An Extended Study of DynSdm: Software-Defined Multicast using Multi-Trees

Julius Rückert, Jeremias Blendin, Rhaban Hark, Timm Wächter, and David Hausheer

Technical Report – PS-TR-2015-01

An Extended Study of DynSdm: Software-Defined Multicast using Multi-Trees
Julius Rückert, Jeremias Blendin, Rhaban Hark, Timm Wächter, and David Hausheer
Technical Report – PS-TR-2015-01
http://www.ps.tu-darmstadt.de/

First published: June 14, 2015
Last revision: June 15, 2015

For the most recent version of this report see
http://www.ps.tu-darmstadt.de/fileadmin/publications/PS-TR-2015-01.pdf

Technische Universität Darmstadt
Department of Electrical Engineering and Information Technology
Institute of Computer Engineering

Peer-to-Peer Systems Engineering
Prof. Dr. David Hausheer

Contents

List of Figures

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 allowing to evenly balance the load on links inside the ISP network, introducing mechanisms to handle group and network dynamics, and proposing a novel SDN-based solution to the service discovery problem along a routing path. DYNSDM was prototypically evaluated, showing its high traffic efficiency at the level of single-layer multicast, its good scalability, and superior traffic distribution characteristics.

1 Introduction

Today, an increasing amount of Internet traffic is caused by over-the-top (OTT) services, where new services and content providers are seen on a regular basis. 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 certainly be seen in missing practical service models as well as missing abstractions to allow realizing new services in a quick and flexible manner. A large part of today's OTT traffic consists of video streams [San14, Cis14, Eri14] that could greatly benefit from a number of network services, most prominently from network-layer multicasting to more efficiently deliver, e.g., the increasing amount of broadcasting content on the Internet [Die13]. Nework-layer multicast support on an Internet-scale showed to be hard to achieve in practice [DLL+00], where IP multicast [Dee89] failed to become a solution that spans more than individual network islands. By now, it seems to be reasonable to assume that a comparable solution following the initial IP multicast design is not expected to become reality, given the structure of the Internet, the involved stakeholders, and the technical challenges of the approach itself [DLL+00].

Today, the delivery of OTT multicast streams is enabled mainly by Content Delivery Networks (CDNs) that emerged as alternative and implment the multicasting functionality at application layer forming a routing overlay among hundred thousands of deployed servers [SKLJ14]. After spreading the content across the nodes, they act as proxies and serve clients in a client-server manner. While this approach shows to scale well up to the nodes' resource limits, CDNs usually end at the edge of ISP networks as ISPs fear them being unpredictable traffic sources inside their well-managed networks [HH11]. This situation leaves the ISPs with most of the traffic load on their networks, where multicast traffic is delivered through the ISP as per-client unicast streams. This is unfortunate as ISPs themselves very well know how efficient multicasting in their own network can be realized. Indeed, they widely use it to delivery their own ISP-internal IPTV service to their broadband access customers [LLW+11].

In a previous work by the authors, Software-Defined Multicast (SDM) is proposed to address this problem in a way that both content providers, or CDNs that act in behalf of them, and ISPs can benefit from. Using the concept of Software-Defined Networking (SDN), it enables a practical service model for one-to-many multicasting services, e.g. for live video and audio streaming. Such service models for multicasting were already discussed by Holbrook et al. [HC99] in 1999 but only recently became practically feasible using the concept of SDNs and actual implementations, e.g., based on the OpenFlow protocol. Using the SDN concept, especially ISP can gain a more natural control over their network by introducing a clear separation between control and data plane. Following this general idea, SDM splits control and data delivery of multicast by introducing a well-defined control-level API run by the ISP that OTT providers can use to create an manage multicast groups to serve the ISPs customers with an OTT stream. For the data path, SDM uses OpenFlow features to enable an efficient, scalable, and transparent delivery of the streams to the clients (cf. Section. 2 for more details).

While SDM coins fundamental concepts for SDN-based OTT multicasting in ISPs, it leaves some important questions unanswered that require a deeper study to allow for a practical applicability in realistic scenarios. First of all, by making OTT stream delivery excplict to the ISP, a more efficient multicast delivery of the stream becomes possible. Yet, for the process of calculating the actual data paths, resulting in the mulitcast trees installed, a simplified approach was taken that does not answer how exactly ISPs can plan multicast trees in line with their internal traffic engineering. Thus, the multicast data paths of SDM

are rather inflexible and lead to a very unbalanced distribution of load across links of the ISP network. Second, SDM lacks mechanisms to support dynamic multicast groups with changing client populations as well as a mechanism to react on network dynamics, such as link failures or load changes. Third, it does not answer the question on how OTT content providers can actually discover the availability of an SDM service instance. This is a rather complex problem as content providers, in general, cannot be assumed to even be aware of the detailed delivery path for their streams across the Internet and, thus, might not even know that an SDM-enabled ISP is involved.

To this end, DYNSDM (Dynamic Software-Defined Multicast) proposes a number of extensions to the original SDM approach to answer the above questions. Thus, the contributions of this paper are fourfold: (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 porposed to better distribute traffic across network links; (3) mechanisms for supporting dynamic groups and fast reactions on link failures or intentional exclusions of links are presented; (4) a novel service discovery approach is described that allows discovering services on a routing path, instead if its location. The mechanisms of DYNSDM are shown to be efficient and scalable, making the general SDM idea more practical and, thus, strengthening its applicability.

The remainder of the paper is structured as follows: Section 2 provides an overview on the original SDM approach and one of its recent extensions. Section 3 presents the design of DYNSDM. Subsequently, Section 4 presents a detailed evaluation of the design. Related works in the area of centralized and SDN-based multicast are discussed in Section 5. Finally, Section 6 concludes the paper.

2 Background: Software-Defined Multicast

In a previous work by the authors, Software-Defined Multicast (SDM) [RBH15] 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) [BRV+15] 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 [RBH13]. 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 own 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[1] and, optimally, parts of the ISP's aggregation network 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 (i.e. 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 matches 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 delivered to the clients. As a result, SDM is transparent to the clients that are not aware of the multicast delivery. Figure 2.1 summarizes the steps of the delivery.

In [BRV+15], 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 trade-off 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 [AH02]. 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

[1] In the SDN context, *switch* and *router* are used interchangable in this work.

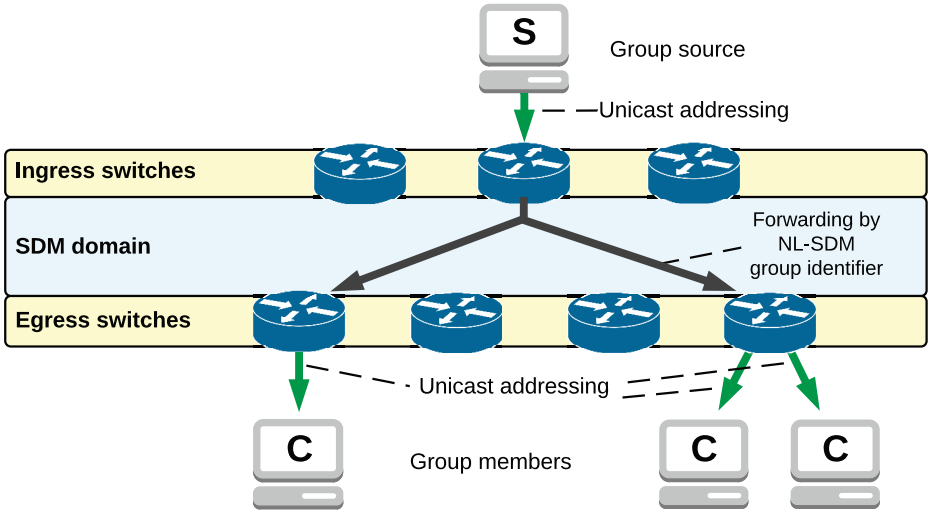


Figure 2.1: Individual steps of multicast delivery in SDM (adapted from [RBH15]).

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. 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 trade-off in that a translation is automatically performed by the switches where the remaining subtree holds less clients than the 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.

