level_id><-><chunk_id>.<ext>

other: <content_id><-><chunk_id>.<ext>

Where <view#> is the character V followed by the number of views, <leveltype> is a character with values L for SVC Layers and D for MDC descriptions, <level_id> is a numeric value identifying the layer (SVC) or description (MDC) and <ext> follows the common media content extension naming.

Examples of the naming convention for a content_id="A123456789" are as following:

SVC: A123456789-L0-00000.264 - chunk 0, layer 0 (base layer)
SVC: A123456789-L1-00000.264 - chunk 0, layer 1 (enhanced layer)
SVC/3D: A123456789-V2-L0-00000.264 - chunk 0, layer 0, 2 views
MDC: A123456789-D0-0000.h264 - chunk 0, description 0
Audio AAC: A123456789-00000.m4a - chunk 0
Video MPEG: A123456789-00000.mp4 - chunk 0

4 Messages syntax and processing

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

4.1 HTTP/XML Encoding

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

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

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

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

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

The Host header field in the Request message follows the standard rules for the HTTP 1.1 Host Header. Other Header fields MAY be included in the Request and Response messages if necessary.

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

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

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

The Request messages MUST include identification and authentication token of the peer user.

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

4.2 Method Fields

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

PPSP Request	XML Request Value String
CONNECT	CONNECT
DISCONNECT	DISCONNECT
JOIN	JOIN
LEAVE	LEAVE
FIND_PEER	FIND_PEER
FIND_CHUNK	FIND_CHUNK
STAT_REPORT	STAT_REPORT
KEEPALIVE	KEEPALIVE

Table 1: Valid Strings for Requests

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

Table 2: Valid Strings for Responses

4.3 Message Processing

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

Upon reception, a message is examined to ensure that it is properly formed. The receiver MUST check that the HTTP message itself is properly formed, and if not, appropriate standard HTTP errors MUST be generated. The receiver must also verify that the XML body is properly formed.

If the message is found to be incorrectly formed or the length does not match the length encoded in the header, the receiver MUST reply with an HTTP 400 response with a PPSP XML body with the Response method set to INVALID SYNTAX.

If the version number of the protocol is for a version the receiver does not supports, the receiver MUST reply with an HTTP 400 response with a PPSP XML body with the Response method set to VERSION NOT SUPPORTED.

If the receiver is unable to process the message for being in an overloaded state, the receiver SHOULD reply with an HTTP 503 response with a PPSP XML body with the Response method set to TEMPORARILY OVERLOADED.

If the receiver encounters an internal error while attempting to process the message, the receiver MUST generate an HTTP 500 response with a PPSP XML body with the Response method set to INTERNAL ERROR.

4.4 CONNECT Message

This method is used when a Peer connects to the system. The Service Tracker records the Peer-ID, connect-time, IP address and link status.

The peer MUST properly form the XML body, set the Request Method to CONNECT, set the PeerID with the identifier of the peer, randomly generate and set the TransactionID and include identification (UserID) and authentication token (AuthToken) of the peer user. The peer SHOULD also include the IP addresses of its network interfaces in the CONNECT message.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol version="#.#">
   <Method>CONNECT</Method>
   <PeerID>***</PeerID>
   <UserID>***</UserID>
   <AuthToken>***</AuthToken>
   <TransactionID>###</TransactionID>
   <PeerAddresses>
      <PeerAddress ip="##.##.##.##" port="###" />
      <PeerAddress ip="hh:hh:hh:hh:hh:hh:hh:hh" port="###" />
   </PeerAddresses>
</PPSPTrackerProtocol>
```

When receiving a well-formed CONNECT Request message, the Tracker processes the peer and user authentication information to check whether they are valid and that they can connect to the service. A Response message with a corresponding response value and method will be generated.

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

The Response message is a HTTP 200 OK with a SUCCESSFUL response in case of success. In case of error one of the response methods of Table 2 is returned. The response MUST have the same TransactionID

value as the request.

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

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

4.5 DISCONNECT message

This method is used when the Peer intends to leave the system and no longer participate in any swarm. The Service Tracker deletes the corresponding activity records related to the peer (including its status and all content status for all swarms) updating the information on the historical profile record of that peer.

In the case where the peer serves multiple clients (on the same or on different swarms), the DISCONNECT message SHALL NOT be used while there are more than one active clients. There is no need for a peer that sends a DISCONNECT to send a separate LEAVE to the Tracker for each swarm it is participating, in the case there is only one active client left that intends to leave the system and no longer participate in any swarm.

The peer MUST properly form the XML body, set the Request Method to DISCONNECT, set the PeerID with the identifier of the peer, randomly generate and set the TransactionID and include identification (UserID) and authentication token (AuthToken) of the peer user.

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

