OpenFlow Rule Placement In Carrier Backhaul Networks For Multicast Applications

by

Rafael George AMADO

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF A MASTER'S DEGREE
WITH THESIS IN IN TELECOMMUNICATION NETWORKS
M.A.Sc.

MONTREAL, APRIL 04, 2022

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Mohamed Cheriet, Thesis supervisor
Department of Automated Manufacturing Engineering, École de technologie supérieure

Mr. Kim Khoa Nguyen, Thesis Co-Supervisor
Department of Electrical Engineering, École de technologie supérieure

Mr. Chamseddine Talhi, Chair, Board of Examiners
Department of Software Engineering and Information Technology

Mr. Aris Leivadeas, External Examiner
Department of Software Engineering and Information Technology

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON MARCH 31, 2022

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

Placement des règles OpenFlow dans les réseaux de liaison de l'opérateur pour les applications de multidiffusion

Rafael George AMADO

RÉSUMÉ

Aujourd'hui, les consommateurs mobiles utilisent de plus en plus les applications de multidiffusion (par exemple, les jeux en ligne, la réalité virtuelle/augmentée (RV/RA), les médias sociaux) pour diffuser des vidéos via les réseaux des opérateurs. Grâce à sa flexibilité, la mise en réseau définie par logiciel (SDN) compatible OpenFlow permet l'application de politiques de haut niveau et prend en charge le découpage du réseau de bout en bout, ce qui est substantiel pour les exigences de ces applications. Un niveau d'abstraction plus élevé masque la complexité des périphériques réseaux et expose une interface simple aux opérateurs. Cependant, cette flexibilité implique la tâche complexe d'allouer les règles de bas niveau dans le réseau réel, ce qui nécessite de gérer des contraintes telles que la mémoire disponible limitée pour les commutateurs et la capacité des liaisons. En raison des modèles de commutateurs hérités, les travaux antérieurs se concentrent uniquement sur les entrées de la table de flux qui ne peuvent pas prendre en charge efficacement le trafic multidiffusion d'un périphérique (ou serveur) vers de nombreux périphériques sur le réseau.

Notre approche consiste à tirer parti de l'utilisation des Tables de Groupe, récemment introduites dans la spécification OpenFlow 1.1, pour prendre en charge les flux de multidiffusion et économiser la mémoire du commutateur. Le trafic vers plusieurs destinations peut être agrégé pour correspondre à une seule entrée de Table de Flux par commutateur, ce qui permet d'économiser des ressources de liaison importantes. Le contrôleur a la tâche difficile de calculer où installer efficacement les entrées (règles) des Tables de Flux et de Groupe dans le réseau.

Dans cette thèse, nous optimisons le placement des règles dans les réseaux Openflow à ressources limitées pour les flux monodiffusion et multidiffusion. Nous formulons notre modèle comme un problème de Programmation Non-linéaire en Nombres Entiers (PNLNE), en tenant compte des limites du réseau, telles que la capacité de mémoire des Tables de Flux et de Groupe et la bande passante de liaison disponible. L'objectif est de maximiser le trafic livré aux destinations sous contraintes de ressources. Pour résoudre ce modèle, nous utilisons un solveur (Gurobi) pour obtenir l'allocation optimale et proposons deux algorithmes pour calculer la règle d'allocation en temps polynomial : une approche gloutonne nommée OpenFlow Multicast Allocation Algorithm (OFMAA) et une version améliorée basée sur Steiner-tree (ST-OFMAA).

Nos résultats expérimentaux sur trois topologies différentes montrent que la Table de Groupe est un facteur clé pour réduire l'utilisation de la mémoire sur le réseau. Notre solution peut prendre en charge un nombre de flux plus élevé que les solutions proposées par les travaux antérieurs, qui ne tiennent pas compte des tables de groupe, en réduisant à la fois l'utilisation des liens jusqu'à 30% et le nombre d'entrées de flux nécessaires pour acheminer le trafic vers les destinations de 22%.

Mots-clés: réalité virtuelle, réalité augmentée, réseaux sociaux, réseaux définis par logiciel, optimisation des politiques de routage, placement de règles de multidiffusion, openflow, table de groupe

OpenFlow Rule Placement In Carrier Backhaul Networks For Multicast Applications

Rafael George AMADO

ABSTRACT

Today, mobile consumers increasingly use multicast applications (e.g., online gaming, Virtual/Augmented Reality (VR/AR), social media) to stream video through carrier networks. Thanks to its flexibility, OpenFlow-enabled Software-defined networking (SDN) allows the enforcement of high-level policies and support end-to-end network-slicing, which are substantial for these applications' requirements. A higher abstraction level hides the complexity of the network devices and exposes a simple interface to operators. However, this flexibility brings the complex task of allocating the low-level rules in the actual network, which requires handling constraints such as limited available memory on the switches and the capacity of the links. Due to legacy switch models, prior work focuses only on Flow Table entries that cannot efficiently support multicast traffic from one device (or server) to many devices across the network.

Our approach is to leverage the use of Group Tables, which is recently introduced in the OpenFlow 1.1 specification, to support multicast flows and save switch memory. Traffic to multiple destinations can be aggregated to match a single Flow Table entry per switch, saving significant link resources. The controller has the challenging task of calculating where to efficiently install Flow and Group Tables entries (rules) in the network.

In this thesis, we optimize rule placement in resource-constrained Openflow networks for both unicast and multicast flows. We formulate our model as an Integer Nonlinear Programming (INLP) problem, considering the network's limitations, such as the memory capacity of both Flow and Group Tables and the available link bandwidth. The objective is to maximize the traffic delivered to the destinations under resource constraints. To solve this model, we use a solver (Gurobi) to obtain the optimal allocation and propose two algorithms to calculate the rule allocation in polynomial time: a greedy approach named OpenFlow Multicast Allocation Algorithm (OFMAA) and an improved version based on Steiner-tree (ST-OFMAA).

Our experimental results on three different topologies show that Group Table is a key factor in reducing memory usage across the network. Our solution can support a higher number of flows than the solutions proposed by prior work, which do not consider Group Tables, by reducing both the link usage by up to 30% and the number of flow entries needed to deliver the traffic to destinations by 22%.

Keywords: virtual reality, augmented reality, social media, software-defined networks, routing policy optimization, multicast rule placement, openflow, group table

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

AR	Augmented Reality
BGP	Border Gateway Protocol
BS	Base Station
DDoS	Distributed Denial-of-Service
DPS	Depth-First Search
DNS	Domain Name System
EFRR	Effective Flow Rule Reduction
eMBMS	evolved Multicast Broadcast Multimedia Systems
IGMP	Internet Group Management Protocol
INLP	Integer Nonlinear Programming
IP	Internet Protocol
LLDP	Link Layer Discovery Protocol
M2U	Multicast-to-Unicast
MEC	Multi-access Edge Computing
MLU	Maximum Link Utilization
NAL	Network-wide Average Link Utilization
NBI	Northbound Interface
OFMAA	OpenFlow Multicast Allocation Algorithm
OSPF	Open Shortest Path First

OVS	Open vSwitch
PIM	Protocol Independent Multicast
QoS	Quality of Service
RPP	Rule Placement Problem
RP	Rendezvous Point
SBI	Southbound Interface
SDAN	Software-defined Access Networks
SDN	Software Defined Networks
ST	Steiner Tree
TCAM	Ternary Content Addressable Memory
U2M	Unicast-to-Multicast
VR	Virtual Reality
WLAN	Wireless Local Area Network

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

Mbps	Megabits per second

CHAPTER 1

INTRODUCTION

1.1 Context and motivation

Recently, we have been experiencing exponential growth of mobile video services. In its 2017 Visual Networking Index (VNI) Forecast [Cisco (2017)], Cisco has predicted that mobile video traffic alone will account for 78% of the global mobile data traffic by 2022, with Virtual/Augmented Reality (VR/AR) traffic alone increasing by 82% between 2017 and 2022. Moreover, the sudden shift in the global traffic pattern during the 2020 lockdown has resulted in even more acute growths in gaming, video-on-demand, and video-conferencing traffic [Feldmann *et al.* (2021)].

The rapid growth of such multicast applications imposes a severe challenge for the operators, especially for the upcoming 5G mobile technology [Cao, Jiang, Chen & Zhang (2015)], for which SDN is becoming the standard due to its programmability features. The most deployed standard of SDN is OpenFlow: it is mature, widely available, and highly flexible. OpenFlow 1.1+ introduced the Group Table capability, which provides the tool to route multicast traffic [OpenFlowSpec (2015)]. In this thesis, we study the combined placement of Group and Flow Table entries to efficiently route multicast traffic.

Next, we present the applicability of our work regarding the current applications' requirements and describe the challenges imposed by combining the two problems we tackle in this thesis.

1.1.1 Applicability

Traditional computer networks are built from a large number of devices, such as switches, routers, and middleboxes (e.g., Firewalls, DNS, DDoS). These devices are vertically integrated, which means that the control plane is integrated with the data plane. They contain proprietary switching hardware that implements the Internet standard protocols (e.g., IP, OSPF, BGP), they run

proprietary code as Operating Systems and additional software to the main protocols. Moreover, traditional routing protocols are distributed: routers exchange connectivity information, and each router builds its own routing table. These are complex systems that network operators need to manage to configure policies to respond to many network events and application requirements. They have to manually translate these high-level policies to device-specific low-level commands, which is error-prone and time-consuming [Kreutz *et al.* (2014)].

However, new challenges emerge with the recent applications and services, which are increasingly complex. The traditional vendor-locked closed architecture of network devices slows down the innovation the operators need to address these new challenges [Nunes, Mendonca, Nguyen, Obraczka & Turletti (2014)]. Each new feature might require waiting until the subsequent device life cycles or firmware releases from vendors, increasing the operational costs and slowing down the delivery of new features to customers.

Recent multicast applications, such as Augmented Reality (AR) crowdsource, leverage collaborative tasks like annotating the physical world. In such applications, users send and receive traffic from both an application server and all users in the same session [Ren *et al.* (2020)]. For example, in an AR-based game, all players in the same session must simultaneously receive the scene video stream to maintain fairness. In both examples, the video is a multicast connection, and any input from the players/participants is a unicast connection. Thus, this application requires both unicast and multicast flows flexibly routed in the network according to the mobility of the streamer (or the video server) and the online demand of the receivers (players).

Nonetheless, due to legacy switch models, most of the existing network architectures focus only on IP multicast [Islam, Muslim & Atwood (2017)]. These are complex by design due to the nature of the protocols involved, such as Internet Group Management Protocol (IGMP) and Protocol Independent Multicast (PIM) [Fernández, Contreras, Moyano & García (2021)]. The dynamic network service provisioning required for recent multicast applications is such that traditional IP-based solutions cannot deliver this traffic efficiently.

In order to tackle these limitations, large scale operators have been recently incorporating Software Defined Networking (SDN) in different areas of their network such as Wide-Area-Network (WAN) [Jain *et al.* (2013)], Multi-access Edge Computing (MEC) [Shah, Gregory, Li & Fontes (2020)], Mobile Networks [Nguyen, Do & Kim (2016a)], Sensor Networks [Kobo, Abu-Mahfouz & Hancke (2017)], Optical Networks [Bhaumik *et al.* (2014)] and 5G [Zaidi *et al.* (2018)]. SDN separates the data plane and the control plane, which dramatically simplifies the network management since the forwarding hardware is decoupled from the control decisions. Furthermore, commodity hardware can be used for the now simple forwarding devices that can be controlled via a standardized open interface (e.g., ForCES [Haleplidis *et al.* (2015)], OpenFlow [Nguyen, Saucez, Barakat & Turletti (2016b)]), which facilitates scalability and lowers operational and expansion costs.

SDN technology is essential to accommodate the demands of these applications [Islam, Muslim & Atwood (2018)]. SDN is designed to allow end-to-end support for network slicing since its control plane is separated from the data plane. Complex decisions are centrally taken in the controller (control plane) and enforced to the forwarding devices (data plane).

The most popular SDN platform in academia and industry is OpenFlow [Petale & Thangaraj (2020)], and many vendors have implemented OpenFlow in recent years (e.g., HP, NEC, Pronto, Extreme, Cisco, Brocade, Juniper, Huawei) [Costa *et al.* (2021)]. It is often the choice as the southbound interface of SDN [Kreutz *et al.* (2014)], and OpenFlow-enabled switches use match-action tables to forward traffic in a flow-based manner setting them apart from traditional destination-based (IP routing). The Flow Tables match specific fields in the packet header (e.g., source IP address, destination IP address, TCP port, UDP port) and apply the corresponding action to the packet. Among the many possible actions, the packet can be sent to an output port, to another Flow Table for further processing, or to a Group Table, which can replicate the packets to different ports. The controller must define these rules and push them to the switches via OpenFlow SBI.