3 Dynamic Software-Defined Multicast

While SDM[RBH15] 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[BRV+15] extends SDM by the possibility to control the tradeoff between bandwidth and network state, thus addresses an orthogonal problem (cf. Section 2). In the following, the core of Dynamic Software-Defined Multicast (DYNSDM) is presented, which adds the actual traffic engineering support, which is considered highly relevant for a practical adoption of the overall SDM 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 even more fine-granular traffic engineering of the multicast traffic. Third, the mechanisms are presented that allow DYNSDM to dynamically react on changing client populations and network conditions, such as congestion or link failures. Finally, a practical service discovery approach is presented that allows OTT content providers to transparently discover SDM service instances on their delivery paths. The latter proposes a solution to the inherent discovering problem for cross-layer services in a large multi-tier environment, where OTT content providers need to be able to efficiently discover available services on the fly, thereby improving the practical applicability of SDM.

3.1 Multicast Tree Planning and Management

The multicast tree planning and management of DYNSDM is summarized in Figure 3.1, including the multi-tree planning as detailed in Section 3.2.

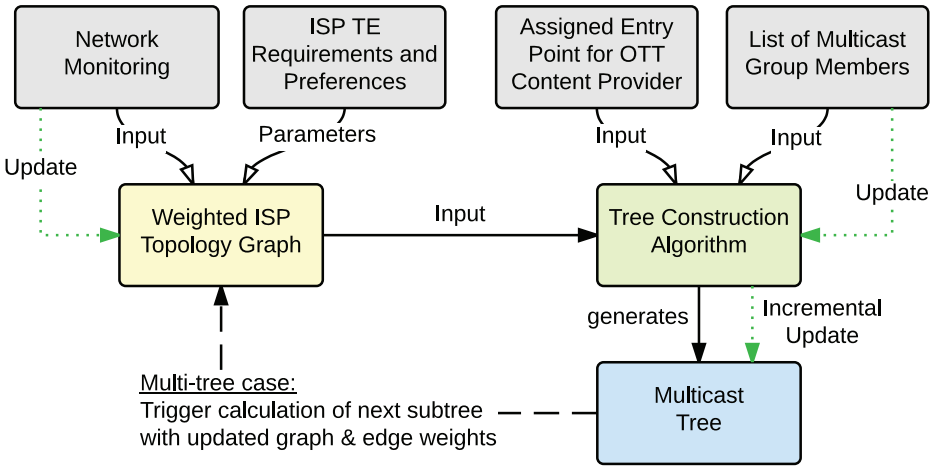


Figure 3.1: Overview of main multicast tree planning and management steps.

Graph model of the ISP topology

As a first step, the network topology of the ISP is modeled as a weighted graph. The information required for this step 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 network topology is assumed to consist of OpenFlow-enabled switches/routers and physical links between them. While, for simplicity, a

pure OpenFlow scenario is envisioned, hybrid networks could be supported as well with some requirements on the routing process of non-OpenFlow network devices, giving them a passive forwarder role in the multicasting. The graph $G = (V, E)$ derived in this step models the network topology, representing each switch as a vertex $v \in V$ and each link as edge $e \in E$ between vertices.

Calculating edge weights

As a next step, edge weights are assigned to 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 edges and vertices in this step, resulting in a sub-graph $G_{MC} \subseteq G$. The calculation of the weights is performed based on monitoring information about the current traffic conditions, individual Quality-of-Service (QoS) metrics, and the available resources of network links. For this, the ISP defines the importance of the individual properties and network aspects and, thus, enables a delivery in line with the its overall traffic engineering approach. To calculate a single weight per edge, an 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) [WNSM14], DynSdm calculates the weight of an edge e using (3.1).

$$\begin{aligned} \text{weight}_e := (K_1 \times \text{bandwidth}_e) + (K_2 \times \text{utilization}_e) + (K_3 \times \text{delay}_e) \\ + (K_4 \times \text{lossrate}_e) + (K_5 \times \text{failurerate}_e) \end{aligned} \tag{3.1}$$

Here, different monitored QoS metrics of a link are aggregated to derive a single edge weight as input for the multicast tree construction algorithm. Depending on the requirements and specifics of the ISP, more metrics could be added to guide the mechanism in the link selection process. The coefficients K_1 to K_5 can be used to tune the importance of individual QoS parameters for the final weights. The idea is to provide ISP's with a powerful mechanisms that they can easily parameterize according their topology and requirements. The process of tuning these parameters is not covered in this work as it is not different from tuning similar parameters in routing protocols such as the one mentioned above. Yet, it is important to highlight that there is less danger to actual cause instability by wrong settings. The reason is that, in contrast to distributed routing protocols, routing convergence is not an issue due to the logically centralized control and tree calculation in this SDN environment. In the worst case, the tree construction algorithm builds non-optimal trees, their installation in the network, however, is possible without causing routing inconsistencies.

Multicast tree construction

For the actual multicast tree construction for a registered multicast group, two more pieces of information are required: the entry point, i.e. the switch at that the unicast traffic send by the content provider enters the ISP network as well as the list of group members to be served. Both are available at the service API machine, which is used by the content provider to interface with the ISP for registering and managing its multicast groups. The entry point is defined by the *group socket* assigned by the ISP on group registrations as introduced in Section 2. By announcing BGP routes to the subnets used for DynSdm group sockets, the ISP can influence where the respective traffic enters its network, thereby defining the entry point required for the tree construction algorithm. The actual construction of the tree is be 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. Depending on the requirements on the resulting multicast tree, different algorithms can be applied for this task. In contrast to traditional multicast approaches, no distributed tree calculation protocol is required and, thus, a variety of existing 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 in G_{MC}, this problem can be classified as a Steiner tree problem, which is well known to be NP-hard [GGJ77]. Yet, a set of heuristics and approximation algorithms exist that can be applied to build nearly optimal trees more efficiently [BH01]. 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 right 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 [Dij59].

Network-layer path setup

As a last step, the constructed 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. 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.

3.2 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 [ZH12] and application-layer multicasting [HASG07], 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 become instable in case of dynamics in the client population. Besides, they 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 [LGL08].

Translated to network-layer multicasting, were switches perform the duplication of traffic, splitting a monolithic multicast stream into smaller sub-streams 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 avoid other, more important and more rigid traffic in the network. While ISP network topologies themselves are typically rather static as such, they, nevertheless, have to cope with link outages on a regular basis, caused by cable breaks or failing network equipment. Therefore, traffic engineering mechanisms have to be able to dynamically adapt to changes in the network topology. Thus, by distributing multicast traffic over individual 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. Thus, in this work, also first ideas are shared on how such a repair and rerouting functionality could be realized. Yet, for brevity, the focus of this work is more on the traffic characteristics of the multi-tree approach as well as the impact of different scenario and system parameters on its performance and costs.

To realize a multi-tree extension to SDM, the tree planning and management mechanism is extended to iteratively calculate multiple multicast trees for the same entry point and client set (cf. Figure 3.1). For the multi-tree case, after building the first tree, more trees are generated iteratively, inspired by [KSS12]. By changing 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 achieved [MFB99, XCT03]. In reality, edge-disjointness showed 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 usually limited. Therefore, disjointness is targeted but not enforced if no alternative paths exist to allow multiple trees to be built even if they have some shared edges. The implications of this design choice are discussed when studying the choice for the number of DynSDM subtrees.

Data-plane traffic splitting

Besides the actual planning of the multiple trees, their efficient support at network layer is a key aspect to be discussed. 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, transparent to content provider and clients. This is achieved by mapping the 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 one key contribution of this approach is proposing the usage of OpenFlow's group table *select* type at the respective ingress switch of the multicast group as shown in Figure 3.2.

