ISSN 1975-8359(Print) / ISSN 2287-4364(Online)
The Transactions of the Korean Institute of Electrical Engineers Vol. 66, No. 1, pp. 89~99, 2017
http://doi.org/10.5370/KIEE.2017.66.1.89

DHT 오버레이 네트워크에서 클라우드 보조의 P2P 라이브 비디오 스트리밍

Cloud Assisted P2P Live Video Streaming over DHT Overlay Network

림 펑 언* · 최 창 열* · 최 황 규†
(Pheng-Un Lim · Chang-Yeol Choi · Hwang-Kyu Choi)

Abstract - Many works have attempted to solve the scalability, the availability, and the low-latency problems of peer-to-peer (P2P) live video streaming; yet, the problems still remain. While tree-based systems are vulnerable to churn, the mesh-based systems suffer from high delay and overhead. The DHT-aided chunk-driven overlay (DCO) [1] tried to tackle these problems by using the distributed hash table (DHT), which structures into a mesh-based overlay to efficiently share the video segment. However, DCO fully depends on the capacity of the users' device which is small and unstable, i.e., the users' device may leave and join the network anytime, and the video server's bandwidth can be insufficient when the number of users joining the network suddenly increases. Therefore, cloud assist is introduced to overcome those problems. Cloud assist can be used to enhance the availability, the low-latency, and the scalability of the system. In this paper, the DHT is used to maintain the location of the streaming segments in a distributed manner, and the cloud server is used to assist other peers when the bandwidth which required for sharing the video segment is insufficient. The simulation results show that by using the threshold and cloud assist, the availability and the low-latency of the video segments, and the scalability of the network are greatly improved.

Key Words : Live video streaming, Cloud assist, P2P network, Distributed hash table

1. Introduction

As the speed of the internet is continuously upgraded, the demand for high quality video streaming is also becoming a standard expectation of the users. However, because of the scalability, the availability and the low-latency problems, the current live video streaming still cannot meet the user demand. Peer-to-Peer (P2P) live video streaming is one of the most popular areas which have been proposed to solve these problems.

Shen (et. al) tried to tackle this problems by using the DHT network to improve the performance of the P2P live video streaming [1]. However, the availability and the low-latency problems are still the main obstacles of this work because, the stream sharing fully depends on the capacity of the users' device. In P2P live video streaming, the clients not only receive the video segments from the servers, but also send and receive the video segments to and from other clients. Therefore, if the users' device are not stable, other users will not get the desirable quality of the video stream. Moreover, the server's bandwidth will be insufficient because the server has to sends the video segments to more clients' device as the number of users increases. Therefore, this paper proposed a cloud assisted live video streaming over DHT overlay network to improve the scalability, the availability and the low-latency of the system.

In this work, the P2P network is based on a scalable DHT network which structures into a mesh-based overlay to efficiently manage the video stream sharing. In this system, in order to maintain the availability, the low-latency and the scalability of the system, cloud assist is deployed. The cloud server joins the DHT network as a node, the same as other users' device. However, the cloud server has bigger capacity, and it is stable. The cloud server is used to assist other nodes when the number of available segments is less than the pre-defined threshold.

The threshold refers to the number of nodes which has to receive the video segment at each delay time (T_{delay}). The delay time is the allowable times that one video

† Corresponding Author : Dept. of Computer Science and Engineering, Kangwon National University, Korea.
E-mail: hkchoi@kangwon.ac.kr
* Dept. of Computer Science and Engineering, Kangwon National University, Korea.
Received : October 1, 2016; Accepted : December 9, 2016

segment will be available in the network. For example, if the delay time is set to 3 seconds, during the delay time zero (T_{delay0}) the video sever needs to send the video segment i to β_0 nodes. Then, at the first delay time (T_{delay1}=1 second), the nodes which received the video segment from the video sever need to share the video segment to other β_1 nodes. At the second delay time (T_{delay2}=2 second), all nodes which already received the video segment need to send the video segment to β_2 nodes. Finally, at the third delay time (T_{delay3}=3 second), all the received segment nodes need to disseminate the video segment to all remained nodes since it is the last allowable time for sharing the video segment i.

The threshold is also used to limit the number of nodes that the server is responsible for sending the video segment to. The server only sends the video segment to the threshold nodes. After that, the video segment will be shared among nodes in the network until every node receives the video segment. Therefore, the threshold must be carefully calculated in order to efficiently use the server capacity.

The rest of this paper is organized as follow: Section 2 will talk about the related works. Section 3 will be concentrated on the cloud-assisted live video streaming system. The performances analysis will be elaborated in Section 4. Finally, the conclusion will be presented Section 5.

2. Related Works

Many existing works have been studying on P2P live video streaming; however, most of those works focused on tree-based [2-4], mesh-based [5-7], and hybrid structure [8-10]. Only a few works have focused on using DHT [1] and cloud assist in P2P live video streaming [11-13].

