PPSP
Internet Draft
Intended status: Informational
Expires: April 26, 2012

L.Xiao
Nokia Siemens Networks
D.Bryan
Cogent Force, LLC/Huawei
Y.Gu
Huawei
X.Tai
China Mobile/BUPT

October 25, 2011

A PPSP Tracker Usage for Reload
draft-xiao-ppsp-reload-distributed-tracker-03

Status of this Memo

Copyright Notice

Abstract

This document defines PPSP tracker usages for REsource LOcation And Discovery (RELOAD). Although PPSP assumes a centralized tracker from peer's point of view, the logical centralized tracker could be realized by a cluster of geographically distributed trackers. In this draft, we design distributed trackers system, which are organized by RELOAD. It provides lookup service for file/channel indexes and Peer Status among the distributed trackers.

Table of Contents

1. Introduction

PPSP assumes that a centralized 'tracker' is used to communicate with the PPSP Peers for content registration and location. The content index is stored in the tracker with location information that which peers have the content.

However, the logically centralized 'tracker' could be also realized by a cluster of geographically distributed trackers or deployed in multiple servers in a data center, which can increase the content availability, the service robustness and the network scalability or reliability. The management and locating of index information are totally internal behaviors of the tracker cluster, which is invisible

to PPSP Peers. The signaling between PPSP Peers and trackers is still the simple request/reply mechanism defined in PPSP tracker protocols. Therefore, the Tracker Nodes themselves construct and maintain a P2P overlay as shown in Figure 1. This document is to define the behaviors of the tracker overlay as a usage for RELOAD. Because all protocols this draft applies to - the PPSP tracker protocol, PPSP peer protocol and RELOAD - are still changing, this draft will certainly change in the future, and is very much a work in progress. The authors encourage list discussion of the draft and any suggestions are welcome.

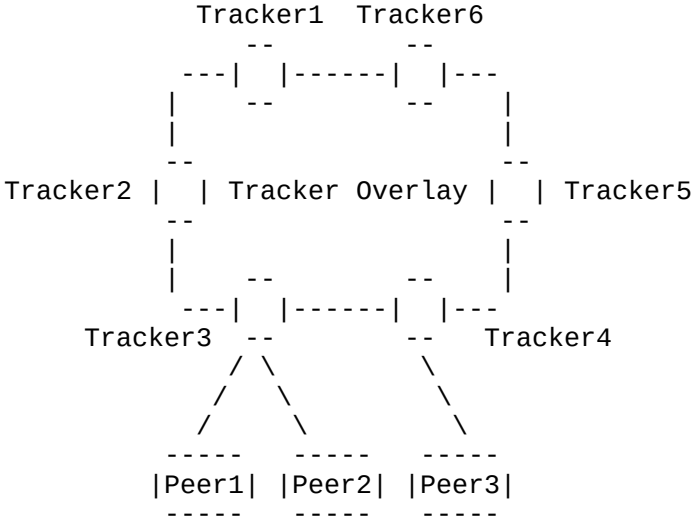

Figure 1 PPSP Peers connect with Tracker Overlay by Tracker Protocol

The document tries to handle the data maintenance for two kinds of information: content index and PPSP peer status, among a cluster of distributed trackers. Four basic functions in tracker overlay are defined as follows:

o Content/ channel index information registration: PPSP Peers registrar/update their contents/channels to a Connection Tracker.(How to find the initial tracker locally is out of scope.) This tracker takes the advantage of the RELOAD data storage functionality to store the index information to tracker nodes in the tracker overly accordingly. At the same time, the local Connection Tracker keeps a copy of local peer's content information for traffic localization.

o Look up a content/channel index: Once a PPSP Peer search for certain content/channel, it makes the request to a local Connection Tracker as defined in PPSP tracker protocol. If the swarm cannot be found or there is not enough peer records for such swarm in the Connection Tracker locally, the tracker will further locate the required index information in the tracker overlay on behalf of the requesting PPSP Peer. Once the full Peer List is fetched, the PPSP Peer will set up communications with the PPSP Peers in the Peer List as defined in PPSP Peer protocol;

o PPSP peer status registration: PPSP Peers registrar/update their status in the tracker overlay. All PPSP peers should firstly register their status to the local Connection Tracker. In order to enable this information being aware globally, the Connection Tracker should then store the position of the PPSP peer's status in the tracker overlay according to RELOAD scheme. The following peer status updates are only sent to the local Connection Tracker, the RELOAD based tracker overlay here only offers a way for remote nodes to find the location of requested peer status.

o Look up status of a certain peer: the tracker overlay can look up the status of a certain PPSP Peer. If the peer status cannot be found in the local Connection Tracker (that means it's not a local peer), the local tracker then searches the Status Position Tracker for the requested peer in the tracker overlay by RELOAD, which gives a route to access the status of the remote peer.

