British Journal of Mathematics & Computer Science
17(2): 1-20, 2016, Article no.BJMCS.23897
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

Motif-based QoS-aware Dynamic Optimization of P2P Streaming Networks

Kazuki Ono[1], Andrii Zhygmanovskyi[1], Noriko Matsumoto[1] and Norihiko Yoshida[1*]

[1] *Graduate School of Science and Engineering, Saitama University, Japan.*

Authors' contributions

This work was carried out in collaboration between all authors. Author NY designed the study, wrote the protocol and supervised the work. Author NM managed the analyses of the study and literature searches. Author KO, in collaboration with author AZ, wrote the codes for simulation and performed the analyses. Author KO wrote the first draft of the manuscript, and authors AZ, NM and NY edited the manuscript. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/BJMCS/2016/23897
Editor(s):
(1) Junjie Chen, Department of Electrical Engineering, University of Texas at Arlington, USA.
Reviewers:
(1) Aeizaal Azman Abdul Wahab, Universiti Sains Malaysia, Malaysia.
(2) R. Jaichandran, AVIT-Vinayaka Missions University, Tamil nadu, India.
(3) Yueran Gao, Southern Illinois University Carbondale, USA.
(4) Srinivas Kanakala, Vaagdevi College of Engineering, Warangal, Telangana State, India.
Complete Peer review History: http://sciencedomain.org/review-history/14972

Original Research Article

Received: 28th December 2015
Accepted: 4th June 2016
Published: 10th June 2016

Abstract

In recent years, Peer-to-Peer (P2P) streaming networks have been becoming popular because of not only fault tolerance, but also load balancing. However, in P2P streaming networks, there are some problems such as topology imbalance, interruption of streaming, latency in segment delivery, and degradation of received content caused by connecting low-performance nodes. Therefore, in this paper, we propose a new approach which reconfigures streaming networks dynamically using network motifs which can be used as an approach to characterize the complex networks, and evaluate our approach by simulation. The results indicate that our approach greatly decreases height and latency in generated trees, and provides an ability to reconfigure dynamically the network in case of predecessor's disconnection or defection. Moreover, our approach improves the quality of contents in a streaming network regardless of its network size.

**Corresponding author: E-mail: yoshida@mail.saitama-u.ac.jp;*

Keywords: Peer-to-peer streaming networks; network motifs; resilience; dynamic reconfiguration.

2010 Mathematics Subject Classification: 53C25, 83C05, 57N16.

1 Introduction

Traditional client-server based streaming systems can be constructed easily and provide good performance, although it usually imposes high deployment and maintenance costs on the owner. P2P streaming systems attract much attention because of their superior features: cost reduction, flexibility in case of flash crowds, inherent bandwidth, resource scalability, multiple network paths and self-organization. However, there also exist some problems, for example, increased maximum number of hops, concentrated connection or churn [1].

There already exist many proposals addressing these issues, such as selecting topologies [2], scheduling algorithms [3], incentives [4], re-transmission, coding [1] and balancing topologies according to network motifs [5]. However, since these methods presuppose static topology during streaming operations, they cannot address the problem of dynamic reconstruction in resilient P2P streaming systems. Hence, we proposed a reliable P2P live streaming system which reconstructs its own overlay autonomously [6]. This proposal consists of three processes: observing the state of neighbor nodes, selecting next parent node and spare nodes, which are called predecessors, and balancing topologies using network motifs [7]. However, in our previous work, we do not concern the existence of low-performance node. Therefore, in this paper, we propose a new approach which reconfigures streaming networks dynamically and provide high quality content. This paper is an extended version of our conference paper [6], adding some new topics and results regarding QoS awareness.

This paper is organized as follows: we describe research background in Section 2, and present the definition of resilience, related works and our previous work in Section 3. In Section 4, we describe our system design. Subsequently, we show the evaluation of our method through simulation in Section 5. Finally, Section 6 contains the conclusion and future work.

2 Background

The demand for user-friendly streaming systems for both content consumers and recipients is increasing. In general, a streaming system is constructed based on client-server architecture. It is important to clarify the difference between client-server based and P2P-based streaming systems. First of all, we describe features of these two, and then discuss features of P2P streaming systems.

2.1 Streaming systems

In streaming systems, large files are transmitted, causing the original content to be divided into many small segments. The recipients play back received segments while downloading next segments. Streaming systems can be divided into two types: client-server based systems and P2P-based systems. Furthermore, these systems are classified by distribution method: Video-On-Demand (VOD) and Live. In VOD steaming systems, all clients that participate in the streaming network can play segments at any time. However, in live streaming systems, all clients need to play the same segments at the same time. On that account, constructing live streaming systems based on P2P is more difficult than doing it based on client-server approach. Therefore, we focus on P2P-based live streaming systems.

2.1.1 Client-server based streaming systems

In client-server based streaming systems [8], all clients request desired content from the server who owns them. If the number of clients increases significantly, the server becomes heavy loaded. Moreover, in the worst case of server down, all clients will be unable to download any content at all. There are some methods that address these issues, the simplest one being to prepare extra servers in advance. This solution incurs high deployment and maintenance costs, although it is a very easy way which performs well. For example, current estimated cost of YouTube [8] is 1 million dollars per day [1].

2.1.2 P2P-based streaming systems

It is necessary to note that unlike client-server based streaming systems, all clients in P2P-based streaming systems [9][10] can be both senders and receivers. In general, P2P-based systems are not as costly as client-server based systems, because the former can distribute the load of server using all resources of participating nodes. Therefore, various P2P streaming systems were proposed to perform Live and Video-On-Demand streaming to mass audience with better quality at low server and deployment costs. While having these advantages, P2P streaming system has also the drawback of network topology becoming highly dynamic because of churn. Therefore, interruption of streaming, latency in segment delivery, and degradation of received content caused by connecting low-performance nodes are frequently observed in P2P streaming systems.

In order to address these problems, it is important to ensure that network is resilient. A definition of resilience is given by Abbound et al. [1] as follows: "The persistence of avoiding too frequent or severe failures in the presence of changes." We show some approaches that ensure resilience in Section 3.

2.2 Problems in P2P streaming systems

As mentioned above, there are some problems in P2P streaming systems. They can be roughly divided into the following four kinds:

1. **Highly dynamic topology**
   P2P-based streaming systems let all clients decide freely when to join or leave. This results in frequent topology changes.

2. **Strict real-time constraints**
   In streaming systems, all nodes must ensure synchronous playback. However, in P2P live streaming systems, the number of hops differs for each node. Therefore, it becomes difficult to accomplish synchronous playback.

3. **Topology imbalance**
   In P2P streaming networks, all nodes select a predecessor randomly. This causes an increase in the number of hops and connection concentration.