DCO [1] is one of the most recent works and also maybe the first work that used DHT to improve the scalability, the availability, and the low-latency of P2P live video streaming systems. This work has a two-layer hierarchical DHT-based infrastructure where big capacity nodes form a Chord DHT in the upper tier and other nodes connect to the DHT nodes in the lower tier. The performance of this system is far better than that of mesh-based and tree-based system. However, it fully depends on the capacity of the users' device which is unstable, i.e., users join and leave the system at any time, and the video server's bandwidth will be insufficient when the number of users suddenly increases.

Cloud assist has been proposed to maintain the video quality and the scalability of the system, and also to deal with the inconsistency of the clients' device. Sweha (et. al) proposed a cloud-based service called AngelCast that enables content providers to elastically assist P2P streaming as needed [11]. By subscribing to AngelCast, a content provider can use extra resources on demand from the cloud to maintain the video stream quality. The client nodes in AngleCast are arranged in tree-based structure. Payberah (et. al) proposed Cloud-Assisted P2P Live Streaming (CLive) to tackle the bottlenecks in the available upload bandwidth at media source and inside the overlay network that can limit the quality of the server (QoS) [12]. CLive dynamically rents the cloud resources to increase the amount of total available bandwidth and the probability of receiving the video on time. CLive is based on mesh-pull P2P overlay network for video streaming using a gossip-based aggregation protocol. Cloud-Assisted Live Media Streaming (CALMS) [13] adaptively leases and adjusts cloud server resources in a fine granularity to accommodate temporal and spatial dynamics of demands from live streaming users. CALMS provides a generic framework that simplifies the migration of existing live media streaming services to a cloud assisted solution.

None of the above works uses cloud assist with the DHT network to improve the performances of the P2P live video streaming. Hence, in this paper, cloud assist and the DHT network have been used together to enhance the video quality as well as the scalability of the network. Since the DHT can be effectively used to find the location of each node in the network, we deploy DHT in the P2P live video streaming to find the location of the nodes which store the specific segment. Each node uses the DHT methods such as:

- *Get()* to find the location of node which stores the buffer map information list of the specific segment and request the buffer map information from that node.
- *Put()* to send the buffer map information to store in the buffer map information list of the specific segment.

Cloud assist is employed in this work in order to complement the video server's bandwidth as well as to supplement the required bandwidth for sharing the video segment to nodes in the network. As the number of nodes joins the DHT network increases, the video server may not have enough bandwidth to send the video segment to the threshold nodes. Thus, cloud assist can be used to complement. Moreover, because some problems such as nodes leave and join the network, some nodes fail to receive the video segment due to the network problem, and

so on, the required bandwidth for sharing the video segment among nodes in the network may be insufficient. Therefore, cloud assist can be used as the supplement.

3. Cloud Assisted Live Video Streaming System

3.1 System Overview

The cloud assisted live video streaming system is built over DHT overlay network to efficiently share the video stream among peers, as shown in Fig. 1. There are two layers in this figure. The top layer is the distributed hash table management layer. Each node in the DHT network has a buffer to catch the video segment and the buffer map to store the information of the buffer (i.e., the states of the segment in the buffer, e.g., empty or buffered states). After the video segment is already played, this segment will be deleted from the buffer map to reserve a space for new segment. Every node can be used to store the buffer map information list. The buffer map information list contains a list of buffer map information. The buffer map information consists of the buffer map, starting segment's number, and IP address of the node which stores the buffer map. Hence, each node can use this buffer map information to look for the node which stored the required segment. The bottom layer is the delivery network layer. This layer shows how the video segments are transmitted. As illustrated in this figure, the video server transmits the video segment to some nodes and to cloud server. Then, the received nodes and cloud server retransmit the segment to other peers.

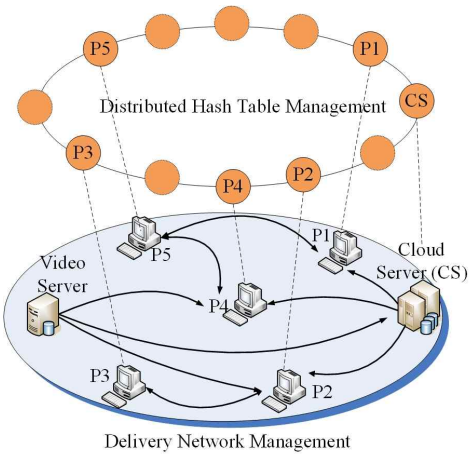

Fig. 1 The system architecture

In this system, all nodes are arranged in a mesh-based structure, and the DHT method such as *put (key, buffer map information)* and *get (key)* are used to send and obtain the buffer map information from the DHT node which is responsible for storing the buffer map information list of each segment. Moreover, the cloud servers join the DHT network as nodes the same as other users' device. While each node receives the video segments from the video server, cloud and other peers, the cloud server only receives the video segments directly from the video server. Therefore, the quality and the availability of the video segments in the cloud server can be maintained. Cloud server will assist other nodes in case the number of nodes which received the video segment is less than the pre-defined threshold. For segment i, at each delay time, cloud server periodically requests the buffer map information list of this segment from the node which stored the list. Then, cloud server counts the buffer map information in the buffer map information list. The number of the buffer map information in the buffer map information list equals the number of nodes which received the video segment because all nodes must register its buffer map information to buffer map information list after receiving the video segment. After that, if the number of buffer map information of segment i is less than the threshold, cloud assist will be used. There are two possible methods of cloud assist: *push-based assist* and *pull-based assist*.