Physical SDN switches usually implement OpenFlow Flow Tables with Ternary Content Addressable Memory (TCAM) since they can flexibly match different patterns by using wildcard rules with high performance. However, TCAM is expensive and power-hungry [Katta, Alipourfard, Rexford & Walker (2014)] and, though recent high-performance Application-Specific Integrated Circuits (ASIC) are increasing their capacity [Lanner (2021)], the applications' requirements also increase in complexity, demanding more resources. Furthermore, Data Centers (DC) often deploy software switches (e.g., Open vSwitch [vSwitch (2015)]), which have a large Flow Table capacity but are limited in forwarding and lookup speed. The solution is to store flow tables in CPU caches to accelerate the operations, which raises the capacity issue again since these caches are small.

The design of OpenFlow allows the enforcement of high-level policies such as the ones required by these high-demanding applications in real networks with limited resource capacity, as described before. Nonetheless, though OpenFlow provides the tools, it is difficult to allocate these policies efficiently. We describe, next, the challenges involved in enforcing high-level policies in the context of multicast applications and SDN/OpenFlow.

1.1.2 Challenges

A higher abstraction level hides the complexity of the network devices and exposes a simple interface to the operators. However, this flexibility brings the complex task of allocating the low-level rules by solving the *Rule Placement Problem* (RPP) [Nguyen *et al.* (2016b)], whose solution defines the rules and their placement in the network. The challenge is to allocate the flows by choosing where to place the low-level rules while considering the constraints of memory and link capacity. When unicast flows need to be installed, the challenge is to solve the RPP for Flow Table entries under network constraints, which is NP-hard [Nguyen, Saucez, Barakat & Turletti (2015)]. Furthermore, since each user can request more than one type of traffic, the flows might have different requirements, augmenting the granularity required and, thus, increasing the number of flow entries to be placed. Ideally, the forwarding rules that install a flow should be placed in the switches with the shortest path to the destination. However, due

to memory limitations, installing all the necessary flow entries into the shortest paths might not be possible, which increases the overall link utilization since longer paths could be chosen.

To route multicast applications efficiently, the SDN controller must find a multicast tree capable of delivering the traffic to all the receivers in the same session considering the applications' requirements regarding Quality of Service (QoS) under constraints such as memory and link bandwidth capacity [Molnár, Bellabas & Lahoud (2012)]. The QoS multicast routing problem is a nonlinear combinatorial optimization problem, and it is proved to be NP-Hard [Wang & Crowcroft (1996)], so the usual approach is trying to solve it by algorithmic approximation [Zhang, Shen & Yu (2019)].

The *Endpoint Policy* defines the ingress (source) and egress (destinations) links where a flow must enter and leave the network. The specific path followed by a packet is defined by the *Routing Policy* [Kang, Liu, Rexford & Walker (2013)]. We consider that a multicast flow respects the Endpoint Policy since the source and destinations are well defined. However, the Routing Policy can be relaxed, and the flows do not need to follow a strict path, so the network resources can be used more efficiently [Nguyen *et al.* (2015)]. Leveraging this flexibility, we formulate our model as an Integer Nonlinear Programming (INLP) problem that calculates the QoS multicast tree for a set of flows. Our approach is to simultaneously solve the Rule Placement Problem for both Flow (unicast) and Group (multicast) Table entries while considering memory and link bandwidth constraints. We merge two challenging problems, design a model and optimize the combined placement of both types of rules. For a given flow and topology, some nodes receive unicast rules only (Flow Table entries), while others receive both unicast and multicast rules. The solution to this new problem is the optimal placement of Flow and Group Table entries that jointly describe the multicast tree and define where to place the low-level rules in the network.

The following section details the scenario we propose for delivering multicast traffic in OpenFlow-enabled SDN networks. We describe the use of Flow and Group Tables and how their combination can efficiently multicast streaming traffic.

1.2 Illustrative Example

To motivate our research, we consider an example illustrated in Figure 1.1, where an AR application is used by several users simultaneously. In this scenario, all users in the same session need to receive the same video stream (which is multicast traffic), and they can interact with each other and the background scene. Traditional distributed protocols (e.g., IGMP) are very rigid and cannot handle the dynamic nature of this application. Instead, a centralized SDN application defines the multicast groups and can move them around flexibly according to the demands of network resources by installing rules in the appropriate switches. Flow Table entries are used to match flows and apply actions to the matched traffic. Furthermore, we can use Group Table entries to distribute multicast traffic efficiently.

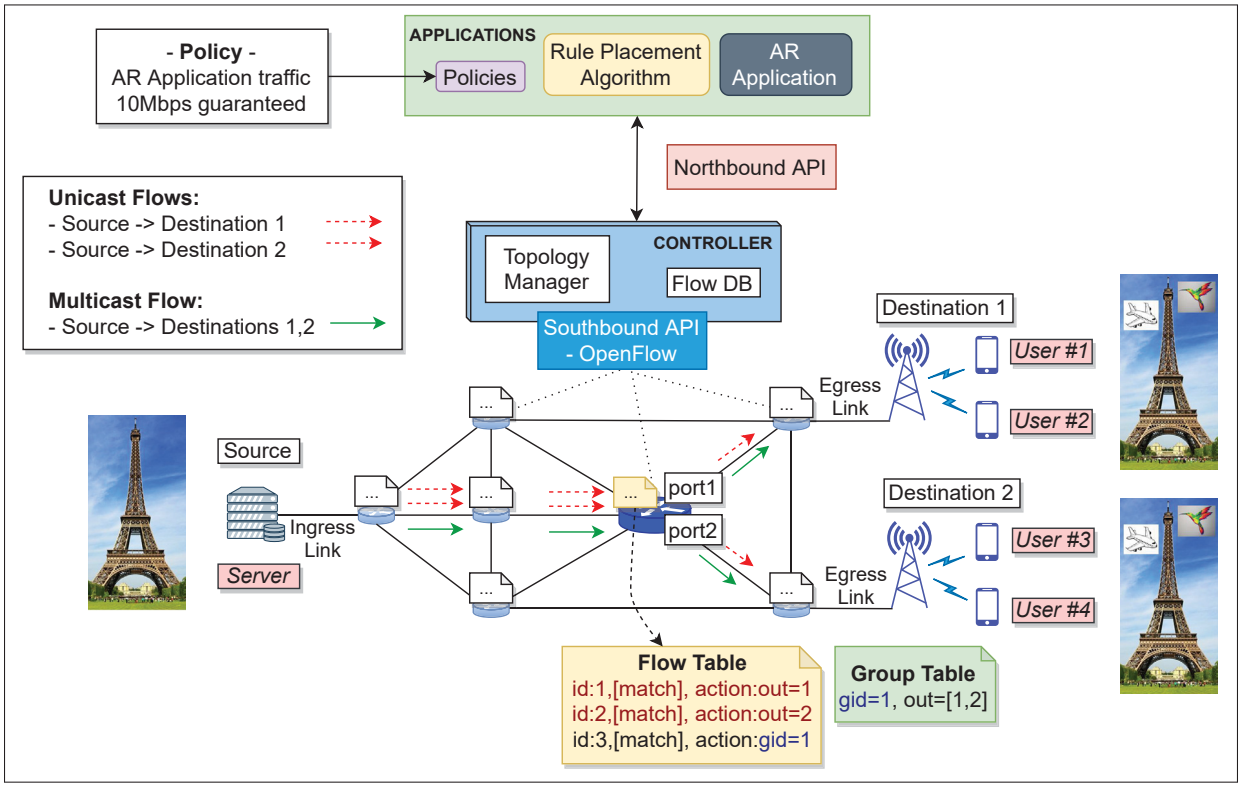

Figure 1.1 Interactive AR application streaming video to multiple users

The controller enforces an endpoint policy to guarantee the end-to-end bandwidth of the collaborative AR application. When the users want to use the application, the controller installs flow entries into the switches to deliver the video-stream traffic from the server across the

network. Each user can interact within the same scene since several users can participate in the same session and insert overlay information over the video in real-time. All users in the session receive the original video from the server (illustrated by the Eiffel Tower on the left-hand side) and the interactions generated by each user. The user interactions are replicated by the server to the other participants, as illustrated by the inserted elements in the right-hand side of Figure 1.1. The traffic source is a server connected to one aggregation switch in the network via an ingress link. The egress links connect the aggregation switches to the destination Base Station (BS), to which the users are attached. The controller receives a request from the application and installs the flow entries to the switches within the network, i.e., from one ingress link to one or more egress links. If only Flow Table entries are supported by the application, each pair (source, destination) requires one entry in the Flow Table of every switch along the path, as shown by the red dashed arrows. Fortunately, Openflow Group Table allows the controller to make a more scalable decision by replicating traffic towards each destination. In other words, a single flow entry is added to the Group Table of each switch, as shown by the solid green arrow. Such a mechanism is very efficient in video multicasting scenarios with many receivers.

However, the forwarding rules that install a flow should be placed, ideally, in the switches along the shortest path to the destination. The challenge is that, in the actual network, switch memory is a limited resource, and it might not be possible to install all the necessary flow entries into the shortest paths. Longer paths mean a higher use of the links whose capacity is also limited. Furthermore, as the number of users/destinations grows, the complexity of allocating all the rules under the network constraints increases.

In summary, on the one hand, OpenFlow provides flexibility to enforce high-level policies. On the other hand, the efficient allocation of flow rules remains difficult. At the same time, the Group Table can be a very efficient tool to handle multicast traffic due to its aggregation nature.

1.3 Problem statement

To route both multicast and unicast traffic flows in an OpenFlow network, we need to tackle the QoS Multicast Routing and OpenFlow Rule Placement problems simultaneously by finding the optimal placement of OpenFlow rules considering the flows' requirements and the network constraints.

1.4 Research question

The problem statement above suggests an interesting research question:

RQ: Can we find an OpenFlow rule placement solution that provides the QoS Multicast Trees and Unicast Paths to maximize the allocation of traffic flows?

1.5 Objectives

The main objective of this work is to maximize both OpenFlow Multicast and Unicast traffic flows to a set of destinations under switch table memory size and network bandwidth constraints. To fulfill this objective, we consider two sub-objectives:

- **SO1.**: Formulate an optimization model that combines both Flow Table and Group Table entries for both multicast and unicast services delivered by OpenFlow.
  - We need to build a system with a centralized Control Plane capable of controlling OpenFlow-enabled SDN switches with Group Table capability.
  - This control plane must be able to communicate with applications that contain high-level policies and other network-related services (i.e., topology manager).
  - We define some metrics and a scenario to compare the performance of our model with the prior work, based only on Flow Table entries.

- **SO2.** Propose an efficient algorithm to solve the optimization problem.
  - To maintain certain quality in the user experience, newly arrived flows need to be allocated in tractable time.
  - To achieve this, the problem must be solved in polynomial time so that the solution can be used in large Carrier-grade networks.

1.6 Plan

This thesis includes an introduction, three chapters, and a conclusion with research direction for future work. The thesis is organized as follows:

Chapter 1 is the introduction. We first present the context and motivation of this study, then the problem statement, the research questions, and the objectives are presented.

Chapter 2 discusses the related work. We present a review of the prior research about multicast traffic and rule placement in OpenFlow networks and Augmented Reality applications.

Chapter 3 presents the methodology. According to the objectives previously stated, we present the system description and the model designed under the multicast scenario. Then, the proposed rule allocation algorithm is described.

Chapter 4 presents the experimental setup, simulation scenarios, numerical results and then discusses the simulation results.

Finally, we present our Conclusion and future work.

CHAPTER 2

LITERATURE REVIEW

This chapter summarizes some previous work on Multicasting for Carrier Networks and the most recent approaches for tackling the Rule Placement Problem in Software-Defined Networks.

2.1 Multicasting

Multicasting is a requirement for many recent applications, from gaming to interactive video, so this section aims to review recent work on different multicast types and solutions.

The authors in [Chen, Hu, Chung & Chou (2019)] investigate multicast AR streams over multiple wireless access points to various users. They formulate a model to maximize the number of satisfied users and propose a multi-view allocation (MVA) algorithm to distribute users across the available APs. The goal is to overcome the scarcity of bandwidth in current wireless networks to simultaneously serve multi-view VR/AR 3D video for many users. This work is centered on multicast views on WLANs and steering locally generated traffic, thus not applicable for geographically distant users.

Focusing on tiled panoramic video content, [Majidi & Zahran (2020)] proposes a solution to improve received video quality and reduce battery consumption for small groups of eMBMS (evolved Multicast Broadcast Multimedia Systems) [Lecompte & Gabin (2012)] users. The combination of unicast and multicast is used to model the content delivery to multiple users under the same Base Station (BS). Though the solution is developed for 4G and 5G, all the users are tied to the same BS.