4. **Existence of low-performance node**
   If nodes connect to a low-performance node, they receive low-quality contents and transmit the contents to their successors. This causes a degradation of qualities in streaming networks.

In order to address these problems, especially the dynamic change of network topologies as well as strict real-time constraints, the system needs to ensure resilience.

3 Resilience in P2P Streaming Systems

To ensure resilience in P2P streaming systems, different approaches has been proposed. In this section, first we explain several of them, then introduce the related work that concerns balancing topologies using network motifs. Finally, we introduce our approach.

3.1 Approaches to ensure resilience

There are the following approaches to ensure resilience in P2P streaming systems:

1. **Topology selection**
   There are three basic topologies: a tree, a mesh, and a hybrid of them. Topology selection should be based on actual load, streaming interruptions and amount of effort needed to construct an overlay among others.

2. **Scheduling algorithms**
   In P2P streaming systems, each node has different performance. To maximize the quality of content and reduce end-to-end delay, a system should determine which segments should be re-transmitted and when these transmissions should be carried out.

3. **Re-transmission**
   Since a node can request a missing packet from another node (not necessary being its original sender), re-transmission provides resilience. However, in live streaming, the node must receive missing packet before proceeding with current stream playback. Therefore, this approach has strict time constraints.

4. **Incentives**
   System performance may suffer significantly if nodes do not contribute their resources fully, for example, restricting upload bandwidth or leaving the system once segments have been delivered. Therefore, the system should encourage nodes to stay connected.

5. **Network coding**
   Network coding is a technique by which nodes in a streaming system encode multiple blocks into one instead of simply sending data blocks. This approach leads to decreasing the amount of packets in the network. However, this algorithm can be difficult to implement.

6. **Media coding**
   Media coding enhances resilience by allowing the receiver to deal better with losses and bandwidth fluctuations.

3.2 Network motif

In P2P streaming systems, topology imbalance poses a serious problem. In order to address this problem, a method of balancing topologies using network motifs is proposed [5].

3.2.1 Network motifs

Network motifs is defined as recurrent of statistically significant subgraph or pattern. The aim of this concept is to narrow the gap between local and global knowledge of large networks and to understand network structure better [7]. In network motifs, each node has at least one input or output.

Network motifs can be used as an approach to compare the difference of network characteristics in biological studies, World Wide Web (WWW) or electric circuit networks [11].

Fig. 1 shows all patterns of network motifs which are composed of three nodes.

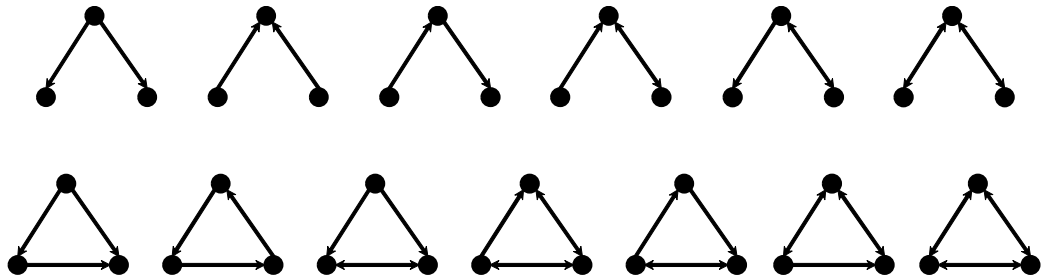

Fig. 1. Network motifs

3.2.2 Approach in related work

In the related work [5], all nodes make request to their predecessors/successors and all transfers are performed from successor to predecessors when they enter the network or are already connected to successors. Each node's behavior is as follows:

1. When a node enters the streaming network

1. Select a predecessor randomly.
2. A node, which enters the streaming network, obtain immediate predecessor's information and predecessor's parent information.
3. Reconnect to a new predecessor that has sufficient load margin.

2. When a node is connected to successors

1. Get neighbor node's information.
2. Compare neighbor node's load with one's own.
3. Reconnect a successor to a new predecessor that has sufficient load margin.

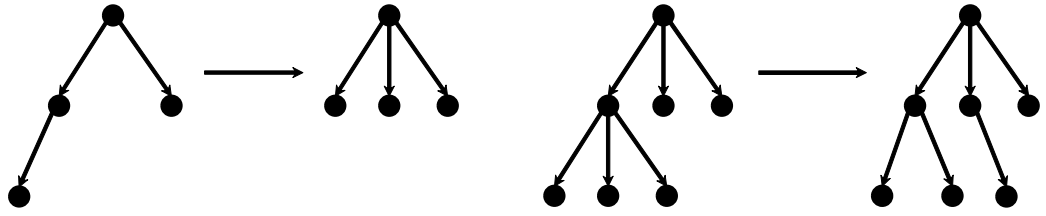

Fig. 2. Successor exchange operations

Fig. 2 shows examples of the above two transfers.

This approach makes use of balanced tree topologies, decreases the server load and increases the scalability in case of large audience. However, this exchange operation is effective only when nodes enter a streaming network or are already connected to successors, and it does not consider the predecessor's disconnection or defection. Therefore, it can be said that this approach ensures resilience only in static manner.

3.3 Previous work

At present, most live streaming systems are client-server based; but, since this approach incurs high deployment and maintenance costs, P2P live streaming systems draw much attention. However, as mentioned above, there are still many problems that arise in P2P live streaming systems.

In order to address these problems, several approaches [3][4][5] were proposed; however, none of them concern reconfiguring topologies dynamically which is important in P2P networks, and overall there are few works that address such situation. Therefore, in our previous work, we proposed a reliable P2P live streaming system which reconstructs its own topology dynamically. This approach consists of three methods: observing neighbor node's state, selecting a predecessor and spare predecessors, and balancing topologies using network motifs [6].

3.3.1 Observing neighbor nodes state

Each node observes its own neighbor nodes while playing back current stream. If node's predecessor has sufficient margin in the load, then the node reconnects its successor to that predecessor. If the predecessor has not enough margin in the load but some of its siblings have margin larger than its own margin, then the node reconnects its successor to one of its predecessor's siblings.

Figs. 3 and 4 show examples of such a behavior. In these figures, each node is allowed to be connected with at most three nodes.

