Journal of Knowledge Information Technology and Systems

ISSN 1975-7700

http://www.kkits.or.kr

An Efficient Data Exchange Protocol for Peer-to-Peer Live Video Streaming based the Web

Jinsul Kim[*1], **Sanghyun Park**[1], **Dong-Geon Lee**[2]

[1] *School of Electronics& Computer Engineering, Chonnam National University*
[2] *Department of Computer Science, Gwangyang Health College*

ABSTRACT

Currently, service providers are facing with problem is slash dot effect or flash crowds when many users increase significantly and at the same time access into one live event. Now to address these issues, the combined benefits of P2P CDN CDN P2P hybrid system is being used, but the server's processing ability of a hybrid P2P CDN system has reached the critical point and the majority of users are constantly want to receive better quality content. Accordingly, the service provider has to spend money to add servers continuously. Peer-to-Peer network is one of a solution to solve this problem. To decrease request data to the server, users can request data from other users in the same network to get the desired content. In this paper, we propose the efficient data exchange protocol for P2P live video streaming over the Web in order to decrease the number of requests to servers that means reduce the cost of transmission. Also by set up short timeout for request to server and split video into small chunk, our system has smaller end-to-end delay than unconnected mesh system in CDN-P2P-hybrid. In our setup environment, we compared the experiment the number of requests for change in delay between the server and the termination of the proposed method and the conventional method.

KEYWORDS: Content delivery network, Peer-to-peer, Live video streaming, WebRTC

ARTICLE INFO: Received 22 September 2015, Revised 8 October 2015, Accepted 8 October 2015.

1. Introduction

With the development of high-speed and broadband networking, the content delivery service

[*]Corresponding author is with School of Electronics& Computer Engineering, Chonnam National University, Gwangju, 500-757 Republic of Korea.
E-mail address: jsworld@jnu.ac.kr

has been grown up widely. There are a lot of providers for online streaming via Content Delivery Network (CDN) and Peer to Peer (P2P) network, each having its own set of advantages and disadvantages. CDN provide excellent quality to end-users when the work load is within the limits. However, CDN servers are expensive to deploy and maintain. While P2P network has better scalability and less deployment costs, but the instability of dynamic peers, and the low performance when participated peers are insufficient are a bottleneck [1]. So, by combining advantages of two system we will take advantage of both.

The following goals are set to efficiently streaming live video over internet: optimization cost for streaming live video, propose new idea that balance between cost and delay for live video streaming service make it easy for user to streaming user using their smart phone, PC without any more installed software by using Web Real-time Communication technique (WebRTC) [2][3]. By using web browser with supported WebRTC [4], user don't need to install any application or software to streaming live video [5]. This approach is promise way because today, all of smart phones can run web browsers such as Chrome or Firefox. In this paper, we propose the hybrid CDN-P2P structure with topology optimization for P2P structure for live video streaming over the Internet by using WebRTC in order to decrease the number of requests to CDN servers, reducing the cost of transmission and also reduce end-to-end delay while watching live events.

2. Related Work

The main purpose of a CDN is to distribute contents over a set of web servers highly distributed around the world, so as to guarantee a reliable, scalable and efficient delivery of the contents to end users [6][7]. Let's give a real example to understand about CDN: A website is hosted on a web server that's located in Korea. Now if a visitor from US want to access this website, the page loading time for him will be relatively high because of the geographic distance between Korea and US. Now a content delivery network has servers across the world and they automatically determine the fastest (or the shortest) route between the server hosting the site and the end-user [8]. There are CDN servers in either UK or Mexico, the page would load much faster for that visitor from US. We are not aware of any real implementation and deployment that actually demonstrates the benefits of such a hybrid approach for live video streaming. In this paper, we present the design, implementation, and real-world deployment and evaluation of a hybrid CDN-P2P live streaming system. By combining advantages of two system we will take advantage of both.

3. Proposed System

In this paper, we design our whole system as <Figure 1>. For detail of system:

One broadcaster: users use their own camera to capture live video and broadcast to media server [9][10].

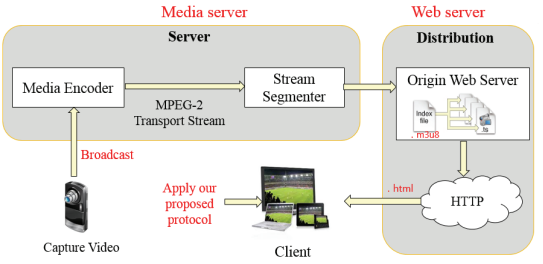


그림 1. 전체 시스템 아키텍처
Figure 1. Whole System Architecture

One media server, we use HTTP live streaming (HLS) technique [11] to encode RTMP stream into .ts file and create index.m3u8 file that manages .ts files as shown in <Figure 2>.