[Humernbrum, Hagedorn & Gorlatch (2016)] present a model to determine the multicast tree route, which suppresses the number of required multicast entries for a multicast request in the SDN. The authors develop algorithms for calculating multicast trees that allow for maximally reusing unicast flow table entries for multicast. Since only Flow Table entries are considered, the reuse of entries is limited to the tree trunk. Furthermore, they apply Early-Branching Shortest

Path Tree, which may increase the overall bandwidth utilization in the network. At the branching point, one entry is needed per receiver due to the lack of the Group Table.

Centered on IP multicast, the authors of [Kim, Yun, Kim & Kim (2017)] propose an SDN multicast mechanism based on group shared tree with flexible group management. Based on VLAN tags and heuristics, the Rendezvous Point (RP) selection algorithm gives the solution, which finds the best placement of the RP switch for each multicast group. Similarly to [Humernbrum *et al.* (2016)], each receiver needs a dedicated Flow Table entry at the branching point, which leads to an increase in memory utilization.

As a hybrid approach, [Luo, Xing & Fan (2021)] proposes SDM (Software-Defined IP Multicast) provide "native" IP multicast support to distributed cloud applications. The idea is to use Open vSwitches (OVS) to perform Multicast-to-Unicast (M2U) then Unicast-to-Multicast (U2M) translations at the virtualized network edge. The main focus is on the throughput and maximum latency of the multicast tasks. Though the authors' approach is based on OVS, they do not consider the memory limitations, and as the number of flows and receivers grow, this is one of the main constraints to consider.

OpenFlow-based multicast has been implemented in state-of-the-art research using Flow Table entries [Kotachi, Sato, Shinkuma & Oki (2020)], which is expensive since each destination on the multicast group needs one Flow Table entry. The authors propose a routing model that calculates multicast tree routes and minimizes the number of flow entries for multicast requests in SDN. The main idea is to share unicast entries between the multicast trees. However, its scalability is limited since the model applies to IP multicast, where the sender must join a multicast group defined by the controller, similar to the traditional IGMP approach.

The Group Table has been implemented in OpenFlow 1.1, but only recently available hardware has been released with support for newer versions of OpenFlow [Costa *et al.* (2017)]. Though the Group Table is not a new introduction to the OpenFlow specification, it was used only as an auxiliary tool within the 5G core [Zhang, Yang, Zhao, Zhang & Ge (2017)] for studies on spectrum resource optimization.

Since prior work has not yet taken advantage of newly released OpenFlow hardware with Group Table support, they cannot efficiently support AR applications with multicast traffic from one device (or server) to many devices across the network.

To efficiently route multicast traffic using OpenFlow, we need to study the *Rule Allocation Problem*, so we can efficiently install the rules. The following section describes prior work on that matter.

2.2 Rule allocation problem

To enforce high-level policies such as the ones required by multicast applications, flows need to be allocated to ensure the necessary resources are reserved into the switches in the form of rules/entries. In order to efficiently place these rules, it is necessary to solve the *Rule Placement Problem*.

The problem of flow allocation has been addressed by [Nguyen *et al.* (2015)], which relaxes the routing policy and uses a default path towards the controller to allocate the lower priority traffic. In this manner, default rules are used to reach the destination reducing memory requirements. The authors model the problem for unicast traffic and propose a greedy algorithm called OFFICER that deflects the packets following the default path towards the egress link.

The authors of [Cheng, Hwang, Wu, Lin & Syu (2020)] tackle the congestion problem in Flow Tables by reducing the number of flow entries for an OpenFlow switch proposing what they call effective-flow-rule-reduction (EFRR) algorithm. It is designed for IP-based flow rules, and the compression is done considering the non-prefix string of the IP address and action field. They focus on minimizing the memory of the Flow Table for unicast flows only, so the solution is not suitable for multicast traffic.

[Huang, Guo, Li, Ye & Stojmenovic (2015)] proposes a rule multiplexing scheme for rule placement for a set of unicast sessions with QoS requirements. Often the best path cannot accommodate all traffic belonging to a session, so multipath routing is necessary. The model

relies on the idea that only one copy of rules to be deployed onto the common nodes of multipath will be enough to route traffic belonging to different paths. Though the model can handle multipath, it compresses based on "common destination", making it appropriate for unicast traffic only.

Aimed at compressing forwarding rules, [Rifai *et al.* (2017)] presents a two-phase solution: the compression phase and the routing phase. The network is modeled as a directed graph, and each stage has its own heuristic. The proposed solution can perform table compression with aggregation by the IP-source, IP-destination, and also by the default rule. Since the solution is applicable for IP-based traffic, it would require a high level of signaling for accommodating multicast traffic and yet be limited to IP-multicast traffic.

[Giroire, Moulierac & Phan (2014)] proposes an energy-aware routing model in an SDN-enabled network. They leverage the existing rule space available at the hardware switches for rule placement, aiming to minimize the link utilization reducing energy consumption. This approach is inefficient in a multicasting scenario since it is based on flow entries only, which requires one entry per destination, increasing the overall link and memory utilization.

The authors in [Bera, Misra & Jamalipour (2019)] propose an adaptive flow-rule placement scheme in SDN aiming to provide per-flow statistics to SDN. Their approach is two-fold: (1) formulate a max-flow-min-cost problem from the aspects of SDN while considering QoS requirements of the flows in the network, and (2) formulate an ILP to find an optimal number of switches to place exact-match rules. Since an exact-match flow entry is tied to only one flow to get per-flow statistics, the goal is to find the optimal number of switches receiving such rules to maximize the network visibility while avoiding rule overflow.

Focusing on rule distribution, the authors of [Sheu, Lin & Chang (2018)] propose decomposing the large Ternary Content-Addressable Memory (TCAM) table located at the network ingress switch into smaller sub-tables. These can be distributed across the network so that each switch will have a smaller memory requirement. They propose a Sub-table Allocation (SA) algorithm to maximize the number of sub-tables to lower the average load of each switch TCAM. The

solution proposed was tested for Access Rules only: Firewall (FW), Access Control Lists (ACL), and IP Chain (IPC).

The rule placement for traffic engineering was considered by [Guo, Luo, Wang, Yin & Wu (2021)]. The authors propose to impose constraints on the number of routing paths (i.e., path cardinality constraints) when optimizing flow routing to reduce the number of flow entries in the Ternary Content Addressable Memory (TCAM) in a hybrid SDN. They optimize routing with path cardinality constraints to minimize Maximum Link Utilization (MLU). The approach is to formulate the optimization problem and define an incremental deployment method for obtaining a hybrid SDN and an H-permissible Paths Routing Scheme (HPRS).

[Saha, Misra & Bera (2021)] proposes a QoS-aware flow-rule aggregation mechanism for Software-defined IoT based on path-selection and combination of match-fields. The authors' strategy consists of finding a suitable combination of match-fields to reduce the number of rules and minimize the impact of the QoS of newly arriving IoT flows.

2.3 Gap Analysis

Most of the work on Multicast and SDN focuses on the Flow Table entries, which means that for each receiver, we need one specific entry in the branching node. More recent works consider both Flow and Group Table entries, but they assume the tables have infinite size, which is unrealistic. In the Rule Placement review, the studies can target many different objectives. However, the recent studies consider Flow Table entries only, not being efficient for multicast routing. The summary of our literature review is presented in Table 2.1.

Our proposed Rule Placement model leverages the combination of Flow and Group Table entries to deliver Multicast services across the network efficiently.

Table 2.1 Related Work

Topic	Author	Methodology	Remarks
Multicasting with OpenFlow	[Humernbrum (2016)] [Kim (2017)]	- IP-Multicast - Heuristic-based	- Flow Table entries only - One Flow Table entry per receiver
	[Kotachi (2020)] [Luo (2021)]	- IP-Multicast - Tackles both unicast and multicast rules - Solved by algorithm	- The models do not take into account memory limitations.
Rule Placement	[Giroire (2014)] [Huang (2015)] [Nguyen (2015)] [Rifai (2017)] [Chang (2018)] [Bera (2019)] [Cheng (2020)] [Guo (2021)]	Optimize Rule Placement jointly with different objectives: - Max. traffic satisfaction - Min. number of allocated rules - Max. network visibility - Min. energy consumption - Integer Linear Programming Models (ILP) - Link capacity/switch memory constraints - Solved by greedy algorithm	- Flow Table entries only - Not efficient for multicast

CHAPTER 3

METHODOLOGY

3.1 Group Table

The specification of an OpenFlow Logical Switch [OpenFlowSpec (2015)] states that a switch may have one or more Flow Tables and one Group Table (Figure 3.1). The former matches the packet headers, and the latter can be used to forward multicast traffic (Group Type 'ALL'). Group Tables receive the packets from the Flow Tables when the flow that matches the packets has an action *Group*, and the Group ID indicates the group pointed by the flow entry. Depending on the implementation, the size of the Flow and Group Tables can be either fixed (Noviflow (2019a)) or defined during the compilation process when the switch has a programmable data plane (Noviflow (2019b)). Since each flow entry can only send traffic to one output port, we can use the Group Table type 'ALL' to forward this traffic if the same matched flow needs to be sent to multiple ports. They can be a powerful tool to reduce the number of table entries since the flows with various destinations can be installed using a single entry in the Group Table instead of one entry per output port.

The optimized placement of flow entries in Group Tables of switches in a network is challenging, as illustrated in an example in Figure 3.2. This example shows a network composed of 7 OpenFlow switches and a list of 2 flows with a total bandwidth of 6 to be routed. Each flow has requested bandwidth, ingress, and egress points. For the sake of simplicity, we denote the ingress and egress links as the switches to which they are connected. In this example, every link has a maximum capacity of 5, each Flow Table stores at most five flow rules, and the Group Tables, two entries at most. Each thick line denotes a flow entry combined with a group entry. The numbers next to the lines refer to the flow allocated to that rule entry. On the left-hand side, multicast flow #1 from A toward F and G is assigned in the Group Table of switch A, replicating the traffic to links A-C and A-B. Since the destinations of flow #1 are F&G, flow entries are installed into intermediate switches B, C, and E, as denoted by the number 1 in their tables.

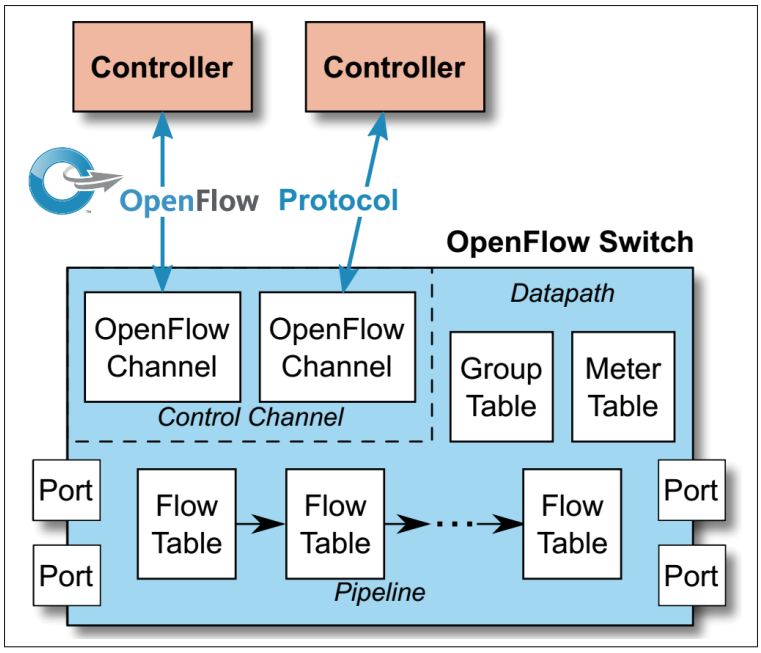

Figure 3.1 Main components of an OpenFlow switch, from OpenFlowSpec (2015)

Totally, the option of placing a group table entry of flow #1 at switch A results in 4 entries and 15 bandwidth in the network. On the right-hand side, another option of placing the entries of the same flow #1 in the switches is shown. At first, flow #1 is sent through link A-C with a simple entry in the flow table of switch A. Then, a Group Table entry is inserted into switch C, which replicates the traffic to links C-F and C-G. The option of placing a group table entry of flow #1 at switch C results in 2 entries and 9 bandwidth in the network. A similar result can be achieved for flow #2 by choosing the Group Table placement on switches D or E.

This example shows that the optimal placement of the Group Table entries can maximize the number of flows in the network and, at the same time, save resources.

After explaining the OpenFlow switch rule placement, the following section presents the Control Plane responsible for managing the switches and allocating the Table entries.

