An OpenFlow-Assisted Framework for Peer-to-Peer Live Streaming

Jianwei Zhang*
Shandong Provincial Key Laboratory of Computer Networks
Shandong Computer Science Center
(National Supercomputer Center in Jinan)
Jinan, China
Email: janyway@outlook.com

Chunling Yang
Information Technology Center
Zhejiang University
Hangzhou, China
Email: chly@zju.edu.cn

***Abstract*—In this paper, we propose an OpenFlow-assisted framework for P2P (peer-to-peer) live streaming, which incorporates an OpenFlow-assisted architecture and an OpenFlow-assisted algorithm. The architecture combines the characteristics and advantages of both BitTorrent system and OpenFlow network. The algorithm can dynamically calculate the path and corresponding available bandwidth for each flow according to the periodical neighbor selection results passed from P2P system to OpenFlow controller. Simulation results show that, the proposed framework performs much better than the traditional OSPF-based method in terms of the playback probability, especially in overloaded network.**

***Keywords*-OpenFlow; software-defined network (SDN); peer-to-peer (P2P); live streaming; playback.**

I. Introduction

Peer-to-peer (P2P) networking proved to be the most efficient way for large-scale live streaming during past decades. However, limited by controllability and manageability of the core routers in traditional networks, existing research works mainly focus on the improvement of application layer topology construction, peer selection and chunk selection algorithms. Very few studies [1]–[3] have focused on the transport layer or network layer.

In traditional networks, Open Shortest Path First (OSPF) protocol is one of the most widely deployed Interior Gateway Protocols (IGPs). Link weights can be set to achieve the goals of traffic engineering [4]. A flow can be evenly split across different paths only when their costs are equal [5]. By contrast, Software-Defined Networking (SDN) and OpenFlow (the most popular instance of SDN) feature the centralized control, open northbound interface, programmable network [6], and are able to split a flow across multiple paths in the manner specified by the controller in advance. Therefore, how to realize the full potential of the OpenFlow network has great theoretical and practical significance for the optimization of P2P live streaming.

Yang et al. [9] propose an OpenFlow-based streaming framework, which supports Scalable Video Coding (SVC) and provides flexible flow identification, processing, and management. Egilmez et al. [10], [11] present an analytical framework for optimization of dynamic Quality of Service (QoS) for SVC videos over OpenFlow networks. They apply the Lagrangian Relaxation Based Aggregated Cost (LARAC) algorithm to the video layer of high QoS level, while the default routing algorithm is applied to the video layer of low QoS level. Through emulation, Vicino et al. [14]–[16] show that the BitTorrent system will achieve lower propagation efficiency when the logical topology of SDN changes periodically, which contradicts the general view that changes in the logical topology would not affect applications. They emphasize the necessity of adapting applications to SDNs carefully before they are deployed.

Unlike above works, this paper focuses on designing an OpenFlow-assisted framework for more efficient P2P live streaming. Specifically, the contributions are as follows.

- We propose an OpenFlow-assisted framework for mesh-based P2P live streaming, which incorporates an OpenFlow-assisted architecture and an OpenFlow-assisted algorithm.
- The architecture combines characteristics and advantages of both BitTorrent system and OpenFlow network. The algorithm can dynamically calculate the path and corresponding available bandwidth for each flow.
- We perform simulation to compare the OpenFlow-assisted framework with the traditional OSPF-based method in terms of the playback probability, and analyze the impacts of the concurrent uploading number of P2P nodes and the average load of the network.

II. The OpenFlow-Assisted Framework

The proposed OpenFlow-assisted framework consists of an OpenFlow-assisted architecture and an OpenFlow-assisted algorithm.

A. The OpenFlow-Assisted Architecture

In Fig. 1, the right side represents a typical BitTorrent-like live streaming system, where there is a centralized tracker server maintaining information about streaming progresses of ordinary P2P nodes. When a P2P node joins the overlay, it immediately contacts the tracker, and requests some randomly selected nodes already participating in the streaming. The left side of Fig. 1 represents a typical OpenFlow network, which provides the following assistance [10], [11].

