DynSdm: Dynamic and Flexible Software-Defined Multicast for ISP Environments

Julius Rückert, Jeremias Blendin, Rhaban Hark, and David Hausheer
Peer-to-Peer Systems Engineering Lab, Technische Universität Darmstadt, Germany
Email: {rueckert|jblendin|rhark|hausheer}@ps.tu-darmstadt.de

***Abstract*—A number of today's over-the-top (OTT) services could greatly benefit from a scalable and efficient network-layer multicast support on the Internet. IP multicast showed to not meet these requirements and, thus, is not available for this purpose. Content Delivery Networks emerged as global alternative but usually end at the border of ISP networks. Software-Defined Multicast (Sdm) is proposed in a previous work by the authors, enabling ISP-internal network-layer multicast delivery of OTT traffic. While it coins fundamental concepts, it does not detail the ISP-internal traffic and service management and leaves important questions unanswered. To this end, DynSdm is proposed in this paper, detailing the multicast planning and management, proposing a novel network-layer multi-tree mechanism to distribute traffic on links inside the ISP network, and introducing mechanisms to handle group and network dynamics. DynSdm was prototypically evaluated, showing its high traffic efficiency, good scalability, and superior traffic distribution characteristics.**

I. Introduction

Today, an increasing amount of Internet traffic is caused by over-the-top (OTT) services. For years, Internet Service Providers (ISPs) find themselves being used as "dumb pipes", both unable to participate in the revenue of OTT services and having little incentive to upgrade their networks to help improving OTT service qualities for their own customers. Yet, from a pure technical point of view, network providers could be a driving force for innovation on the Internet by providing new network services to the OTT world and their own customers. A major reason for this situation can be seen in missing practical service models and abstractions to allow realizing new services in a quick and flexible manner. A large part of today's OTT traffic consists of video streams [5], [11], [31] that could greatly benefit from a number of network services, most prominently from network-layer multicast to more efficiently deliver, e.g., the increasing amount of live content on the Internet [7], [27]. Multicast support on an Internet-scale showed to be hard to achieve in practice [9], where IP multicast [6] failed to become a solution that spans more than individual network islands. By now, it seems even reasonable to assume that solutions following the initial IP multicast design are unlikely to become reality, given the structure of the Internet, the involved stakeholders, and the technical challenges of the approach itself [9]. Today, global OTT multicast delivery is enabled by Content Delivery Networks (CDNs) that emerged as alternative, implementing multicast functionality at application layer with hundred thousands of world-wide deployed server nodes [32]. While this approach scales with the nodes' resources, CDNs usually end at ISP network edges, which fear CDN nodes as being unpredictable traffic sources inside their well-managed networks [14]. This situation leaves the ISPs with heavy traffic load on their networks as multicast traffic is delivered through the ISP as unicast streams from the CDN nodes to the clients. This is unfortunate as ISPs very well know how efficient multicasting in their network can be realized as they typically use it for their ISP-internal IPTV services [22].

In a previous work by the authors [30], Software-Defined Multicast (Sdm) is proposed to address this problem in a way that both ISPs and OTT content providers can benefit from. Using the concept of Software-Defined Networking (SDN), it enables a practical service model for one-to-many multicasting services. The service model itself was already discussed by Holbrook et al. [15] in 1999 but only recently became practical based on SDN and its implementations, such as the OpenFlow protocol. Using the SDN concept, Sdm introduces a well-defined control API run by the ISP to allow OTT providers creating and managing network-layer multicast groups for OTT data delivery inside the ISP network. For the data path, Sdm uses OpenFlow features to enable an efficient, scalable, and transparent delivery of the streams. While Sdm coins fundamental concepts for SDN-based OTT multicasting in ISP networks, it leaves some important questions unanswered that require a deeper study to make a practical applicability of the approach feasible. By making OTT streams explicit to the ISP, an efficient multicast delivery becomes possible. Yet, to calculate the actual data paths, a simplified approach was taken, not specifying the planning of multicast trees in line with the ISP's internal traffic engineering policies. Thus, the multicast data paths are rather inflexible and lead to an unbalanced distribution of load in the ISP network. In addition, Sdm lacks mechanisms to support dynamic multicast groups with changing client populations and mechanisms to react on network dynamics, such as link failures or load changes.

To this end, DynSdm (Dynamic Software-Defined Multicast) introduces a number of extensions to the original Sdm approach to answer the above questions. Thus, the contributions of this paper are threefold: (1) a detailed multicast tree planning and management approach is introduced to better support ISP preferences in planning the multicast trees; (2) a novel network-layer multi-tree approach is proposed to distribute traffic across network links; (3) mechanisms for supporting dynamic groups and fast reactions on link failures are presented. Due to space constraints, a fourth extension, an novel SDN-based service discovery approach to allow a discovery of DynSdm services along OTT routing paths, is presented in an extended version of this paper [28]. The proposed mechanisms are shown to be efficient and scalable, making the general Sdm idea more practical and, thus, strengthening its applicability.

978-3-901882-77-7

The remainder of the paper is structured as follows: Section II provides an overview on the original SDM approach. Section III presents the design of DYNSDM. Subsequently, Section IV presents the evaluation. Related works are discussed in Section V and, finally, Section VI concludes the paper.

II. BACKGROUND: SOFTWARE-DEFINED MULTICAST

In a previous work by the authors, Software-Defined Multicast (SDM) [30] is proposed for efficient delivery of OTT multicast streams in ISP environments. The core concepts of SDM and the recent extension Adaptive Software-Defined Multicast (ASDM) [3] are presented in the following as DYNSDM conceptually builds on them and proposes essential extensions.