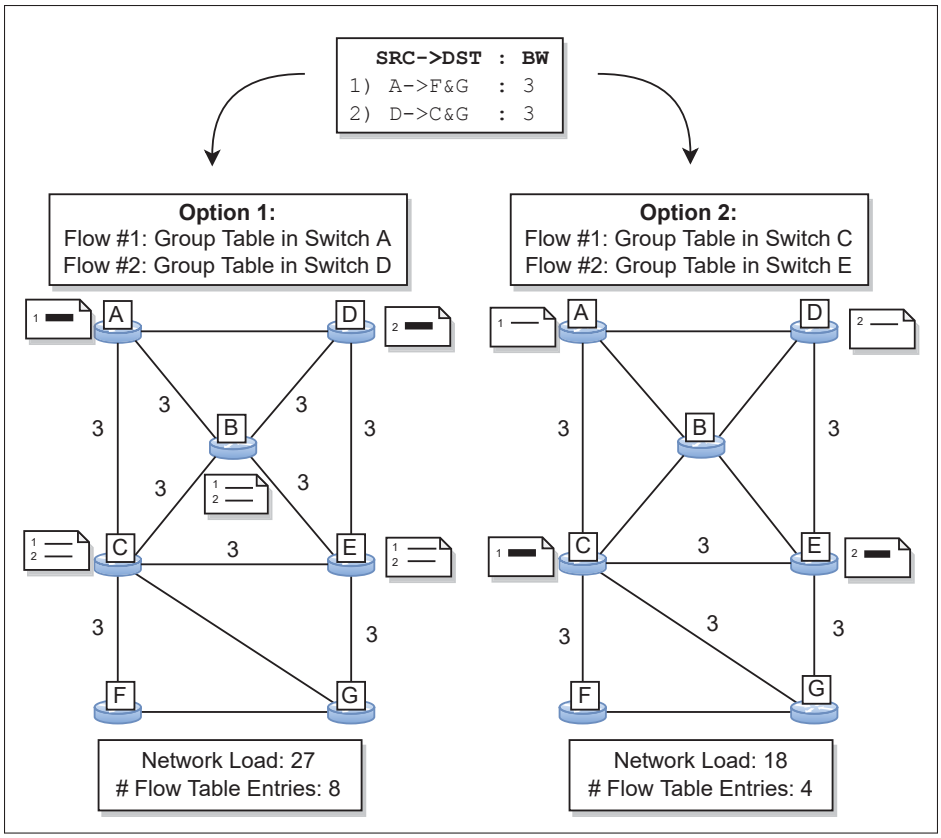

Figure 3.2 Group Table placement can increase the efficiency of the network

3.2 Control Plane

We illustrate our system in Figure 3.3, where a centralized controller calculates where to place entries into Flow and Group Tables of all switches in a backhaul network. The controller exposes a Northbound Interface (NBI) where high-level policies and applications can be implemented, and via the Southbound Interface (SBI), manages the switches through OpenFlow messages. The Topology Manager is an application that uses LLDP packets to discover the topology. With this information, the controller can create an abstracted topology graph where the policy enforcement decisions will be made.

We have presented our rule allocation strategy and the control plane necessary to enable its implementation. The following section describes the Integer Nonlinear Programming (INLP) model we propose to allocate Flow and Group Table entries optimally.

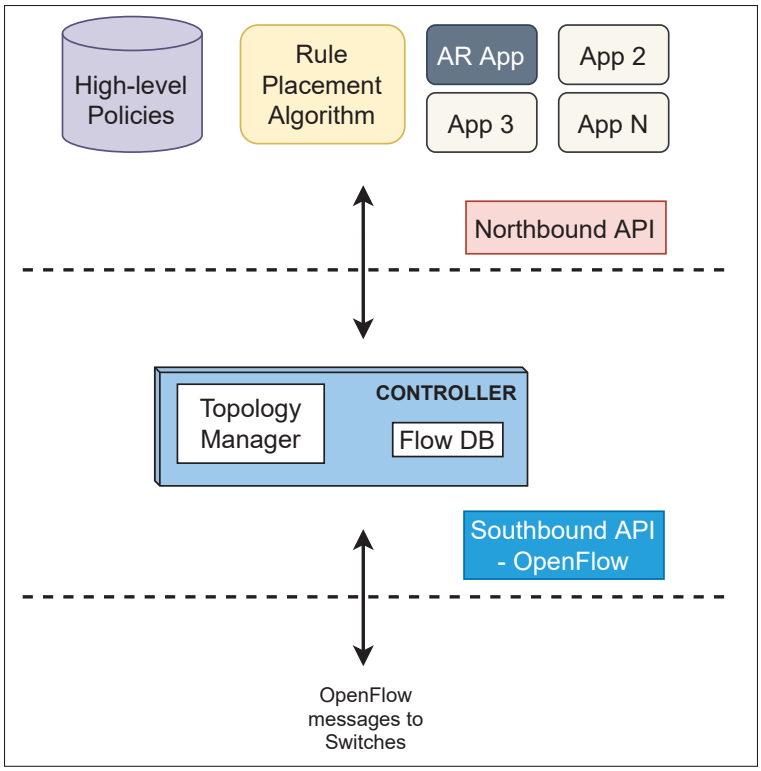

Figure 3.3 Control Plane

3.3 Problem formulation

We model our network as a undirected graph $V(S, L)$, where S is the set of switches and L is the set of links, with capacity B_l, interconnecting them. A set of flows F needs to be mapped into flow rules in the switches in the network. Each flow $f \in F$ specifies a packet rate p_f, a source $l_f \in I$ which is the ingress link connecting the AR application server to the Aggregation switch where the packet enters the backhaul network. The same flow $f \in F$ has one or more destinations $d \in E(f)$ $(E(f) \subseteq E)$, which are the egress links where the packets leave the backhaul network to one or more destination Base Stations (BS). $L^+ = L \cup I \cup E$ is the set of all directed links. Each switch $s \in S$ has a Flow Table and a Group Table with limited capacity, respectively, C_s and G_s. The Flow Table performs the matching of packets for each flow $f \in F$ and forwards them to a specific port towards the destination $d \in E(f), |E(f)| = 1$. When a flow has more than one destination (multicast), the action in the Flow Table is to steer the packets to the Group Table, which clones the packets and send them to the list of ports towards the egress

links $d \in E(f), |E(f)| > 1$. In this case, two entries are necessary: one in the Flow Table, where the matching happens, and the other in the Group Table, where the action is performed (clone packets and send them out). Table 3.1 summarizes the notations used in this paper.

We present the placement solution by two matrices of binary variables: $A = (a^d_{f,l})$, where $a^d_{f,l} = 1$ when a flow f, with destination $d \in E(f)$, passes through the directional link $l = (u, v)$; $u, v \in S^+$ from node u to node v; and $G = (g_{f,s})$, where $g_{f,s} = 1$ when a flow f allocated to switch s is also put in its Group Table.

The allocation matrix is a source of information for an operator as it provides at the same time the forwarding table, switch memory occupation, and link usage for a given high-level objective and endpoint policy.

Constraints (3.1) and (3.2) verify that $a^d_{f,l}$ and $g_{f,s}$ are binary variables.

$$\forall f \in F, \forall l \in L^+, \forall d \in E(f) : a^d_{f,l} \in \{0, 1\} \tag{3.1}$$

$$\forall f \in F, \forall s \in S : g_{f,s} \in \{0, 1\} \tag{3.2}$$

Bandwidth constraint (3.3) ensures that the sum of all flows (for all destinations) allocated to link l does not exceed its capacity B_l. Two terms are used here because when a flow has more than one destination, but the Group Table is not allocated, all $d \in E(f)$ sharing link l will consume one unit of packet rate p_f.

$$\forall s \in S, \forall l \in L^+ :$$
$$\sum_{f \in F} \sum_{d \in E(f)} \frac{p_f a^d_{f,l}}{|E(f)|}(1 - g_{f,s}) + \sum_{f \in F} \sum_{d \in E(f)} p_f a^d_{f,l} g_{f,s} \leq B_l \tag{3.3}$$

We also need to make sure not to exceed the memory limitation of each switch s. Constraints (3.4) indicates that all destinations $d \in E(f)$ will match the same flow table entry, while (3.5) is the Group Table capacity constraint.

$$\forall s \in S : \sum_{d \in E(f)} \sum_{v \in N^{\leftarrow}(s)} \sum_{f \in F} \frac{a^{d}_{f,(s,v)}}{|E(f)|} \leq C_s \tag{3.4}$$

$$\forall s \in S : \sum_{f \in F} g_{f,s} \leq G_s \tag{3.5}$$

Constraint (3.6) indicates that packets belonging to flow f will only traverse their ingress link l_f.

$$\forall f \in F, \forall d \in E(f) : a^{d}_{f,l} = \begin{cases} 0 & \text{if } l \in I \setminus \{l_f\} \\ 1 & \text{if } l = l_f \end{cases} \tag{3.6}$$

Flow conservation constraints assure that the incoming traffic of switch s leaves through the egress links. While (3.7) is the usual flow conservation constraint used for the unicast flows, (3.8) has a per-destination approach which is necessary since, in a multicast transmission, one incoming packet at an intermediate node might produce one or more outgoing packets, which violates the flow conservation principle.

$$\begin{aligned} &\forall f \in F, \forall s \in S : \\ &\sum_{d \in E(f)} \sum_{v \in N^{\rightarrow}(s)} (1 - g_{f,s}) p_f a^{d}_{f,(v,s)} - \\ &\sum_{d \in E(f)} \sum_{v \in N^{\leftarrow}(s)} (1 - g_{f,s}) p_f a^{d}_{f,(s,v)} = 0 \end{aligned} \tag{3.7}$$

$$\forall f \in F, \forall s \in S, \forall d \in E(f), \forall v \in N^{\leftarrow}(s) : \\ \sum_{u \in N^{\rightarrow}(s)} g_{f,s} p_f a^d_{f,(u,s)} - g_{f,s} p_f a^d_{f,(s,v)} = 0 \tag{3.8}$$

As described in Equation 3.9, the objective function aims to maximize the volume of traffic satisfying the Endpoint Policy, which it is NP-hard [Nguyen *et al.* (2015)].

$$\text{Maximize} \quad \sum_{f \in F} \sum_{l,d \in E(f)} a^d_{f,l} p_f \tag{3.9}$$

where p_f is the packet rate of flow $f \in F$.

3.4 OpenFlow Multicast Allocation Algorithm (OFMAA)

As discussed above, the optimization problem (Equation 3.9) is NP-hard, so the optimal solution cannot be obtained in real-time when the number of flows and nodes in the network is extensive. Therefore, we develop an algorithm named OFMAA (Algorithm 3.1), which aims to install the flows with a higher packet rate p_f, so we can approximate the optimal goal of maximizing the traffic that satisfies the Endpoint Policy.

Line `5` of the algorithm sorts the flows according to their packet rate, starting with the flows with the highest requested bandwidth. The algorithm follows a greedy approach in the sense that it tries to allocate larger flows first and fills the remaining resources with smaller flows. Based on the Depth-First Search (DPS) algorithm, the function `paths`$(l_f, E(f))$ finds all the single paths, with enough available capacity for p_f, from the ingress link l_f to each egress link $d \in E(f)$ of flow f and stores them in $P(f)$. Line `8` calls the function `groupTableNode`, which receives set of paths $P(f)$ and checks if there is one node s that intercepts the paths to all

Table 3.1 Notation

Notation	Description
F	Set of flows.
p_f	Packet rate of flow $f \in F$.
S	Set of Openflow switches composing the network.
C_s	Flow Table capacity of switch s.
G_s	Group Table capacity of switch s.
S_e	Set of external nodes directly connected to the network.
S^+	Set of all nodes ($S^+ = S \cup S_e$).
$N^{\rightarrow}(s) \subseteq S^+$	Set of incoming neighboring nodes of switch $s \in S$ (i.e., neighbors from which s can receive packets).
$N^{\leftarrow}(s) \subseteq S^+$	Set of outgoing neighboring nodes of switch $s \in S$ (i.e., neighbors towards which s can send packets).
L	Set of directed links, defined by $(u, v) \in S \times S$, where u is the origin of the link and v is its termination.
I	Set of directed links connecting the multicast application servers to the aggregation switches (ingress links). The ingress link of a flow $f \in F$ is written l_f by abuse of notation.
E	Set of directed links connecting the aggregation switches to the destination Base Stations (egress links).
$E(f) \subseteq E$	Set of egress links for flow $f \in F$ according to the endpoint policy.
L^+	Set of all directed links (i.e., $L^+ = L \cup I \cup E$).
B_l	Capacity of link $l \in L^+$.
Variables	**Description**
$a^d_{f,l}$	Equals 1 if flow f to destination d passes through link l, 0 otherwise.
$g_{f,s}$	Equals 1 if flow f allocated to switch s is also put in its group table.

the destinations. If so, the function `allocate`$(a^d_{f,l}, g_{f,s})$ places the Group Table entry in the node s by setting $g_{f,s} = 1$. It also places flow table entries (setting $a^d_{f,l} = 1$) from the source l_f to the node s, as well as from s to all $d \in E(f)$ by choosing the paths with minimum average load over all links. Finally, the allocation matrices A and G are updated and the process is repeated for every flow in M. The asymptotic time complexity of the OFMAA is driven by the loop in line `6` and runs in $O\left(|M| \cdot (|S| + |L^+|)\right)$.

