Journal of Knowledge Information Technology and Systems

ISSN 1975-7700

http://www.kkits.or.kr

An Efficient Architecture by Organizing Streaming and Storage Users Position in P2P Network for Sharing Live Video

Tran Thi Thu Ha[1], Byeongok Jang[*2]

[1]*School of Electronics& Computer Engineering, Chonnam National University*
[2]*Dept. of Digital Content, Korea Nazarene University*

ABSTRACT

Peer-to-Peer (P2P) streaming is a method for sharing data on the network without the support of server. One of the main challenge is to provide a high quality of service through the dynamic behavior of the network because a user may join or leave anytime. We realized that the streaming users- who use mobile devices with 3G/4G connection, may stop watching live video after a while if they find the video is out of their interest. Users leaving causes dynamic and affect the data delivery. On the other hand, the storage users-who use PC with wired Internet, are downloading the video do not have the concern of interest and playback quality, until they start to watch the video. Hence, the storage users are relatively more stable than streaming users. This paper, we propose an efficient architecture by organizing storage users position are closer to the broadcaster than streaming users position. And then, we apply our idea on hybrid model that combines the benefits of tree-based and mesh-based models to decrease startup delay, packet lost and control overhead compare with other systems.

KEYWORDS: peer-to-peer, live video streaming, file sharing, chunks, tree-based, mesh-based

ARTICLE INFO: Received 26 January 2015, Revised 13 February 2015, Accepted 13 February 2015.

1. Introduction

P2P live video streaming is becoming an increasingly popular technology with many researches [1, 2, 3] to improve video quality also balance performance and fairness in P2P live video systems. Current approaches in P2P video streaming can be classified as tree-based,

[*]Corresponding author is with Dept. of Digital Content, Korea Nazarene University, 331-718 Republic of Korea.
E-mail address: bojang@kornu.ac.kr

mesh-based. With tree-based model uses a push method to transfer data [4]. This model has low start-up delay. However, there are two main problems in this method: if the bandwidth of parent node is low, children nodes will be lose data and when parent node failure, other nodes can't receive data until completing the recovery of the tree. On the other hand, mesh-based model uses a pull method to request necessary data from a number of neighbor nodes [5]. However, mesh-based model requires large buffers to support pull data from neighbors and there is an adjustment between minimum delay by sending pull request and overhead of whole system. So, both models have their own strengths and weaknesses. The existence of the two types of users for video streaming - there are wired Internet users and wireless mobile users [6, 7]. Therefore, this paper proposes a new architecture system design for P2P live video streaming with storage users are next to the broadcaster, because they are more stable than streaming users which can leave system anytime. And also we design topology which combines the advantages of pull and push methods. The remainder of this paper is organized as follows. We briefly discuss the related work in Section 2. The formulation for topology optimization in tree/mesh-based P2P streaming systems, simulation setting and numerical results are presented in Section 3. Finally, the paper is concluded in Section 4.

2. Related Works

2.1 Tree-based and Mesh-based Systems

Mesh-based topology uses pull-based scheduling as show in <Figure 1> (right side). Choosing which video frame to be request based on own buffer map, choose which peer to send chunks request based on the neighbour's buffer map, resend request to partners for necessary chunks. A mesh-based system is therefore more potent, but longer delays and higher control overhead. Compared with mesh-based systems, the tree-based systems as show in <Figure 1> (left side) have well-organized overlay structures and typically distribute video by actively pushing data from a peer to its children peers. However, maintaining the tree overlay with node churns is a difficult task.

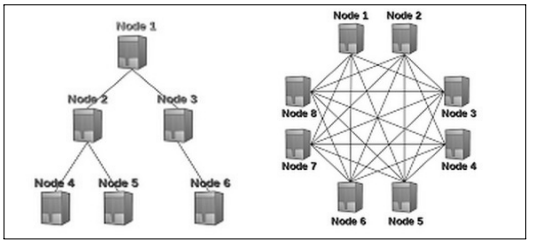

그림 1. 트리기반(좌)과 메시기반(우) 모델

Figure 1. Tree-based (left) and Mesh-based (right) models

We make a survey about the overlay of streaming video systems and comparison these systems with metrics as shown in Table I.

