

VoDCAST: Efficient SDN-based Multicast for Video on Demand

Christian Koch^{*†}, Stefan Hacker[†] and David Hausheer[†]

[†] Peer-to-Peer Systems Engineering Lab, Technische Universität Darmstadt, Germany

Email: {ckoch|stefan.hacker|hausheer}@ps.tu-darmstadt.de

^{*} since May 2017: Multimedia Communications Lab, Email: Christian.Koch@kom.tu-darmstadt.de

Abstract—Video constitutes the majority of traffic in ISP networks. The demand for more and higher quality videos is growing rapidly while network capacities are limited. Today, OTT video is mostly delivered over CDNs. For ISP-internal video such as IPTV channels, IP multicast provides an efficient delivery method. However, this does not scale for OTT videos due to their large number and their Zipf-distributed popularity. SDN-based multicast enables ISPs to deliver content based on efficient network-level multicast. However, so far this concept has not been applied for OTT Video-on-Demand (VoD) scenarios. To this end, this paper’s contribution is threefold. First, a novel network-based multicast approach for OTT VoD delivery is presented, named VoDCAST. Second, trace-based network simulations are designed and conducted. Third, the achievable bandwidth reduction and practicability of VoDCAST is shown based on a two-week YouTube trace of a nation-wide mobile provider. VoDCAST is able to decrease traffic volume and variance, while keeping network provider costs in terms of state overhead and state change low.

I. INTRODUCTION

Video streaming constitutes the majority of network traffic in today’s Internet. YouTube alone accounts for more than 20% downstream traffic in North America’s mobile networks [1], constituting the largest single source of downstream traffic. Content Delivery Networks (CDNs) like Akamai or Google Global Cache help delivering content globally. By providing caches close to the users, the content provider is relieved from traffic load and a low initial video buffering time can be achieved. CDNs are used, among others, for streaming of both on-demand as well as live video. However, CDNs are often not allowed to operate inside Internet Service Providers (ISP) because they fear CDNs as unpredictable and hard to manage traffic sources [2]. Therefore, CDNs are often not present within ISP networks and cannot help to optimize the network usage there. For their own internal IPTV services, ISPs often use multicast for efficient content delivery [3]. The resources for managing multicast groups render traditional IP multicast for OTT services unlikely to scale, because each router needs to provide multicast capabilities [4]. Additionally, much more on-demand content exists compared to IPTV channels, e.g., about 1.2 billion¹ videos on YouTube. As the number of multicast groups is limited on a hardware switch, not all of

these videos can be delivered by network-based multicast. As VoD popularity is Zipf-distributed, only a small fraction of the content catalog is popular and, therefore, efficient to be delivered by multicast. Selecting videos for multicast is challenging as popularity changes dynamically based on, e.g., video age and daytime [5]. In VoDCAST, this selection is performed by choosing the most popular videos of the recent past, thereby adapting to popularity changes and new videos. Popular YouTube videos are watched thousands of times per hour within broadband access ISP networks. Therefore, it is likely that people stream the same video with only small differences in their playback positions. In this case, there is a potential for multicast to efficiently reduce the ISP’s and CDN’s link utilization, thereby decreasing operational expenditures. So far, none of the existing works explores the potential of VoD streams with a small playback distance and Zipf-distributed popularity. Current related approaches, assume network resources to be unlimited or ignore the user’s need for a small buffering time. Therefore, in VoDCAST, segments which cannot be served in a timely manner via multicast, are served via unicast. Thereby, playback stalling can be avoided at the cost of additional network traffic. VoDCAST assumes the network switches to be SDN(Software-defined Networking)-enabled, as they are likely to be state-of-the-art in future networks. This work’s contribution is three-fold. First, a novel approach for network-based multicast delivery for OTT VoD scenarios, called VoDCAST, is proposed. To the best of our knowledge, this paper is the first that jointly considers the costs of multicast group state and group changes, user waiting time, state-of-the-art parameter settings, and offers support for encrypted video streaming. Second, trace-based simulations are designed and conducted. Third, the system’s bandwidth reduction and practicability is shown based on a YouTube trace of a mobile nation-wide network provider over two weeks. The remainder of this paper is structured as follows: Sec. II discusses related work. In Sec. III, the system design of VoDCAST is presented. Sec. IV presents the dataset used for evaluation and the network simulation design. Finally, the paper is concluded and future work is described in Sec. V.