We present a detailed block diagram of the OFMAA in Figure 3.4.

Algorithm 3.1 OpenFlow Multicast Allocation Algorithm

INPUT: Set of flows F, set of switches S, and set of links L^+
OUTPUT: Allocation matrices $A = (a^d_{f,l})$ and $G = (g_{f,s})$
$A \leftarrow [0]_{F,L^+,E(f)}$
$G \leftarrow [0]_{F,S}$
$M \leftarrow \texttt{sort}(F, p_f, descending)$
for all $f \in M$ **do**
  $P(f) \leftarrow \texttt{paths}(l_f, E(f))$
  if $\texttt{groupTableNode}(P(f))$ **then**
    $\texttt{allocate}(a^d_{f,l}, g_{f,s})$
    $\texttt{update}(A, G)$
    break
  end if
end for

3.5 Steiner Tree Based OFMAA

The greedy algorithm OFMAA is fast, but it is not able to adequately approximate the optimal solution. Therefore, we designed the ST-OFMAA (Algorithm 3.2), which also tries to allocate each flow f sequentially. However, it calculates a ST T_f for the set of terminals $(l_f \cup E(f))$ and places the Group Table entry at one of the Steiner points.

Algorithm 3.2 Steiner Tree Based OFMAA

INPUT: Set of flows F, set of switches S, and set of links L^+, graph $V(S, L^+)$
OUTPUT: Allocation matrices $A = (a^d_{f,l})$ and $G = (g_{f,s})$
$A \leftarrow [0]_{F,L^+,E(f)}$
$G \leftarrow [0]_{F,S}$
$M \leftarrow \texttt{sort}(F, p_f, descending)$
for all $f \in M$ **do**
  $T_f \leftarrow \texttt{steinerCalc}(V, l_f, E(f), p_f)$
  if $\texttt{canAllocate}(T_f)$ **then**
    $\texttt{allocate}(a^d_{f,l}, g_{f,s})$
    $\texttt{update}(A, G, V)$
  else
    break
  end if
end for

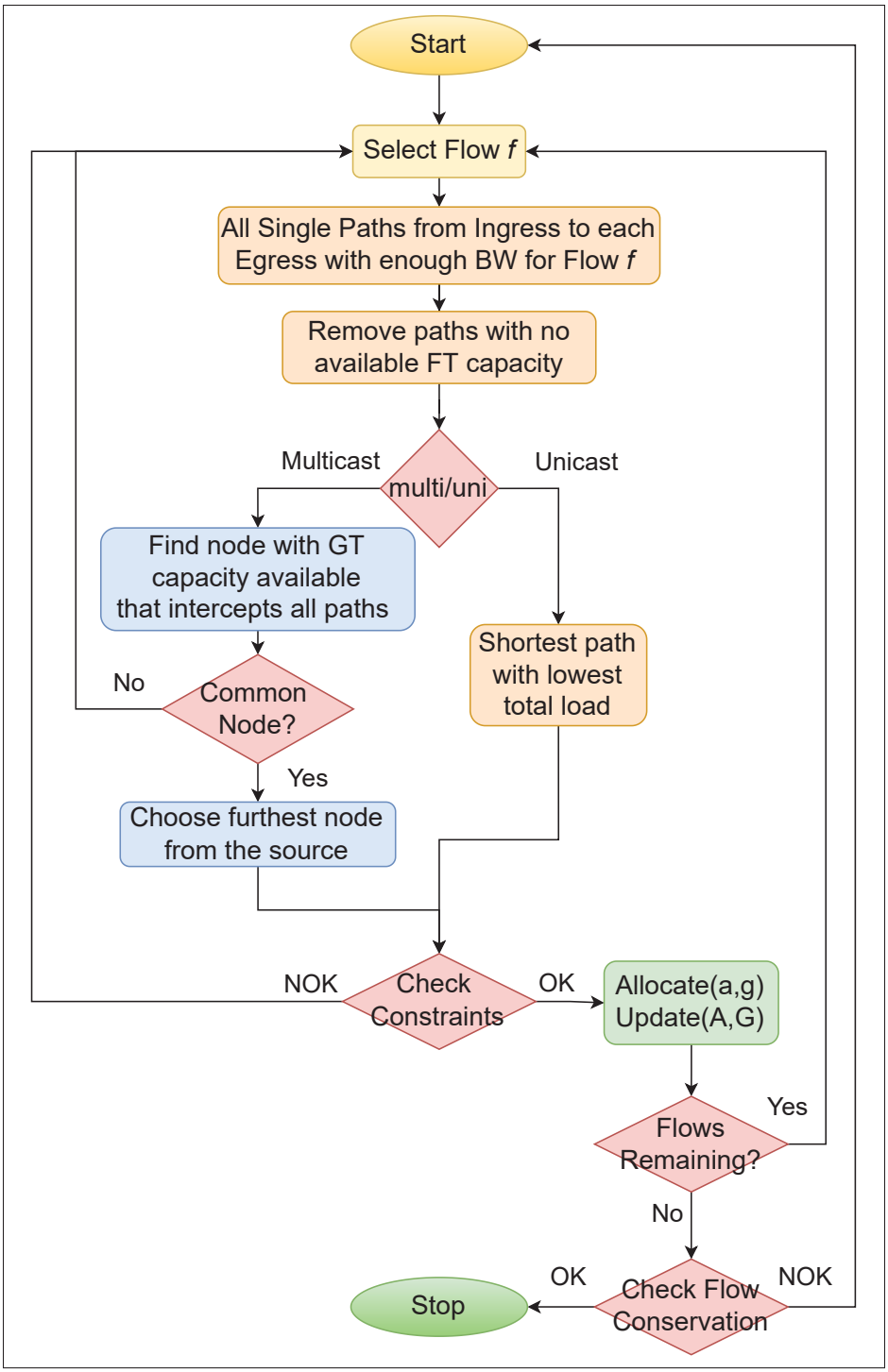

Figure 3.4 OFMAA Diagram

Based on an approximation of the Directed Steiner Tree [Watel & Weisser (2016)], the function `steinerCalc`$(V, l_f, E(f), p_f)$ checks for the value of p_f, generates a sub-graph with the

links with enough available capacity and calculates a Steiner Tree using the source l_F and the set of destinations $E(f)$ as terminals and stores this tree in T_f. `canAllocate`(T_f) checks if T_f is available. If so, the function `allocate`($a^d_{f,l}, g_{f,s}$) installs the Group Table entry in the Steiner (intermediate) node s furthest from l_f that can reach all destinations $E(f)$ by setting $g_{f,s} = 1$. It also installs flow table entries (setting $a^d_{f,l} = 1$) from the source l_f to the chosen node s, as well as from s to all $d \in E(f)$ in the tree branches. Finally, the allocation matrices A and G, and also the graph V are updated. In case `canAllocate`(T_f) returns false, flow f is not allocated and the process is repeated for every flow in M. The asymptotic time complexity of the ST-OFMAA is driven by the loop in line `6` and runs in $O(\ |M| \cdot ((|L^+||log(S)|k + |S^2||k^3|)))$, where k is the set of terminals ($l_f + |E(f)|$) in each allocation tree for a flow $f \in F$.

We present a detailed block diagram of the ST-OFMAA in Figure 3.5.

The outputs of both OFMAA and ST-OFMAA are the allocation matrices A and G. The matrix A provides the information necessary for defining Flow Table entries since it informs which links each flow will pass through. Likewise, the matrix G shows the switches in which the Group Tables must be installed for each flow. The operator can quickly generate FLOW_MOD messages from both matrices to push the appropriate rules in each switch.

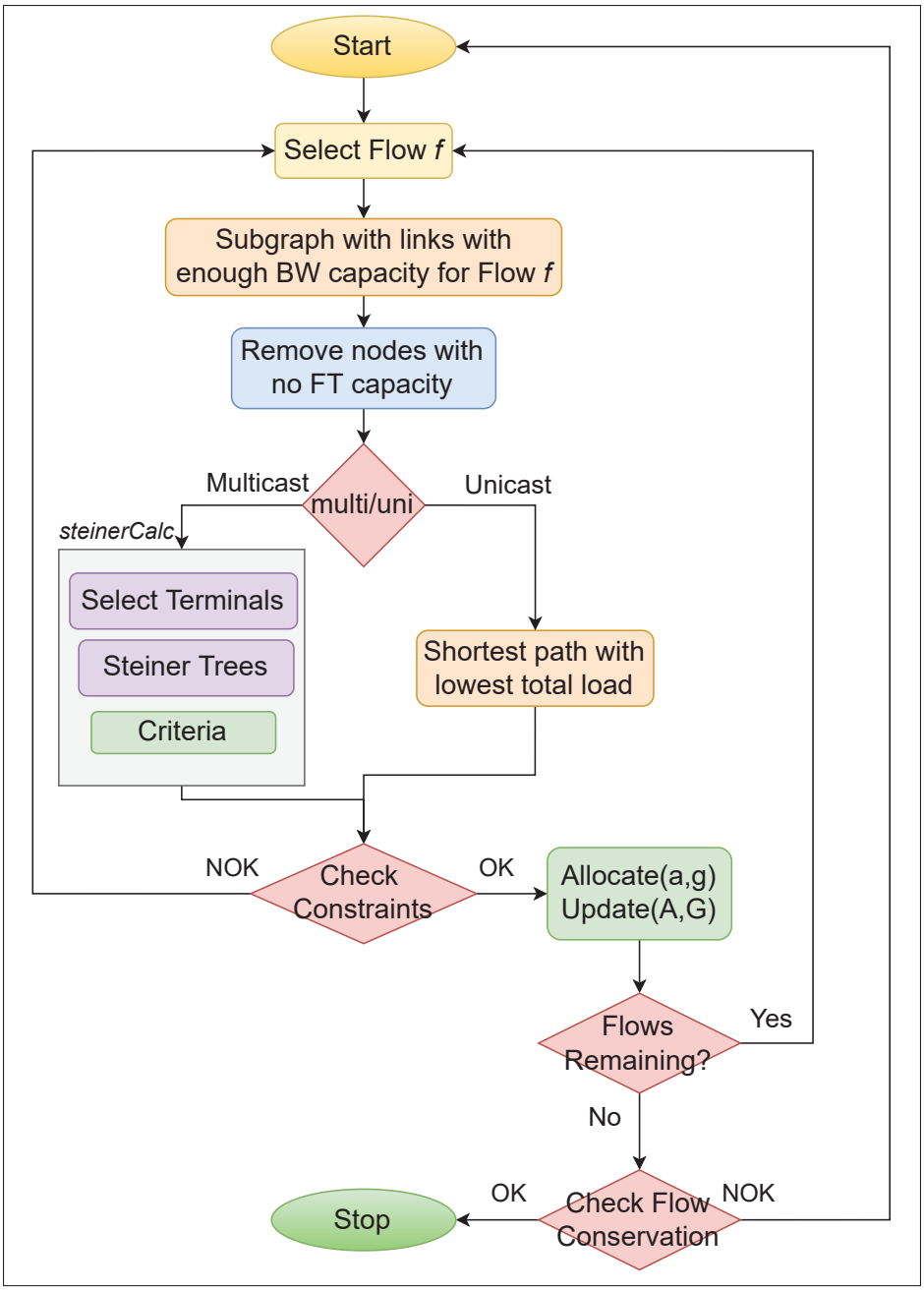

Figure 3.5 ST-OFMAA Diagram

CHAPTER 4

NUMERICAL RESULTS

We evaluate the performance of the proposed model in two simulated topologies: a 10-node mesh topology, the NSF network (depicted in Figure 4.1) both with equal link capacity and also on a nationwide testbed at TELUS Lab in Edmonton, AB, which is a packet-optical network with SDN capabilities. Telus Lab topology is depicted in Figure 4.2. All three topologies were assessed with unidirectional links.

