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/ietf/1id-abstracts.txt.

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

This Internet-Draft will expire on April 24, 2010.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (http://trustee.ietf.org/license-info).
Abstract

This document defines a P2P IPTV Usage for Resource Location And Discovery (RELOAD). The IPTV Usage provides the functionality of IPTV servers in a fully-distributed system using P2PSIP RELOAD. The IPTV Usage provides the metadata storage, channel peer list storage, and channel peer group storage using the P2PSIP overlay. This documents defines three kinds about these usages.

Table of Contents

1. Introduction ................................................ 3
2. Terminology .................................................. 3
3. P2P IPTV Architectures ...................................... 4
   3.1. Tracker based architecture ............................ 4
   3.2. Server-controlled architecture ........................ 5
   3.3. Full distributed architecture .......................... 6
4. Metadata storage ............................................ 7
   4.1. Overview ............................................. 7
   4.2. Registering MetaKeyword ................................ 8
   4.3. Looking up MetaKeyword ............................... 9
   4.4. IPTV-META-KEYWORD Kind Definitions .................. 10
   4.5. Registering MetaFullDesc ............................. 11
   4.6. Looking up MetaFullDesc .............................. 12
   4.7. IPTV-META-FULLDESC Kind Definitions ................ 12
5. Channel peer list storage .................................. 13
   5.1. Overview ............................................. 13
   5.2. Registering channel peer list ........................ 14
   5.3. Looking up channel peer list .......................... 15
   5.4. IPTV-CHAN-PEER-LIST Kind Definition .................. 15
6. Active storage & Channel peer group storage ................. 15
   6.1. Active Storage & Overview ............................ 15
   6.2. Registering channel peer group ....................... 16
   6.3. Looking up channel peer group ........................ 17
   6.4. IPTV-CHAN-PEER-GROUP Kind Definition ................. 18
7. Security Considerations .................................... 18
8. IANA Considerations ....................................... 18
9. Acknowledgments ............................................ 18
10. References ................................................ 19
    10.1. Normative References ................................ 19
    10.2. Informative References ............................. 19
1. Introduction

RELOAD provides distributed storage and distributed service [I-D.ietf-p2psip-base]. RELOAD does not only support SIP based VoIP service [I-D.ietf-p2psip-sip] but also other services. This IPTV Usage of RELOAD allows a node to store or fetch various P2P IPTV related information within the overlay.

Nowadays, Streaming service and User Create Contents are getting more popular. The number of streaming channel is growing up very quickly. They require huge storage and high bandwidth. It seems also impossible that central servers cover all streaming and clients. For economic reasons, P2P streaming must be used. There are already some P2P based streaming services. Some standardization groups have been starting to make a standard for P2P streaming.

This document provides the method to use the RELOAD overlay as a distributed P2P IPTV server. The RELOAD overlay can store meta data about IPTV channel. And, The RELOAD overlay can store list of the peers which are attending P2P IPTV streaming. Although this document focuses on Live P2P IPTV service, one can use this method for VoD or another streaming service.

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].

The concepts used in this document are compatible with "Concepts and Terminology for Peer to Peer SIP" [I-D.ietf-p2psip-concepts] and the P2PSIP base protocol RELOAD [I-D.ietf-p2psip-base].

Channel : a media stream from a content provider to viewers.

Channel ID : a globally unique ID which identifies a certain channel. It may be a human readable ID.

Hashed Channel ID : a result of hash function of Channel ID. To store channel information to the overlay, channel ID should be hashed.

Content Provider : a source entity which produces media for channel to others.
Viewer: an entity which receives media from other peers. A Viewer may/may not upload media stream to others. Client is an entity which does not upload media stream while it is downloading media.

Metadata: All description and information about a certain channel. It includes title, category, language, credit, and so on. It may be used to find a certain channel.

User: a person handles a certain agent.

Agent: an entity helps a User. A software which has user interface and communication protocols.

