MULTIPLE-DESCRIPTION CODED STREAMING VIDEO MULTICAST OVER SOFTWARE-DEFINED NETWORKS

A Thesis

by

Kyoomars Alizadeh Noghani

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Science

Özyeğin University
August 2014

MULTIPLE-DESCRIPTION CODED STREAMING VIDEO MULTICAST OVER SOFTWARE-DEFINED NETWORKS

Approved by:

Assoc. Prof. Dr. M. Oğuz Sunay, Advisor
Department of Computer Science
Özyeğin University

Prof. Dr. Reha Civanlar
Department of Computer Science
Özyeğin University

Assist. Prof. Dr. Ali Özer Ercan
Department of Computer Science
Özyeğin University

Date Approved: 28 August 2014

To my Mother and Father,

who have blessed me with an admiration of nature

and to my Wife,

Farzaneh,

whose love and confidence is a constant source of inspiration and encouragement.

ABSTRACT

Video has become one of the most prominent applications of the Internet. Many of the video streaming applications involve the distribution of content from a CDN source to a large population of interested clients. However, widespread support of IP-Multicast has been unavailable to a large extent due to technical and economical reasons, all stemming from the non-programmable nature of today's Internet. As a solution, streaming multicast video is commonly operated using application level multicast. However, this technique introduces excessive delays for the clients and increased traffic load for the network. This thesis is concerned with the introduction of a SDN based framework that allows the network controller to not only deploy IP-Multicast between a source and subscribers, but also control, via a simple northbound interface, the distributed set of sources where multiple-description coded video content is available. Standard and premium users are envisioned. While standard subscribers are to receive one of the descriptions of the video, premium subscribers will receive multiple descriptions, each from a different source, simultaneously and combine these descriptions prior to playback for increased video quality. In the framework, the controller constructs and maintains a dynamic multicast tree from each source and formulates the associated multicast routes. An experimental testbed has been setup on Mininet to assess the performance of the SDN-based streaming multicast video application using QoS performance metrics on a well-known test videos. We observe that for medium to heavily loaded networks, relative to todays solution of application layer multicast in a non-SDN network, the SDN-based streaming multicast video framework increases the PSNR of the received video significantly, from a level that is practically unwatchable to one that has good quality.

ÖZETÇE

Video internetin en önemli uygulamalarından biri haline geldi. Video akışı sağlıyan uygulamaların birçoğu CDN kaynağından, video talep eden çokça kullanıcıya dağıtılır. Fakat, günümüz internetinin programlanabilir olmayan yapısından dolayı, IP-Multicast tabanlı uygulamaların geniş kitlelere ulasamamaktadır. Multicast video akışı için kullanılan çözümlerden birisi, uygulama katmanında multicast yapmaktır. Fakat bu teknik, iletişim ağı için ise trafik yoğunluğunun ve kullanıcılar için gecikmenin artmasına sebep olur. Bu calısma, SDN tabanlı, ag denetleyicisinin bir kaynak ve kullanıcılar arası IP-Multicast servisini sagladığı gibi, basit bir üst seviye yönelimli (northbound) arayüz ile coklu betimlenmis video icerigi barındıran dağıtık kaynakları da kontrol ettigi bir cerceve uzerinedir. Bu çalışmada standart ve öncelikli (premium) kullanıcılar olduğunu varsaydık. Standart kullanıcılar video nun tek betimlemesini (description) alırken, öncelikli kullanıcılar birden fazla betimlemesini alabiliyor. Öncelikli kullanıcılar, farklı kaynaklardan, aynı anda sözbu betimlemeleri alıp, oynatımdan önce birleştirerek, video kalitesini arttırmaktadırlar. Bu çalışmada kontrolör -kullanıcılardan aldığı bilgi ışığında- dinamik bir multicast ağacı inşa edip, bu ağacın yenilenmesini sürdürerek, multicast rotaları oluşturur. SDN tabanlı multicast video akışının performansını değerlendirme amacıyla, deneysel kurulum Mininet ile inşa edildi. Bu çalışmada, SDN tabanlı multicast video akış metodolojisinin, günümüz SDN tabanlı olmayan uygulama katmanındaki multicast çözümlerine oranla, yoğun trafik bulunan ağlar için, video PSNR değerini kayda değer şekilde arttırdığı gözlemlenmiştir.

ACKNOWLEDGEMENTS

I would like to express my endless gratitude to Assoc. Prof. M. Oğuz Sunay for his excellent advisory, reliable guidance and full support. This thesis have not been written without his profound knowledge. I also extend my gratitude to Prof. Reha Civanlar and Assist. Prof. Ali Özer Ercan for being in my thesis committee and for their valuable time.

My special appreciation goes to Dr. Volkan Yazici for his friendship, encouragement, guidance, and number of fruitful discussions. I thank my friends Ali Arsal and Kıvanç Çakmak, for making these two years enjoyable.

Last but not least, I thank to my mother, Akram, my father Mehdi, my brother Siamak and my sister Mahsa for their everlasting love and support. Obviously, I thank the first light of my life, Farzaneh, for her eternal love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

ALM	Application Layer Multicast, p. 4.
BFS	Breadth First Search, p. x.
CDN	Content Delivery Network, p. iv.
DF	Designated Forwarder, p. 30.
DFS	Depth First Search, p. 34.
DiffServ	Differentiated services, p. 77.
DM	Dense Mode, p. 28.
DP	Description Provider, p. 19.
DPI	Deep Packet Inspection, p. 17.
DR	Designated Router, p. 29.
IGMP	Internet Group Management Protocol, p. 17.
ISM	Internet Standard Multicast, p. 31.
ISP	Internet Service Providers, p. 13.
LC	Layer Coding, p. 8.
LLDP	Link Layer Discovery Protocol, p. 21.
MD	Multiple-Description, p. 8.
MDC	Multiple-Description Coding, p. 2.
MST	Minimum Spanning Tree, p. 37.
P2P	Peer-to-Peer, p. 11.
PIM	Protocol Independent Multicast, p. 28.
PSNR	Peak Signal-to-Noise Ratio, p. iv.
QoE	Quality of Experience, p. 4.
QoS	Quality of Service, p. iv.
RP	Rendezvous Point, p. x.
RPT	Rendezvous Point Tree, p. 30.

SDN	Software-Defined Networking, p. iv.
SM	Sparse Mode, p. 28.
SSM	Source Specific Multicast, p. 28.
STP	Spanning Tree Protocol, p. 52.
SVC	Scalable Video Coding, p. viii.
VoD	Video on Demand, p. 15.

CHAPTER I

INTRODUCTION

Due to the explosive growth of the Internet and increasing demand for multimedia information on the web, streaming video over the Internet has received tremendous attention from academia and industry [1]. However, video streaming over best-effort networks (i.e. today's Internet) is always challenged by a number of factors such as high bit rates, delay, and loss sensitivity. To this end, many solutions have been proposed from different perspectives such as source and channel coding, protocols and network perspective.

In addition, decreasing total network traffic by means such as deploying IP-Multicast (which subsequently leads to decrease the probability of network congestion) or utilization of Quality of Service (QoS) for multimedia traffics would be also helpful to enhance the end-to-end multimedia packets likelihood. However, many parts of the Internet architecture were developed 30 years ago and its structure is not secure, robust, manageable and flexible for fundamental changes [2]. Hence, fulfilling the dreams of utilizing innovative ideas such as QoS and IP-Multicast in its real application are not simply achievable.

By combining solutions for efficient video streaming over best-effort networks with network solution for decreasing network congestion level, we designed and implement video streaming framework which is practical, resilient to packet loss and has good streaming performance over heavily loaded networks in comparison to today's video streaming frameworks. The proposed streaming video multicast framework is built on following pillars:

1. Path Diversity

2. Multiple Description Coding (MDC)

3. Distributed Video Streaming

4. IP Mulitcast

5. Software-Defined Networking (SDN)

Path diversity is a transmission technique that sends data simultaneously through two or more paths in a packet-based network. By using multiple paths at the same time (from single or multiple sources) the end-to-end video application effectively sees an "average" path behavior. We refer to this as path diversity. Generally, seeing this average path behavior provides better performance than seeing the behavior of any randomly chosen individual path. The benefits of path diversity include:

- The application sees a virtual average path which exhibits a smaller variability in communication quality than exists over an individual path,
- Burst packet losses are converted to isolated packet losses,
- The probability of an outage (where all packets in the packet stream are lost for the duration of the outage) is greatly reduced.

Above mentioned improvements provide some interesting benefits to video communication performance under packet loss, and may also simplify general packet-based communication system design.

When path diversity is properly combined with adaptive encoding solutions such as MDC [3], the definition of multiple end-to-end paths from the server to the client can dramatically improve the quality of service. MDC addresses the problem of encoding a source for transmission over a communication system with multiple channels. MDC coding produces two or more bitstreams, or descriptions, each of roughly equal importance and it has two important properties. First, each description can be independently decoded to provide a usable reproduction of the original signal. Second, the multiple descriptions contain complementary information so that the quality of the decoded signal improves with the number of descriptions received. MDC coding enables a useful reproduction of the signal when any description is received.

Path diversity and MDC combines particularly well with multiple sources, in which the different descriptions are explicitly distributed over servers at different physical locations. General framework offered by overlay networks such as Content Delivery Network (CDN) architectures are realizing multiple sources fact by making content available at multiple sources. Originally CDN idea has been developed to overcome performance problems, such as network congestion and server overload by using edge architecture and make content available at multiple sources. Since content is delivered from the closest edge server and not from the origin server, the content is sent over a shorter network path, thus reducing the request response time, the probability of packet loss, and the total network resource usage.

Our proposed method use MDC schemes because video content which coded this scheme can be easily distributed among multiple servers over different locations (by leveraging infrastructure like CDN) and it could easily be adapted to take advantage of path diversity idea. The main motivation in doing so is to exploit path diversity in order to achieve higher throughput, and to increase tolerance to packet loss and delay due to network congestion. In addition we implemented multicasting at IP level for streaming video packets to decrease the probability of network congestion and subsequently enhance the end-to-end multimedia packets likelihood.

It is predicted that approximately 73% of all IP traffic will be video by 2017 [4], of which some 14% will be from Internet video to TVs. Not surprisingly, streaming of live content is increasingly more prevalent on the Internet replacing the traditional means of TV broadcasting. One well-known method to alleviate the traffic load due to streaming video is to use IP-Multicast, which has been in existence for a long time. However, in today's networks, IP-Multicast has remained largely undeployed due to concerns on security, reliability and scalability, not to mention the requirement to have all routers in the network support the related protocols and be appropriately configured [5]. For this reason, Application-Layer Multicast (ALM) has found prominence in the Internet where transmission of the content to the subscriber group is managed at the application layer and IP-unicast is used in the network layer for delivery with multiple copies of the same data transmitted over common links, incurring heavy loads on the Internet traffic. Additionally, compared to the IP-Multicast, ALM incurs longer latencies. The fundamental reasons behind the prevalence of ALM despite its shortcomings, are its immediate deployability, adaptability and updatability [6].

The rapid emergence of SDN with significant industry backing [7] provides the perfect opportunity to implement IP-Multicast without any of its problems. Indeed, it is possible to construct, and maintain the multicast tree between a source and all its subscribers using a control application running on the logically centralized SDN-controller that has a global network view. The programmable nature of SDN allows for immediate deployability, scalability, adaptability, and updatability - traits all previously associated with ALM and not IP-Multicast. In this thesis, we present an IP-Multicast application running on the SDN controller that also keeps track of the subscription activities via a simple northbound interface.

IP-Multicast is an ideal approach to mitigate the traffic load generated by streaming video services. A more efficient delivery of the video packets reduces the congestion probability in the network, which in turn improves the performances of both the corresponding streaming video system and all other concurrently running services on the network. In this thesis, we first present an IP-Multicast framework for SDN, where we give a detailed description of the streaming video application, and its interaction with the SDN controller. Then we investigate how an actual implementation of IP-Multicast improve the streaming video performance relative to ALM in terms of Quality of Experience (QoE) metrics.

We require the streaming video application to be designed to satisfy the following:

1. Support for different types of QoE-level based subscriptions should be present.
2. Resilience to network congestion and packet losses should be provided.
3. The video coding and decoding complexities should be as low as possible.

To satisfy these requirements, we consider an architecture that has the following properties:

1. Multiple streaming video servers, distributed across the network are to be deployed.
2. H.264-based MDC is to be employed for video coding [8, 9] so that the same video content is described by multiple descriptions where reception of one such description is sufficient for standard-quality playback, but delivery of multiple descriptions and a simple combining procedure of these descriptions prior to playback result in an increase in the video quality.

In the proposed system, we consider a streaming server with two descriptions, available at two distinct locations of the network. We consider two subscription types including Standard and Premium. While a standard user is to receive content from one of the servers, premium users need to receive both descriptions for enhanced service quality, all orchestrated by a streaming-video specific IP-Multicast application running on the SDN controller.

In the subsequent chapters we first provide a literature survey on various video delivery frameworks and IP-Multicast implementations in today's Internet and SDN structures. We then present the proposed MDC-based streaming video service, followed by the SDN architecture on which it will operate and discuss the unnecessary interaction between the video application and the SDN controller. Next, we present experimental performance results for the video application using the QoE parameters of PSNR, and the number of pauses. Conclusions are drawn in the last section.

CHAPTER II

LITERATURE REVIEW

This thesis touches upon three research domains:

1. Video streaming over the Internet,
2. IP-Multicast,
3. SDN.

In the following section we review previous works in aforementioned domains.

2.1 Streaming video

With the explosive growth of video applications over the Internet, many approaches have been proposed from different perspectives to stream video effectively over best-effort networks in a way that simultaneously maximizes the display quality at the receiver, meets bit-rate limitations, and satisfies latency constraints. All systems, therefore, require efficient compression, some form of rate scalability, and error-resiliency techniques. Among all proposed solutions, effective source coding techniques can dramatically enhance video streaming quality.

Scalable Video Coding (SVC) [10] and MDC are common approaches of video coding for streaming video over the Internet but each of them are designed to cover special objectives. SVC has been proposed for heterogeneous networks where links bandwidth are different from each other while MDC is designed to alleviate effects of unreliable video transmission conditions. Loss of compression efficiency and the transmission overhead are the major drawbacks of MDC in comparison to SVC while robustness of MDC against packet loss (even burst packet loss) and simplicity of utilizing path diversification by distributing the descriptions are important advantage of that over SVC. Hence, some studies [11–14] try to combine both Layer Coding LC and MDC simultaneously to benefit from advantageous of each LC and MDC approach and at the same time to avoid the individual shortcomings of these source coding techniques.

As a side note, in the following sections the description of LC is not included because it is out of scope of this thesis.

2.1.1 Multiple Description Coding

MDC has been proposed as an alternative to layered coding for streaming over unreliable channels. An Multiple-Description (MD) coder generates multiple streams (referred to as descriptions) for the source video. A simple implementation of MD coding can be achieved by splitting even and odd numbered frames. Advanced methods include interleaving of sub-sampled lattice, MD scalar quantization, and MD transform [15]. The descriptions are then distributed over multiple paths, preferably disjoint, to enhance robustness and to accommodate user heterogeneity. In order to decode the media stream, any description can be used, however, the quality improves with the number of descriptions received in parallel. Since an arbitrary subset of descriptions can be used to decode the original stream, network congestion or packet loss (which are common in best-effort networks such as the Internet) will not interrupt the stream but only cause a temporary loss of quality. The quality of a stream can be expected to be roughly proportional to data rate sustained by the receiver.

Besides increased fault tolerance, MDC allows for rate-adaptive streaming by sending all descriptions of a stream without paying attention to the download limitations of clients. Receivers that can not sustain the data rate only subscribe to a subset of these streams, thus freeing the content provider from sending additional streams at lower data rates. [16–22] study multiple description coded video over unreliable channel and propose different ways to compensate errors occurred in these channels.

2.1.2 Path Diversity

Path diversity is a robust mechanism to overcome the effects of transmission errors and data loss in the quality of multimedia streaming applications. If the network is congested along single fixed route between the receiver and the sender, video streaming suffers from high loss rate and jitter. Even if there is no congestion, as the round-trip time between the sender and the receiver increases, the TCP throughput may reduce to unacceptably low levels for streaming applications.

Path diversity can provide several types of benefits depending on how the paths are used. A straightforward benefit of multiple paths is increased bandwidth available by using all paths at once. A complementary benefit is load balancing, by decreasing per-path bandwidth by splitting a stream across multiple paths. Another benefit of path diversity is reduce variability of packet losses, e.g., reduced excursions between periods of no loss and high loss that are common on the Internet. The end-to-end application sees the average network behavior across the paths, which generally has reduced variability [23].

Path diversity also reduces the length of burst losses (i.e. losses of consecutive packets). Distributing packets across multiple paths increases the interpacket spacing on each path, and therefore for a network congestion event of a given duration fewer packets are lost. Reducing burst losses provides a number of benefits for media streaming. For example, for video it is easier to recover from multiple isolated losses than from an equal number of consecutive losses. For two paths with equal average packet loss rates, sending even packets on one path and odd packets on the other has no effect on the end-to-end loss rate but does reduce burst losses.

Path diversity idea historically introduced by [24] where author proposes to send complementary descriptions of a MD coded video through two different Internet paths, as opposed to the default scenario where the stream of packets proceeds along a single path. In [25, 26] the authors employ path diversity in the context of video communication using unbalanced MD coding to accommodate the fact that different paths might have different bandwidth constraints. In [27] the authors study image and video transmission in mobile radio networks. It is shown that combining MDC and multiple path transport in such a setting provides higher bandwidth and robustness to end-to-end connections. In [28] a framework for video transmission over the Internet is presented, based on path diversity and rate-distortion optimized reference picture selection. Here, based on feedback, packet dependency is adapted to channel conditions in order to minimize the distortion at the receiving end, while taking advantage of path diversity. In [29], authors propose a routing-aware MDC approach with path diversity to enhance the error robustness of video transmission over wireless ad-hoc networks. By using the routing messages as a packet loss indicator, they dynamically select the reference frames for MDC to reduce the error propagation. Proposed method does not require any additional feedback channel or extra overhead, however, the packet loss information provided by the route messages is not completely accurate.

A number of recent papers have addressed the problem of selecting optimal paths for MDC video streaming hence, the received media quality is directly affected by link quality metrics. [30] investigates how to assign bandwidth to each description in order to maximize overall user satisfaction by formulating it as an optimization problem. [31] models multi-path streaming and propose a multi-path selection method that chooses a set of paths maximizing the overall quality. The simulation results show that the average PSNR improves if source video packets are routed over intelligently selected multiple paths. In [32], the authors propose a new QoS metric, link and path correlation model for multi-path selection problem. [33] shows that mesh-based Peer-to-Peer (P2P) when combined with MDC results in improvement in delivered video quality making it acceptable for ad-hoc networks.

2.1.3 Distributed Video Streaming

Having multiple senders is in essence a diversification scheme in that it combats unpredictability of congestion in the Internet. Multiple sources achieve when video content is distributed among multiple streaming sources. Video streaming using distributed server mainly studied in [34]. Here authors proposed a framework for streaming video from multiple mirror sites simultaneously to a single receiver in order to achieve higher throughput, and to increase tolerance to loss and delay due to network congestion. Their work is later continued in [35–38].

Multiple sources can be provided through framework offered by overlay networks such as P2P, mesh, and specially edge architecture like CDN architectures. When aforementioned overlay frameworks properly combined with adaptive encoding solutions such as MDC the definition of multiple end-to-end paths from the server to the client can dramatically improve the quality of service. In [39] the performance of path diversity and multiple description coding in CDN is studied. 20-40% reduction in distortion is reported over conventional CDNs for the network conditions and topologies under consideration. [9] presents and tests three MD coding schemes over CDN. Results shows that for CDN the better improvement on ratio distortion properties can be achieved using the schemes of decomposing the odd-and-even frame in the multiple description coding. Video streaming using server diversity and MD video coding are widely studied in [40–44].

In this thesis, we propose a framework for streaming video from multiple mirror sites while video content is divided to odd and even descriptions distributed between servers. Therefore, implemented system exploiting path diversity by utilizing distributed video streaming system and MDC helps us to fairly distribute the content among the providers. Implemented system provides higher throughput, and increases tolerance to loss and delay due to network congestion.

2.2 Multicasting

IP-Multicasting [45] is the ability of a communication network to accept a single message from an application and to deliver copies of the message to multiple recipients at different locations. Although this can be done by sending different unicast (point-to-point) messages to each of the destination hosts, there are many reasons which make having the multicasting capability desirable. A schematic representation of the multicasting model can be seen in Figure 1.