3.2 Push-Based Assist

Push or server push technology refers to a style where the server sends the data to the clients without requiring a request from the clients. It contrasts with the pull-based technology where the client or receiver initiates the transaction request [14]. In push-based, for P2P live video streaming system, the video server or the peers, which stored the video segment, actively push the video segment to other peers when they have free upload bandwidth, where in pull-based, clients or the requested peers send the download requests to the server or other video segment's source [15].

In our work, for push-based assist method, at each delay time if the number of buffer map information in the buffer map information list of each segment is less than the threshold, cloud server randomly chooses some nodes in the DHT network to receive the video segment. The number of chosen nodes equals the threshold (β) minus the number of nodes which already received the segment ($n_{chosen} = \beta - n_{received}$). Then, cloud server directly push the video segment to the chosen nodes. After receiving the segment, those

nodes register its buffer map information to the buffer map information list to share the video segment to other nodes.

Figure 2 reveals the process of push-based assist in details. At segment 100, cloud server requests the buffer map information list from node P4. After receiving the buffer map information list, cloud server counts all the buffer map information in that list. Because the number of buffer map information is less than the threshold β, cloud server starts to send the video segment to random nodes, in this case node P1 and P5 are chosen to receive the video segment from cloud server.

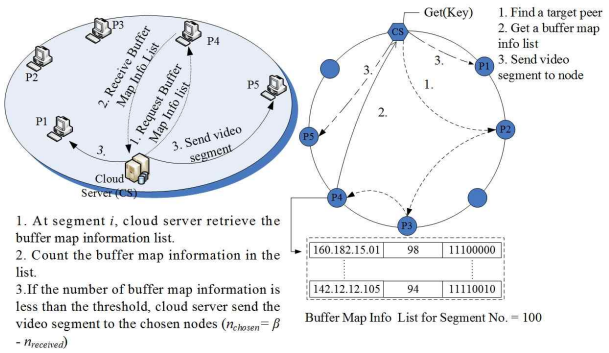

Fig. 2 Push-based Assist

Some nodes may fail to receive the video segment at each delay time because of the problems such as more nodes join the network after the segment has already been sent to the β nodes, nodes which received the video segment unexpectedly leave the network, and some nodes fail to receive the video segment due to the network problems, etc. If the number of nodes which failed to receive the video segment at each delay time represents by ϵ, push-based method needs (segment size $\times \epsilon$) of the bandwidth to send the video segment to the ϵ nodes to maintain the low-latency and the availability of the system.

3.3 Pull-Based Assist

Pull or client pull technology refers to the technology in which the initial request of the data originates from the client; and, the server or other data sources respond to that request [14]. In pull-based method, every node periodically send the notification to its neighbor of what segments it has. Then, each node requests the missing segment from its neighbor according to the notification [16]. The simplicity and the robustness are the main advantages of pull-based method.

In this work, at each delay time, if the number of nodes received the video segment is less than the threshold, cloud server generates the hash key and sends its buffer map information to store in the buffer map information list of this segment by using the generated hash key. Thus, other nodes can request the cloud server's buffer map information from the node that stores the buffer map information list of the specific segment. By using this buffer map information, those nodes can request the video segment from the cloud server. After receiving the video segment, those nodes also register their buffer map information to the network to share the segment to other nodes. Therefore, the cloud server will not be overloaded with many requests. The detail process of pull-based assist is illustrated in Fig. 3. First, cloud server requests the buffer map information list of video segment 100 from node P4. Then, cloud server counts the buffer map information in that list. Because the number of the buffer map information of this segment less than the threshold β, cloud server registers its buffer map information to the buffer map information list of the segment 100 which stores in node P4. After that, the nodes which fail to receive the video segment 100 start to request the cloud server's buffer map information from node P4. Finally, by using this buffer map information, those nodes can request the video segment 100 from cloud server since this buffer map information contains the IP Address of the cloud server.

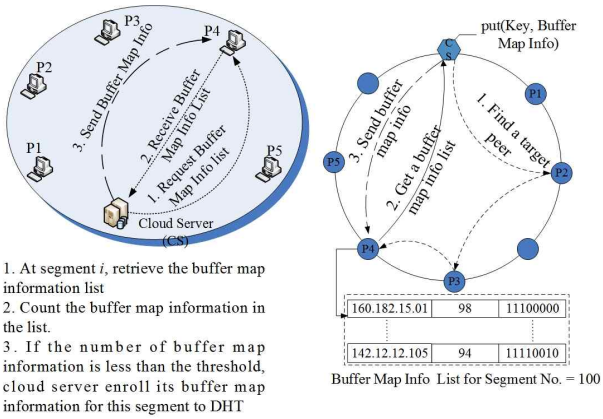

Fig. 3 Pull-based Assist

If ϵ refers to the number of nodes which failed to receive the video segment at each delay time due to the problems as mentioned above, in pull-based assist, the required bandwidth for sending the video segment to those ϵ nodes is (segment size $\times \epsilon$). In order to maintain the low-latency and the availability of the system, the cloud server has to reserve the bandwidth at least (segment size

× ϵ) for sending the segment to ϵ nodes.