```
<?xml version="1.0" encoding="utf-8"?>
<PPSPTrackerProtocol version="#.#">
   <Method>DISCONNECT</Method>
   <PeerID>***</PeerID>
   <UserID>***</UserID>
   <AuthToken>***</AuthToken>
   <TransactionID>###</TransactionID>
</PPSPTrackerProtocol>
```

The Response message is a HTTP 200 OK with a SUCCESSFUL response in case of success. In case of error one of the response methods of Table 2 is returned. The response MUST have the same TransactionID value as the request.

Upon receiving a DISCONNECT message, the tracker MUST remove the peer from the peer list and from all swarms the peer joined, as if a LEAVE message was received for each swarm.

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

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

4.6 JOIN Message

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

The JOIN message is used when the peer does not have any chunks or has some or all the chunks of a content. The JOIN is used for both VoD or Live streaming modes.

The peer MUST properly form the XML body, set the Request Method to JOIN, set the PeerID with the identifier of the peer, set the SwarmID with the identifier of the swarm it is interested in, randomly generate and set the TransactionID and include identification (UserID) and authentication token (AuthToken) of the peer user. The peer MAY include an ExpireTime set to a non-zero value expressed in seconds, indicating that the peer expects to no longer be participating at the end of that time. The ExpireTime MUST be set to zero if the peer does not wish to set an expiration time.

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

```
<?xml version="1.0" encoding="utf-8"?>
<PPSPTrackerProtocol version="#.#">
   <Method>JOIN</Method>
   <PeerID>***</PeerID>
   <UserID>***</UserID>
   <AuthToken>***</AuthToken>
   <SwarmID>***</SwarmID>
   <ExpireTime>###</ExpireTime>
   <TransactionID>###</TransactionID>
</PPSPTrackerProtocol
```

When receiving a well-formed JOIN Request message, the Tracker processes the peer and user authentication information to check

whether they are valid and that they can JOIN to the swarm of interest. A Response message with a corresponding response value and method will be generated and the tracker enters the information into the internal peer list and swarm activity.

The Response message is a HTTP 200 OK SUCCESSFUL message in case of a successful JOIN to the requested swarm. The request may be refused by the Tracker, depending on the peer user profile, with an HTTP 403 Forbidden message response with a PPSP XML body of MESSAGE FORBIDDEN. The Tracker MUST reject the JOIN message with an HTTP 400 Bad Request Response with a PPSP XML body of INVALID SYNTAX for other conditions, with one of the response methods of Table 2.

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

An example of the SUCCESSFUL Response message structure for the JOIN request is the following:

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

4.7 LEAVE Message

This method is used when peers want to indicate to the Service Tracker that they no longer wish to participate in a particular swarm. The Service Tracker deletes the corresponding activity records related to the peer, including its status and all content status for all swarms, updating the information on the historical profile record of that peer.

The peer MUST properly form the XML body, set the Request Method to LEAVE, set the PeerID with the identifier of the peer, set the SwarmID with the identifier of the swarm the peer is not anymore interested, randomly generate and set the TransactionID and include identification (UserID) and authentication token (AuthToken) of the peer user.

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

