

MaxStream: SDN-based Flow Maximization for Video Streaming with QoS Enhancement

Abolfazl Samani and Mea Wang

Department of Computer Science, University of Calgary

{abolfazl.samani, meawang}@ucalgary.ca

Abstract—Along with the increasing demand for video streaming, network service providers and video content providers are challenged to maximize their service in terms of both quantity and quality while guaranteeing Quality-of-Service (QoS). In this paper, we propose MaxStream, a SDN-based flow maximization framework. The framework exploits the global view of the network available at the SDN controller to formulate integer multi-commodity flow problems with objectives to maximize the number of streaming sessions accommodated in a network to improve providers revenue (quantity of the service) and to maximize bandwidth provisioning for QoS enhancement (quality of the service). Our simulation results confirm that on average, MaxStream accepts 13% more streaming sessions and offers 50% increase in bandwidth provisioning compared to variations of the widest-shortest path algorithm.

I. INTRODUCTION

Internet video traffic will constitute about 82% of all consumer Internet traffic by 2021, up from 73% in 2016 [1]. In order to provide a good viewing experience, video streaming services have stringent requirements on bandwidth and delay. It is essential and challenging to provide a Quality-of-Service (QoS) guarantee for video streaming services [2]. Software-Defined Networking (SDN) is an emerging networking paradigm realizing QoS [3]. The concept decouples network control and data forwarding into the control plane and data plane, respectively. Such decoupling leads to greater flexibility in network management. More specifically, networking devices such as routers and switches become simple forwarding devices in the data plane, while the control plane determines the forwarding rules according to the network status centrally collected by the controller. In contrast to conventional traffic management, network traffic is managed at the flow level (from source to destination) instead of at the per-hop level (from one host to the next). When admitting a flow, the control plane determines the path from source to destination and may also reserve resources along this path. This design leads to global view and management of network traffic, which makes it possible to coordinate QoS guarantee and forwarding decisions at large scale [4], [5], [6].

Towards end-to-end QoS guarantee, Civanlar *et al.* prioritize the delivery of the base layer of an SVC (Scalable Video Coding) video with QoS guarantee in terms of packet loss and end-to-end delay [7]. This guarantees a minimum level of quality. The framework routes best-effort traffic using the shortest path, while the QoS traffic routed on QoS paths using the linear programming algorithm. HiQoS uses the Equal-Cost Multipath Routing (ECMP) algorithm along with queuing to ensure QoS guarantees for the QoS traffic [8]. ECMP calculates multiple end-to-end routes that meet certain QoS requirements. QoS provisioning can also be achieved through packet scheduling and priority queue management. The first framework in this category is QoSFlow, which reorders packets into stochastic fairness queues (SFQ) so that certain flows are treated with

high priority [9]. This framework is implemented in the data plane and is tested with up to three TPLink 1043ND switches.

Providing the flexible QoS control in SDN, we seek to address two open challenges in video streaming service provisioning: service maximization and QoS enhancement. Service maximization refers to the quantity of video streaming services, *i.e.*, maximizing the number of streaming sessions served. QoS enhancement aims to improve the quality of video streaming services – a primary factor impacting viewing experience. Although video quality is ultimately measure by the visual perception, in terms of transmission, it is reflected by the bitrate (bps). Thus, bandwidth guarantee is essential to ensure a high quality viewing experience. To this end, the closest existing attempt is the widest-shortest path (WSP) algorithm, a well-known algorithm for finding QoS-rich end-to-end paths with desired bandwidth [10]. WSP determine the widest (most bandwidth reservation) paths among the set of shortest paths for a streaming session. The WSP is used in [11] as the QoS-aware routing algorithm. However, WSP only maximizes quality, but not quantity of the service.