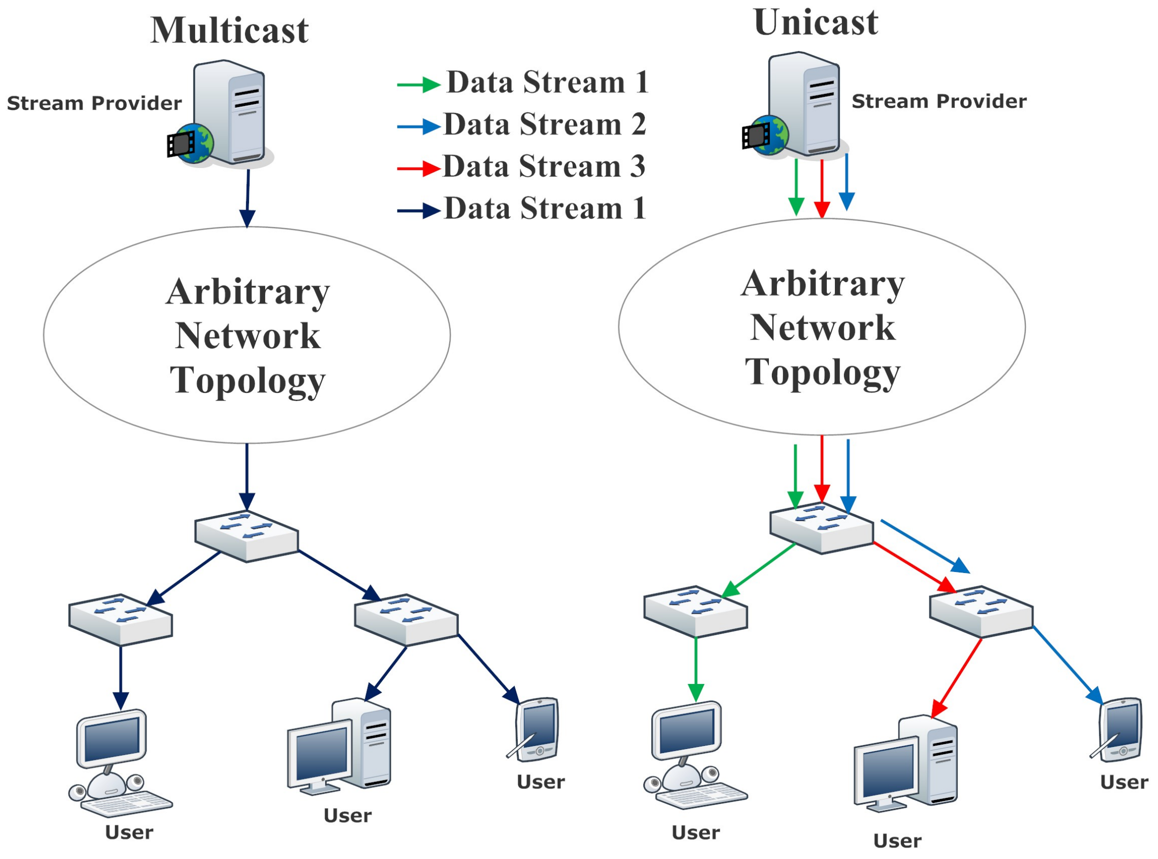


Figure 1: Multicasting vs Unicasting

The first major advantage of using multicasting is the decrease of the network load. Since multicasting requires the transmission of only a single packet by the source and replicates this packet only if it is necessary (at forks of the multicast delivery tree), multicast transmission can conserve the so much needed network bandwidth. An efficient implementation of multicasting permits much better use of the available bandwidth by transmitting at most one copy of the data (i.e. stream data) on each link in the network. Another important feature of multicasting is its support for data casting applications. In recent years, multimedia transmission has become more and more popular. The audio and video signals are captured, compressed and transmitted to a group of receiving stations. Instead of using a set of point-to-point connections between the participating nodes, multicasting can be used for distribution of the multimedia data to the receivers. In real world stations may join or leave an audio-cast or a video-cast at any time. The flexibility in joining and leaving a group provided by multicasting can make the variable membership much easier to handle. There has been an explosion of research literature on multicast communication.

2.2.1 Multicasting in Today's Networks

Current Internet supports both unicast and multicast capability however unicast is the most prevalent mode of communication with multicast being almost non-existent. While many researchers have advocated a wide deployment and use of multicasting in the Internet, a large number of researchers along with the Internet Service Providers (ISPs) have largely remained skeptical about its real benefits. While researchers have been mostly concerned about the added complexity that multicasting will introduce in the core Internet, ISPs are concerned more with the current payment model of the Internet. Presently, ISPs charge users based upon the amount of link bandwidth they use; in such a model, allowing multicasting may reduce the bandwidth usage by individual users, thus the revenues of ISPs may decline. Clearly, in order for the ISPs to have enough motivation to deploy multicasting capability, the payment and service model of the Internet usage needs to be changed.

There are many other complicating factors, which generally dilutes the benefits of multicasting in the current Internet. For instance, it is generally accepted that a large fraction of the total Internet bandwidth is consumed in one-to-one communication. Thus, if only 1% of all Internet communication operates in a one-to-many mode, then there will be little incentive in deploying the multicasting capability. The situation becomes even worse, because the one-to-many mode of communication is generally limited within small local networks, e.g., sending emails to all users in a company. There are very few instances where one-to-many communication occurs with destinations distributed across the world. One can argue that the situation is similar to the chicken and egg problem; since ISPs do not support multicasting, there are few services which need them, and since there are few services which needs them, ISPs do not have enough motivation to implement multicast.

With the introduction of the Overlay networks, the situation has changed considerably. Now, there are many applications and services, which use ALM over an Overlay network. These services, usually build a multicast tree with the source at the root of the tree and the destinations at the leaves. The intermediate nodes which are essentially the overlay nodes are arranged in such a way that the overall bandwidth usage is minimized. With the introduction of these Overlay based applications, multicasting has received a new push, and now many argue that ISPs should also deploy the multicasting capability in their network.

While these Overlay networks have demonstrated that multicasting can be useful for a wide variety of services, there are yet many technical barriers. First and foremost, the complication arises due to the presence of hundreds of independent ISPs in the current Internet, and due to their conflicting interests. Each ISP wants to provide the best possible service to its own customers, and generally they do not have enough motivation to improve the service received by its non-customers. In such a scenario, an efficient multicast routing which requires construction of multicast trees from source to all destinations can become problematic. Moreover, if a small number of ISPs do not allow multicasting, then it may become problematic to implement multicasting capability by other ISPs. Another problem is that ISPs usually do not advertise the topology of their network to other ISPs, and especially to all sources; without such information, it may become difficult to construct efficient multicast trees.

With these technical barriers, there are other issues too, which generally arise due to the dynamics of today's Internet usage. CDN, Video on Demand (VoD), Video Conferencing and many other innovations are used to be responder for such a huge amount of requests which are sent through Internet. It clearly appears that multicasting is required for these services but with so many barriers and problems at hand such as:

- Resistance by the ISPs,
- Resistance by the content providers,
- Natural shift away from the one-to-many communication mode (due to the introduction of CDNs),
- Relatively few services for which multicasting is essential,
- Amount of difficulties to implement it is unlikely that multicasting will become a popular mode of communication in the Internet.

Because implementing of multicast faces some fundamental problems, thinking about live migration and having distributed multicast servers are more like dreams.

2.3 Software-Defined Networking

SDN, leverages a centralized, logical view of the network that can be easily manipulated via software to implement complex networking rules. The resulting benefits include support for multi-vendor environments, more granular network control (at session, user and device levels), improved automation and management, accelerated service deployments and unprecedented scalability and flexibility at lower cost. In the simplest possible terms, SDN entails the decoupling of the control plane from the forwarding plane and offloads its functions to a centralized controller [46]. Rather than each node in the network making its own forwarding decisions, a centralized software-based controller (likely running on commodity server hardware) is responsible for instructing subordinate hardware nodes on how to forward traffic. Because the controller effectively maintains the forwarding tables on all nodes across the network, SDN-enabled nodes do not need to run control protocols among themselves and instead rely upon the controller to make all forwarding decisions for them. The network, as such, is said to be defined by software running on the controller. SDN structure is depicted in Figure 2.

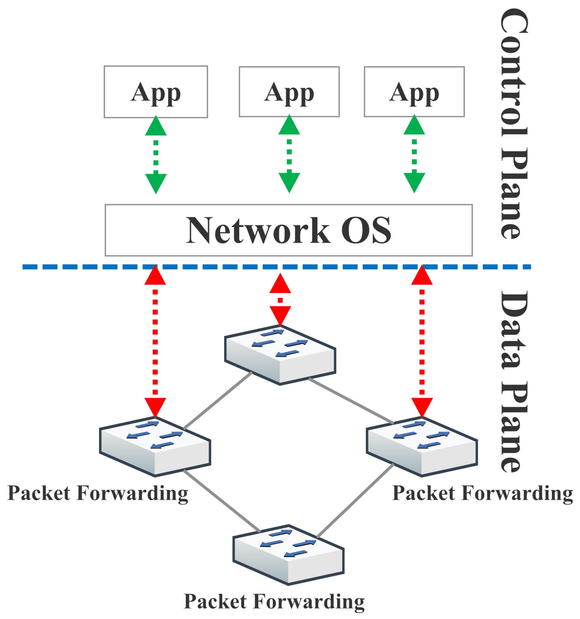


Figure 2: Software-Defined Networking Structure

In this thesis we used OpenFlow Standard which is the first standard communications interface defined between the control and forwarding layers of a software-defined network architecture. Capitilizing on the benefits that SDN bring, there have been a number of studies on SDN video streaming in the literature.

Authors in [47] benefits from central view provided by SDN and propose multimedia delivery with end-to-end QoS. According to this idea flows are divided to multimedia and data categorize where multimedia flows are routing to special QoS routing method while data flows are routing using shortest-path. Using the same idea authors in [48, 49] benefit from SDN to differentiate routing policies for base layer and enhancement layers. As investigated before in [16–19], base layer is guaranteed to be delivered correctly while complement layers are routed either as lossy-QoS or best-effort flow. Civanlar et al. [50] have described an optimization model to improve packet routing. Such model considers delay and packet loss. The optimization model, through the linear programming, computes a QoS path for video traffics and a shortest path to best-effort traffics. [51] presents an SDN-enabled content-based routing framework where Youtube flows are identified via Deep Packet Inspection (DPI) and is always forwarded via least congested links.

There have also been a number of studies of multicasting over SDN. In [52] an innovative way of managing IP-Multicast in overlay networks is proposed. The authors propose using OpenFlow instead of Internet Group Management Protocol (IGMP). The main contribution is to eliminate periodic join/leave messages and use of multipath in the layer-2 network. In [53] authors propose a scalable network-layer single-source inter-domain multicast framework by making use of an Locator/ID Separation Protocol (LISP) router overlay. In [54] the authors propose an IP-Multicast-based forwarding system optimized for fast recovery in case of path failures. For each multicast group, the controller calculates two different multicast trees spanning all switches of the network. If a switch fails, the controller disables the currently used tree and enables the complementary tree, which is likely unaffected by the failure. [55] proposes a clean-slate approach for multimedia multicasting, where routes between the source and all of the subscribers are computed a priori, with the purpose of speeding up the processing of multicast events over SDN framework.

CHAPTER III

MULTICASTING OVER SDN

The streaming video multicast framework presented herein is composed of two distinct parts: i) Streaming Video Multicast Application, and ii) SDN Controller and the Streaming Video Multicast Control Application running on it. Figure 3 illustrates a preview of implemented streaming video multicast framework.

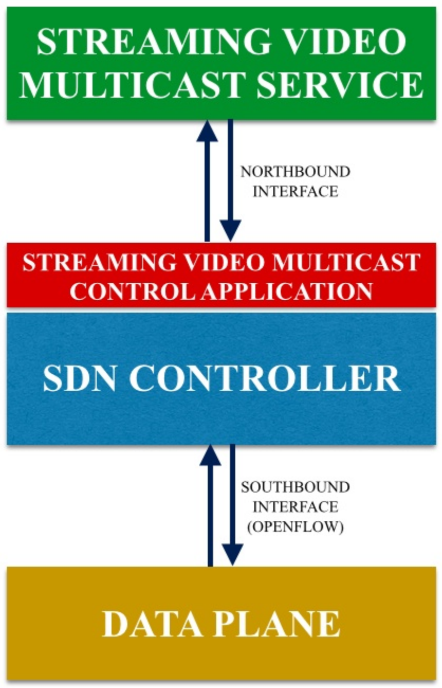

STREAMING VIDEO MULTICAST SERVICE
NORTHBOUND INTERFACE
STREAMING VIDEO MULTICAST CONTROL APPLICATION
SDN CONTROLLER
SOUTHBOUND INTERFACE (OPENFLOW)
DATA PLANE

Figure 3: Streaming Video Multicast Framework

Here, the Streaming Video Multicast Service maintains the identities and locations of the active servers and the corresponding descriptions they are multicasting. The application also maintains the up-to-date list of subscribers that are allowed to receive the service. The Streaming Video Multicast Control Application running on the SDN controller is responsible for selecting the description(s) for each subscriber, establishing the corresponding route and maintaining the multicast tree for each description.

Now, let us describe the individual blocks in more detail. The Streaming Video Multicast Service needs to ensure that the SDN controller has up-to-date information regarding both the subscriber and the video server identities. The SDN controller in return, needs to update the multicast service on whether a given subscriber has joined or left the multicast. Implemented system is designed to handle IP-Multicast when both provider and customer are in the same network where a single controller spans across.

3.1 Streaming Video Multicast Control Application

Provided that the identities of the subscribers and the servers are known at the controller, the control application needs to map clients with servers, compute routes and multicast trees for all clients and servers, respectively.

When constructing the multicast trees, two distinct optimization strategies could be considered when a subscriber wants to join the multicast group:

1. Minimize service impact on the network load

2. Maximize average streaming video quality

While the first strategy results in finding the server for which the addition of the subscriber to its multicast tree would result in the least number of additional branches in the tree, the second strategy finds the server that provides the highest QoE to the subscriber. In this thesis we consider the second strategy.

3.1.1 Routing Algorithms

In our implementation routing algorithms identify the best DP for subscribers. When a subscriber sends a Join message, multicast control application applies routing algorithm from each DP to the given client and save the selected path in a list (if there is no path from a DP to the given client, no path saves in the list). At the end, all paths compare with each other and the DP that satisfies the requirements of the algorithm is selected. For example if packets are routing by minimum hop strategy, a DP will be selected among all DPs which has the lowest number of hops to the new subscriber.

Three different routing algorithms are considered in this thesis. Figure 4 illustrates the difference between implemented routing algorithms over a scenario. In this scenario all links bandwidth are equal to each other and values beside each link demonstrate link congestion level (lower number means the link is less congested).

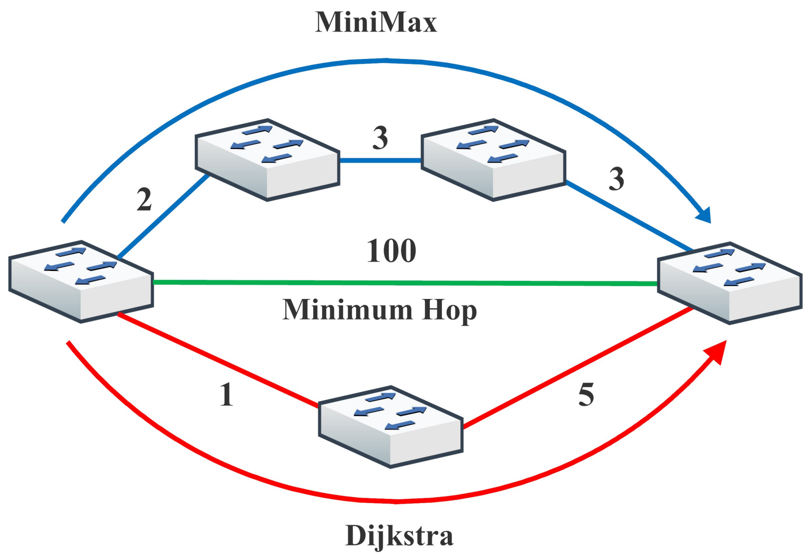


Figure 4: Routing algorithms used for routing multicast packets

1. Minimum Hop: This algorithm simply counts the number of hops from specific source to specific destination and selects the path with the smallest such number. Although green line is fully congested but it selects by Minimum Hop routing algorithm since it identifies lowest number of hops between source and destination.

2. Shortest Path (Dijkstra): The algorithm finds the path with lowest cost (i.e. the shortest path) between the given vertex and every other vertexes. In this

thesis we used Shortest Path algorithm for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined. Dijkstra selects red path for routing packet from source to destination hence it has lowest total cost among other paths (green line congestion level is 100, blue line congestion level is 8 and red line congestion level is 6).

3. MiniMax: The algorithm finds an end-to-end path between two Internet nodes that minimizes the maximum weight of any of its edges. MiniMax algorithm chooses the blue path because each link in this path is less congested than all other links in alternative paths.

Once a server is selected for a subscriber and associated route is computed, the multicast control application adds this user to the corresponding multicast tree. All pre-existing links in the computed route are utilized via IP-Multicast, and the new, necessary links are added to the tree to ensure video delivery. This operation is repeated for every new subscriber.

3.1.2 Link Status Discovery

MiniMax and Dijkstra routing algorithms select the path according to link weights. Weights to the links are dynamic and are based on the traffic load they endure. In the proposed system, these weights are updated periodically to ensure good performance. For this purpose a separate thread is implement in multicast control application which is responsible for creating a topology graph, finding the connection between nodes and assigning weight to the links.

The controller discovers the topology by sending specially crafted Link Layer Discovery Protocol (LLDP) frames between the (virtual) switches. By default LLDP packets are processed by TopologyManager module in the controller and then are discards. We asked the controller to pass LLDP packets to both TopologyManager and MulticastManager and then discard the LLDP packets. Therefore, in multicast controller we have enough information to create a topology graph according to switches and their links. In addition to aforementioned thread which is creating network topology graph, another thread is required to assign weight to links and update them periodically during the time that simulation is running.

To compute the link weights dynamically, multicast controller launches a thread which periodically (In our implementation every 10 seconds) queries switches on their port statistics using STATISTICS_REQUEST and parse switches response over the request. The switches reply by the appropriate feedback in form of STATISTICS_REPLY. Figure 5 illustrates designed and implemented system for this purpose.

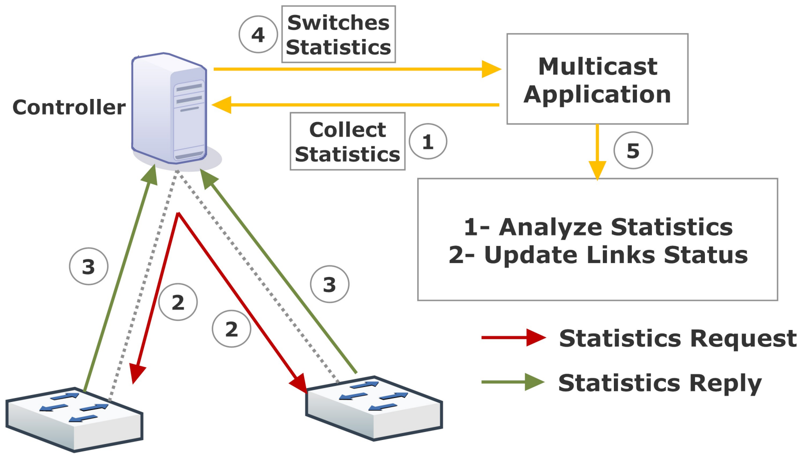


Figure 5: Implemented system to record statistics

The port statistics include the amount of received and/or transferred data. The controller then takes the average of N previous statistics to determine a given link weight (N=10 is used in our implementation). However having a good performance requires decreasing the time period for querying the switches, which incurs a higher network traffic and higher computational cost. When the network is not congested, decreasing the time period for querying the switches is endurable and they can process the request, however, when the network is fully congested, switches are incapable to route network packets and processing statistics request is additional trouble. Therefore, we set 10 second as a time interval between two consecutive query to not impose unbearable pressure over the switches when the network is fully congested. Obviously this time period ought to be the same during all simulation to observe fairness whether the network is congested or not.

3.2 Streaming Video Multicast Service

The streaming MDC-based video multicast servers are referred to description providers (DPs). A newly launched DP sends a packet in multicast IP range containing information about its description which is always forwarded on to the controller. The multicast control application creates and stores a distinct multicast tree in the form of a data structure per DP. As soon as the source message arrives at the controller from a new DP, the control application establishes a new tree for that DP. Subsequently, the new DP is added to the list of available DPs so that for a new subscriber (or for an update for a current subscriber), when joint DP selection and routing is computed, this description is also considered.

At any given time, a DP may experience a failure due to: i) Crash/Shut down and ii) Disconnection from connected switch. As soon as a DP is out of service due to one of above mentioned reasons either a proactive or a reactive solution may be developed for this scenario. For a proactive solution one of the following procedures may be implemented:

1. A back-up server may be made available for potential failures,

2. An alternate DP and associated multicast tree may be constructed for every subscriber a priori for fast tree switching.

For a reactive solution on the other hand, one of the following procedures may be implemented:

1. A new DP may be selected randomly after the failure is observed,
2. The best DP is computed for each client at the time of failure.

Both proactive and reactive methods have benefits and ill effects. Although proactive approach seems as an ideal solution to the problem of DP failure but its benefits achieve at the cost of computation overhead to find back-up server for clients.

In this thesis, we consider a reactive approach where the new best DP is selected upon the failure of the existing one.

3.3 Subscribers

Subscribers join or leave the multicast streaming video service at any time via Join/Leave messages. When a new subscriber is to be added to the multicast tree, the control application conducts the following sequential procedure:

1. It first checks whether the subscriber is already being served,
2. If not, it then checks whether the subscriber is to be served via communication with the multicast service,
3. Based on the routing algorithm in use, it selects the best DP for it,
4. It adds the subscriber to that DP's JTree data structure,
5. It computes the necessary additional ports and/or branches to the multicast tree,
6. It pushes the corresponding forwarding rules to the switches using OpenFlow.

When a new subscriber joins a multicast tree, one of two scenarios may take place:

1. Joining the multicast tree may involve just the addition of a packet duplication rule to one switch in the network,

2. Joining the multicast tree might involve adding new switches and links to the multicast tree, in which case, rules for all affected switches are pushed.

Figure 6 describes the above scenarios. In the first scenario controller asks subscriber immediate switch (switch 4) to replicate packet on specific port. In the second scenario subscriber receives video stream when controller asks several switches (switches number 2, 3, 5) to amend their flow tables.

