Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

LIVE MEDIA PRODUCTION: MULTICAST OPTIMIZATION AND VISIBILITY FOR CLOS FABRIC IN MEDIA DATA CENTERS

by
Ammar Latif

Media production data centers are undergoing a major architectural shift to introduce digitization concepts to media creation and processing workflows. Content companies such as NBC Universal, CBS/Viacom and Disney are modernizing their workflows to take advantage of the flexibility of IP and virtualization.

In these new environments, multicast is utilized to provide point-to-multi-point communications. In order to build point-to-multi-point trees, Multicast has an established set of control protocols such as $IGMP$ and PIM. The existing multicast protocols do not optimize multicast tree formation for maximizing network throughput. This leads to decreased fabric utilization and decreased total number of admitted flows. In addition, existing multicast protocols are not bandwidth-aware and could cause links to over-subscribe leading to packet loss and lower video quality.

TV production traffic patterns are unique due to ultra high bandwidth requirements and high sensitivity to packet loss that leads to video impairments. In such environments, operators need monitoring tools that are able to proactively monitor video flows and provide actionable alerts. Existing network monitoring tools are inadequate because they are reactive by design and perform generic monitoring of flows with no insights into video domain.

The first part of this dissertation includes a design and implementation of a novel Intelligent Rendezvous Point algorithm iRP for bandwidth-aware multicast routing in media DC fabrics. iRP utilizes a controller-based architecture to optimize multicast tree formation and to increase bandwidth availability in the fabric. The

system offers up to 50% increase in fabric capacity to handle multicast flows passing through the fabric.

In the second part of this dissertation, $DiRP$ algorithm is presented. $DiRP$ is based on a distributed decision-making approach to achieve multicast tree capacity optimization while maintaining low multicast tree setup time. $DiRP$ algorithm is tested using commercially available data center switches. $DiRP$ algorithm offers substantially lower path setup time compared to centralized systems while maintaining bandwidth awareness when setting up the fabric.

The third part of this dissertation studies the utilization of machine learning algorithms to improve on multicast efficiency in the fabric. The work includes implementation and testing of $LiRP$ algorithm to increase $iRP's$ fabric efficiency. This is done by implementing k-fold cross validation method to predict future multicast group memberships for time-series analysis. Testing results confirm that $LiRP$ system increases the efficiency of iRP by up to 40% through prediction of multicast group memberships with online arrival.

In the fourth part of this dissertation, The problem of live video monitoring is studied. Existing network monitoring tools are either reactive by design or perform generic monitoring of flows with no insights into video domain. $MediaFlow$ is a robust system for active network monitoring and reporting of video quality for thousands of flows simultaneously using a fraction of the cost of traditional monitoring solutions. $MediaFlow$ is able to detect and report on integrity of video flows at a granularity of 100 mSec at line rate for thousands of flows. The system increases video monitoring scale by a thousand fold compared to edge monitoring solutions.

LIVE MEDIA PRODUCTION: MULTICAST OPTIMIZATION AND VISIBILITY FOR CLOS FABRIC IN MEDIA DATA CENTERS

by
Ammar Latif

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Helen and John C. Hartmann Department of
Electrical and Computer Engineering

August 2020

APPROVAL PAGE

LIVE MEDIA PRODUCTION:
MULTICAST OPTIMIZATION AND VISIBILITY FOR CLOS FABRIC IN MEDIA DATA CENTERS

Ammar Latif

Dr. Abdallah Khreishah, Dissertation Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Nirwan Ansari, Committee Member — Date
Distinguished Professor of Electrical and Computer Engineering, NJIT

Dr. Cristian Borcea, Committee Member — Date
Professor of Computer Science, NJIT

Dr. Qing Liu, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Ammar Rayes, Committee Member — Date
Distinguished Engineer, Cisco Systems

BIOGRAPHICAL SKETCH

Author: Ammar Latif

Degree: Doctor of Philosophy

Date: August 2020

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, New Jersey, 2020
- Master of Engineering in Telecommunications,
  University of Toronto, Toronto, Canada 2001
- Masters in Business Administration,
  Concordia University, Montreal , Canada, 2011

Major: Electrical Engineering

Presentations and Publications:

A. Latif and R. Parameswaran and S. Vishwarupe and A. Khreishah, "MediaFlow : Professional Media In-Network Video Flow Multicast Monitoring," *Submitted for publication.*

A. Latif and P. Kathail and S. Vishwarupe and S. Dhesikan and A. Khreishah and Y. Jararweh, "Multicast Optimization for CLOS Fabric in Media Data Centers," *IEEE Transactions on Network and Service Management*, Vol, 16, no. 4, 1855-1868, 2019.