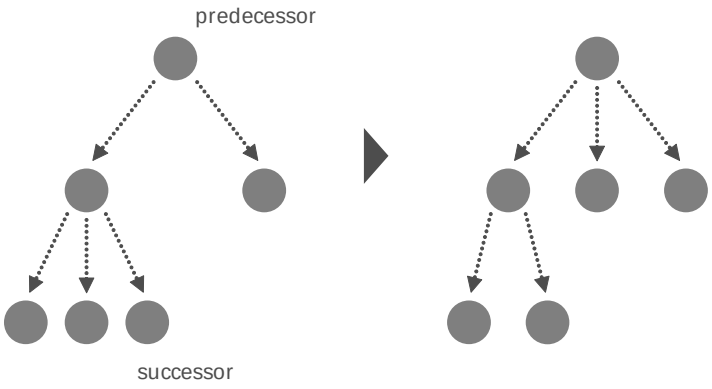

Fig. 3. Situation when predecessor has sufficient margin in load

3.3.2 Selecting next predecessor and spare predecessor

Each node is able to continue streaming by quickly replacing its own predecessor in case of disconnection or defection of the latter. Exchange operations during replacing of predecessor are as follows:

1. Each node observes its predecessor's state: normal, disconnected or defected.
2. A node selects new predecessor from predecessor's sibling nodes by largest margin in the load.
3. In case of predecessor's disconnection or defection, each node reconnects to the new predecessor.

Fig. 5 shows an example of this behavior.

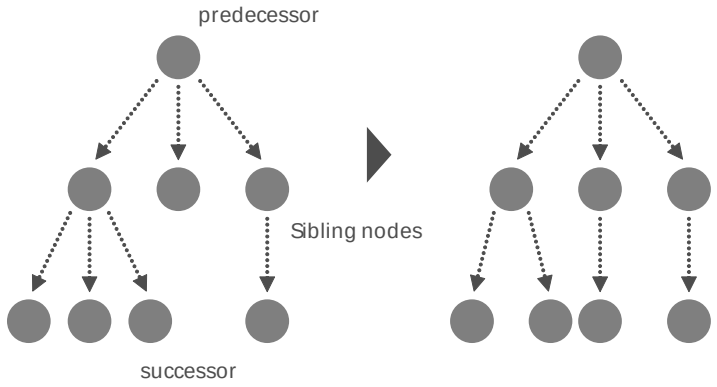

Fig. 4. Situation when sibling nodes have larger margin

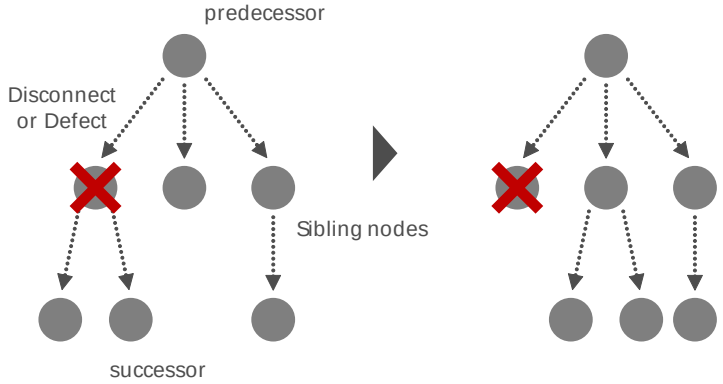

Fig. 5. Replacing predecessor

3.3.3 Balancing topologies using network motifs

P2P streaming networks are prone to topology imbalance resulting in an increase in the number of hops as well as connection concentration. Therefore, in this work we adopt a topology balancing

approach that was proposed by Krumov et al. [5].

Using this approach, we achieve both an optimization of hop count and static load balancing in each node.

Fig. 6 shows an example of balancing topologies, and Algorithm 1 presents a pseudo code of our previous approach.

This approach decreases the height and latency in generated trees, and provides an ability to dynamically reconfigure the network in case of predecessor's disconnection or defection. However, in this approach, we do not consider the existence of low-performance node and degradation of contents in streaming networks.

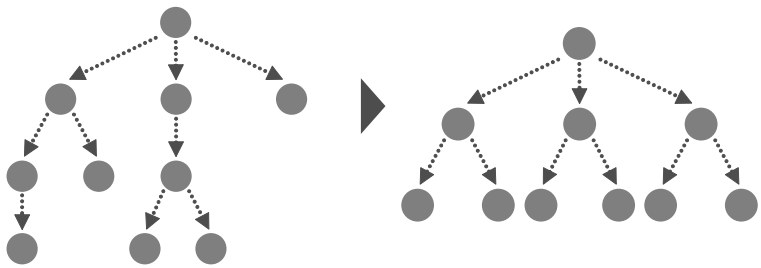

Fig. 6. Balancing topologies using network motifs

Algorithm 1 Pseudo code without considering content qualities

Require: $Neighbor_i, Parent, Child$;
  while playback segments **do**
    $ObserveNeighbor(Neighbor_i)$;
    $NewParent \Leftarrow Neighbor_1$;
    $SubParent \Leftarrow Neighbor_2$;
    if $NumChilds > Neighbor_i'sNumChilds$ **then**
      $Reconnect(Child, Neighbor_i)$;
    end if
    if Parent is Disconnect or Defect **then**
      if NewParent is Alive **then**
        $Parent \Leftarrow NewParent$;
      else
        $Parent \Leftarrow SubParent$;
      end if
    end if
  end while

4 System Design

In this section, we describe our approach to address the degradation of content qualities in streaming networks. This approach adds two methods to our previous approach [6] : observing sibling nodes and observing a predecessor.

4.1 Observing sibling node's bitrate for playback

Each node observes its sibling nodes while playing back a current stream. If the bitrate of a sibling node for playback is very close to node's successor's value, then the node reconnects its successor to that sibling node.

Fig. 7 shows an example of this successor exchange operation.

4.2 Observing parent node's bitrate for playback

Each node observes its predecessor while playing back the current stream. If the predecessor's bitrate for playback is smaller than the node's bitrate, then the node exchanges its own position with the predecessor's position.

Fig. 8 shows an example of this position exchange operation, and Algorithm 2 presents a pseudo code of our new approach.

Algorithm 2 Pseudo code considering content qualities

Require: $Neighbor_i, Parent, Child$;
  while playback segments **do**
    $ObserveNeighbor(Neighbor_i)$;
    $NewParent \Leftarrow Neighbor_1$;
    $SubParent \Leftarrow Neighbor_2$;
    if $NumChilds > Neighbor_i'sNumChilds$ **then**
      if $CheckBitrate(Neighbor_i)$ **then**
        $Reconnect(Child, Neighbor_i)$;
      end if
    end if
    if $Parent'sBitrate < MyBitrate$ **then**
      $Swap(Parent)$
    end if
    if Parent is Disconnect or Defect **then**
      if NewParent is Alive **then**
        $Parent \Leftarrow NewParent$;
      else
        $Parent \Leftarrow SubParent$;
      end if
    end if
  end while

5 Evaluation

In this section, we present evaluation results of our approach. They include structural properties of generated trees and dynamic reconfiguration.