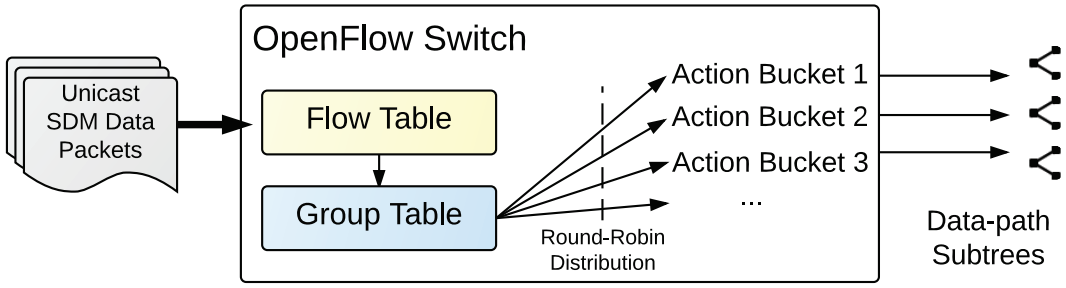


Figure 3.2: DynSDM traffic splitting based on OpenFlow group *select* feature.

The OpenFlow standard specifies this feature as being optional and being based on a switch-computed selection algorithm (cf. [PLH12]). 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 multicast group to different action buckets. While SDM would just rewrite and forward the packets according a single tree, DynSDM uses the *select* feature to splits the traffic into subsets that are delivered over individual trees (cf. Figure 3.3). 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 could be realized, e.g. using an ISP's knowledge on estimated path reliability or cost metrics to distribute traffic to the DynSDM subtrees with unequal shares.

Practical considerations

While the proposed multi-tree mechanisms has clear benefits, it is to note that it surely has some limitations that, depending on the scenario, might or might not be of practical relevance. One to be carefully considered is the impact of network delay on the multi-tree delivery. As DynSDM aims at building distinct and independent subtrees, it is likely that the paths between entry point and the same client in each tree

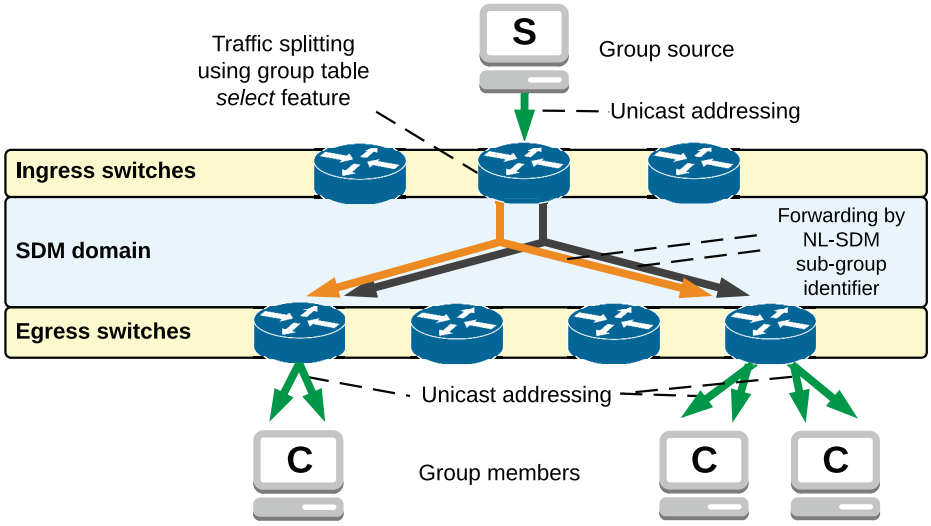


Figure 3.3: Multicast delivery in DynSdm using multiple delivery trees.

experience different transmission delays. As a result, the multicast traffic can be subject to permanent packet reordering, as packet delivery happens with a different delay on every subtrees. How large this difference in transmission delays in practice can be, clearly depends on the network. As DynSdm, similar to Sdm and traditional IP multicast, targets the delivery of connection less data streams, applications have to be able to deal with reordering. In case the reordering is not acceptable, subtrees could be built with low delay differences that are tolerable for the multicast application using the service. Certainly, video streaming applications with large buffers of several seconds 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 calculation of the subtrees.

3.3 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 management is handled completely by the OTT content provider. It is assumed that clients maintain a control connection to the server of the content provider and are served by using IP unicast. The content provider decides on which clients are to be included in the multicast service and either establish a new DynSdm group or add the client to an existing one using the service API. Successfully added to an active group, the client continues receiving the content provider's data stream, which is indistinguishable from the unicast stream. Such a dynamic behavior requires the new client to be attached to the active delivery process of a group. A trivial solution would be a re-initiation of the tree calculation process as presented in Section 3.1. Yet, this would cause an immense overhead as the optimal trees derived in the calculation process might look different with every new client. Therefore, DynSdm adopts an approach based on a mechanism proposed by Lee et al. [MWLH+14], attaching new clients by finding paths using individual breadth-first searches from the new client to the active subtrees. This way, new clients can quickly be connected by 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 approach can result in non-optimal tree structures over time, DynSdm checks the established structures on each event and can incrementally optimize them if required. For this, established tree structures are monitored and regularly compared to the optimal structures using a scoring function. Based on the comparison, DynSdm

can incrementally rebuild an optimal structure by only changing flow rules that differ from the existing configuration. This avoids costly and time-consuming rule changes, especially for large multicast groups. Changes are non-intrusive in that rules are only removed when new ones are installed, thus avoiding any negative impact on active multicast data deliveries. Figure 3.4 summarizes the steps.

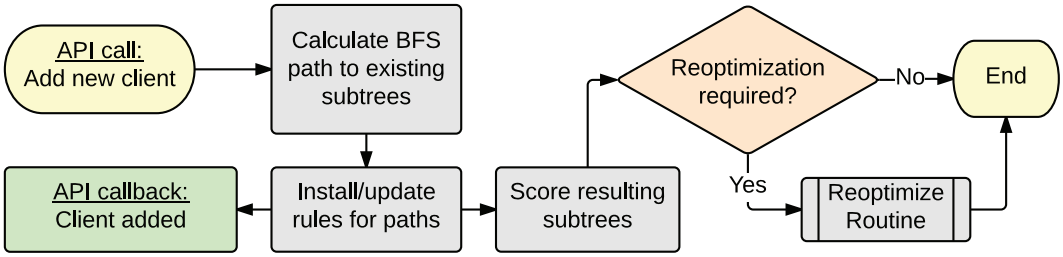


Figure 3.4: Steps taken to add new client to multicast group.

If a client is removed using the service API, DynSdm traverses the graph backwards starting from the client to the first switch performing duplication. At this switch, the client's branch is pruned by deleting its output action. The remaining flow rules of the branch can be scheduled for a later removal as no strict time constraints exist. 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 link is directly adjacent to the ingress switch. In this case, the *group table select feature* can automatically handle the failure as the link failure (i.e. the link is down) is directly detected at the switch and the select algorithm 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 be affected, thus the controller alone has to handle the outage. On failure detection, it immediately determines the affected trees and disables them by removing the affected action buckets from the group table of the ingress switch, also distributing the traffic to the remaining trees. In parallel, new subtrees are calculated, installed and, subsequently, enables by again adding a new action bucket. In the first case, no packet loss happens due to the automatic reroute, whereas in the second case, a certain packet loss is inevitable, yet only affecting a fraction of the multicast traffic due to the multi-tree distribution.

3.4 Transparent Service Discovery

As presented, DynSdm addresses a number of ISP-internal challenges with a focus on traffic engineering. In addition, a mechanisms is proposed that solves another major problem, the actual service discovery. This step is a prerequisite for any multicast delivery as the OTT content provider needs to locate the service, i.e. the service API endpoint of the ISP. While service discovery in general is a well-studied topic, it still is a hard problem in wide-area network scenarios. Within the same network domain or LAN, broadcasting, flooding, or multicast mechanisms have been used for this purpose [KK01, PFKL08, SSvdGS04, App13]. For multi-domain and wide-area use cases, IP multicast-based approaches [CZH+99, GWvB+01, VP97] and DNS-based solutions [Gul00] have been proposed. Besides, also overlay-based mechanisms were studied to address the problem in specific use cases [CDKR02]. Looking at the discovery for a network service function inside the network, it becomes clear that, in this case, the discovery of DynSdm or similar services has specific requirements that, to the best of the authors knowledge, are not met by any of the above mentioned approaches.