3. P2P IPTV Architectures

3.1. Tracker based architecture

Tracker based P2P IPTV architecture is a simple and basic architecture. The following figure shows this type of architecture.

```
+--------+
| Server |
+--------+

^                        +--------+
|                        | Tracker | (peer list)
|                        +--------+
| all peer list          +--------+

v  v                        +--------+
| Peer J |<---> Peer A |
|--------+----+--------+
|        +----+--------+
| Peer B |<---> Peer C |
|--------+----+--------+
```

There is the server which stores channel information. This server gives tracker information of a certain channel to new joining peer. Tracker information includes tracker’s IP address. The new joining peer (Peer J in the figure) connects to the tracker and asks a peer list. Tracker stores all peer list. This peer list includes IP address and port number of all peers which attend to the same channel.
The Content Provider of this channel is also included in this peer list. After the joining peer (Peer J) receives all peer list from the tracker, this peer checks which peer is the best peer in the peer list (Peer A, B, C in the figure). There are some methods to find the best peer among peers. For example, measuring RTT to each candidate peer is a simple method. Using (geographical) location information from IP address or the network coordinate [GNF] is one of useful methods. These methods consider peer’s capabilities such as upload bandwidth. After Peer J finds the best peer (Peer A, for example) in the list, Peer J connects Peer A and asks Peer A to send media. After Peer J receives media stream from peer A, Peer J registers itself to the tracker. Peer J is added into the peer list in the tracker. The main feature of this architecture is that a tracker stores only a peer list for the channel. This peer list includes only coarse information about peer such as IP address and port.

Instead that a tracker provides all peer list for a new joining peer, a tracker can give initial random peer list. A peer can use gossiping which selects other random peers and communicate with each other to get more peers.

3.2. Server-controlled architecture

In contrast to the tracker based architecture, server-controlled P2P IPTV architecture uses a control server. This server stores all information about the channel. This information includes a peer list and virtual link information of the overlay. The following figure shows this architecture.

```
+----------------+
| Control Server |
+----------------+
| best candidate peer |
+----------------+
| Peer A |
+--------+
| Peer B |
```

A new joining peer (Peer J) connects the control server. Peer J finds the channel from the server and asks the server for the peer list. The control server calculates virtual distances from Peer J to the existing peers in overlay topology and finds the better candidate peers for Peer J in the peer list. The virtual distance is a logical distance reflecting bandwidth and delay between two peers. In
contrast to the tracker based architecture, the control server does not give all peer list to the joining peer (Peer J). Instead of all peer list, the control server gives small number of peer list which are the better candidates (Peer A,B, in the figure) for the joining peer. Peer J selects the best peer among the candidates(Peer A,B). If Peer J selects Peer A as the best peer, Peer J notifies the control server that Peer J selects to Peer A. The control server makes Peer A send the media to Peer J. The control server knows everything about this channel. i.e. The server knows who sends to who and who receives from who.

3.3. Full distributed architecture

Full distributed architecture replaces the control server with DHT(distributed hash table). P2PSIP is one of DHT implementation [I-D.ietf-p2psip-base]. The following figure shows this architecture.

```
+----------------+ (meta data)  
| DHT (P2PSIP)   | (peer list)  
+----------------+              
          +--------+  
      v            +-------------> | Peer A |  
          +--------+  
      Peer J --+  
          +--------+  
```

DHT stores meta data and peer list of channels in distributed manner. A streaming channel has many meta data which describes this channel. A user can find a certain channel by meta data searching. In the near future, thousands of IPTV steam channels will be expected to be emerging. Not only commercial content provider but also normal users, devices, sensors, and CCTVs will be contributors for lots of IPTV media stream channels. We need a realtime searching mechanism for global channels. This distributed meta data storage is one of realtime channel searching mechanism. Using P2PSIP event notification extension[I-D.ietf-p2psip-event], we can wait for a specific content channel.

