Wireless Pers Commun (2017) 94:513–525
DOI 10.1007/s11277-015-3144-1

Design and Deployment of Low-Delay Hybrid CDN–P2P Architecture for Live Video Streaming Over the Web

Tran Thi Thu Ha[1] · **Jinsul Kim**[1] · **Jiseung Nam**[1]

Published online: 23 December 2015

Abstract In the recent years, with the development of high-speed and broadband networking, the content delivery service 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 provisioning 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. So, by combining advantages of two system we will take advantage of both. In this paper, we propose the hybrid CDN–P2P structure for live video streaming over the Web in order to decrease the number of requests to CDN 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 Seyyedi and Akbari (Hybrid CDN–P2P architectures for live video streaming: comparative study of connected and unconnected meshes. In: International symposium on computer networks and distributed systems, 2011). By set up a reality, we compare two metrics: the number of requests to CDN server and end-to-end delay with previous researches. Through experiment, we will show that a hybrid CDN–P2P approach is very effective approach in providing a live streaming service to the public.

Keywords Peer-to-peer network · Content delivery network · Live video streaming · Hybrid CDN–P2P

✉ Jinsul Kim
jsworld@chonnam.ac.kr

Tran Thi Thu Ha
thuhabkhn@gmail.com

Jiseung Nam
jsnam@jnu.ac.kr

1 Department of Electronics and Computer Engineering, Chonnam National University, Gwangju, Korea

1 Introduction

According to Cisco's visual networking report, the sum of all forms of video (TV, video on demand, Internet, and P2P) will continue to exceed 91 % of global consumer traffic by 2014. Internet video alone will account for 57 % of all consumer Internet traffic in 2014. That means video streaming over the Internet has the promising future. Now, 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. It will make server down. Server issues can ruin whole event. For example, when Apple launched a new range of devices in September, 2014, many fans were disappointed when they couldn't see the live stream.

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 smartphone, PC without any more installed software by using Web Real-time Communication technique (WebRTC) [2]. By using web browser with supported WebRTC, user don't need to install any application or software to streaming live video. This approach is promise way because today, all of smart phones can run web browsers such as Chrome or Firefox.

The remaining of this paper is as follows: part 2 gives some related works which we learnt about P2P, CDN and currently hybrid CDN–P2P. Part 3 is the body of this paper, it discusses the structure of the system and give detail protocol for our idea. Part 4 shows the scenario and experiment results. Finally, part 5 is conclusion of paper, discusses the problems and the future task.

2 Related Works

2.1 Peer-to-Peer Video Streaming

P2P live video streaming is becoming an increasingly popular technology [3, 4]. Streaming generally is a method for intelligent broadcasting of data on the mobile phone, it differs from conventional multimedia services because it isn't necessary to wait for the end of downloading video and able to start playing back. P2P live video streaming is a technology that allows a user is called broadcaster generates a video that is transmitted to other users in real-time. P2P systems solve the scalability issue by leveraging the resources of the participating peers. P2P video streaming systems have attracted a large number of Internet viewers, as demonstrated by the huge popularity of applications.

But P2P live video streaming system is facing with the dynamic join/leave of users, a peer may join or leave anytime so the structure of the P2P streaming must be able to deal with all type of dynamic changes, also mobile devices have limitations: processor capacity, bandwidth, memory, and battery.

According to P2P overlay topology, P2P streaming methods can be classified into two categories: tree-based (e.g., ChunkySpread [5]—is unstructured, using multiple trees to balance load among peers, it also reacts quickly to membership changes and scales well with low overhead) and mesh-based (e.g., CoolStreaming [6]—every peer periodically exchanges data availability information with a set of partners, and retrieves unavailable data from partners). Tree- based model has content flows from root to children along the tree, peer failures affect a complete sub-tree and long recovery time. Push approach is applied to P2P tree topology where all peers in the system form a tree structure. Parent peers push video frames to their child peers. This approach is reported to achieve lower delay. The scheduling in a tree-based streaming system includes: choose which video frame to push to child peer, choose which child peer to push selected frame based on the tree structure. Besides that, the scheduling in a mesh-based streaming system includes: choose which video frame to be request based on own buffer map, choose which peer to send frame request based on the neighbor's buffer map, resend request to partners for necessary frame. Mesh-based has peers exchanges data availability information with neighbor peers, resilient to peer failure, high control overhead, meta-data exchange consumes bandwidth, longer delay for downloading each chunk.