A. Latif and P. Paul and R. Chhibber and A. Singh and R. Parameswaran and A. Khreishah and Y. Jararweh, "DiRP: Distributed Intelligent Rendezvous Point for Multicast Control Plane," *IEEE International Green and Sustainable Computing Conference*, 2018.

A. Latif and P. Kathail and S. Vishwarupe and S. Dhesikan and A. Khreishah, "iRP: Intelligent Rendezvous Point for Multicast Control Plane," *IEEE 38th Sarnoff Symposium*, 2017.

A. Latif and T. Ohanian, "Software-Defined Network for Media Workflows," *The Society of Motion Picture and Television Engineers Annual Technical Conference*, 2013.

To the memory of my father, Firas; I will forever be grateful for your believe in me. Your kindness to others and devotion to something bigger than oneself is an inspiration.

To my dear mother, Nawal; thanks for being my example of perseverance, hard-work, and self reliance.

To my dear wife, Sarab; it was your support that gave me the ability and the strength needed to complete this work.

To my young children Sarah Rose and Adam Ferris, you bring joy and happiness to your parents, guess who is up next for a PhD.....

ACKNOWLEDGMENT

I am eternally grateful to my dear advisor, Dr. Abdallah Khreishah, for taking the risk and investing his time to supervise a student coming from the industry . His research guidance, encouragement and overall support were paramount in each step of my PhD journey.

I would like to express my appreciation to my dissertation committee: Dr. Nirwan Ansari, Dr. Cristian Borcea, Dr. Qing Liu and Dr. Ammar Rayes for their time and for their valuable feedback. I am also thankful to the Department of Electrical and Computer Engineering for giving me the opportunity to pursue my Ph.D. degree at New Jersey Institute of Technology.

I have been fortunate to be part of a great community of brilliant engineers at Cisco Systems media team who provided me with lots of great insight and feedback. Special thanks to Rahul Parameswaran, Sachin Vishwarupe, Sunil Gudurvalmiki, Subha Dhesikan and Pradeep Kathail for their collaboration, ideas and inputs to my research.

I am thankful for my NJIT colleagues Dr. Adel Aldalbahi, Dr. Hazim Shakhatreh, Dr. Ammar Gharaibeh and Dr. Sihua Shao for sharing their experiences and valuable tips that helped me navigate the PhD journey.

I would also like to thank my friend since middle school, Dr. Marwan Younis, who taught me the ropes of latex and did the initial review of my first published paper.

I would like to thank my big and warm family, always extremely proud of my personal and professional accomplishments. To my mother, my wife, my children, my siblings, my in-laws and my extended family: your support and love have been paramount to me, and I would never have made it here without each and everyone of you.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

In summer 2012, the author visited his first media production data center (DC). The facility was owned by a major broadcaster located in Washington, D.C. Metropolitan area and it looked like a data center but was very different. To the surprise of the author, there were plenty of thick coax cables in the facility. The cables carried live video signal throughout the facility using Serial Digital Interface (SDI) format which is neither Ethernet nor IP based. The facility did not have much Ethernet and fiber cables. In addition, there were many purpose built devices with no server farms nor virtualization. That visit triggered the author's curiosity to understand the workflow and study methods to modernize the technology stack which is the effort that lead to this dissertation.

Figure 1.1 Serial Digital Interface.

1.1 Introduction to Media Workflows

Media production and processing workflows are going through massive digitization wave driven by adoption of IP and virtualization technologies. This adoption is

enabling content providers to have more agile and responsive business in the internet era. One of the main topics in media workflow digitization is the migration to IP-based transport of uncompressed video from legacy non-IP infrastructure based on SDI [101, 96, 97]. Utilising IP for transport of video signal would enable the content providers to benefit from cloud and virtualization. In this work, we identify real life challenges with this transition and propose solutions to address them [30, 55].

Global broadcasters such as NBC Universal, CBS/Viacom and Disney are embracing IP and virtualization technology when building new media production facilities. A good example is NBC10 TV station that was built in the Comcast Technology Center in Philadelphia, USA [103]. The NBC10 TV station was one of the first TV stations to utilize IP and virtualization technologies.