Towards optimization of both quality and quantity, we propose MaxStream, an SDN-based flow maximization framework, that determines the maximum number of streaming sessions that can be accommodated in a network and also provides QoS enhancement in terms of supported bitrate. This framework has two modules; both are integer multi-commodity flow (IMCF) formulations. The first module, Most-flow IMCF, determines the maximum set of streaming sessions that can be accommodated in an SDN network. The second module, Maximum IMCF, maximizes bandwidth provisioning for the selected streaming sessions by Most-flow IMCF. In our performance analysis, we show that with Most-flow IMCF, we can accept 13% more flows on average, in comparison to widest-shortest algorithm. Maximum IMCF offers 50% increase on average in terms of cumulative bitrate.

The remainder of this paper is organized as follows. Sec. II presents the proposed MaxStream framework, followed by the performance evaluations in Sec. III. Sec. IV concludes the paper.

II. MAXSTREAM: A FRAMEWORK FOR FLOW MAXIMIZATION AND QOS ENHANCEMENT

An SDN network G consists of an SDN controller and a set of nodes V that can be end hosts (*e.g.*, servers or clients) or switches. These network devices are connected through a set of links E with certain weights representing bandwidth capacity. The link capacity is denoted by $c_{(u,v)}$, where $(u,v) \in E$. Since each flow has a direction from the source to destination, the network is modeled as a directed graph, thus $(u,v) \neq (v,u)$. In MaxStream, multimedia streaming sessions are modeled as a set of k commodities, where each streaming session is defined by (s_i, t_i, r_i) , with s_i and t_i as the video server and

the video playback client of session i , respectively. Moreover, r_i is the required bitrate for the corresponding streaming session. For practicality, we assume that each flow (*i.e.*, each streaming session) takes a single path from the server to the client. Thus, determining paths for the k streaming sessions in the SDN network is an integer multi-commodity flow problem – an NP-hard problem [12]. This inspired us to model the the flow maximization problem and the QoS enhancement problem as two integer multi-commodity flow (IMCF) problems. In this section, we present the detail of the two problem formulations as well as the integration of the two formulations in MaxStream implementation.

A. Most-flow IMCF: flow maximization

One of the objective of MaxStream is to maximize the number of supported streaming sessions in the SDN network for better resource utilization. To do so, we define the Most-flow IMCF problem as follows.

$$\begin{aligned}
 &\text{maximize} && \sum_{i=1}^k a_i \\
 &\text{subject to} && (1) \sum_{i=1}^k f_{i,u,v} \cdot r_i \cdot a_i \leq c_{u,v} \quad \forall (u,v) \in E \\
 &&& (2) \sum_{v \in V} f_{i,u,v} - \sum_{v \in V} f_{i,v,u} = 0 \cdot a_i \quad \forall u \neq s_i, t_i, \forall i \\
 &&& (3) \sum_{v \in V} f_{i,s_i,v} - \sum_{v \in V} f_{i,v,s_i} = 1 \cdot a_i \quad \forall i \\
 &&& (4) \sum_{v \in V} f_{i,v,t_i} - \sum_{v \in V} f_{i,t_i,v} = 1 \cdot a_i \quad \forall i \\
 &&& (5) f_{i,u,v} \in \{0, 1\} \quad \forall i \in S, \forall (u,v) \in E \\
 &&& (6) a_i \in \{0, 1\} \quad \forall i \in S
 \end{aligned}$$

In this formulation, we introduce a binary selector for each streaming session a_i . This selector is set to zero if a flow is rejected and is set to one if a flow is accepted in the solution. The objective here is to maximize the number of selected sessions. We also define a binary variable $f_{i,u,v}$, which is a selector indicating whether a link (u,v) is selected in the path for flow i . The first constraint ensures that the sum of all flows for all streaming sessions routed over a link should not exceed its capacity. The second constraint is to enforce the flow conservation on intermediate nodes, *i.e.*, the amount of flow for each of the streaming sessions i entering an intermediate node u should be the same as the amount that exits node u . The third constraint is to enforce the flow conservation at the source node. Thus, flow of each session i should exit its server node completely. The fourth constraint is to enforce the flow conservation at the destination node. Thus, flow of each session i should enter its client node completely. The last two constraints are the integral constraints on the binary variables. The solution for the problem is having an assignment for the variables $f_{i,u,v}$ and a_i that maximizes the number of streaming sessions while satisfying the constraints.