3.4 System Modeling

At each delay time, cloud assist is used if the number of nodes which received the video segment i, is less than the threshold (β). The threshold (β) refers to the number of nodes which has to receive the video segment at each delay time (T_{delay}) in order to effectively share the video segment in the system during the defined delay time. The delay time is the allowable times that one video segment can be available in the network. The upper bound of the threshold should meet the following conditions:

$$n - \beta = y \tag{1}$$

$$\mu \times \beta = y \tag{2}$$

where n is the total number of nodes, y is the number of nodes which has not received the video segment, and μ is the node's capacity which can be calculated:

$$\mu = \frac{N^{TB}}{s} \tag{3}$$

where N^{TB} is the node's total bandwidth, and s is segment size.

From Eqs. (1) and (2), β can be calculated as follows:

$$\beta = \frac{n}{\mu + 1} \tag{4}$$

If the server sends the video segment to this upper bound β, during the first delay time all nodes will receive the video segment.

However, β value must be optimized in order to improve the scalability of the system as well as to reduce the required video server's bandwidth. In this case, T_{delay} can be used to optimize the value of β as shown in Fig. 4. The delay time in Fig. 4 is the time required for distributing a segment from the video sever to all peers with the help of peers' relay. If the delay time is longer, the threshold β will becomes smaller because the allowable time for sharing each segment will be longer. So that, the required video server's bandwidth will reduce. T_{delay}, in Fig. 4, refers to the maximum allowable delay time of each segment, i.e., if the T_{delay} time units pass, the segment will not be available in the network anymore as shown in Fig. 5. As illustrated in this figure, the segment 98 will be available in the network from when the first node P4 receives the video segment and last until the T_{delay} times pass, i.e., until P1 finishes playing this segment 98. Thus, after 3 seconds (T_{delay} = 3) pass, P2 will miss the segment 98.

In our work, after each node receives the video segment, it will register its buffer map information to the buffer map information list in order to share the segment with other

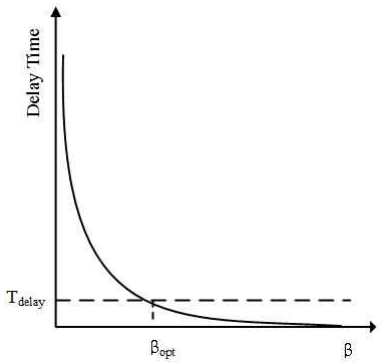

Fig. 4 Optimized threshold

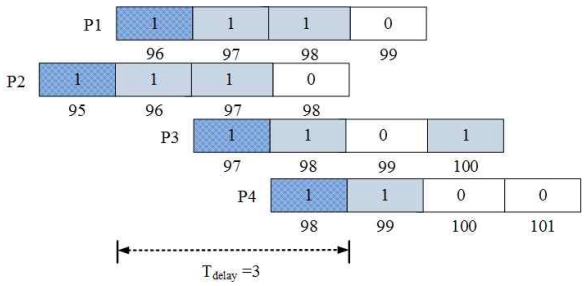

Fig. 5 The allowable delay time of segment 98

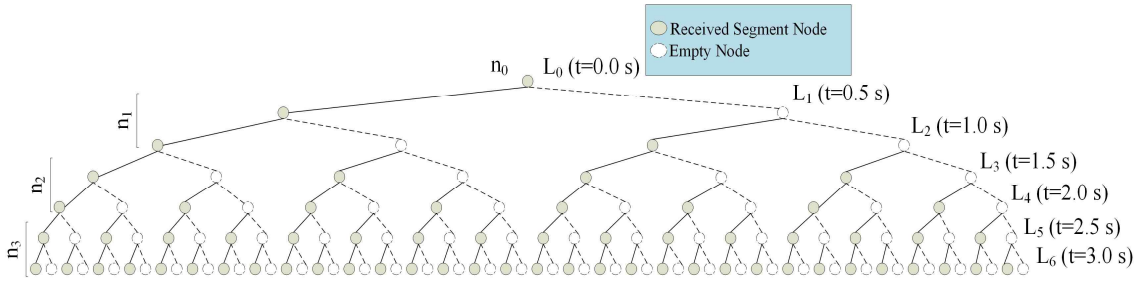

Fig. 6 Segment transmission (μ = 2, T_{delay} =3)

nodes. The number of nodes which receives the video segment from a node depends on its capacity. If μ is the capacity of the node, the segment transmission time (T_{trans}) is $\frac{1}{\mu}$ second (μ can be calculated by using Eqs. 3). For instance, if the node's capacity is 2, the segment transmission time is 0.5 second. Thus, after 0.5 second, a node receives the video segment from the source nodes. For another 0.5 second, the source node and another node, which received video segment from the source node, send the video segment to a node each. Therefore, after a second, three nodes will receive the video segment from one source node, as shown in Fig. 6. According to this figure, the total nodes n can be calculated as follows:

$$n = L_0 + L_1 + L_2 + ... + L_t, \; where \; t = \frac{T_{delay}}{T_{trans}} \tag{5}$$