Overcast [8] constructed and maintained a high bandwidth distribution tree from a source to multiple nodes. So, it utilizes a tree algorithm to maximize the bandwidth from the root towards each node. CoolStreaming [9] is a mesh-based overlay. The video stream is segmented into chunks and is broadcasted to neighbors.

Comparing to a simple tree-based overlay, observe that playback continuity is much better. Hence, some mixed strategies have also been proposed to exploit the advantages of both schemes. Prime [10] presented topology for mesh-based. Prime is designed to work with MDC to minimize content bottlenecks and bandwidth bottlenecks.

표 1. 트리기반과 메시기반 모델 비교

Table 1. Comparison between tree-based and mesh-based models

Metrics	Tree-based	Mesh-based
Construction & Maintenance	Easy to construct but difficult to maintain	Easy to maintain
End-to-End Delay	Low	High
Start-up Delay	Low	High
Overhead	Low	High
Bandwidth Utilization	Cannot utilize children nodes' bandwidth	Good but depends on peer selection

3. System Design and Implementation

3.1 System Design

Our proposed system combines low overhead of tree-based system with the stability of a mesh-based system. The source node divides the video stream into several sub-trees, with each chunk being sent to multiple sub-trees and also designs for stable nodes close to the source, and dynamic nodes far away from the source. In hybrid pull-push scheduling is proposed to enable push in a mesh-based system to lower the system overhead.

The tree length can be too high if there are a large number of nodes. We expect a topology, in which both the length and the number of nodes in each level grow linearly. If level 1 has 2^k nodes, then the number of nodes at level i has 2^{k-i}. Thus, total number of nodes from level 1 to level i is:

$$N_i = \prod_{j=1}^{i} 2^{k-j+1}$$

Also, the total number of nodes in the network can be calculated as:

$$N = \sum_{i=1}^{k} N_i$$

Then, from the number of users join to P2P network, we can calculate both the number of levels and number of node in each level.

Number of chunks which will be pushed into each sub-stream given by equation:

$$Number\_of\_Chunks = \frac{Files \times File\_size}{Chunk\_size}$$

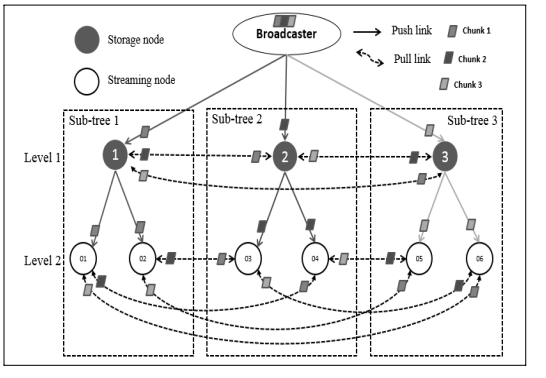

그림 2. 오버레이로 구성된 전체 시스템

Figure 2. Overlay construction of whole system

The scenario we design the storage nodes are nearby to the broadcaster. <Figure 2> shows the organization of two types of nodes (storage node and streaming node) in the overlay. For this case, we can split overlay tree into three sub-trees. User are organized in separate sub-trees, except the broadcaster. That means a node belongs to only one sub-tree. Each node also has links to other nodes of other sub-trees. Each node maintains two kinds of connections: connection belonging to a sub-tree (push link- dash line) and connections of nodes in different sub-trees (pull link- dot line as seen in <Figure 2>).

At the pushing phase (dash line), broadcaster pushes three chunks to nodes at level 1. These nodes continue to push these parts to other child nodes in their sub-trees. As the result, through the pushing phase, nodes have minimum data required to display.