2. Terminology and Conventions

This document makes extensive use of the terminology and definitions from the RELOAD Base Protocol [I-D.ietf-p2psip-base], PPSP Requirements and Problem Statements [I-D.ietf-ppsp-problem-statement][I-D.ietf-ppsp-reqs] and the Gu PPSP Tracker Protocol proposal [I-D.gu-ppsp-tracker-protocol].

This document defines the following additional terminology:

PPSP Peer: The peer in PPSP protocol for content sharing and distribution among swarms.

Tracker Node: The RELOAD Node with PPSP tracker usage. Each Tracker Node takes the responsibility to store and maintain certain content/channel index.

Tracker Overlay: A RELOAD overlay constructed by Tracker Nodes. This Overlay is logically separated with overlay formed by PPSP Peers.

Connection Tracker: The Tracker Node to which the PPSP Peer will connect when it wants to join the PPSP system.

Swarm Tracker: The Tracker Node who is responsible for the swarm in the overlay, and stores the content information (e.g. Peerlist) of the swarm.

Status Position Tracker: A Tracker Node which is responsible to store the Position of certain peers' status of a particular list of Peers.

3. Content Information Registration and Update

To fulfill the functions of content information registration and update mentioned in Section 1, Tracker Node must maintain such resources related to peers;

Content Registration: Information about the content which belongs to a specific swarm. It can be stored in a data structure denoted as ContentRegistration, which primarily includes an identification of the swarm, a name of the content, and a Peer List.

3.1. Data structure of ContentRegistration

Structure The data structure of ContentRegistration uses the RELOAD dictionary kind whereas the DictionaryKey value is the Swarm ID of the content required. The data structure of type ContentRegistration is shown as follows:

```
struct{

       Uint32 index;

       ChunkID chunk_id;
```

```
}ArrayChunkListData;

struct{
       PeerID peer_id;
       ArrayChunkListData  chunklist_data;
}PeerListData;

struct{
       uint16 length;
       PeerListData peerlist_data;
}PeerList;

struct {
       uint16 length;
       opaque content_name<0..2^16-1>;
       PeerList peerlist <0..2^16-1>;
} ContentRegistration;
```

The content of the PeerList structure are as follows:

length

the length of the data structure

content_name

the name of the content

peerlist

the content of Peer List

3.2. Message flows

When a PPSP Peer wishes to share its contents to others, it will inform Tracker Overlay with the swarm information of the contents, then Swarm Tracker need to add this PPSP Peer into the corresponding Peer List to the swarm, or create a new swarm when there is no record of the swarm. A local record of the swarm may also be set up at the Connection tracker. Correspondingly, When a PPSP Peer deletes some old contents locally, it will inform Tracker Overlay that it would like to leave from a particular swarm, then both Connection Tracker and Swarm Tracker need to delete this PPSP Peer from the corresponding Peer List which is defined in the requirement of PPSP [I-D.ietf-ppsp-reqs].

An example is given as the figure has shown below:

1. PPSP Peer wants to join into a swarm to share the content, first it will send a PPSP message "Join" with a Swarm-ID to TrackerA, which is a connection tracker of the Tracker Overlay for PPSP Peer connects to;

2. TrackerA first handles the registration locally, then finds the Swarm Tracker by mapping the swarm ID to node ID of the Swarm Tracker, to forward the request. So TrackerA sends a RELOAD message "StoreReq" to TrackerB who is the Swarm Tracker for the content swarm;

3. When Swarm Tracker (TrackerB) receives the request (or if TrackerA is responsible for the Peer List of the swarm, TrackerB=TrackerA), it searches locally the Peer List of the swarm whose ID is the Swarm-ID, then add the Node-ID of the PPSP Peer into the Peer List or delete it from that, and send the result of the operation (e.g. successful or failed) in a RELOAD message "StoreReqAns" to TrackerA through Tracker Overlay;

4. Finally, TrackerA analyses the received message, and responds to the requesting Peer by a corresponding PPSP message: "Successful (OK)" or some error messages.