The main objective of SDM is enabling ISPs to provide efficient and well-controlled network-layer multicast services to OTT content providers, CDN networks, or even peer-to-peer networks [29]. In addition, it aims at being transparent to clients of the service, i.e. the ISP's broadband access customers served. Clients receive normal IP unicast packets that are not distinguishable from packets directly send by the content provider. For this purpose, SDM intentionally avoids the usage of legacy IP multicast-related protocols for group management and routing. The content provider is assumed to be aware of its clients in an OTT scenario and thus envisioned to manage the multicast group directly using an API provided by the ISP. SDM as well as DYNSDM assume a future ISP scenario where all or a subset of the routers are SDN-enabled. Due to its increasing adoption by hardware vendors, an OpenFlow-based realization is proposed for the core mechanisms of SDM. The SDM service API entity directly communicates with the ISP's network controller, which manages the configuration of a set of network functions that constitute the core multicasting functionality. After registering a new multicast group, the ISP provides the group source with a so called *group socket*, consisting of an IP address and port allocated by the ISP. The group source sends the multicast stream in form of IP unicast UDP packets addressed to the group socket, which is delivered to the ISP following normal IP routing through the Internet. At the ingress switch of the ISP, the packets are matched using an OpenFlow rule that was installed by the controller at multicast group registration. To allow for an efficient forwarding and processing within the ISP network, either the group socket information is used to uniquely identify the packets of individual multicast groups or the ingress switch marks any desired header field of the packets with an appropriate group identifier. SDM proposes to use normal IP unicast addresses and ports for this purpose as this allows compatibility to normal IP routing within the SDM domain. As the group socket uniquely identifies the multicast data stream, it is used to install corresponding forwarding and duplication rules at switches involved in the group's multicast tree that was also calculated and configured by the network controller upon group registration. At the edge of the network, the individual data streams arrive at the individual egress switches, which are the last OpenFlow-enabled switches before the packets are delivered to the individual clients. At this point, packets might be duplicated again but more importantly, the packet header is rewritten using another rewrite action, replacing the group socket information by the individual client's IP address and port as specified by the group source for each individual client. This way, the multicast stream is translated to individual unicast streams to the clients. As a result, SDM is transparent to the clients that are not aware of the multicast delivery.

In [3], a second work by the authors, ASDM is proposed as extension to the original SDM concept. It extends the packet duplication mechanism, providing ISPs an improved control on the tradeoff between bandwidth and network state per multicast group. This shows to be highly beneficial to support a large variety of OTT multicast streams, typically following a Zipf-like popularity distribution [1]. ASDM allows ISPs to dynamically define where in the network the multicast-to-unicast translation is performed. This is a generalization of SDM's static translation at the egress switches. This way, different strategies for the duplication and translation can be realized. The strategy resulting in the lowest bandwidth consumption is called *late-duplication* and is very similar to the default SDM strategy. Here, the translation happens together with the last duplication step, thus likely close to the clients. This strategy pays for its bandwidth efficiency by a large number of flow rules, in worst case spread across all involved switches/routers. For large multicast groups this is likely to be acceptable in the light of the achieved traffic reductions. For heterogeneous and Zipf-like populated multicast groups, however, network state can quickly become a precious good as the rule space of the switches is limited. Thus, it might not be worth using up the space for many small long-tail groups or in parts of the network where other network functions are more important. Therefore, ASDM introduces the ability to use other strategies, e.g. *early-duplication*. Here, as another extreme, multicast streams are translated to unicast as soon as they enter the ISP network at the ingress switch. This way, the content provider still benefits from the multicasting in that it only sends the stream once. The ISP drastically reduces the required network state and concentrates it at a single switch. The rest of the topology remains unaware of the group and performs normal routing. The ISP trades this reduction in state for an increased traffic inside its network. Besides these extremes, ASDM introduces a translation threshold to efficiently control the tradeoff between bandwidth and network state. Here, a translation is automatically performed by the switches if a remaining subtree holds fewer clients than the translation threshold, effectively limiting the network state required per client. While these features greatly improve the ISP's control on the service, DYNSDM goes even further in this direction and proposes mechanisms orthogonal to the ones presented above. In a productive implementation, it is envisioned that a combination of all three approaches is used to allow ISPs to leverage the full feature set of all of them.

III. DYNAMIC SOFTWARE-DEFINED MULTICAST

While SDM [30] proposes the basic functional building blocks for an ISP-provided multicast service for OTT streams, it does not detail how ISPs can efficiently manage and plan the multicast delivery in line with the rest of their traffic. ASDM [3] extends SDM by the possibility to control the tradeoff between bandwidth and network state, which addresses an orthogonal problem (cf. Section II). In the following, Dynamic Software-Defined Multicast (DYNSDM) is presented, which adds the actual traffic engineering support to SDM, making the proposed mechanisms highly relevant for a practical adoption of the overall approach. For this, first, DYNSDM's multicast tree planning and management approach is presented. Second,

an efficient network-layer multi-tree extension is proposed to allow for a fine-granular traffic engineering of the multicast traffic. Third, mechanisms are presented that allow DYNSDM to dynamically react on changing client populations and network conditions, such as congestion or link failures. Due to space constraints, the fourth constituent of DYNSDM, a novel, practical, and transparent service discovery approach, is presented in an extended version of this paper [28].

A. Multicast Tree Planning and Management

The multicast tree planning and management process of DYNSDM and its input parameters are summarized in Figure 1, including the multi-tree planning as detailed in Section III-B.

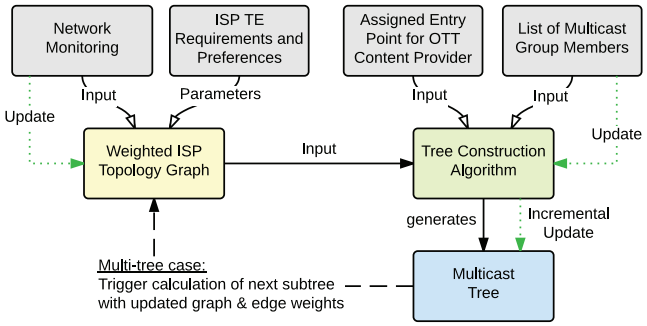


Fig. 1. DYNSDM's multicast tree planning and management process.

Graph model of the ISP topology: As a first step, the network topology of the ISP is modeled as graph. The information required for this is assumed to be available based on static and real-time information provided by the ISP's monitoring system. As this work envisions a future ISP scenario, the physical network topology is assumed to consist of OpenFlow-enabled switches/routers and links between them. For simplicity, a homogeneous OpenFlow scenario is assumed in the following. Hybrid networks scenarios could be supported as well with some basic requirements on the routing behavior of non-OpenFlow routers. They would act as passive forwarders of the multicast streams, not performing any duplication or rewriting. The result of this initial step is a mapping of the network topology to a graph $G = (V, E)$, where each OpenFlow switch is represented as vertex $v \in V$ and each link as edge $e \in E$.

