Resilient Live-Streaming with Dynamic Reconfiguration of P2P Networks

Kazuki Ono, Andrii Zhygmanovskyi, Noriko Matsumoto, Norihiko Yoshida
Graduate School of Science and Engineering
Saitama University
Saitama, Japan
Emails: {ono, andrew, noriko, yoshida}@ss.ics.saitama-u.ac.jp

***Abstract*—Establishing high robustness and resilience against churn or unexpected behavior of users is an important issue in building reliable Peer-to-Peer (P2P) streaming systems. In P2P-based live streaming systems, interruption and network latency in segment delivery are of particular concern. In order to address these problems, we propose a reliable P2P live streaming system which reconstructs its own topology dynamically, by observing the state of neighbor nodes, selecting a predecessor and spare predecessors, and balancing topologies using network motif. Through various experiments, we show that our approach can reconstruct network topologies in case predecessor's disconnection or defection has occurred.**

***Keywords*—Peer-to-Peer Streaming Networks; Resilience; Dynamic Reconfiguration**

I. INTRODUCTION

Traditional client-server based live streaming systems can be constructed easily and provide good performance, although it usually imposes high deployment and maintenance costs on the owner. P2P live 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 [1], scheduling algorithms [2], incentives [3], re-transmission, coding [1] and balancing topologies according to network motif [4]. However, since these methods presuppose static topology during streaming operations, they cannot address the problem of dynamic reconstruction in resilient P2P live streaming systems. Hence, we propose a reliable P2P live streaming system which reconstructs its own overlay autonomously. 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 motif.

This paper is organized as follows: we describe research background in Section 2, and present the definition of resilience, related works and our approach 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.

II. 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 live-streaming systems.

A. 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.

1) Client-server based streaming systems: In client-server based streaming systems [5], 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 [5] is 1 million dollars per day [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 [6][7] 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 live streaming has also the drawback of network topology becoming highly dynamic because of churn. Therefore, interruption of streaming and latency in segment delivery are frequently observed in P2P live streaming systems.

In order to address these problems, it is important to ensure that network is resilient. A definition of resilience is given by

 ISBN: 978-1-61208-357-5

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.

B. P2P Live-Streaming

As mentioned above, there are some problems in P2P live streaming systems. They can be roughly divided into the following three 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 live streaming, 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 networks, all nodes select a predecessor randomly. This causes an increase in the number of hops and connection concentration.

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.

III. Resilience in P2P streaming systems

To ensure resilience in P2P live 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 motif. Finally, we introduce our approach.

A. Approaches to ensure resilience

There are the following approaches to ensure resilience in P2P live 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.

B. Network motif

In P2P live streaming systems, topology imbalance poses a serious problem. In order to address this problem, a method of balancing topologies using network motif is proposed [4].

1) Network Motif: Network motif 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. In network motif, each node has at least one input or output.

Network motif can be used as an approach to compare the difference of network characteristics in biological studies, World Wide Web (WWW) or electric circuit networks [8].

2) Approach in related work: In the related work [4], 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.

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.

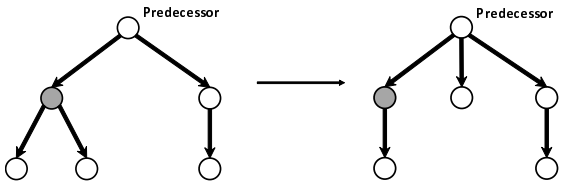

Fig. 1. Situation when predecessor has sufficient margin in load.

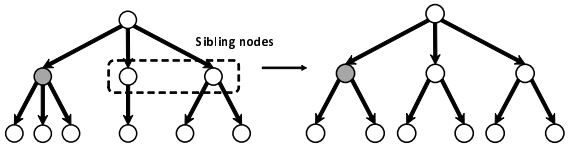

Fig. 2. Situation when sibling nodes have larger margin.

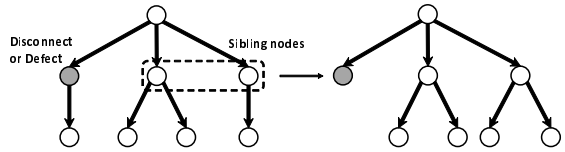

Fig. 3. Replacing predecessor.

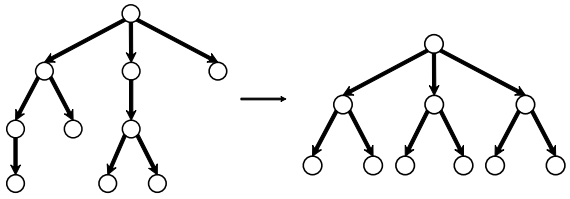

Fig. 4. Balancing topologies using network motif.