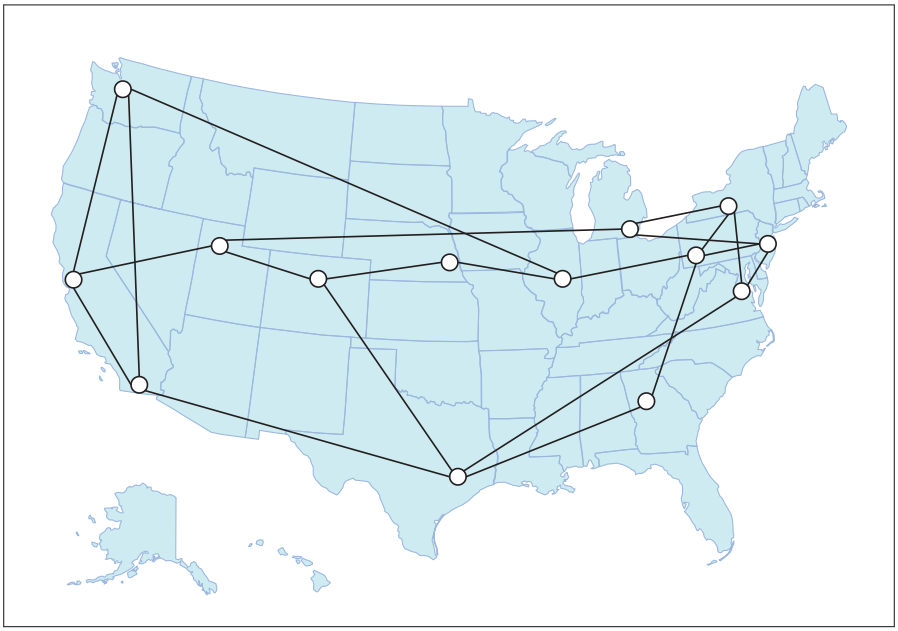

Figure 4.1 NSFNET - The National Science Foundation Network

We compare our algorithms with two baselines: i) a prior work that optimally allocates OpenFlow rules with no multicast consideration (Flow Table entries only), and ii) an optimal solution of the model in Equation (9) obtained by a mathematical solver. We generate a set of flows from two different ingress points and four different egress points in each topology. They are ordered by decreasing requested bandwidth with the highest values up to 5x the lowest ones to have flows of different sizes to allocate. The flows are mixed multicast and unicast to reflect the nature of multicast application traffic traversing the network alongside other types of streaming data. To compare the performance of our multicast model to the unicast approach, we create

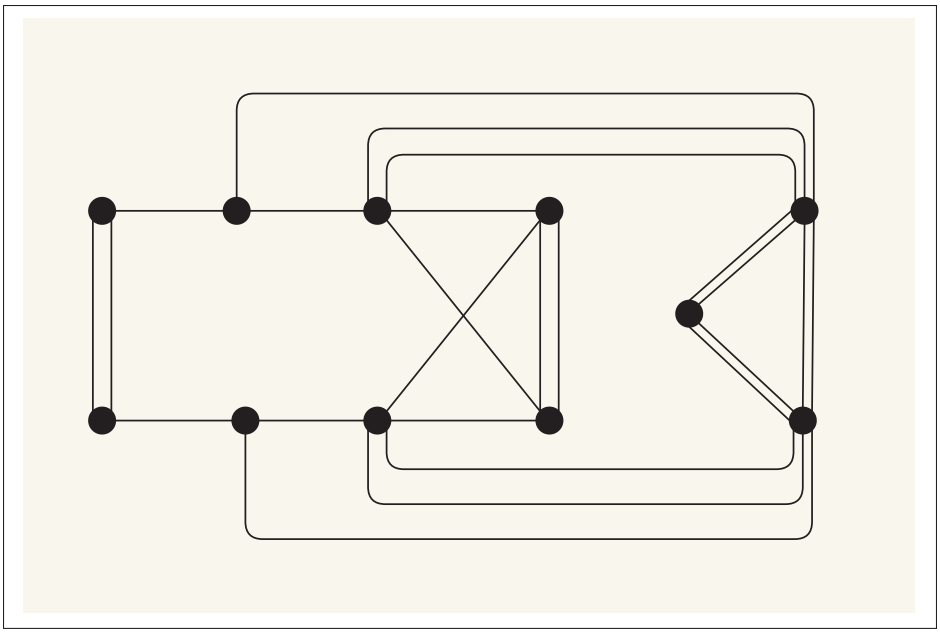

Figure 4.2 Topology of testbed at TELUS-Ciena Laboratory

different scenarios where the ratio of multicast to the total requested traffic increases, and we compare various key performance indicators in all scenarios. The ratio increases when more users receive traffic from the multicast application, thus demanding more resources from the network compared to other unicast traffic. We assume that the flow of any user is set up only if the bandwidth requirements (policy) are met. We show how our method consumes the network resources (links and flow table entries) in the different scenarios where the amount of multicast traffic traversing the network rises. We also compare our solution with a unicast-only model used to deliver the flows to the same egress points in each scenario.

We evaluate the Network-wide Average Link (NAL) utilization in the three topologies, and we compare the NAL when the different demands with multicast/unicast ratio are allocated using the optimization model, both Algorithms 3.1 and 3.2, and the equivalent unicast-only scheme. The initial point in each curve is the lowest load, in which each flow is delivered to one destination only, and they are different for each topology due to their design.

We now discuss the average link utilization in all three topologies evaluated. For instance, Telus topology is the lowest because we added the bidirectional links increasing the number

of links (Figure 4.3). So more options are available, and the flows can be more distributed across the layers and links. NSF is the more restrictive topology (fewer links available), so the link utilization is always higher than the other ones. Though Mesh topology NAL has the highest number of links available, since they are all unidirectional, effectively, there are fewer allocation options than the Telus network, so its NAL is slightly higher (Figure 4.4). The results show a similar trend: increasing the number of multicast flows by adding egress points (a.k.a. destinations) increases the traffic ratio since these flows require more resources. However, the number of unicast flows also increases because a new flow is needed for each new destination added. Nevertheless, when allocated using our model, the same flows require less link capacity overall. The path chosen by the optimization model will consume the same amount of resources for all destinations along the path from the source to the node where the Group Table is installed for that particular flow.

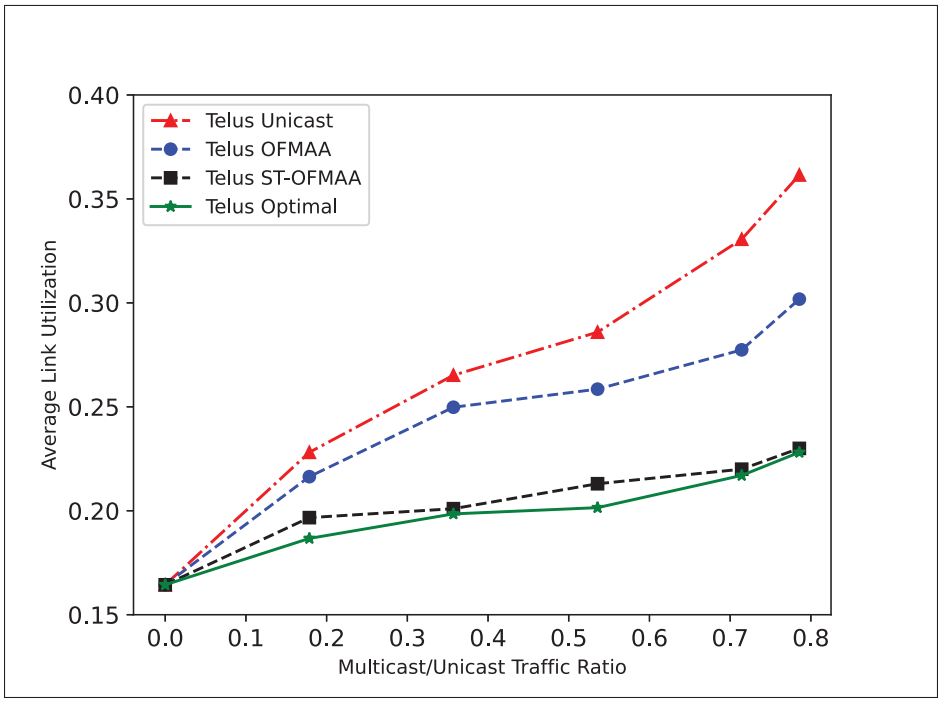

Figure 4.3 Network-wide Average Link (NAL) Utilization - Telus Topology

Figure 4.6 shows the savings in link utilization when we allocate the flows optimally and with our algorithms when compared to the baseline unicast case. We can see that for the extreme case where almost 80% of the traffic is multicast, the optimal allocation can consume, on average,

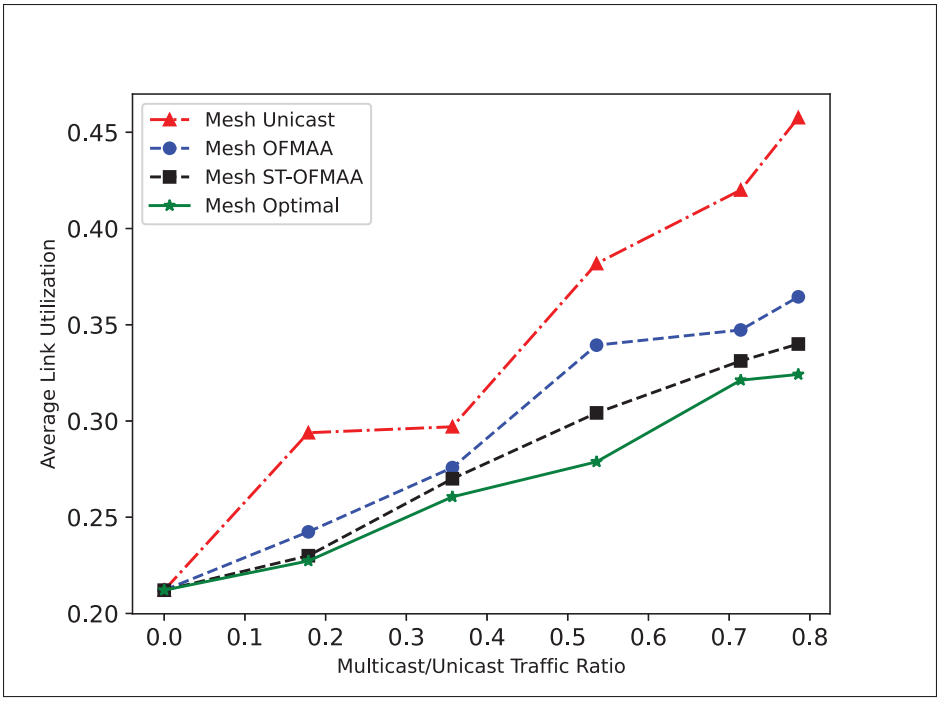

Figure 4.4 Network-wide Average Link (NAL) Utilization - Mesh Topology

30% less link capacity. Therefore, the ST-OFMAA algorithm saves up to 27% for the same case. When we compare Figure 4.6 and the NAL for each topology, it is clear that for the NSF network, the reduction is lower due to the limited number of options for the Group Table (see Figure 4.5). Telus testbed, in turn, can benefit significantly, as displayed by the difference between the optimal and the unicast curves in Figure 4.3.

In summary, ST-OFMAA saves up to 27%, which means 92% of the optimal performance.

We also evaluate the increasing number of installed flow table entries necessary to deliver the traffic to one destination Base Station (egress link). As the number of destinations increases (and also bandwidth traversing the network), the proposed method consumes fewer flow entries than the unicast equivalent. This behavior is possible because the model installs one flow entry per node along the path until the Group Table is installed. From there, the traffic is replicated to the different egress links. Figure 4.7 illustrates the average link utilization growth as the number of users in the application increases. The proposed method shows a much lower slope in