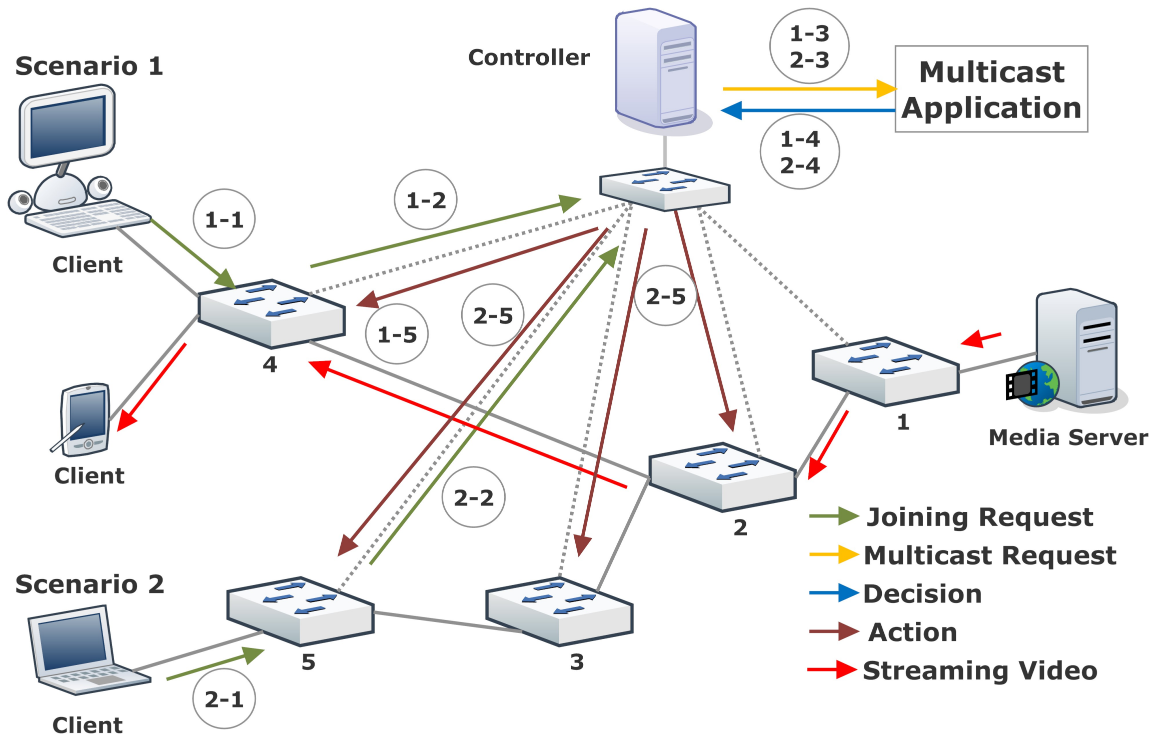


Figure 6: Procedure of client joining

Subscribers may leave their multicast group politely or impolitely. When the leave is polite, the subscriber informs the multicast group a priori, but when it is impolite, the subscriber may leave without any a priori notification. In our framework, a user leaves its multicast group as a result of i) Crash/Shut down ii) Disconnection from connected switch iii) Service Leave message. Figure 7 illustrates leaving procedure for two mentioned scenario.

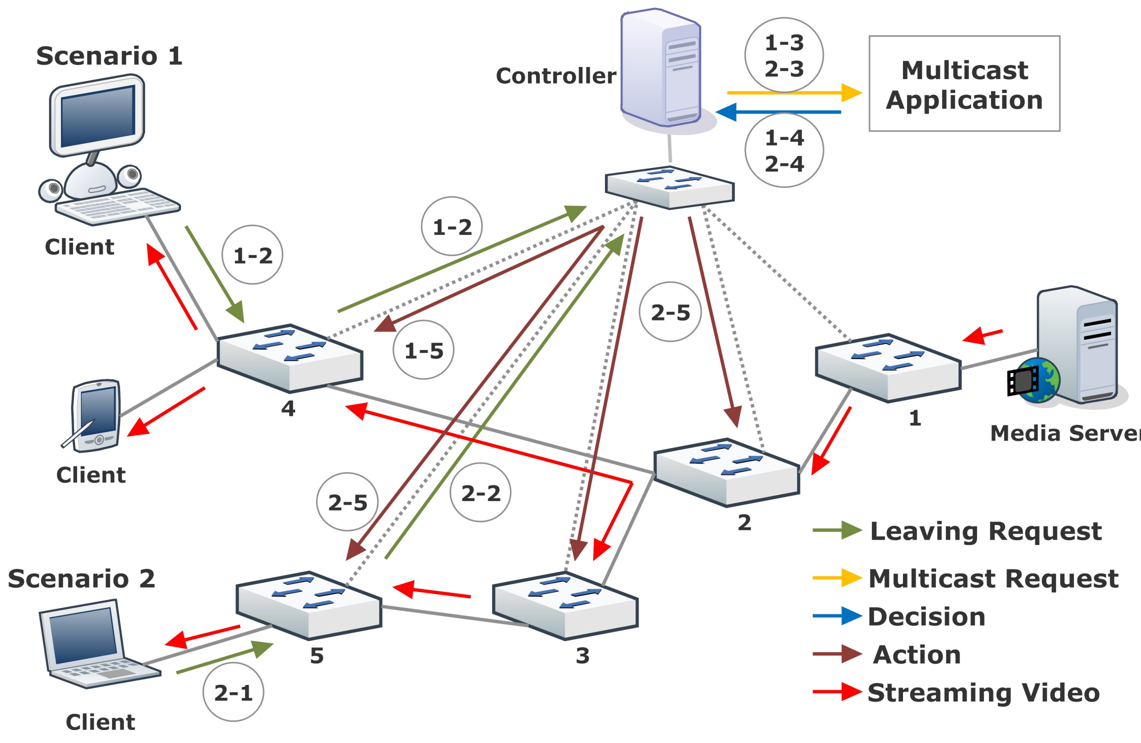


Figure 7: Procedure of client leaving

Similarly, when a subscriber leaves the service, the control application conducts the following sequential procedure:

1. It first checks whether the subscriber is being served,
2. If so, it then removes the subscriber from its serving DP's JTree data structure,
3. It then removes the port and/or switch and link from multicast tree,
4. It pushes the corresponding forwarding (expiration) rules to the switches using OpenFlow.

Similar to the subscriber join case, one of two scenarios may take place when a subscriber leaves the service:

1. Leaving the multicast tree may involve just the removal of the port of a switch from it (first scenario in Figure 7),

2. Leaving the multicast tree may involve removal of a switch and link from it (second scenario in Figure 7).

In the proposed multicast service, it is possible for a subscriber to migrate from one DP to another. The main purpose for this migration is to increase user satisfaction from the service. Due to the dynamic nature of the network, it is possible that the DP which was chosen as the best provider for a subscriber is no longer suitable. For this purpose, a separate thread periodically checks each client's best serving DP. If the current DP for one of the clients is no longer the best, the subscriber first leaves and then rejoins the service following the procedures outlined above.

CHAPTER IV

PROPOSED MULTICAST ARCHITECTURE PERFORMANCE

In this chapter we will compare the performance of our proposed IP-Multicast system with different possible implementation of common approaches of IP level multicasting in today's network.

4.1 Traditional Multicast Routing Protocols

Protocol-independent multicast (PIM) is a set of four specifications that define modes of Internet multicasting to allow one-to-many and many-to-many transmission of information. It is termed protocol-independent because PIM does not include its own topology discovery mechanism, but instead uses routing information supplied by other routing protocols. The family of PIM protocols includes dense-mode (DM) [56], sparse-mode (SM) [57], source specific multicast (SSM) [58], and bidirectional (Bidir) PIM [59]. The initial set of protocols only included dense-mode and sparse-mode, but after a few years of deployment experience, the protocols have evolved and been optimized to better support the emerging multicast applications. The traditional PIM protocols (DM and SM) provided two models for forwarding multicast packets, source trees, and shared trees. Source trees are rooted at the source of the traffic while shared trees are rooted at the rendezvous point (RP). Each model has its own set of characteristics and can be optimized for different types of applications. The source tree model provides optimum routing in the network, while shared trees provide a more scalable solution. Dense mode is ideal for groups where many of the nodes will subscribe to receive the multicast packets, so that most of the routers must receive and forward these packets.

Four variants of PIM are as follows:

1. **PIM Dense Mode (PIM-DM)**: (PIM-DM) implicitly builds shortest-path trees (SPT) by flooding multicast traffic domain wide, and then pruning back branches of the tree where no receivers are present. Pruning mechanism is using as an attempt to optimize the data flow for long lived conversations in order to avoid sending the data into portions of the Internet with no receivers for G. If a router receives a packet for G, and has nobody to forward it to, it sends a "prune" message for G to the neighbor from which it received the packet for G. Each router is responsible for keeping track of all the groups their neighbors are not interested in listening to. PIM-DM is straightforward to implement but generally has poor scaling properties. It is infeasible to flood traffic for all groups everywhere, in case some distant node would like to listen, and it is infeasible (if multicast were successful) to keep state for all groups each neighbor is not interested in receiving.

2. **PIM Sparse Mode (PIM-SM)**: In SM operation, data packets are not broadcasted and only routers on the multicast tree need to keep state information for a group. The state of a multicast tree is set up when receivers designated routers (DR) send join messages toward a RP. PIM-SM is built around a single, unidirectional shared tree whose root is the RP. The RP knows all the receivers and all the sources and make a connection between both therefore avoiding the flood and prune behavior of PIM DM.

   One of the receiver's local routers is elected as the DR for that subnet. On receiving the receiver's expression of interest, the DR then sends a PIM Join message towards the RP for that multicast group. This Join message is known as a (*, G) Join because it joins group G for all sources to that group. The (*,

G) Join travels hop-by-hop towards the RP for the group, and in each router it passes through, multicast tree state for group G is instantiated. Eventually, the (*, G) Join either reaches the RP or reaches a router that already has (*, G) Join state for that group. When many receivers join the group, their Join messages converge on the RP and form a distribution tree for group G that is rooted at the RP. This is known as the Rendezvous Point Tree (RPT), and is also known as the shared tree because it is shared by all sources sending to that group.

A multicast data sender just starts sending data destined for a multicast group. The sender's local router (DR) takes those data packets, unicast-encapsulates them, and sends them directly to the RP. The RP receives these encapsulated data packets, decapsulates them, and forwards them onto the shared tree. The packets then follow the (*,G) multicast tree state in the routers on the RP Tree, being replicated wherever the RP Tree branches, and eventually reaching all the receivers for that multicast group.

3. **Bidirectional PIM (BIDIR-PIM)**: Bidir PIM was developed to help deploy emerging communication and financial applications that rely on a many-to-many applications model. In bidirectional mode, traffic is routed only along a bidirectional shared tree that is rooted at the RP for the group. Data from the source can flow up the shared tree (*, G) towards the RP and then down the shared tree to the receiver. There is no registration process and so source tree (S, G) is created. Bidirectional trees are built using a fail-safe Designated Forwarder (DF) election mechanism operating on each link of a multicast topology. With the assistance of the DF, multicast data is natively forwarded from sources to the RP and hence along the shared tree to receivers without requiring source-specific state. The DF election takes place at RP discovery time and provides the route to the RP, thus eliminating the requirement for data-driven protocol

events.

4. **PIM Source-Specific Multicast (SSM)**: Mainly, SSM was developed to easily support one-to-many applications by greatly simplifying the protocol mechanics for deployment ease. In SSM, delivery of datagrams is based on (S, G) channels. Traffic for one (S, G) channel consists of datagrams with an IP unicast source address S and the multicast group address G as the IP destination address. Systems will receive this traffic by becoming members of the (S, G) channel. In both SSM and Internet Standard Multicast (ISM), no signaling is required to become a source. However, in SSM, receivers must subscribe or unsubscribe to (S, G) channels to receive or not receive traffic from specific sources. In other words, receivers can receive traffic only from (S, G) channels to which they are subscribed, whereas in ISM, receivers need not know the IP addresses of sources from which they receive their traffic.

4.2 Performance Comparison

In order to have a fair performance comparison and as first step we ought to do all simulations over a system similar to SDN controller which has a comprehensive view over the network topology. For this purpose we have developed a system which is designed to parse a network topology and creates its corresponding graph. Then, we distributed pairs of servers and 10 multicast clients over created graph and finally, we evaluated the output and performance of different implementation of SM, DM, Dijkstra and MiniMax algorithms for given number of multicast requests at the specific time. Location of multicast servers has significant impact on the performance of multicast trees which will be created.

Overall performance comparison of all implemented systems is tabulated in Table 1. These results have been obtained by running each implemented system over 133 different topologies with at least 25 switches. Servers, clients and rendezvous points are distributed randomly through the topologies. In addition, links in each topology are weighted randomly proportional to the number of switches that a given topology contains.

The following metrics are investigated:

1. Average number of branches: This value shows the average number of branches that each multicast tree contains. The lower average value might show that better multicast trees are established although, this fact it is not certain. The worst multicast tree is established when there is no common point between clients and multicast packets traverse a distinct path per each client and the established multicast tree operates exactly as unicast where a distinct flow is sent per client.

2. Average tree load: This value shows the average weight of links which constitute the multicast trees. Theoretically lower value shows that created multicast trees consist of less congested links and consequently a lower probability of packet loss is expected. If the multicast tree is established over high weighted edges, the probability of packet drop/loss increases automatically.

3. Average client load: This value shows the average link weight from a given client to its multicast content provider. Similar to the two aforementioned metrics, the lower value is ideal theoretically because it shows that clients have more chance to receive multicast packets correctly.

4. Time complexity: This value quantifies the amount of time taken by implemented algorithms. Lower complexity leads to great reduction of computational complexity and subsequently a faster reaction from controller to multicast group events.

5. Space complexity: This value shows total space taken by the algorithm with respect to the input size.

Lower value for the first three metrics would not lead to creation of optimized multicast tree essentially. As we described earlier, MiniMax routing algorithm finds the path which has more available bandwidth. On one hand obtained multicast tree using this algorithms is guaranteed to have more available bandwidth and on the other hand established tree might not have a low client load, tree load or number of branches average. Regardless of values for tree and client metrics, established tree using MiniMax is guaranteed to have a good performance because having more available bandwidth (capacity) means higher chance to receive end-to-end packets.

Time and space complexity can be also interpreted as metric of scalability for an algorithm. The algorithm which has the lower order of time and space complexity is more scalable.

Table 1: Performance comparison of different multicast implementation models

Multicast Methods	Branch Average	Tree Load Average	Client Load Average	Time Complexity	Space Complexity
Traditional Sparse	28.8	974.5	455.1	O(V^3)**	O(V)
Traditional Sparse + BFS	13.7	398.8	104.0	O(V^3)**	O(V)
Sparse + no RP	6.5	193.6	85.8	O($C \times (E+V)$)	O(V)
Sparse + Load Balancing + no RP	7.5	220.6	83.3	O($C \times (E+V)$)	O(V)
Dense + BFS	7.3	215.4	64.2	O(E+V)	O(V)
Dense + Kruskal	10.0	150.5	66.3	O($N_{lu} \times ElogV$)	O(V)
Dijkstra	8.2	148.4	47.2	O($N_{lu} \times S \times (ElogV)$)	O($S \times (V+E)$)
MiniMax	10.1	145.8	68.3	O($N_{lu} \times S \times (ElogV)$)	O($S \times V$)

Where:

- S defines the number of multicast servers (1 < S < finite value). Finite value is proportional to the number of nodes in a given topology,
- C defines the number of multicast clients ($1 < C < \infty$),
- N_{lu} defines the number of link weight update per minute ($1 < N_{lu} < 60$),
- V defines the number of graph vertices (switches in the topology) ($1 < V < \infty$),
- E defines the number of graph edges (links between switches) ($0 < E < V^2$),

- ** defines that no fast solution is known for the original problem (NP Complete) but the heuristic solution [60] solves this problem within the time complexity as indicated in Table 1.

Due to the tradeoff between the parameters, optimizing each parameter would result in other ones being non-optimal. In general, the algorithm which has an acceptable value for all of the parameters could be selected as an ideal approach for multicast implementation over the network structures like SDN.

4.2.0.1 Implemented methods

Each of the implemented algorithms has their own set of characteristics. We will describe each method in detail and will demonstrate their different behaviors over a sample topology which is shown in Figure 8. Nodes colored red/purple shows corresponding server's switch, node which is colored in green shows the RP and nodes colored in orange demonstrate corresponding client's switches. Applying each of the aforementioned algorithms leads to creation of different multicast trees.

Traditional sparse mode: This algorithm follows the traditional implementation of sparse mode where clients express their interest by sending a Join message to its Designated Router (DR). On receiving the receiver's expression of interest, the DR then sends a PIM Join message towards the RP for that multicast group. Eventually, the Join request either reaches the RP or reaches a router that already joined that group. This scenario could be implementing by using Depth First Search (DFS) algorithm and extracting the path from the given client to the switch in multicast group. As shown in Table 1 neither the multicast tree parameters nor multicast parameters are optimized.

In addition, the main problem concerning the construction of a shared multicast tree is selection of a root of the shared tree or the core point. The selection of core directly affects the performance of multicast. A poor selection may lead to many

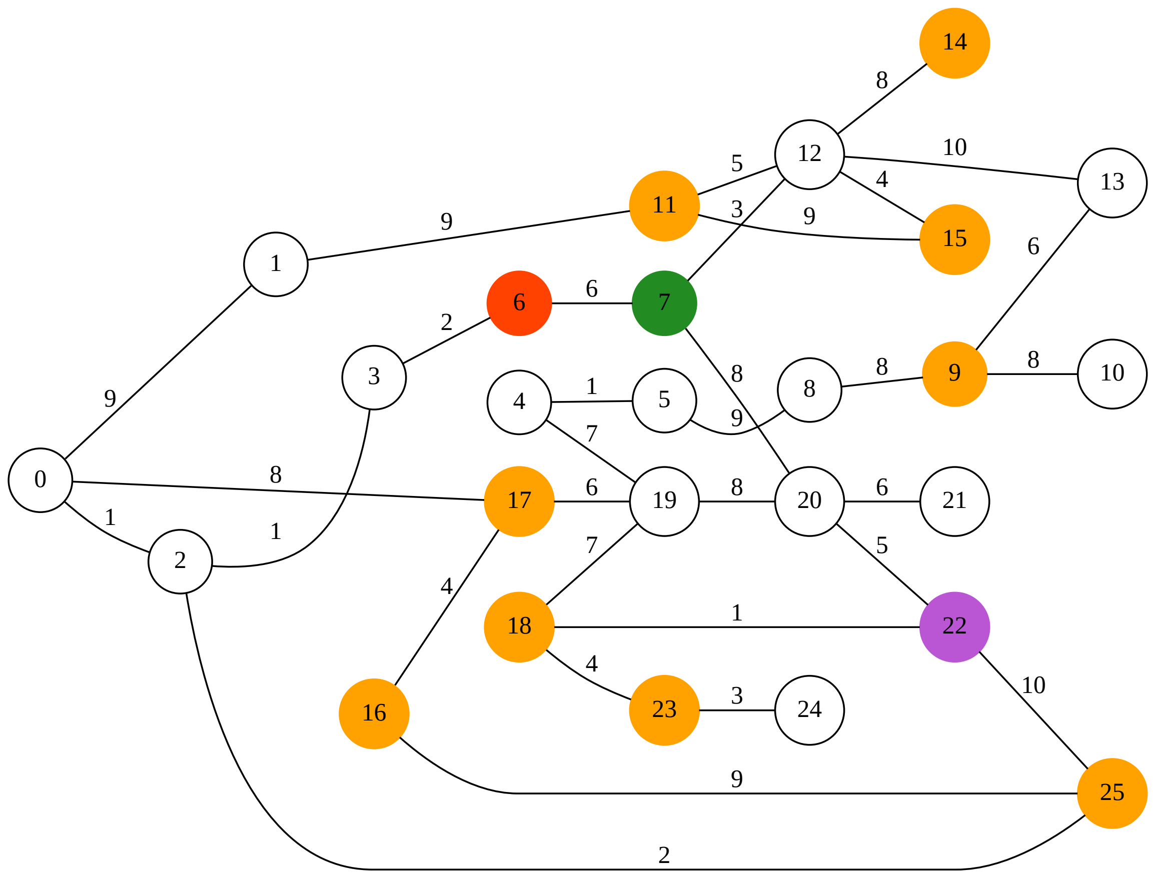


Figure 8: Network topology under consideration

performance problems such as high cost, significant delay and increased congestion. Therefore, it is very important to select a suitable core to have an effective multicast. However, the core selection is an NP complete problem which needs to be solved using heuristic algorithm. Researchers have already proposed several solutions to this problem. However, the problem of finding a better or the best core node has not yet been completely solved. Applying traditional sparse mode algorithm over illustrated topology leads to creation of following multicast tree (Figure 9).

Modified sparse mode: The modified version of sparse mode significantly increases the performance of multicasting by simply replacing the DFS with Breadth First Search (BFS). In contrast to DFS which explores as far as possible along each branch before backtracking, BFS begins at the client switch and inspects all the neighboring nodes till it reaches RP or a node which has already joined multicast group. First, client expresses its interest to join a multicast group by spreading the request on all available attachment points. Then each switch propagates the packet to all neighbor nodes till the request reaches the closest switch in the multicast tree. Although multicast parameters improve, time complexity of finding an appropriate RP affects the total efficiency of this algorithm. Applying a modified version of sparse mode algorithm over illustrated topology leads to creation of following multicast tree (Figure 10).

Sparse without RP: One method to overcome the drastic complexity of choosing the RP is to move RP to multicast servers attachment point. Transferring the RP across switches associated with multicast sources and increasing the number of RP will close the gap between the traditional definition of dense mode and sparse mode by omitting the concept of shared multicast tree, however, the procedure of joining to multicast group is still fundamentally different. Likewise sparse mode, the join request spreads across network topology using BFS until the request reaches the closest available multicast trees. Omitting the complexity of finding a RP makes this approach applicable and evaluated performance values for multicast tree and multicast client confirm the efficiency of this algorithm. Figure 11 represents the structure of multicast trees after applying this algorithm.

Sparse mode with load balancing: In aforementioned implementation of sparse mode and when multiple sources are available and located close to each other, it is highly possible to have one tree which almost supports all multicast clients while the other sources are idle. By joining each new multicast client to a specific multicast tree, that multicast tree grows and has more chance to collect further Join requests due to higher number of switches it has in comparison to other multicast trees. Consequently, over the time all new multicast request converges to one multicast tree and that multicast tree spans through whole topology while other multicast trees remain small. Therefore, a mechanism of load balancing will enhance the performance of sparse mode dramatically. In more intelligent implementation of sparse mode, a controller finds path from a given client to all available multicast providers but finally it assigns a multicast tree to a new client which satisfies the specific requirements.

In smart sparse mode we consider a weight for each multicast tree which is equal to total weight of all links in tree rooted at the server. When a client expresses its interest to join the multicast group, the controller will find the path from that given client to all sources using BFS algorithm. The tree which has the lower summation of tree weight and path length to the given client will be responsible to support that client. As a result, the traffic load prorates over all available multicast clients in expense of adding a method for tree comparison. Figure 12 illustrates obtained multicast tree structure after applying this algorithm.

