

Implementation of PREP (P2P-based stREaming Protocol) Client

Wook Hyun^a, SeungHei Kim^a, ChangKyu Lee^b, ShinGak Kang^a
^a ETRI(Electronics and Telecommunications Research Institute), Korea
^b University of Science & Technology

Corresponding Author: whyun@etri.re.kr, Fax: +82 42 861 5404, Tel: +82 42 860 1565

Abstract— on providing multimedia streaming services such as live streaming, VoD, server-client model has been used. By the way, server-client model does not provide enough scalability and load concentration problems. Hence, there are several P2P solutions have been emerged. P2P-based streaming service can provide very high scalabilities and accommodate massive number of users simultaneously. In this paper, we present P2P-based streaming technologies and our prototype implementations.

Keywords— P2P, Streaming Client

I. INTRODUCTION

According to the “Internet Study 2008/2009” of Iopque[1], P2P traffic takes more than 42.51% of total Internet traffic. Multimedia contents may be exchanged as a form of file, streaming. In case of streaming service, traditional server-client model have limitations on accommodating large number of simultaneous users, and suffer from traffic concentrations on its contents media server. This will degrade the performance and service quality. Hence, there are several solutions has been emerged by using P2P techniques. By using P2P approach, multimedia data can be delivered to massive number of receiver without suffering from bottleneck problems. There are several P2P-based ongoing streaming services such as *octoshape*[2] and *pptv*[3]. Also, there are other experimental implementations such as *goalbit*[4] and *NAPA/WINE*[5]. Furthermore, *BitTorrent*[6] takes an experiments on providing live streaming using existing file-sharing infrastructure.

The P2P-based streaming method is very adequate to provide contents to massive number of users simultaneously. The performance of P2P increases proportionally to the number of participating users, i.e., peer.

II. P2P-BASED STREAMING MODEL

In constructing P2P overlay network for providing multimedia streaming service, we can classify three categories; tree-based, mesh-based and hybrid-based. Different from P2P file-sharing services, the users of streaming services keep watching similar part of contents. P2P-based file sharing service can be delay-tolerant and does not need to delivery timely fashion.

In this chapter, we describe the three overlay models briefly.

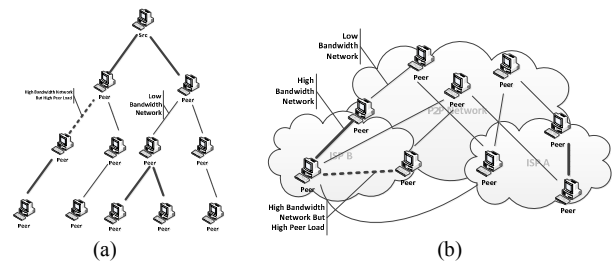


Figure 1. Architectural model for P2P-based streaming service

A. Tree-based Overlay Network

As the name implies, it create an overlay architecture looks like a tree. The contents server will be the root node of the tree, and joining peers will be child of root node or other peers as shown in figure 1(a). Tree-based model can provide more real-time streaming than mesh-based network. But it is resilient to the failure of intermediate nodes. Usually, push-type data delivery is used in this model.

B. Mesh-based Overlay Network

In constructing mesh-based overlay network, there is no hierarchy as shown in figure 1(b), and all nodes have equal position. In this model, media stream data is fragmented into fixed-size chunks. Data exchanges among peers are performed in the basis of chunk. Hence, it may not appropriate hard real-time streaming such as video communication when it comes to compare with tree-based model. By the way, mesh model can provide high scalability and robustness even though there are failures on the intermediate node.

C. Hybrid-based Overlay Network

Tree model and mesh model both have their pros and cons. The hybrid-model can provide high reliable, low delay and low start-up delay, but it also inherits the disadvantages of both approaches. This method is still under research among several research groups.

III. ARCHITECTURAL MODEL FOR P2P-BASED STREAMING SERVICE

We have designed an architectural model for providing P2P-based streaming services and client systems in the previous paper [7].