DHT stores peer list for each channel. Similar to server controlled architecture, DHT provides the peer list and additional information to the joining peer. The joining peer can use additional information to select the best peer to connect.
For example scenario, first, Peer J sends keywords into DHT to find a channel. DHT gives channel lists which has the matched metadata with keywords. Peer J selects one channel in the channel list. Peer J sends channel ID to DHT. DHT gives the peer list to Peer J. Peer J selects the best peer in the list. Peer J asks Peer A to send media. After that, Peer J registers its information into DHT as a peer in the channel.

In mesh-full architectures or VoD services, DHT may stores chunk information[I-D.chunk-discovery]. DHT helps to find who have a certain chunk data in whole file or stream. However, chunk discovery and registration process may cause overlay overload. To avoid this problem, peers MAY exchange chunk map with each others.

4. Metadata storage

4.1. Overview

In Metadata storage, simply speaking, the input to DHT is keyword and output is channel ID. User sends FetchReq message to DHT which includes a hashed keyword and description type(title, actor, or synopsis). The response, FetchAns, includes channel-ID and short description or title for a quick view.

Contents Provider registers Metadata for each meaningful keyword to P2PSIP DHT using StoreReq message. The resource name is keyword and description type. The resource-ID is hashed value of the resource name. StoreReq includes channel-ID and short description (for example, title) for this channel. The responsible peer for this resource ID should store these metadata.

User, especially Viewer, will search for a channel with keyword. Viewer should send FetchReq to DHT. As a results, Viewer can get a channel ID and short description.

The Metadata storage in this draft supports to store a full description into DHT. The resource name of full description is Channel-ID.

* Comment: The issue about large P2PSIP message has been discussed in IETF P2PSIP working group. Even if WG decided to use small size message only, this draft asks RELOAD to support as large message as about 1M bytes.
To get notification of a specific channel registration, User MAY use P2PSIP event notification extension [I-D.ietf-p2psip-event]. This mechanism may be used for dynamic metadata where a meta data register changes metadata frequently[MyEdir].

If a Content Provider want to make own IPTV channel, it should make a Channel ID, and describes this channel. And, it should register this channel information into the DHT.

This document does not define the way to make Channel ID. But, Channel ID MUST be globally unique.

In ordinary centralized IPTV, a Content Provider registers its channel information to an IPTV server. However, in this draft, a Contents Provider MAY register its channel information to the DHT. To register its channel information, a RELOAD peer stores IptvMetaKeyword structure or IptvMetaFullDesc structure. This draft explains how to store meta data into a P2PSIP RELOAD overlay.

4.2. Registering MetaKeyword

All Peers should work together to store metadata. A Content Provider can request to store keywords and Channel-ID into the overlay. This request is StoreReq that includes IptvMetaKeyword structure. Kind-ID for this storage is IPTV-META-KEYWORD. Formal definition of IPTV-META-KEYWORD is described in section 4.4 in this draft.

As a simple example, if the title of the channel is "Bob’s live talk show", the Content Provider MAY make 4 words (Bob, live, talk, show) as title metadata keyword. It hashes each word and stores each title metadata keywords into the overlay. Each metadata will be stored in distributed other peers.

The contents of the IptvMetaKeyword are as follows:
enum {
    iptv_metatype_title(1),
    iptv_metatype_synopsys(2),
    iptv_metatype_genre(3),
    iptv_metatype_language(4),
    iptv_metatype_cast(5),
    iptv_metatype_group(6),
    iptv_metatype_credit(7),
    iptv_metatype_review(8),
    iptv_metatype_crid(9),
} IptvMetaType;

struct {
    IptvMetaType type;
    opaque keyword<0..2^16-1>;
    opaque short_description<0..2^16-1>;
    opaque channel_id<0..2^16-1>;
} IptvMetaKeyword;

The contents of the IptvMetaKeyword PDU are:

    type
    the description type of the metadata keyword. a number.

    keyword
    partial keyword.

    short_description
    short description about the content. This will be shown as a
    result of the keyword searching. Normally, this is a title of the
    content.

    channel_id
    the unhashed channel id

4.3. Looking up MetaKeyword

When a Viewer wants to look up a certain channel with some keywords,
the look-up process is going to be done in the following order.
1. The user inputs keywords per each description type.
2. The Viewer Agent hashes the inputted keywords and looks up IPTV metadata from the overlay.
3. The Viewer Agent lists up the result of looking-up in order. The user can see the list of the channels which have keywords. This list shows short description of each channel.
4. The user can choose one among the channels on the list.