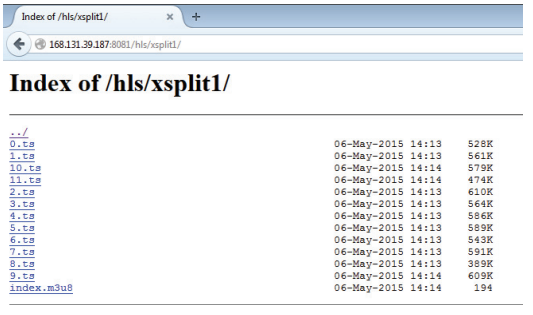


그림2. HLS 서버 디렉토리
Figure 2. HLS Server's Directory

One HTTP server to run .html in Client's Side. One tracker server: track peer join, leave, push it into swarm.

We use ISP-location and geo-location awareness concepts to build swarm of peers that can exchange messages between them. When a peer wants to watch a live streaming, the peer join procedure: peer A hits/accesses a link URL (live events…), tracker will publish peer A on the swarm with other peers (request to the same link URL), create the room for P2P connection. Other Peers acknowledge that Peer A join the network, and Peer A establishes a P2P connection with other peers on the same swarm.

```
discoverMyRoom: function() {
  var response = utils.request(ROOM_DISCOVER_URL);
  var room = response? JSON.parse(response)['room']: "live";
  utils.updateRoomName(room);
  return room;
},

onOpen: function(dc, id) {
  console.log("Peer entered the room: " + id);
  self.swarm[id] = buffered(dc, {calcCharSize: false});
  self.swarm[id].on('data', function(data) { self.onData(id, data); });
  utils.updateSwarmSize(self.swarmSize());
},

onData: function(id, data) {
  var parsedData = utils.parseData(data);
  var resource = parsedData['resource'];

  if (self.isDesire(parsedData) && resource in self.chunksCache && self.currentState != PEER_UPLOADING) {
    console.log("HAVE RESOURCE, SENDING DESACK " + id + ":" + resource);
    self.currentState = PEER_UPLOADING;
    var desAckMessage = utils.createMessage(CHUNK_DESACK, resource);
    self.swarm[id].send(desAckMessage);

  } else if (self.isDesAck(parsedData) && self.currentState == PEER_DESIRING) {
    console.log("RECEIVED DESACK, SENDING REQ " + id + ":" + resource);
    clearTimeout(self.requestTimeout);
    self.currentState = PEER_DOWNLOADING;
    var reqMessage = utils.createMessage(CHUNK_REQ, resource);
    self.swarm[id].send(reqMessage);
    this.requestTimeout = setTimeout(function() { self.getFromCDN(resource); }, REQ_TIMEOUT *1000);

  } else if (self.isReq(parsedData)) {
    console.log("RECEIVED REQ, SENDING CHUNK " + id + ":" + resource);
    var offerMessage = utils.createMessage(CHUNK_OFFER, resource, self.chunksCache[resource]);
    self.swarm[id].send(offerMessage);
    utils.incrementCounter("chunksToP2P");
    self.currentState = PEER_IDLE;

  } else if (self.isOffer(parsedData) && resource == self.currentUrl) {
    console.log("RECEIVED OFFER, GETTING CHUNK " + id + ":" + resource);
    clearTimeout(self.requestTimeout);
    self.sendToPlayer(parsedData['chunk']);
    utils.incrementCounter("chunksFromP2P");
    self.currentState = PEER_IDLE;
  }
},
```

그림3. 청크 교환 프로토콜
Figure 3. Chunk Exchange Protocol

After the player has received and parsed the play list, it starts to request video chunks as shown in algorithm in <Figure 3>. Instead of the common HTTP request for the chunk from the CDN, the node sends a REQUEST_CHUNK to every node of the peer swarm and each node that receives the REQUEST_CHUNK searches for the chunk in its cache. Our implementation stores the last 10 chunks watched, and if the chunk requested is cached, the node sends back an ACK_REQUEST. The desiring peer looks for the best node from the pool of nodes that sent ACK_REQUEST and sends a REQ to the chosen one. Every peer that receives a REQ is ensured by the previous step that the chunk is in its cache and then it sends an OFFER with the chunk to the desiring peer. When the desiring

peer sends the REQUEST_CHUNK to swarm, it waits for a timeout of 1 seconds and if nobody answers, it requests directly to the CDN using the traditional server client HTTP schema. The same occurs if the node chosen to send the chunk takes more than 1 second to send the file itself

Using Xsplit Broadcaster to broadcast live video: the streaming was split in chunks (.ts files and are managed by index.m3u8 file) with 6 seconds of duration and 650Kbps of bitrate quality using HTTP live streaming protocol.