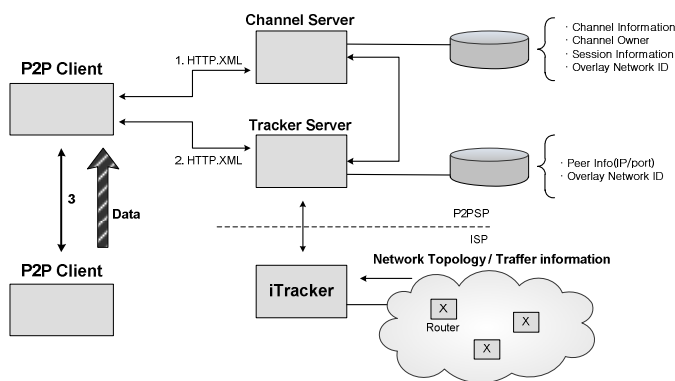


Figure 2. Architectural model for P2P-based streaming service

We have chosen mesh-based overlay network for providing streaming services and all data is exchanged using pull-type protocol.

A. Channel Creation

When a P2P client wants to provide streaming services, it registers meta-information into the channel server. This meta-information usually contains title, running time, categories, and so on. On receiving channel creation request from a peer, channel server requests to tracker server for issuing an overlay network ID. After that, it responds to P2P client with the ID. This ID is used for manipulating overlay network.

B. Overlay Network Creation

A client that wants to provide streaming service create overlay network by requesting to tracker server with the ID that is issued on channel creation. The tracker server makes a logical overlay network by preparing database.

C. Channel Search

A user who wants to search channel information, it access to channel server. Because our implementation uses a HTTP protocols for conveying channel information, we can provide two kind of interfaces; Web-interface and XML-interface. A user can see the channel list through web browser. Besides, we can also provide XML-based channel descriptions. In the latter case, it will be useful for embedded devices that does not provide web interface. When a user selects a channel that he wants to receive, the channel server returns the ID of the overlay network for the contents.

D. Overlay Network Join

A client that wants to receive a specific streaming session, it accesses to tracker server with the ID. When it joins a specific overlay network, the tracker server returns *peerlist* information that contains addresses of participating peers. It is also possible to optimize *peerlist* by use of *iTracker* that have information about underlying network.

E. Peer Interaction

When a peer receives *peerlist* from tracker server, it connects to each peer for retrieving media data. Before

retrieving chunk of media, it negotiates with each other by exchanging buffermap. When a peer wants to receive specific chunk, it request by indicating the chunk ID.

IV. IMPLEMENTATION OF P2P-BASED STREAMING CLIENT

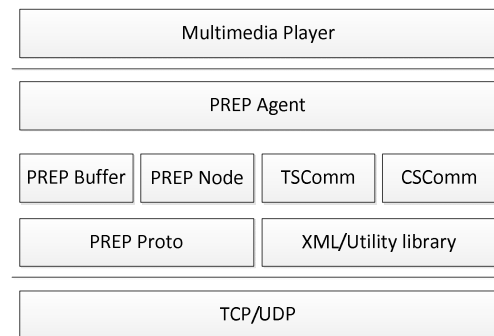


Figure 3. Functional blocks for P2P streaming client

We have used VLC[8] for displaying media contents. VLC is also capable of transmitting media data through UDP. Hence, it is also possible to receive media stream from other VLC client through UDP. The PREP client of source node can receive UDP stream from VLC client, and it generates chunks with fixed-size and maintains internal buffermap. This buffermap is used for negotiating with the other peer. It is also possible to operate independently with VLC client. Logically, VLC client is divided with the PREP Agent that control overlay behaviour. In this approach, PREP agent is just transport mechanism and does not interrupt the behaviour of VLC client.

In this chapter, we present the details of internal function blocks.

A. TSComm/CSComm

TSComm library is used for communicate with tracker server, and CSComm library is used for interacting with channel server. In both cases, we used XML-based descriptions and HTTP for deliver this information.

1) Tracker server communication: When a peer joins an overlay network, it firstly communicates with tracker server to retrieve peerlist of overlay network.

The figure 4 shows the structure of XML document for describing peerlist of specific overlay network. It mainly consists of two parts; prep-network and prep-node. The *prep-network* field contains an overlay network identifier, and this ID is used for differentiating overlay network. The *prep-node* fields contain peerlist of the overlay network, and this field can be repeated if there are multiple peers within an overlay network. Each peer has own identifier, and it express its preferences on handling incoming connections. The node-activity field shows the action of requesting peer, such as join, leave, pause, and also it can describe its status including upload, download and active time for the overlay network. This information is used for providing incentives, but our implementation does not consider how to handle it yet.