A Viewer SHOULD make ResourceID by hashing the user inputted keyword and description type. The Viewer sends FetchReq to the overlay, which includes ResourceID and Kind-ID. Kind-ID in the FetchReq is IPTV-META-KEYWORD.

The Viewer can get the result of searching a specific keyword with IptvMetaType. The result includes the short description of the channel and Channel-ID. After the user choose the channel, Viewer agent SHOULD look for Channel peer list.

4.4. IPTV-META-KEYWORD Kind Definitions

The distributed IPTV metadata storage for keyword searching is provided using the IPTV-META-KEYWORD Kind-ID:

- Kind ID - The Resource name for the IPTV-META-KEYWORD is IptvMetaType and keyword. The format of Resource name is: IptvMetaType "+" Keyword

  The IptvMetaType in Resource Name is a char string. For example, if the IptvMetaType is iptv_metatype_title and the keyword is "Bob", the resource name is "1+Bob". The data stored is a IptvMetaKeyword, which contains short description and Channel ID.

- Data Model - The data model for the IPTV-META-KEYWORD is dictionary. There MAY be many records with the same Resource-ID. The dictionary key is the Channel-ID. A Peer can distinguish stored items with Channel-ID. The data structure for IPTV-META-KEYWORD is IptvMetaKeyword

- Access Control - If certificate-based access control is being used, stored data of Kind-ID IPTV-META-KEYWORD must be signed by certificate which contains Channel-ID matching the Channel-ID in IptvMetaKeyword structure.
4.5. Registering MetaFullDesc

When a Content Provider wants to store the full description about his content, the Content Provider MAY use IptvMetaFullDesc structure to store the full description about the channel into the overlay. User also can get full description from the overlay before he watches it.

A Content Provider sends StoreReq to the overlay. The Resource Name of this message is Channel-ID. The responsible peer of the hashed channel-ID stores the requested IptvMetaFullDesc data structure. Kind ID of this StoreReq is IPTV-META-FULLDESC.

The contents of the IptvMetaFullDesc are as follows:

```c
struct {
    opaque         title<0..2^16-1>;
    opaque         synopsis<0..2^16-1>;
    opaque         genre<0..2^16-1>;
    opaque         language<0..2^16-1>;
    opaque         cast<0..2^16-1>;
    opaque         group<0..2^16-1>;
    opaque         credit<0..2^16-1>;
    opaque         review<0..2^16-1>;
    opaque         crid<0..2^16-1>;
} ContentsDescription;

struct {
    uint64           start_time;
    uint64           finish_time;
    opaque           contents_provider_name<0..2^16-1>;
    opaque           URL<0..2^16-1>;
    opaque           logo<0..2^16-1>;
} InstanceDescription;

struct {
    opaque           channel_id<0..2^16-1>;
    opaque           short_description<0..2^16-1>;
    ContentsDescription contents_desc;
    InstanceDescription instance_desc;
} IptvMetaFullDesc;
```

The content of the IptvMetaFullDesc PDU are:
channel_id
  the identifier of the channel. Unhashed ID

short_description
  short description about the content. Normally, this is a title of
  the content.

contents_desc
  Descriptions and other metadata for this channel

instance_desc
  Start, finish time, and other information of this channel

The contents descriptions and instance descriptions are inherited
from [TVAF]. If a user needs more information, he can use additional
URL in InstanceDescription.

* Comment: This length of element in IptvMetaFullDesc is not fixed
  in this version of draft. The total size of RELOAD message SHOULD be
  smaller than 16K bytes. This limitation will be acceptable for
  IptvMetaFullDesc.