- Topology management. Detecting and maintaining the network topology [10], including the OpenFlow

978-1-5090-5957-7/17/$31.00

switches as well as the streaming server and the P2P nodes.

- Network monitoring [12], [13]. Probing link capacities and estimating the amount of real time traffic on each link via counters associated with tables, ports, queues, and table entries in OpenFlow switches.
- Routes calculation. Calculating optimized routes and bandwidth allocation results for live streaming flows. To realize these results, the meter table and group table should be utilized, and the flows should be able to be split precisely in the switches [5], [7], [8].
- Flow tables sending and updating. Sending flow tables to OpenFlow switches and updating them when required to realize the routes calculation results.
- Information exchange with the tracker server. Retrieving neighbor selection results periodically from the tracker server in P2P system [19].

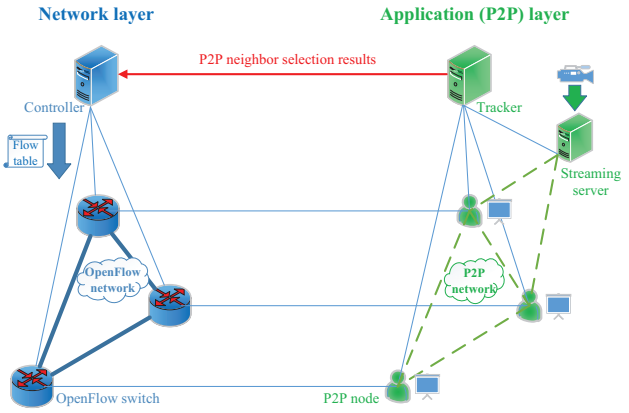

Figure 1. The OpenFlow-assisted architecture.

B. The OpenFlow-Assisted Algorithm

The OpenFlow-assisted algorithm is formulated as Algorithm 1. U_i is the upload bandwidth of P2P node N_i. $S(N_i)$ represents the switch N_i is connected to. $|S(N_i)|$ stands for the number of P2P nodes (N_i inclusive) connected to the same switch as N_i.

In each unchoking round, a source P2P node N_0 selects at most m neighboring nodes $\{N_1, N_2, ..., N_m\}$ as the targets to push chunks to. The OpenFlow controller periodically retrieves the neighbor selection results from the tracker, and determines the source switch S_0 that N_0 is connected to and the target switch set $\{S_1, S_2, ..., S_k\}$ ($k \leq m$) that the target P2P node set is connected to. Then, the optimization can be easily transformed into a standard *single-source multi-sink maximum flow problem* at the network layer. For each target switch S_i, the path $P(S_i)$ and the available bandwidth $B(S_i)$ along the path can be calculated.

For the application layer, there are two special cases. If more than one target P2P node is connected to the same OpenFlow switch, the available bandwidth along the path is evenly partitioned. If there exist some target P2P nodes connected to the same OpenFlow switch as N_0,

Algorithm 1 The OpenFlow-assisted algorithm

```
Controller ← detect and maintain topology
for each unchoking round do
  Select at most m neighboring nodes to upload to:
  N_0 → {N_1, N_2, ..., N_m}
  Controller ← Tracker ← P2P neighbor selection
  Controller ← monitor the real time traffic
  Solve the maximum flow problem:
  S_0 → {S_1, S_2, ..., S_k}
  for switch S_i, i ∈ [1, k] do
    path(S_i) = P(S_i)
    available bw(S_i) = B(S_i)
  end for
  for node N_i, i ∈ [1, m] do
    if S(N_i) ≠ S(N_0) then
      path(N_i) = P(S(N_i))
      available bw(N_i) = min(U_0/m, B(S(N_i))/|S(N_i)|)
    end if
  end for
  Controller → send and update flow tables
end for
```

the available bandwidth will not be limited by any link between switches.