C. Our approach

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 [2][3][4] 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 this paper, we propose a new approach and evaluate it by simulation.

IV. System design

In this section, we describe our approach for ensuring resilience by reconfiguring topologies dynamically. This approach consists of three methods: observing neighbor node's state, selecting a predecessor and spare predecessors, and balancing topologies using network motif.

A. 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.

Figures 1 and 2 show examples of such a behavior. In these figures, each node is allowed to be connected with at most three nodes.

B. 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.

Figure 3 shows an example of this behavior.

Algorithm 1 Pseudo code of our approach

Input: $Neighbor_i, Parent, Child$;
  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

C. Balancing topologies using network motif

P2P live streaming is 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. [4].

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

Figure 4 shows an example of balancing topologies, and Algorithm 1 presents a pseudo code of our approach.

V. Evaluation

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

TABLE I. SIMULATION PARAMETERS.

Parameter	Value
Number of predecessors	1
Number of successors	3
Number of original senders	1
Number of pseudo segments	5
Playback time for pseudo segment	5000 ms
Simulation time	25000 ms

We compare “Normal Approach”, in which each node in a streaming topology connects to a predecessor randomly, with “Proposed Approach”, which corresponds to our approach. 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 three different perspectives: topology’s height and latency, connectivity and exchange steps for each node, and Topo-metric values.

For each node, Topo-metric value is defined in [4] 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 preferrable.

To evaluate our approach, we prepared a simulator which is executed on single computer. Some major parameters of the simulation are summarized in Table I.

A. Height and latency in produced topologies

Figure 5 shows the height of produced topologies, and Figure 6 shows their latency in a range from 10 to 100 nodes. In this experiment, we did the following actions randomly: addition of a new node, predecessor’s disconnection or predecessor’s defection. These results show the average of 10 executions for each experiment.

Figure 7 shows the change in topology’s height in case predecessor’s disconnection or defection has occurred. In this figure, these events occurred at approximately 2000, 10000 and 16000 milliseconds after the simulation had started.

From these results, we can conclude that both the height and latency in produced topologies are reduced using our approach. Furthermore, the height of topology increases after predecessor’s disconnection or defection, although it gradually decreases afterwards. Therefore, our approach can reconfigure the topologies dynamically in case of predecessor’s disconnection or defection, and decreases maximum number of hops.

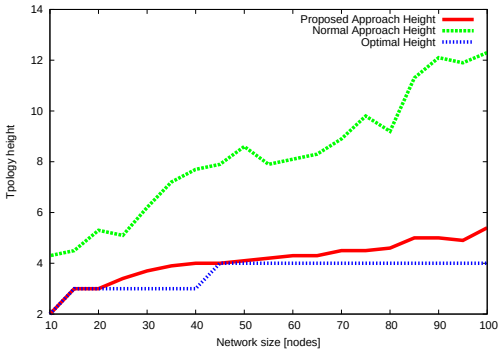

Fig. 5. Height of streaming topologies.

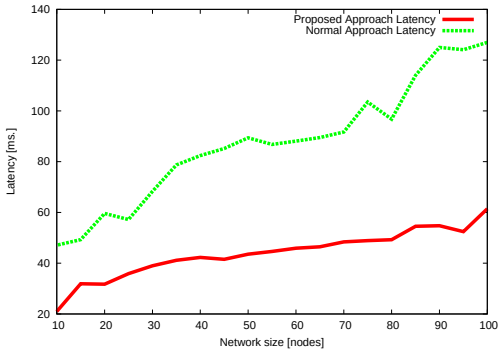

Fig. 6. Latency of streaming topologies.

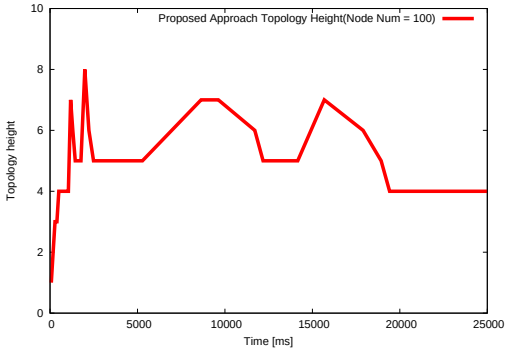

Fig. 7. Change of topology’s height in time.

B. Number of exchange steps and connectivity in produced topologies

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.

Figure 8 shows the average connectivity for each node in produced topologies, and Figure 9 shows the number of exchange steps for each node in produced topologies, in a range from 10 to 100 nodes. In this experiment, we did the following actions randomly: addition of a new node, predecessor’s disconnection or predecessor’s defection. These results show the average of 10 executions for each experiment.