Calculating edge weights: As a next step, the graph is transformed into a weighted graph by assigning edge weights that reflect the ISP's traffic engineering preferences, its internal policies, and the current network conditions. Individual links or parts of the topology can be excluded from the multicast transport by removing the respective vertices and edges in this step, resulting in a sub-graph $G_{\text{MC}} \subseteq G$. The calculation of the weights is performed based on monitoring information on the current traffic conditions, individual Quality-of-Service (QoS) metrics, and the available resources of network links. For this purpose, the ISP defines the importance and ranking of the individual properties and network aspects and, thus, enables a delivery in line with its overall traffic engineering approach. To calculate a single weight per edge, a number of monitored and static information on the link could be used. Following the approach of typical routing protocols, e.g. the Enhanced Interior Gateway Routing Protocol (EIGRP) [34], DYNSDM calculates the weight of an edge e using (1).

$$\begin{aligned}\text{weight}_e :=& (K_1 \times \text{bandwidth}_e) + (K_2 \times \text{utilization}_e) \\ &+ (K_3 \times \text{delay}_e) + (K_4 \times \text{lossrate}_e) \\ &+ (K_5 \times \text{failurerate}_e)\end{aligned} \tag{1}$$

The coefficients K_1 to K_5 can be used to influence and tune the importance of individual QoS parameters for the final weights. The idea is to provide ISPs with a powerful mechanism that they can parameterize according their topology and requirements. The process of tuning the parameters is not covered here as it is not different from tuning similar parameters in routing protocols like the one mentioned above.

Multicast tree construction: For the actual multicast tree construction, two more pieces of information are required: the entry point, i.e. the switch at that the unicast traffic sent by the content provider enters the ISP network and the list of group members to be served. Both are available at the service API server, which is used by the content provider to interface with the ISP for group registration and management. The entry point is defined by the *group socket* assigned by the ISP on group registrations as introduced in Section II. By assigning DYNSDM group sockets in line with the externally announced BGP routes, the ISP can influence where the respective traffic enters the network. Typically, large ISPs consist of multiple Autonomous Systems (ASes) that might be reached over different paths and peering connections. Assigning group sockets to one of the ASes, thus, allows influencing the entry point to the ISP network. If this is not possible, the entry point could also be learned or probed on service discovery. The actual construction of the tree is being done using well-studied graph algorithms. Here, it is important to note that the tree is to be calculated from the entry point to all clients in the group, which usually is a subset of all clients connected to an ISP. In contrast to traditional multicast approaches, no distributed tree calculation protocol is required and, thus, a variety of existing graph algorithms could be used here. For an ISP scenario, the use of multicast algorithms constructing a *Minimum Spanning Tree* (MST) that minimize the sum of the weights of used edges might be a good choice. As the tree is only required to include a subset of vertices, this problem can be classified as a Steiner tree problem, which is well known to be NP-hard [12]. Yet, a set of heuristics and approximation algorithms exists that can be applied to build nearly optimal trees more efficiently [2]. Depending on the requirements, it might be also sufficient to build a *Shortest Path Tree* (SPT) that minimizes the path between the entry point and the receivers individually, making it a good choice to, e.g., reduce the delay of the multicast delivery. In the focus of this work, no particular algorithm is preferred over another one. The choice is intentionally kept open, providing a framework for varying application scenarios and allowing ISPs to pick the best algorithm that fits their needs. For the prototypical implementation, it was decided to exemplarily adopt an algorithm that calculates SPTs, namely a variant of Dijkstra's algorithm [8].

Network-layer path setup: As a last step, the multicast tree is mapped back to the actual network topology and flow rules are installed at the involved switches. Here, three different switch roles can be distinguished: ingress switches, internal switches, and egress switches. Ingress switches function as entry points for the traffic and perform the unicast-to-multicast

translation using header rewriting and marking packets with its specific group identifier similar to SDM. For the multi-tree case, sub-group identifiers are used as introduced in the next section. Internal switches purely forward and duplicate packets of the stream based on this identifier. Egress switches additionally perform the multicast-to-unicast translation by rewriting the packet headers for the individual receivers.

B. Network-Layer Multi-Tree Support

DYNSDM fundamentally extends the mechanisms presented in the last subsection to make the multicast traffic elastic. Elasticity, here, refers to the ability to dynamically distribute multicast traffic over links of the ISP topology, reducing imbalances between links and avoiding links used for other services. This is realized by breaking up the rigid definition of multicast streams, adopting a mechanism from peer-to-peer overlays [37] and application-layer multicasting [16], namely the usage of multi-tree topologies. They were originally introduced to overcome limitations of single-tree overlay topologies that are known to be prone to instability in case of dynamics in the client population. Besides, single trees unevenly distribute load across peers as almost half of them are leaf nodes and, thus, cannot contribute to the distribution. Using multiple independently built trees, each carrying a substream of the original stream, these problems can be addressed [21]. Translated to network-layer multicasting, where switches perform the duplication of traffic, splitting a monolithic multicast stream into smaller substreams and distributing them over individual multicast trees opens a whole new set of possibilities for traffic engineering of multicast streams. The individually transported streams become smaller and can be distributed within the network more evenly, thereby avoiding congestion on strategic links and allowing to elastically avoiding other, more important and more rigid traffic in the network. While ISP network topologies themselves are typically rather static as such, they have to cope with link outages on a regular basis, caused by cable cuts or failing network equipment [13]. Therefore, traffic engineering mechanisms have to be able to dynamically adapt to changes in the network topology. Thus, by distributing multicast traffic over subtrees, a network-layer multi-tree delivery could greatly benefit from its higher resilience, as each switch and used link becomes less important for the overall delivery process of a multicast stream. Together with a fast rerouting and repair mechanisms, a multi-tree approach is expected to greatly improve the quality of the multicast service for the users.

To realize a multi-tree extension to SDM, the tree planning and management mechanism is refined to enable calculating multiple multicast trees for the same entry point and client set (cf. Figure 1). For the multi-tree case, after building the first tree, more trees are generated iteratively, inspired by [19]. By increasing the weight of edges used by previous trees, or even removing them from the graph, a variety of different characteristics, such as edge disjointness can be supported [25], [35]. In reality, edge disjointness shows to be hard to achieve because of the structure of typical ISP networks, where in certain parts of the topology the number of alternative network paths to the same client is limited. Therefore, disjointness is targeted but not enforced if no alternative paths exist. This enables multiple trees to be built even if they share some edges.

Data-plane traffic splitting and merging: A key feature of the original SDM approach is that, after installation, all required features are fully run within the data path of the network, while being transparent to content provider and clients. This is achieved by mapping its features to actual network functionality supported by standard OpenFlow switches. DYNSDM targets the same network-layer efficiency, without increasing the load on the control path of the SDN network. Therefore a key contribution of this approach is the usage of OpenFlow's group table *select* type at the respective ingress switch of the multicast group as shown in Figure 2.