Server side: using Javascript programming
HLS server with IP address: 167.131.39.187:8081
HTTP server with IP address: 167.131.39.187:8000
Tracker server with IP address: 167.131.39.187:3000

Client side: using HTML programming. We used a total of 10 computers running Mozilla Firefox 40.0 browsers have compatible with WebRTC. All devices were in the same IP network, which means that they were in the same location. All devices were observed during one hour of streaming using our method and then compared with one hour of streaming using traditional method.

표 1. 설치 실험에 사용된 매개변수
Table 1. Parameters Used in the Setup Experiment

Name	Parameter	Description
Experiment time	3600s	Length of the experiment scenario
Number Peers	10	Total number of peers
Chunk length	6s	Length of each chunk
Video length	3600s	Length of broadcasting video file

When we use the conventional Turbo code with the information length of , the total amount of bits to transmit are , where is the number of tail bits of RSCs and here we ignore this value for simple notation.

4. Experiment Results

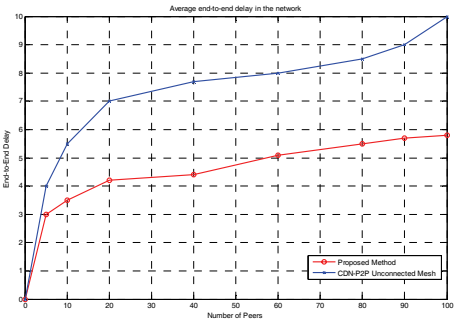


그림 4. 네트워크의 평균 End-to-End 지연시간
Figure 4. Average End-to-End Delay in the Network

End-to-end delay is a time interval between frame creation in the source nodes and playing at destination node. This metric is one of the most important metric for live video streaming. We also compare our proposed system with unconnected mesh in paper [12] to proof that our system has low end-to-end delay than their system. In [12], they proposed a simulation of hybrid CDN-P2P system for live video streaming and compared end-to-end delay with system using P2P mesh-based. We looked at the result for end-to-end delay with simulation for 100 users is 10 seconds. So, we also deploy this system to compare with our method as shown in <Fig.4>. With the same number of peers in the network, for example 100 peers, end-to-end delay of our system only is 5 seconds compare with their system is 10 seconds.

5. Conclusion

In this paper, we proposed the reality experiment hybrid CDN-P2P for live video streaming over the Internet in order to decrease the number of requests to CDN servers, reducing the cost of transmission and enhancing system's scalability. By set up a reality, we compare the number of requests to CDN server in two cases: our method, and traditional CDN server. In this paper, we present the design, implementation, and real experiment deployment and evaluation of a hybrid CDN-P2P for live video streaming. The key contributions of this paper are: design and implementation a scenario of a hybrid CDN-P2P live video streaming for effectively scaling the capacity of a CDN by reducing requests to the CDN server with support of P2P technique, also understanding key challenges in integrating the P2P component into the CDN, overcoming the weaknesses of traditional P2P systems for live streaming. Through experiment, we will show that a hybrid CDN-P2P approach is very effective approach in providing a live streaming service to the public.

References

[1] Lu, Zhihui, Ye Wang, and Yang Richard Yang, *An analysis and comparison of CDN-P2P-hybrid content delivery system and model*, Journal of Communications, Vol. 7, No. 3, pp. 232-245, 2012.

[2] Seyyedi, S. M. Y, and Akbari, B, *Hybrid CDN-P2P architectures for live video streaming: Comparative study of connected and unconnected meshes*, In Computer Networks and Distributed Systems (CNDS), 2011 International Symposium IEEE on. pp. 175-180, 2011.

[3] Jennings, Cullen, Ted Hardie, and Magnus Westerlund, *Real-time communications for the web*, Communications Magazine, IEEE Vol. 51, No. 4, pp. 20-26, 2011.

[4] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang and Bo Li, *Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky*, Proceedings of the 17th ACM international conference on Multimedia, ACM, pp. 25-34, 2009.

[5] Vogt, Christian, Max Jonas Werner, and Thomas C. Schmidt, *Leveraging WebRTC for P2P content distribution in web browsers*, Network Protocols (ICNP), 2013 21st IEEE International Conference on, IEEE, pp. 1-2, 2013.

[6] Enrico Baccaglinia, Marco Grangettob, Emanuele Quacchioc and Simone Zezzad, *A study of a hybrid CDN–P2P system over the Planet Lab network*, Signal Processing: Image Communication, Vol. 27, No. 5, pp. 430-437, 2012.

[7] El Dick, Manal, Esther Pacitti, and Bettina Kemme, *Flower-CDN: a hybrid P2P overlay for efficient query processing in CDN*, Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, ACM, pp. 427-438, 2009.