Figure 10 shows the result of a dynamic change of the average connectivity for each node in case predecessor’s disconnection or defection has occurred. In this figure, these events occurred at approximately 2000, 10000 and 16000 milliseconds after the simulation had started.

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 in total it grows

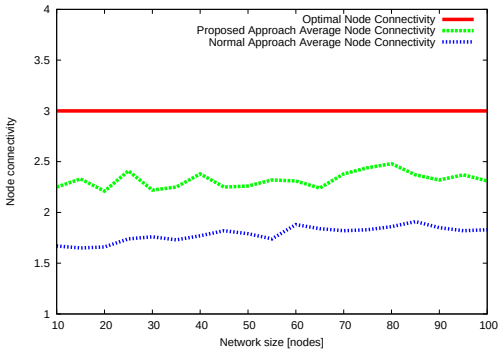

Fig. 8. Average node connectivity.

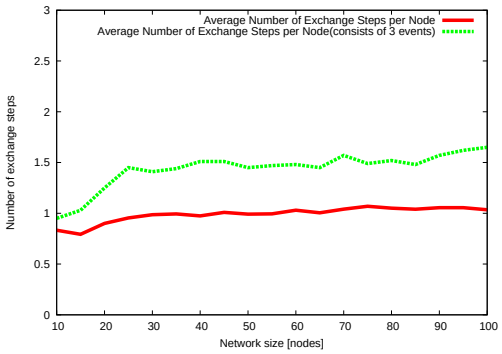

Fig. 9. Number of exchange steps per node.

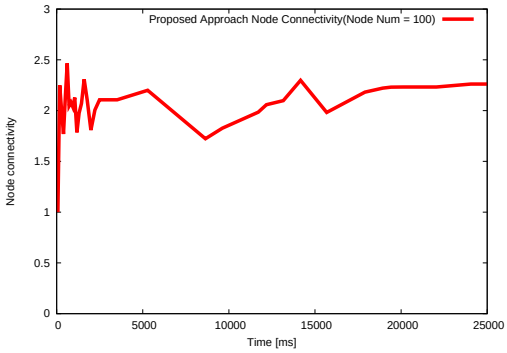

Fig. 10. Change of average node connectivity in time.

sublinearly with respect to network size, increasing from 1 to more than 1.5.

Furthermore, the connectivity of each node decreases after predecessor's disconnection or defection has occurred, however, it gradually increases afterwards. Therefore, our approach can reconfigure the topologies dynamically in case of predecessor's disconnection or defection, and achieves dynamic load balancing.

C. Topo-metric values in produced topologies

Figure 11 shows the Topo-metric value in produced topologies, in a range from 10 to 100 nodes. In this experiment, we did the following actions randomly: addition of a new node,

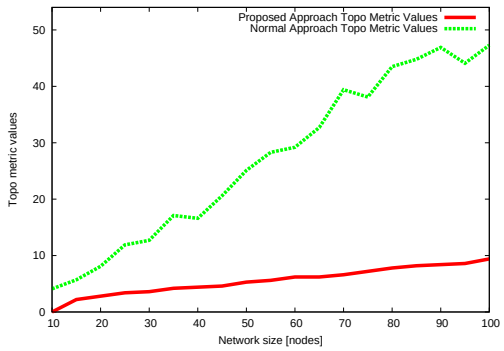

Fig. 11. Change of Topo-metric value wuth regard to network size.

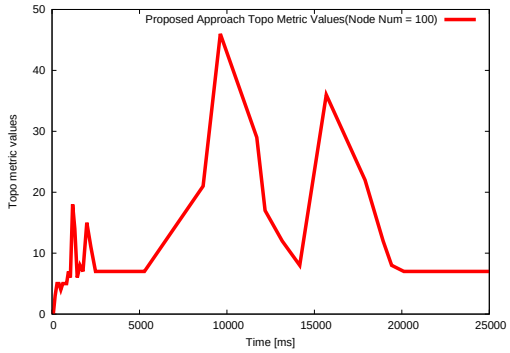

Fig. 12. Change of Topo-metric value in each time.

predecessor's disconnection or predecessor's defection. These results show the average of 10 executions for each experiment.

Figure 12 shows the result of a dynamic change of the Topo-metric value in case predecessor's disconnection or defection has occurred. Here, these events occurred at approximately 2000, 10000 and 16000 milliseconds after the simulation had started.

From these results, we can conclude that Topo-metric value in produced topologies is lowered using our approach. Furthermore, the Topo-metric value increases after predecessor's disconnection or defection, although gradually decreases afterwards. Therefore, our approach can reconfigure topologies dynamically in case of predecessor's disconnection or defection, and produces balanced topologies.