2.2 Content Delivery Network

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 [7, 8]. 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. There are CDN servers in either UK or Mexico, the page would load much faster for that visitor from US.

In a CDN scenario such as Akamai [9] and Limelight, the streaming content is delivered to sibling CDN servers that are placed in various geographical regions and used to reduce the overall load on the streaming source. When a client requests for the streaming content, the CDN server closest to that client will deliver the stream and not the CDN server acting as the main source of the stream. The major drawback of the CDN model is its inability to take advantage of the upload bandwidth of the clients, which effectively puts all the load onto the CDN infrastructure. Comparison between CDN and P2P is shown in Table 1.

Table 1 Comparison between CDN and P2P

Comparison item	CDN	P2P
Service capability	Service capability is limited	Service capability can grow up with peer node increases
Scalability	Expansion cost is higher	Expansion cost is lower
Reliability	High reliability	Low reliability
Stability	Good stability	Dynamic, and poor stability
Network-friendly	Friendly	Unfriendly
Orderly flow	Flow is controlled	Traffic disorder, cross
Content monitor	Can be monitored	It is difficult to monitor
User management	Centralized user management	Loose or less user management
QoS guarantee	Can be guaranteed within the maximum service capacity	Best-effort, can't be controlled

2.3 Existing Hybrid CDN and P2P System

Beside the advantages such as CDN provide excellent quality, and P2P network has better scalability and less deployment costs. However, CDN servers are expensive to deploy and maintain. While P2P is facing with the instability of dynamic peers, and the low performance when participated peers are insufficient are a bottleneck. Hybrid CDN–P2P inherits the best of both worlds: the quality control and reliability of a CDN and the inherent scalability of a P2P system.

Recently, some research papers have proposed hybrid streaming systems that combine CDN and P2P technology such as Xu et al. [10] presented one of the early analysis on hybrid CDN–P2P architectures. They perform an in-depth analysis of the system dynamics which involves a transition between a CDN mode to a P2P mode and use simulations to demonstrate that such a hybrid approach is cost-effective. Huang et al. [11] present the potential savings in using hybrid CDN–P2P systems for two major CDNs: Akamai and Limelight. Other researchers have also recommended the use of hybrid CDN–P2P systems [1]. These systems work well with VOD services. Most of this work is based on the simulation or traces collected from pure CDNs.

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.

Applying CDN–P2P hybrid techniques into live streaming and VoD streaming is different due to:

- 1st, end-to-end delay is more important for live streaming than VoD streaming.
- 2nd, a user joining an on-going live streaming session is only interested in stream starting from his/her joining time, while in VoD streaming case whole video must be delivered to new user.

By combining advantages of two system we will take advantage of both. 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.

3 Proposed System

3.1 System Design

In this paper, we design our whole system as shown in Fig. 1 with upper layer is CDN layer and lower layer is P2P layer. For detail of system, it is shown in Fig. 2.

One broadcaster: users use their own camera to capture live video and broadcast to media server following flow chart in Fig. 3.

One media server, we use HTTP live streaming (HLS) technique to encode RTMP stream into.ts file and create index.m3u8 file that manages.ts files.

One web server, we use HTTP server to run.html in Client’s Side.

One tracker server: track peer join, leave, push it into swarm.