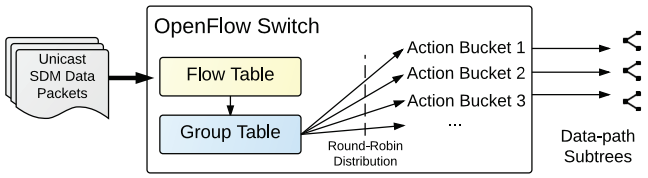


Fig. 2. DYNSDM's traffic splitting based on OpenFlow group *select* feature.

The OpenFlow standard specifies this feature as being optional and being based on a switch-computed selection algorithm (cf. [26]). Due to its high value for all kinds of network-layer load balancing implementations, its support in state-of-the-art OpenFlow switches is assumed to be very likely. Following the specification, the *select* feature, in its simplest version, allows to distribute incoming packets of a multicast group to different action buckets. While SDM rewrites and forward the packets according a single tree, DYNSDM uses this feature to splits the traffic into subsets that are delivered over individual trees (cf. Figure 3). Each tree uses its own sub-group identifier, implemented as individual rewrite and output actions in its respective action bucket. In case a weighted distribution algorithm is implemented by the ingress switch, more elaborate traffic splitting approaches can be realized, e.g. using an ISP's knowledge on estimated path reliability or cost metrics to distribute traffic to subtrees with unequal shares. At the egress switches, similar to SDM, the packets are translated to unicast packets addressed to the individual clients. For an efficient realization in case of multiple subtrees, per-client header rewrite and output actions are managed using a dedicated group table. This way, exactly one flow rule and matcher is required per DYNSDM subtree, all pointing to the same group table for the unicast translation. Consequently, the required TCAM space linearly increases with exactly one new matcher per subtree. It is independent of the number of clients as group tables are not managed in TCAM.

Practical considerations: While the proposed multi-tree mechanism has clear benefits, it also has some limitations that, depending on the scenario, might or might not be of practical relevance. One is the impact of network delay on the multi-tree delivery process. As DYNSDM aims at building distinct and independent subtrees, it is likely that the paths between the entry point and the same client in each tree experience different transmission delays. As a result, the multicast traffic can be subject to permanent packet reordering as the packet delivery happens with different delays on each subtrees. Whether this problem is of practical relevance depends on the actual network structure. As DYNSDM, similar to SDM and traditional IP multicast, targets the delivery of connectionless data streams, applications have to be able to deal with packet reordering. In

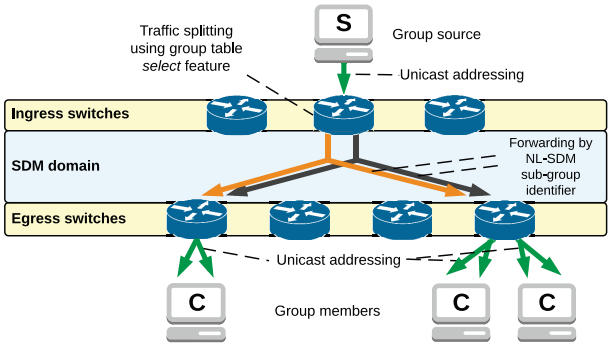


Fig. 3. Multicast delivery in DynSdm using multiple delivery trees.

case the level of reordering is unacceptable, subtrees could be constructed that respect a required application-specific maximum delay difference. Certainly, video streaming applications with large buffers of several seconds of video content are more tolerant in this respect than more data-driven real-time applications. The DynSdm service, therefore, could offer different service classes for typical applications and, thereby, respect different requirements in the subtree calculation process.

C. Handling Dynamics

Client Dynamics: To support dynamically changing client populations, DynSdm realizes a number of mechanisms to quickly connect new clients to the delivery trees as well as disconnecting them. Following the original Sdm approach, the group events are initiated exclusively by the content provider. The content provider decides on which clients are to be included in the multicast service and either establishes a new DynSdm group or adds the client to an existing group using the service API. As a result, new clients need to be attached to the active delivery process of a group. A trivial solution for this would be a re-initiation of the normal tree calculation process. Yet, this would imply a high and unnecessary overhead as the resulting optimal trees might look different with every new client (i.e. in case MSTs are built). Therefore, DynSdm adopts an approach based on a mechanism proposed in [23], attaching new clients by finding paths using a breadth-first search (BFS) from the new client to the active subtrees. This way, new clients can quickly be connected by only adding new flow rules on the BFS paths and modifying a single rule per subtree at the switch connecting the path to the active tree. As this can possibly result in non-optimal tree structures over time, DynSdm checks the established structures on each event and incrementally optimizes them if required. For this, established tree structures are monitored and regularly compared to the optimal structures using a scoring function. If a client is removed using the service API, DynSdm traverses all subtrees backwards starting from the client to the closest switches performing duplication. At these switches, the branches are pruned by deleting their output actions. At this point, the stale branches' remaining flow rules can be scheduled for a later removal as no strict time constraints exist once they are detached from the active delivery process. The content provider is informed about the successful removal and the remaining tree is re-evaluated for a potential optimization.

Reacting on link failures: In case a network link used in one of the DynSdm subtrees fails, two cases are to be distinguished. In the first case, the failing link is directly adjacent to the ingress switch. In this case, the *group table select feature* automatically handles the failure as the outage (i.e. the link is down) is directly detected at the switch and the select algorithm immediately continues distributing the traffic to the remaining subtrees. In the meanwhile, the network controller is informed, triggering the installation of an alternative subtree. In the second case, a link deeper in the ISP topology is affected, thus the controller has to handle the outage. On failure detection, it immediately disables the affected trees by removing the affected action buckets from the group table of the ingress switch, distributing the traffic to the remaining trees. In parallel, new subtrees are calculated, incrementally installed, and, finally, enabled by adding a new action bucket. In the first case, no packet loss occurs due to the automatic rerouting to alternative subtrees. In the second case, packet loss is inevitable, yet only affecting a fraction of the multicast traffic due to the multi-tree distribution.

IV. Evaluation

Measurement Methodology

To investigate the applicability, performance, and costs of DynSdm, the core mechanisms were implemented as prototype and an emulation-based evaluation was conducted. The implementation consists of a set of Ryu[1] OpenFlow controller components, including the DynSdm multicast group management, the multicast tree management, and the discovery protocol. Experiments were conducted using Mininet[2] and Open vSwitch[3]. The Open vSwitch implementation was extended to fully support the group table *select* features as described by the OpenFlow standard. This was necessary as Open vSwitch only supports a simplified implementation of this feature, which is limited to the use of a hash function on the destination MAC address of packets to select random action buckets for packet processing. A packet header-agnostic stochastic distribution mechanism was added to fully support the tree-splitting feature of DynSdm and allow splitting streams in a weighted manner.