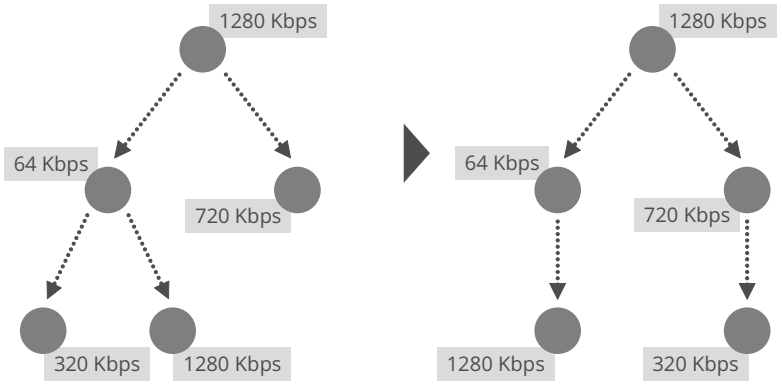

Fig. 7. Observing sibling node

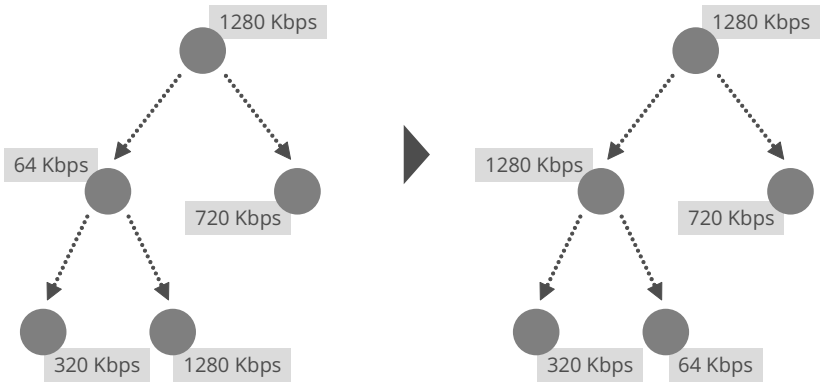

Fig. 8. Observing predecessor

We compare four approaches : Primary Approach, Motif-based Approach, Bitrate-based Approach, and Proposed Approach.

- **Primary Approach**
  Every node in a streaming tree connects to its predecessor randomly.
- **Motif-based Approach**
  This corresponds to our previous approach which uses network motifs [6]. Every node observes the predecessor's disconnection or defection, and reorganizes the tree dynamically.
- **Bitrate-based Approach**
  Every node in a streaming tree connects to its predecessor reflecting its playback bitrate,

while not observing the predecessor's disconnection or defection.

- **Proposed Approach**
  This corresponds to our new approach which uses network motifs. Every node reflects its playback bitrate, and observes the predecessor's disconnection or defection.

In all of our experiments, we use the following scenario: there is one root node providing the system with the original streaming signal.

We investigate our approach from four different perspectives: topology's height and latency, connectivity and exchange steps for each node, the bitrate of received segments, and Topo-metric values.

For each node, Topo-metric value is defined in [5] as the difference between the longest and the shortest branches of succeeding subtrees, starting from certain node in the whole tree. If Topo-metric value is large, the streaming topology becomes imbalanced. Therefore, smaller value is preferable.

Fig. 9 shows a sample tree with the topo-metric values of the nodes in the tree. The topo-metric value of this sample tree is defined as the total of the topo-metric values of these nodes. In this example, topo-metric value is $2 + 1 + 0 + 0 + 0 + 0 = 3$.

To evaluate our approach, we prepared a simulator which runs on a single computer. This simulator is implemented in Java, and composes a virtual P2P streaming network. Every node in the P2P streaming network receives segments from its predecessor, and if a nodes has successors, it sends the received segments to its successors. Some major parameters of the simulation are summarized in Table 1.

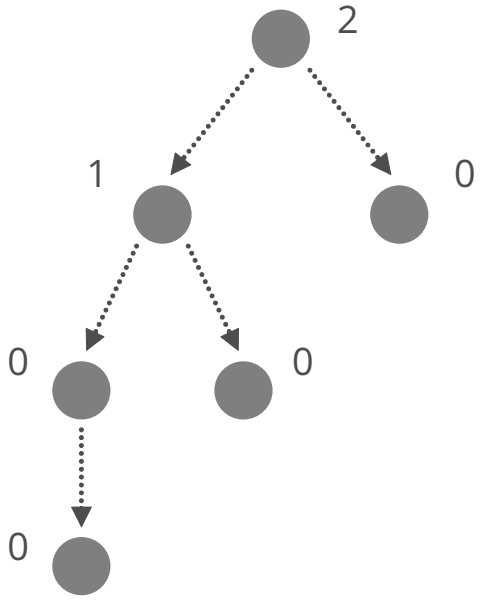

Fig. 9. Sample tree with corresponding Topo-metric values

Table 1. Simulation parameters

Parameter	Value
Number of predecessors	1
Number of successors	3
Number of original senders	1
Number of pseudo segments	10
Playback time for pseudo segment	5000 ms
Simulation time	50000 ms

5.1 Height and latency in generated trees

In these experiments, we confirmed how the height and latency in generated trees were changed using our approach.

Fig. 10 shows the height of generated trees, and Fig.11 their latency in a range from 10 to 1000 nodes. In addition, we assume that our simulator uses 100 Mbps bandwidth. Therefore, we set the latency of communication between each node to 10 milliseconds, and we target the terminal nodes to calculate the latency. In this experiment, we did the following actions randomly: addition of a new node, predecessor's disconnection or predecessor's defection. Moreover, our simulator generates a streaming tree randomly. Therefore, the generated tree's height differs in each simulation. These results show the average of 20 executions for each experiment.

Fig. 12 shows the change in topology's height in case addition of a new node, predecessor's disconnection, and predecessor's defection has occurred. In this figure, addition of a new node and predecessor's defection occurred at approximately 10000 milliseconds, addition of a new node and predecessor's disconnection occurred at approximately 25000 milliseconds, and predecessor's disconnection and defection occurred at approximately 45000 milliseconds after the simulation had started, and streaming topology consists of 200 nodes.

From these results, we can conclude that both the height and latency in generated trees are reduced using our approach, and the latency does not depend simply on the height of streaming topology because the number of hops is different for each node. Furthermore, the height of topology increases after predecessor's disconnection or defection,although it immediately decreases afterwards. The spike in Fig. 12 shows them. Therefore, our new approach can reconfigure the topologies dynamically in case of predecessor's disconnection or defection, and decreases maximum number of hops.

5.2 Number of exchange steps and connectivity in generated trees

In these experiments, we confirmed how the number of exchange steps and connectivity in generated trees were changed using our approach.

Connectivity is the number of nodes to which a node is connected. In this work, each node allows up to 3 connected nodes, therefore theoretically optimal node connectivity is exactly 3.

Fig. 13 shows the average connectivity for each node in generated trees, and Fig. 14 shows the number of exchange steps for each node in generated trees, in a range from 10 to 1000 nodes. In this experiment, we did the following actions randomly: addition of a new node, predecessor's disconnection or predecessor's defection. Moreover, our simulator generates a streaming tree randomly.