```
<?xml version="1.0" encoding="utf-8"?>
<PPSPTrackerProtocol version="#.#">
   <Method>LEAVE</Method>
   <PeerID>***</PeerID>
   <UserID>***</UserID>
   <AuthToken>***</AuthToken>
   <SwarmID>***</SwarmID>
   <TransactionID>###</TransactionID>
</PPSPTrackerProtocol
```

When receiving a well-formed LEAVE Request message, the Tracker processes the peer and user authentication information to check whether they are valid. If the request is valid, a Response message with a corresponding response value and method will be generated and the tracker deletes the corresponding activity records for that peer.

The Response message is a HTTP 200 OK SUCCESSFUL message in case of a successful LEAVE to the requested swarm. The Tracker MUST reject the LEAVE message with an HTTP 400 Bad Request Response with a PPSP XML body of INVALID SYNTAX if this condition has occurred, or for other conditions, with one of the response methods of Table 2.

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

An example of the SUCCESSFUL Response message structure for the LEAVE request is the following:

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

4.8 FIND_PEER Message

This method allows peers to request to the Service Tracker the Peer list for a specific swarm.

The peer MUST properly form the XML body, set the Request Method to FIND_PEER, set the PeerID with the identifier of the peer, set the SwarmID with the identifier of the swarm the peer is interested, optionally include information about the content, like Clip Name and Type, randomly generate and set the TransactionID and include identification (UserID) and authentication token (AuthToken) of the peer user.

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

```
<?xml version="1.0" encoding="utf-8"?>
<PPSPTrackerProtocol version="#.#">
   <Method>FIND_PEER</Method>
   <PeerID>***</PeerID>
   <UserID>***</UserID>
   <AuthToken>***</AuthToken>
   <SwarmID>***</SwarmID>
   <Clip>
      <Name>***</Name>
      <Type>(video, audio, etc.)</Type>
   </Clip>
   <TransactionID>###</TransactionID>
</PPSPTrackerProtocol>
```

When receiving a well-formed FIND_PEER Request message, the Tracker processes the peer and user authentication information to check whether they are valid.

If the request is valid, a Response message with a corresponding response value and method will be generated and the tracker will search the internal data store to select and return an appropriate list of peers, with their identifiers and IP Addresses, that will be able to provide the desired content.

The Response message is an HTTP 200 OK SUCCESSFUL message in case of success. The Tracker MUST reject the FIND_PEER message with an HTTP 400 Bad Request Response with a PPSP XML body of INVALID SYNTAX if this condition has occurred, or for other conditions, with one of the response methods of Table 2. The response MUST have the same TransactionID value as the request.

An example of the SUCCESSFUL Response message structure for the FIND_PEER request is the following:

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

The <PeerInfo> field record in the XML body can be instantiated multiple times in the <PeerInfoList> element, one instance per peer.

The response message also carries information associated with network topology information for each peer listed.

4.9 FIND_CHUNK Message

This method allows Peers to request to the Service Tracker the Peer list for a specific Chunk of a particular swarm.

The peer MUST properly form the XML body, set the Request Method to FIND_CHUNK, set the PeerID with the identifier of the peer, set the SwarmID with the identifier of the swarm the peer is interested, include Clip Name, Type and the Type information of the content, randomly generate and set the TransactionID and include identification (UserID) and authentication token (AuthToken) of the peer user.

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