The evaluation scenarios used in the following were chosen to allow a detailed studying of DynSdm's core mechanisms. A sensitivity analysis is presented that focuses on the multi-tree parameters of DynSdm and investigates the scalability of the approach under different parameter settings in combination with changing topology sizes and group populations. Based on the default parameter settings defined in this first step, the traffic efficiency is investigated and compared to a single-tree multicast and unicast delivery. To complete the picture, the efficiency of DynSdm for handling dynamics is studied. Due to space constraints, the last part is only presented in an extended version of this paper [28]. As DynSdm is intended for ISP scenarios, a representative ISP topology is used, introduced in the following subsection. Most experiments were repeated 50 times, the remaining ones at least 30 times, and 95% confidence intervals are reported for all mean values.

Evaluation Scenarios

The scenarios all use a representative PoP-level ISP topology as basis, derived from a presentation by Deutsche

[1]Ryu v3.16, https://osrg.github.io/ryu/ [Access: June 9, 2015]
[2]Mininet v2.1.0, https://github.com/mininet/mininet/ [Access: June 9, 2015]
[3]Open vSwitch v2.3.1, http://openvswitch.org/ [Access: June 9, 2015]

Telekom [10]. This topology comprises three different parts, the *inner core* (IC), the *outer core* (OC), and the regional or *aggregation* (AGG) network as depict in Figure 4.

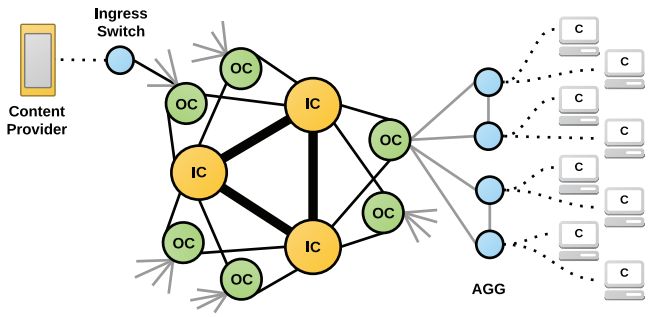


Fig. 4. The three-part ISP topology and the connected entities: *Inner Core* (IC), *Outer Core* (OC), and *Aggregation* (AGG).

The details of the aggregation and access network are abstracted for this study, assuming that clients are directly connected to one of the AGG switches. IC switches are interconnected as a full mesh, OC switches are connected to two IC switches each, and AGG switches are connected to a single OC switch as well as to one or two other AGG switches. To ease the deployment and configuration, the content provider, similar to the clients, is connected to a randomly chosen AGG switch, which in this case becomes the ingress switch for the scenario. The number of switches in the different areas was varied to maintain the characteristic structure of the network but investigate different network sizes. Table I summarizes the different topology parameters used. If not otherwise stated, the underlined (default) parameters are used.

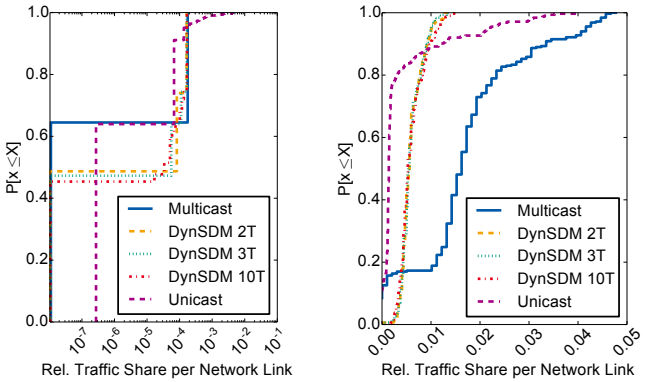
TABLE I. SCENARIO PARAMETERS (DEFAULT VALUES UNDERLINED).

Scenario Parameter	Variations
IC switches	1, 3, 6, 9, 30
OC switches	3, 7, 9, 12
AGG switches per OC switch	3, 7, 9, 12
Number of active clients	1, 15, 31, 63, 126, 189

As a typical application scenario, the delivery of live video streams is considered. To study the core mechanisms of DYNSDM in the emulation environment and the specified topology sizes, the actual delivery of video streams was required to be simplified. A realistic packet rate was derived, while actual video payload was not sent to reduce the load on the software switches, allowing to emulate larger scenarios on the used emulation server machine. The packet rate was calculated in the following way: An Ethernet MTU of 1,500 Bytes is assumed as well as a maximum video payload of 1,000 Byte, after subtraction of delivery protocol overheads. Of this remaining payload size, roughly 10% are required for transport stream encoding, e.g. as typically seen for MPEG-TS [4]. For a typical bitrate of 2.5 Mbit/s as recently reported for Akamai-based video streaming [20], this results in 343.75 pps or 1 video packet every 2.909 ms being delivered. This rate was used for delivering emulated video packets for all scenarios used in the following. The delivered traffic volume is deduced from the number of delivered video packets, multiplied by the above stated packet size.

A. DYNSDM *multi-tree parameters*

The most important parameter of DYNSDM is the number of subtrees used per multicast group. To show that the multi-tree approach improves the traffic distribution in the ISP network, random multicast groups were generated with the default topology size and a number of active clients as stated in Table I. Each experiment was repeated 50 times for varying numbers of subtrees, where the single-tree variant is named *multicast*. For a comparison, also a *unicast* delivery was performed with the same settings and using the shortest paths between content provider and each individual client. This case, thus, describes a traditional OTT delivery where each individual stream is routed according the shortest path in the topology. Figure 5 shows the resulting traffic distribution as the relative traffic share per link in the topology.



(a) Distribution per group. (b) Accumulated for 50 groups.

Fig. 5. Traffic distribution over links for different number of subtrees.

Here, Figure 5a depicts the distribution for one group over 50 repetitions and Figure 5b the overall traffic distribution within the topology after the 50 repetitions. In both figures, a clear difference between the traffic distributions of unicast, single-tree multicast, and DYNSDM is observable. As expected, unicast results in a distribution with a wide range of loads, where most load is carried by a small fraction of links (note log scale of first figure). Multicast removes this inequality between links, resulting in 36% of the links being equally used to deliver the stream. The DYNSDM variants further distribute the load, reducing the unused share of links from 64% (1 tree) to 49% (2 subtrees), 47% (3 subtrees), and 45% (10 subtrees). This translates to a reduction in unused links by 15-19%. Besides, the distribution of shares changes as smaller streams are distributed across more links. Figure 5b shows the effect for the delivery of a large number of OTT streams in parallel. It depicts the aggregated statistics as retrieved from the flow statistics of the switches after running 50 random multicast groups on the same topology. Here, the strength of the multi-tree approach is clearly visible, where links are used in a more equal manner (see steepness of DYNSDM curves). To capture this difference in an intuitive measure, the *fairness index* metric by Raj Jain [17, p. 36] is used. It translates the traffic share distribution to a single fairness value and is calculated using (2), based on the observed per-link traffic shares $(x_1, .., x_n)$.