Therefore, both the generated tree's exchange steps and connectivity differ in each simulation. These results show the average of 20 executions for each experiment.

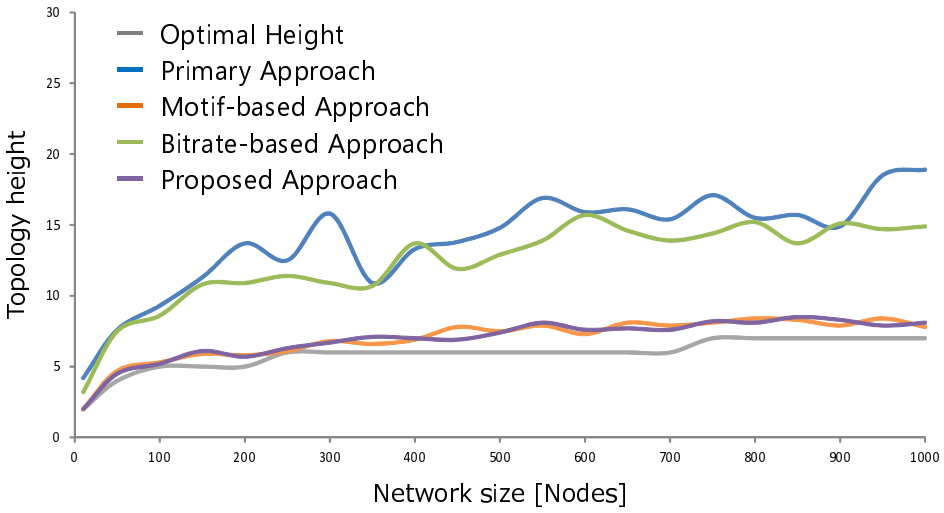

Fig. 10. Change of topology's height with regard to network size

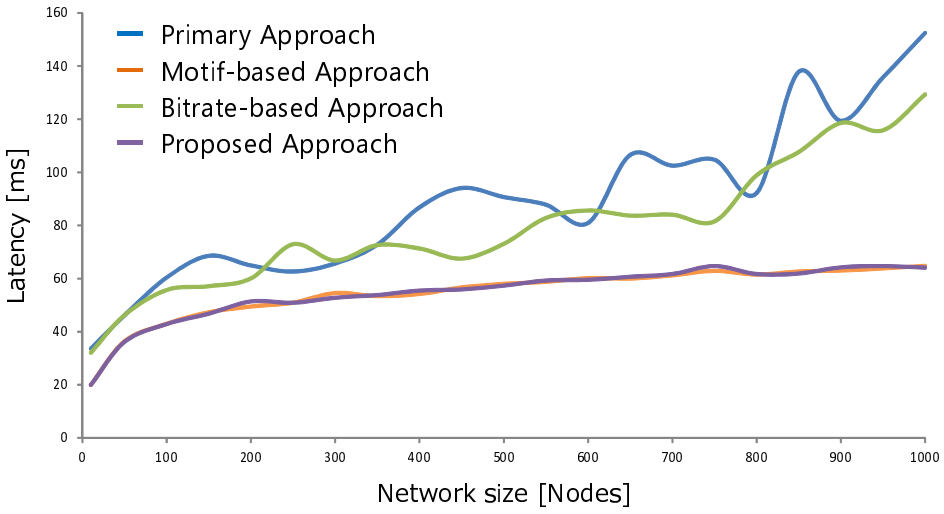

Fig. 11. Change of latency with regard to network size

Fig. 15 shows the result of a dynamic change of the average connectivity for each node in case addition of a new node, predecessor's disconnection, and predecessor's defection has occurred. In this figure, addition of a new node and predecessor's defection occurred at approximately 10000 milliseconds, addition of a new node and predecessor's disconnection occurred at approximately

25000 milliseconds, and predecessor's disconnection and defection occurred at approximately 45000 milliseconds after the simulation had started, and streaming topology consists of 200 nodes.

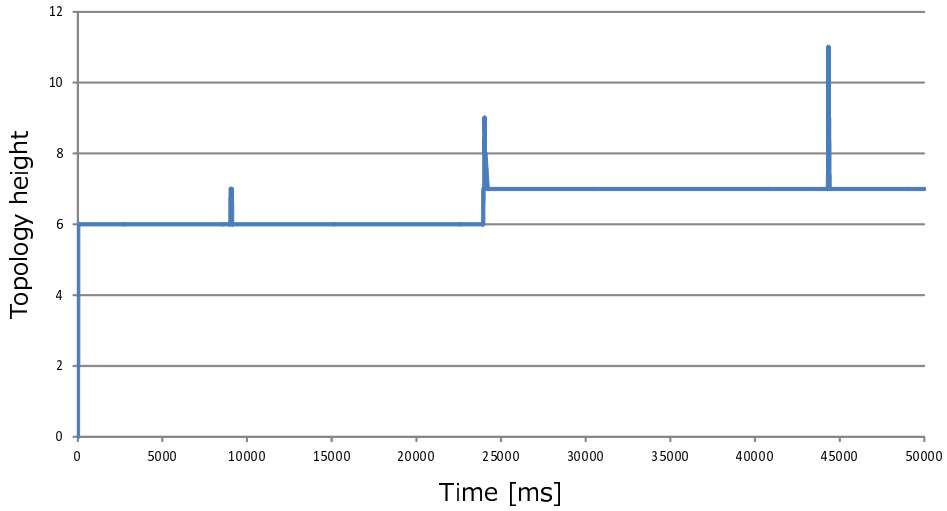

Fig. 12. Change of topology's height in time

From these results we conclude that with our approach amount of successors for each node increases regardless of the network size. The average of exchange steps per node is also independent of the network size, and the value of our approach is approximately three-times higher than of the motif-based approach. However, it grows sublinearly with respect to network size, increasing from 2 to more than 2.9. Furthermore, the connectivity of each node decreases after predecessor's disconnection or defection has occurred, however, it immediately increases afterwards. The spike in Fig. 15 shows them. Therefore, our approach can reconfigure the topologies dynamically in case of predecessor's disconnection or defection, and achieves dynamic load balancing.

5.3 Topo-metric values in generated trees

In these experiments, we confirmed how Topo-metric values in generated trees were changed using our approach.

Fig. 16 shows the Topo-metric value in generated trees, in a range from 10 to 1000 nodes. In this experiment, we did the following actions randomly: addition of a new node, predecessor's disconnection or predecessor's defection. Moreover, our simulator generates a streaming tree randomly. Therefore, the generated tree's Topo-metric value differs in each simulation. These results show the average of 20 executions for each experiment.