III. Performance Evaluation

We integrate a network layer optimization module into a discrete event-driven simulator to conduct the experiments. The network layer topology is the well-known Abilene network (Fig. 2), which has 12 nodes (OpenFlow switches) and 30 links in total. The application layer topology is a random graph with 12 nodes in total and an average degree 6 to offer sufficient connectivity. All link capacities are set to $L = 1000$ KB/s, so that it is quite possible for the network link to become the bottleneck. For simplicity, we only attach one P2P node to each OpenFlow switch. The streaming server is linked to OpenFlow switch 1. In each time slot, the streaming server pushes a new media chunk into the overlay. The chunk size is $C = 256$ KB. The upload bandwidth of P2P nodes is set to mC KB/s.

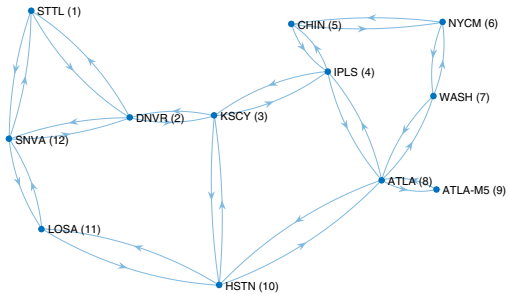

Figure 2. Abilene network. There are 12 nodes and 30 links. Each node represents an OpenFlow switch.

During the simulation, the real time traffic generator is set to: $traffic = rand(1) \times (L - f) + f$ KB/s,

where $rand(1)$ generates a single uniformly distributed random number in the interval (0,1), and f is used to adjust the average link load. In the OSPF-based method, the link weight is assumed to be inversely proportional to the link capacity, and we only consider the single-path routing mode, which is primarily used and has a small overhead [17]. The playback probability is adopted as the QoE (Quality of Experience) performance metric [18]. It is defined as the probability that a chunk is present in corresponding buffer position, and indicates whether a chunk can be played out in time.

We can see from Figs. 3–6 that, for both two schemes, the playback probability increases with the increasing of the concurrent uploading number, and with the decreasing of the average link load. The OpenFlow-assisted framework outperforms the OSPF-based method in all considered parameters settings, especially in the scenario of high link load ($f = 500$ KB/s). One of the most important reasons is that, it is impossible for the traditional OSPF-based routing scheme to reroute each flow dynamically according to the real time traffic. The OSPF-based method is more sensitive to m than the OpenFlow-assisted framework. The OpenFlow-assisted framework can gain benefit from the multi-path routing in any case (even when $m = 1$). While for the OSPF-based method, this only occurs under the condition that more than one path has the same cost; when multi-path routing is disallowed, this is true only when $m > 1$.

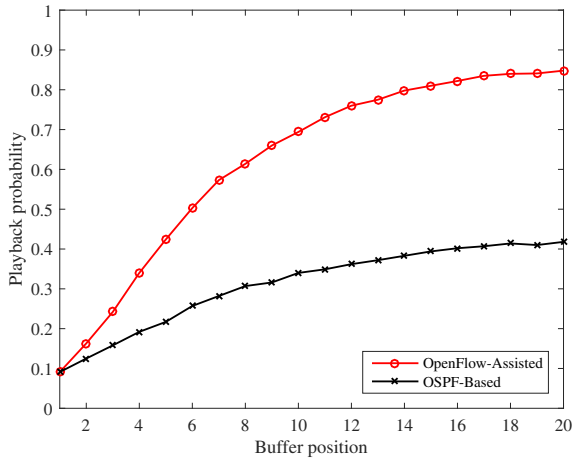

Figure 3. Playback probability versus buffer position. Small concurrent uploading number, low average link load ($m = 1$, $f = 300$ KB/s).