At the pulling phase (dot line), a node will send pull request to other node with same level to receive remain parts of video. The user requires pulling requests of other users which belong to other sub-trees. If this user did not respond pulling requests of other users, it will inform other users about its rejection. These users must look up other users that can send data to them. If it does not send data to any user in a sub-tree, it may not able to pull data from any user in other sub-tree. So, streaming data is distributed to every node in the overlay network by both pushing and pulling methods. The processes shown in <Figure 3>, once a peer joins, the application creates the overlay and starts streaming. If the node leaves, the application is stopped. When a node joins in P2P network and connects to the tracker. The tracker replies with information on the chosen sub-stream and the initial neighborhood for the peer.

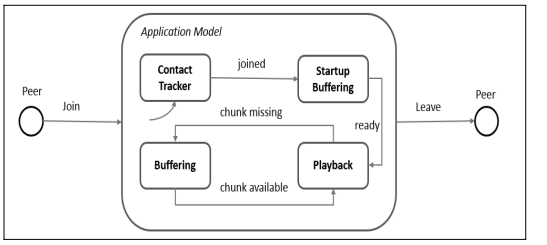

그림 3. 네트워크 내 피어 라이프 사이클

Figure 3. The life cycle of peer in network

This information consists of the current playback position of the source in terms of chunks as well as information about the levels that are available for the given chunk. Based on the information, the peer selects the level depend on the configuration of the peer: streaming user or storage user. After selecting the level and sub-stream, the peer starts to download matching chunks into the buffer. If a chunk is not available in the buffer at the time it is supposed to be played, the user wait until the chunk has been successfully downloaded. This process continues until the peer leaves. This helps in preventing undesired effects, such as dynamic behavior of the network, other peers can pull missing data from another peers in same level instead of pulling data from peer which dropt out.

3.2 Proposed Algorithm

Consider that a local network consisting of 3 users: Broadcaster captures live video and then broadcasting this video on P2P network. We assume that this video is split into three chunks c_1, c_2, and c_3. Broadcaster pushes three chunks to users 1, 2, and 3 with respectively. After finishing receiving each chunk, then users 1, 2, or 3 send requests to each other for remain chunks of video using pull requests. For example, user 1 receives chunk c_1 from broadcaster and then it sends pull request to user 2 and 3 to get chunk c_2 and c_3 of video. Now, user 1 is watching a chunk c_1 which receives from broadcaster and also can broadcast chunk c_1 for users 2, 3 if they request. At the same time, user 1 is also getting chunk c_2, c_3 from users 2, 3 and continues watching whole live video. We can extend this scenarios with many users, building tree overlay with a number of chunks. In each sub-tree, push method is used like the case broadcaster pushes data to users 1, 2 and 3. Besides that, in each level, pull method is used like the case user 1 pull data from user 2 and 3.

3.2 Implementation and Results

Topology is generated by using Geogia Tech Internetwork Topology Models (GT-ITM) generator of ns-2 running in Ubuntu environment to simulate P2P content distribution following our topology. In evaluation, we use the typical metrics: startup delay, packet lost, control overhead as shown in <Figure 4> to compare our solution with Overcast [1] (pure tree-based) and CoolStreaming [3] (pure mesh-based) overlays.

```
BEGIN {
      sends=0;
      recvs=0;
      routing_packets=0.0;
      droppedBytes=0;
      droppedPackets=0;
      highest_packet_id =0;
      sum=0;
      recvnum=0;
    }
    {
  time = $3;
  packet_id = $41;


# CALCULATE START UP DELAY
  if ( start_time[packet_id] == 0 )  start_time[packet_id] = time;
  if (( $1 == "r") &&  ( $35 == "cbr" ) && ( $19=="AGT" )) {  end_time[packet_id] = time;}
       else {  end_time[packet_id] = -1;  }

# CALCULATE PACKET DELIVERY
  if (( $1 == "s") &&  ( $35 == "cbr" ) && ( $19=="AGT" )) {  sends++; }
  if (( $1 == "r") &&  ( $35 == "cbr" ) && ( $19=="AGT" ))   {  recvs++; }

# CALCULATE CONTROL OVERHEAD
  if (($1 == "s" || $1 == "f") && $19 == "RTR" && $35 =="DSR") routing_packets++;
```

그림 4.네트워크 메트릭 계산을 위한 로깅 설정

Figure 4. Logging settings to calculate network metrics

Startup delay: is the time taken by a node between its request of joining the overlay and receiving first chunk to start playing back.