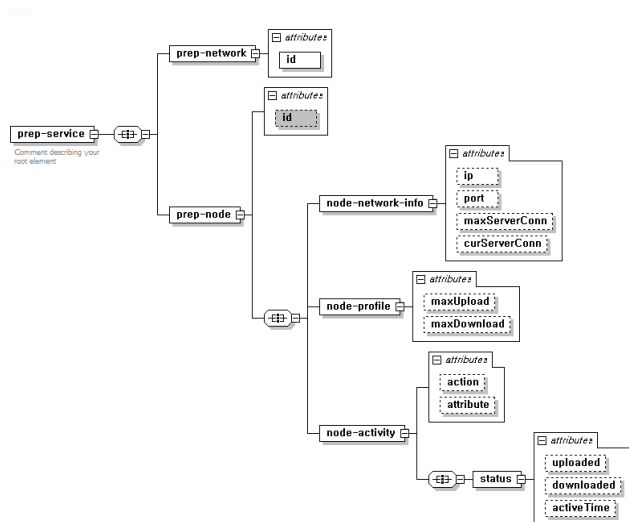


Figure 4. XML structure for describing the activity of P2P nodes

2) Channel server communication: When a contents owner wants to provide streaming service, it firstly registers its contents meta-information into the channel server. It uses the XML as shown in figure 5. It is also possible to provide not only video service but also audio only service by indicating within the *type* attribute.

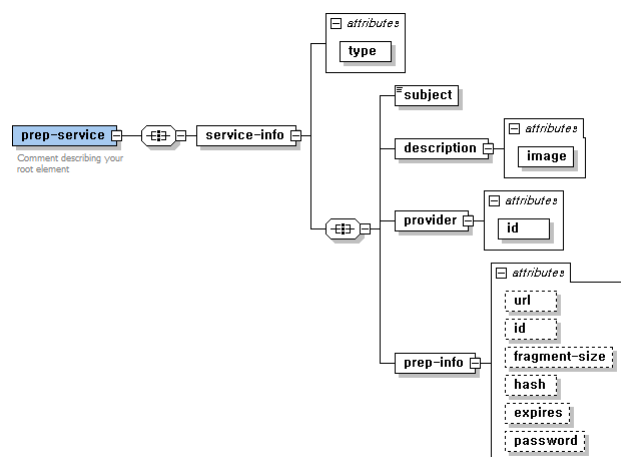


Figure 5. XML structure for describing channel information

This XML contains contents meta-information and overlay specific parameters, such as overlay network ID and fragment (chunk) size.

B. PREP Buffer

PREP Buffer management block maintains chunked data and internal buffermap. When a PREP agent requests buffermap, it returns current status. This functional block provides following functionalities.

1) Internal Buffer Initialization: A PREP client is capable of joining multiple streaming sessions, and it maintains a

buffer for an overlay network. This sub-routine is called for every overlay network join events.

2) Receive stream through UDP: In order to receive raw contents data from another source, it is required to provide interaction point from contents owner. Since we have separated the PREP agent and stream generator/player, we prepared a UDP server that receives media stream data. This sub-routine is generally called when a peer is acting as a streaming source node. This sub-routine also acts as a stand-alone thread for keeping receiving incoming packet regardless of the activity of P2P network.

3) Send/Receive chunk from other peer: If it acts as a generic peer that sends and receives chunk data with other peers, it invokes this function. In order to simulate heterogeneous overlay network, we have added some codes for randomizing the transfer rate. This function keeps monitoring the size of received packet for making a fixed size of chunk.

C. PREP Agent

The PREP Agent takes a main control of the other functional blocks. It internally creates listening thread for communicating with the other peers.

1) Overlay network join/leave: When a PREP agent tries to join a specific overlay network, it uses overlay network identifier that is retrieved from channel server. If the overlay network is valid, it creates agent instance internally. The instance number of PREP agents is same with the number of participating overlay network. When it comes to leave the overlay network, it communicates with tracker server to notify the event. After that, it removes related information from internal memory.

2) Interaction with tracker server: Whenever it joins and leaves overlay network, it communicate with tracker server. Actually, *TSComm* functional block takes charge of protocol-level interaction, and PREP agent calls appropriate function call within *TSComm* library.