Also, if $L_0 = \beta_{opt}$, we can calculated $L_0, L_1, ..., L_i$ as follows:

$$\begin{aligned} L_0 &= \beta_{opt} \\ L_1 &= 2^0 \times \beta_{opt} \\ L_2 &= 2^1 \times \beta_{opt} \\ &\vdots \\ L_t &= 2^{t-1} \times \beta_{opt} \end{aligned} \tag{6}$$

From Eqs. (5) and (6):

$$n = \beta_{opt} + \sum_{t=1}^{\frac{T_{delay}}{T_{trans}}} (2^{t-1} \times \beta_{opt}) \tag{7}$$

Because $T_{trans} = \frac{1}{\mu}$:

$$\beta_{opt} = \frac{n}{1 + \sum_{t=1}^{T_{delay} \times \mu} 2^{t-1}} \tag{8}$$

Depending on the node's capacity μ, the number of nodes receives video segment at each delay time can be calculated as follows:

$$\begin{aligned} n_0 &= L_0 \\ n_i &= L_j + L_{j+1} + L_{j+2} + ... + L_{j+k}, \end{aligned} \tag{9}$$

where $j = (\mu \times (i-1)) + 1$, and $k = \mu - 1$

For each segment i, cloud server requests the buffer map information list from the node which is responsible for storing this list. Then, cloud server uses this list to count the number of the buffered segment of the segment i. If α is the segment status (α ={0,1}, that 0 indicates the empty state, 1 indicates the buffered state of the segment), the function of the segment states can be represented by $f(\alpha) = \alpha$. The empty or the buffered states of the segment can be illustrated as the following function:

$$f(\alpha) = \begin{cases} 1 & \text{if } \alpha > 0 \\ 0 & otherwise \end{cases} \tag{10}$$

$f(\alpha)$ equals 1 in case the buffer's slot has already stored the segment, and equals 0 in case the buffer's slot is empty.

And if θ is the number of nodes which received the segments:

$$\theta = \sum f(\alpha), \forall f(\alpha) > 0 \tag{11}$$

If the number of nodes which received the video segment i is less than the threshold ($\theta < \beta$), cloud assist is used. The number of the nodes which received the video segment i can be less than the threshold in cases:

- Nodes leave and join the network after the segment i is already sent to the β nodes.
- The nodes which have already received the video segment unexpectedly leave the network.
- Some nodes failed to receive the video segment due to the network problems, etc.

If we assume that ϵ are the number of nodes which cannot receive the segment due to the above problems, we need to reserve cloud's bandwidth $C\epsilon$ ($C\epsilon = segment\ size \times \epsilon$) to assist the peers at each delay time. We can used the video streaming history to predict the $C\epsilon$.

Moreover, cloud assist also can be used to improve the scalability of the system. As shown in Fig. 7, when the server reaches its maximum capacity, cloud assist can be used to complement the server's capacity. As the cloud server can be hired on demand, during the peak period, the scalability of the system can be improved by hiring bigger cloud capacity, and vice versa.

If B_T, B_S, and B_C are the total bandwidth, the bandwidth of video server, and the bandwidth of cloud server, respectively, which required for sending video segment to β_0 (the β at T_{delay0}) nodes:

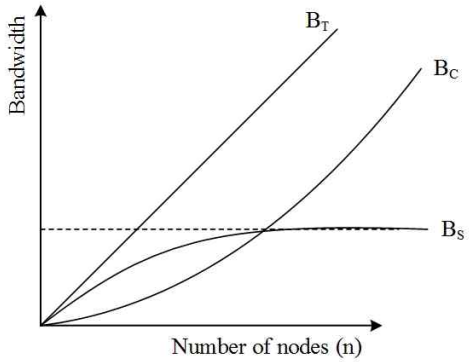

Fig. 7 Bandwidth of cloud and video server require for sending video segment to β_0 nodes

$$B_T = B_S + B_C \tag{12}$$

Also, the scalability of the system can be improved by upgrading the video server capacity or using CDNs instead of cloud server. However, according to [13] using cloud server is the best method to improve the performances of the system with the reasonable cost if compare to CDNs. The performances of CDN is comparable to cloud server but the cost is more expensive. Using cloud server can save up to 39.6 percent if compare to CDN. Moreover, for live video streaming, during the peak period when there are millions of devices watch the video stream at the same time, is only occurred occasionally such as during Olympic Game, World Cup, etc. If the video server's capacity is upgraded to meet the users demand during these peak periods, there is a waste of resources during the normal period which only a small amount of its capacity is used.

Table 1 β_{opt} and the required bandwidth

Number of nodes (n)	Delay time (3 seconds)	
	β_{opt}	Bandwidth (KB)
1000	16	4800
2000	32	9600
3000	47	14100
4000	63	18900
5000	79	23700
6000	94	28200
7000	110	33000
8000	125	37500
9000	141	42300

* The β_{opt} and the require bandwidth are calculated by using Eqs. 8. The value of μ and video segment size are the same as mention in experiment setup.