Dense mode using BFS: In contrast to all aforementioned methods where finding an appropriate multicast tree starts from the client side, another approach is to start creating the multicast tree from multicast server side. Because the multicast tree creation starts from the server side, we refer to this method as a subset of dense mode. In this approach BFS algorithm starts from all available servers till all nodes in the topology will be accessible via one of the servers. This method is very close to our shortest path (minimum hop) approach which we utilized in this thesis. Figure 13 demonstrates created tree using this method.

Dense mode using MST: Instead of finding the shortest path from each server to all multicast client without considering path weights, a more intelligent way could be creating a minimum spanning tree (using well known algorithms such as Kruskal or Prim) and then using BFS algorithm to investigate path from all servers to multicast clients. Figure 14 illustrates created tree using this method.

Dijkstra & MiniMax: In this thesis we have used both Dijkstra and MiniMax algorithms to determine and assign a most suitable multicast tree at the given time to new multicast client comers. Our implemented system is a combination of sparse and dense mode hence, like sparse mode the joining procedure starts by receiving Join messages from the clients and similar to dense mode finding an appropriate multicast tree starts from available sources and multicast trees create per source. Finally, all available paths from servers to specific client are comparing with each other and the server which satisfies the requirements of Dijkstra/MiniMax algorithms will be selected. As it is shown in Table 1 both of these two algorithms has good performance and deemed as practical. On one hand time and space complexity of them are acceptable and on the other hand extracted values for multicast trees metrics are satisfactory. In addition, both of these algorithms are automatically handling the load balancing issues when servers link are normally congested. In contrast, when all links related to one server are totally congested, the multicast controller selects the other servers inevitably. Applying Dijkstra and MiniMax leads to creation of following multicast trees (Figure 15 and 16).

Dense mode: Traditional implementation of dense mode is very close to Dijkstra approach which we utilized in this thesis. In dense mode a server creates a minimum spanning tree and then broadcast the multicast packets through established tree. A node which is not interested in receiving multicast packets will send prune message to the switch, which received broadcast message from it. At the end by pruning branches from spanning tree, a multicast tree remains which consists of both server and clients. Dense mode requires pruning operation which is totally unnecessary when a comprehensive view over the network topology is already provided. Applying dense mode algorithm over illustrated topology leads to creating of same multicast tree of Figure 15.

4.2.0.2 Time Complexity Comparison

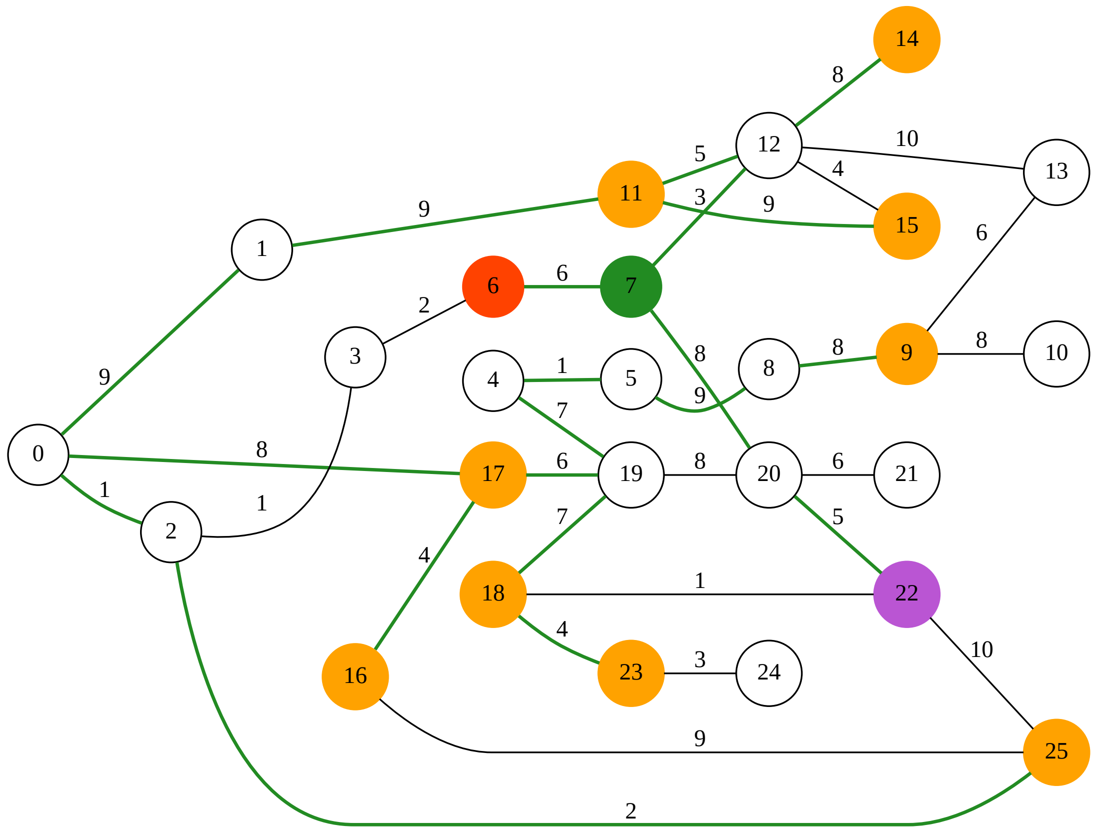
As it is shown in Table 1, 5 parameters interfere in time complexity of implemented algorithms (C, S, N_{lu}, V, E). Among these five parameters V and E are common between all methods and increasing each of these values has almost the same impact on the total time complexity although its impact on the last two algorithms is higher.

C shows the number of multicast clients, thereby no upper bound limits is defined for that. As a result, the time complexity of algorithms which are dependent on C varies from linear complexity to non-linear complexity. Hence the complexity of sparse mode with BFS and sparse mode using a method of load balancing algorithms increase to O$((n \times\ V^2) + (n \times\ E \times\ V))$ in the worst case when each node in the topology has n multicast clients attached. The final complexity is determined by number of edges in the topology. Number of link updates is a tradeoff between increasing the accuracy and better performance of the algorithm and increasing the time complexity of implemented algorithms which are dependent on this parameter. Although, unlike C there is an upper bound limitation for this variable.

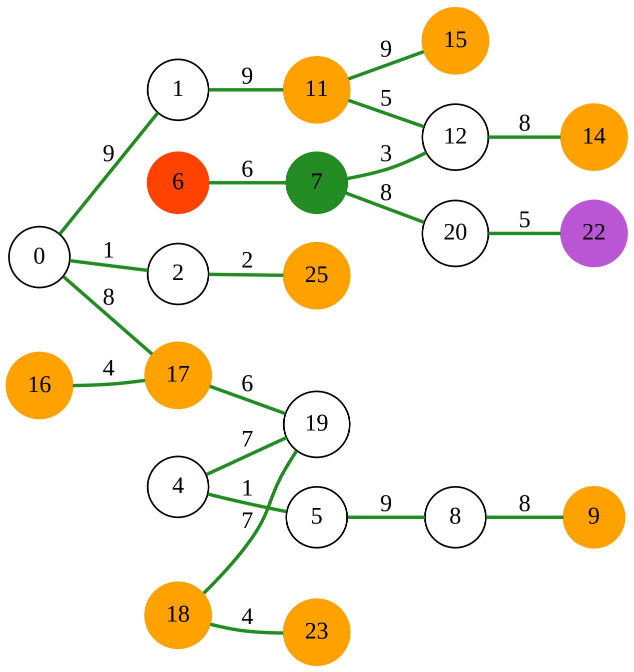
Dense mode using MST is sensitive to each link weight because each change in links weight leads to creation of distinct MST. As a result, after each link update procedure, the new MST will be created and new path from sources to clients will be determined. Dense mode using BFS is only affected by number of vertices and edges and it is independent from other variables.

Dijkstra and Minimax are dependent on number of multicast servers plus their dependency on number of link updates per minute. As we described these algorithms earlier, the controller investigates all available solutions from all multicast servers to a given client and finally compares the outputs and picks out the best solution. Hence in these two methods, the algorithm will be applied per each server but fortunately number of servers has a finite value and will not increase the order of complexity significantly.

Our proposed and implemented approach has both traditional sparse and dense mode specification. Multicast trees establish per each source (like dense mode) while joining procedure starts when a Join message sent from client. Simultaneously, our approach overcomes the overheads of traditional sparse mode which is selecting the RP and dense mode which is pruning the links which are not interested in receiving multicast packets. Unlike methods which assign the closest multicast tree to new multicast comer, our method selects the multicast tree according to some metrics (less total tree weight or more available bandwidth) by comparing all available servers and their corresponding path to the given client and finally picking out the best one which satisfies the requirements. Additionally, time and space complexity of proposed algorithm is acceptable.

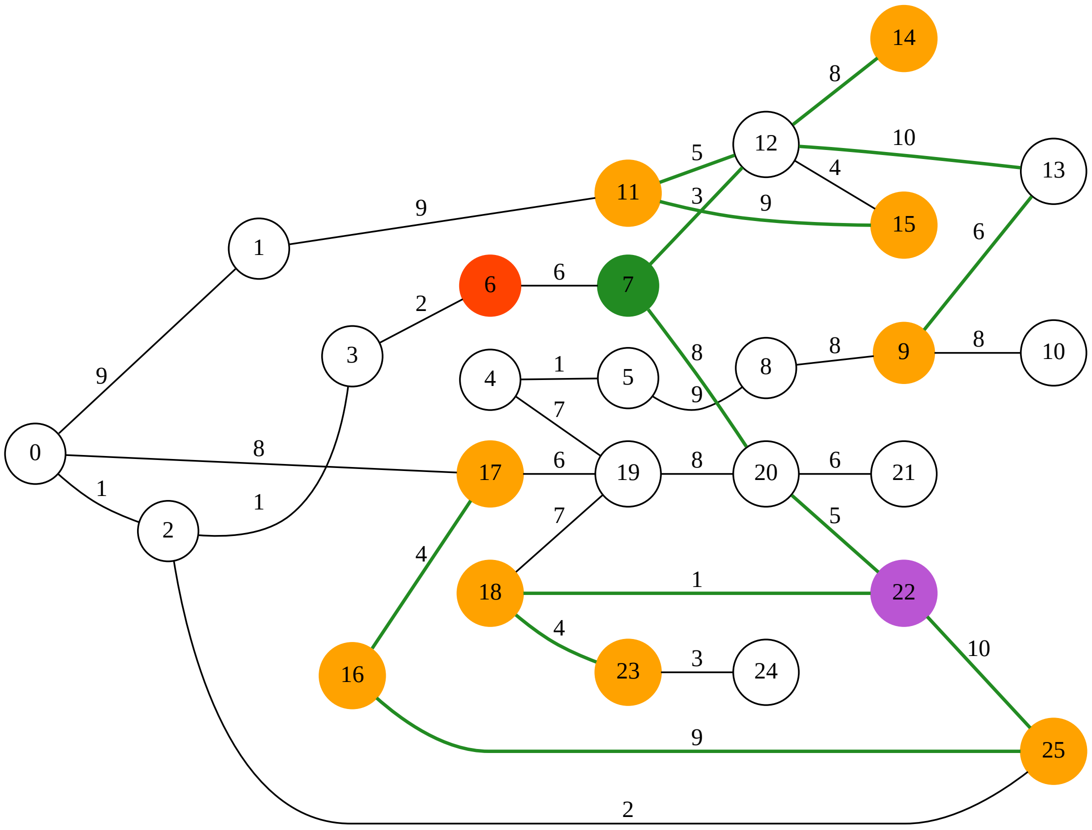


(a) Multicast tree structure in network topology

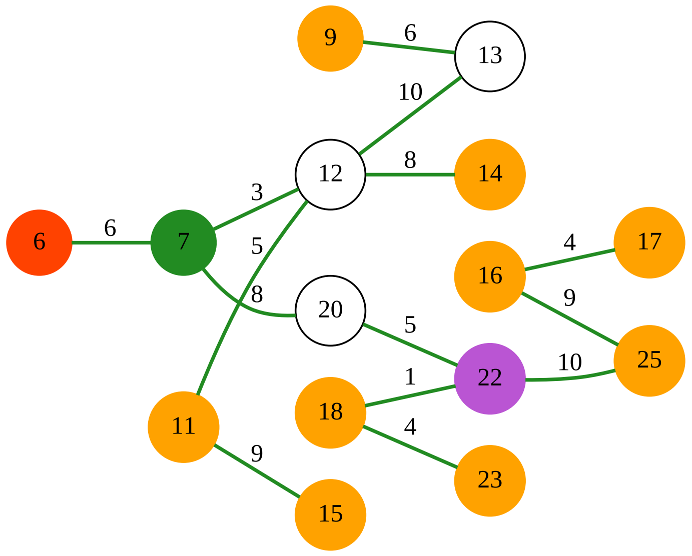


(b) Extracted multicast tree structure

Figure 9: Multicast tree generated using traditional sparse mode

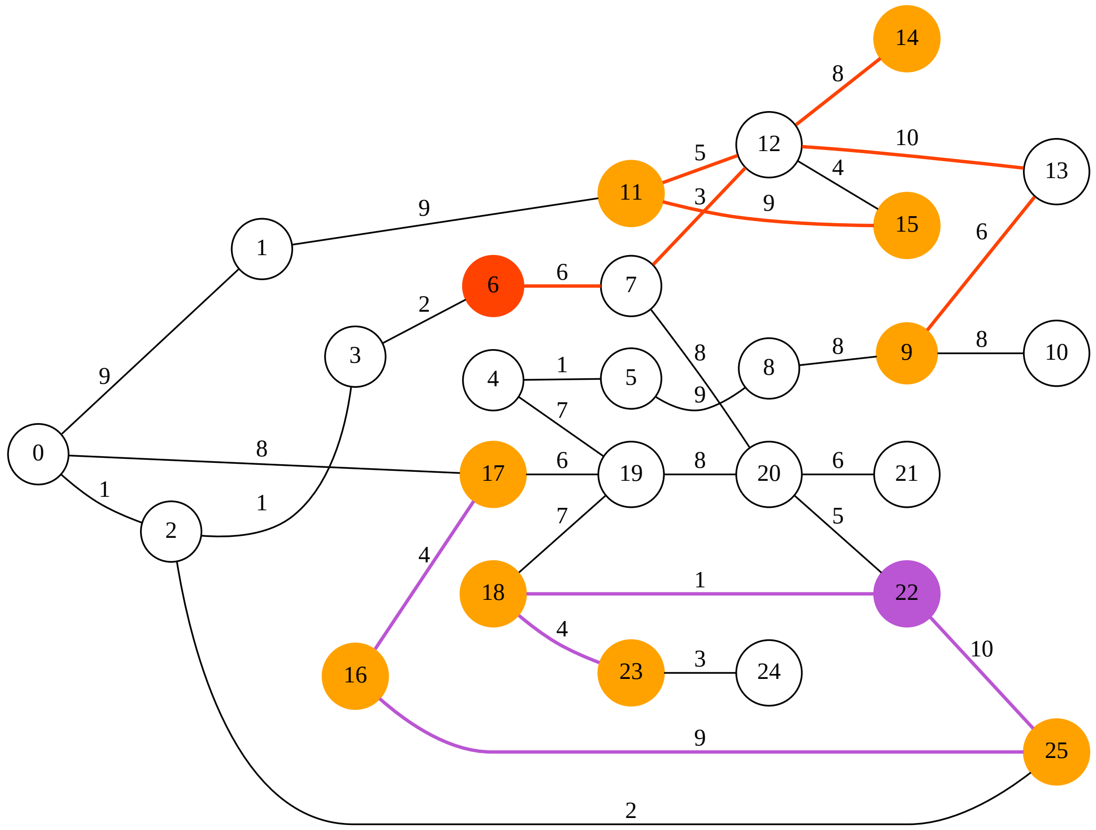


(a) Multicast tree structure in network topology



(b) Extracted multicast tree structure

Figure 10: Multicast tree generated using BFS in traditional sparse mode


(a) Multicast trees structure in network topology

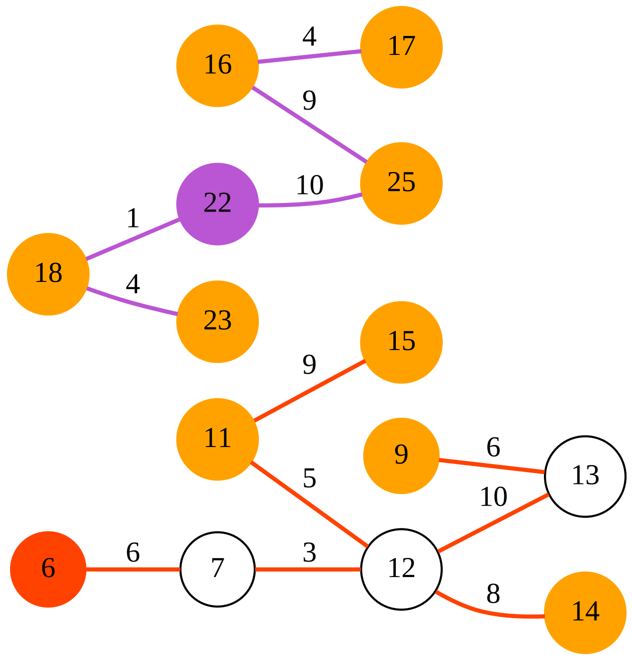
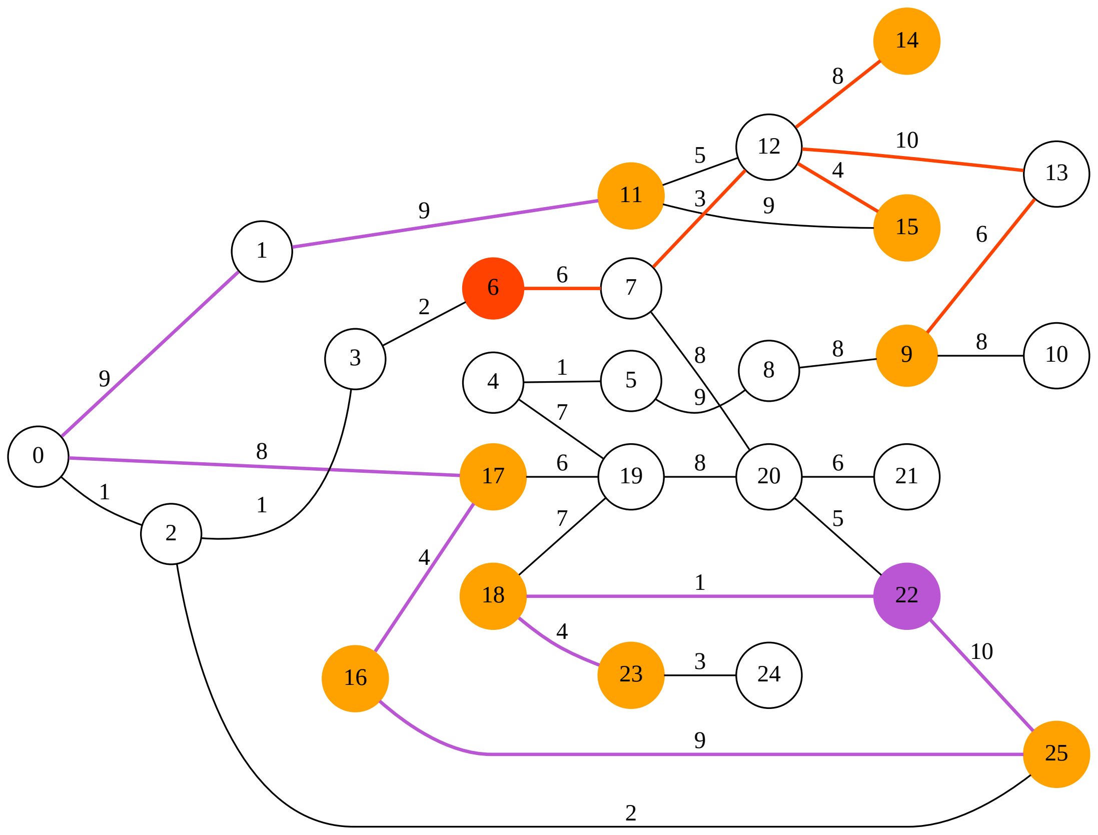
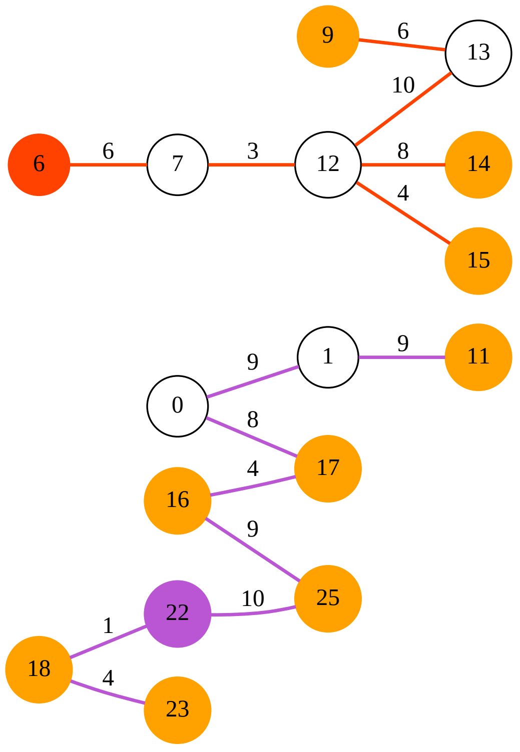

(b) Extracted multicast trees structure

Figure 11: Multicast trees generated using sparse mode (no RP)