D. Change of Topo-metric values in produced topologies

In Figures 13 and 14, we have fixed two positions in the network where disconnection or defection occurs, and labeled them as low and high. In case of a low position, node is located 1 to 2 hops up from the leaf node of a tree, and in case of a high position node is located 1 hop down from the root node. In these figures, both events occurred at 6000 milliseconds after the simulation had started, and streaming topology consists of 100 nodes.

From these results, we can see that if a lower node was selected, the time to stabilize the Topo-metric value becomes shorter compared to selection of a higher node. This result is the same regardless of whether disconnection or defection has

 ISBN: 978-1-61208-357-5

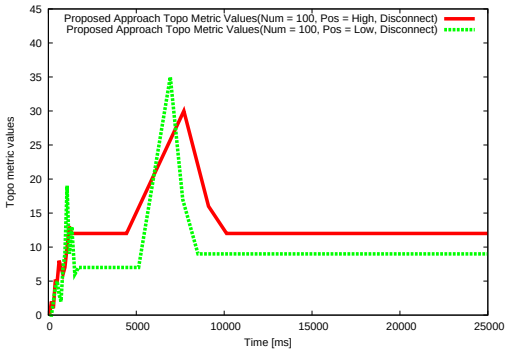

Fig. 13. Change of Topo-metric values in time (in case of predecessor's disconnection).

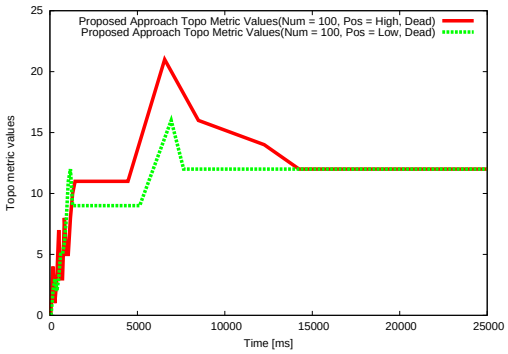

Fig. 14. Change of Topo-metric values in time (in case of predecessor's defection).

occured. From this we can conclude that the imbalance in the topology tends to appear if a higher node is disconnected or defected.

VI. Conclusion and Future work

This paper studied the resilience of P2P live streaming systems that comes as a result of dynamic reconfiguration of peer-to-peer network.

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. In this scenario, our approach greatly decreases height and latency in produced topologies, and provides an ability to dynamically reconfigure the network in case of predecessor's disconnection or defection.

Issues to address in further studies are as follows.

A. 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 live 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 [9]. Therefore, we need to improve our approach and simulator to model actual P2P live streaming systems.

B. 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 [10], and confirm the soundness of our approach.

References

[1] O. Abbound, K. Pussep, A. Kovacevic, K. Mohr, S. Kaune, and R. Steinmetz, "Enabling Resilient P2P Video Streaming: Survey and Analysis", Multimedia System, June, 2011, pp. 177–197.

[2] Y. Guo, C. Liang, and Y. Liu, "AQCS: Adaptive Queue-Based Chunk Scheduling for P2P Live Streaming", Proceedings of 7th International IFIP-TC6 Networking Conference Singapore, May, 2008, pp. 433–444.

[3] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang, "Using Layered Video to Provide Incentives in P2P Live Streaming", Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV, August, 2007, pp. 311–316.

[4] L. Krumov, A. Andreeva, and T. Strufe, "Resilient Peer-to-Peer Live-Streaming using Motifs", IEEE WoWMoM, June, 2010, pp. 1–8.

[5] YouTube, http://www.youtube.com/. [retrived: June, 2014]

[6] BitTorrent, http://www.bittorrent.com/. [retrived: June, 2014]

[7] J. Xiong and R. R. Choudhury, "PeerCast: Improving Link Layer Multicast through Cooperative Relaying", IEEE INFOCOM, April, 2011, pp. 2939–2947.

[8] C. Brandt and J. Leskovec, "Status and friendship: mechanisms of social network evolution", WWW Companion '14 Proceedings of the companion publication of the 23rd international conference on World wide web companion, April, 2014, pp. 229–230.

[9] Y. Yoshida, M. E. Haque, N. Matsumoto, and N. Yoshida, "Efficient Decentralized Evaluation of Node Trustworthiness in Peer-to-Peer Networks", Proceedings of International Conference on Computer Engineering and Technology 2009, July, 2009, pp. 177–179.

[10] HTTP Live Streaming Resources - Apple Developer. https://developer.apple.com/streaming/. [retrived: June, 2014]

 ISBN: 978-1-61208-357-5