B. Maximum IMCF: QoS Enhancement

Since there is often spare bandwidth in the network, the videos may potentially be streamed at a higher quality. Such

a QoS enhancement consequently improves customer satisfaction level. For this reason, we are interested in maximizing the bandwidth reservation for the accepted streaming sessions. To accomplish this, we introduce a new problem formulation, the Maximum IMCF problem, defined as follows.

$$\begin{aligned}
 &\text{maximize} && \sum_{i=1}^k j_i \\
 &\text{subject to} && (1) \sum_{i=1}^k f_{i,u,v} \cdot j_i \leq c_{u,v} \quad \forall (u,v) \in E \\
 &&& (2) \sum_{v \in V} f_{i,u,v} - \sum_{v \in V} f_{i,v,u} = 0 \quad \forall u \neq s_i, t_i, \forall i \\
 &&& (3) \sum_{v \in V} f_{i,s_i,v} - \sum_{v \in V} f_{i,v,s_i} = 1 \quad \forall i \\
 &&& (4) \sum_{v \in V} f_{i,v,t_i} - \sum_{v \in V} f_{i,t_i,v} = 1 \quad \forall i \\
 &&& (5) f_{i,u,v} \in \{0, 1\} \quad \forall i \in S, \forall (u,v) \in E \\
 &&& (6) j_i \geq r_i, \quad \forall i \in S
 \end{aligned}$$

In this formulation, we introduce a new variable j_i for each streaming session, representing the bandwidth reservation for streaming session i . Constraints (1) - (5) are the same as in the Most-flow IMCF formulation, with two differences. First, we replaced r_i with j_i since we need to check the link capacity constraint based on the actual bandwidth being used by the flows. Second, we dropped selector a_i because the input flows to this question are the selected sessions by Most-flow IMCF. The last constraint is to ensure that the bandwidth reserved for each session meets the bitrate of the session. In summary, this problem takes two inputs: the graph representation of the SDN network and the set of accepted flows obtained from solving the Most-flow IMCF problem. The output of the problem are a set of selected paths (*i.e.*, an assignment for the variables $f_{i,u,v}$) that offer the highest aggregated bandwidth among all streaming sessions and bandwidth allocation j_i for each session.

The extra allocated bandwidth for each session may be utilized by the streaming protocols in different ways. For example, the bitrate of a streaming session may be updated to include the extra allocated bandwidth. This bitrate may be recommended to the streaming protocol. For an adaptive streaming protocol, the extra allocated bandwidth may be detected and utilized automatically. The problem of how the extra allocated bandwidth should be utilized by the streaming protocol is orthogonal to this work.

III. PERFORMANCE EVALUATION

In this section, we evaluate the performance of MaxStream. We utilize Mininet [13] as the SDN network emulator, with Apache servers on virtual hosts to emulate HTTP-based streaming servers, and DASH client on other virtual hosts to emulate streaming clients. We adapted Open vSwitch [14] module (supporting OpenFlow) to simulated SDN switches. For the SDN controller, we choose Ryu [15] not only for its component-based architecture, but also for its easy configuration of queue and QoS metrics on SDN switches compared to other controllers. We run all experiments on a MacBook Pro

with 2.7 GHz Intel Core i5 CPU and 8 GB of RAM. Due to Mininet's limit in scalability, by default, we bypass the actual bandwidth reservation in our experiments. This allows us to scale the experiment with more streaming sessions. We enable the bandwidth reservation module to verify the feasibility of MaxStream, but results are omitted here due to page limit. We develop MaxStream as a module that can be installed on any SDN controller. The IMCF problems in MaxStream are solved by linear programming relaxation (LP relaxation) and branch-and-bound algorithm. Due to page limit, details of the solver is also omitted.

For comparison purpose, we also implement Widest Shortest Path (WSP), Largest-First Widest Shortest Path (LFWSP), and Smallest-First Widest Shortest Path (SFWSP). For WSP, streaming sessions are ordered randomly. For LFWSP, the streaming sessions are sorted from the largest in bitrate to the smallest. For SFWSP, the streaming sessions are sorted from the smallest in bitrate to the largest. These algorithms tend to segment the bandwidth on links more, which will greatly reduce the cumulative bitrate of streaming sessions in the network. All three algorithms compute the widest-shortest path starting from the first streaming session in the set and reject a streaming session when it cannot find a path for it.