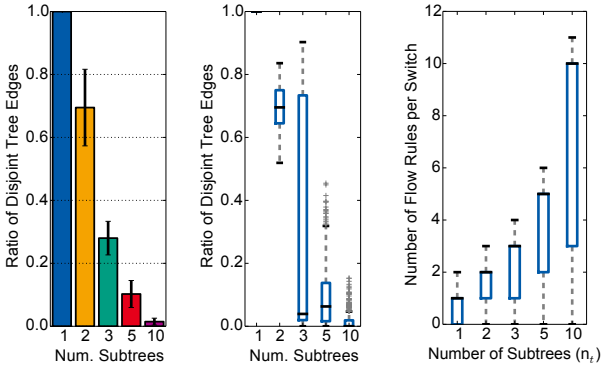
$$f(x_1, .., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \quad (2)$$

The metric ranges between 0 and 1, where 1 depicts the highest fairness and the best distribution of load across the links. Similar to the CDFs, this metric is calculated over all shares of the 50 repetitions, leading to the results as listed in Table II.

TABLE II. OBSERVED FAIRNESS OF TRAFFIC DISTRIBUTION.

Approach	**Fairness index** (per group)	**Fairness index** (50 groups)
Unicast	0.0466	0.2381
Multicast	0.3552	0.7131
DYNSDM 2T	0.4621	0.8839
DYNSDM 3T	0.4527	0.8905
DYNSDM 10T	0.4459	0.8529

Here, it can be observed that DYNSDM and its multi-tree variants clearly achieve a higher fairness and thus better traffic distribution. Besides, it is to note that increasing the number of subtrees to more than two, does only result in minor or no improvement of the distribution. The reason for this behavior can be found in the used ISP topology. Figure 6a shows the ratio of disjoint edges that the different approaches are able to establish. It is not surprising that for a single tree all edges are disjoint. Yet, for an increasing number of subtrees, the number of joint edges quickly rises. Already with 3 trees, the median of disjoint edges drops below 5%. This clearly is a result of the ISP topology structure, were only a limited number of alternative paths exists. Figure 6b additionally shows the costs in terms of network state (i.e. the number of flow rules per switch). It shows that the network state linearly increases (correlation coefficient: $r = 0.999$) with the number of subtrees. This was expected since for each subtree a separate flow is installed at the switches. In sum, the distribution characteristics and cost dependencies led to DYNSDM with 2 subtrees being considered the best choice for the used type of topology. For other topologies more subtrees could be desired to further improve the traffic distribution.



(a) Disjoint edges. (b) Costs: network state.

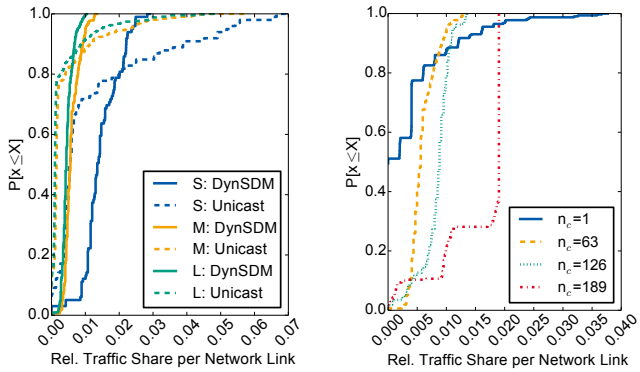
Fig. 6. Sensitivity analysis: Influence of number of subtrees.

As last part of the sensitivity analysis, Figure 7 shows how the traffic share achieved by DYNSDM with two subtrees is influenced by different topology sizes and client populations. For the topology sizes, Table III lists the different topology configurations and their names as used in the following. For an increasing ISP topology size, it can be observed that the fairness index for unicast decreases as more links are added but cannot be used (f_S=0.3866, f_M=0.2281, f_L=0.1933). DYNSDM, in contrast is able to maintain a stable and high fairness index (f_S=0.8928, f_M=0.8838, f_L=0.8912). For the costs (not shown here), only minimal changes are observed as the overall network state mainly becomes more distributed and only slightly increases in case the SPTs include more switches.

TABLE III. TOPOLOGY CONFIGURATIONS AND THEIR NAMES.

Topology Name	**IC Switches**	**OC Switches**	**AGG Switches**[4]	**Number Hosts**	**Number Receivers**
Small (S)	3	6	3	54	18
Medium (M)	3	9	7	189	63
Large (L)	5	12	7	252	84

For changing client populations, the fairness index for DYNSDM stepwise increases until 126 active clients and slightly drops again for 189 clients (f_1=0.2728, f_{63}=0.8839, f_{128}=0.9134, f_{189}=0.87254). The reason for this drop can be seen in Figure 7b, where the CDF for n_C=189 shows clear steps, caused by links that are used more than once by different subtrees and, thus, carry more traffic, which is in line with the above observations on edge disjointness. For the costs (not shown here), more receivers lead to more flow rules being installed. Yet, no linear correlation could be observed (r=0.848) as for a small number of receivers the number of rules increases quickly until most switches are involved in the delivery. From there on, the increase is smaller, converging to a median of just below 2 rules per switch and group, i.e. roughly 1 rule per subtree. These results show that DYNSDM scales well with the topology size and number of clients.



(a) ISP topology size. (b) Client population.

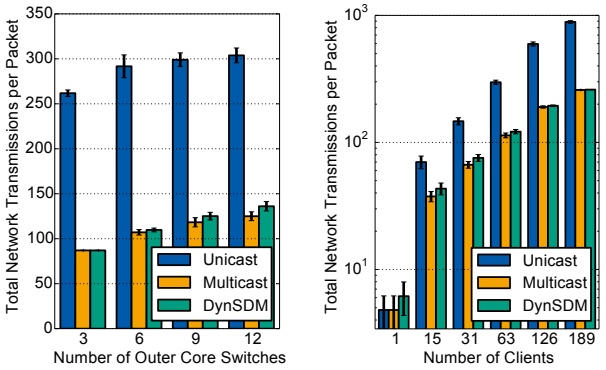
Fig. 7. Sensitivity analysis: Influence of scenario parameters.