¹<http://tinyurl.com/j63ffxd> [Access: May 11, 2017]

II. RELATED WORK

In the following, the two key areas related to SDN-based VoD multicast systems are discussed, that similar to this paper aim at improving the efficiency of content delivery. First, multicast approaches which are using SDN are described. Second, scheduling approaches for video multicast are discussed.

A. SDN-based Multicast Approaches

SDN is a paradigm where a network's forwarding and control plane are separated, thereby allowing innovation in the area of programmatical network management. Next, two recent approaches SDM and SDM²Cast proposing SDN-enabled multicast approaches are discussed.

Software-defined Multicast (SDM), proposed by Rückert et al. [6] designs a one-to-many service which is used between CDNs and ISPs in order to enable multicast of OTT live video streams in ISP networks. Therefore, the ISP offers a control API to the CDN. This API allows the CDN to inform the ISP about video streams to be delivery by multicast. In a next step, this stream is forwarded within the ISP network using SDN-enabled switches. The SDM controller is responsible for the multicast group management and determines the corresponding network-layer multicast trees which need to be established inside the ISP network. Therefore, the shortest-path tree is computed, consisting of the paths on which the video stream is to be forwarded. Details on SDM as originally proposed by Rückert et al. can be found in [6]. After computing the multicast tree, the SDM controller translates the tree topology to the corresponding SDN flow entries and forwards them to the SDN controller. These entries are responsible for forwarding the multicast groups' traffic in the data plane. This also involves video stream duplication at the latest possible point in the network to each of the group clients. If a new client is added to an existing multicast group, the SDM controller updates the corresponding multicast tree and triggers the necessary flow entry changes by notifying the SDN controller. The focus of this work is on live streaming. Hence, VoD and content popularity changes are not addressed.

Yang et al. [7] propose an SDN-based Multimedia Multicast Streaming Framework named SDM²Cast. They highlight scalability and flexibility of their approach which uses scalable video coding (SVC). To quickly prone clients, which are no longer available or interested in the content, from a multicast tree and thereby save network resources. Their evaluation is based on a prototypical implementation using three different video quality layers, 16 OpenFlow switches, and 60 client devices. For the evaluation, each of the clients requests a fixed video quality consisting of one to three of the available video quality layers. The authors showed that their approach reduces the used bandwidth by 50% compared with IP multicast with a single-layer video. However, SDM²Cast is not evaluated using a realistic ISP load and scenario. Furthermore, the work does not consider on-demand videos.

B. Scheduling-based Approaches

Three major approaches have been proposed for scheduling of content for multicast, namely batching, patching, and deadline-oriented approaches. In Batching, video streams delivering the same content are aggregated into one multicast group. Dan et al. [8] propose a fixed batching time-window in which requests for the same video are aggregated, resulting in a separate multicast group for each video at the end of each time-window. Consequently, most client requests cannot be handled immediately and are queued. However, a timely playback start is important for the user experience. Hua et al. [9] present a patching approach taking care of the limited client buffer space. The approach does not create new groups after a certain network-wide group limit is reached, respecting limited network resources. A client immediately joins in an existing group to receive the next B minutes of the video filling its buffer. Additionally, each client joins a temporary multicast group to receive segments between the video start and the previously joined existing group, thereby allowing for a faster playback start compared with batching. However, the first received segments are unlikely to belong to the video start and occupy the client buffer until they are consumed by the video player, requiring another multicast group to patch this gap, which is unlikely to scale and has in most cases no benefit over unicast.