If C is the total cloud bandwidth which is used to allow all clients stream the video smoothly, $C\epsilon$ is a cloud bandwidth which is used to send video segment to the number of nodes which failed to receive the segment due to some problems as mention above, C_β is the cloud bandwidth which is used to complement the video server in case the video server does not have enough bandwidth to send the video segment to β_0 nodes:

$$C = C_\epsilon + C_\beta \tag{13}$$

where $C_\beta = 0$ in case video server has enough bandwidth to send the video segment to the threshold nodes.

4. Performance Evaluation

In order to evaluate the system performance, a simulation program has been developed. In our experiment, we set the upload bandwidth of the server and cloud to 40,000 Kilobits per Second (Kb/s) and the upload and the download bandwidth of each node to 600 Kb/s. However, the upload bandwidth of the server and cloud can be changed in some experiments. The video size is set to 120 segments and segment size is 300 Kb, that one segment can be played for 1 second. Thus, the video length is 2 minutes. The buffer size is set to 10. After the upload bandwidth of the clouds and the nodes which received the video segment are full, the requested nodes are queued and will request the video segment again after the bandwidth is available. The number of nodes which joins and leaves the network are set to the exponential distribution with mean value of 2 percent and 1 percent of total nodes every half second, respectively. Nodes which leave the network are randomly chosen so in some cases it can be the nodes which have already received the video segment. The delay time (T_{delay}) is set to 3 seconds, i.e., all nodes need to receive each segment during 3 seconds. We run each experiment five times and report the average as the final experimental results.

4.1 Latency

Figure 8 demonstrates the average delay time of the segments while the number of nodes is increased. As illustrated in this figure, push-based and pull-based methods are better than no-cloud assist method. In case there is no cloud assist, the delay time of each segment

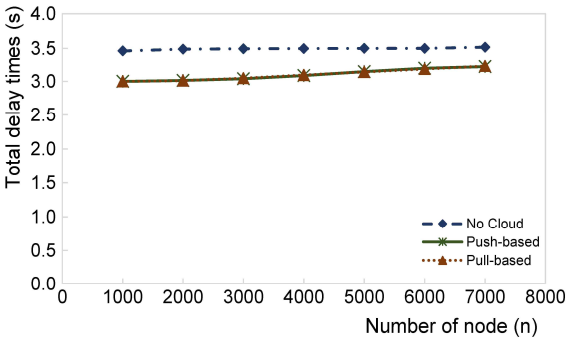

Fig. 8 The delay times versus the number of nodes

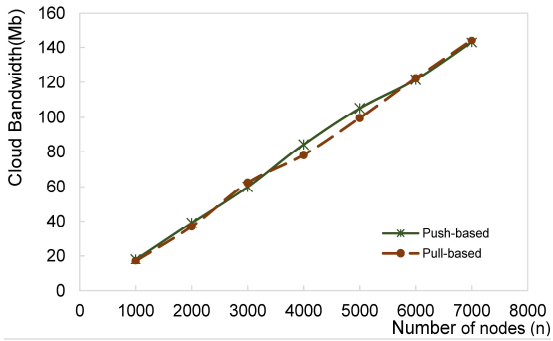

Fig. 9 Cloud bandwidth using in push-based and pull-based methods

exceed 3 seconds. As shown in Table 1, the video server has enough bandwidth to send the video segment to β_0 nodes even when the number of nodes reach 8000 with the delay time 3 seconds. However, the simulation result shows that even when the number of nodes is less than 8000, the average delay time of each segment still exceed 3 seconds when no-cloud method is used. It is because some problems such as nodes leave and join the network after the video segment has already been sent to the β nodes, etc., that lead to the number of nodes which receives the video segment less than the threshold. The delay time of pull-based and push-based methods are slightly different due to the differences in the number of nodes leave and join the network. Because cloud does not have enough bandwidth, the delay time of both methods exceed 3 seconds when the number of nodes reach 3000. The delay time of both methods will not over 3 seconds if cloud server has enough bandwidth.

Figure 9 shows the amount of cloud bandwidth for both methods (push-based and pull-based) which is needed for maintaining the delay time of the network, i.e., if this

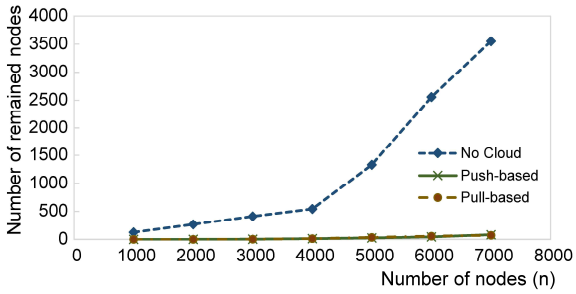

Fig. 10 Number of remained nodes

amount of cloud bandwidth is used for each method, the delay time will not exceed 3 seconds. In this experiment, clouds capacity is set to big enough for sending the video segment to all nodes which failed to receive the video segment. As shown in Fig. 9, the bandwidth of pull-based and push-based methods are marginally different. Both methods have to reserve bandwidth at least (segment size × ϵ) Kb for sending the video segment to nodes which failed to receive the segment. The ϵ refers to the number of nodes which failed to receive the video segment due to the problems such as nodes leave and join the network after the segment is already sent to the β nodes etc. However, push-based method is slightly better than pull-based because, in push-based, cloud server directly send the video segment to the missing segment node without registering its buffer map information to the buffer map information list. To register the buffer map information to the buffer map information list, a DHT search is needed. Also, when each missing segment node requests the segment from the cloud server, that node needs to use a DHT search to search for the cloud server's buffer map information before sending the request to the cloud server. Therefore, by using push-based technique, the number of DHT searches can be reduced.