D. PREP Node

This functional block manages neighbouring peer and communicate with the other peer through gossip protocol. The gossip protocol is embedded within the PREP proto functional block.

V. INTEGRATION WITH GUI APPLICATION AND OPERATIONAL MODEL

We have integrated PREP implementation with GUI interface as shown in figure 5. The GUI application display received stream, and it can also provide media stream to PREP agent for sharing with the other peers. In order to import the functionalities of VLC, we have packaged into COM object.

The figure 6 shows a situation that one contents owner provide its contents through the help of PREP agent. It can

receive input stream from cam, file and network, since we have used VLC. The PREP agent of contents owner makes chunks with fixed size and make them share with the other peers with the fashion of P2P mesh network. The other PREP client sends and receives multiple chunks with each other simultaneously.

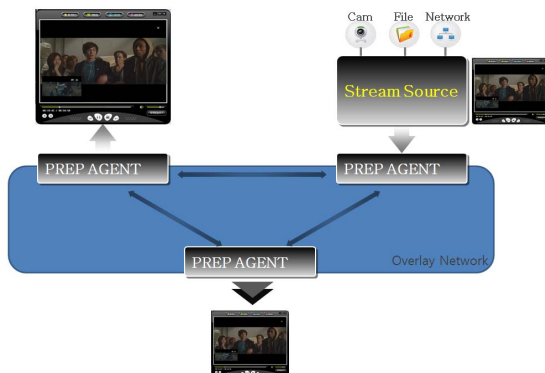


Figure 6. Operation model for service provider and customers using PREP client

One of main characteristics of our implementation is that it can play up to four streaming session at the same time, since a PREP agent can join multiple overlay network. Logically, it can accommodate more session if it can afford to heavy traffic and multimedia processing.

The figure 7 shows the situation that a client joins multiple overlay networks. The screen is consists of two parts; main display part and thumbnail parts. Of course, it is possible to switch with each other.

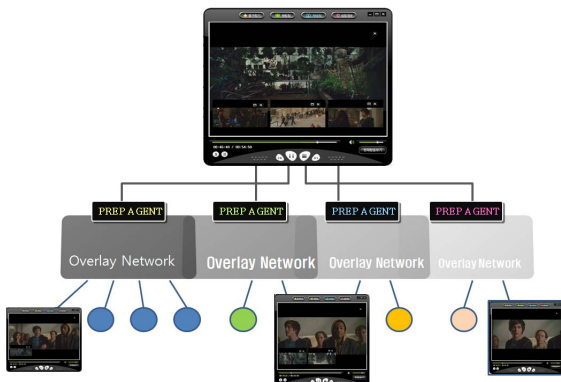


Figure 7. Operation model for watching multiple P2P streaming session.

VI. FUTURE WORKS

Our implementation is based on mesh-based overlay network and pull-type data distribution. It receives media stream through UDP socket, and then it create chunks based on the size of received stream. Each chunk has same fixed size for the initial stage and this size does not change over the session ongoing. By the way, the size of chunk takes effects on the service quality and should be differ from the quality of video stream. Therefore, it is required to find most optimal values for different video quality. In the next paper, we will provide some experimental results.

VII. CONCLUSIONS

In this paper, we present the implementation of PREP (P2P-based streaming protocol) client. It consists of two parts; PREP agent and GUI player. The PREP agent takes charge of P2P-based contents delivery, and GUI plays received media data as well as providing stream to PREP agent. Our implementation can be used for both contents provider and contents consumer.

ACKNOWLEDGMENT

This research was supported by the ICT Standardization program of MKE (The Ministry of Knowledge Economy).

REFERENCES

- [1] Hendrik Schulze, Klaus Mochalski, "Internet Study 2008/2009", ipoque, 2009
- [2] Octoshape, "<http://www.octoshape.com/>"
- [3] PPTV, "<http://www.pptv.com/>"
- [4] Goalbit, <http://goalbit.sourceforge.net/>
- [5] NAPA/WINE project, "<http://www.napa-wine.eu/>"
- [6] BitTorrent, "<http://www.bittorrent.com/>"
- [7] W.Hyun et al, "Study on Architectural Model and Design of P2P-based Streaming Client", ICHIT2011, CCIS 206, pp.1-7, 2001.
- [8] VLC(Video Lan Connection), "<http://www.videolan.org/>"