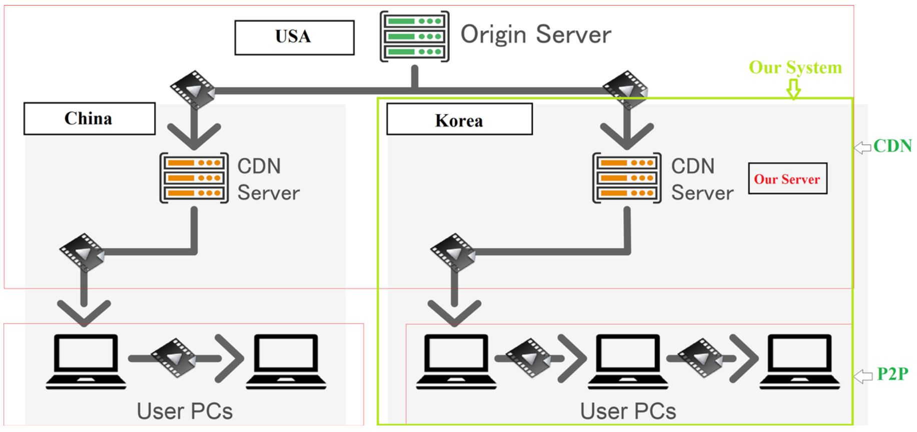


Fig. 1 CDN–P2P hybrid system

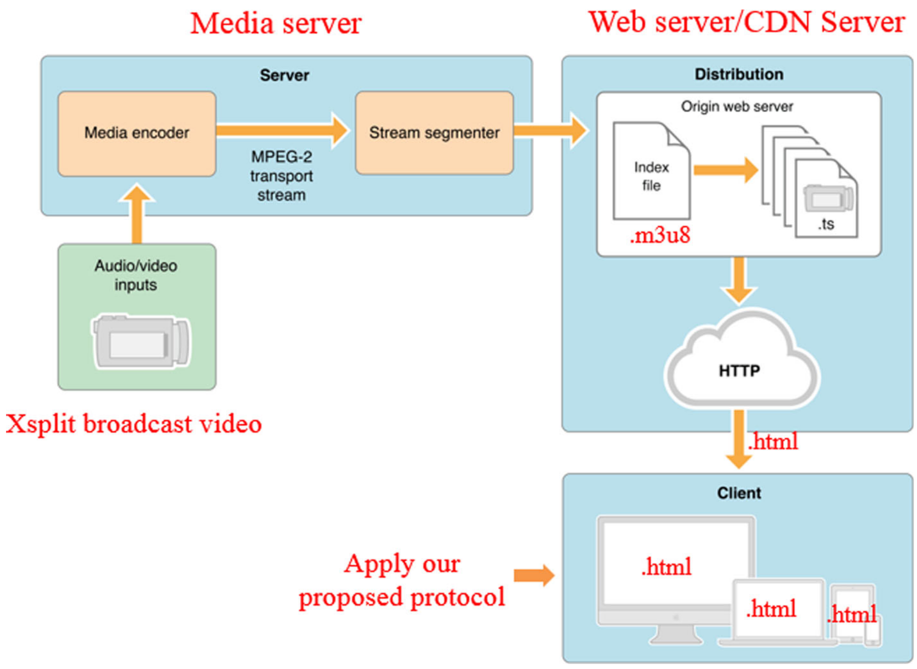


Fig. 2 Whole system architecture

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 as shown in Fig. 4: 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.

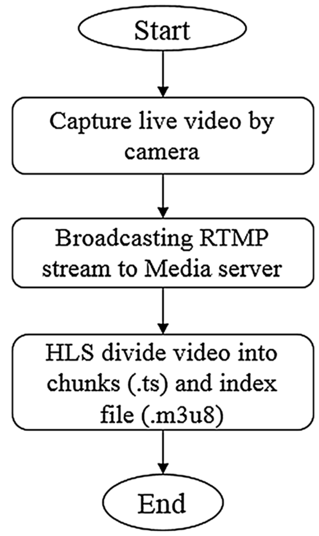

Start
Capture live video by camera
Broadcasting RTMP stream to Media server
HLS divide video into chunks (.ts) and index file (.m3u8)
End

Fig. 3 Flow chart for create live streaming video

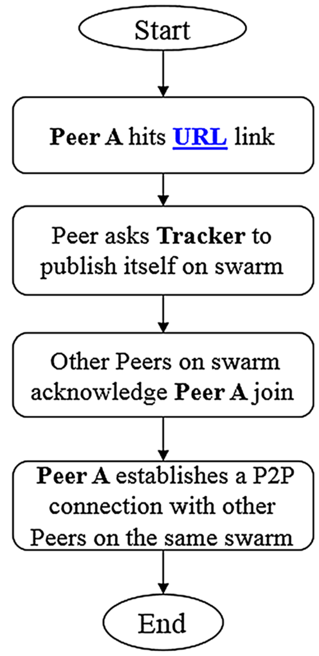

Start
Peer A hits URL link
Peer asks Tracker to publish itself on swarm
Other Peers on swarm acknowledge Peer A join
Peer A establishes a P2P connection with other Peers on the same swarm
End