Aggarwal et al. [10] propose two deadline-oriented schedulers. For the first one, no additional groups are created if the maximum number of groups, which has to be defined, is reached. If a group becomes free, a segment which is needed for playback first by most of the clients is scheduled. Each client who requested this segment joins the group and receives the segment. In case no group is free and a segment reaches its deadline, it is never scheduled, causing playback stalling. For the second approach, the number of groups is unlimited, thereby outperforming the first approach measured by required bandwidth and initial waiting time as the first segment's deadline is set to the current time. However, it does not consider the number of multicast groups as a costly and limited resource. Additionally, most videos, e.g., on YouTube, are not watched fully [11]. Therefore, issuing request for all video segments at once seems unreasonable.

In summary it can be said that the related work discussed does either not consider the user's need for a fast video start or does neglect that the number of multicast groups is a limited resource. For a realistic evaluation, the number of groups and group changes has to be discussed. Furthermore, the existing works do not consider OTT VoD as well as recent workloads for short videos like the YouTube trace used in this paper's evaluation and, thereby it's property of partial video views.

III. VODCAST DESIGN

Three goals have been set for the design of VoDCast. First, the network utilization should be reduced by efficient network-based multicast of parts of close-in-time VoD sessions, which are defined by the distance between the video streams' playback positions. Second, the state, i.e. the number

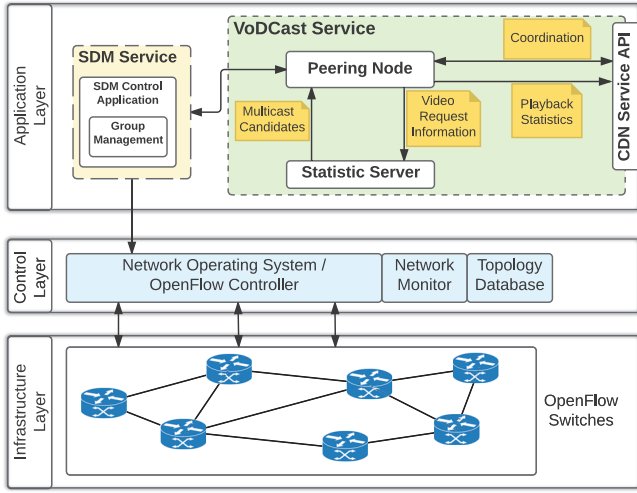


Fig. 1. VoDCAST Network Architecture based on [12]

of flow entries and management overhead, i.e. their changes have to be within the capabilities of current hardware devices. Without respecting these limitations, an SDN-based system is not considered practicable. Third, the user experience should not be decreased, e.g., by additional initial buffering time.

For a popular video, it is likely to find hundreds of simultaneous video sessions. Therefore, some of the customers streams only have a small difference in their playback positions. In this case, there is a potential for multicast to efficiently reduce link utilization. So far, none of the existing works explores the potential of VoD streams with a small playback distance and Zipf-distributed popularity where only a small number of videos are highly popular, i.e., the short tail. Consequently, one challenge for VoDCAST is to determine candidate videos for multicast out of the large number of available videos. This candidate set can be determined in cooperation with the serving CDN knowing which videos are currently popular. For this cooperation, VoDCAST exposes a CDN Service API to get informed about videos to multicast. However, VoDCAST is not depending on the CDN's cooperation, as it is able to collect popularity information of the videos requested by the ISP's customers, in the case videos are delivered unencrypted. An overview of VoDCAST's system modules and their relationships is given in Fig. 1 following the Open Network Foundation (ONF) SDN reference architecture [12].

A. CDN Service API

For CDNs, it is transparent if multiple clients from one ISP network stream the same video. In case such a video is requested frequently and many close-in-time video sessions exist, the CDN can signal to VoDCAST's CDN Service API to prefer this video being delivered via multicast within the ISP network. After both parties have agreed on multicast candidate videos, they are streamed via HTTPS from the CDN to the ISP. Here, video segments are received and forwarded by the ISP over UDP in the case of multicast or over TCP/HTTP in the traditional case of unicast. However,

in order to apply multicast, the ISP cannot support an end-to-end encryption between CDN and clients. A solution can be the CDN supporting a multicast-compatible encryption scheme [13]. Additionally, VoDCAST informs the CDN about join and leave events based on an ID, shared between the two parties, for each multicast group. Furthermore, statistics of the clients, only visible to the ISP, are forwarded to the CDN, in conjunction with this ID, providing information about client playbacks, e.g., playback interruptions, pausing, and quitting.