Fig. 17 shows the result of a dynamic change of the Topo-metric value in case addition of a new node, predecessor's disconnection, and predecessor's defection has occurred. Here, addition of a new node and predecessor's defection occurred at approximately 10000 milliseconds, addition of a new node and predecessor's disconnection occurred at approximately 25000 milliseconds, and predecessor's disconnection and defection occurred at approximately 45000 milliseconds after the simulation had started, and streaming topology consists of 200 nodes.

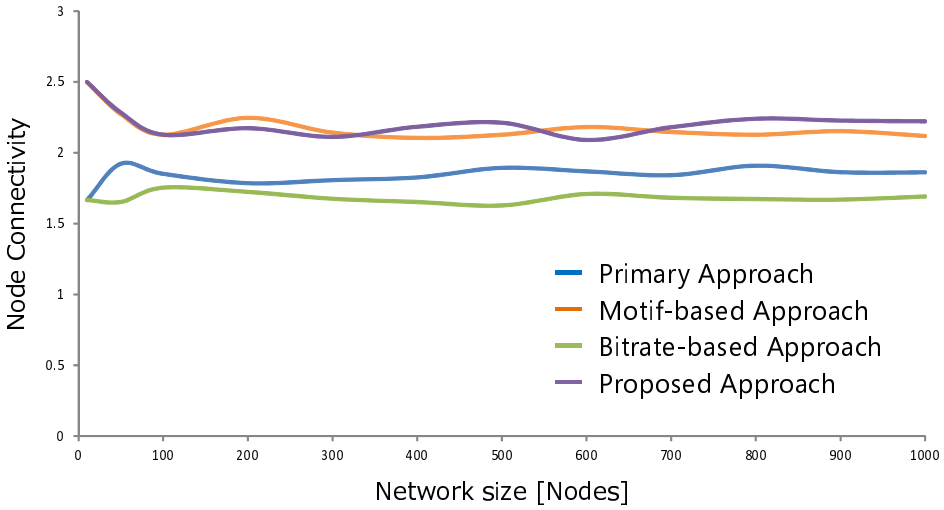

Fig. 13. Change of average node connectivity with regard to network size.

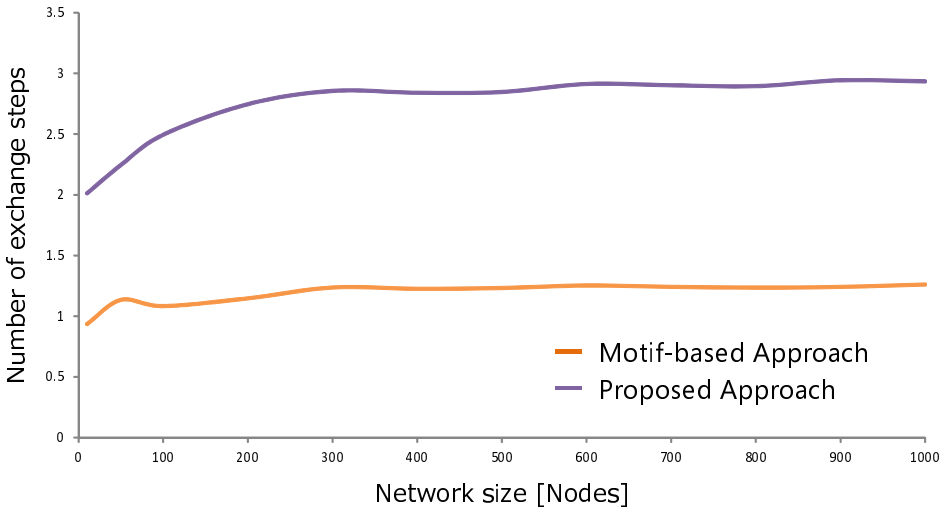

Fig. 14. Change of number of exchange steps per node with regard to network size.

From these results, we can conclude that Topo-metric value in generated trees is lowered using our approach. Furthermore, the Topo-metric value increases after predecessor's disconnection or defection, although immediately decreases afterwards. The spike in figure 5.4 shows them. Therefore, our approach can reconfigure topologies dynamically in case of predecessor's disconnection or defection, and produces balanced topologies.

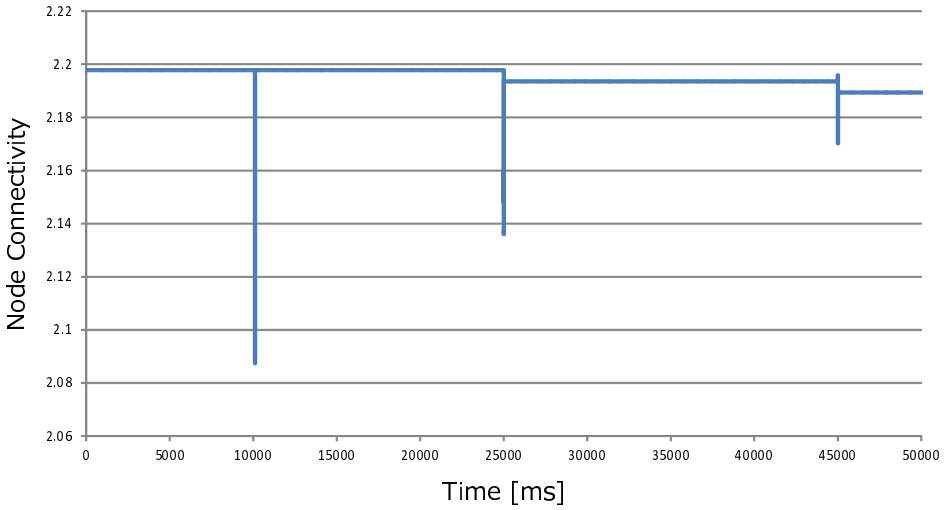

Fig. 15. Change of average node connectivity in time.

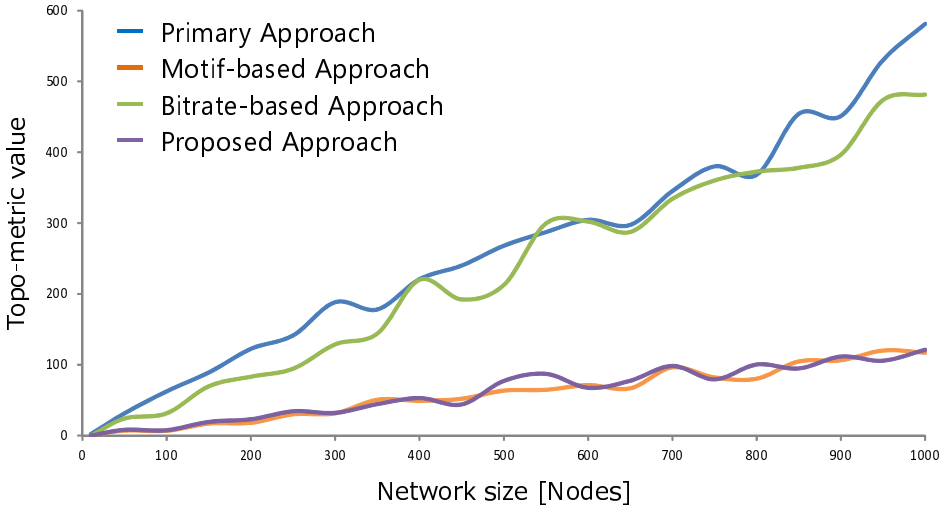

Fig. 16. Change of Topo-metric value with regard to network size

5.4 The bitrate of received segments in generated trees

In these experiments, we confirmed how the bitrate of received segments in generated trees were changed using our approach.