IV. Conclusion

In this paper, we propose an OpenFlow-assisted framework for P2P live streaming, which incorporates an OpenFlow-assisted architecture and an OpenFlow-assisted algorithm. The architecture combines the characteristics and advantages of both P2P/BitTorrent system and OpenFlow network, so that the research space can be greatly broadened. The algorithm can dynamically calculate the path and available bandwidth for each flow. Also, we compare the OpenFlow-assisted framework with the traditional

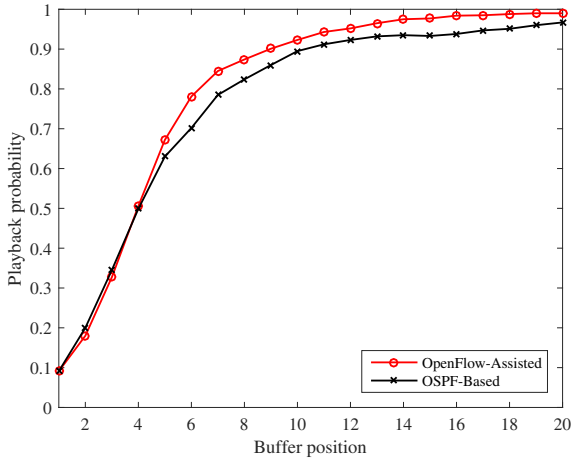

Figure 4. Playback probability versus buffer position. Large concurrent uploading number, low average link load ($m = 3$, $f = 300$ KB/s).

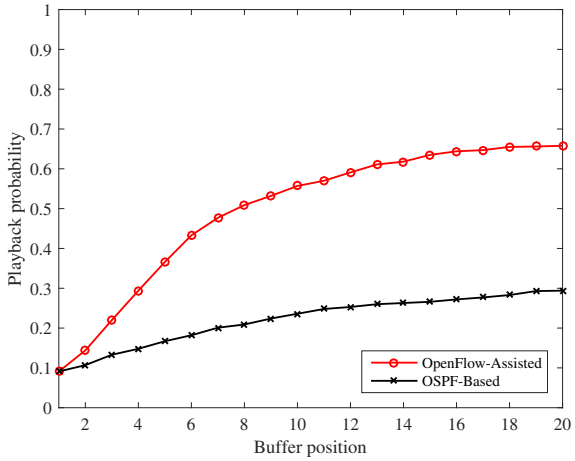

Figure 5. Playback probability versus buffer position. Small concurrent uploading number, high average link load ($m = 1$, $f = 500$ KB/s).

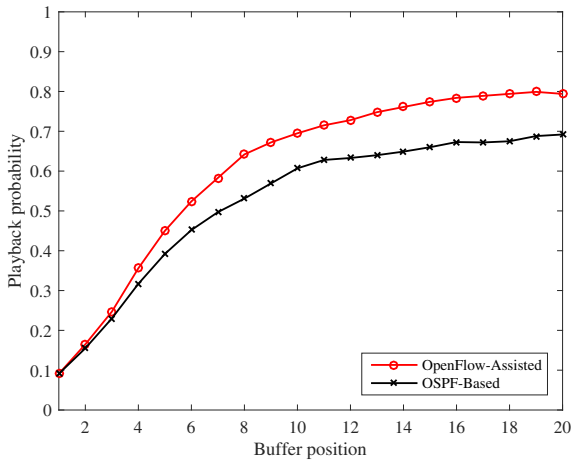

Figure 6. Playback probability versus buffer position. Large concurrent uploading number, high average link load ($m = 3$, $f = 500$ KB/s).

OSPF-based method in terms of the playback probability

by simulation. Simulation results show that, the proposed framework performs much better than the OSPF-based method, especially in overloaded network.

As demonstrated in this paper as well as other literature, the routing strategies in the OpenFlow controller and the node/chunk selection strategies in the P2P nodes deeply influence each other, and may constitute a closed loop. In future work, we will further investigate the interplay between them by game theory and control theory.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grant No. 61472230, the Shandong Provincial Natural Science Foundation of China under Grant No. ZR2015YL021, ZR2015YL019, ZR2014YL042 and ZR2015FL025, and the Youth Science Funds of Shandong Academy of Sciences under Grant No. 2016QN004.