For the beginning, the mesh-based solution (CoolStreaming) performs worst, because it needs a longer time to search and request for neighbor peers. The Overcast performance took only 20 seconds to startup. Even though, our solution need to aware of the coexistence of the two types of nodes and explicitly prioritizes the service to the streaming nodes but at the beginning our topology constructs as tree overlay network. So, nodes need shorter time to startup than the CoolStreaming and almost similar with Overcast startup time.

Packet lost: is the total number of packets dropped during the simulation. It can be calculated by equation: Packet lost = Number of packets sent − Number of packets received.

CoolStreaming with the mesh-based solution performs the best, because it is pull-based and

thus is resilient to the node dynamics. On the other hand, Overcast with the pure tree-based solution performs the worst, because the tree overlay is interrupted to suffer from the node dynamics. Our solution combines pull-based and push-based, it takes advantage of both solutions, so it performs much better than the pure tree-based solution and little less than mesh-based solution. By using hybrid of overlay structure and combine design storage nodes are close broadcaster, our solution can achieve the performance of the mesh-based solution.

Control overhead: is size of the control messages which are sent from nodes to nodes. CoolStreaming costs much higher overhead than other systems because so many messages are sent between nodes by using pull method. Our proposed system has almost the same overhead with pure tree-based of Overcast because nodes only send messages to get correct missing chunks from neighbors.

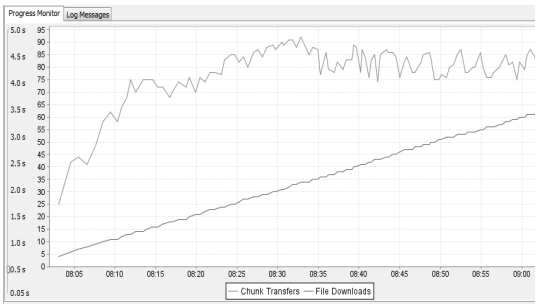

그림 5. 전체파일과 청크 간 전송처리 시산 비교

Figure 5. Comparison of processing time for transferring whole file and chunk file

With using new topology, processing time for receive complete video is much faster than traditional transfer methods using pure mesh-based or pure tree-based because our method separate video into number of chunk files. As shown in <Figure 5> is comparison processing time for transferring chunk files in our proposed method with transfer whole video in traditional method.

4. Conclusion

This paper, we recognized existence of the two types of users for video streaming: wired Internet users and wireless mobile users. Specifically, Internet users have better network connection, and thus they should be considered to be located close to the broadcaster. Moreover, since mobile users are usually charged for data usage, such users are not always suitable for forwarding data. Therefore, in this paper with new architecture for P2P network with lower startup delay, minimize packet loss and decrease control overhead messages compare to the pure mesh-based, pure tree-based networks. In future plan, we will plan to evaluate our proposed model in Planet Lab to examine the effectiveness of our model in a real network.

References

[1] M. Sanna, and E. Izquierdo, *Live scalable video streaming on peer-to-peer overlays with network coding*, IEEE Latin America Transactions, Vol. 11, No. 3, pp. 962–968, 2013.

[2] Hyunseok Chang, Sugih Jamin, and Wenjie Wang, *Live streaming with receiver-based*

peer-division multiplexing, IEEE/ACM Transactions on Networking, Vol. 19, No. 1, pp. 55-68, 2011.

[3] Bo Liu, Yi Cui, Yansheng Lu, Yuan Xue, *Locality-awareness in bittorrent-like P2P applications*, IEEE Transactions on Multimedia, Vol. 11, No. 3, pp. 361-371, 2009.

[4] Di Wu, Yi Liang, Jian He, Xiaojun Hei, *Balancing performance and fairness in P2P live video systems*, IEEE transactions on circuits and systems for video technology, Vol. 23, No. 6, pp. 1029-1039, 2013.