4.6. Looking up MetaFullDesc

When a Viewer knows Channel-ID and he want to get more description
about this channel, he SHOULD send FetchReq to the overlay which Kind
ID is IPTV-META-FULLDESC. The viewer can receive the full description
about the channel in FetchAns from the overlay. The FetchAns SHOULD
include IptvMetaFullDesc as a result.

4.7. IPTV-META-FULLDESC Kind Definitions

The distributed IPTV metadata storage for the full description is
provided using the IPTV-META-FULLDESC Kind-ID:

o Kind ID - The Resource name for the IPTV-META-FULLDESC is Channel
  ID.

o Data Model - The data model for the IPTV-META-FULLDESC is
dictionary. There MAY be many records with the same Resource-ID.
The dictionary key is the Channel-ID. A Peer can distinguish
stored items with Channel-ID. The data structure for IPTV-META-
FULLDESC is IptvMetaFullDesc
o Access Control - If certificate-based access control is being used, stored data of Kind-ID IPTV-META-FULLDESC must be signed by certificate which contains Channel-ID matching the Channel-ID in IptvMetaFullDesc structure.

5. Channel peer list storage

5.1. Overview

P2PSIP DHT can store peer list for a channel. The responsible peer of a certain channel ID stores all peers who are sending, relaying, or receiving this channel. This is very similar to Tracker based architecture that is described in Section 3.1. The Content Provider registers the Channel-ID and its Peer ID to the overlay. If a peer can provide media to other peer, this peer registers its Peer ID to the overlay. A peer which want to watch the channel receives peer list from the overlay. This list consists of a Contents Provider and all peers of the channel. After the peer gets peer list, peer chooses and connects best peers using a specific protocol such as PPSP[PPSP].

The following figure shows the overview of Channel peer list storage.

```
Peer A                                Peer B            Peer C
(1111)          Overlay Peers         (2222)            (3333)
|                  |                  |                 |
|StoreReq(Ch#9)-->
<--------StoreAns

|<--FetchReq(Ch#9)
|FetchAns(1111)-->
|-------------------Attach------------------->
|<==== Media Signaling/Transport ====>
|<==---StoreReq(Ch#9)
|StoreAns-------->|

|<----------------------FetchReq(Ch#9)
|FetchAns(1111,2222)------>
|<---Attach------>
|<== Media S/T ==>
|-------------------Attach------------------->
|<================Media Signaling/Transport==>>
|<-------------------StoreReq(Ch#9)
|StoreAns ----------------->|
```

Peer A is a Content Provider. Peer A sends StoreReq to register new channel and its Peer ID. This StoreReq includes Channel ID (Ch#9) and A’s Peer ID 1111. The resource name of this is Channel ID.

If Peer B wants to watch Ch#9 channel, Peer B sends FetchReq into the overlay. The overlay replies with FetchAns. This FetchAns includes the peer list. The peer list includes Peer ID 1111 now. Peer B sends AttachReq to Peer ID 1111. This message has Peer B’s ICE candidates [I-D.ietf-mmusic-ice]. This message will be arrived at Peer A. Peer A replies with AttachAns message that includes Peer A’s ICE candidates. After that, Peer A and Peer B will perform ICE check and make a connection. Peer B may ask Peer A to send media using a certain media signaling protocol. This protocol is not P2PSIP protocol. This protocol can be PPSP or SIP. After Peer B can relay this channel, Peer B sends StoreReq to register Peer B’s Peer ID into the peer list for Ch#9 Channel.

Peer C also does the same process to Peer B. The response, FetchAns, has two peer IDs that are 1111 and 2222. Peer C attaches Peer B and Peer A. Peer C may choose the best peer between Peer A and Peer B using a certain media signaling protocol. Peer C can get more information about peers by communicating with other peers. After Peer C is ready to relay the media to another peer, Peer C register its Peer ID 3333 into the peer list for this Ch#9 channel by sending StoreReq.