Fig. 4 Peer joining procedure

3.2 Chunk Exchange Protocol

User captures live video and then broadcasting a live video on CDN server. As we proposed in our paper [12], we assume that this video is split into three segments 1.ts, 2.ts, and 3.ts as shown in Fig. 5, the first user join network and request to CDN server to receive 1.ts

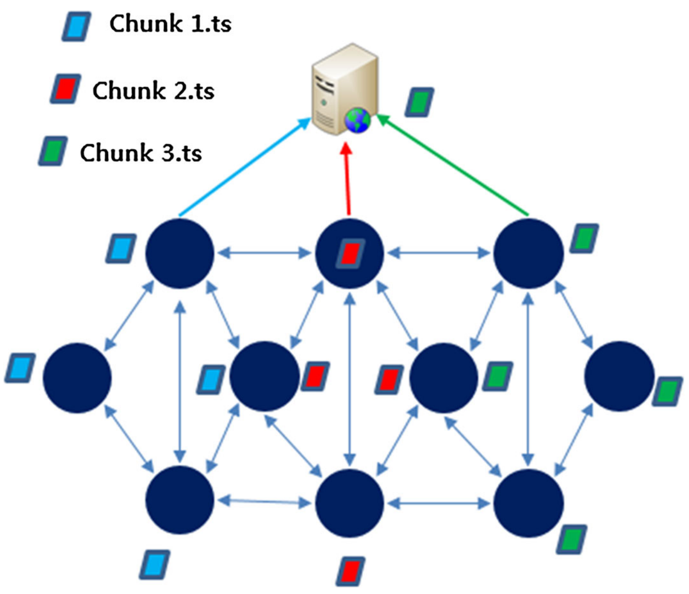


Fig. 5 Overlay hybrid construction

file. After that, it will let other peers in the network know that it has chunk 1.ts. So, other Peer will not ask CDN server to receive chunk 1.ts. However, to assure the low end-to-end delay of network, other peer will set up time out is one second. If after one second, they didn't receive chunk 1.ts from Peer 1, they will ask CDN server to receive chunk. When Peer 2 joins the network, after receiving info that Peer 1 has chunk 1.ts, it will ask CDN server for chunk 2.ts and exchange with Peer 1, other Peers as well. Depend on requirement of each system: low cost or low delay, we can set up timeout small or large.

If setup the timeout is small, that means the system will have low delay but it will have high cost because of high number of requests to CDN server.

If setup the timeout is large, that means the system will have high delay but it will have low cost because Peers in the network have enough time to exchange chunk to each other instead of sending request to CDN server (it works like VOD system).

In this section we describe a protocol used to exchange chunks between peers on the same swarm. After a Peer join the network, it starts to request video chunks. Instead of requesting chunk of video to CDN server, the peer sends a "Request" to every peers on the swarm and each peer that received the "Request" searches for the chunk in its available. If the requested chunk is available, the peer sends back an "ACK". The requesting peer sends a "Chunk" to the chosen one. Every peer that receives a "Chunk" is ensured by the previous step that the chunk is in its cache and then it sends an "Offer" with the chunk to the requesting peer. When the requesting peer sends the "Request" to swarm, it waits for a timeout of 1 s and if nobody answers, it requests directly to the CDN server to get that chunk. On Fig. 6, Peer A presents the requesting peer.

In this paper, each peer while is watching live video, it is also broadcasting that chunk to other peers on swarm. That means, its uploading and downloading at the same time.