Figure 1.2 With the latest in IP infrastructure and storytelling technology, NBCUniversal in Philadelphia has completed its on-air move into the Comcast Technology Center.

In live media production workflows, multicast is widely deployed to provide point to multi-point communications over IP networks. Multicast is popular in transport of multimedia content [80, 68, 100, 65], as it allows the source to send a single copy of the content and let the network handle the replication of the packets to multiple destinations. This leads to lower host and bandwidth utilization usage in the network. Multicast traffic is delivered from the source to multiple destinations over a multicast tree [91]. Such trees are constructed using various methods depending

on the desired outcome such as minimizing delay, maximizing bandwidth utilization or increasing multicast tree scaling [87, 91, 71].

Multicast routing protocols attempt to find a path or a tree linking the source to the set of destinations. Researchers are frequently interested in finding optimal trees connecting sources and destinations while optimizing certain attributes such as cost and/or delay. The resulting problem is called the multicast routing tree problem (MRP) [91, 80, 65]. There are many cost functions to consider based on the application requirements. Some of the popular cost functions are distance, end to end delay, traffic concentration, bandwidth utilized and tree setup time [87, 91, 71]. Multicast tree cost attribute reflects the combined cost metric to establish the tree in the network. The end-to-end network delay attribute is important for delay-sensitive applications such as live media production. Minimizing traffic concentration in parts of the network is a useful measure in applications that do not tolerate over-subscription of links and resulting packet loss [58, 100]. In such environments, traffic concentration can lead to blocking scenarios in some parts of the network while network capacity is available in other parts. Bandwidth utilization is important in applications that require guaranteed quality of service through the network. It is also reflective of total capacity in the network required to service the multicast trees. Path setup time is important for real time applications such as video conferencing or IPTV applications where the receiver needs to receive needed content quickly after it sends a request for such content [24].

Multicast trees are classified into shared trees and source trees. In a shared tree, a single tree is constructed to connect all nodes in a fabric [19, 92, 105, 74]. This approach minimizes path setup time as the tree is already created. Core based trees and Steiner trees are examples of shared trees where traffic from all multicast groups flow through the same shared tree [113]. Shared trees lead to sub-optimal tree path for each of the groups, higher delay and traffic concentration on main paths (trunk)

of the shared tree. Source trees on the other hand are created for each source of a group [74, 19]. They provide lower end to end delay, higher optimization and lower concentration of traffic as groups do not need to share a path. source trees require additional setup time per tree and put more demand on network resources to track the state of each tree. Multicast routing problems are classified under the family of Steiner trees. This set of problems have been extensively researched in the literature [113, 74, 19]

Congestion and over-subscription in multicast networks lead to packet loss [71, 72]. Not all applications are able to handle packet loss in a multicast environment. This is especially true since multicast communication is unidirectional and utilizes connectionless protocols such as udp with no packet recovery mechanism [72]. Packet loss is an important topic for real time multimedia applications because it leads to lost video and audio frames. This loss cannot be recovered and impacts artistic content creation process. In the literature, this problem is usually studied as a constrained Steiner tree problem (CST). CST problem has been proven to be NP-Complete [113].

Multicast routing algorithms can be categorized into distributed or centralized algorithms based on the decision-making process to create the multicast trees [87, 80]. In distributed algorithms, the multicast tree is constructed at the network nodes level on a hop by hop basis. Popular algorithms include PIM [1] and $MOSPF$ [76]. PIM is widely deployed in IP networks due to its versatility in creating multicast trees [80, 65]. With the advancement of software defined networks (SDN) and SDN controller concepts [82], there are proposed algorithms that utilize central controller to maintain network state as well as build multicast trees based on its central view of the network [10]. Distributed systems do not need to maintain network state in a central node and can converge from error states faster. However, many of the distributed systems focus on a single metric such as the shortest path and do not take bandwidth nor delay requirements into consideration. Centralized systems can

construct trees with optimization based on multiple factors such as end to end delay, over-subscription of links and available bandwidth [22].