B. ISP CDN Cooperation

The CDN Service API is offered by the Peering Node which is VoDCAST's central module. In Fig. 1, one Peering Node is depicted. However, in practice multiple Peering Nodes can be placed at the edge of the network, as in reality an incremental deployment of VoDCAST-supporting nodes is more likely. The case of a single Peering Node is considered in the following as smallest deployment scenario. The Peering Node coordinates between the ISP and the CDN or content provider which videos should be streamed via multicast within the ISP network. Thereby, the CDN and the ISP can agree on multicasting not just whole videos but selected parts of them. This can be used for example to exclude the beginning of a video from multicast since it is likely that a user quits the video playback or jumps to another video. In the YouTube trace used (see Sec. IV-A), this behavior is observed mostly within the first 20s of a video stream. Therefore, it is decided to transfer the first $T_{\text{unicast}} = 20s$ always as unicast in this work which excludes 39% of all video sessions in the trace. Hence, overhead introduced by frequent multicast group churn, i.e., group creation/modification is avoided. Also, this helps to keep the initial client buffering time small, as the additional delay caused by a group creation and join is avoided. If the client is added to a group later, the playback is not interrupted, because some segments have been buffered already. For a video selected for multicast, the Peering Node triggers the creation of a multicast group by notifying the SDM controller. If the group is empty, the group is removed. Thereby, situations in which all members leave before the video end, e.g., because of an uninteresting video end, the group can be removed early, hence sparing resources. The established multicast tree of a group is redefined, e.g., by a client joining or leaving. In this case, a new tree is computed and compared with the established one to determine the switches, i.e., nodes in the graph, which need to be updated by installing, altering, or deleting flow entries.

C. Video Streaming Client

The term client refers to the video player on the user device. This player receives, buffers, and playbacks video segments. To receive video segments by multicast, it is assumed that the client is able to receive segments over both HTTP/TCP and UDP. Also, the client has to signal the Peering Node if the player is closed in order to be removed from multicast groups. The trace used to simulate the client behavior contains a session duration for each video request. This allows for

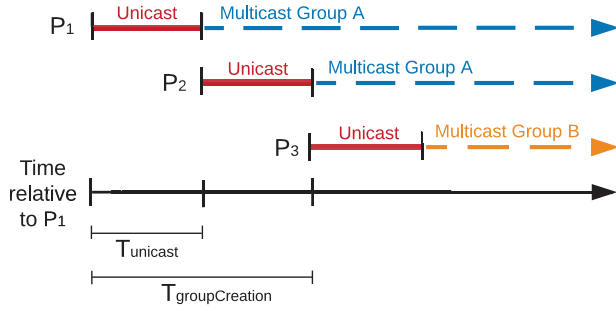


Fig. 2. Example of three video sessions P_1 , P_2 , and P_3

realistically modeling of the number of segments requested by the users which are often only the first couple of segments [11]. A client can join maximal $n = \frac{\text{available bandwidth}}{\text{video bitrate}}$ multicast groups simultaneously. For example, if a client joins in three groups A, B, C, which deliver streams for the same video beginning with the segment 10 (A), 15 (B), 22 (C), then the client leaves A after receiving segment 14. Group B is left after segment 21 is received. Finally, the client remains a member of C till it stops its playback and the buffer is filled or it has received all segments. In case of a video with multiple groups being available and the client can receive more segments than it can buffer, the multicast group which delivers segments farthest from the current playback position is left. If a segment is not delivered by any group, it is requested via unicast immediately before it is needed to avoid playback interruption. The first seconds of a video are always delivered via unicast to minimize initial playback delay. Hence, the number of SDM group changes is kept small as short streaming sessions are excluded from multicast delivery. In this work, the clients receive segments of 5 seconds length and adapt the video quality similar to the YouTube Android app, which has been observed to switch to 1080p after playing about 20s in 480p which is the assumed player behavior.