```
<?xml version="1.0" encoding="utf-8"?>
<PPSPTrackerProtocol version="#.#">
   <Method>FIND_CHUNK</Method>
   <PeerID>***</PeerID>
   <UserID>***</UserID>
   <AuthToken>***</AuthToken>
   <SwarmID>***</SwarmID>
   <Clip>
      <Name>***</Name>
      <Type>(video, audio, etc.)</Type>
      <ChunkSegmentIndex>###</ChunkSegmentIndex>
   </Clip>
   <TransactionID>###</TransactionID>
</PPSPTrackerProtocol>
```

When receiving a well-formed FIND_CHUNK Request message, the Tracker processes the peer and user authentication information to check whether they are valid.

If the request is valid, a Response message with a corresponding response value and method will be generated and the tracker will search the internal data store to select and return an appropriate list of peers, with their identifiers and IP Addresses, that will be able to provide the desired content chunk.

The Response message is an HTTP 200 OK SUCCESSFUL message in case of success. The Tracker MUST reject the FIND_CHUNK message with an HTTP 400 Bad Request Response with a PPSP XML body of INVALID SYNTAX if this condition has occurred, or for other conditions, with one of the response methods of Table 2. If the data is not found an HTTP 404 Not Found with a PPSP XML Response method set to OBJECT NOT FOUND. The response MUST have the same TransactionID value as the request.

An example of the SUCCESSFUL Response message structure for the FIND_CHUNK request is the following:

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

The <PeerInfo> field record in the XML body can be instantiated multiple times in the <PeerInfoList> element, one instance per peer.

The response message also carries information associated with network topology information for each peer listed.

4.10 STAT_REPORT Message

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

The peer MUST properly form the XML body, set the Request Method to STAT_REPORT, set the PeerID with the identifier of the peer, randomly generate and set the TransactionID, include identification (UserID) and authentication token (AuthToken) of the peer user. The report consists of a <Stats> element in the XML body containing multiple <Stat> fields.

The <Stat> field record in the XML body can be instantiated multiple

times in the <Stats> element, one instance per "property" to report. If the property being reported is associated with a swarm then the peer MUST set the SwarmID element with the identifier of the swarm and include other relevant information like Clip Name, Type and information of the content.

The properties listed in Table 3 correspond to the REQUIRED minimum set of information for the STAT_REPORT.

Property Value	Definitions/Description
ChunkMap	a base64 encoded bitmap of chunks available
StreamStats	information on network streaming:
:UploadedBytes	total bytes sent to other peers in the swarm
:DownloadedBytes	total bytes received from peers in the swarm
:AvailBandwidth	total bytes received from peers in the swarm

Table 3: Minimum set of Property Types for STAT_REPORT messages

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

```
<?xml version="1.0" encoding="utf-8"?>
<PPSPTrackerProtocol version="#.#">
   <Method>STAT_REPORT</Method>
   <PeerID>***</PeerID>
   <UserID>***</UserID>
   <AuthToken>***</AuthToken>
   <TransactionID>###</TransactionID>
   <Stats>
  <Stat property="ChunkMap">
     <report:ChunkMapReport>
        <SwarmID>***</SwarmID>
        <Clip>
           <Name>***</Name>
           <ChunkSegments type="video">
               <Segment from="###" to="###"
                        bitmapSize="###">
                 ... encoded string ...
              </Segment>
           </ChunkSegments>
        </Clip>
     </report:ChunkMapReport>
  </Stat>
  <Stat property="StreamStats">
     <report:StramStatsReport>
        <SwarmID>***</SwarmID>
        <UploadedBytes>###</UploadedBytes>
        <DownloadedBytes>###</DownloadedBytes>
        <AvailBandwidth>###</AvailBandwidth>
     </report:StramStatsReport>
  </Stat>
   </Stats>
</PPSPTrackerProtocol>
```

When receiving a well-formed STAT_REPORT message, the Tracker processes the peer and user authentication information to check whether they are valid.

If the request is valid, a Response message with a corresponding response value and method will be generated and the tracker MAY process the received information for future use.

The Response message is an HTTP 200 OK SUCCESSFUL message in case of success. The Tracker MUST reject the STAT_REPORT message with an HTTP 400 Bad Request Response with a PPSP XML body of INVALID SYNTAX if this condition has occurred, or for other conditions, with one of the

response methods of Table 2. The response MUST have the same TransactionID value as the request.

An example of the SUCCESSFUL Response message structure for the STAT_REPORT is the following:

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

4.11 KEEPALIVE Message

These messages are periodically sent from peers to the Service Tracker to notify it that the peer is still alive (although not actively exchanging data with other peers), to prevent the Service Tracker from disconnecting the Peer (after some pre-configured time) assuming that the peer was no longer available).

The peer MUST properly form the XML body, set the Request Method to KEEPALIVE, set the PeerID with the identifier of the peer, randomly generate and set the TransactionID, include identification (UserID) and authentication token (AuthToken) of the peer user.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<PPSPTrackerProtocol version="#.#" >
   <Method>KEEPALIVE</Method>
   <PeerID>***</PeerID>
   <UserID>***</UserID>
   <AuthToken>***</AuthToken>
   <TransactionID>###</TransactionID>
</PPSPTrackerProtocol>
```