Optimization techniques for multicast tree calculation utilize well known graph theories [9, 8, 90]. Multicast routing problems are known to be NP-Hard [108, 119]. As such, multicast algorithms use approximation techniques to address NP-hardness attribute related to multicast routing. Multicast algorithms that utilize optimization tools can be classified into online and offline algorithms based on methods used to build a multicast tree. Offline methods assume that the multicast demand matrix (groups and destinations) is known in advance and consider tree creation as an optimization problem. Offline methods create the most optimal multicast tree for the targeted metric [9]. The challenge with offline methods is that tree route calculation is an NP hard problem. Even with approximation methods, calculation time is exponential to both the size of the network and the required tree. This leads to unacceptable response time for many applications. Assuming full knowledge of demand matrix is also not realistic for many applications as multicast sources and group members are dynamic in nature. Online optimization algorithms do not assume full knowledge of Multicast traffic patterns and demand matrix. Rather, they attempt to handle multicast routing as the sources and destinations arrive in an online fashion [27, 66, 22, 47]. Online algorithms achieve lower performance compared to offline optimization due to their lack of full knowledge.

1.2 Research Impact

A key motivation for this work is the transition happening in the uncompressed Video transport technologies from legacy non-IP format based on Serial Digital Interface (SDI) [98] to transport based on IP. Work in the standards bodies is currently underway around defining standards and relevant encapsulations. The work in [30] and [55] provide an overview of the live video industry's current challenges, activities and forward-looking direction to migrate to IP based video flows. Multicast is a key

part of underlying infrastructure needed to support the industry activities. However, current multicast implementations (specifically in PIM, which is the most widely deployed multicast routing protocol) are not able to address the industry requirements around maximizing overall number of flows and capacity of the fabric while avoiding over-subscription of links.

The traffic patterns of uncompressed video within a Broadcast Studio (TV stations, Sports Venues, etc.) consist of video flows that are very large in throughput requirements. Table 1.1 shows the rates for uncompressed video which could have throughput of 1.5Gbps, 3Gbps and 12Gbps. These requirements are expected to grow as video formats are evolving [99]. With such high throughput, three or six of these video streams (depending on rates) can saturate a 10G link. These video flows do not tolerate packet loss as it severely impacts the delivered video quality leading to image pixilation or total loss of image [58]. Another characteristic of an uncompressed video flow is being long-lived with constant bit rate (CBR) profile [100]. Such flow is usually based on UDP with no expectation for packet re-transmission as a late packet is as good as a lost packet in live video applications.

Table 1.1 Uncompressed Video Data Rates

VIDEO FORMAT	DATA RATES in Mbps [101]
Standard Definition SD-SDI	270
High Definition HD-SDI 1080i	1485
3G-SDI 1080p	2970

For multicast tree construction problem, a popular optimization focuses on finding a routing path or a tree linking the source to the set of destinations, while simultaneously minimizing some cost function. Researchers have been interested in finding trees connecting sources and destinations with optimization for a network attribute such as total path cost or delay. The resulting problem is called the *multicast routing tree problem* (MRP) [91]. There are many cost functions that are relevant depending on the application requirements. Some of the popular cost

functions include distance [87], end to end delay [91], traffic concentration, bandwidth utilized and/or tree setup time [71]. Tree total cost reflects the combined cost metric to establish the tree in the network. End to end delay is important for delay-sensitive applications [91]. Minimizing traffic concentration in parts of the network is a useful measure in applications that do not tolerate over-subscription of links and resulting packet loss. In such environments, traffic concentration can lead to blocking scenarios in parts of the network while network capacity is available in other parts of the network. Bandwidth utilization is important in applications that require guaranteed quality of service through the network [91]. It is also reflective of total capacity in the network to service the required multicast trees. Path setup time is important for real time applications such as video conferencing or IPTV applications where the receiver needs to receive needed content quickly after it sends a request for such content [24].

In state of the art data center (DC) deployments, multicast implementations are based on Protocol Independent Multicast (PIM) defined in RFC7761 [35]. PIM is a family of protocols that defines various delivery mechanisms for one-to-many and many-to-many communications over IP. PIM Sparse mode ($PIM-SM$) is one of the protocols in the PIM family that focuses on building shared trees as well as creating source specific trees. PIM relies on existing unicast routing protocols such as OSPF [77] for gathering routing information about the network. $PIM-SM$ uses the pull model where traffic is sent on a link only if we receive an explicit join on that interface. $PIM-SM$ guarantees that multicast traffic is limited to the part of the network where receivers reside. In contrast to the PIM Dense-Mode ($PIM-DM$) [1] which utilizes the push model, $PIM-SM$ prevents flooding in the network. $PIM-SM$ requires the definition of a Rendezvous Point (RP) to play the role of the root of the shared tree. Multicast shared tree allows both the senders and receivers to reach the RP. However, almost all the actual multicast traffic is sent down a different tree called the source tree which is the tree with the source as its root.