In an OTT scenario, content providers and ISPs, in general, cannot be assumed to explicitly cooperate. While the reason in most cases is non-technical, one reason is that on-the-fly and automatically establishing a relationship is technical hard to achieve. The OTT traffic is routed towards the clients using BGP, usually traveling a number of different networks on its path between the datacenter that the provider uses and the edge of the ISP network. While large CDN providers, such as Akamai, certainly have a very detailed view on the structure of the Internet and can optimize this path, the typical OTT provider might not be able and interested in this rather network-related view. Yet, for such an OTT provider it becomes very hard to abstract from the network and delivery path details, and, at the same time, make use of emerging cross-layer approaches. Therefore, DynSdm proposes a novel in-line service discovery approach that allows the content provider to effectively probe the delivery path for the availability of network services. The ISP or even a transport network on the path can register for the probes with little overhead by installing a single flow rule at one of the switches on the path. This way, the normal OTT-to-ISP relation or even more complex and recursive relations, e.g. with nested DynSdm support both within transport networks and the ISP, can be established.

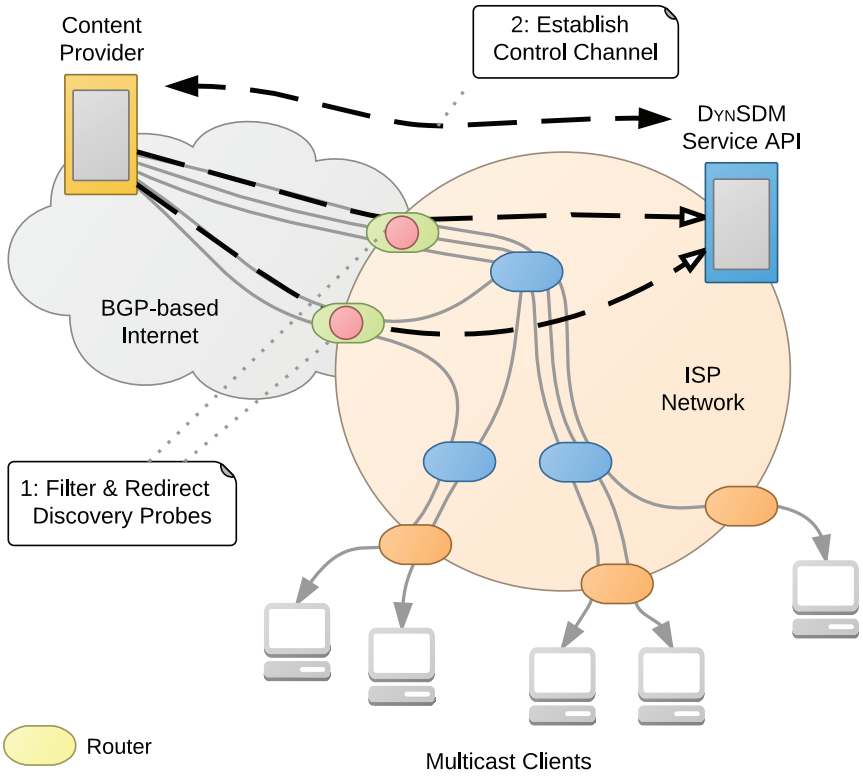


Figure 3.5: Transparent DynSdm discovery process.

Figure 3.5 shows a simplified instance of the in-line discovery use case. The content provider send probes, which are special IP packets, together with the OTT traffic of a client. The probe packets are addressed to the client and thus follow the same path as the rest of the traffic. To allow for an efficiently filtered of the probes by any network provider on the path, a new lightweight signaling protocol is introduced for this purpose. For this protocol it is proposed to register a new IP protocol number, used in the IP header to specify the next header protocol inside an IP packet. To allow for a Internet-wide support of the discovery protocol, the usage of the number should be standardized and allocate one of the so far unassigned protocol numbers[1], i.e. between 143 and 252. Based on such a well-defined protocol number, discovery probes can be efficiently filtered, i.e. using a single OpenFlow rule matching this protocol number as part of the IP header. For the prototypical implementation used in the evaluation, the protocol number 253 was used, which is reserved for experimentation and testing purposes. As

[1] `http://www.iana.org/assignments/protocol-numbers/protocol-numbers.txt` [Access: May 6, 2015]

alternative, filtering the probes could also be achieved by marking probes in any other header field that can be matched at network layer and is not used by other mechanisms. Yet, such a field is hard to find and is assumed to easily lead to packets being dropped, e.g. by firewalls. Therefore, the above approach is clearly preferred. Filtering of the probes in an ISP scenario could be done by installing respective flow rule on every ingress switch of the ISP as depict in Figure 3.5. After filtering, the network provider can forward the probes to the service API server or a separate machine to process the packets and answer them with respective service offers. To avoid DoS attacks on the service machine by sending a large number of probes, it is proposed to use the *meter* feature of OpenFlow to limit the bandwidth of forwarded probes directly at the ingress switch. Details for answering the probe can be included in the packet using a standardized protocol structure for the discovery protocol. For simplicity, the DynSdm prototype uses a custom JSON-based protocol for this purpose. With a service offer, the ISP can informs the content provider about the availability of the service and point him to the service API that, from here on, is used as control channel. In this work, the discovery is clearly envisioned to be used for DynSdm, yet it could be similarly applied to any other OTT service discovery that is bound to a delivery path.Especially CDNs are envisioned to greatly benefit from such a protocol, as it allows CDN nodes to discover and use network services in behalf of content providers.

Nested discovery scenarios

For the nested discovery application, e.g., a transport network with DynSdm support could answer the service probes and establish a multicast group from the ingress to the egress of its network. To allow networks down the path from the transport network to also apply the approach, the transport provider could inject new service probes into the multicast tree that, when leaving the transport network, are translated to unicast packets just as the normal multicast traffic. This way, the transport provider can become a user of DynSdm services in connected networks, e.g. other transport networks or ISPs. In this case, DynSdm service machines of the network providers establish a control channel to each other, effectively building a hierarchical relationship, where the OTT content provider only interacts with the first DynSdm providing network on the delivery path. The incentive for network providers to establish such a recursive service relationship is the shift of the multicast-to-unicast translation and, thus, reducing the outbound traffic from their own network.

4 Evaluation

Measurement Methodology

To investigate the applicability, performance, and costs of DynSdm, the core mechanisms of the approach 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 discovery protocol. Experiments were set up and 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 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 also allow splitting streams in a weighted manner. As discussed before, it is expected that upcoming commercial OpenFlow switches include an implementation of a similar select feature due to a number of other beneficial applications, e.g. for network-level load balancing and fast failover mechanisms.

The evaluation scenarios used in the following where chosen to allow a detailed studying of DynSdm's core mechanisms. Therefore, first 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 the first step, the traffic efficiency of DynSdm is investigated and compared to single-tree multicast and unicast. To complete the picture, the efficiency of DynSdm for handling dynamics is studied. Here, the focus is on the costs for relevant group and network events. In sum, the evaluation studies relevant aspects for multicast approaches as proposed by Keshav and Paul [KP99], plus the impact of DynSdm's multi-tree approach on the traffic distribution that is not considered for traditional multicasting. Besides *efficiency* and *scalability*, Keshav and Paul also name *incremental deployability* on an Internet-wide scale as a third main aspect that is not explicitly studied in this evaluation. This aspects is implicitly covered by Sdm and its derivates due to its well-defined scope to be applied to single network domains. Using the transparent service discovery mechanisms, in addition, individual DynSdm islands can be paired, allowing an incremental deployment on an per AS or domain basis. Within a single network domain, furthermore, DynSdm enables an incremental deployment as well, at the cost of a reduced efficiency in terms of the achievable traffic reductions. As DynSdm aims at 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 Telekom [DGJ+06]. This topology comprises three different parts, the *inner core* (IC), the *outer core* (OC), and the regional, here called *aggregation* (AGG), network as depict in Figure 4.1.