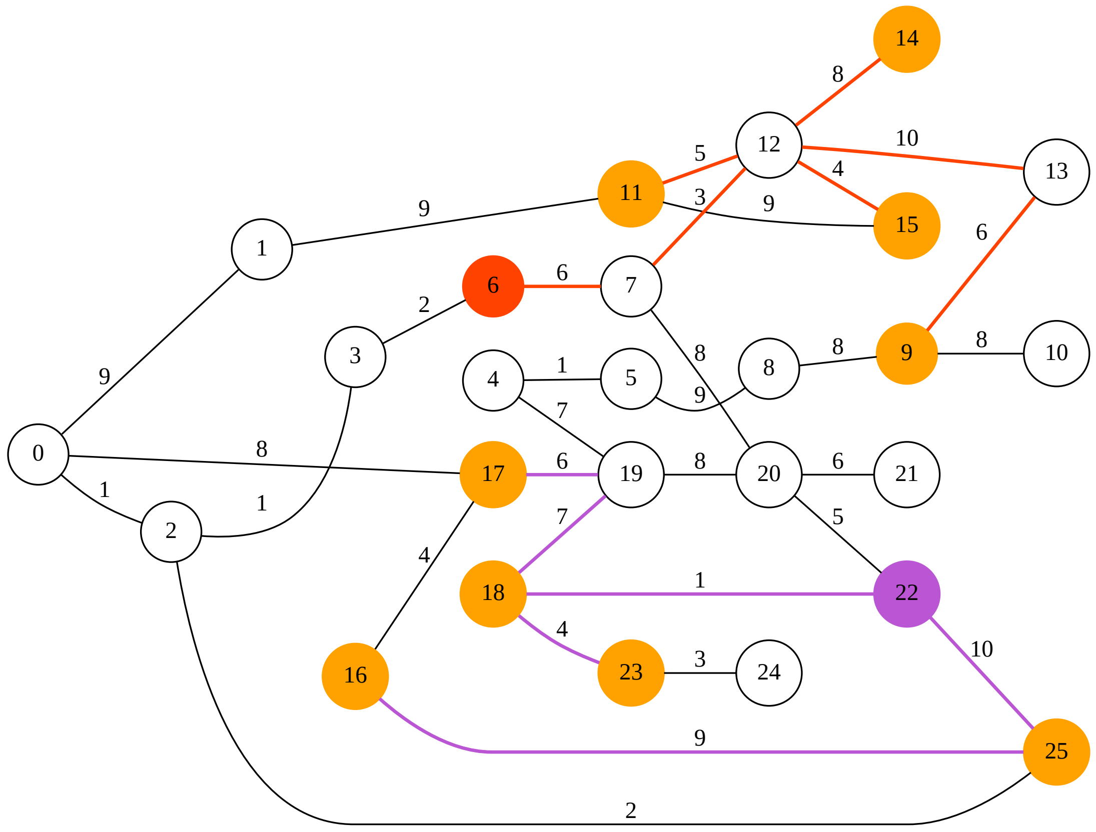


(a) Multicast trees structure in network topology



(b) Extracted multicast trees structure

Figure 12: Multicast trees generated using sparse mode and load balancing (no RP)


(a) Multicast trees structure in network topology

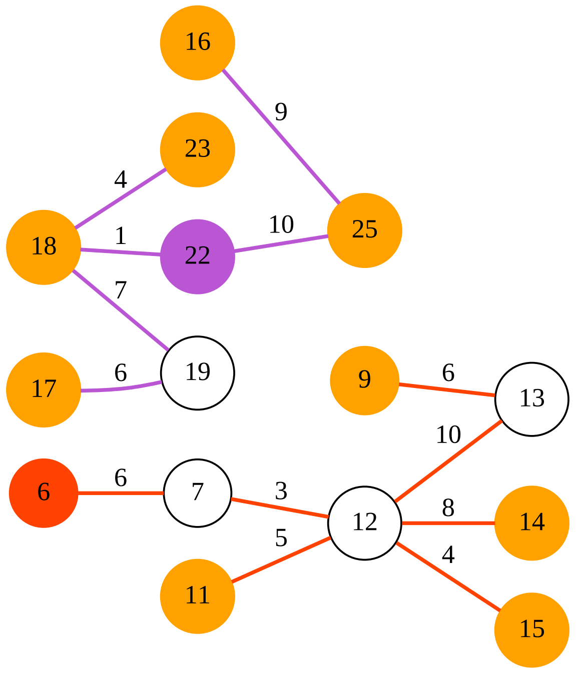
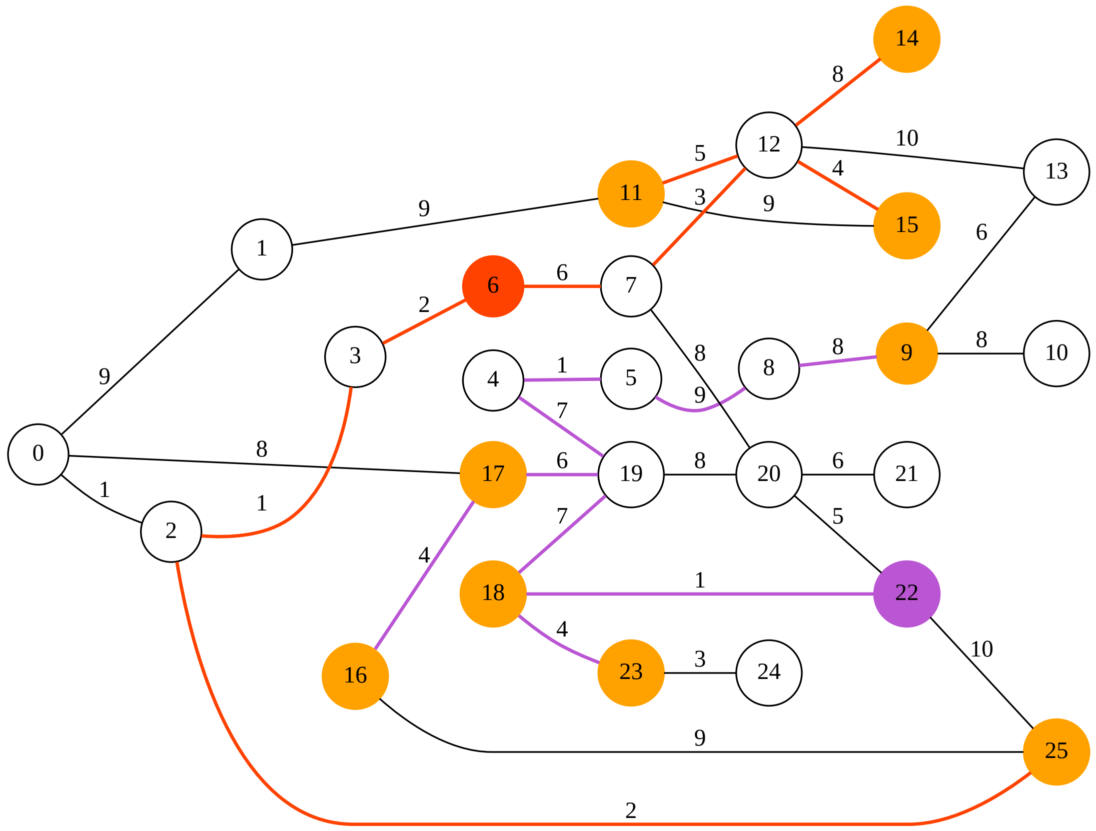

(b) Extracted multicast trees structure

Figure 13: Multicast trees generated using BFS in dense mode


(a) Multicast trees structure in network topology

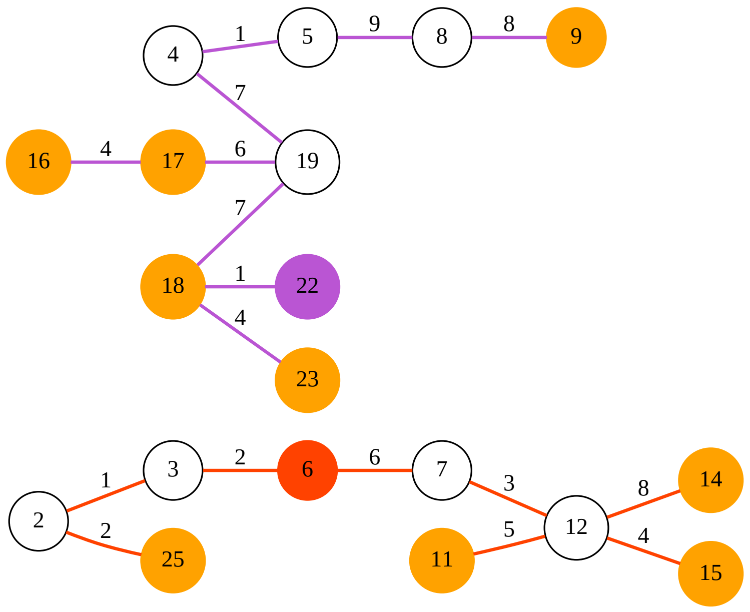
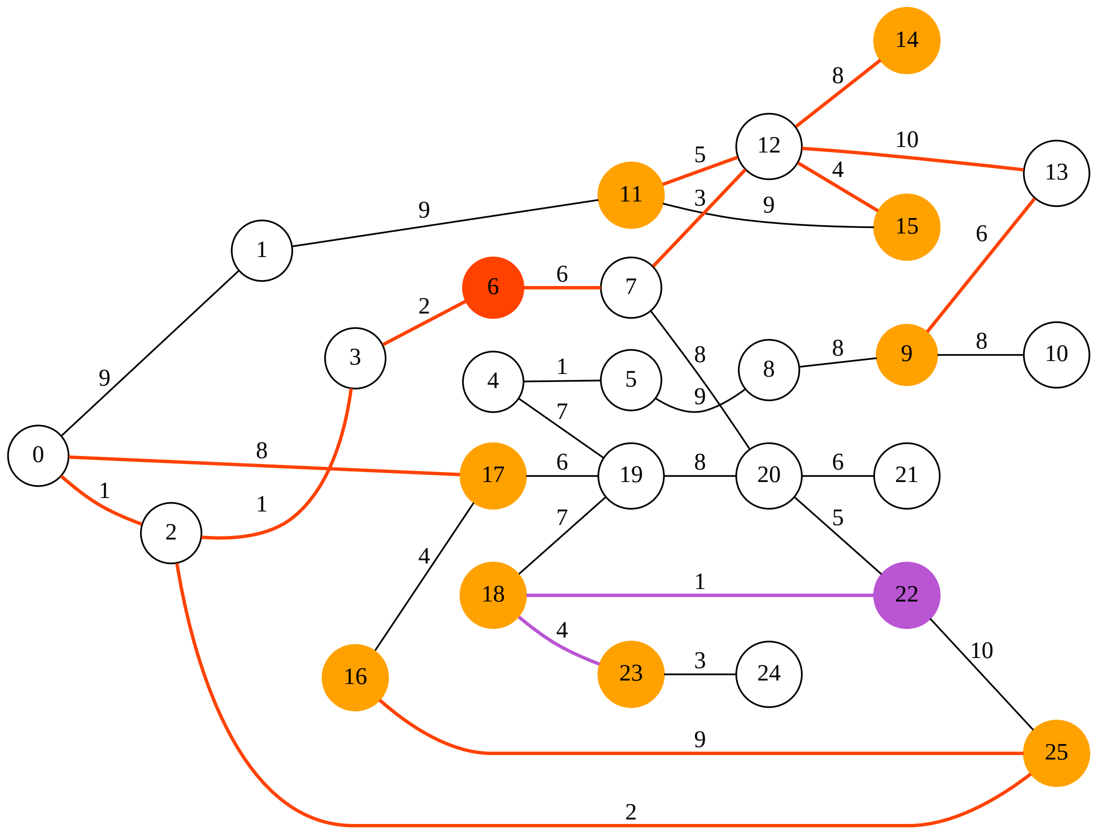
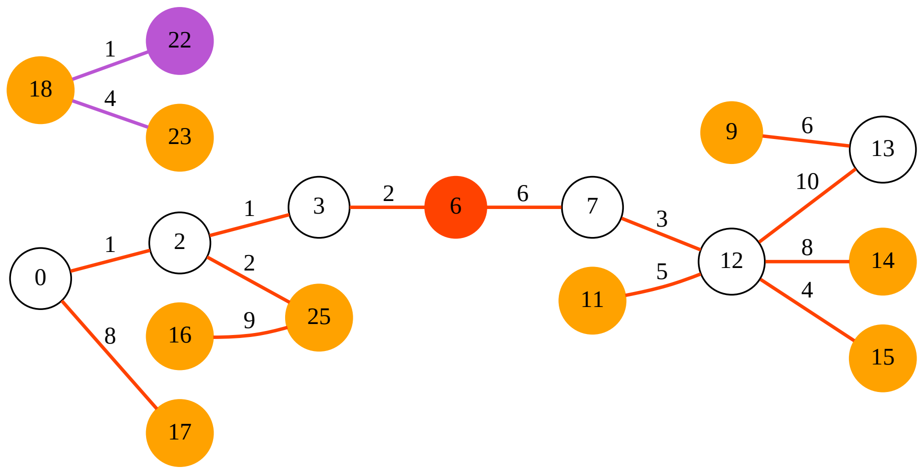

(b) Extracted multicast trees structure

Figure 14: Multicast trees generated using kruskal algorithm in dense mode

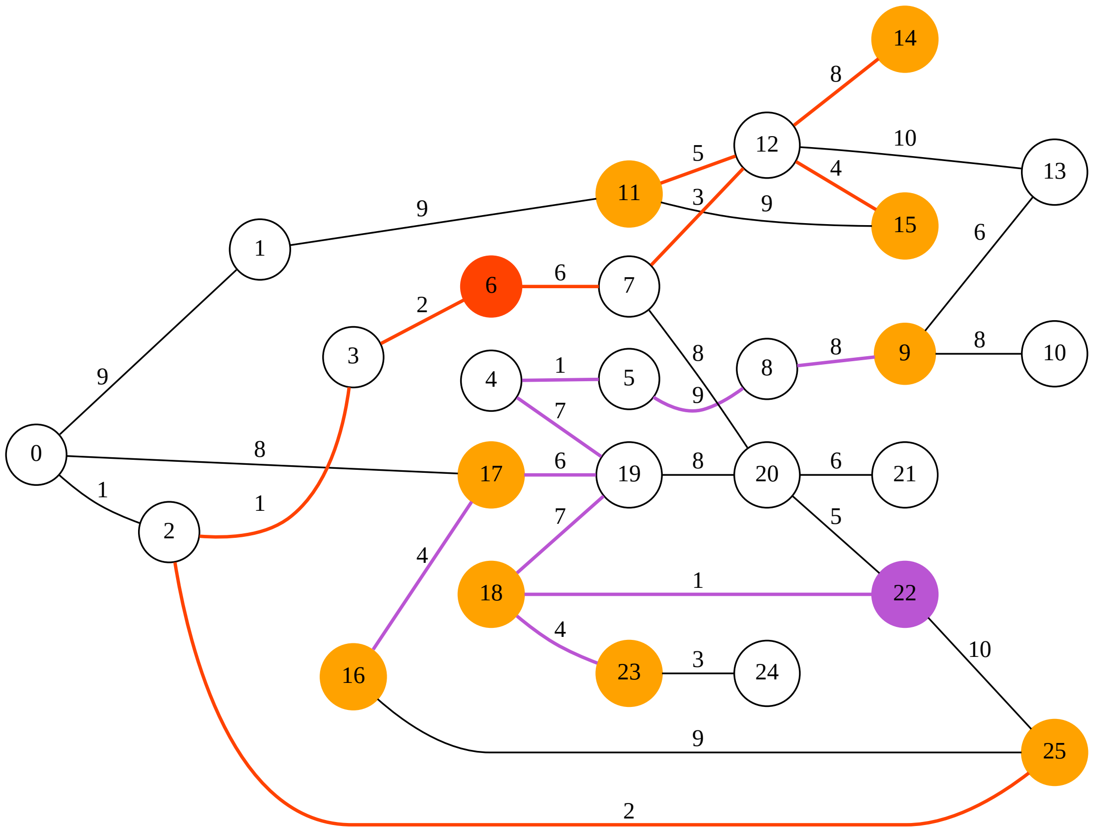


(a) Multicast trees structure in network topology

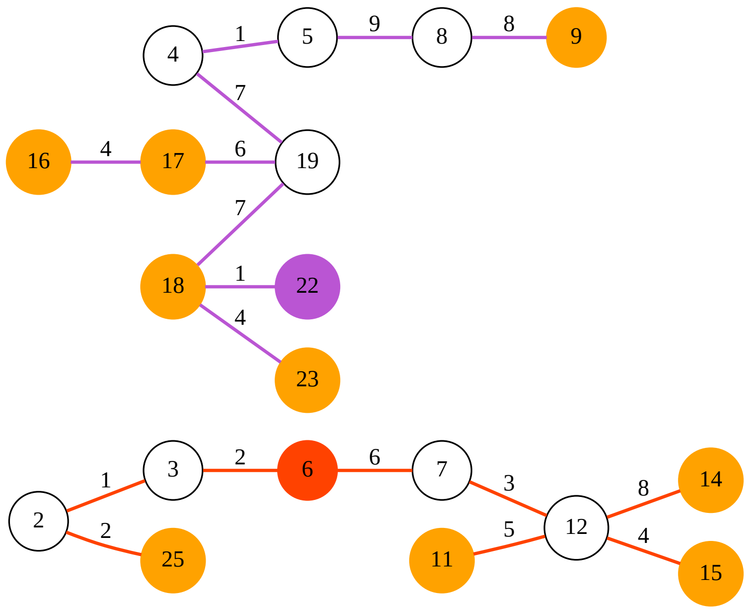


(b) Extracted multicast trees structure

Figure 15: Multicast trees generated using dijkstra algorithm



(a) Multicast trees structure in network topology



(b) Extracted multicast trees structure

Figure 16: Multicast trees generated using minimax algorithm

CHAPTER V

PROPOSED ARCHITECTURE & IMPLEMENTATION

This chapter proposes a framework architecture for video streaming over OpenFlow networks managed by a single control plane. We apply the proposed architecture to streaming of MDC videos, where each video is divided to odd and even descriptions and distributed over servers at different physical locations. The rest of this chapter investigates the architecture components in more detail.

5.1 Architecture

The proposed streaming video multicast framework is built on three pillars:

- IP-Multicast
- Multiple-Description Coding (MDC)
- Software-Defined Networking (SDN)

MDC encodes the video into multiple, independently decodable streams where any description can be used to decode the media stream to provide error resilience to the system at the expense of a slight reduction in compression efficiency. Descriptions are distributed across the network to benefit from multipath routing. A similar benefit of error resiliency may be realized with Scalable-Video Coding with better coding efficiency. However, this improvement comes at the expense of the need for continuous careful orchestration of what different servers transmit and how packets from multiple servers are processed at the subscriber hardware, both of which require more advanced hardware realizations.

The use of IP-Multicast minimizes the unnecessary transmission of replicated packets in the network. Implementing multicast in network layer not only decreases the probability of network congestion, but also increases the end-to-end packet delivery likelihood for all media and other services for clients at the same time.

In our implementation, the streaming video content is MD-Coded with 2 distinct descriptions. We have a number of servers in the network, each server streaming one of the two descriptions. Two classes of subscriptions are possible for the service including Standard and Premium. The premium users subscribe for a high quality streaming experience. They achieve this via reception of both descriptions. For this purpose, premium users are connected to two distinct multicast trees at a given time. When delivery of one description fails, the premium user will still be able to continue its playback, albeit, at a reduced quality level. The standard users, on the other hand, subscribe for a standard quality streaming experience. At a given time, the standard user receives only one description for playback which belongs to only one multicast tree. When the delivery from this tree fails, the standard user experiences a pause until the failure is corrected or the user is migrated to a new server. The premium user experiences a pause only when both trees experience failures. Figure 17 illustrates comprehensive view of multiple-description video using multicast over SDN.

5.2 Implementation

In addition to multicast control application which is described in previous chapter, the proposed architecture constitutes of following items.

5.2.1 OpenFlow Controller

SDN aims to reduce network reaction time to traffic variations by moving path allocation from individual devices to centralized controller software that lives on a workstation or server.

In traditional networks, Spanning Tree or routing protocols often take on the task of topology management, such as ensuring freedom from loops. Due to the distributed algorithms of these protocols, a number of difficulties arise, such as a

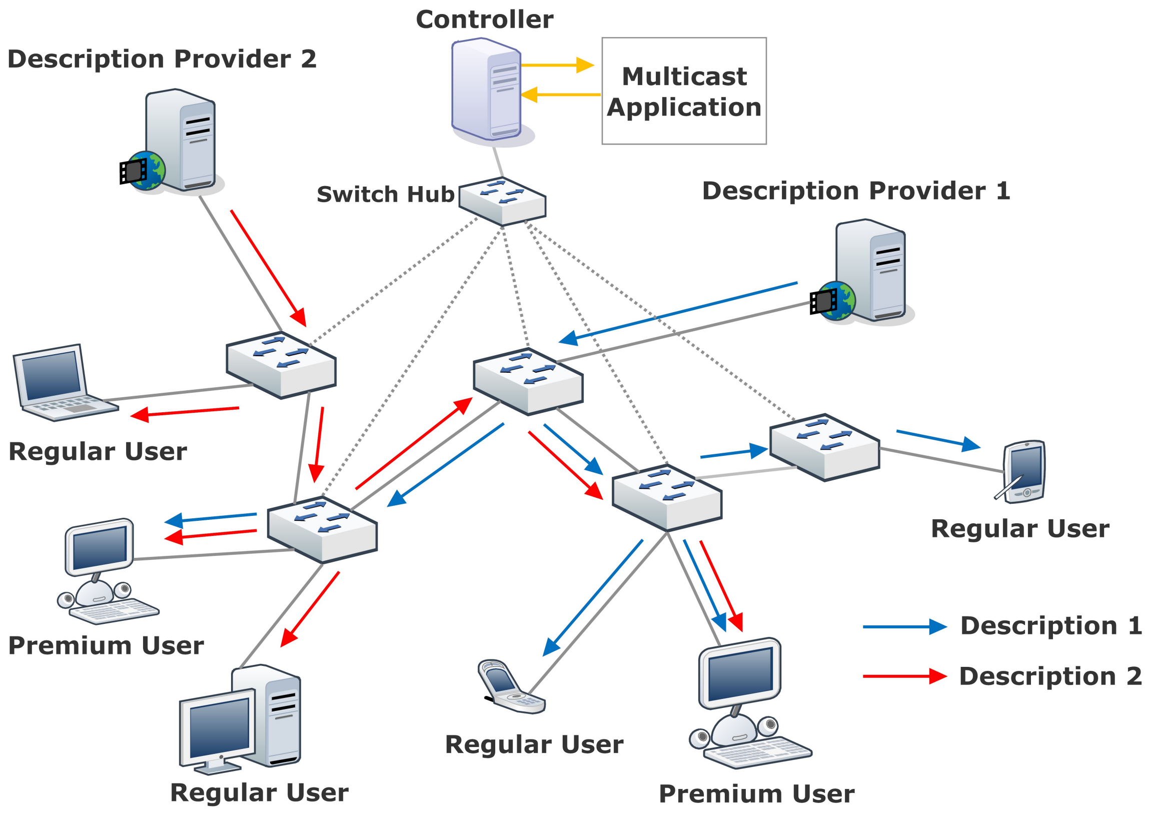


Figure 17: General perspective

complex configuration, a limited number of hops or long convergence times for changes in the underlying network infrastructure. Exploiting multiple paths between the start and destination of a data flow involves considerable effort and the use of other protocols. In contrast, SDN controllers have a central view of all network components and can therefore greatly simplify topology management. The controller component communicates with each device in the network, receiving updates on load and link status and then managing traffic flows among the devices.