[5] Bin Fan, John C. S. Lui, Dah-Ming Chiu, *The design trade-offs of bittorrent-like file sharing protocols, IEEE/ACM Transactions on Networking*, Vol. 17, No. 2, pp. 365-376, 2009.

[6] Xu Cheng, Jiangchuan Liu, Haiyang Wang, Chonggang Wang, *Coordinate live streaming and storage sharing for social media content distribution*, IEEE Transactions on Multimedia, Vol. 14, No. 6, pp. 1558-1565, 2012.

[7] Tran Thi Thu Ha, Yonggwan Won, and Jinsul Kim, *Topology and architecture design for peer to peer video live streaming system on mobile broadcasting social media,* International Conference on Information Science and Applications, pp. 1-4, 2014.

[8] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and James W. O'Toole, *Overcast: Reliable multicasting with an overlay network*, In Proceedings of the ACM Symposium on Operating System Design and Implementation, Vol. 4, pp. 14-29, 2000.

[9] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum, *Coolstreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming,* In Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Vol. 3, pp. 2102-2111, 2005.

[10] Nazanin Magharei and Reza Rejaie, *PRIME: Peer-to-peer receiver-driven mesh-based streaming,* IEEE/ACM Transactions on Networking, Vol. 17, No. 4, pp. 1052-1065, 2009.

P2P 라이브 스트리밍 서비스를 위한 스트리밍 사용자 및 저장 사용자 결합형 오버레이 구조 설계 기법

Tran Thi Thu Ha[1], 장 병 옥[2]

[1] *전남대학교 전자컴퓨터공학부*
[2] *나사렛대학교 디지털콘텐츠학과*

요 약

Peer-to-Peer(P2P)는 네트워크상에서 서버 없이 사용자들 간에 데이터를 공유하는 방법을 일컫는다. 사용자가 언제든 서비스에 참여하거나 떠날 수 있는 특징 때문에, 유동적인 네트워크 환경에서 高 퀄리티의 서비스를 안정적으로 제공하는 것이 P2P의 기술적 과제라 할 수 있다. 3G/4G 망을 사용하는 모바일 스트리밍 사용자는 라이브 비디오 시청 시 해당 비디오에 흥미가 없다고 느끼면 일정한 시간 내에 비디오 시청을 그만두는 경향이 높다. 이러한 피어 이탈은 네트워크를 불안정하게 하고 콘텐츠 전달 지연을 야기할 수 있다. 반면 유선 인터넷 및 PC 기반의 저장 사용자는 모바일 사용자에 비해 콘텐츠를 저장하는 비중이 높고, 저장 완료 후 시청 전까지는 비디오 퀄리티를 신경 쓰지 않는 편이다. 따라서 저장사용자는 스트리밍 사용자보다 안정적이라 할 수 있다. 우리는 상대적으로 안정적인 저장 사용자를 브로드캐스터의 자리에 가깝게 배치함으로써 효율적인 데이터 공유 구조를 제안한다. 또한 본 논문에서 제시한 P2P 스트리밍 기

법은 트리기반 모델과 메시 기반 모델의 장점을 결합한 하이브리드 모델에 적용하여 스타트업 지연시간(Startup Delay)과 패킷 손실 및 컨트롤 오버헤드를 감소시킨다.

Acknowledgments

This research was supported by Research Program of Korea Nazarene University (Prof. Jang Byeongok), 2014.

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, Peer-to-Peer Network.

E-mail address: thuhabkhn@gmail.com

Byeongok Jang received is computer science from Kyonggi University in 1999, and Ph.D. He worked as a professor in the treatment of Kwangwoon University, Seoul, Korea from 1999 to 2000. Currently, he is a professor in department of broadcasting contents, Korea Nazarene University, Cheonan, Korea. His research interests include Digital Contents, Broadcasting, Web-based Multimedia Service, Internet Information System and Digital Media Arts.

E-mail address: bojang@kornu.ac.kr