[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]

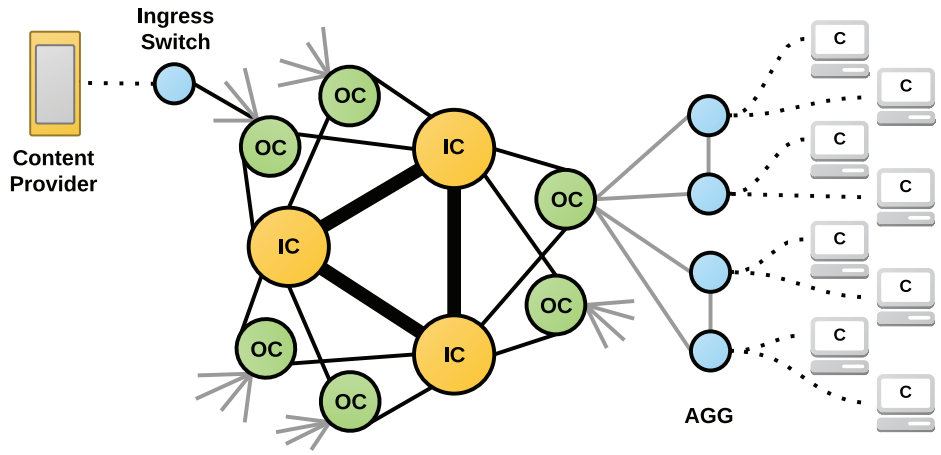


Figure 4.1: The three-part ISP topology and the connected entities: *Inner Core* (IC), *Outer Core* (OC), *Aggregation* (AGG).

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 keep the characteristic structure of the network but investigate different network sizes. Table 4.1 summarizes the different topology parameters used. If not otherwise stated, the underlined (default) parameters are used.

Table 4.1: 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[4]. 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 substraction 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[5]. For a typical bitrate of 2.5 Mbit/s as recently reported for Akamai-based video streaming [KZS15], this results in 343.75 pps or 1 video

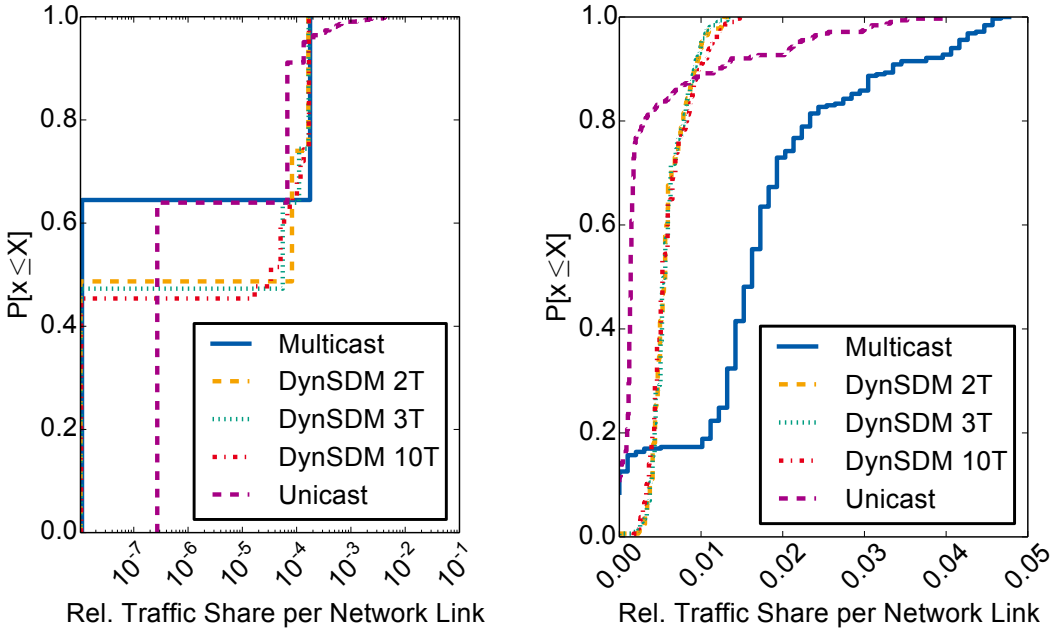
[4] Server details: 24x2.6GHz Intel(R) Xeon(R) CPU, 128GB main memory

[5] http://blog.zencoder.com/2011/12/08/announcing-the-clouds-most-efficient-http-live-streaming/ [Access: June 9, 2015]

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.

4.1 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 4.1. 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 4.2 shows the resulting traffic distribution as the relative traffic share per link in the topology.



(a) Distribution per group.

(b) Accumlated for 50 groups.

Figure 4.2: Traffic distribution over links for different number of subtrees.

Here, Figure 4.2a depicts the distribution for one single group over 50 repetitions and Figure 4.2b 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 the log scale of the 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 on more links, reducing the share of unused 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, also the distribution of shares changes as smaller streams are distributed across more links. Figure 4.2b shows the effect for the delivery of a large number of OTT streams in parallel on the traffic distribution. It depicts the aggregated statistics as retrieved from the flow rule 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 overall links are used in a more equal manner (see steepness of the DynSdm curves). To capture this difference in an easier to understand measure, a metric called *fairness index* described by Raj Jain [Jai91, p. 36] is used that translates the traffic share distribution to a single fairness value. It is calculated as depict in (4.1), using the set of per link traffic shares $(x_1, .., x_n)$ as input.

$$f(x_1, .., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \tag{4.1}$$

The index ranges between 0 and 1, where 1 depicts the highest fairness, i.e. 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 4.2.

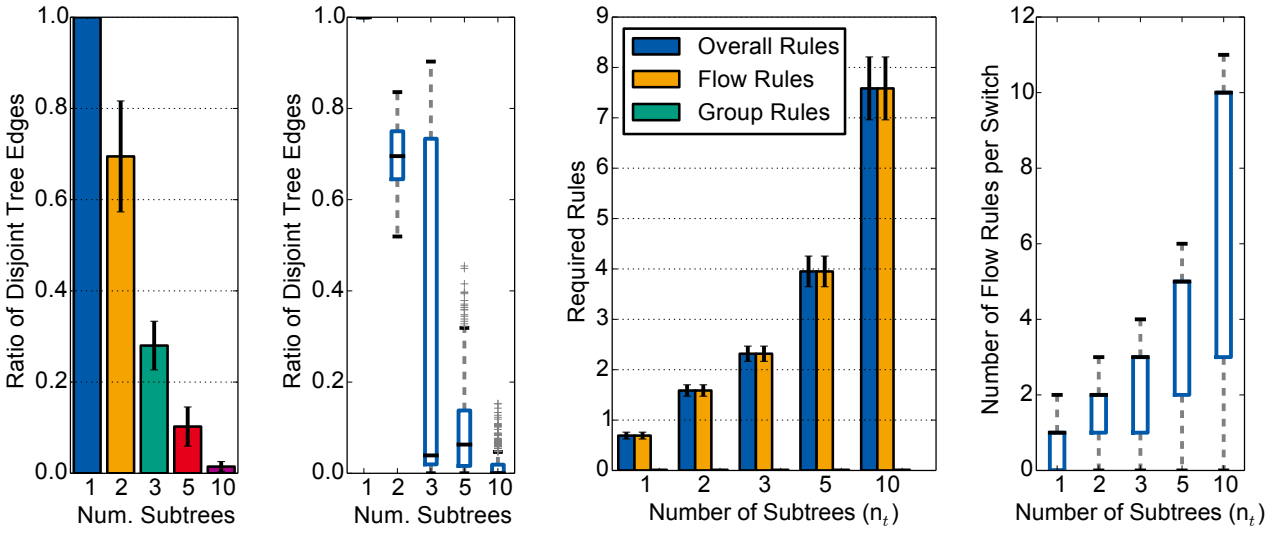
Table 4.2: Observed fairness of traffic distribution.

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 ISP topology used. Figure 4.3a shows the ratio of disjoint edges that the subtrees are able to establish for multicast and the different variants of DynSdm. 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 increases. Already with 3 trees, the median of disjoint edges is below 5%. This clearly is a result of the ISP topology structure, were only a limited number of alternative paths exist. Figure 4.3b in addition shows the costs in terms of network state (i.e. the number of flow rules) required for the different variants 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 every subtree a separate tree has to be installed at the switches. In sum, the distribution characteristics and cost dependencies led to DynSdm with 2 subtrees being considered the best choice for this type of topology. For other topologies more subtrees could be desired to further improve the traffic distribution.