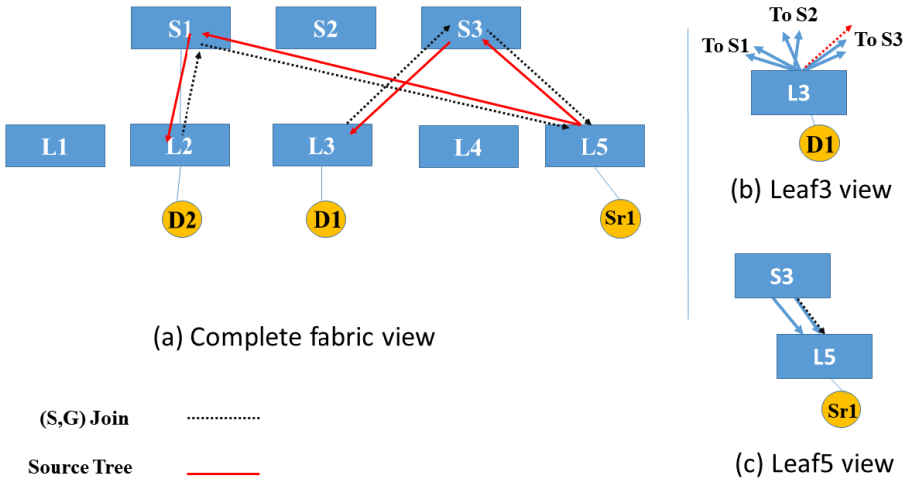


Figure 1.3 PIM SM source tree creation example.

In Figure 1.3, we show a traditional source tree creation based on $PIM-SM$. In this example, We skip the PIM shared tree creation process for brevity. In this example, the IP fabric has five leafs and three spines. Leafs are connected to each spine using two links. The source is connected to L_5 and the receiver is connected to L_3. L_3 joins the source tree by sending an IGMP join with specific source and group (S,G). Request will be sent to the next hop switch towards the source based on unicast routing information available on L_3. In this example, L_3 has six equal cost paths towards L_5 in its routing table: two paths for each of the three spines S_1, S_2 or S_3 as shown in Figure 1.3b. First link of L_3 towards S_3 is selected as next hop towards L_5 based on a static hash function defined by the switch. S_3 has two links to L_5 and selects Link2 towards L_5 based on the same hash (Figure 1.3c). Traffic will flow on the source tree ($L_5 > S_3 > L_3$). Let's assume a second receiver D2 joins the same source and group. L_2 joins the source tree by sending (S,G) IGMP join towards L_5. L_2 has six equal cost paths towards L_5: two links to each of S_1, S_2 or S_3. Based on static hash, S_1 Link2 is selected as next hop towards L_5. S_1 creates the last leg of the source tree towards L_5 using the same logic and Traffic will flow on the source tree ($L_5 > S_1 > L_2$). The example highlights issues with PIM multicast trees around multicast load balancing, link bandwidth protection,

optimized tree formation, multicast stream membership protection and admission control. PIM algorithm suffers from the following limitations:

- Multi-pathing and load balancing Modern DC architectures are predominantly based on folded Clos fabric [26] with a high cross section bandwidth that is built using multiple equal cost links between leafs and spines. PIM based multicast routing handles multi-path selection through a hash which is blind to bandwidth requirements. This PIM property leads to unpredictable performance and potential over-subscription of available links. In Chapter 4 Section 4.4, system testing confirms that PIM hashing based load balancing leads to over-subscription of a particular link while there is enough bandwidth in other links. This leads to unpredictable behavior where a system is oversubscribed in a link while enough bandwidth is available in alternative paths.
- Multicast tree formation: In IP fabric with multiple paths, PIM could lead to source trees with branching close to the source and thus consuming more bandwidth compared to source trees that branch closer to the receivers. Figure 1.3 shows an example where S_1 is originating a multicast flow with two destinations subscribing to the multicast group connecting to Leafs L_2 and L_3. In this example, PIM based routing caused L_5 to send the same multicast stream to both S_1 and S_3. This source tree branching allocation is not optimizing bandwidth being used on uplinks for L_5 as it is doubling the bandwidth used in L_5 uplinks and lowering the overall bandwidth available to other streams originating from L_5. A better solution would be for L_5 to send the multicast stream to a single spine (S_1 for example) and let S_1 replicate the multicast streams to both L_2 and L_3.
- Multicast admission control PIM based multicast do not provide an option for multicast flow admission control. PIM could lead to over-subscription of certain links in DC fabric and would impact existing and new flows that happen to be going through that oversubscribed link based on the hash selection.