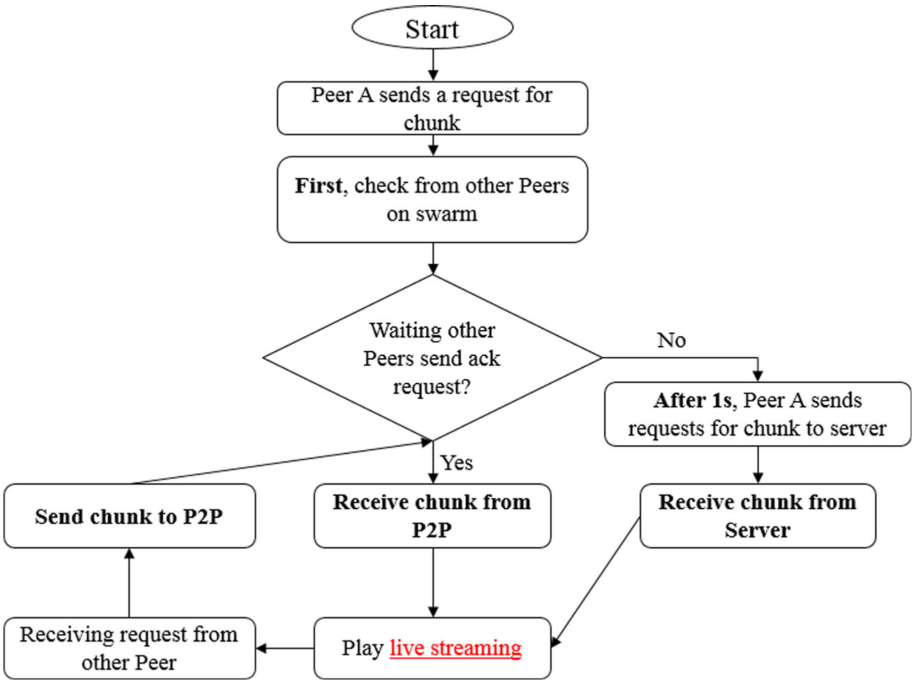

Start
Peer A sends a request for chunk
First, check from other Peers on swarm
Waiting other Peers send ack request?
No
After 1s, Peer A sends requests for chunk to server
Yes
Send chunk to P2P
Receive chunk from P2P
Receive chunk from Server
Receiving request from other Peer
Play live streaming

Fig. 6 Proposed protocol for hybrid CDN–P2P

4 Experiment Results

4.1 Set Up the Experiment

Using Xsplit Broadcaster [13] to broadcast live video: the streaming was split in chunks (.ts files and are managed by index.m3u8 file) with 6 s of duration and 650 Kbps of bit rate quality using HTTP live streaming protocol [14].

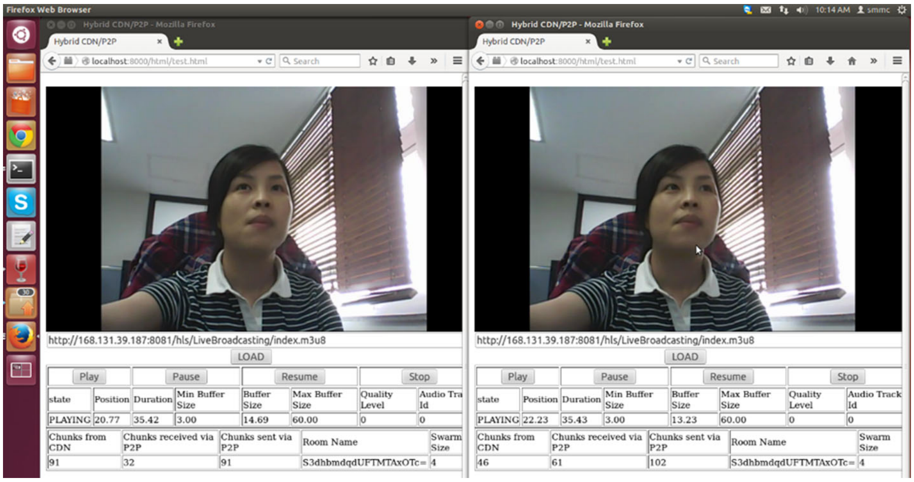
Server side using Javascript programming

- *HLS server* with IP address: 167.131.39.187:8081 to store parts of video which are being broadcasted from Xsplit.
- *HTTP server* with IP address: 167.131.39.187:8000 to run.html file.
- *Tracker server* with IP address: 167.131.39.187:3000 to manage peers and create room for streaming video and P2P connection.