As last part of the sensitivity analysis, Figure 4.4 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 4.3 lists the different topology configurations and their names used in the figure. 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 network state is mainly distributed to switches and only slightly increases in case the SPTs include more switches.

[6] Aggregation Switches per Outer Core Switch



(a) Disjoint edges.

(b) Costs: network state.

Figure 4.3: Sensitivity analysis: Influence of number of subtrees.

Table 4.3: Topology configurations and their names.

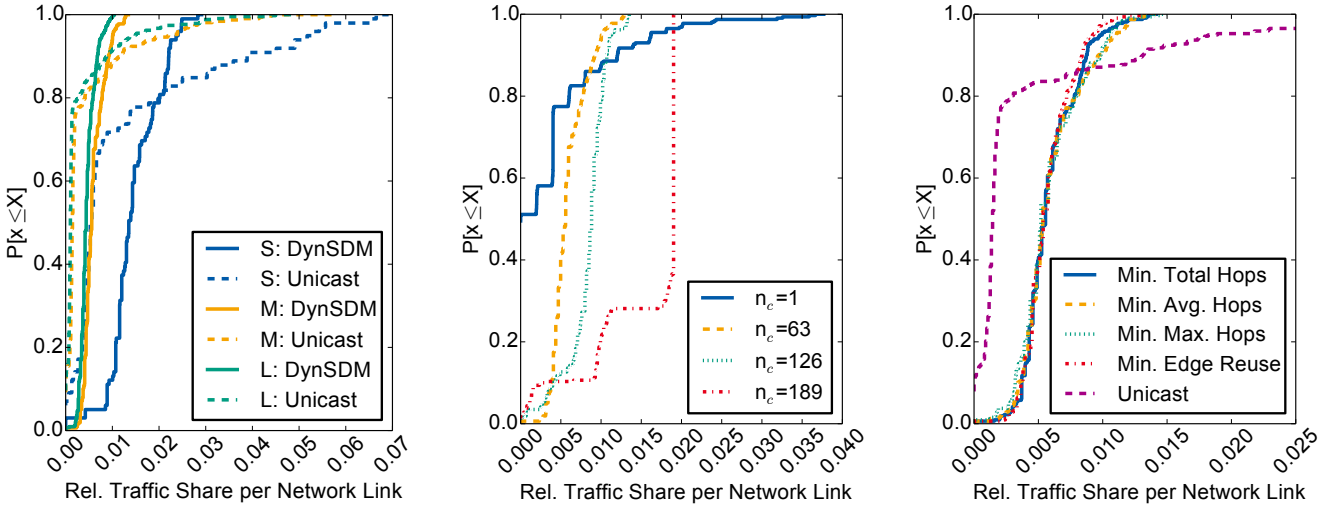
Topology Name	IC Switches	OC Switches	AGG Switches	Number of Hosts	Number Receivers
Small (S)	3	6	3	54	18
Medium (M)	3	9	7	189	63
Large (L)	5	12	7	252	84

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 4.4b, 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 required rules increases quickly until most of the switches are involved in the delivery. From there on, the increase in new rules is smaller, converging to a median of just below 2 rules per switch and group, i.e. roughly 1 rule per subtree. In sum, these results show that DynSdm scales well with the size of the topology as well as the number of clients.

Figure 4.4c, in addition, shows the influence of using different traffic distribution mechanisms for the subtrees. Here, no significant difference between the variants can be observed.

4.2 DynSdm efficiency

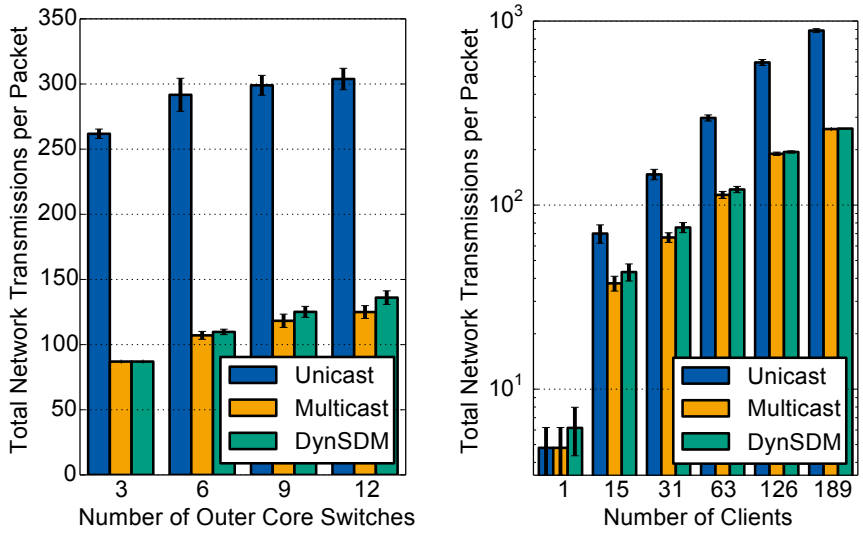
For the efficiency, it is important to show that introducing the multi-tree approach, has only limited impact on the overall amount of traffic. By that, the traffic should be well below the traffic of unicast



(a) ISP topology size. **(b)** Client population. **(c)** Traffic distribution algorithm..

Figure 4.4: Sensitivity analysis: Influence of scenario parameters.

deliveries to maintain the original incentive for the ISP to use an SDM-based service. Therefore, Figure 4.5 compares the total number of network transmissions (per video packet) performed to serve all active clients for different topology sizes (exemplary shown for number of OC switches) and client populations. Each time, an individual packet traverses a link on its path through the ISP 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 [KP99].

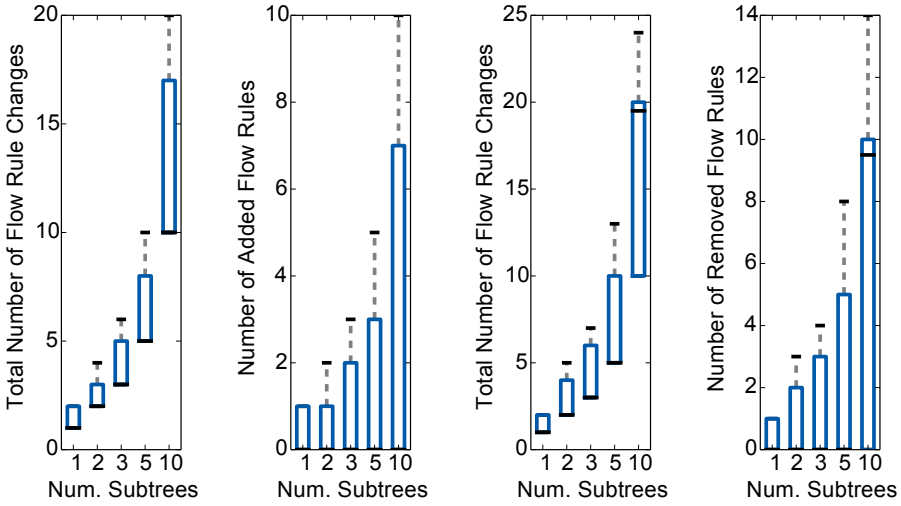


(a) ISP topology parameters. **(b)** Number of clients.

Figure 4.5: Traffic reduction for different exemplary scenario parameters.

4.3 Handling dynamics

As DynSdm aims at supporting dynamicity, it also includes mechanisms to handle group events, i.e. joining and leaving clients, and network events, i.e. link failures or links that an ISP decides to exclude from multicast deliveries. Handling such dynamicity, implies costs in terms of flow installations, modifications, or deletions. The results for supporting joins and leaves of peers are depicted in Figure 4.6. Here, it can be observed that, in general, the cost for handling these two types of events is very low due to the incremental approach taken by DynSdm that quickly attaches new clients. The mean overall costs for joins, i.e. the sum of changes, modifications, and deletions, linearly increase (r=0,9998) with the number of subtrees (1T: 1.46, 2T: 2.52, 3T: 4.06, 5T: 6.52, 10T: 12.86). The mean overall costs for leaves, are similarly observed to increase linearly (r=0,9967) with the number of subtrees (1T: 1.48, 2T: 2.82, 3T: 4.4, 5T: 7.1, 10T: 16.84).