The first OpenFlow controller platform, NOX, was released in early 2008. NOX originally used cooperative threading to process events in a single threaded manner. In 2011 a version of NOX was released [61] with a multithreaded learning switch application. Many other OpenFlow controllers have been released after NOX, which are written using different programming languages. Obviously, the programming language and development environment have a significant impact on the productivity of developers, and could also be a limiting factor in application performance.

Among all existing programming languages, Java seems an appropriate choice for developing an OpenFlow controller. Java is a cross platform high performance programming language, and has this ability to automatically manage the memory. Other programs written in Java such as Hadoop and Tomcat exhibited high performance. Although many other user friendly programming languages exist, their performance is unknown when used in an OpenFlow controller.

5.2.1.1 Beacon

Beacon is a Java-based open source OpenFlow controller created in 2010. It has been widely used for teaching, research, and as the basis of Floodlight. Beacon enhanced the OpenFlow controller design from various aspects, with a focus on being developer friendly, high performance, and having the ability to start and stop existing and new applications at runtime. Beacon showed surprisingly high performance, and was able to scale linearly with processing cores, handling 12.8 million Packet-In messages per second with 12 cores, while being built using Java [62].

We initially started our implementation over Beacon but this OpenFlow controller could not handle topologies with loops. Loops in topologies occur when there is more than one path between two endpoints which is widely common in every network topology. The loop creates broadcast storms as broadcasts and multicasts are forwarded by switches on every port. Then the switch or switches will repeatedly rebroadcast the broadcasted messages flooding the network. The solutions such as Spanning Tree Protocol (STP) are designed and implemented on the network switches. In this thesis, we did not bind our network topologies to specific characteristics and this purpose was not achievable by Beacon due to absence of ability to handle topologies with loops. In addition, Beacon could not handle devices with multiple attachment points (a device reachable from multiple switch ports) and was not provided with new features for many years. Hence, we moved our implemented system from Beacon to Floodlight controller.

5.2.1.2 FloodLight

Floodlight is a Java based OpenFlow controller that has forked from one of the two pioneering OpenFlow controllers developed at Stanford called Beacon controller. Floodlight attributes to the simplicity and yet high performance of the controller and provides a modular programming environment. Beside supporting topologies with loops, non-OpenFlow domains, and multiple device attachment points, it even introduced lots of other features over Beacon.

Floodlight implements a sophisticated mechanism for automatically detecting the topology of an OpenFlow network. Using a link-discovery module, the controller generates both LLDP and broadcast packets (referred to as BDDPs) and sends them to all neighboring switches on a regular basis. Assuming all switches consume LLDP messages and forward broadcast packets, Floodlight can identify active connections by receiving its own messages and computing the network topology.

Floodlight makes a distinction between direct links and broadcast links; a direct connection is always assumed if it receives its own LLDP packets. In this case, two OpenFlow switches are directly connected under the control of the same Floodlight instance. Based on the information of the link discovery mechanism, the topology service computes a topology map in the form of a directed graph. The map contains all the relevant information about interconnectivity between switches, and they can be used by other applications, such as computing a spanning tree.

Floodlight currently provides two modules for automatic packet forwarding between endpoints. A relatively simple Forwarding module mainly serves as an exemplary introduction to Floodlight and the complex Learning Switch module implements behavior similar to standard switches; The Learning Switch detects and learns about new devices based on their MAC addresses.

We implemented our framework over FloodLight hence, it is designed to be an enterprise grade, it has high performance and simultaneously satisfies our expectations from the SDN controller by handling loops in topologies and multiple attachment points. Although we provide our plugin using FloodLight as the proof of concept, we believe that it should be easy to extend our approach for other standard OpenFlow controllers.

5.2.2 Topologies

Topologies are selected out of a collection which contains 243 different network topologies. Then topologies which has more than one island (there are two nodes in topology graph such that no path in topology graph has those nodes as endpoints) have been excluded from the collection. Then each file in collection are parsed and spatial relationships between connected nodes and adjacent features are extracted and saved in a separate file. The way that switches connected to each other are fixed however, other parameters are effective in creating different scenarios in one topology. We consider multiple variables for each of the effective parameters. In our implementation parameters which are effective in creating scenarios and their corresponding values are as follows:

1. Description Providers: Varies from 2 to 4,
2. Multicast Clients: Varies from 10 to 40 in 4 steps by 10 client increase at each step,
3. Cross Traffic Servers Pairs: Varies from 10 to 320 in 6 steps. These 6 steps are 10, 40, 80, 160, 240 and 320 respectively.

In addition, for each topology we considered three scenarios for multicast clients as follow:

1. 90% of the clients are assumed to be standard users, and remaining 10% are premium users.

2. 70% of the clients are assumed to be standard users, and remaining 30% are premium users.

3. 50% of the clients are assumed to be standard users, and remaining 50% are premium users.

In total 51840 scenarios (240 topologies * 216 different scenarios per topology) has been generated and prepared for further access.

5.2.3 Multicast Data Structure

A customized version of the Java JTree is used by IP-Multicast controller to maintain multicast tree structure. JTree commonly used to display hierarchical data and we find it appropriate for keeping multicast structure due to following reasons:

- No Restriction: This data structure is very flexible because of the purpose that is design for. As mentioned earlier, this data structure is designed to display hierarchical data which means that there is no restriction about depth of tree, number of children and number of siblings. Multicast trees do not have a specific data structure and they require a flexible structure to keep their content.

- Number of practical functions: Many practical functions have been implemented in this data structure (i.e. a function which gives two given node in the tree and returns their common ancestor immediately). Such functionality facilitates seeking and editing procedures in multicast tree.

- Already Implemented: This data structure is already implemented and tested in Java and widely used in different test cases, so good performance and minimum number of bugs are guaranteed.

Source codes have been downloaded, customized and imported to Mulitcast application to keep multicast tree structures. The control application establishes a new tree for each DP using this data structure.

5.2.4 Description Providers

DPs stream packets with the exact size and rate of the actual streamed video which is 15fps with an average 1000 kb/s bit rate, but instead of transmitting the video data, we let the DPs populate the packets with parameters which help us analyze and track them further a posteriori. These parameters are:

- Information regarding the source DP,
- Frame number,
- Frame sequence number,
- Packet sent time,
- Packet sequence number.

DPs have access to video frame list, fragment each frame to specific number of subdivisions and insert them in a sending queue according to their frame and subdivisions number. Before sending a packet, DPs refer to a subdivision which has to be transmitted at that round (the subdivision which exists in front of the queue) and extract frame number and subdivision number from it. Extracted information will join to other mentioned information (i.e. packet sequence number) and rest of packet fills with dummy data. Packets are sending over UDP protocol hence DPs are streaming multicast packets. Multicast traffic is handled at the transport layer with

UDP, as TCP provides point-to-point connections which is not feasible for multicast traffic.

DPs are written in Java and streaming packets on specific multicast IP address (225.0.0.38).

5.2.5 Multicast Clients

Two types of users are envisioned: Standard and Premium. While standard subscribers are to receive one of the descriptions of the video, premium subscribers will receive multiple descriptions, each from a different source, simultaneously and combine these descriptions prior to playback in order to increase video quality.

Main duty of multicast client is to listen on specific port and multicast IP address in order to capture the packets, parse its containing data, stamp packet with its receiving time and saving them to a file for posterior analyze. Both premium and regular users are identically following this procedure however, premium ones are capturing packets with higher rate. Likewise DPs, multicast client implementation done using Java.

Beside capturing multicast packets, multicast clients send specific messages to controller to join or leave multicast group. Following messages are sent by multicast clients:

- Regular Join: Sent by a given node identifies willingness of the regular client to join the multicast group.
- Premium Join: Sent by a given node identifies willingness of the premium client to join the multicast group.
- Leave: Sent by a given node identifies willingness of the client to leave the multicast group.

We distinguish control packets from each other and from streaming packets by assigning different multicast IP addresses to them (i.e. If a packet reaches to the controller with destination IP address of 225.0.0.1, the controller follows the procedure of regular joining for the client which sent this message). In addition, Join messages can specify the IP-Multicast that the host is attempting to join, but since we consider one active multicast group in our implementation, body of Join messages are empty.

Important issue about the client is the procedure of joining and leaving the multicast group which has to be fixed during all simulations, otherwise experiment result will not be reliable. For this purpose and before running the main simulations, Video consumers are determining a reference pattern in a separate simulation. In this simulation, clients first submit their Join message and stay in multicast group for 45 seconds. Afterwards, the clients randomly choose to either submit a new request (Leave request if they are already being serviced and vice versa) or remain in their current state for another 45 seconds. We assume that the probability of submitting a leave request is 20% and the probability of submitting a join request is 80%. A higher probability is considered for subscribing to the service since the clients may capture more packets this way which in turn improves the evaluation accuracy. Also Clients record the time they joined and leaved streaming service.

5.2.6 Cross Traffic Generators

To emulate the behavior of realistic networks, we designed a client server traffic generator application, where multiple clients continuously pump data to the given servers over UDP. The main purpose of the cross traffic generation is to congest the network and since this is not achievable by TCP due to its inherent congestion control mechanism, all cross-traffic packets are transmitted using UDP in the experiment.

Artificial traffic between cross clients and servers are copied from 4 real patterns. These 4 patterns are: HTTP, FTP, audio and video conference (Skype) and video streaming (YouTube). Traffics captured from interaction between real host and its related service provider. Then captured traffic is analyzed and its related pattern is extracted (size of packets and sending bit rate) and finally extracted pattern hard-coded to cross traffic servers.

In order to have same traffic pattern during all simulations, cross traffic servers are following a fix scenario obtained from a separate simulation where cross traffic servers (320 servers in total) determining a reference pattern. In this simulation each server chooses one of the 4 traffic patterns, then simulate first 1024 packets of that traffic by sending fake packets with exact size and rate of reference traffic pattern. When 1024th packet is departing the server, cross server chooses one more traffic pattern to simulate. Each cross traffic server transmits data to a predetermined receiver that remains status throughout the experiment. We captured all decisions made by cross traffic servers and set them to follow the same pattern in our real simulations.

Cross traffic servers are implemented using Java and cross traffic clients are implemented in Python to reduce memory usage by the host computer where simulations are run over that.

CHAPTER VI

EXPERIMENT SETUP & EVALUATION

We conduct experiments to assess the performance of the proposed architecture when Minimum Hop, Shortest Path and MiniMax routing algorithms are deployed. To assess the benefit of SDN-based IP-Multicast MDC video streaming, we also investigate the performance of ALM for both SDN as well as non-SDN networks. In SDN networks, end-to-end routes are established for ALM by the controller which has global network view. The non-SDN network, on the other hand, is today's Internet, where routes are computed in a distributed manner by the individual switches which have local network views of their neighborhoods. Non-SDN network is simulated by disabling Forwarding module and enabling Learning Switch module in floodlight controller. The Learning Switch module implements behavior similar to standard switches.

6.1 Test Setup

We conduct the experiment on Mininet version 2.0 in four different topologies with 15-20 switches, implemented with Open switch 1.4, with each switch connected to an average of 2.67 other switches. We assume that each link has a bandwidth of 100 Mbps. The topology sizes are selected so that investigation of a heavily congested network is possible. Congesting bigger topologies with higher link bandwidths is difficult with limited resources such as memory. Cross traffic generators have been created real traffic patterns. System performance at different network loads is desired. Thus, cross traffic from a range of 10-320 servers is considered.

For each network topology, the experiment is conducted for 20 minutes. An additional 5 minutes is set aside to initialize the emulation testbed and 2 minutes between each emulation to calm down the CPU and memory usage. All experiments run over an IBM Server with 12 cores of CPU and 28 Gigabytes of RAM.

During the initialization phase, the cross-traffic generators start congesting the network for the first minute. The DPs start streaming the two emulated descriptions of the video for another minute after which the subscribers start joining the service.

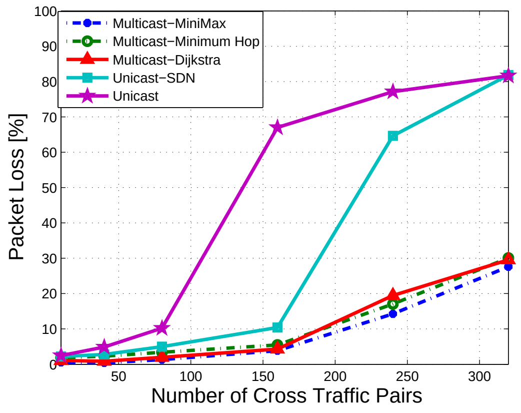
In all experiments, the sequence of joining and leaving for both standard and premium users are fixed and they are following the predetermined pattern. In our simulations we consider 10 multicast clients which are distributed through selected topologies. Finally following QoE metrics are investigated out of 80 subscriptions:

1. Packet Loss
2. Denied Service
3. Pauses & Downgrades
4. Pre-Roll Delay
5. PSNR

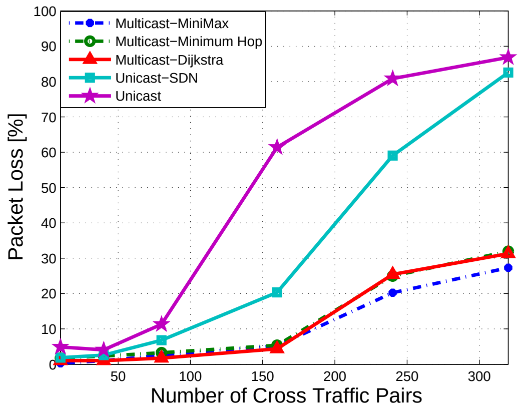
In our simulation we investigate metrics in two domains: i) Network, ii) Multimedia. First two metrics are related to network aspects and remaining metrics are relevant to multimedia aspects.

6.2 Packet Loss

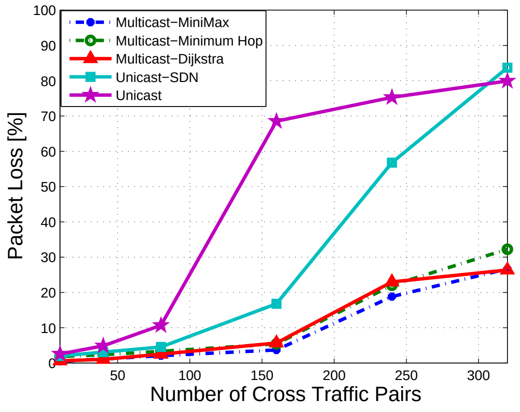
We first investigate the percentile loss of video packets due to congestion in the network. Packet loss percentage is calculated by tracking packet sequence number and having known that how many packets is sent by source DP during the time that client was subscribed in service. The results are depicted in Figure 18. We observe that as the cross-traffic in the network increases (loaded network), the ALM performance becomes significantly worse than SDN-based IP-Multicast performance. While SDN



(a) 10% Premium User



(b) 30% Premium User



(c) 50% Premium User

Figure 18: Packet Loss

with ALM generally performs better than a non-SDN with ALM, both have significantly worse performances than the SDN with IP-Multicast. This result confirms that while SDN is essential in reaping the gains of the IP-Multicast architecture, it is not sufficient on its own. It acts as an enabler to easily implement architectures that would be difficult, or even impossible, in today's Internet, which in turn provides significant performance gains.

Of the three routing algorithms, MiniMax incurs the lowest packet loss. However, the performance difference between them is not very large. This result is dependent on the topologies on which the experiment is run. MiniMax may achieve more significant gains over different topologies where there are more available paths between switches.

6.3 Denied Service

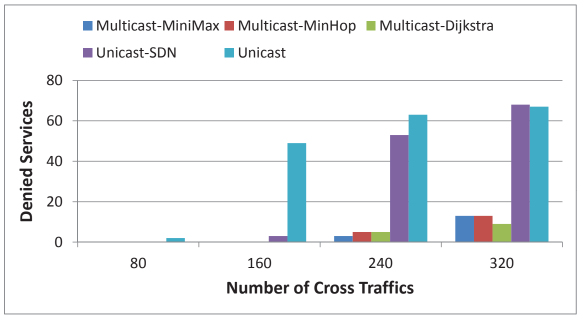
Denied service is defined as time intervals where consumers subscribe to video service and leave it without receiving any packet. Denied service happens due to following reasons:

- Loss of join request. This challenge intensifies when network is congested but it is also probable to occur in a least congested network due to links unreliability.
- Join request is received by the mulitcast control service but due to congestion all video content packets traveling fail to reach their consumer.

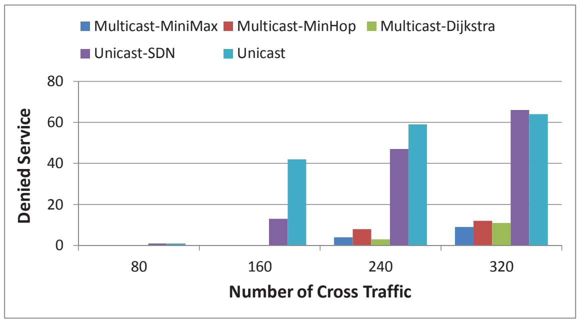
In both aforementioned cases, consumer is denied to receive the content which is eager to watch. Figure 19 illustrates total number of denied service occurred when 10%, 30% and 50% of video consumers are premium ones respectively. Hence, number of denied services are 0 when the network is less congested (with 10 and 40 cross traffic servers) therefor, we exclude them from the chart. Following inferences can be extracted from the available data:

- SDN with IP-Multicast outperforms SDN and non-SDN with ALM and has remarkable less number of denied service.
- Increasing number of premium users leads to decrease in total value of denied service hence premium users are supposed to receive two independent streams from different paths and as a result probability of facing denied service is lower.
- MiniMax incurs the lowest number of denied service.

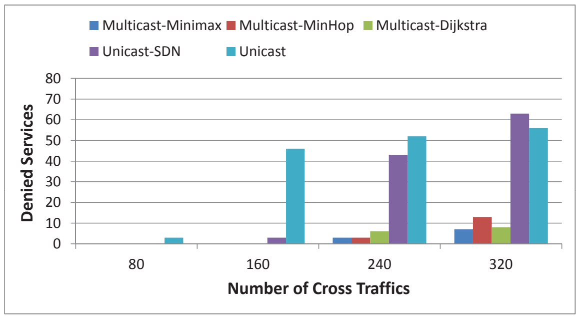
Denied service for regular users can be interpreted as a fatal pause and for premium users can be interpreted as two-stage failure: i) Fatal downgrade from high quality video to lower quality and ii) Fatal pause.



(a) 10% Premium User



(b) 30% Premium User



(c) 50% Premium User

Figure 19: Denied Service

6.4 Downgrade/Pause

Downgrade includes time intervals where premium user is watching the video like regular user and complement description is not delivered in time or not delivered at all. In such time intervals video quality is downgraded from premium quality to regular one. Table 2 tabulates the observed number of downgrades for SDN-based IP-Multicast with different routing algorithms, as well as SDN and non-SDN-based ALM for different network congestion levels. The table lists the values for the top performing 90% and 100% subscribers. The table entries with dashes correspond to the case where there are less than 90% or 100% of the users that can receive the service in that scenario as a number of the users are denied service completely due to congestion.

Pause is defined as the time interval where there is no frame to display and video demonstration buffer is empty. In such cases clients wait till their buffer is filled

Table 2: Number of Downgrades

Cross Traffic	Multicast-MiniMax		Multicast-Minimum Hop		Multicast-Dijkstra		Unicast-SDN		Unicast	
Name	90%	100%	90%	100%	90%	100%	90%	100%	90%	100%
10	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	1	-
160	0	0	0	0	0	0	2	-	-	-
240	3	-	5	-	5	-	-	-	-	-
320	-	-	-	-	-	-	-	-	-	-

(a) 10% Premium User

Cross Traffic	Multicast-MiniMax		Multicast-Minimum Hop		Multicast-Dijkstra		Unicast-SDN		Unicast	
Name	90%	100%	90%	100%	90%	100%	90%	100%	90%	100%
10	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	1	-
160	0	2	0	0	0	0	1	-	-	-
240	9	-	13	-	13	-	-	-	-	-
320	13	-	-	-	-	-	-	-	-	-

(b) 30% Premium User

Cross Traffic	Multicast-MiniMax		Multicast-Minimum Hop		Multicast-Dijkstra		Unicast-SDN		Unicast	
Name	90%	100%	90%	100%	90%	100%	90%	100%	90%	100%
10	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	2	-
160	0	5	0	0	0	0	3	-	-	-
240	16	-	24	-	16	-	-	-	-	-
320	19	-	-	-	23	-	-	-	-	-

(c) 50% Premium User

up to initial threshold. When initial buffer is filled up, video playback starts again. Although neither of downgrade and pause are desired but occurrence of pause has more destructive effect on video playback rather than downgrade. Table 3 shows number of pauses in top 90 and 100 percentages of subscriptions. Following inferences can be extracted from the available data:

- Number of downgrades increase as number of premium users increase.
- As a result some of the dashes in table 1 part b and c, the 30% and 50% premium users, will be replace by relevant number of downgrades.
- By increasing number of premium users it is most probable that a given user can receive the service. As a result some of the dashes in Table 3 part b and c, the 30% and 50% premium users, will be replace by relevant number of downgrades

Table 3: Number of Pauses

Cross Traffic	Multicast-MiniMax		Multicast-Minimum Hop		Multicast-Dijkstra		Unicast-SDN		Unicast	
Name	90%	100%	90%	100%	90%	100%	90%	100%	90%	100%
10	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	-
160	0	0	0	0	0	0	1	-	-	-
240	26	-	39	-	46	-	-	-	-	-
320	-	-	-	-	-	-	-	-	-	-

(a) 10% Premium User

Cross Traffic	Multicast-MiniMax		Multicast-Minimum Hop		Multicast-Dijkstra		Unicast-SDN		Unicast	
Name	90%	100%	90%	100%	90%	100%	90%	100%	90%	100%
10	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	-
160	0	0	0	0	0	0	3	-	-	-
240	23	-	34	-	29	-	-	-	-	-
320	25	-	-	-	-	-	-	-	-	-

(b) 30% Premium User

Cross Traffic	Multicast-MiniMax		Multicast-Minimum Hop		Multicast-Dijkstra		Unicast-SDN		Unicast	
Name	90%	100%	90%	100%	90%	100%	90%	100%	90%	100%
10	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	-
160	0	0	0	0	0	0	0	-	-	-
240	19	-	23	-	23	-	-	-	-	-
320	40	-	-	-	71	-	-	-	-	-

(c) 50% Premium User

(i.e. in Multicast-MiniMax top 90 when 30% of users are premium ones and Multicast-Dijkstra top 90 when 50% of users are premium). This fact confirms the claim that increasing the number of premium user will increase the chance of receiving the service.