When receiving a well-formed KEEPALIVE message, the Tracker processes the peer and user authentication information to check whether they are valid.

If the request is valid, a Response message with a corresponding response value and method will be generated and the tracker SHOULD update an internal timer to indicate that the tracker has heard from the peer.

The Response message is an HTTP 200 OK SUCCESSFUL message in case of success. The Tracker MUST reject the KEEPALIVE message with an HTTP

400 Bad Request Response with a PPSP XML body of INVALID SYNTAX if this condition has occurred, or for other conditions, with one of the response methods of Table 2. The response MUST have the same TransactionID value as the request.

An example of the SUCCESSFUL Response message structure for the KEEPALIVE is the following:

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

5 Security Considerations

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

To protect the PPSP signaling from attackers pretending to be valid peers (or peers other than themselves) all messages received in the tracker are required to be received from authorized peers and MUST be digitally signed with the peer user authentication token, providing therefore end-to-end security for communications.

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

6 IANA Considerations

There are presently no IANA considerations with this document.

7 Acknowledgments

The authors would like to thank all the people participating in the EU FP7 project SARACEN for contributions and feedback to this document.

SARACEN (Socially Aware, collaboRative, scAlable Coding mEdia distributioN), is a research initiative funded partially by the Information Society and Media Directorate General of the European Commission under the Seventh Framework programme (contract no. ICT-248474).

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

The Response method names and some text describing them, was borrowed from [I-D.gu-ppsp-tracker-protocol].

8 References

8.1 Normative References

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

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

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

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

8.2 Informative References

[I-D.ietf-ppsp-reqs] Zong, N., Zhang, Y., Avila, V., Williams, C., and L. Xiao, "P2P Streaming Protocol (PPSP) Requirements", draft-ietf-ppsp-reqs-02 (work in progress), February 2011.

[I-D.ietf-ppsp-problem-statement] Zhang, Y., Zong, N., Camarillo, G., Seng, J., and Y. Yang, "Problem Statement of P2P Streaming Protocol (PPSP)", draft-ietf-ppsp-problem-statement-01 (work in progress), January 2011.

[I-D.ietf-ppsp-survey] Yingjie, G., Zong, N., Zhang, H., Zhang, Y., Lei, J., Camarillo, G., and L. Yong, "Survey of P2P Streaming Applications", draft-gu-ppsp-survey-02 (work in progress), March 2011.

[I-D.cruz-ppsp-http-peer-protocol] Cruz, R., Nunes, M. and Taveira, J., "HTTP-based PPSP Peer Protocol", draft-cruz-ppsp-http-peer-protocol-00 (work in progress), May 2011.

[I-D.gu-ppsp-tracker-protocol] Yingjie, G., Bryan, D., Zhang, Y., and H. liao, "PPSP Tracker Protocol", draft-gu-ppsp-tracker-protocol-04 (work in progress), May 2011.

[MP4-Reg] MP4REG, The MPEG-4 Registration Authority, URL: <http://www.mp4ra.org>.

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

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

Joao Pedro Taveira Pinto Silva
IST/INOV
Phone: +351.966913777
Email: joao.taveira@inov.pt