5.2. Registering channel peer list

To register a channel peer ID, StoreReq message SHOULD include IptvChanPeerList structure. Normally, StoreReq include only one Peer ID in IptvChanPeerList structure. But, it is possible to use more than one Peer ID in IptvChanPeerList in StoreReq. The Kind ID of this storing data is IPTV-CHAN-PEER-LIST.

The contents of IptvChanPeer structure are defined as follows:

```
struct {
    Destination     peer_id
    opaque         location_info<0..2^16-1>
} IptvPeerInfo;

struct {
    uint32         peer_list_length;
    IptvPeerInfo     peer_list[peer_list_length];
} IptvChanPeer;
```

The contents of the IptvChanPeer PDU are:
peer_id - the identifier of peers of this channel.

location_info - this peer’s virtual position. peer can get this position from network coordinate system[GNP], ISP(AS) number, or IP address based mapping. ISP number can be worked with ALTO service[RFC5693]. Peer MAY user this location information to select the best peer.
* Comment : The format of location_info is not fixed yet.

peer_list_length - the number of the peers in the list.

peer_list - the list of IptvPeerInfo.

5.3. Looking up channel peer list

To get channel peer list of a channel, FetchAns message SHOULD include IptvChanPeerList structure. The responsible peer replies with all peer list.

5.4. IPTV-CHAN-PEER-LIST Kind Definition

The distributed IPTV channel peer list storage is provided using the IPTV-CHAN-PEER-LIST Kind-ID:

- Kind ID - The Resource name for the IPTV-CHAN-PEER-LIST is Channel ID.

- Data Model - The data model for the IPTV-CHAN-PEER-LIST is dictionary. The dictionary key is the Peer-ID.

- Access Control - If certificate-based access control is being used, stored data of Kind-ID IPTV-CHAN-PEER-LIST must be signed by certificate which contains Peer-ID matching the Peer-ID in IptvChanPeerList structure.

6. Active storage & Channel peer group storage

6.1. Active Storage & Overview

The existing P2PSIP RELOAD supports passive storage only. The responsible peer SHOULD store a requested data without any change. The response also SHOULD be the same to what was stored. The responsible peer SHOULD NOT re-arrange, modify, or select the stored
data. The responsible peer works passively. For redundancy, the stored data in the redundancy responsible peer (the successor of the responsible peer) SHOULD be synchronized to it in the main responsible peer. The responsible peer uses the same StoreReq that it received from the requesting peer. That’s why P2PSIP RELOAD is passive storage.

To make the responsible peer more active, this document suggests an active storage. In the active storage, the responsible peer has some active logic. It can re-arrange, modify, or select the stored data. It can store the requested data into another peer.

The channel peer list storage which is explained in Section 5 is not scalable. It is possible that there is lots of peer in a channel. Under the flash crowd, the number of peer can be more than a million. It is impossible for one peer to store all peer list. Moreover, RELOAD message cannot carry this huge data.

If the responsible peer of the channel has active logic, it can store the data in another distributed manner. The responsible peer forms peer groups and stores grouped peer list into another peers. This grouping logic SHOULD be the same to the redundancy responsible peer’s. This responsible peer keeps information about groups, which information MAY includes group ID and the number of peer in a group. This document provides virtual network coordinate based grouping strategy. Every peer can get its virtual position using a network coordination system or an IP address mapping. Grouping is performed using peer’s location, the number of peer in group, and the number of group. This grouping can be performed hierarchically. We can call an intermediate responsible peer a group manager. A group manager can reassign groups with StoreReq(IPTV-CHAN-PEER-GROUP).

* Comment : The exact grouping strategy is to be defined.

6.2. Registering channel peer group

After the responsible peer of the Channel ID receives a StoreReq with IPTV-CHAN-PEER-LIST, it assigns this requesting peer to a group. The responsible peer sends a StoreReq with IPTV-CHAN-PEER-GROUP to the overlay. The Resource name of this message is Channel ID, group level, and group ID. This message uses IptvChanPeerGroup structure.