- By increasing number of premium users, number of pauses decrease hence, clients have more chance to receive at least one description.
- MiniMax performs better than two other routing algorithms.

In order to investigate downgrades and pauses two queues have defined:

1. Regular Odd/Even queues which keep packets from odd/even DP.
2. Initial queue. Size of initial buffer is bigger than regular ones and it is used for

capturing packets for starting playback.

Utilizing the aforementioned buffers and by using the following sequential procedure number of downgrades and pauses are calculated:

1. Select a client, then go through packets captured by this client and push them in initial queue until it is filled up.

2. In a separate process, start fetching packets from initial queue and simultaneously push other packets to relative normal queues according to their DP sequence numbers (we repeat this procedure till client leaves the group). Fetching packets from queues (odd/even for regular users and both odd and even for premium ones) and speed of fetching depends on user type (regular user 15 fps and premium users 30 fps).

3. When initial buffer is empty, the same procedure continues with normal queues.

4. For premium user two cases may happen before it leaves the service: i) One of the queues (odd/even) is empty while the other one has packets to serve, ii) Both queues are empty. Former case interprets as a downgrade while latter case interprets as a pause.

5. The procedure for regular users is simpler. When there is no packet to fetch and client has not left the service yet, it means that a pause occurred.

6.5 Pre-Roll Delay

Pre-Roll Delay is defined as the difference between the time when a user subscribes for service and first packet for playback is received at the receiver. Table 4 tabulates the observed pre-roll delays (time values in seconds) for SDN-based IP-Multicast with different routing algorithms, as well as SDN and non-SDN-based ALM for different network congestion levels. As previously indicated, the table entries with dashes

Table 4: Pre-Roll Delay

Cross Traffic	Multicast-MiniMax		Multicast-Minimum Hop		Multicast-Dijkstra		Unicast-SDN		Unicast	
Name	90%	100%	90%	100%	90%	100%	90%	100%	90%	100%
10	0.5	0.5	0.4	0.4	0.5	0.5	0.5	0.5	0.5	0.5
40	0.4	0.6	0.4	0.5	0.5	0.6	0.5	0.5	0.5	0.5
80	0.5	0.6	0.4	0.4	0.4	0.5	0.4	0.5	0.6	-
160	0.6	1.7	0.5	0.5	0.5	1.3	0.5	-	-	-
240	9.8	-	19.2	-	19.0	-	-	-	-	-
320	-	-	-	-	-	-	-	-	-	-

(a) 10% Premium User

Cross Traffic	Multicast-MiniMax		Multicast-Minimum Hop		Multicast-Dijkstra		Unicast-SDN		Unicast	
	90%	100%	90%	100%	90%	100%	90%	100%	90%	100%
10	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
40	0.4	0.5	0.4	0.4	0.4	0.5	0.4	0.4	0.4	0.4
80	0.4	0.5	0.4	0.4	0.4	0.4	0.4	-	0.5	-
160	0.9	3.5	0.4	0.5	0.8	2.3	-	-	-	-
240	15.2	-	21.3	-	18.7	-	-	-	-	-
320	25.3	-	-	-	-	-	-	-	-	-

(b) 30% Premium User

Cross Traffic	Multicast-MiniMax		Multicast-Minimum Hop		Multicast-Dijkstra		Unicast-SDN		Unicast	
	90%	100%	90%	100%	90%	100%	90%	100%	90%	100%
10	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
40	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
80	0.3	0.4	0.3	0.3	0.3	0.3	0.3	0.3	0.5	-
160	0.5	0.7	0.4	0.6	0.4	3.4	0.5	-	-	-
240	20.1	-	30.1	-	23.8	-	-	-	-	-
320	28.7	-	-	-	35.7	-	-	-	-	-

(c) 50% Premium User

correspond to the case where there are less than 90% or 100% of the users that can receive the service in that scenario as a number of users are denied service completely due to congestion.

Increasing number of premium users leads to following consequences:

- While all service options perform similarly for lightly loaded networks, IP-Multicast continues to serve over 90% of its clients even when the network is very heavily congested, albeit, with increased average pre-roll delays.
- Increasing number of premium user leads to decrease in pre-roll delay when the network is normally congested hence, more clients are benefiting from average path behavior.
- In contrast negative aspects of having more premium user is to impose more

traffic load to network. When the network is already congested, adding more traffic makes the condition worse. Hence, we see some increase in pre-roll delay when the network is highly congested.

6.6 PSNR

Peak Signal-to-Noise Ratio (PSNR) is defined as the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. PSNR is commonly used to quantify the quality of a video. The well-known test videos used for PSNR calculation. A representative video frame from these videos are depicted in Figure 20. Videos are selected due to different motion speeds and number of abrupt changes during the video.

(a) Foreman

(b) Football

(c) Soccer

Figure 20: Sample Video Frame from the Test Videos

Due to its large size, a video frame is usually segmented into multiple packets for transmission over the network by the DP. The PSNR value for the video is calculated using the following sequential procedure:

1. Sort the packets which are received for each video frame,
2. Check if any packet/frame is missing,
3. If so, invoke error concealment,
4. Combine packets to populate the video frames,
5. When all frames are generated at the receiver, compare them with their original counterparts.

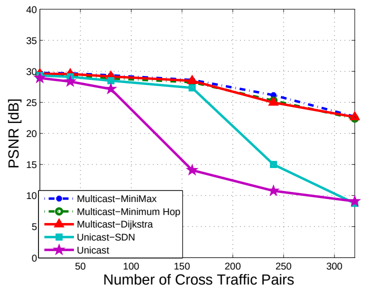
Video compression may be divided into two basic schemes: i) Intra-only compression that completes the compression processing within an individual frame, and ii) Inter (Long GOP) compression in which the processing is completed over multiple frames.

Long GOP compression schemes utilize temporal correlation in order to generate lower bit rates than Intra-only schemes by using the assumption that the picture content of the adjacent frames is similar. However, if the sequence has a low frame correlation between adjacent frames, the bit rate savings generated by using Long GOP compression would be reduced, making it closer to the bit rate generated by using Intra-only coding. Because multiple frames are utilized to exploit temporal correlations when coding a sequence with Long GOP coding, there is a greater processing delay associated with Long GOP coding.

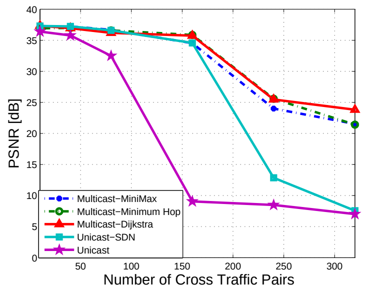
In contrast, the processing delay is lower in Intra-only compression because each frame is coded independently. Editing within post-production is more difficult for Long GOP coding because access to individual frames is complicated since multiple frames are used within the coding. Since Intra-only compression codes each individual frame, recording, editing and manipulation are easily accomplished because the access to each individual image is easy and image quality is kept stable while being totally immune to adjacent frame content influence, which also leads to a lower multi generation deterioration. The quality of a coded sequence using Long GOP coding will worsen much faster than using Intra-only coding over multiple coding generations leading to a greater multi-generation deterioration for Long GOP coding due to the dependency between coded frames. As stated previously, because Intra-only compression codes each frame independently, the quality of a coded sequence over multiple coding generations can be kept higher using Intra-only coding. An error within a frame using Long GOP coding has the possibility to affect several frames leading to greater error propagation when comparing to Intra-only coding. In comparison, an error within a frame using Intra-only coding is contained within a single frame leading to lower error propagation. While parallel processing is used for both coding schemes, parallel processing is more difficult for Long GOP coding due to using multiple frames in the coding process.

In this thesis we assume that video is coded using intra-only compression method therefore, each frame is independently decodable and probabilistic error will not propagate. As a result, if a packet (or all packets corresponding to one frame) is not received, the video player at the client end invokes a simple error concealment procedure in which the missing part/frame is replaced by the preceding video frame.

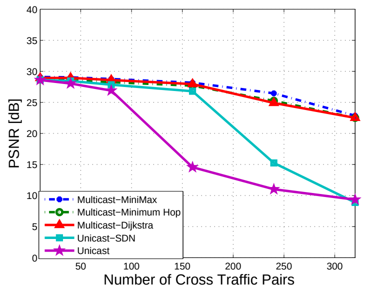
PSNR values for test videos are demonstrated in Figures 21, 22, 23. We observe that for a network with medium load, IP-Multicast, with either of the routing algorithms, provide near lossless video quality of 37 dB and 29 dB, for premium and standard users, respectively. The corresponding non-SDN ALM values on the other hand are very low, indicating a non-watchable video. The PSNR loss with IP-Multicast due to increased congestion remains tolerable throughout, but the ALM performance results in non-watchable videos throughout.



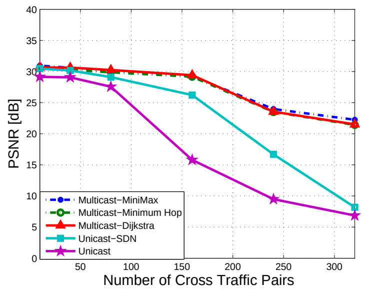
(a) All Users (10%)



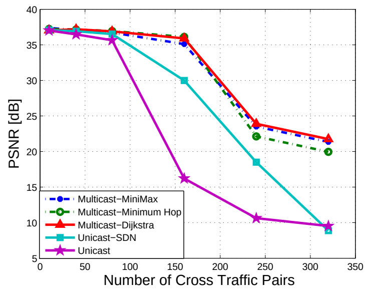
(b) Premium User (10%)



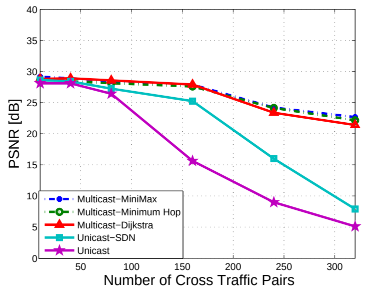
(c) Regular Users (10%)



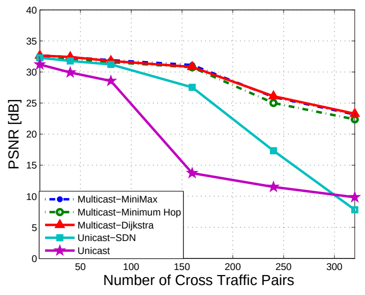
(d) All Users (30%)



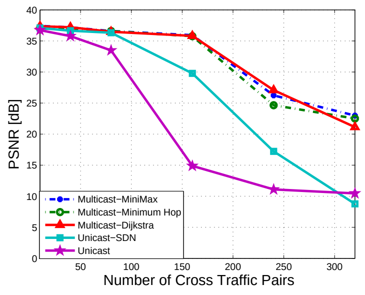
(e) Premium Users (30%)



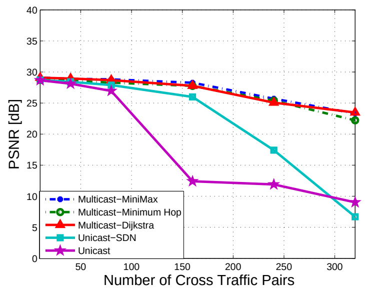
(f) Regular Users (30%)



(g) All Users (50%)

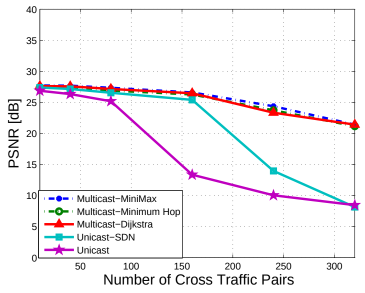


(h) Premium Users (50%)

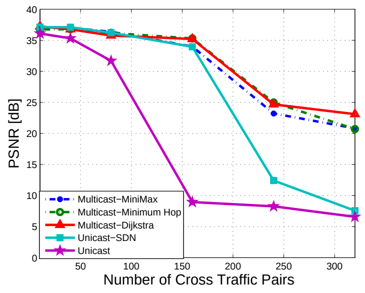


(i) Regular Users (50%)

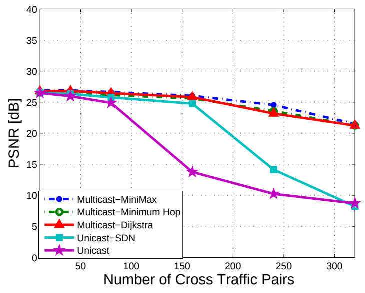
Figure 21: Foreman - PSNR Values



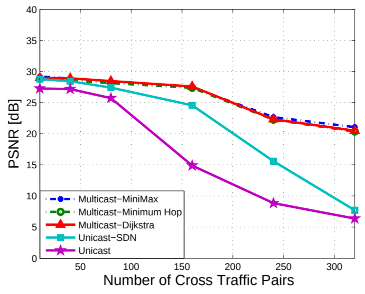
(a) All Users (10%)



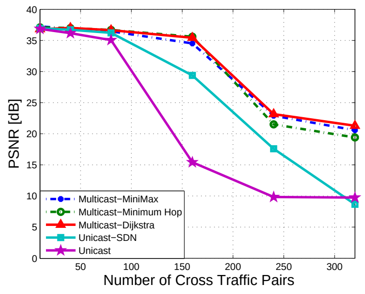
(b) Premium User (10%)



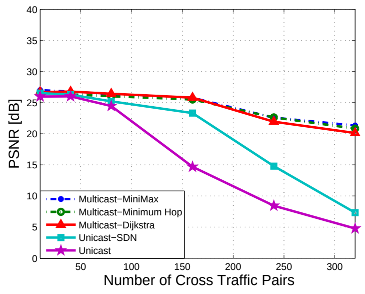
(c) Regular Users (10%)



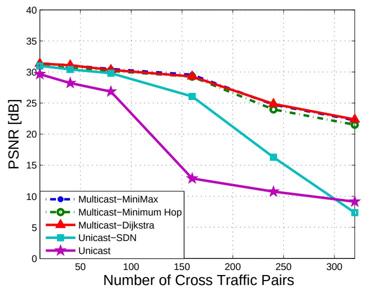
(d) All Users (30%)



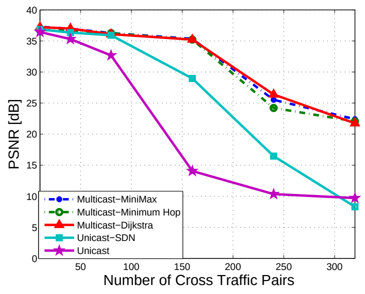
(e) Premium Users (30%)



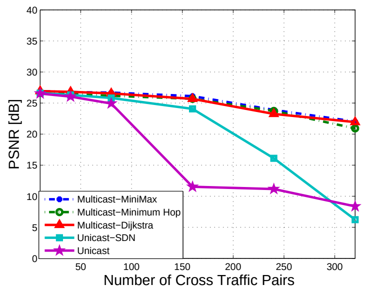
(f) Regular Users (30%)



(g) All Users (50%)

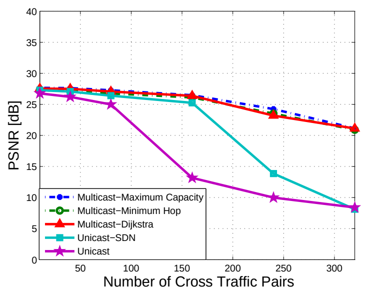


(h) Premium Users (50%)

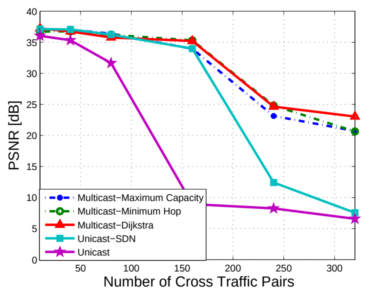


(i) Regular Users (50%)

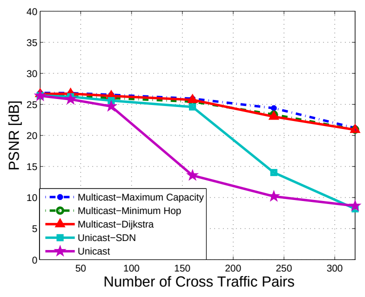
Figure 22: Football - PSNR Values



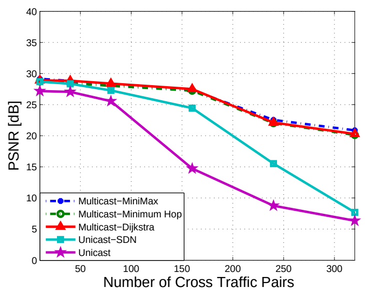
(a) All Users (10%)



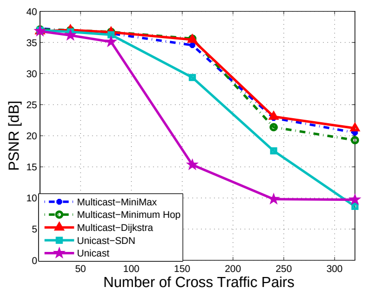
(b) Premium User (10%)



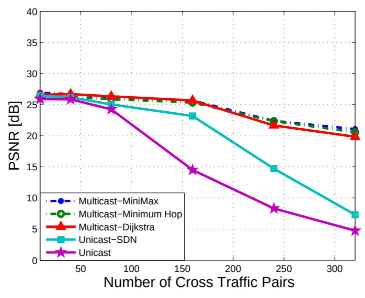
(c) Regular Users (10%)



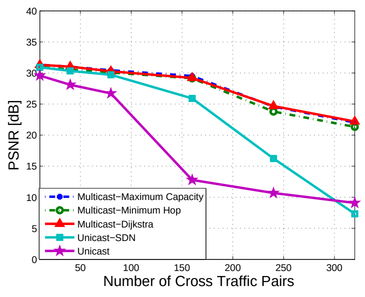
(d) All Users (30%)



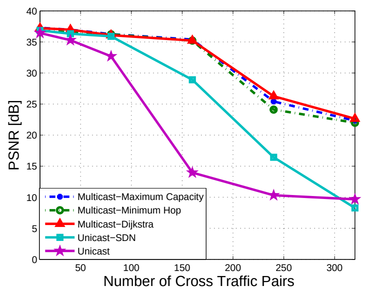
(e) Premium Users (30%)



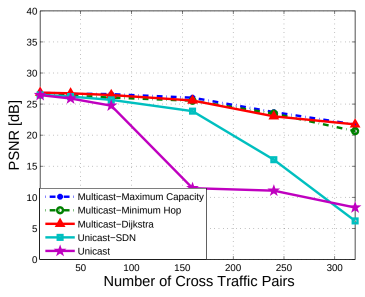
(f) Regular Users (30%)



(g) All Users (50%)



(h) Premium Users (50%)



(i) Regular Users (50%)

Figure 23: Soccer - PSNR Values

CHAPTER VII

CONCLUSION & FUTURE WORKS

In this chapter the conclusion is derived by describing the progress made towards the design and implementation of video streaming framework using SDN. This chapter also suggests some future research directions that could provide the next steps along the path to a practical and widely applicable video streaming system.

7.1 Conclusion

Video has become one of the most prominent applications of the Internet. Many of the video streaming applications involve the distribution of content from a CDN source to a large population of interested clients. However, widespread support of IP-Multicast has been unavailable to a large extent due to technical and economical reasons, all stemming from the non-programmable nature of today's Internet. As a solution, streaming multicast video is commonly operated using ALM. However, this technique introduces excessive delays for the clients and increases traffic load for the network.

This thesis introduces a Multiple-Description Coded streaming video multicast framework that can be easily realized using Software-Defined Networking. We observe that for medium to heavily loaded networks, relative to today's solution of ALM in a non-SDN network, the SDN-based streaming multicast video framework increases the PSNR of the received video significantly, from a level that is practically unwatchable to one that has good quality. Unlike today's solution of ALM, over 90% of the subscribers receive the video service, albeit at a higher pre-roll delay. In addition, SDN-based streaming multicast video framework has significant lower number of denied service, downgrade and pause in comparison to ALM solution both in today's

Internet and SDN structures. We conclude that SDN is a powerful enabler of easily deployable, programmable, powerful network controller, with which, it is possible to observe significant performance gains.

7.2 Future Directions

A number of open problems must be solved to allow the development of a truly general video streaming system. These problems suggest a variety of research directions that need to be pursued to make such a system feasible.