B. DYNSDM *efficiency*

For the efficiency, it is important to show that introducing the multi-tree approach, has only limited impact on the overall amount of traffic. By that, the traffic should be well below the traffic of unicast deliveries to maintain the original incentive for the ISP to use an SDM-based service. Therefore, Figure 8 compares the total number of network transmissions (per video packet) performed to serve all active clients for different

[4]Aggregation switches per outer core switch

topology sizes (exemplary shown for number of OC switches) and client populations. Each time an individual packet traverses a link of the network it is counted. From this metric, the total network traffic can be easily calculated by multiplying the total number of transmissions with the packet rate and bitrate of the stream. The results show that the efficiency of DYNSDM, in average, is at the same level as the single-tree multicast and, as expected, way below unicast. This is true for all studied topology sizes and client population, confirming the efficiency of the approach following the relevant aspects proposed in [18].



(a) ISP topology parameters. (b) Number of clients.

Fig. 8. Traffic reduction for different exemplary scenario parameters.

V. RELATED WORK

In 1999, Keshav and Paul [18] proposed a separation of control and data plane of IP multicast, introducing a logically centralized control entity. This work coins fundamental concepts that inspired today's SDN-based multicast solutions. Although it is not fully compatible with recent SDN/OpenFlow works, conceptually, the work can be seen as an archetype for core parts of DYNSDM and other state-of-the-art approaches. Yu et al. [36] propose OFM (OpenFlow Multicast) which follows this approach and centralizes the multicast control at the controller of an SDN/OpenFlow network. They show that this centralization allows reducing the join time for new clients. While they do not introduce particular new functional aspects relevant to DYNSDM, they consider an interesting multi-controller scenario, where controllers can agree on a common multicast tree connecting their individual networks. With the same objective to reduce group events delays, Marcondes et al. [24] introduce CASTFLOW. The approach pre-calculates network-wide per-group MSTs, allowing the controller to quickly push newly populated parts of the tree to the switches. This way, CASTFLOW trades a reduced processing time during group events for increased memory requirements to manage the MSTs. In realistic ISP scenarios with millions of clients, this would imply a substantial overhead. Therefore, DYNSDM uses a more efficient BFS-based attachment from [23], avoiding unnecessary calculations and making groups easier to manage.

Kotani et al. [19] present an SDN-based multicast approach based on IGMP and using multiple data-path trees to reduce packet losses on network link failures. Inspired by [33], they enable pre-installed backup trees on link failures. While DYNSDM's multi-SPT calculation approach was inspired by this work, the work does not improve the traffic distribution as only one tree is active at a time. Zou et al. [38] also use IGMP but focus on secure group management. Similar to DYNSDM, the controller implements the group and tree management. The approach differs from DYNSDM in that it uses a single data-path tree and does not consider traffic distribution features.

Finally, Lee et al. [23] present a closely related approach that, in fact, inspired some mechanisms of DYNSDM. A major difference is that DYNSDM is a general multicasting service, while Lee et al. specifically focus on video streaming. Most commonly, both aim at traffic distribution across network links, although, Lee et al. motivate it with robust stream delivery, which in DYNSDM is only considered a positive side effect. To achieve robustness, they use a loss-tolerant multi-layer video codec and deliver the individual layers over different subtrees. This allows compensating packet losses by reducing the video quality. The approach is limited to the used video codec, which contradicts the objective of DYNSDM to be transparent to the clients. Besides, the approach is not transparent to the content provider due to the content encoding and explicit substreams exposed to the network. DYNSDM, in contrast, splits the multicast stream at the network layer, making the multi-tree delivery transparent for the content provider. Other contributions by the authors are seen orthogonal to this work.

VI. CONCLUSION

In this paper, DYNSDM is presented, introducing a novel approach to make OTT multicast traffic delivered by SDM more flexible and to achieve a better traffic distribution across links in the ISP network. For this, it proposes a mechanism to enable a fully network-supported multi-tree delivery of multicast streams that is transparent to the content provider and the clients. DYNSDM also adds the so far missing support for dynamic multicast groups and the ability to react on network events, such as link failures. The result of the emulation-based evaluation using realistic ISP topologies allowed to derive adequate settings for core parameters of DYNSDM and shows their impact on the traffic distribution. Using the multi-tree approach enables tuning the traffic distribution of DYNSDM to fully exploit the actual ISP network structure and, thereby, achieve a fair balancing of traffic across the links of the network. DYNSDM is shown to be scalable with the network topology size and the number of active clients with limited costs in terms of network state. Its traffic efficiency is at the level of single-tree multicast, where the traffic also scales linearly with the topology size and the number of clients, as desired for multicast approaches. Dynamic reaction on group and network events is possible at low costs in terms of flow rule changes as presented in more detail in the extended version of this work [28]. Overall, DYNSDM shows to be highly promising to be applied in a future ISP scenario, finally enabling an efficient and well-managed delivery of OTT multicast traffic, maintaining traffic efficiency and full flexibility for the ISP.

ACKNOWLEDGMENT

This work has been funded in parts by the European Union (FP7/#317846, SmartenIT and FP7/#318398, eCOUSIN) and the German Research Foundation (DFG) as part of project C03 within the Collaborative Research Center (CRC) 1053 – MAKI. The authors would like to thank Dominik Stingl and Timm Wächter for their valuable input and contributions.

REFERENCES

[1] L. A. Adamic and B. A. Huberman, "Zipf's Law and the Internet," *RAM Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002.

[2] C. F. Bazlamaçcı and K. S. Hindi, "Minimum-weight Spanning Tree Algorithms a Survey and Empirical Study," *Elsevier Computers & Operations Research*, vol. 28, no. 8, pp. 767–785, 2001.

[3] J. Blendin, J. Rückert, T. Volk, , and D. Hausheer, "Adaptive Software Defined Multicast," in *IEEE Conference on Network Softwarization (NetSoft)*, 2015.

[4] Brightcove Inc., "Announcing the Cloud's Most Efficient HTTP Live Streaming," Blog post, Dec 2011, http://blog.zencoder.com/2011/12/08/announcing-the-clouds-most-efficient-http-live-streaming/ [Accessed: August 24, 2015].

[5] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2013 – 2018," Tech. Rep., 2014.

[6] S. E. Deering, "Host Extensions for IP Multicasting," RFC 1112, Aug 1989. [Online]. Available: https://tools.ietf.org/rfc/rfc1112.txt

[7] Die Medienanstalten, "Digitisation 2013 - Broadcasting and the Internet - Thesis, Antithesis, Synthesis?" Tech. Rep., 2013.