In VoDcast, the video buffer is filled quickly by downloading one segment after another till the buffer is at least filled with T_{unicast} segments. This is an important behavior as some of the related works ignore the impracticality of letting users wait for playback start more than a few seconds, which decreases the user experience. Afterwards segments can be received from multicast groups if available.

D. System Parameters

Table I gives an overview of VoDcast's parameters.

Min. group creation interval: In case a multicast group exists for a requested video, the minimum group creation interval $T_{\text{groupCreation}}$ defines a trade-off between letting a client join in the existing group or in a newly created one. If the group's playback position is close to the video start, a new client would join the existing group. This closeness is defined by $T_{\text{groupCreation}}$. A high value can cause the client buffer being filled completely by segments far away from the user's current playback position, thereby occupying space for

TABLE I
OVERVIEW OF VoDCAST'S SYSTEM PARAMETERS

Symbol	Definition	Used default
S_{seg}	Video segment size	5s
T_{unicast}	Video start unicast interval	20s
$T_{\text{groupCreation}}$	Min. group creation interval	30s
T_{join}	Max. join playback interval	120s
T_{res}	Min. residence interval	60s
C_{buff}	Client buffer space	50MB
C_t	Client throughput	30Mbps
P_t	Popularity time window	20m
P_k	Top k% of popular videos	20%

segments closer to its playback position. A reasonable range for $T_{\text{groupCreation}}$ is considered to be 10 – 60s.

Max. join playback interval: The max. join playback distance T_{join} defines the maximum allowed difference between the client's and the group's playback position measured in seconds where a client is still added to an already existing group. This parameter prevents the buffer from being filled with segments too far away from the client's playback position. This value is set to 2 minutes considering this to be roughly the median of session lengths in the used trace.

Min. residence time: The min. residence time T_{res} defines the time a client is able to receive segments from a group if it joins. In case the group playback ends after $x \leq T_{\text{res}}$ segments, because the video end is reached, a client will not join.

Popularity time window: The popularity time window P_t defines the size of a sliding window for which popularity statistics are stored. It allows for continuous determination of videos which should be delivered by multicast. This approach is chosen as popularities of videos change over daytime and the video's live time [14]. In this work, 20 minutes is considered as a reasonable setting.

Popularity limit: The popularity limit P_k defines the top $k\%$ of popular videos in time window P_t considered for multicast.

IV. EVALUATION

VoDcast's goal is to reduce ISP network load for OTT VoD traffic. The following sections describe the used trace and network topology, the network simulation design as well as measurements of cost and performance metrics.

A. Trace

For the evaluation, a YouTube request trace consisting of 10 million HTTP requests to 1.6 million different videos is used. The trace was collected and anonymized within a nationwide mobile network operator in two weeks of April 2014. As shown in [11], most videos are only watched for a few seconds. About 30% of all video sessions last less than 10s.

B. Network Model

For the evaluation, a representative ISP topology inspired by Deutsche Telekom is used [15]. As depicted in Fig. 3, its

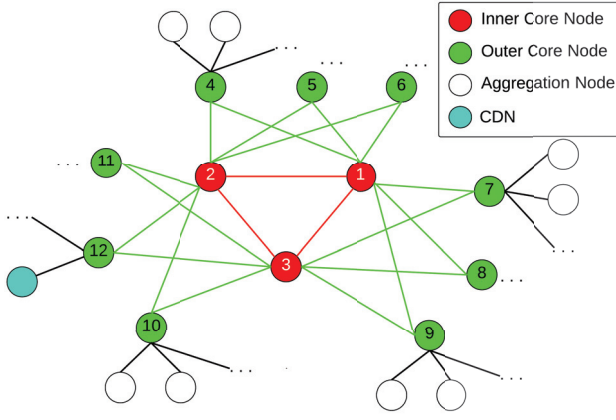


Fig. 3. Used ISP core topology, with some aggregation nodes

three inner core nodes are fully meshed. They are connected to nine outer core nodes, each connected to two different inner core nodes. There are 900 aggregation nodes [15], which are logically connected to one of the outer core nodes. In the 5G scenario considered, these 912 nodes are equipped with SDN-enabled switches. VoDcast is assumed to be installed on an outer core node which is connected to a CDN. Following these results, in the trace-driven simulations conducted, the clients are directly connected to the aggregation nodes.