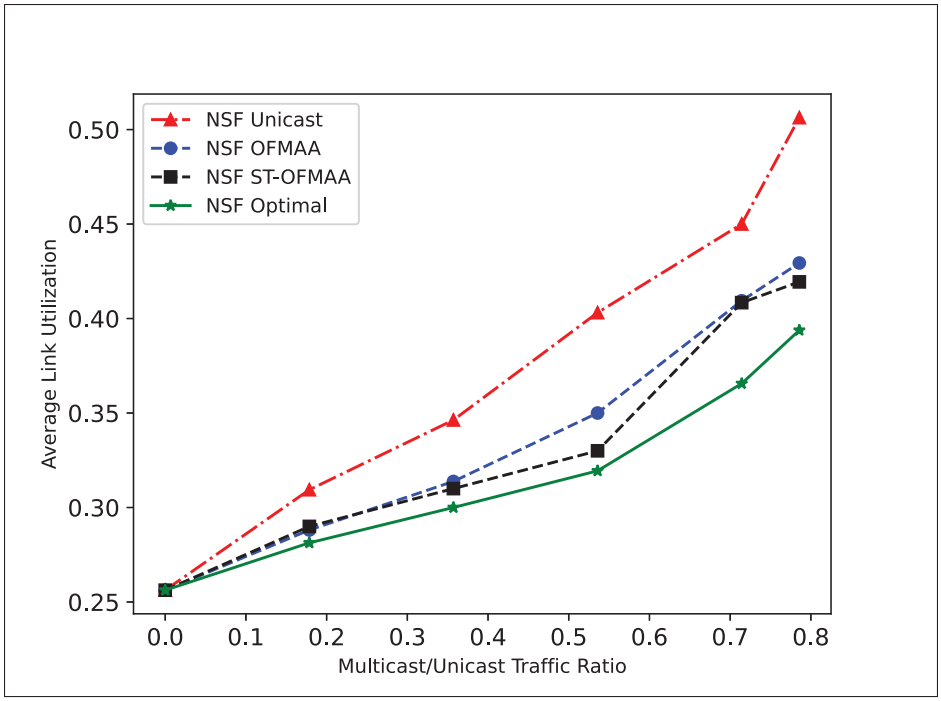

Figure 4.5 Network-wide Average Link (NAL) Utilization - NSF Topology

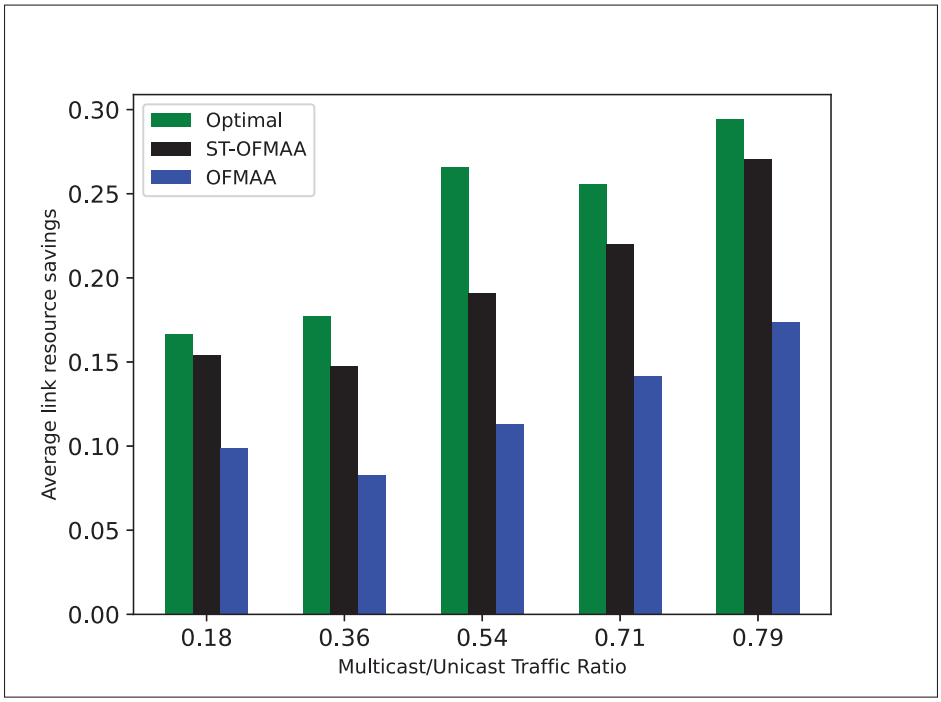

Figure 4.6 Link utilization reduction

the curve, which means we can accommodate a much higher number of users under the same network when compared with the unicast baseline. Finally, we calculate the number of flows required to afford a total demand in four cases: no multicast, allocating multicast using OFMAA and ST-OFMAA, and optimally allocating multicast, as displayed in Figure 4.8. As the number of users increases, the number of paths to be allocated to the increasing number of destinations grows. Also grows the number of flow table entries necessary to deliver the flows to the egress points. The proposed method consumes between 6% and 22% fewer flow table entries for the same demand than the unicast-only allocation.

Our results show that the proposed model and algorithm increase the network's overall efficiency, accommodating a higher number of receivers than the unicast-only solution. Furthermore, our solution supports both unicast, and multicast flows by design, making it suitable for multicast applications that share the same network with many other types of traffic.

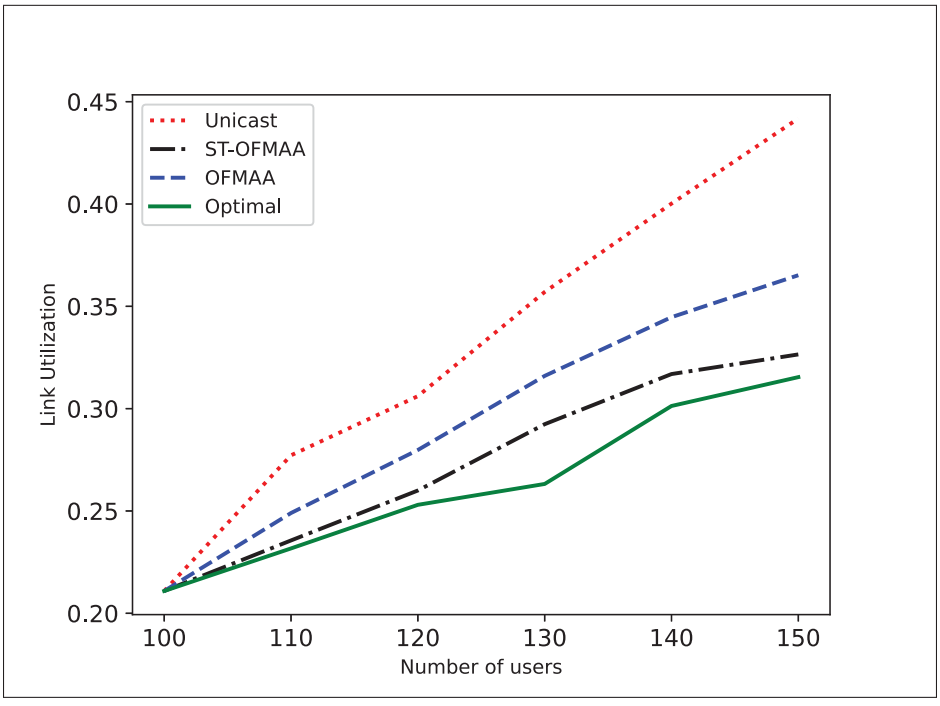

Figure 4.7 Link Utilization increases as the number of users grows

We also evaluate the increasing number of installed flow table entries necessary to deliver the traffic to one destination Base Station (egress link). As the number of destinations increases

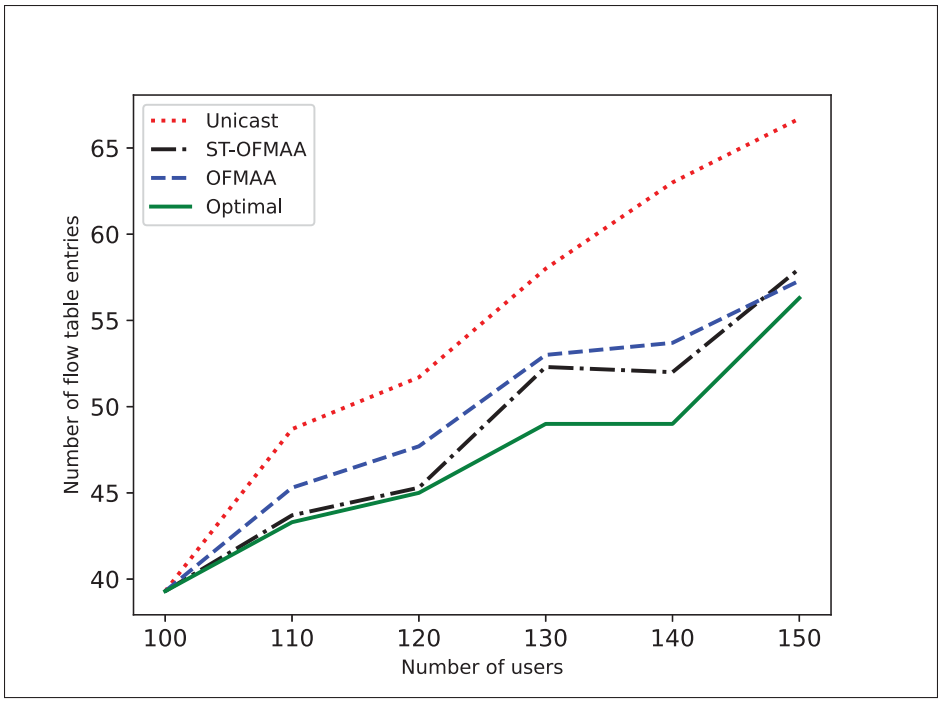

Figure 4.8 Flow Table entries - users

(and also bandwidth traversing the network), the proposed method consumes fewer flow entries than the unicast equivalent. This behavior is possible because the model installs one flow entry per node along the path until the Group Table is installed. From there, the traffic is replicated to the different egress links. Figure 4.9 shows that, as the number of users in the application (and also the multicast/unicast traffic ratio) increases, our approach consumes, on average, a smaller number of flow table entries and, with this, saves memory in the switches. Finally, when we compare the average reduction in the Flow Table entries (Figure 4.10), we can see that ST-OFMAA saves up to 18%, which means 97% of the optimal performance.

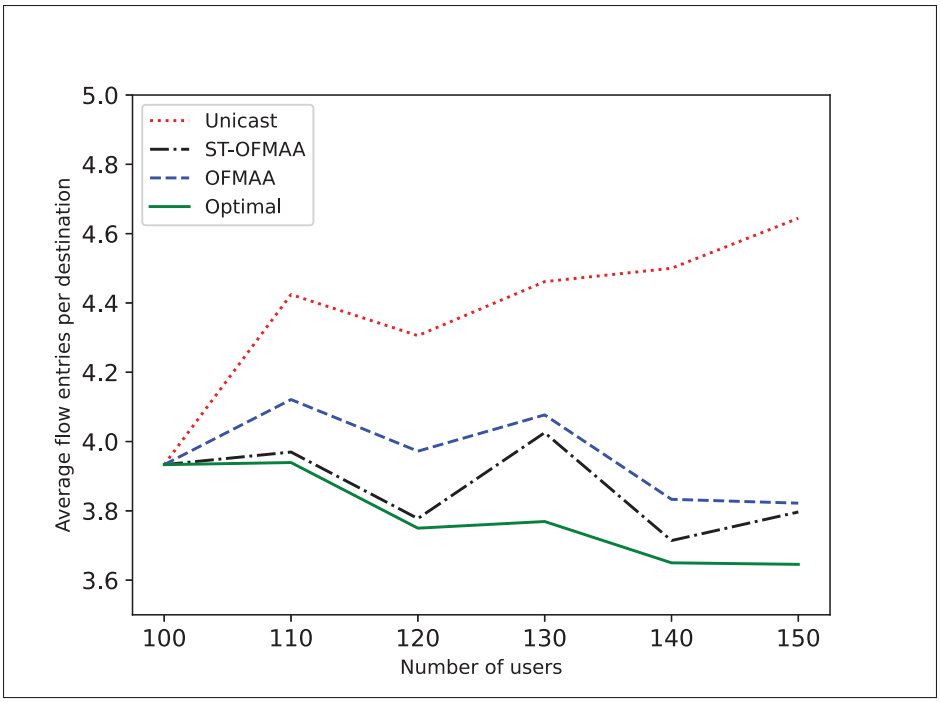

Figure 4.9 Average Flow Table entries per destination Base Station

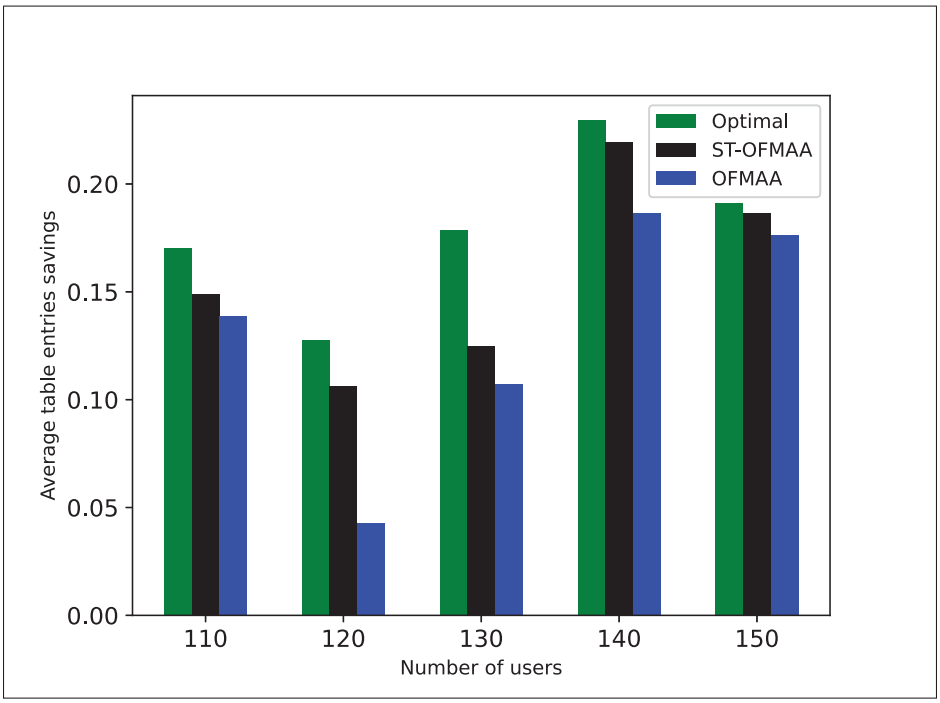

Figure 4.10 Average Flow Table entries savings

CONCLUSION AND RECOMMENDATIONS

In this thesis, we addressed the problem of routing traffic of multicast applications through OpenFlow-enabled carrier networks. OpenFlow is the most used SDN platform in the industry nowadays, and operators are increasingly integrating programmable switches into their access and core networks.