We are interested in the performance of MaxStream as the network scales as well as its effectiveness in supporting various streaming sessions. In our experiment, we collect the number of accepted streaming sessions (to verify the service maximization), the cumulative bitrate among all sessions (to verify the QoS enhancement). All results reported are the average calculated over ten runs of the experiment.

A. Effect of the Network Size

The first experiment is to see the effect of the network size. In this experiment, we vary the number of switches from 5 to 25 in intervals of 5. In general, the number of edges is twice the number of nodes, except for the case with 5 nodes, in which only 6 edges are used (to avoid making the network fully connected). The number of accepted sessions is illustrated in Fig. 1(a). On average, MaxStream (results from Most-flows IMCF) accepts 13.3% more sessions than WSP and its variations do. SFWSP accepts more sessions than WSP and LFWSP because it starts from the smallest sessions first. In contrast, LFWSP accepts the least because it starts from the largest sessions. WSP-based algorithms all compute a set of shortest paths and select a widest path among the shortest paths. The cumulative bitrate for each algorithm is shown in Fig. 1(b). The cumulative bitrate by Most-flows IMCF is on average 11% more than what WSP-based algorithms offer. Maximum IMCF is much more complex than Most-flows IMCF because it checks many possible bitrate for the sessions. For practicality, we limit its running time to 20 seconds. Thus, the solution is not yet fully optimized. Even so, the cumulative bitrate of Maximum IMCF is 50% more than that of WSP-based algorithms.

B. Effect of the Streaming Sessions

Next, we want to show the effect of the number of streaming sessions. In a network with 15 switches and 30 edges, the number of streaming sessions accepted by any algorithm

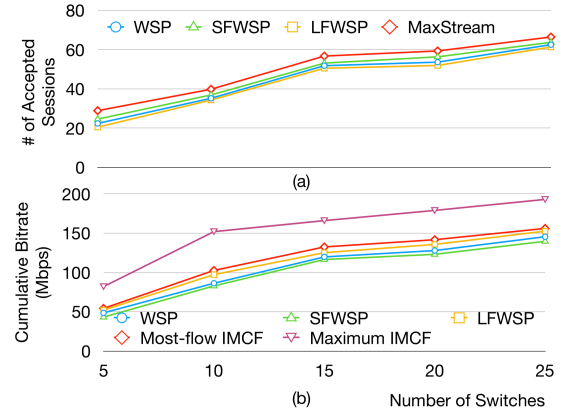


Fig. 1. Performance comparison when varying the number of switches

depends on the bitrate of the streaming sessions. Thus, the lower the bitrate, the more sessions will be accepted. In order to approximate the network capacity, we divide the bitrate range into three intervals: 1-3 Mbps (low bitrate), 4-6 Mbps (medium bitrate), and 7-9 Mbps (high bitrate). For low-bitrate sessions, all algorithms accept at least 30 sessions, but start to behave differently when the number of sessions grows beyond 30. Thus, for low-bitrate sessions, we scale the number of sessions from 30 to 60 (the maximum number that can be accommodated in the network). For medium-bitrate sessions, all algorithms accept at least 5 sessions, but start to behave differently when the number of sessions grows beyond 5. Thus, for medium-bitrate sessions, we scale the number of sessions from 5 to 35 (the maximum number that can be accommodated in the network). For high-bitrate sessions, we scale the number of sessions from 5 to 25 (the maximum number that can be accommodated in the network).

Figures 2(a), 2(b), and 2(c) illustrate the number of accepted sessions as the number of sessions increases for low, medium, and high-bitrate sessions, respectively. MaxStream, on average, accepts 4%, 11%, and 13% more sessions for the low, medium, and high bitrate cases, respectively, compared to WSP-based algorithms. As a result, the cumulative bitrate is also higher than the WSP-based algorithms, as illustrated in Figures 3(a), 3(b) and 3(c). On average, Most-flows IMCF improves the cumulative bitrate by 4.5%, 10.5% and 16% for low, medium and high-bitrate sessions, respectively, while the improvements are 95%, 47.5%, and 26% for Maximum IMCF.