C. Flow Entry State and Changes

The number of multicast flow entries and their changes state practical limitations of each SDN-enabled system [16], as the Ternary Content-Addressable Memory (TCAM) required for accessing these entries is a costly and limited resource on an SDN-enabled hardware switch. Therefore, these metrics are measured per switch to show that they are bounded at a low level and, hence, that VoDcast is practicable. Fol-

lowing a simple approach, one SDN flow entry per group is implemented in a switch on the group's multicast tree. Each client joining/leaving a group causes flow entry changes. During the simulation, several multicast events occur. On a regular time interval, a snapshot SP_t is taken, containing these events. For each SP , the number of groups, flow entries, and flow entry changes per switch as well as the bandwidth per link are measured. Therefore, the network is modeled as a graph consisting of nodes, e.g., $u, v \in V$ connected by edges $(u, v) \in E$. Two kinds of nodes exist: switches $S \subset V$ and clients $C \subset V$. The number of flow entries F installed during a snapshot can be calculated as follows. Assumed are a multicast group $g \in G_{SP_t}$ of the snapshot and the switches $S_{SP_t,g}$ holding a flow entry for the multicast tree of this group. These tree nodes consist of clients $c \in C_{SP_t,g}$ and switches $s \in S_{SP_t,g}$. The number of installed flow entries $F(SP_t)$ are defined by Eq. 1. Flow entry changes FC are determined by comparing the installed flow entries at SP_{t-1} and the current snapshot SP_t , defined by Eq. 2.

$$F(SP_t) = \sum_{\forall g \in G_{SP_t}} |S_{SP_t,g}| \quad (1)$$

$$FC_{SP} = |(f \in F_{SP_t}, f \notin F_{SP_{t-1}}) \cup (f \in F_{SP_{t-1}}, f \notin F_{SP_t})| \quad (2)$$

Fig. 4 and Fig. 5 depict the number of rules and their changes per minute for three different classes of switches: switch 12 which serves as an intermediate node between the CDN and all clients, switches 2 and 3 which are directly connected to switch 12, and all other switches 4 - 11. Switch 1 is not considered in the evaluation as none of the shortest paths from the CDN to the clients uses switch 1, as only one Peering Node is considered. Fig. 4 shows the number of SDN flow entries, i.e., OpenFlow rules, forwarding network traffic. For all core switches in the network topology used, the number of flow entries are depicted. By using VoDcast, the median of the maximum number of flows installed on the switch per minute is below 30 for all switches and different configurations of P_k , except for switch 2, 3, and 12, where no higher number of installed flows then 100 is observed. Even with the highest configuration $P_k = 50$, the 75-percentile is just 84 for switch 12 and, thereby bound at a low level. The maximum observed number of flow entries at switch 12 is 160. This is about two magnitudes lower than the number of flow entries supported by current hardware switches, offering space for up to 60k L2/L3 flow entries [16]. Fig. 5 depicts the maximum numbers of flow entry changes per minute, i.e. OpenFlow rule modifications and drops send to a switch. It is shown, that the flow entry changes caused by VoDcast are well within the capabilities of nowadays hardware switches which supports up to 32k flow entry changes [16] per minute. The mean of all core switches, except 2, 3, and 12 is about 10 changes per minute. For the highest configuration $P_k = 50$, no value larger than 912 is observed. A deployment with more than one Peering Node would further reduce the load on switch 12, 2, and 3.