Figs. 18, 19 and 20 show the the bitrate of received segments in generated trees, in a range from 10 to 1000 nodes. In these experiments, we did the following actions randomly: addition of a new node, predecessor's disconnection or predecessor's defection. Moreover, our simulator generates a

streaming tree randomly. Therefore, the bitrate of received segments differs in each simulation. In addition, we changed each node's bitrate for playback ([Kbps]) in each figure. These results show the average of 20 executions for each experiment.

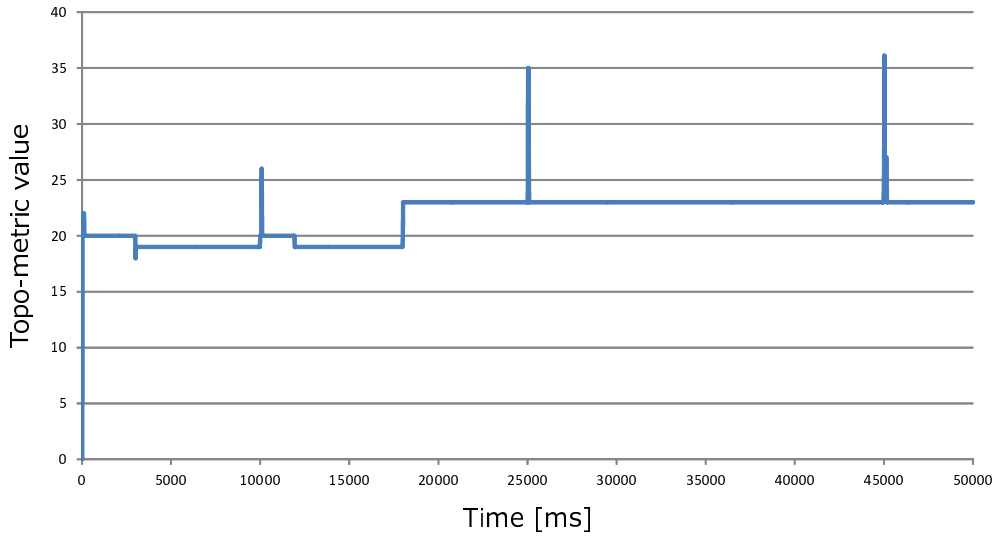

Fig. 17. Change of Topo-metric values in generated trees

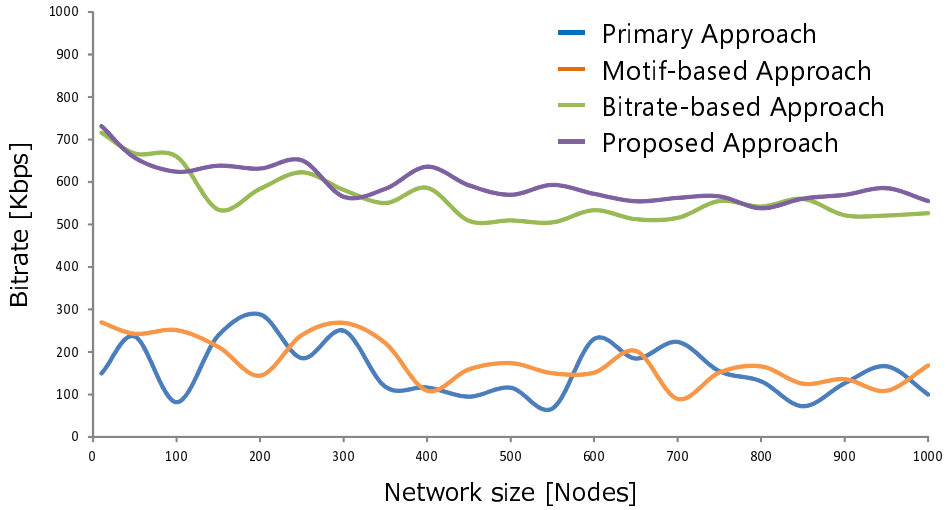

Fig. 18. Change of the bitrate of received segments with regard to network size (1280 [Kbps] : 720 [Kbps] : 320 [Kbps] : 64 [Kbps] = 2 : 3 : 3 : 2)

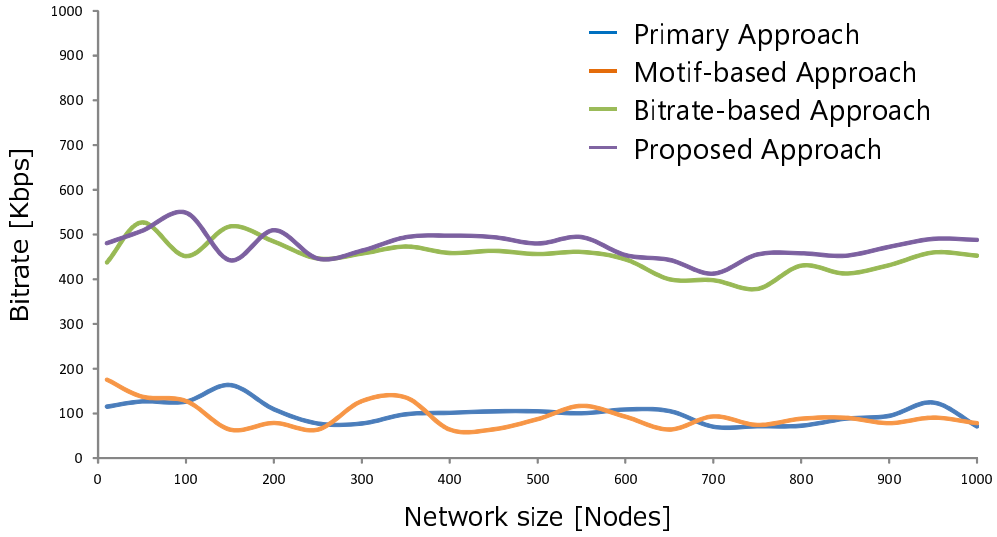

Fig. 19. Change of the bitrate of received segments with regard to network size (1280 [Kbps] : 720 [Kbps] : 320 [Kbps] : 64 [Kbps] = 2 : 2 : 2 : 4)