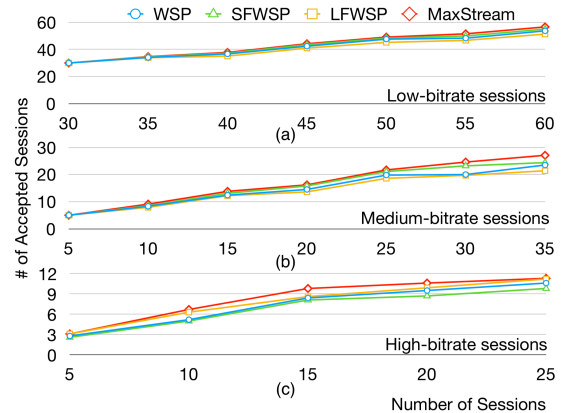


Fig. 2. Impact of session size on the number of accepted sessions

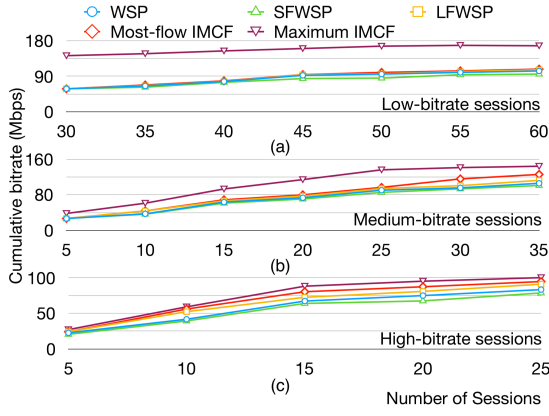


Fig. 3. Impact of session size on the cumulative bitrate

C. Heterogeneous Streaming Sessions

We wrap up the experiment with a more general, close-to-reality case where the streaming sessions have diverse bitrates. In this experiment, we fix the number of sessions to 30. We vary the number of each type of (low/medium/high bitrate) session from 5 to 20. The combination is written in the form X-Y-Z, where X, Y, and Z are the number of low, medium, and high bitrate sessions, respectively. As illustrated in Fig. 4, MaxStream on average accepts 17% more sessions than the others do. According to Fig. 5, on average, Most-flows IMCF and Maximum IMCF offer 30.5% and 70% higher cumulative bitrate than WSP-based algorithms do, respectively.

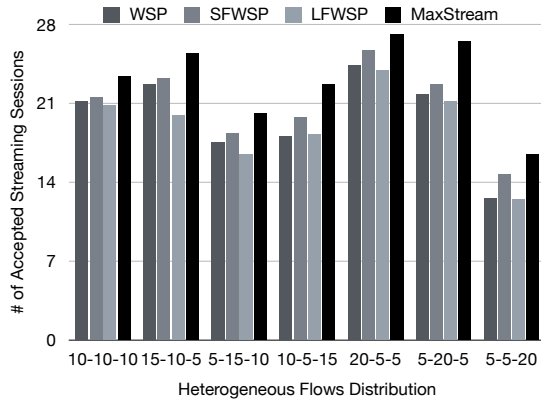


Fig. 4. Number of accepted sessions for different combinations of bitrates

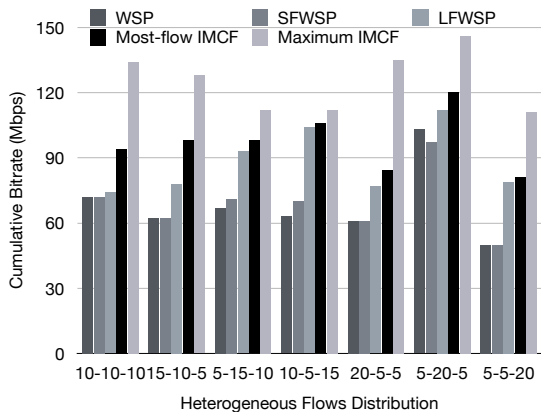


Fig. 5. The cumulative bitrate for all accepted sessions for different combinations of bitrates