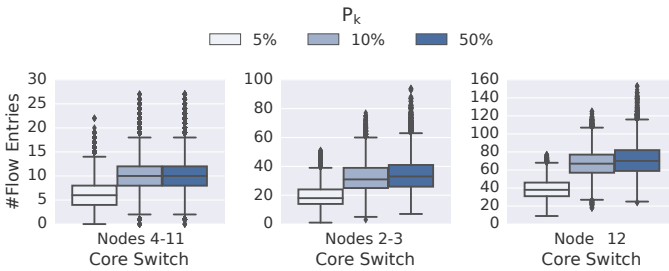


Fig. 4. Maximum number of flow entries per minute

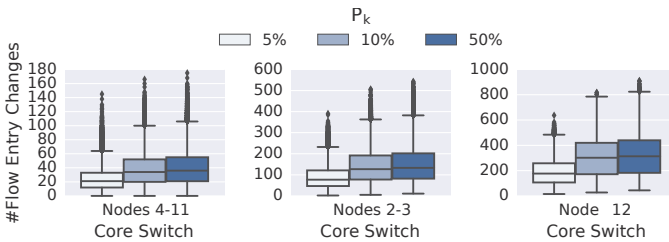


Fig. 5. Maximum number of flow entry changes per minute

TABLE II
VoDCAST'S MULTICAST(V-MC) AND UNICAST(V-UC) TRAFFIC

	P_k	0.01%	1%	5%	10%	50%
V-MC	mean	0	1.9	4.4	8.1	8.7
	median	0	2	4	8	9
	std	0	0.7	1.3	1.6	1.9
V-UC	mean	68.7	67.6	64.5	59.9	59.1
	median	76.5	76	71	65	65
	std	22.5	21.4	19.3	16.6	15.9
Bandw. Red. ($\Delta\mu$)		1.9	2.2	2.8	3.7	3.9
Std. UC Red. ($\Delta\sigma$)		0	0.6	2.7	5.4	6.1

D. Bandwidth Utilization

In order to calculate VoDCAST's bandwidth utilization, first the bandwidth consumed by one segment is computed for segments $s \in S_{(u,v)}$ sent over an edge $(u,v) \in E$. The consumed bandwidth depends on their duration S_{seg} and the bitrate $r_{s,q}$ of the used quality level q . Eq. 3 defines how the bandwidth of all segments on one link (u,v) is calculated.

$$B(S_{(u,v)}) = \sum_{s \in S} (S_{\text{seg}} * r_{s,q}) \quad (3)$$

To calculate the bandwidth of group G_j the bandwidth over all edges $(u,v) \in E_{G_j}$ used by it is summed up. The segments delivered to this group are $s \in S_{G_j}$. Consequently, the bandwidth consumed for group G_j is defined by Eq. 4.

$$B_{G_j} = \sum_{\forall (u,v) \in E_{G_j}} B(u,v) = \sum_{s \in S_{G_j}} (S_{\text{seg}} * r_{s,q} * |E_{G_j}|) \quad (4)$$

Following this methodology, the bandwidth consumed over all groups is calculated for each 30 minute interval between 9am and 11:30pm for 14 days. Compared with unicast, VoDCAST has the potential to reduce costs for ISPs and CDNs as the variance as well as the overall sum of network load can be efficiently reduced requiring only a low number of flow entries and flow entry changes. The share of unicast and multicast caused by VoDCAST compared to the highest unicast load per hour of the day is depicted in Table II, for different settings. For multiple configurations of P_k , the bandwidth reduction $\Delta\mu$ and the reduction of unicast traffic's standard deviation $\Delta\sigma$ are depicted. The reduction in link utilization achieved by VoDCAST is shown by $\Delta\mu$ which is the difference between the link utilization with and without VoDCAST. The reduction in standard deviation for the unicast achieved by VoDCAST is represented by $\Delta\sigma$. It can be seen that the bandwidth reduction and variance decreases with an increasing size of P_k , up to 3.9% bandwidth reduction and 6.1 reduction of the standard deviation. However, the increase flattens from P_k 10% to 50%.