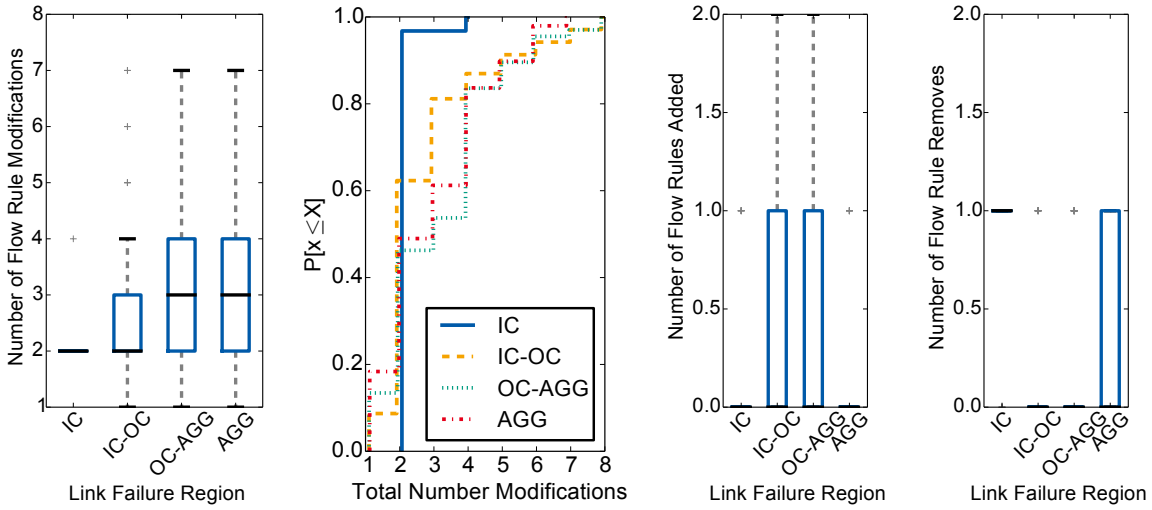
(a) Join: Added and modified rules. **(b)** Leave: Deleted and modified rules.

Figure 4.6: Client dynamics: Avg. number of rule changes per client join/leave.

To dynamically handle link failures or replan a subtree to exclude an individual link, specifically a number of flow rule modifications is required. This number is influenced by the topology part or the connection between two parts that the affected link belongs (cf. Figure 4.7). Here, individually, randomly chosen links of a tree were chosen and disabled using Mininet to force DynSdm to react on this change. The results show that the inner core failures result usually in only 2 rule modifications, required for redirecting the affected multicast tree. For the other parts, more rule might be required as shown, depending on the number of affected next hops or clients. Yet, the costs are considered being low and on an acceptable level with an upper limit of 8 modifications per removed link.

4.4 Discussion of service discovery

To evaluate the service discovery component, a proof-of-concept implemented was realized as part of DynSdm. For this, the Ryu controller framework was extended to support the discovery protocol introduced in Section 3. The component was used for the initial service discovery for all experiments presented above and showed to work as expected. A single flow rule had to be installed at each ingress switch that filters the discovery probes and forwards them to the service API machine, which consecutively sends the content provider a service offer and waits for the content provider to establish a control channel for any further DynSdm group operations. Due to the deterministic behavior of the service discovery mechanism, a quantitative analysis was not conducted.



(a) Modified flow rules. **(b)** Added flow rules. **(c)** Deleted flow rules.

Figure 4.7: Topology dynamics: Avg. number of rule modifications, adds, and deletions per link failure in different topology regions and their inter-connections (IC: Inner Core, OC: Outer Core, AGG: Aggregation Area).

5 Related Work

In the following, works on SDN-based multicasting are discussed, highlighting their relation to DynSdm.

In 1999, Keshav and Paul [KP99] proposed a first approach that postulates the separation of control and data plane of IP multicast, describing first ideas on core mechanisms also addressed in troday's SDN-based multicast approaches. This includes mechanisms to maintain efficient delivery structures, i.e. in terms of network traffic or transmission delays, under changing group populations as well as the possibility for an uninterrupted data delivery during updates to the active data-path tree topologies. Although the approach is not fully compatible with recent SDN works and in particular not an implementation using OpenFlow, the approach, conceptually, is an archetype for core parts of DynSdm and other state-of-the-art approaches, including [MSG+12, MWLH+14, ZSGH13, KSS12, YZXS12] that are further discussed in the following.

Marcondes et al. [MSG+12] propose CastFlow, an SDN-based multicast approach focusing on quickly reacting on multicast group events. For this, they focus on reducing the time to calculate a network path to attach new clients to a multicast tree. Using Prim's algorithm [CLR+01], a complete MST is calculated at group creation, covering all potential clients in the network. Using this pre-calculated MST, the controller dynamically pushes those flow rules to the switches that connect the active clients of the group. Thereby, the authors trade a reduction in processing time during group events for an increased amount of memory to manage the full MST. As the approach calculates paths for clients that might never join the group, it is expected that the initial group registration for realistic ISP scenarios with millions of customers and huge topologies would take a substantial amount of time. This downside and the expected high memory requirements for maintaining an ISP-wide MST per group, led to the decision to follow another approach in DynSdm. Avoiding unnecessary calculations, DynSdm uses the more efficient BFS-based attachement from [MWLH+14] to connect clients to the active topology, making groups easier to manage with little delays during joins in the studied scenarios.

Kotani et al. [KSS12] present another SDN-based multicast approach using multiple delivery trees that is related but also fundamentally differs from DynSdm. The focus of the authors is on the reduction of packet loss during multicast delivery on network link failures. Inspired by Wang and Li [WL08], they propose to use two redundant delivery trees and quickly switch from a broken tree to a backup tree. Thereby, the approach does not achieve a better traffic distribution as either one of the trees is used exclusively. Similar to the multiple trees in DynSdm, the backup tree is precalculated and installed at the switches together with the primary tree. This way, the autors can show that switching trees only requires a single flow rule change at the ingress switch, minimizing the reaction time and packet losses caused by link outages. DynSdm's multi-tree calculation approach was inspired by this and other previous works in that it calculated SPTs using a variant of Dijkstra's algorithm [CLR+01], increasing the weights of used edges when iteratively calculating further trees on the same topology. Besides, the approach assumes a traditional IP multicast scenario based on IGMP control behavior that is not envisioned for DynSdm and its application in an ISP scenario.

Zou et al. [ZSGH13] describe an SDN/OpenFlow-based multicast mechanism, focusing on secure group management, where the network provider has full control on the admission control for both multicast sources and clients. For this, the authors propose to rely on IGMP and forward all relevant protocol messages to the network controller that implements the group and multicast tree management. Based on MSTs calculated using Kruskal's algorithm [Kru56], the controller installs new flow rules to realize the multicast trees. The work differs from the DynSdm in that it only uses a single tree per multicast group and in its use of IGMP that was intentionally avoided in Sdm to allow content provider and ISP

to interact in a more practical manner, enable automatich discovery of the multicast service, and even allow accounting for the service by the ISP.

Yu et al. [YZXS12] propose OFM (OpenFlow Multicast) which, similar to most of the above described works, centralizes the multicast control at the network controller of an SDN/OpenFlow network. While they do not introduce particular new aspects, they consider a multi-controller scenario, where each controller is responsible for a part of the network and the controllers agree on a multicast tree to be installed at the routers/switches. Using this approach, they show that a reduced joining time can be achived, compared to IP mulitcast.