REFERENCES

[1] C. Testa and D. Rossi, "Delay-based congestion control: flow vs. BitTorrent swarm perspectives," Computer Networks, vol. 60, 2014, pp. 115–128.

[2] C. Testa, D. Rossi, A. Rao, and A. Legout, "Data plane throughput vs control plane delay: experimental study of BitTorrent performance," 2013 13th IEEE International Conference on Peer-to-Peer Computing. IEEE, 2013, pp. 1–5.

[3] W. Yang and N. Abu-Ghazaleh, "GPS: a general peer-to-peer simulator and its use for modeling BitTorrent," 2005 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems. IEEE, 2005, pp. 425–432.

[4] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," Computer Networks, vol. 71, 2014, pp. 1–30.

[5] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in SDN/OSPF hybrid network," 2014 22nd IEEE International Conference on Network Protocols (ICNP). IEEE, 2014, pp. 563–568.

[6] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," Proceedings of the IEEE, vol. 103, no. 1, 2015, pp. 14–76.

[7] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," INFOCOM 2013. IEEE, 2013, pp. 2211–2219.

[8] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," IEEE/ACM Transactions on Networking, vol. 3, no. 4, 1995, pp. 365–386.

[9] J. Yang, E. Yang, Y. Ran, and S. Chen, "SDM^{2}Cast: an OpenFlow-based, software-defined scalable multimedia multicast streaming framework," IEEE Internet Computing, vol. 19, no. 4, 2015, pp. 36–44.

[10] H. E. Egilmez, S. Civanlar, and A. M. Tekalp, "An optimization framework for QoS-enabled adaptive video streaming over OpenFlow networks," IEEE Transactions on Multimedia, vol. 15, no. 3, 2013, pp. 710–715.

[11] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "OpenQoS: an OpenFlow controller design for multimedia delivery with end-to-end quality of service over software-defined networks," 2012 Asia-Pacific Signal & Information Processing Association Annual Summit and Conference. IEEE, 2012, pp. 1–8.

[12] K. Phemius and M. Bouet, "Monitoring latency with OpenFlow," 2013 9th International Conference on Network and Service Management. IEEE, 2013, pp. 122–125.

[13] D. Raumer, L. Schwaighofer, and G. Carle, "MonSamp: a distributed SDN application for QoS monitoring," 2014 Federated Conference on Computer Science and Information Systems. IEEE, 2014, pp. 961–968.

[14] D. Vicino, C. H. Lung, G. Wainer, and O. Dalle, "Evaluating the impact of software-defined networks reactive routing on BitTorrent performance," Procedia Computer Science, vol. 34, 2014, pp. 668–673.

[15] D. Vicino, C. H. Lung, G. Wainer, and O. Dalle, "Investigation on software-defined networks' reactive routing against BitTorrent," IET Networks, vol. 4, no. 5, 2015, pp. 249–254.

[16] M. K. Guraya, R. S. Bajwa, D. Vicino, and C. H. Lung, "The assessment of BitTorrent's performance using SDN in a mesh topology," 2015 6th International Conference on Network of the Future. IEEE, 2015, pp. 1–3.

[17] M. Wang, C. W. Tan, W. Xu, and A. Tang, "Cost of not splitting in routing: characterization and estimation," IEEE/ACM Transactions on Networking, vol. 19, no. 6, 2011, pp. 1849–1859.

[18] J. Zhang, C. Yang, and X. Zhang, "A high-bandwidth live streaming model in mesh-based peer-to-peer networks," IEEE Communications Letters, vol. 20, no. 12, 2016, pp. 2390–2393.

[19] A. Al-Hamra, N. Liogkas, A. Legout, and C. Barakat, "Swarming overlay construction strategies," 18th Internaton-al Conference on Computer Communications and Networks (ICCCN). IEEE, 2009, pp. 1–6.