On the other hand, one of PIM key advantages is its distributed nature where multicast path decision making is done at the local node level leading to faster setup time.

As with other data centers (DC), media DC production environment contains many network elements such as routers, switches and firewalls [18, 32]. Faults such as silent packet drops, routing loops, high delay in packet buffering and bugs occur frequently [48, 50, 123, 117]. Network packet losses can cause various temporal and

spatial damages to the video being transported. The visual effect of an encoded video packet loss can be severe and leads to a whole or partial video frame loss resulting in video impairments, video frame pixelation or frame loss on the receiver [94, 4, 85]. The authors in [25] show that a single packet loss leads to visual impairment in a video frame 94% of the time. In such live production environment, troubleshooting and monitoring of video flow quality is very critical. As we show in Section 7.2 and Chapter 2, commercial video monitoring tools rely on edge monitoring using devices that are expensive and lack scale. As an example, monitoring an average size video production facility with 3000 active flows would require 1000 servers which is very expensive and not scalable.

1.3 Dissertation Outline

In this section, the organization and contributions of the dissertation are outlined.

In Chapter 2, we review related work in the field of live media production, multicast optimization and video quality monitoring. In Chapter 3, we propose a formulation for multicast tree offline optimization in a multi-spine fabric.

In Chapter 4, We study the online multicast optimization. We include a design and implementation of a novel Intelligent Rendezvous Point algorithm iRP for bandwidth-aware multicast routing in media DC fabrics. iRP utilizes a controller-based architecture to optimize multicast tree formation and to increase bandwidth availability in the fabric. iRP algorithm maintains the creation, expansion and removal of source trees based on flow bandwidth and security requirements. The system offers up to 50% increase in fabric capacity to handle multicast flows passing through the fabric.

In Chapter 5, we study the problem of minimizing multicast tree setup time through distributed approach. $DiRP$ algorithm is presented utilizing distributed decision-making architecture to optimize multicast tree formation while maintaining low path setup time. The system is implemented using off-the-shelve commercially

available switches. $DiRP$ algorithm maintains the creation and removal of source trees based on bandwidth requirements. $DiRP$ algorithm is tested using Cisco's Nexus commercially available switches. Testing results confirm that the $DiRP$ algorithm is able to setup multicast tree paths based on available bandwidth while maintaining distributed decision making in the fabric to lower path setup time. The system offers substantially lower path setup time compared to centralized systems while maintaining bandwidth awareness when setting up the fabric.

In Chapter 6, we study the utilization of machine learning algorithms to increase multicast efficiency in the fabric. The work includes analysis of TV studio repetitive traffic patterns to demonstrate the benefits of time series forecasting including predicting multicast group membership and bringing online optimization efficiency closer to offline optimization results. We introduce $LiRP$ algorithm which increases $iRP's$ fabric utilization through machine learning. Machine learning is implemented using k-fold cross validation method to predict future multicast group memberships leading to optimized multicast tree placement. $LiRP$ algorithm is implemented using controller-based system and testing is done using Cisco Nexus data center switches. Testing results confirm that $LiRP$ system increases the efficiency of iRP by up to 40% through prediction of multicast group memberships with online arrival.

In Chapter 7, we study the problem of media relevant monitoring. In live media production environment (such as TV studios and sports venues), IP networks are utilized to carry live video using multicast for point to multi-point delivery. TV production traffic patterns are unique due to ultra high bandwidth requirements and high sensitivity to packet loss which causes video impairments. Existing network monitoring tools are either reactive by design or perform generic monitoring of flows with no insights into video domain. We introduce $MediaFlow$, a robust system for active network monitoring and reporting of video quality for thousands of flows simultaneously using a fraction of the cost of traditional monitoring solutions. To

the best of the authors' knowledge, $Mediaflow$ is the first of its kind to offer active in-network monitoring of video flow quality. We implement $MediaFlow$ using data center switches and our testing results confirm that $MediaFlow$ reduces video error detection and correction time from minutes to milliseconds as compared to current state-of-the-art methods. $MediaFlow$ is able to detect and report on integrity of video flows at a granularity of 100 mSec at line rate for thousands of flows. The system increases video monitoring scale by a thousand fold compared to edge monitoring solutions.