Finally, Lee et al. [MWLH+14] present an SDN-based robust multipath multicasting approach for video delivery. It is considered the closest related work to DYNSDM and inspired some mechanisms of it. While both approaches share some objectives, DYNSDM is a more general multicasting service that is not limited to video streaming. Both aim at improved load balancing of traffic across network links and the support for dynamic network conditions and client populations. Besides, Lee et al. focus on robust stream delivery, which in DYNSDM is only considered a positive side effect of distributing the traffic. They propose composing the video streams out of multiple substreams delivered using individual multicast trees. Clients are served by all trees and need to merge the substreams before video playback. In contrast to DYNSDM, it is assumed that delivered data is encoded using the loss-tolerant Multiple Description Coding (MDC) [Goy01] video representation. This way, packet losses on a tree can be compensated by presenting a lower-quality video representation to the user. In DYNSDM this approach is not applicable as it is limited to video data. In addition, the application-layer encoding contradicts the core objective of SDM to providing a transparent multicast service to the clients. Besides and as result of the application-layer encoding, the group source has to make the substream explicit to the network by sending them to separate trees. DYNSDM, in contrast, provides the multi-tree delivery transparent for the content provider by automatically splitting the data stream at network layer. This allows the ISP a more flexible planning of the multiple trees and load balancing across them. Other contributions by the authors are seen orthogonal to this work, such as the proposed tree planning algorithms, which also could be adopted in DYNSDM if desired by the ISP. Yet, in the studied scenarios, DYNSDM already achieves a good load balancing across network link using the simpler SPT-based tree calculation algorithms.

6 Conclusion

In this paper, DynSdm is presented, introducing a novel approach to make OTT multicast traffic delivered by Sdm more flexible and allows balancing the traffic distribution and link loads within the ISP network. For this, it proposes a mechanism to enable fully network-supported multi-tree delivery of multicast streams. Thereby, the mechanism is transparent to the content provider as well as 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. Besides, a novel service discovery approach is presented that solves the dynamic discovery of network services along a network path in multi-domain environments, which is an important practical aspect for the realization of network services such as DynSdm.

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 in the network. 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.

Bibliography

[AH02] Lada A Adamic and Bernardo A Huberman. Zipf's Law and the Internet. *RAM Glottometrics*, 3(1):143–150, 2002.

[App13] Apple Inc. *Bonjour Overview: Bonjour Concepts*, 2013.

[BH01] Cüneyt F Bazlamaçcı and Khalil S Hindi. Minimum-weight Spanning Tree Algorithms a Survey and Empirical Study. *Elsevier Computers & Operations Research*, 28(8):767–785, 2001.

[BRV+15] J. Blendin, J. Rückert, T. Volk, , and D. Hausheer. Adaptive Software Defined Multicast. In *IEEE Conference on Network Softwarization (NetSoft)*, 2015.

[CDKR02] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. One Ring to Rule Them All: Service Discovery and Binding in Structured Peer-to-Peer Overlay Networks. In *ACM SIGOPS European Workshop*, 2002.

[Cis14] Cisco. Cisco Visual Networking Index: Forecast and Methodology, 2013 – 2018. Technical report, 2014.

[CLR+01] T. Cormen, C. Leiserson, R. Rivest, C. Stein, et al. *Introduction to Algorithms*. MIT Press Cambridge, 2001.

[CZH+99] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz. An Architecture for a Secure Service Discovery Service. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1999.

[Dee89] S. E. Deering. Host Extensions for IP Multicasting. RFC 1112, Aug 1989.

[DGJ+06] M. Dueser, A. Gladisch, M. Jaeger, F.J. Westphal, and H.M. Foisel. Evaluation of Next Generation Network Architectures and Further Steps for a Clean Slate Networking-Approach. *ITG EuroView, Presentation*, 2006.

[Die13] Die Medienanstalten. Digitisation 2013 - Broadcasting and the Internet - Thesis, Antithesis, Synthesis? Technical report, 2013.

[Dij59] E.W. Dijkstra. A Note on two Problems in Connexion with Graphs. *Springer Numerische Mathematik*, 1(1):269–271, 1959.

[DLL+00] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment Issues for the IP Multicast Service and Architecture. *IEEE Network*, 14(1):78–88, 2000.

[Eri14] Ericsson ConsumerLab. TV and Media 2014 - Changing Consumer Needs are Creating a New Media Landscape. Technical report, 2014.

[GGJ77] M. R. Garey, R. L. Graham, and D. S. Johnson. The Complexity of Computing Steiner Minimal Trees. *SIAM Journal on Applied Mathematics*, 32(4):835–859, 1977.

[Goy01] Vivek K Goyal. Multiple Description Coding: Compression Meets the Network. *IEEE Signal Processing Magazine*, 18(5):74–93, 2001.

[Gul00] A. Gulbrandsen. RFC2782: A DNS RR for Specifying the Location of Services (DNS SRV). 2000.

[GWvB+01] S. D. Gribble, M. Welsh, R. von Behren, E. A. Brewer, D. Culler, N. Borisov, et al. The Ninja Architecture for Robust Internet-scale Systems and Services. *Elsevier Computer Networks*, 35(4):473–497, 2001.

[HASG07] Mojtaba Hosseini, Dewan Tanvir Ahmed, Shervin Shirmohammadi, and Nicolas D. Georganas. A Survey of Application-layer Multicast Protocols. *IEEE Communications Surveys and Tutorials*, 9(3):58–74, 2007.

[HC99] H. W. Holbrook and D. R. Cheriton. IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications. 29(4):65–78, 1999.

[HH11] G. Hasslinger and F. Hartleb. Content Delivery and Caching from a Network Provider's Perspective. *Computer Networks*, 55(18):3991–4006, 2011.

[Jai91] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[KK01] Ilango Kumaran and S Ilango Kumaran. *Jini Technology: An Overview*. Prentice Hall PTR, 2001.

[KP99] S. Keshav and S. Paul. Centralized Multicast. In *IEEE International Conference on Network Protocols (ICNP)*, 1999.

[Kru56] J. B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[KSS12] 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.

[KZS15] 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.

[LGL08] Y. Liu, Y. Guo, and C. Liang. A Survey on Peer-to-Peer Video Streaming Systems. *Peer-to-Peer Networking and Applications*, 1:18–28, 2008.

[LLW+11] 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.

[MFB99] 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)*, 7(5):641–652, 1999.

[MSG+12] 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.

[MWLH+14] 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.

[PFKL08] A. Presser, L. Farrell, D. Kemp, and W. Lupton. *UPnP Device Architecture 1.1*. UPnP Forum, 2008.

[PLH12] B. Pfaff, B. Lantz, and B. Heller. *OpenFlow Switch Specification, Version 1.3.0*. Open Networking Foundation, 2012.

[RBH13] 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.

[RBH15] J. Rückert, J. Blendin, and D. Hausheer. 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*, 23(2), 2015.

[San14] Sandvine. Fall 2014 Global Internet Phenomena Report, 2014.

[SKLJ14] Ramesh K Sitaraman, Mangesh Kasbekar, Woody Lichtenstein, and Manish Jain. Overlay Networks: An Akamai Perspective. In *Advanced Content Delivery, Streaming, and Cloud Services*. John Wiley & Sons, 2014.

[SSvdGS04] V. Sundramoorthy, M. D. Speelziek, G. J. van de Glind, and J. Scholten. Service Discovery with FRODO. In *IEEE International Conference on Network Protocols (ICNP)*, 2004.

[VP97] J. Veizades and C. E. Perkins. RFC2165: Service Location Protocol. 1997.

[WL08] D. Wang and G. Li. Efficient Distributed Bandwidth Management for MPLS Fast Reroute. *IEEE/ACM Transactions on Networking (TON)*, 16(2):486–495, 2008.

[WNSM14] R. White, J. Ng, D. Slice, and S. Moore. Enhanced Interior Gateway Routing Protocol. RFC Draft, April 2014.

[XCT03] Guoliang Xue, Li Chen, and Krishnaiyan Thulasiraman. Quality-of-Service and Quality-of-Protection Issues in Preplanned Recovery Schemes Using Redundant Trees. *IEEE Journal on Selected Areas in Communications*, 21(8):1332–1345, 2003.

[YZXS12] Y. Yu, Q. Zhen, L. Xin, and C. Shanzhi. OFM: A Novel Multicast Mechanism Based on OpenFlow. *Advances in Information Sciences and Service Sciences*, 4, 2012.

[ZH12] Xiangyang Zhang and Hossam Hassanein. A Survey of Peer-to-Peer Live Video Streaming Schemes - An Algorithmic Perspective. *Computer Networks*, 56(15), 2012.

[ZSGH13] 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.