4.2. Availability

Figure 10 shows the number of remained nodes when the number of nodes in the network is increased. The remained nodes refer to the nodes which do not receive the video segment after T_{delay} time (T_{delay} is set to 3 seconds in this experiment) pass. The number of remained node of no-cloud method is worse than other methods, while the remained nodes of push-based and pull-based methods are comparable. When the number of nodes reaches 3000,

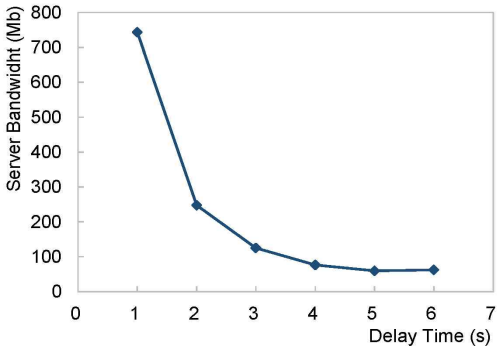

Fig. 11 Amount of the required server bandwidth when the delay time is increased

pull-based and push-based methods have some remained nodes due to the lack of cloud's bandwidth. If cloud has enough bandwidth, all nodes will receive the video segment during 3 seconds. Even the delay time, as shown in Fig. 8, of each method is insignificantly different, the number of remained nodes of no-cloud assist is more than that of other methods. For instance, when the number of nodes reaches 3000, the delay time of no-cloud method is 3.49, and the delay time of push-based and pull-based methods are around 3.05. However, according to Figure 10, for no-cloud assist, after delay time 3 (after 3 second) pass, the number of remained nodes is 413 comparing to the remained nodes of pull-based and push-based methods are only around 5.

4.3 Scalability

Figure 11 demonstrates the required servers' bandwidth when the delay time is increased. In this experiment, the number of nodes is set to 6000 and the bandwidth of the server and cloud are set to big enough to efficiently send the video segment to all nodes without exceed the defined delay time. As illustrates in Fig. 11, when the delay time is increased, the server bandwidth, which required for sending video segment to β_0 nodes, is decreased. It is because, as mention in Sect. 4.4, the threshold β depend on the delay time T_{delay}. The bigger the T_{delay}, the smaller the threshold β. Thus, in order to maintain the video quality and the scalability of the system with the effective cost, the delay time must be carefully chosen. The server bandwidth here refers to the bandwidth of both the video server and the cloud server.

As demonstrated in Fig. 12, cloud server also help to maintain the scalability of the system. In case the video

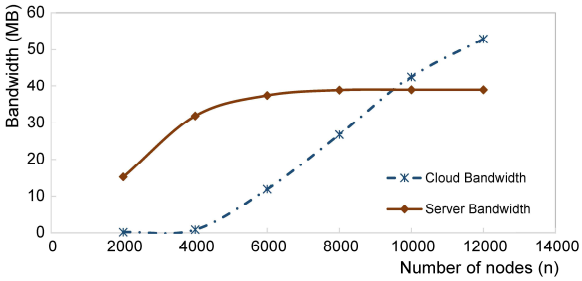

Fig. 12 Cloud bandwidth and server bandwidth required for sending the video segment to β0 nodes

server does not have enough bandwidth to send the video segment to all β_0 nodes, cloud's bandwidth can be used to complement. As the number of nodes increases, the threshold also increase. Thus, the video server may not have enough bandwidth to send the segment to all threshold nodes. When the video server's capacity reaches its peak, the cloud server can be used to complement. According to Fig. 12, when the number of nodes reaches 9000, server does not have enough bandwidth to send the video segment to all β_0 nodes. Thus, the cloud's bandwidth is used for both, to assist peers in case the bandwidth required for sharing the video segment is inadequate, and to complement the video server in case the video server does not have enough bandwidth to send the segment to the β_0 nodes. Because the cloud server can be dynamically hired, during the peak period, the scalability of the system can be improved by hiring bigger cloud capacity, and it can be reduced when the number of users decrease.

5. Conclusion

In this paper, cloud assisted live video streaming over DHT overlay network has been proposed. Cloud assist is introduced to improve the availability, the low-latency and the scalability of the system. Because cloud server is consistent, has dynamic capacity, and receives the video segments directly from the video server, the availability, the low-latency and the scalability of the system can be maintained.

The performance evaluation results show that with the use of the cloud assist, the performances of the system are greatly improved. If the cloud server has enough bandwidth, the low-latency and the availability of the video segment can be guaranteed. Moreover, cloud server can be used to improve the scalability of the system. In case the video

server's capacity reaches its peak and does not have enough bandwidth to send the video segment to β_0 nodes, cloud server can be used as a complement.

In the future, in order to effectively share the video segment among peers, the users' location is also needed to study in details.