7.2.1 Scalable Video Coding (SVC)

One such direction would be to investigate streaming SVC coded video over implemented framework and compare the results in both MDC and SVC domains. Rate scalability can be elegantly achieved by scalable video codecs [63, 64] that provide layered embedded bit-streams that are decodable at different bitrates. A layered video codec deals with heterogeneity and time-varying nature of the Internet by adapting its bit rate to the available bandwidth and have been widely studied, e.g., in [65, 66]. A scalable representation of video signals consists of a base layer and multiple enhancement layers. The base layer is necessary for the media stream to be decoded whereas enhancement layers are applied to improve stream quality. However, the first enhancement layer depends on the base layer and each enhancement layer n + 1 depends on its subordinate layer n, thus can only be applied if n was already applied.

Scalable video representations aid in TCP-friendly streaming, as they provide a convenient way for performing the rate control required to mitigate network congestion, see, e.g., [67–70]. In receiver-driven layered multicasting [71], video layers are sent in different multicast groups, and rate control is performed individually by each receiver via subscribing to the appropriate groups [72–74].

Hence, media streams using the layered approach are interrupted whenever the base layer is missing and as a consequence, the data of the respective enhancement layers is rendered useless. The same applies for missing enhancement layers. Due to this fragility of layered coding, [75] introduces way to increase robustness of layered coding towards channel errors and lossy conditions. Alternatively, [76–79] propose ways to use differentiated quality of service (DiffServ) [80] to guarantee transmission of base layer packets with improved, but more expensive QoS, while using best-effort to deliver enhancement part. Comparison of MDC and SVC are also widely studied in [81–88].

The designed and implemented system is completely flexible to stream any types of coded videos. The only required effort for this purpose is to simulate different video coding and distribute the contents between servers. Among all different video coding approaches, salable coded video simulation, streaming layers over the implemented framework and finally compare the results with MD coded video could be a valuable milestone for performance comparison of MDC and SVC.

7.2.2 Enhance System Development

Although many aspects considered in framework implementation, there are issues that can complement the system and convert it to comprehensive video streaming system over SDN. Important issues can be enumerated as follow:

- More various types of routing policies can be developed and used for routing regular and multicast packets. The idea can be further developed to have the centralized routing control plane separated from forwarding elements for more flexible, intelligent, and traffic-engineered route control and eliminating the inefficiencies and complexities of traditional routing protocols.
- Implemented system does not differentiate Multimedia packets from regular traffics such as HTTP. A complementary patch for implemented system might

be leverage off OpenFlow's enhanced network control capabilities to deliver multimedia with QoS. QoS implementation helps the current framework to distinguish audio and video packets and route them in a manner to provide required QoS.

- Developing a separate thread to check the multicast trees frequently and optimizes them when it is required.

- Developing IP-Multicast implementation in order to handle client silent leave. Silent leave refers to condition which client leaves the group without informing the multicast group.

Bibliography

[1] D. Wu, Y. Hou, W. Zhu, Y.-Q. Zhang, and J. Peha, “Streaming video over the internet: approaches and directions,” *IEEE Trans. Circuits Syst. Video Technol*, vol. 11, pp. 282–300, Mar. 2001.

[2] R. Jain, “Internet 3.0: Ten problems with current internet architecture and solutions for the next generation,” in *Proc. Military Commun. Conf. (MILCOM)*, (Washington, DC), pp. 1–9, Oct. 2006.

[3] V. Goyal, “Multiple description coding: compression meets the network,” *Signal Processing Mag.*, vol. 18, pp. 74–93, Sept. 2001.

[4] C. S. Inc., “The zettabyte era - trends and analysis,” *White Paper*, May 2013.

[5] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen, “Deployment issues for the IP multicast service and architecture,” *IEEE Network*, vol. 14, pp. 78–88, Jan. 2000.

[6] M. Hosseini, D. Ahmed, S. Shirmohammadi, and N. Georganas, “A survey of application-layer multicast protocols,” *IEEE J. Commun. Surveys and Tutorials*, vol. 9, pp. 58–74, Mar. 2007.

[7] O. N. Foundation, “Software-Defined Networking: The new norm for networks,” *ONF White Paper*, Apr. 2012.

[8] M. Pereira, M. Antonini, and M. Barlaud, “Multiple description coding for internet video streaming,” in *Proc. IEEE Image Process. Conf. (ICIP)*, vol. 3, (Barcelona), pp. 281–284, Sept. 2003.

[9] R. Yang, S. Zheng, and T. Cao, “H.264 based multiple description video coding for internet streaming,” in *Proc. Multimedia Technol. Conf. (ICMT)*, (Ningbo), pp. 1–4, Oct. 2010.

[10] W. Li, “Overview of fine granularity scalability in MPEG-4 video standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 301–317, Mar. 2001.

[11] F. de Asis Lopez Fuentes, “Adaptive mechanism for P2P video streaming using SVC and MDC,” in *Proc. Complex, Intelligent and Software Intensive Systems Conf. (CISIS)*, (Krakow), pp. 457–462, Feb. 2010.

[12] Y. Su, T. Tao, J. Lu, and J. Wang, “Channel-Optimized video transmission over WCDMA system,” in *Proc. Vehicular Technol. Conf. (VTC)*, vol. 1, (Birmingham, Al), pp. 265–269, IEEE, May 2002.

[13] H. Wang and A. Ortega, “Robust video communication by combining scalability and multiple description coding techniques,” in *Proc. SPIE Symp. Electronic Imaging*, (San Jose, CA), pp. 111–124, International Society for Optics and Photonics, May 2003.

[14] P. A. Chou, H. J. Wang, and V. N. Padmanabhan, “Layered multiple description coding,” in *Proc. Packet Video Workshop*, vol. 7, (Nantes), Apr. 2003.

[15] Y. Wang, A. Reibman, and S. Lin, “Multiple description coding for video delivery,” *IEEE J. Proceedings*, vol. 93, pp. 57–70, Jan. 2005.

[16] A. Reibman, H. Jafarkhani, Y. Wang, M. Orchard, and R. Puri, “Multiple description coding for video using motion compensated prediction,” in *Proc. IEEE Image Process. Conf. (ICIP)*, vol. 3, (Kobe), pp. 837–841 vol.3, Oct. 1999.

[17] D. Comas, R. Singh, and A. Ortega, “Rate-distortion optimization in a robust video transmission based on unbalanced multiple description coding,” in *Proc. IEEE 4th Workshop on Multimedia Signal Process. (MMSP)*, (Cannes), pp. 581–586, Oct. 2001.

[18] Y. Wang and S. Lin, “Error-resilient video coding using multiple description motion compensation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 438–452, Jun 2002.

[19] Y.-C. Lee and Y. Altunbasak, “A collaborative multiple description transform coding and statistical error concealment method for error resilient video streaming over noisy channels,” in *Proc. IEEE Acoustics, Speech, and Signal Process. Conf. (ICASSP)*, vol. 2, (Orlando, FL), pp. 2077–2080, May 2002.

[20] T. Chan and S.-W. Ho, “Robust multiple description coding; Joint coding for source and storage,” in *Proc. IEEE Symp. Inform. Theory Proc. (ISIT)*, (Istanbul), pp. 1809–1813, July 2013.

[21] I. Radulovic, P. Frossard, Y.-K. Wang, M. Hannuksela, and A. Hallapuro, “Multiple description video coding with H.264/AVC redundant pictures,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, pp. 144–148, Jan. 2010.

[22] M. Kazemi, K. Sadeghi, and S. Shirmohammadi, “A high video quality multiple description coding scheme for lossy channels,” in *Proc. IEEE Multimedia and Expo Conf. (ICME)*, (Barcelona), pp. 1–6, July 2011.

[23] J. Apostolopoulos and M. Trott, “Path diversity for enhanced media streaming,” *IEEE Commun. Mag.*, vol. 42, pp. 80–87, Aug. 2004.

[24] J. G. Apostolopoulos, “Reliable video communication over lossy packet networks using multiple state encoding and path diversity,” in *Proc. SPIE Electronic Imaging*, (Seattle, WA), pp. 392–409, International Society for Optics and Photonics, Jan. 2000.

[25] J. Apostolopoulos and S. Wee, “Unbalanced multiple description video communication using path diversity,” in *Proc. Image Process. Conf. (ICIP)*, vol. 1, (Thessaloniki), pp. 966–969 vol.1, Oct. 2001.

[26] P. Correia, P. A. Amado Assuncao, and V. Silva, "Multiple description video streaming over asymmetric channels," in *Proc. IEEE Packet Video Workshop*, (San Jose, CA), pp. 1–6, Dec. 2013.

[27] N. Gogate, D.-M. Chung, S. Panwar, and Y. Wang, "Supporting image and video applications in a multihop radio environment using path diversity and multiple description coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 777–792, Sept. 2002.

[28] Y. Liang, E. Setton, and B. Girod, "Channel-adaptive video streaming using packet path diversity and rate-distortion optimized reference picture selection," in *IEEE Workshop on Multimedia Signal Process. (MMSP)*, (Virgin Islands), pp. 420–423, Dec. 2002.

[29] Y. Liao and J. Gibson, "Routing-aware multiple description video coding over wireless ad-hoc networks using multiple paths," in *Proc. IEEE Image Process. Conf. (ICIP)*, (Hong Kong), pp. 1265–1268, Sept. 2010.

[30] P. Xia, S.-H. Chan, and X. Jin, "Optimal bandwidth assignment for Multiple-Description-Coded video," *IEEE Trans. Multimedia*, vol. 13, pp. 366–375, Apr. 2011.

[31] A. Begen, Y. Altunbasak, and O. Ergun, "Multi-Path selection for multiple description encoded video streaming," in *Proc. IEEE Commun. Conf. (ICC)*, vol. 3, (Anchorage, AK), pp. 1583–1589, May 2003.

[32] Z. Ma, H.-R. Shao, C. Shen, *et al.*, "A new multi-path selection scheme for video streaming on overlay networks," in *Proc. IEEE Commun. Conf. (ICC)*, (Paris), pp. 1330–1334, Jun 2004.

[33] N. N. Qadri, A. Liotta, M. Altaf, M. Fleury, and M. Ghanbari, "Effective video streaming using mesh p2p with mdc over manets.," *Citeseer J. Mobile Multimedia*, vol. 5, no. 4, pp. 301–316, 2009.

[34] T. P. Nguyen and A. Zakhor, "Distributed video streaming over internet," in *Proc. SPIE Multimedia Computing and Networking (MMCN)*, (San Jose, CA), pp. 186–195, International Society for Optics and Photonics, Jan. 2002.

[35] J. Kim, R. Mersereau, and Y. Altunbasak, "Distributed video streaming using unbalanced multiple description coding and forward error correction," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, vol. 6, (San Francisco, CA), pp. 3553–3557 vol.6, Dec. 2003.

[36] T. Nguyen and A. Zakhor, "Multiple sender distributed video streaming," *IEEE Trans. Multimedia*, vol. 6, pp. 315–326, Apr. 2004.

[37] T. Nguyen, P. Mehra, and A. Zakhor, "Path diversity and bandwidth allocation for multimedia streaming," in *Proc. IEEE Multimedia and Expo Conf. (ICME)*, vol. 1, (Baltimore, MD), pp. 1–4, July 2003.

[38] T. Nguyen and A. Zakhor, "Path diversity with forward error correction (PDF) system for packet switched networks," in *Proc. IEEE INFOCOM*, vol. 1, (San Francisco, CA), pp. 663–672, Mar. 2003.

[39] J. Apostolopoulos, W. tian Tan, and S. Wee, "Performance of a multiple description streaming media content delivery network," in *Proc. IEEE Image Process. Conf. (ICIP)*, vol. 2, (Rochester, NY), pp. II–189–II–192, Sept. 2002.

[40] X. Xu, Y. Wang, S. Panwar, and K. Ross, "A peer-to-peer video-on-demand system using multiple description coding and server diversity," in *Proc. IEEE Image Process. Conf. (ICIP)*, vol. 3, (Singapore), pp. 1759–1762, Oct. 2004.

[41] E. Akyol, A. Tekalp, and M. Civanlar, "A flexible multiple description coding framework for adaptive peer-to-peer video streaming," *IEEE J. Sel. Topics Signal Process.*, vol. 1, pp. 231–245, Aug. 2007.

[42] V. Padmanabhan, H. Wang, and P. Chou, "Resilient peer-to-peer streaming," in *Proc. IEEE Network Protocols Conf.*, (Atlanta, GA), pp. 16–27, Nov. 2003.

[43] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. ACM Workshop Network and Operating Systems Support for Digital Audio and Video. (NOSSDAV)*, (Miami, FL), pp. 177–186, May 2002.

[44] P. Frossard, J. De Martin, and M. Civanlar, "Media streaming with network diversity," *IEEE J. Proceedings*, vol. 96, pp. 39–53, Jan. 2008.

[45] S. Deering, "Host extensions for IP multicasting." RFC 1112 (INTERNET STANDARD), Aug. 1989. Updated by RFC 2236.

[46] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Mar. 2008.

[47] H. Egilmez, S. Dane, K. Bagci, and A. Tekalp, "OpenQoS: An openflow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks," in *Proc. Asia-Pacific Signal Inform. Process. Assoc. Annu. Summit and Conf. (APSIPA ASC)*, (Hollywood, CA), pp. 1–8, Dec. 2012.

[48] H. Egilmez, B. Gorkemli, A. Tekalp, and S. Civanlar, "Scalable video streaming over openflow networks: An optimization framework for QoS routing," in *Proc. IEEE Image Process. Conf. (ICIP)*, (Melbourne), pp. 2241–2244, Sept. 2011.

[49] H. Egilmez, S. Civanlar, and A. Tekalp, "An optimization framework for QoS-Enabled adaptive video streaming over openflow networks," *IEEE Trans. Multimedia*, vol. 15, pp. 710–715, Apr. 2013.

[50] S. Civanlar, M. Parlakisik, A. Tekalp, B. Gorkemli, B. Kaytaz, and E. Onem, "A QoS-enabled openflow environment for scalable video streaming," in *Proc. IEEE Global Telecommun. Workshop (GLOBECOM)*, (Miami, FL), pp. 351–356, Dec. 2010.

[51] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran-Gia, "SDN-based application-aware networking on the example of youtube video streaming," in *Proc. European Workshop on Software Defined Networks (EWSDN)*, (Berlin), pp. 87–92, Oct. 2013.

[52] Y. Nakagawa, K. Hyoudou, and T. Shimizu, "A management method of IP multicast in overlay networks using openflow," in *Proc. Hot Topics in Software Defined Networks (HOtSDN)*, (Helsinki), pp. 91–96, ACM, 2012.

[53] F. Coras, J. Domingo-Pascual, F. Maino, D. Farinacci, and A. Cabellos-Aparicio, "Lcast: Software-defined inter-domain multicast," *Elsevier J. Comput. Networks*, vol. 59, pp. 153–170, Feb 2013.

[54] D. Kotani, K. Suzuki, and H. Shimonishi, "A design and implementation of OpenFlow controller handling IP multicast with fast tree switching," in *Proc. IEEE Symp. Applicat. and the Internet (SAINT)*, (Izmir), pp. 60–67, July 2012.

[55] L. Bondan, L. F. Müller, and M. Kist, "Multiflow: Multicast clean-slate with anticipated route calculation on OpenFlow programmable networks," *Elsevier J. Applied Computing Research*, vol. 2, no. 2, pp. 68–74, 2013.

[56] A. Adams, J. Nicholas, and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)." RFC 3973 (Experimental), Jan. 2005.

[57] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)." RFC 4601 (Proposed Standard), Aug. 2006. Updated by RFCs 5059, 5796, 6226.

[58] S. Bhattacharyya, "An Overview of Source-Specific Multicast (SSM)." RFC 3569 (Informational), July 2003.

[59] M. Handley, I. Kouvelas, T. Speakman, and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)." RFC 5015 (Proposed Standard), Oct. 2007.

[60] P.-R. Sheu and S.-T. Chen, "A fast and efficient heuristic algorithm for the delay- and delay variation-bounded multicast tree problem," *J. Computer Communications*, vol. 25, no. 8, pp. 825 – 833, 2002.

[61] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. USENIX Workshop Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, vol. 54, (San Jose, CA), Apr. 2012.

[62] D. Erickson, "The beacon openflow controller," in *Proc. ACM SIGCOMM 2nd workshop on Hot topics in software defined networking*, (Hong Kong), pp. 13–18, ACM, Aug. 2013.

[63] M. Ghanbari, "Two-layer coding of video signals for VBR networks," *IEEE J. Sel. Areas Commun.*, vol. 7, pp. 771–781, Jun 1989.

[64] B.-J. Kim, Z. Xiong, and W. Pearlman, "Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D spiht)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 1374–1387, Dec. 2000.

[65] M. Khansari, A. Zakauddin, W.-Y. Chan, E. Dubois, and P. Mermelstein, "Approaches to layered coding for dual-rate wireless video transmission," in *Proc. IEEE Image Process. Conf. (ICIP)*, vol. 1, pp. 258–262 vol.1, Nov. 1994.

[66] H. Radha, Y. Chen, K. Parthasarathy, and R. Cohen, "Scalable internet video using mpeg-4," *Elsevier J. Signal Processing: Image Communication*, vol. 15, no. 1, pp. 95–126, 1999.

[67] J. Mahdavi, "TCP-friendly unicast rate-based flow control," *Technical note sent to the end2end-interest mailing list*, Jan. 1997.

[68] W. tian Tan and A. Zakhor, "Internet video using error resilient scalable compression and cooperative transport protocol," in *Proc. IEEE Image Process. Conf. (ICIP)*, (Chicago, IL), pp. 458–462 vol.3, Oct. 1998.

[69] W. tian Tan and A. Zakhor, "Real-time internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Trans. Multimedia*, vol. 1, pp. 172–186, Jun 1999.

[70] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," *SIGCOMM Comput. Commun.*, vol. 30, pp. 43–56, Aug. 2000.

[71] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," *SIGCOMM Comput. Commun. Rev.*, vol. 26, pp. 117–130, Aug. 1996.

[72] S. McCanne, M. Vetterli, and V. Jacobson, "Low-complexity video coding for receiver-driven layered multicast," *IEEE J. Sel. Areas Commun.*, vol. 15, pp. 983–1001, Aug. 1997.

[73] W.-t. Tan and A. Zakhor, "Multicast transmission of scalable video using receiver-driven hierarchical FEC," in *Proc. Packet Video Workshop*, vol. 99, (New York, NY), Apr. 1999.

[74] P. Chou, A. Mohr, A. Wang, and S. Mehrotra, "Error control for receiver-driven layered multicast of audio and video," *IEEE Trans. Multimedia*, vol. 3, pp. 108–122, Mar. 2001.

[75] U. Horn, K. Stuhlmüller, M. Link, and B. Girod, "Robust internet video transmission based on scalable coding and unequal error protection," *Elsevier J. Signal Process. Image Commun.*, vol. 15, no. 1, pp. 77–94, 1999.

[76] E. Masala, D. Quaglia, and J. De Martin, "Adaptive picture slicing for distortion-based classification of video packets," in *IEEE 4th Workshop Multimedia Signal Process.*, pp. 111–116, Oct. 2001.

[77] J. Shin, J. Kim, and C.-C. Kuo, "Relative priority based QoS interaction between video applications and differentiated service networks," in *Proc. IEEE Image Process. Conf. (ICIP)*, vol. 3, pp. 536–539 vol.3, 2000.

[78] J. Shin, J. Kim, and C.-C. Kuo, "Quality-of-Service mapping mechanism for packet video in differentiated services network," *IEEE Trans. Multimedia*, vol. 3, pp. 219–231, Jun 2001.

[79] D. Quaglia and J. De Martin, "Delivery of MPEG video streams with constant perceptual quality of service," in *Proc. Multimedia and Expo Conf. (ICME)*, vol. 2, (Lausanne), pp. 85–88, Aug. 2002.

[80] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services." RFC 2475 (Informational), Dec. 1998. Updated by RFC 3260.

[81] Y. Wang, S. Panwar, S. Lin, and S. Mao, "Wireless video transport using path diversity: multiple description vs layered coding," in *Proc. IEEE Image Process. Conf. (ICIP)*, vol. 1, (Rochester, NY), pp. I–21–I–24 vol.1, Sept. 2002.

[82] J. Chakareski, S. Han, and B. Girod, "Layered coding vs. multiple descriptions for video streaming over multiple paths," *Springer J. Multimedia Systems*, vol. 10, no. 4, pp. 275–285, 2005.

[83] A. Reibman, H. Jafarkhani, M. Orchard, and Y. Wang, "Performance of multiple description coders on a real channel," in *Proc. Acoustics, Speech, and Signal Processing Conf.*, vol. 5, (Phoenix, AZ), pp. 2415–2418, Mar. 1999.

[84] R. Singh, A. Ortega, L. Perret, and W. Jiang, "Comparison of multiple description coding and layered coding based on network simulations," in *Proc. SPIE Image Video Proc.*, (San Jose, CA), pp. 929–939, Jan. 2000.

[85] A. R. Reibman, Y. Wang, X. Qiu, Z. Jiang, and K. Chawla, "Transmission of multiple description and layered video over an EGPRS wireless network," in *Proc. IEEE Image Process. Conf. (ICIP)*, vol. 2, (Vancouver, BC), pp. 136–139, IEEE, Sept. 2000.

[86] Y.-C. Lee, J. Kim, Y. Altunbasak, and R. M. Mersereau, "Performance comparisons of layered and multiple description coded video streaming over error-prone networks," in *Proc. IEEE Commun. Conf. (ICC)*, vol. 1, (Anchorage, AK), pp. 35–39, IEEE, May 2003.

[87] F. Mogus, “Performance comparison of multiple description coding and scalable video coding,” in *Proc. IEEE Commun. Software and Networks Conf. (ICCSN)*, (Xi’an), pp. 452–456, May 2011.

[88] Y.-H. Chiang, P. Huang, and H. Chen, “SVC or MDC? That’s the question,” in *Proc. IEEE Symp. Embedded Systems for Real-Time Multimedia (ESTIMedia)*, (Taipei), pp. 76–82, Oct. 2011.

VITA

Kyoomars Alizadeh Noghani was born in Tehran, Iran, on September 20, 1988. After completing his degree at Khatam High School, in 2006, he entered the University of Tehran, Iran's oldest modern university, where he received the degree of Bachelor of Information Technology in September, 2011. He entered The Graduate School of Engineering and Science of Özyeğin University, Istanbul, in February, 2012. During his stay in OZU he was a member of WiserLab.