[8] Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross, *Understanding hybrid*

CDN-P2P: why limelight needs its own Red Swoosh, In Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video, ACM, pp. 75-80, 2008.

[9] XSplit Broadcaster, https://www.xsplit.com/products/broadcaster

[10] Javier Cervino, Pedro Rodriguez, Irena Trajkovska, Alberto Mozo, and Joaquin Salvachua, *Testing a cloud provider network for hybrid P2P and cloud streaming architectures*, Cloud Computing (CLOUD), 2011 IEEE International Conference on, IEEE, pp. 356-363, 2011.

[11] HTTP Live Streaming, https://developer.apple.com/streaming/

[12] S.M.Y. Seyyedi and B. Akbari, *Hybrid CDN-P2P architectures for live video streaming: Comparative study of connected and unconnected meshes*, International Symposium on Computer Networks and Distributed Systems, 2011.

웹 기반의 p2p 라이브 비디오 스트리밍을 위한 효율적인 데이터 교환 프로토콜

김진술[1], 박상현[1], 이동건[2]
[1] *전남대학교 전자컴퓨터공학부*
[2] *한국전자통신 연구원*

요 약

현재 서비스 제공 업체가 직면하고 있는 문제로는 많은 사용자가 라이브 이벤트를 동시에 액세스하려고 할 때 도트 효과 또는 플래시 크라우드 현상이 증가한다는 것이다. 현재 이러한 문제를 해결하기 위해 CDN의 장점과 P2P의 장점을 결합한 CDN P2P 하이브리드 시스템이 이용되고 있지만, 하이브리드 CDN P2P 시스템 서버의 처리능력은 한계점에 도달했으며, 대다수의 사용자들은 지속적으로 더 나은 품질의 콘텐츠를 제공받기를 원한다. 이에 따라 서비스 제공 업체는 서버를 추가하기 위해 지속적으로 비용을 지출해야만 한다. p2p 네트워크는 이러한 문제점을 해결하는 하나의 해결책으로 서버에 요청되는 데이터를 줄이기 위하여 사용자가 원하는 콘텐츠를 동일한 네트워크 내의 다른 사용자들로부터 데이터를 요청할 수 있다. 본 논문에서는 송신 비용을 줄이기 위해 서버에 대한 요청 개수를 감소시키는 방법을 사용한다. 즉, 웹을 통한 P2P라이브 스트리밍 비디오의 효율적인 데이터 교환 프로토콜을 제안한다. 또한, 작은 청크안에 서버와 분할 비디오의 요청시간이 작기 때문에 CDN P2P 하이브리드 시스템에 연결되어 있지 않은 때 보다 연결시간이 적게 든다. 동일한 환경에서, 우리는 제안된 방법과 기존 방식의 서버 및 종단 간 지연에 대한 요청개수 변화를 실험을 통해 비교 분석한다.

Acknowledgments

This research was supported by Chonnam National University, 2013.

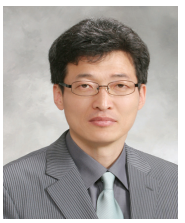
Jinsul Kim received the B.S. Degree in computer science from University of Utah, Salt Lake City, Utah, USA, in 2001, and the M.S. and Ph.D. degrees in digital media engineering, department of information and communications from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2005 and 2008. He worked as a researcher in IPTV Infrastructure Technology Research Laboratory, Broadcasting/ Telecommunications Convergence Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea from 2005 to 2008. He worked as a professor in Korea Nazarene University, Chon-an, Korea

from 2009 to 2011. Currently, he is a professor in Chonnam National University, Gwangju, Korea. He has been invited reviewer for IEEE Trans. Multimedia since 2008. He has been invited for TPC(Technical Program Committee), IWITMA2009/2010, and PC(Program Chair), ICCCT2011 His research interests include QoS/QoE, Measurement/ Management, IPTV, Mobile IPTV, Smart TV, Multimedia Communication and Digital Media Arts.

E-mail address: jsworld@jnu.ac.kr

Sanghyun Park received his B.S. Degree in Computer and Information from the University of Korea Nazarene in 2010, and the M.S. degree in School of Electronics and Computer Engineering, Chonnam National University, South Korea. He worked as an engineer in System Development Team of Media Flow Company from 2010 to 2012. He is now studying Ph.D Degree in School of Electronics and Computer Engineering, Chonnam National University. His research interests are Interactive Media, Systems Development, Embedded systems, Digital Media and Cloud computing.

E-mail address: sanghyun079@gmail.com



Dong-Geon Lee received the Ph.D. in Electronic Commerce from Chonnam National University. Since 1995, he is working for Gwangyang Health College as assistant professor in the Department of Computer Science. His current research interests include Management Information Systems, Smart Computing, Electronic Commerce, IT Service and Ubiquitous Business.

E-mail address : dklee@gy.ac.kr