The contents of a IptvChanPeerGroup are as follows:
struct {
    uint32         level;
    uint32           group_id;
    uint32          peer_list_length;
    IptvPeerInfo     peer_list[peer_list_length];
} IptvChanPeerGroup;

level - grouping level. This document supports hierarchical grouping. The first level is 1.

group_id - group id in this level. peer info is stored in this group.

peer_list_length - the number of the peers in the list.

peer_list - the list of IptvPeerInfo.

The following figure shows an example of Channel peer group storage.

<table>
<thead>
<tr>
<th>Peer D</th>
<th>Peer X</th>
<th>Peer Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4444)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>StoreReq(Ch#9,4444) --&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>assign group</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StoreReq(Ch#9,L1,G3,4444)-&gt;</td>
</tr>
</tbody>
</table>

Peer D is new peer. Peer X is the responsible peer of this channel. After Peer X receives Peer D’s StoreReq, Peer X decides to assign Peer D to group#3 on group level 1. And, Peer X sends StoreReq with IPTV-CHAN-PEER-GROUP to Peer Y. Peer Y is the responsible peer of Channel-ID plus level plus group ID. Peer Y stores the peer information into this group.

6.3. Looking up channel peer group

The responsible peer of the Channel ID receives FetchReq with IPTV-CHAN-PEER-LIST. The response, FetchAns, uses IptvChanPeerGroup. This response SHOULD include all peer information list or a part of all peer information in the group. The responsible peer of Channel ID MAY give the requesting peer initial peer list instead of whole peer list in the group.
6.4. IPTV-CHAN-PEER-GROUP Kind Definition

The distributed IPTV channel peer group storage is provided using the IPTV-CHAN-PEER-GROUP Kind-ID:

- Kind ID - The format of Resource name for the IPTV-CHAN-PEER-GROUP is:
  
  Channel ID '+' grouping level '+' group ID

- Data Model - The data model for the IPTV-CHAN-PEER-GROUP is dictionary. The dictionary key is the Peer-ID.

- Access Control - If certificate-based access control is being used, stored data of Kind-ID IPTV-CHAN-PEER-GROUP must be signed by certificate which contains Peer-ID matching the Peer-ID in IpvTvChanPeerGroup structure.

7. Security Considerations

TODO - fill in

8. IANA Considerations

IANA SHALL add IPTV-META-KEYWORD, IPTV-META-FULLDESC, IPTV-CHAN-PEER-LIST, and IPTV-CHAN-PEER-GROUP into "RELOAD Data Kind-ID" Registry. The additional contents of the registry are:

<table>
<thead>
<tr>
<th>Kind</th>
<th>Kind-ID</th>
<th>RFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPTV-META-KEYWORD</td>
<td>16</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>IPTV-META-FULLDESC</td>
<td>17</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>IPTV-CHAN-PEER-LIST</td>
<td>18</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>IPTV-CHAN-PEER-GROUP</td>
<td>19</td>
<td>RFC-AAAA</td>
</tr>
</tbody>
</table>

9. Acknowledgments

TODO - fill in
10. References

10.1. Normative References


10.2. Informative References


Author’s Addresses

Seok-Kap Ko
ETRI
1000-6 Oryong-dong, Buk-gu, Gwangju, 500-480, Korea
Phone: +82-62-970-6677
Email: softgear@etri.re.kr

Young-Han Kim
Soongsil University
Sangdo-dong, Dongjak-Gu, Seoul, Korea
Phone: +82-2-814-0151
Email: younghak@ssu.ac.kr

Seung-Hun Oh
ETRI
1000-6 Oryong-dong, Buk-gu, Gwangju, 500-480, Korea
Phone: +82-62-970-6672
Email: osh93@etri.re.kr

Byung-Tak Lee
ETRI
1000-6 Oryong-dong, Buk-gu, Gwangju, 500-480, Korea
Phone: +82-62-970-6624
Email: bytelee@etri.re.kr

Victor Pascual Avila
Tekelec
Am Borsigturm 11, Berlin
Germany
Email: victor@iptel.org