Client side using HTML programming. We used a total of 10 computers running Mozilla Firefox 37.0 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 1 h of streaming using our method and then compared with 1 h of streaming using traditional method. Setting up the parameters shown in Table 2 below:

Table 2 Parameters used in the setup experiment

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



http://168.131.39.187:8081/hls/LiveBroadcasting/index.m3u8

state	Position	Duration	Min Buffer Size	Buffer Size	Max Buffer Size	Quality Level	Audio Tra Id
PLAYING	20.77	35.42	3.00	14.69	60.00	0	0

Chunks from CDN	Chunks received via P2P	Chunks sent via P2P	Room Name	Swarm Size
91	32	91	S3dhbmdqdUFTMTAxOTc=	4

http://168.131.39.187:8081/hls/LiveBroadcasting/index.m3u8

state	Position	Duration	Min Buffer Size	Buffer Size	Max Buffer Size	Quality Level	Audio Track Id
PLAYING	22.23	35.43	3.00	13.23	60.00	0	0

Chunks from CDN	Chunks received via P2P	Chunks sent via P2P	Room Name	Swarm Size
46	61	102	S3dhbmdqdUFTMTAxOTc=	4

Fig. 7 Client's interfaces

Table 3 Results of experiment with 10 users

Time (min)	Proposed method requests	Traditional CDN method requests
5	460	600
10	870	1200
15	1370	1800
20	1800	2400
25	2040	3000
30	2520	3600
35	3040	4200
40	3520	4800
45	3960	5400
50	4320	6000
55	4640	6600
60	5030	7200

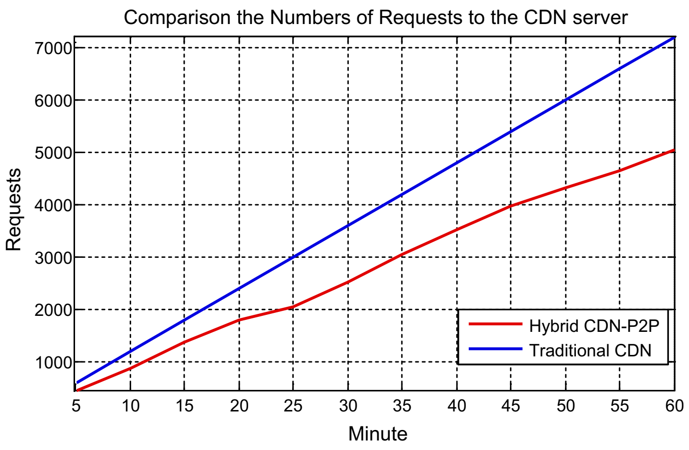


Fig. 8 Comparison the number of requests to CDN server

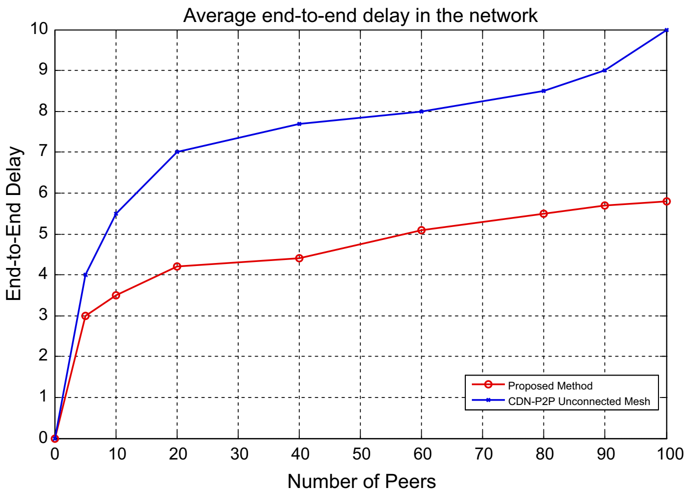


Fig. 9 Average end-to-end delay in the network

4.2 Experiment Results

As shown in Fig. 7, both browser interfaces have received chunks from CDN server and chunks from P2P, also a peer broadcast its chunks for other Peer in the same swarm.

The results have shown in Table 3 with observation requests to CDN server in the different time: after 5, 10 min, and so on. We see that the traditional CDN method total made 7200 requests to the CDN. While our method using hybrid of overlay structure total made 5030 requests, this means a reduction of 30.14 % on the total of direct requests to the CDN. This is figured out in Fig. 8.

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 [1] to proof that our system has low end-to-end delay than their system. In [1], 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 s. So, we also deploy this system to compare with our method as shown in Fig. 9. With the same number of peers in the network, for example 100 peers, end-to-end delay of our system only is 5 s compare with their system is 10 s.

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. A lot of challenges still remain. The approach used on this paper has shown effective for a small amount of peers. In future works, we should be investigated to improve method for peers' convergence and manage the swarm. From that the number of chunks transferred over the P2P overlay will be increased.