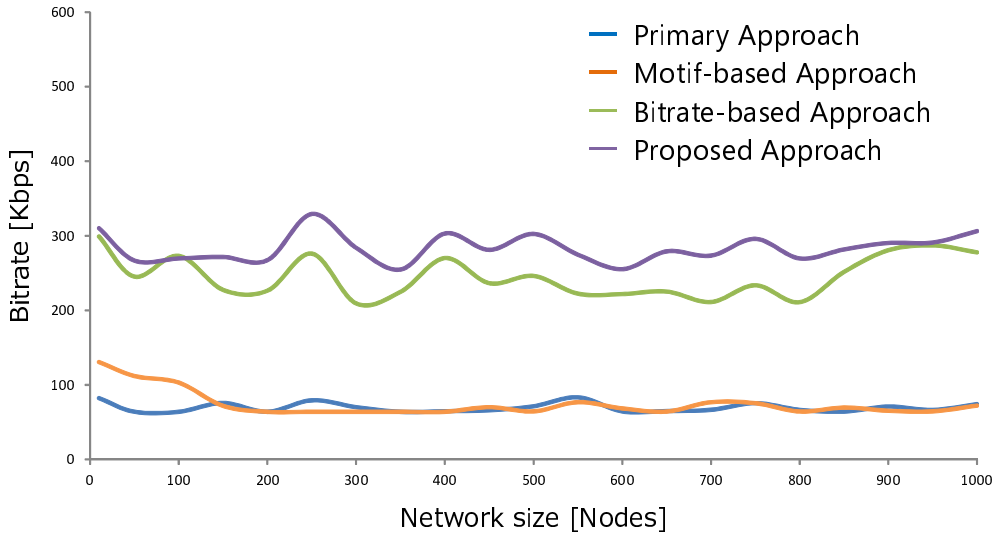

Fig. 20. Change of the bitrate of received segments with regard to network size (1280 [Kbps] : 720 [Kbps] : 320 [Kbps] : 64 [Kbps] = 1 : 1 : 2 : 6)

From these results, we can conclude that the average the bitrate of received segments increases using our new approach. Furthermore, this approach can improve the quality of contents in streaming

network regardless of its network size, and many participants can receive high quality content. Therefore, our new approach can reconfigure the topologies dynamically based on node's bitrate for playback.

From all results, we can conclude that our approach greatly decreases height and latency in generated trees, and provides an ability to dynamically reconfigure the network in case of predecessor's disconnection or defection. Moreover, our approach improves the quality of contents in streaming networks regardless of its network size. Therefore, our approach regarding QoS awareness is more effective in improving the quality of contents than our previous work [6].

6 Conclusion and Future Work

In previous work, we studied the resilience of P2P streaming systems that comes as a result of dynamic reconfiguration of peer-to-peer network, however we do not concern the existence of low-performance node. Therefore, in this paper, we consider how to deal with low-performance nodes and dynamic reconfiguration of peer-to-peer network using network motifs.

In this study, we considered the scenario which consists of a source node that provides the original streaming signal, and other nodes in topology, that distribute the signal among each other. As mentioned above, our new approach which extends our previous approach [6] provides an ability to dynamically reconfigure the network in case of predecessor's disconnection or defection. Moreover, our approach improves the quality of contents in streaming networks regardless of its network size.

Issues to address in further studies are as follows.

6.1 Dealing with free-riders and malicious nodes

Our simulator does not consider free-riders or malicious nodes: all nodes in the streaming network receive segments and send them to successors. However, in real P2P streaming systems not all nodes are like this. Even if free-riders and malicious nodes are present, it is difficult to detect them, and the efficiency of load balancing using our approach suffers. In order to address this problem, Simple Trust Exchange Protocol (STEP) is advocated as a possible solution [12]. Therefore, we need to improve our approach and simulator to model actual P2P streaming systems.

6.2 Practical experiments using HTTP Live Streaming (HLS)

Our simulator was executed on single computer, and lacks the following features: recording, encoding and segmentation. Therefore, the load applied on root node in our simulator is much smaller compared to real world scenario. Furthermore, since we used pseudo segments in our simulator, the latency is not exact. Because of these issues, we need to measure latency and load rigorously when segments are delivered to successors. In order to address these problems, we need to consider using HTTP Live Streaming [13], and confirm the soundness of our approach.

6.3 Considering node's requiring quality

Our simulator does not consider nodes' requiring qualities: nodes which have high bitrate are located near the original sender, and nodes which have low bitrate are located for the terminal. However, in real P2P streaming systems, several nodes want to playback segments which have low bitrate and latency. Therefore, we need to improve our approach and simulator to deal with node's requiring quality.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Abbound O, et al. Enabling resilient P2P video streaming: Survey and analysis. Multimedia System. 2011;17(3):177-197.

[2] Wu C, et al. Exploring large-scale peer-to-peer live streaming topologies. ACM Transactions on Multimedia Computing, Communications, and Applications. 2008;4(3):1-23.

[3] Guo Y, et al. AQCS: Adaptive queue-based chunk scheduling for P2P live streaming. Proc 7th International IFIP-TC6 Networking Conference Singapore. 2008;433-444.

[4] Liu Z, et al. Using layered video to provide incentives in P2P live streaming. Proc 2007 workshop on peer-to-peer streaming and IP-TV. 2007;311-316.

[5] Krumov L, et al. Resilient peer-to-peer live-streaming using motifs. Proc 2010 IEEE International Symposium on World of Wireless Mobile and Multimedia Networks. 2010;1-8.

[6] Ono K, et al. Resilient live-streaming with dynamic reconfiguration of P2P networks. Proc 6th International Conference on Emerging Network Intelligence. 2014;6-11.

[7] Milo R, et al. Superfamilies of evolved and designed networks. Science. 2004;303(5663):1538-1542.

[8] YouTube. Accessed 6 June 2016.
Available: http://www.youtube.com/

[9] BitTorrent. Accessed 6 June 2016.
Available: http://www.bittorrent.com/

[10] Xiong J, Choudhury RR. Peer cast: Improving link layer multicast through cooperative relaying. Proc 2011 IEEE INFOCOM. 2011;2939-2947.

[11] Brandt C, Leskovec J. Status and friendship: Mechanisms of social network evolution. WWW Companion '14 Proc the 23rd International Conference on World Wide Web. 2014;229-230.

[12] Yoshida Y, et al. Efficient decentralized evaluation of node trustworthiness in peer-to-peer networks. Proc International Conference on Computer Engineering and Technology. 2009;177-179.

[13] HTTP Live Streaming Resources - Apple Developer. Accessed 6 June 2016.
Available: https://developer.apple.com/streaming/

©2016 Ono et al.; This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/14972