[8] E. Dijkstra, "A Note on two Problems in Connexion with Graphs," *Springer Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[9] 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.

[10] M. Dueser, A. Gladisch, M. Jaeger, F. Westphal, and H. Foisel, "Evaluation of Next Generation Network Architectures and Further Steps for a Clean Slate Networking-Approach," *ITG EuroView, Presentation*, 2006. [Online]. Available: http://www3.informatik.uni-wuerzburg.de/euroview/2006/presentations/talk_Dueser.pdf

[11] Ericsson ConsumerLab, "TV and Media 2014 - Changing Consumer Needs are Creating a New Media Landscape," Tech. Rep., 2014.

[12] M. R. Garey, R. L. Graham, and D. S. Johnson, "The Complexity of Computing Steiner Minimal Trees," *SIAM Journal on Applied Mathematics*, vol. 32, no. 4, pp. 835–859, 1977.

[13] M. Gunkel, F. Wissel, W. Weiershausen, M. Franzke, V. Fürst, and A. Mattheus, "Multi-layer interworking with rate-adaptive transmission technology-benefit and challenges of a new use case," in *Proceedings of the VDE/ITG Symposium on Photonic Networks*, 2015.

[14] G. Hasslinger and F. Hartleb, "Content Delivery and Caching from a Network Provider's Perspective," *Computer Networks*, vol. 55, no. 18, pp. 3991–4006, 2011.

[15] H. W. Holbrook and D. R. Cheriton, "IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications," vol. 29, no. 4, pp. 65–78, 1999.

[16] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A Survey of Application-layer Multicast Protocols," *IEEE Communications Surveys and Tutorials*, vol. 9, no. 3, pp. 58–74, 2007.

[17] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[18] S. Keshav and S. Paul, "Centralized Multicast," in *IEEE International Conference on Network Protocols (ICNP)*, 1999.

[19] D. Kotani, K. Suzuki, and H. Shimonishi, "A Design and Implementation of OpenFlow Controller Handling IP Multicast with Fast Tree Switching," in *IEEE/IPSJ International Symposium on Applications and the Internet (SAINT)*, 2012.

[20] D. K. Krishnappa, M. Zink, and R. K. Sitaraman, "Optimizing the Video Transcoding Workflow in Content Delivery Networks," in *ACM Multimedia Systems Conference (MM)*, 2015.

[21] Y. Liu, Y. Guo, and C. Liang, "A Survey on Peer-to-Peer Video Streaming Systems," *Peer-to-Peer Networking and Applications*, vol. 1, pp. 18–28, 2008.

[22] Y. Liu, Z. Liu, X. Wu, J. Wang, and C. Yang, "IPTV System Design: An ISP's Perspective," in *IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2011.

[23] M.-W.Lee, Y.-S. Li, X. Huang, Y.-R. Chen, T.-F. Hou, and C.-H. Hsu, "Robust Multipath Multicast Routing Algorithms for Videos in Software-Defined Networks," in *IEEE International Symposium of Quality of Service (IWQoS)*, 2014.

[24] C. Marcondes, T. Santos, A. Godoy, C. Viel, and C. Teixeira, "CastFlow: Clean-Slate Multicast Approach using In-Advance Path Processing in Programmable Networks," in *IEEE Symposium on Computers and Communications*, 2012.

[25] M. Médard, S. Finn, and R. Barry, "Redundant Trees for Preplanned Recovery in Arbitrary Vertex-redundant or Edge-redundant Graphs," *IEEE/ACM Transactions on Networking (TON)*, vol. 7, no. 5, pp. 641–652, 1999.

[26] B. Pfaff, B. Lantz, and B. Heller, *OpenFlow Switch Specification, Version 1.3.0*, Open Networking Foundation, 2012.

[27] K. Pires and G. Simon, "YouTube Live and Twitch: A Tour of User-Generated Live Streaming Systems," in *ACM Multimedia Systems Conference (MMSys)*, 2015.

[28] J. Rückert, J. Blendin, R. Hark, T. Wächter, and D. Hausheer, "An Extended Study of DynSdm: Software-Defined Multicast using Multi-Trees," Peer-to-Peer Systems Engineering Lab, TU Darmstadt, Germany, Tech. Rep., 2015, http://www.ps.tu-darmstadt.de/fileadmin/publications/PS-TR-2015-01.pdf.

[29] J. Rückert, J. Blendin, and D. Hausheer, "RASP: Using OpenFlow to Push Overlay Streams into the Underlay (Demo Paper)," in *IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2013.

[30] ——, "Software-Defined Multicast for Over-the-Top and Overlay-based Live Streaming in ISP Networks," *Springer Journal of Network and Systems Management (JNSM), Special Issue on Management of Software-defined Networks*, vol. 23, no. 2, 2015.

[31] Sandvine, "Fall 2014 Global Internet Phenomena Report," 2014.

[32] R. K. Sitaraman, M. Kasbekar, W. Lichtenstein, and M. Jain, "Overlay Networks: An Akamai Perspective," in *Advanced Content Delivery, Streaming, and Cloud Services*. John Wiley & Sons, 2014.

[33] D. Wang and G. Li, "Efficient Distributed Bandwidth Management for MPLS Fast Reroute," *IEEE/ACM Transactions on Networking (TON)*, vol. 16, no. 2, pp. 486–495, 2008.

[34] R. White, J. Ng, D. Slice, and S. Moore, "Enhanced Interior Gateway Routing Protocol," RFC Draft, April 2014. [Online]. Available: https://tools.ietf.org/html/draft-savage-eigrp-02

[35] G. Xue, L. Chen, and K. Thulasiraman, "Quality-of-Service and Quality-of-Protection Issues in Preplanned Recovery Schemes Using Redundant Trees," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 8, pp. 1332–1345, 2003.

[36] Y. Yu, Q. Zhen, L. Xin, and C. Shanzhi, "OFM: A Novel Multicast Mechanism Based on OpenFlow," *Advances in Information Sciences and Service Sciences*, vol. 4, 2012.

[37] X. Zhang and H. Hassanein, "A Survey of Peer-to-Peer Live Video Streaming Schemes - An Algorithmic Perspective," *Computer Networks*, vol. 56, no. 15, 2012.

[38] J. Zou, G. Shou, Z. Guo, and Y. Hu, "Design and Implementation of Secure Multicast based on SDN," in *IEEE International Conference on Broadband Network Multimedia Technology (BNMT)*, 2013.