IV. CONCLUSION

In this paper, we propose MaxStream, a new framework for determining the optimal solution for streaming service in terms of the number of supported streaming sessions and the achievable bitrate. The framework is based on two integer multi-commodity flow problem formulations: Most-flows IMCF for selecting the maximum set of streaming sessions and Maximum IMCF for determining the paths maximizing the bitrate for all selected streaming sessions. Our emulation results show that MaxStream accepts more streaming sessions than WSP-based algorithms (the common technique for determining a path for a communication session) do. The Maximum IMCF module in MaxStream can significantly increase the bitrate for the accepted sessions. As future work, we plan to extend the framework to cope with arbitrary starts and stops of streaming sessions, we take advantage of SDN to dynamically reroute streaming sessions in order to keep the number of streaming sessions in the network close to maximal.

REFERENCES

- [1] Cisco Systems, "Cisco Visual Networking Index: Forecast and Methodology, 2016-2021," June 2017. [Online]. Available: <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf>
- [2] 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," in *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, Hollywood, CA, Dec 2012, pp. 1-8.
- [3] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmoly, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, Jan 2015.
- [4] M. S. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, and Y.-Q. Song, "FlowQoS: QoS for the Rest of Us," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, Chicago, Illinois, Aug 2014, pp. 207-208.
- [5] M. Afaq, S. U. Rehman, and W.-C. Song, "A Framework for Classification and Visualization of Elephant Flows in SDN-based Networks," in *Proceedings of the International Conference on Communications, Management, and Information Technology*, Prague, Czech Republic, Apr 2015, pp. 672-681.
- [6] H. Krishna, N. L. M. van Adrichem, and F. A. Kuipers, "Providing Bandwidth Guarantees with OpenFlow," in *Proceedings of the Symposium on Communications and Vehicular Technologies (SCVT)*, Mons, Belgium, Nov 2016, pp. 1-6.
- [7] S. Civanlar, M. Parlakisik, A. M. Tekalp, B. Gorkemli, B. Kaytaz, and E. Onem, "A QoS-enabled OpenFlow Environment for Scalable Video Streaming," in *Proceedings of the 2010 IEEE Globecom Workshops*, Miami, Florida, Dec 2010, pp. 351-356.
- [8] J. Yan, H. Zhang, Q. Shuai, B. Liu, and X. Guo, "HiQoS: An SDN-based Multipath QoS Solution," *China Communications*, vol. 12, no. 5, pp. 123-133, May 2015.
- [9] A. Ishimori, F. Farias, E. Cerqueira, and A. Abelm, "Control of Multiple Packet Schedulers for Improving QoS on Openflow/SDN Networking," in *Proceedings of the Second European Workshop on Software Defined Networks*, Berlin, Germany, Oct 2013, pp. 81-86.
- [10] Q. Ma and P. Steenkiste, "On Path Selection for Traffic with Bandwidth Guarantees," in *Proceedings of the International Conference on Network Protocols*, Atlanta, Georgia, Oct 1997, pp. 191-202.
- [11] H. Krishna, "Providing End-to-End Bandwidth Guarantees with OpenFlow," Master's thesis, Delft University, July 2016.
- [12] L. Brunetta, M. Conforti, and M. Fischetti, "A Polyhedral Approach to an Integer Multicommodity Flow Problem," *Discrete Applied Mathematics*, vol. 101, no. 1, pp. 13-36, Apr 2000.
- [13] Open Networking Foundation, *Mininet Overview*, 2017. [Online]. Available: <http://mininet.org/overview/>
- [14] Open vSwitch Team, *Open vSwitch*, 2017. [Online]. Available: <http://openvswitch.org>
- [15] NTT, *RYU SDN Framework*, 2017. [Online]. Available: <https://osrg.github.io/ryu/>