To tackle the requirements of such applications, we proposed a non-linear integer optimization model to efficiently allocate forwarding rules for both unicast and multicast flows in OpenFlow-based networks. Our model leverages the Group Table feature of OpenFlow and optimizes the combined placement of Flow Table and Group Table entries. The objective is to reduce the number of Flow Table entries in the switches, increasing the throughput and the number of flows that can be installed. We designed two versions of an algorithm to solve the optimization problem in polynomial time, then carried out experiments showing that our solution can support a higher number of flows than the solutions proposed by prior work, which do not consider Group Tables, by reducing both the link usage by up to 30% (in the extreme case where almost 80% of the traffic is multicast). We also showed that the number of flow entries needed to deliver the traffic to destinations is reduced by 22% compared to the baseline, which allows the accommodation of a higher number of users in the application.

However, we do not cover the strategies to handle the new arriving policies. One approach would be to enforce the new policy with the current ones, which is challenging because of the residual bandwidth and memory capacity constraints. The placement will not be optimal, thus decreasing the overall efficiency. In contrast, when a new policy arrives, the controller might enforce the old and new policies as a set, which would result in better use of the overall resources, but it is more costly and can lead to traffic disruption. In this context, we could leverage predictors (e.g., Machine Learning tools) based on the historical data about the incoming requests in an

hourly fashion. With this, we can minimize the changes in the network, which will cause less disruption and save costs.

In future work, we will study the strategies mentioned above to extend the model to handle new policy arrivals. We are also interested in investigating multicast in different scenarios, such as software-defined radio.

Publications

The main content of this thesis has been accepted in the ACM SAC'22 conference and will be published in April 2022:

- Rafael George Amado, Kim-Khoa Nguyen, and Mohamed Cheriet. "OpenFlow Rule Placement In Carrier Networks For Augmented Reality Applications". In The 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22), April 25–29, 2022, Virtual Event. ACM, New York, NY, USA. https://doi.org/10.1145/3477314.3507101

BIBLIOGRAPHY

Bera, S., Misra, S. & Jamalipour, A. (2019). FlowStat: Adaptive flow-rule placement for per-flow statistics in SDN. *IEEE Journal on Selected Areas in Communications*, 37(3), 530–539.

Bhaumik, P., Zhang, S., Chowdhury, P., Lee, S.-S., Lee, J. H. & Mukherjee, B. (2014). Software-defined optical networks (SDONs): a survey. *Photonic Network Communications*, 28(1), 4–18.

Cao, Y., Jiang, T., Chen, X. & Zhang, J. (2015). Social-aware video multicast based on device-to-device communications. *IEEE Transactions on Mobile Computing*, 15(6), 1528–1539.

Chen, M.-H., Hu, K.-W., Chung, I.-H. & Chou, C.-F. (2019). Towards VR/AR multimedia content multicast over wireless LAN. *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6.

Cheng, M.-H., Hwang, W.-S., Wu, Y.-J., Lin, C.-H. & Syu, J.-S. (2020). An Effective Flow-Rule-Reducing Algorithm for Flow Tables in Software-Defined Networks. *2020 International Computer Symposium (ICS)*, pp. 25–30.

Cisco, V. N. I. (2017). Global Mobile Data Traffic Forecast Update, 2017-2022. *white paper*.

Costa, L. C., Vieira, A. B., e Silva, E. d. B., Macedo, D. F., Gomes, G., Correia, L. H. & Vieira, L. F. (2017). Performance evaluation of OpenFlow data planes. *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 470–475.

Costa, L. C., Vieira, A. B., e Silva, E. d. B., Macedo, D. F., Vieira, L. F., Vieira, M. A., Junior, M. d. R. M., Batista, G. F., Polizer, A. H., Gonçalves, A. V. G. S. et al. (2021). OpenFlow data planes performance evaluation. *Performance Evaluation*, 147, 102194.

Feldmann, A., Gasser, O., Lichtblau, F., Pujol, E., Poese, I., Dietzel, C., Wagner, D., Wichtlhuber, M., Tapiador, J., Vallina-Rodriguez, N. et al. (2021). A year in lockdown: how the waves of COVID-19 impact internet traffic. *Communications of the ACM*, 64(7), 101–108.

Fernández, D., Contreras, L. M., Moyano, R. F. & García, S. (2021). NFV/SDN Based Multiple Upstream Interfaces Multicast Proxy Service. *2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pp. 159–163.

Giroire, F., Moulierac, J. & Phan, T. K. (2014). Optimizing rule placement in software-defined networks for energy-aware routing. *2014 IEEE Global Communications Conference*, pp. 2523–2529.

Guo, Y., Luo, H., Wang, Z., Yin, X. & Wu, J. (2021). Routing optimization with path cardinality constraints in a hybrid SDN. *Computer Communications*, 165, 112–121.

Haleplidis, E., Salim, J. H., Halpern, J. M., Hares, S., Pentikousis, K., Ogawa, K., Wang, W., Denazis, S. & Koufopavlou, O. (2015). Network programmability with ForCES. *IEEE Communications Surveys & Tutorials*, 17(3), 1423–1440.

Huang, H., Guo, S., Li, P., Ye, B. & Stojmenovic, I. (2015). Joint optimization of rule placement and traffic engineering for QoS provisioning in software defined network. *IEEE Transactions on Computers*, 64(12), 3488–3499.

Humernbrum, T., Hagedorn, B. & Gorlatch, S. (2016). Towards efficient multicast communication in software-defined networks. *2016 IEEE 36th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pp. 106–113.

Islam, S., Muslim, N. & Atwood, J. W. (2017). A survey on multicasting in software-defined networking. *IEEE Communications Surveys & Tutorials*, 20(1), 355–387.

Islam, S., Muslim, N. & Atwood, J. W. (2018). A Survey on Multicasting in Software-Defined Networking. *IEEE Communications Surveys Tutorials*, 20(1), 355-387. doi: 10.1109/COMST.2017.2776213.

Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M. et al. (2013). B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM Computer Communication Review*, 43(4), 3–14.

Kang, N., Liu, Z., Rexford, J. & Walker, D. (2013). Optimizing the "One Big Switch" Abstraction in Software-Defined Networks. *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, (CoNEXT '13), 13–24. doi: 10.1145/2535372.2535373.

Katta, N., Alipourfard, O., Rexford, J. & Walker, D. (2014). Infinite cacheflow in software-defined networks. *Proceedings of the third workshop on Hot topics in software defined networking*, pp. 175–180.

Kim, H.-s., Yun, S., Kim, H. & Kim, W.-T. (2017). A novel SDN multicast for large-scale IoT environments. *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 823–828.

Kobo, H. I., Abu-Mahfouz, A. M. & Hancke, G. P. (2017). A survey on software-defined wireless sensor networks: Challenges and design requirements. *IEEE access*, 5, 1872–1899.

Kotachi, S., Sato, T., Shinkuma, R. & Oki, E. (2020). Multicast Routing Model to Minimize Number of Flow Entries in Software-Defined Network. *IEICE Transactions on Communications*.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S. & Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1), 14–76.

Lanner. [[Accessed: 2022-01-05]]. (2021). Lanner HTCA blade HLM-1100. Retrieved from: https://www.lannerinc.com/products/telecom-datacenter-appliances/modules-and-blades/hlm-1100.

Lecompte, D. & Gabin, F. (2012). Evolved multimedia broadcast/multicast service (eMBMS) in LTE-advanced: Overview and Rel-11 enhancements. *IEEE Communications Magazine*, 50(11), 68–74.

Luo, S., Xing, H. & Fan, P. (2021). Softwarized IP Multicast in the Cloud. *IEEE Network*, 35(6), 233-239. doi: 10.1109/MNET.100.2100045.

Majidi, A. & Zahran, A. H. (2020). Optimized Joint Unicast-Multicast Panoramic Video Streaming in Cellular Networks. *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pp. 1–6.

Molnár, M., Bellabas, A. & Lahoud, S. (2012). The cost optimal solution of the multi-constrained multicast routing problem. *Computer Networks*, 56(13), 3136–3149.

Nguyen, V.-G., Do, T.-X. & Kim, Y. (2016a). SDN and virtualization-based LTE mobile network architectures: A comprehensive survey. *Wireless Personal Communications*, 86(3), 1401–1438.

Nguyen, X.-N., Saucez, D., Barakat, C. & Turletti, T. (2015). OFFICER: A general optimization framework for OpenFlow rule allocation and endpoint policy enforcement. *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 478–486.

Nguyen, X.-N., Saucez, D., Barakat, C. & Turletti, T. (2016b). Rules Placement Problem in OpenFlow Networks: A Survey. *IEEE Communications Surveys Tutorials*, 18(2), 1273-1286. doi: 10.1109/COMST.2015.2506984.

Noviflow. [[Accessed: 2021-07-26]]. (2019a). NoviSwitch 2122 Spec Sheet. Retrieved from: https://noviflow.com/wp-content/uploads/2019/11/NoviSwitch-2122-Datasheet-400_V5.pdf.

Noviflow. [[Accessed: 2021-07-26]]. (2019b). NoviWare 500.2 for Barefoot Tofino Chipset. Retrieved from: https://noviflow.com/wp-content/uploads/2019/12/NoviWare-Tofino-500.2-Datasheet-FINAL.pdf.

Nunes, B. A. A., Mendonca, M., Nguyen, X.-N., Obraczka, K. & Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications surveys & tutorials*, 16(3), 1617–1634.

OpenFlowSpec. [[Accessed: 2021-07-26]]. (2015). The OpenFlow Switch Specification Version 1.5.1. Retrieved from: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

Petale, S. & Thangaraj, J. (2020). Link Failure Recovery Mechanism in Software Defined Networks. *IEEE Journal on Selected Areas in Communications*, 38(7), 1285-1292. doi: 10.1109/JSAC.2020.2986668.

Ren, P., Qiao, X., Huang, Y., Liu, L., Pu, C., Dustdar, S. & Chen, J.-L. (2020). Edge AR X5: An Edge-Assisted Multi-User Collaborative Framework for Mobile Web Augmented Reality in 5G and Beyond. *IEEE Transactions on Cloud Computing*.

Rifai, M., Huin, N., Caillouet, C., Giroire, F., Moulierac, J., Pacheco, D. L. & Urvoy-Keller, G. (2017). Minnie: An sdn world with few compressed forwarding rules. *Computer Networks*, 121, 185–207.

Saha, N., Misra, S. & Bera, S. (2021). Q-Flag: QoS-Aware Flow-Rule Aggregation in Software-Defined IoT Networks. *IEEE Internet of Things Journal*.

Shah, S. D. A., Gregory, M. A., Li, S. & Fontes, R. D. R. (2020). SDN enhanced multi-access edge computing (MEC) for E2E mobility and QoS management. *IEEE Access*, 8, 77459–77469.

Sheu, J.-P., Lin, W.-T. & Chang, G.-Y. (2018). Efficient TCAM rules distribution algorithms in software-defined networking. *IEEE Transactions on Network and Service Management*, 15(2), 854–865.

vSwitch, O. [[Accessed: 2021-07-28]]. (2015). Open vSwitch. Retrieved from: http://www.openvswitch.org/.

Wang, Z. & Crowcroft, J. (1996). Quality-of-service routing for supporting multimedia applications. *IEEE Journal on selected areas in communications*, 14(7), 1228–1234.

Watel, D. & Weisser, M.-A. (2016). A practical greedy approximation for the directed steiner tree problem. *Journal of Combinatorial Optimization*, 32(4), 1327–1370.

Zaidi, Z., Friderikos, V., Yousaf, Z., Fletcher, S., Dohler, M. & Aghvami, H. (2018). Will SDN be part of 5G? *IEEE Communications Surveys & Tutorials*, 20(4), 3220–3258.

Zhang, X., Shen, X. & Yu, Z. (2019). A novel hybrid ant colony optimization for a multicast routing problem. *Algorithms*, 12(1), 18.

Zhang, X., Yang, M., Zhao, Y., Zhang, J. & Ge, J. (2017). An SDN-based video multicast orchestration scheme for 5G ultra-dense networks. *IEEE Communications Magazine*, 55(12), 77–83.