V. CONCLUSION AND FUTURE WORK

In this work, VoDCAST, a novel SDN-based system for multicasting OTT on-demand video streams is presented. VoDCAST is highly configurable and delivers streams with small playback differences partially via multicast. It supports cooperation with CDNs and efficiently reduces traffic in ISP

networks. The results presented rely on a real YouTube trace over two weeks and show the impact of realistically chosen key parameter settings on the achievable bandwidth reduction. By using VoDCAST, the ISP's network load is reduced 3.9% and its standard deviation about 6% compared to unicast. VoDCAST's practicability is demonstrated, as the resource utilization in terms of SDN flow entry state and changes are shown to stay well within the limits of current SDN-enabled hardware switches and at a low level. In future work, a parameter study and a comparison against related approaches are planned.

ACKNOWLEDGMENT

This work has been funded in parts by the German Science Foundation (DFG) as part of the Collaborative Research Center 1053 MAKI. The authors would like to thank Julius Rückert from TU Darmstadt for his valuable feedback.

REFERENCES

- [1] Sandvine, "Global Internet Phenomena Report - H1," 2016.
- [2] G. Hasslinger and F. Hartleb, "Content Delivery and Caching from a Network Provider's Perspective," *Elsevier Computer Networks*, Vol. 55, No. 18, pp. 3991–4006, 2011.
- [3] Y. Liu, Z. Liu, X. Wu, J. Wang, and C. C. Y. Yang, "IPTV System Design: An ISP's Perspective," in *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2011, pp. 234–240.
- [4] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network*, Vol. 14, No. 1, pp. 78–88, 2000.
- [5] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System," in *SIGCOMM Conference on Internet Measurement (IM)*. ACM, 2007, pp. 1–14.
- [6] J. Rückert, J. Blendin, R. Hark, and D. Hausheer, "Flexible, Efficient, and Scalable Software-Defined Over-the-Top Multicast for ISP Environments with DynSDM," *IEEE Transactions on Network and Service Management (TNSM)*, Vol. 99, pp. 1–14, 2016.
- [7] J. Yang, E. Yang, Y. Ran, and S. Chen, "SDM² Cast An OpenFlow-Based, Software-Defined Scalable Multimedia Multicast Streaming Framework," *IEEE Internet Computing*, Vol. 19, No. 4, pp. 36–44, 2015.
- [8] A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic Batching Policies for an On-Demand Video Server," *Springer International Publishing, Multimedia Systems*, pp. 112–121, 1996.
- [9] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A Multicast Technique for True Video-on-Demand Services," in *International Conference on Multimedia*, No. 6. ACM, 1998.
- [10] V. Aggarwal, R. Caldebank, V. Gopalakrishnan, R. Jana, K. K. Ramakrishnan, and F. Yu, "The Effectiveness of Intelligent Scheduling for Multicast Video-on-Demand," in *Proceedings of ACM International Conference on Multimedia (MM)*, 2009.
- [11] C. Koch and D. Hausheer, "Optimizing Mobile Prefetching by Leveraging Usage Patterns and Social Information," in *IEEE 22nd International Conference on Network Protocols (ICNP)*, 2014, pp. 293–295.
- [12] "SDN Architecture," Open Networking Foundation, Tech. Rep. ONF TS-502, 2014. <https://goo.gl/Bb1jQI>
- [13] Y. Duan and J. Canny, *How to Construct Multicast Cryptosystems Provably Secure Against Adaptive Chosen Ciphertext Attack*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 244–261.
- [14] X. Cheng, C. Dale, and J. Liu, "Statistics and Social Network of YouTube Videos," in *Proceedings of IEEE International Workshop on Quality of Service (IWQoS)*, 2008.
- [15] A. Betker, I. Gamrath, D. Kosiankowski, C. Lange, H. Lehmann, F. Pfeuffer, F. Simon, and A. Werner, "Comprehensive Topology and Traffic Model of a Nationwide Telecommunication Network," *Journal of Optical Communications and Networking*, Vol. 6, No. 11, pp. 1038–1047, 2014.
- [16] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, Vol. 103, pp. 14–76, 2015.