Acknowledgement

This study was supported by 2016 Research Grant from Kangwon National University(No. D1000648-01-01)

References

[1] H. Shen, L. Zhao, Z. Li, and J. Li, "A DHT-aided chunk-driven overlay for scalable and efficient peer-to-peer live streaming," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no.11, 2013, pp. 2125－2137.

[2] J. Jannotti, D.K. Gifford, K.L. Johnson, M.F. Kaashoek, and J.M. OToole, "Overcast: reliable multicasting with an overlay network," Proc. Fourth Conf. Operating System Design and Implementation (OSDI), 2000.

[3] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," Proc. 19th ACM Symp. Operating Systems Principles (SOSP), 2003.

[4] T. Kouchi, S. Fujita, "Maintaining Tree-Structured P2P Overlay Being Resilient to Simultaneous Leave of Several Peers," IEICE Trans. Information and Systems, vol. E98-D, no. 9, September 2015, pp. 1667－1674.

[5] W.P.K. Yiu, X. Jin, and S.H.G. Chan, "VMesh: distributed segment storage for peer-to-peer interactive video streaming," IEEE J. Selected Areas in Comm., vol. 25, no. 9, 2007, pp. 1717－1731.

[6] X. Zhang, J. Liu, B. Li, and T.P. Yum, "Cool Streaming-/DONet: a data-driven overlay network for peer-to-peer live media streaming," Proc. IEEE INFOCOM, 2005, pp. 2102－2111.

[7] M. K. Bideh, B. Akbari, A. G. Sheshjavani, "Adaptive content-and-deadline aware chunk scheduling in mesh-based P2P video streaming," Peer-to-Peer Netw. Appl., 2016, pp. 436-448.

[8] F. Wang, J. Liu, and Y. Xiong, "Stable peers: existence, importance, and application in peer-to-peer live video streaming," Proc. IEEE INFOCOM, 2008.

[9] Y. Liu, "Delay bounds of chunk-based peer-to-peer video streaming," IEEE/ACM Trans. Networking, vol. 18, no. 4, 2010, pp. 1195－1206.

[10] K. Pal, M.C. Govil, M. Ahmed, "A new hybrid approach for overlay construction in P2P live streaming," IEEE Intl Conf. ICACCI, 2015.

[11] R. Sweha, V. Ishakian, and A. Bestavros, "AngelCast: cloud-based peer-assisted live streaming using optimized multi-tree construction," Proceedings of the 3rd Multimedia Systems Conference, MMSys12, 2012, pp. 191－202.

[12] A. H. Payberah, H. Kavalionak, V. Kumaresan, A. Montresor, and S. Haridi, "CLive: cloud-assisted P2P live streaming," 12th IEEE Intl Conf. Peer-to-Peer Computing (P2P), 2012, pp. 79－90.

[13] F. Wang, J. Liu, and M. Chen, "CALMS: cloud-assisted live media streaming for globalized demands with time/region diversities," IEEE Infocom, 2012, pp. 199－207.

[14] S. Acharya, M. J. Franklin and S. B. Zdonik, Balancing Push and Pull for Data Broadcast, Procs. of the ACM SIGMOD Conference, May 1997.

[15] F. Picconi, L. Massoulie, "Is there a future for mesh-based live video streaming?," Eighth Intl Conf. on Peer-to-Peer Computing, 2008, pp. 289－298.

[16] M. Zhange, Q. Zhang, L. Sun, S. Yang, "Understanding the Power of Pull-based Streaming Protocol: Can We Do Better?," IEEE J Sel Areas Commun, 2007, pp. 1678－1694.

저 자 소 개

림 평 언 (Pheng-Un Lim)

2009: B.S. in Department of Computer Science, Royal University of Phnom Penh, Phnom Penh, Cambodia. 2011-Present: Pursuing M.S. and Ph.D. in Department of Computer Science and Engineering, Kangwon National University, Chuncheon, Korea. Research Interests: Multimedia system, P2P overlay network, and cloud computing.

E-mail : limphengun@kangwon.ac.kr

최 창 열 (Chang-Yeol Choi)

1979: B.S. in School of Electronics Engineering, Kyungpook National University, Korea. 1981: M.S. in School of Electronics Engineering, Kyungpook National University, Korea. 1995: Ph.D. in Department of Computer Engineering, Seoul National University, Korea. 1984~1996: ETRI, A Principal Researcher at Computer Research Lab. Present: Professor with Department of Computer Science and Engineering, Kangwon National University, Korea. Research Interests: Computer System Architecture, Embedded System, Cloud Computing
E-mail : cychoi@kangwon.ac.kr

최 황 규 (Hwang-Kyu Choi)

1984: M.S. in Department of Electric and Electrical Engineering, KAIST, Seoul, Korea. 1989: Ph.D. in Department of Electric and Electrical Engineering, KAIST, Seoul, Korea. Present: Professor with Department of Computer Science and Engineering, Kangwon National University, Chuncheon, Korea. Research Interests: Multimedia system with the architecture and algorithms of streaming system on wired or wireless, P2P overlay network, and cloud computing.
E-mail : hkchoi@kangwon.ac.kr