Note: When PPSP Peer is the first node of the swarm, which means it is the first one who stores this kind of content in the network, TrackerB doesn’t have records of the new swarm, TrackerB will create a new ContentRegistration for the swarm locally, and put the identification of PPSP Peer into Peer List of this new

ContentRegistration, then send the result of the operation (e.g. successful or failed) in a RELOAD message "StoreReqAns" to TrackerA through Tracker Overlay.

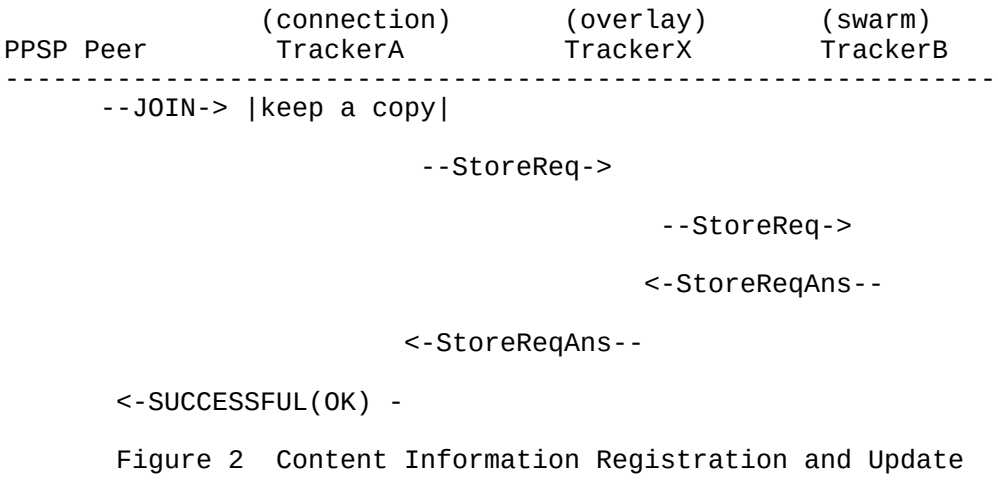

Figure 2 Content Information Registration and Update

If PPSP Peer wishes to update content information, for example, list of chunks it has, it sends a PPSP message "JOIN_CHUNK" to TrackerA. TrackerA makes update in its local table, and then sends the corresponding RELOAD message to TrackerB to update the detailed chunk-IDs in the Swarm according to the request message.

4. Lookup Content Index (a Swarm)

When a PPSP Peer wants to use some streaming service, which means it wants to download some interested contents from the system, it firstly needs to get related Peer List from Tracker Overlay. As the figure has shown below:

1) PPSP Peer wants to watch a video belonging to a swarm with a Swarm-ID, firstly it sends a PPSP message "Find" with the Swarm-ID to Connection TrackerA;

2) If TrackerA has enough local peer record for swarm, it can reply the request directly. Or it maps the Swarm-ID into a Node-ID to identify the Swarm Tracker, TrackerB, which stores the Peer List of the requested swarm. It then sends a RELOAD message "FetchReq" to TrackerB;

3) When Swarm TrackerB receives the request (or if TrackerA is responsible for the Peer List of the swarm, TrackerbB=TrackerA), it searches the Peer List of the swarm locally, then send the Peer List which is organized by the data structure of PeerList in a RELOAD message "FetchReqAns" to TrackerA through Tracker Overlay;

4) Finally, TrackerA analyses the received PeerList structure, and reconstructs it into a PPSP message "Successful(OK)", then forwards it to the PPSP Peer.