Acknowledgments This work was supported by the ICT R&D program of MSIP/IITP. [B0101-15-0429, Development of Open Screen Service Platform with Cooperative and Distributed Multiple Irregular Screens], 2015–2016.

References

1. Seyyedi, S. M. Y., & Akbari, B. (2011). Hybrid CDN–P2P architectures for live video streaming: Comparative study of connected and unconnected meshes. In *International symposium on computer networks and distributed systems*.
2. Burnett, D. C., Bergkvist, A., Jennings, C., & Narayanan, A. (2012). WebRTC 1.0: Real-time communication between browsers. *Working draft*.
3. Sanna, M., & Izquierdo, E. (2013). Live scalable video streaming on peer-to-peer overlays with network coding. *IEEE Latin America Transactions, 11*(3), 962–968.
4. Chang, H., Jamin, S., & Wang, W. (2011). Live streaming with receiver-based peer-division multiplexing. *IEEE/ACM Transactions on Networking, 19*(1), 55–68.
5. Venkataraman, V., Yoshida, K., & Francis, P. (2006). Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In *Proceedings of the IEEE ICNP*.
6. Zhang, X., Liu, J., Li, B., & Yum, T. S. P. (2005). Coolstreaming/DONet: Adata-driven overlay network for peer-to-peer live media streaming. In: *Proceedings of the INFOCOM*.
7. Baccaglini, E., Grangetto, M., Quacchio, E., & Zezza, S. (2012). A study of a hybrid CDN–P2P system over the planet lab network. In *Signal processing: image communication*.
8. Passarella, A. (2012). A survey on content-centric technologies for the current internet: CDN and P2P solutions. *Computer Communications, 35*(1), 1–32.

9. Akamai. http://www.akamai.com/html/technology/index.html.
10. Xu, D., Kulkarni, S. S., Rosenberg, C., & Chai, H. K. (2006). Analysis of a CDN–P2P hybrid architecture for cost-effective streaming media distribution. *Multimedia Systems, 11*(4), 383–399.
11. Huang, C., Wang, A., Li, J., & Ross, K. W. (2008). Understanding hybrid CDN–P2P: Why limelight needs its own red swoosh. In *NOSSDAV'08* (pp. 75–80).
12. Ha, T. T. T., Won, Y., & Kim, J. (2014). Topology and architecture design for peer to peer video live streaming system on mobile broadcasting social media. In *International conference on information science and applications (ICISA)*.
13. https://www.xsplit.com/products/broadcaster.
14. https://developer.apple.com/streaming/.

Tran Thi Thu Ha is currently a M.S candidate at Smart Mobile and Media Computing Laboratory, School of Electronics and Computer Engineering, Chonnam National University, South Korea. She received B.S degree from the School of Electronics and Telecommunications, HaNoi University of Science and Technology, Vietnam in 2011. She was a solution engineer at Asian Communication Solution, JSC in 2012. Her major interests are in the research areas of Mobile Cloud Computing, Next Generation of Mobile Platform, Mobile Operating System, and Peer-to-Peer Network.

Jinsul Kim received the B.S. degree in computer science from University of Utah, Salt Lake City, Utah, USA, in 1998, 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 2004 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 2004 to 2009. He worked as a professor in Korea Nazarene University, Chonan, 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 as IEEE Member. He has been invited for TPC(Technical Program Committee) for IWITMA2009/2010, PC(Program Chair) for ICCCT2011, IWMWT2013/2014/2015, General Chair for ICMWT2014. His research interests include QoS/QoE, Measurement/Management, Mobile-IPTV, SocialTV, Cloud Computing Multimedia Communication and Digital Media Arts.

Jiseung Nam received the B.S. degree in electronics engineering from Inha University, Korea, in 1981, and the M.S. degree in electrical engineering from University of Alabama and Ph.D degrees in electrical and computer engineering, University of Arizona, in 1986 and 1992. He worked as a researcher in Computer Communication Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea from 1992 to 1995. Currently, he is a professor in Chonnam National University, Gwangju, Korea since 1995. He has been invited as a journal editor of Korea Information Processing Society from 1998 to 2002. His research interests include Computer Network, Distributed VoD System, IPTV, Communication Protocol and Cloud Computing.