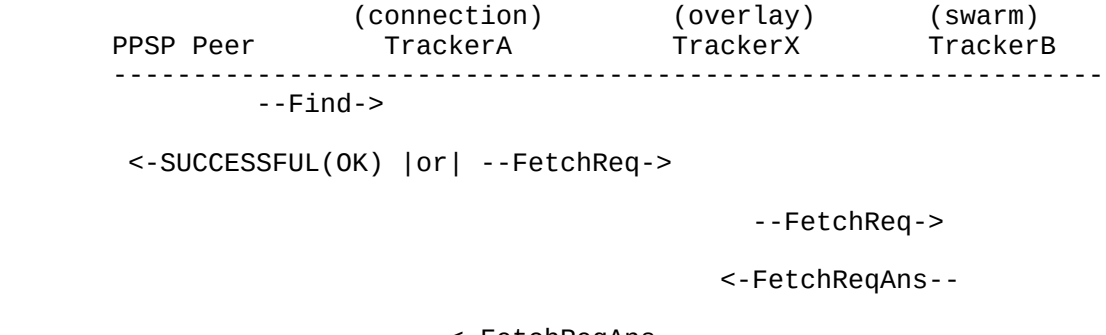

Figure 3 Content Information Lookup

5. Peer Status Registration, Update and Lookup

To fulfill the functions of peer status registration, update and lookup mentioned above, Tracker Node must maintain such resource related to peers:

Information about status of peers: the local Connection Tracker takes the responsibility to maintain the PPSP Peer status locally, including online time, link status, node capability and other streaming parameters, etc. It can be stored in a data structure denoted as PeerStatus.

Position of PPSP peer status: each PPSP Peer can be mapped to a Status Position Tracker in the tracker overlay. The status Position Tracker takes responsibility to only record the route (i.e., the address of the local Connection Tracker of the Peer) to access the PPSP Peer status.

5.1. Data Structure of PeerStatusIndex

The data structure of PeerStatusIndex uses the RELOAD dictionary kind whereas the DictionaryKey value is the Peer ID. The data structure of type PeerStatusIndex is shown as follows:

```
struct{
       TrackerID Connection_Tracker_ID;
}PeerStatusIndex;
```

The content of the PeerStatusIndex structure are as follows:

trackerID the ID of the Peer’s Connection Tracker;

6. Kind Definition

6.1. CONTENT-REGISTRATION Kind Definition

This section defines the CONTENT-REGISTRATION kind.

- Name: CONTENT-REGISTRATION

- Kind IDs: The Resource Name for the CONTENT-REGISTRATION Kind-ID is Swarm Name. The data stored is a CONTENT-REGISTRATION, which contains a identification of the swarm, a name of the content, and a list of PPSP Peer-IDs with or not a list of chunk-IDs for each PPSP Peer to show which chunks the PPSP Peer has.

- Data Model: The data model for the CONTENT-REGISTRATION Kind-ID is dictionary. The dictionary key is the Swarm-ID of the peer action as focus.

- Access Control: USER-NODE-MATCH.

6.2. PEER-STATUS Kind Definition

This section defines the PEER-STATUS kind.

- Name: PEER-STATUS

o Kind IDs: The Resource Name for the PEER-STATUS Kind-ID is Peer Status. The data stored is a PEER-STATUS, which contains a identification of the peer and a identification of the peer's connection tracker.

o Data Model: The data model for the PEER-STATUS Kind-ID is dictionary. The dictionary key is the Peer-ID.

o Access Control: USER-NODE-MATCH.

7. Security Considerations

This document does not currently introduce security considerations.

8. IANA Considerations

This document does not specify IANA considerations.

9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

References

9.1. Normative References

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

[2] [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-03 (work in progress), July 2011.

[3] [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-03 (work in progress), August 2011.

[4] [I-D.ietf-p2psip-base] Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-18 (work in progress), August 2011.

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

9.2. Informative References

Author's Addresses

Lin Xiao
Nokia Siemens Networks
No.14 Jiuxianqiao Road
Beijing, 100016
P.R.China

Phone: +86-13810361287
Email: lin.xiao@nsn.com

David A. Bryan
Cogent Force, LLC / Huawei

Email: dbryan@ethernot.org

Yingjie Gu
Huawei
No. 101 Software Avenue
Nanjing, Jiangsu Province 210012
P.R.China

Phone: +86-25-56624760
Email: guyingjie@huawei.com

Xuan Tai
China Mobile/BUPT

Phone: +86-13581762082
Email: taixuanyueshi@gmail.com