The dissertation is finally concluded in Chapter 8 by providing a summary of the dissertation as well as future directions for the research.

CHAPTER 2

RELATED WORK

A key motivation for this paper is the transition happening in the uncompressed Video transport from legacy non-IP format based on Serial Digital Interface (SDI) [98] to transport based on IP. Work in the standards bodies is currently underway around defining standards and relevant encapsulations. The work in [30] and [55] provide an overview of the live video industry current challenges, activities and forward-looking direction to migrate to IP based video flows.

The studies in [66], [22] and [47] rely on fabrics that require openflow support to implement the optimization. In [66], the authors propose a controller system based on openflow to address large scale multicast in switches with limited TCAM space using controller-based optimization. Authors in [47] offer an algorithm for building multicast tree that takes into consideration bandwidth requirements. The work in [16] takes a decentralized approach to multicast ECMP problem rather than solving it through a centralized controller approach. It proposes PIM extensions as well as PIM Bundle concepts to achieve that. The work in [22] offers a multicast routing algorithm based on flow stats provided by openflow.

Work in [27] describes the benefits of a media controller architecture and its building blocks. Most of the recent research around multicast and SDN assume that openflow [73] is the underlying protocol in data centers. Based on the authors industrial experience, most of data center architectures continue to be based on distributed routing protocols such as OSPF, ISIS and BGP rather than centralized routing such as openflow. Most of enterprise and service provider data centers continue to be built and operated based on traditional routing. As such, deployable solutions would need to assume plurality of routing protocols inside data center to work with.

The studies in [66], [22] and [47] rely on fabrics that require openflow support to implement the optimization. In [66], the authors propose a controller system based on openflow to address large scale multicast in switches with limited TCAM space using controller-based optimization. Authors in [47] offer an algorithm for building multicast tree that takes into consideration bandwidth requirements. The work in [16] takes a decentralized approach to multicast ECMP problem rather than solving it through a centralized controller approach. It proposes PIM extensions as well as PIM Bundle concepts to achieve that. The work in [22] offers a multicast routing algorithm based on flow stats provided by openflow.

The study in [41] provides good model for multicast optimization in DC networking, however it assumes batched arrival of flows in that the members for a group all arrive at the same time which is not a realistic expectation for multicast flows. The work in [64] addresses multicast reliability issues in DC but does not focus on increasing fabric efficiency. [63] addresses load balancing of multicast streams in Media DC but still requires openflow as protocol for implementation.

Online optimization of flows in a fabric has been studied in the literature. In [8], capacity constrained algorithm selects a flow path based on existing link utilization as well as potential profit using exponential function. Profit consideration adds complexity to the algorithm and is not applicable to our use case as the flows are all treated equally. Also, [8] does not take advantage of clos fabric structure leading to more complicated utilization tracking criteria. The more recent work in [44] and [21] tackles multicast optimization in capacity constrained SDN environments. [44] suggests an online algorithm for unicast and multicast optimization with incorporating TCAM utilization in the optimization scheme. However, the paper does not consider existing multicast tree structure in online optimization algorithm. [21] proposes OBSTA algorithm to optimize multicast throughput through taking into consideration temporal correlations between trees which is a novel approach.

However, the proposed algorithm depends on rerouting of existing multicast trees which would lead to interruption of traffic and is not acceptable to many industrial use cases such as live video production in media data centers.

Recent research has offered new approaches to active network troubleshooting and diagnosis and can be classified into two main categories: data sampling and selective reroute of data for off-network processing.

The work in EverFlow [123] utilizes selective rerouting concept allowing operators to select traffic samples to be monitored. While sampling approach minimizes bandwidth overhead, it fails to capture bursty network failures. Also, EverFlow is reactive in nature and requires a failure to happen to trigger an investigation which lags in time and leads to much higher MTTD compared to *MediaFlow*. Everflow builds on sampling approach that is implemented using Sflow [84]. FlowRadar [67] offers a netflow implementation using sketches inside switches to support flow queries in order to lower bandwidth overhead.

The work in 007 [7] and Confluo [49] is based on sending sampled network data to an edge receiver for further processing. Pingmesh [40] approach is based on sending probes to detect failures in the path. This approach of probing can lead to gaps in detection as probes are inherently sampling-based which leads to gaps in detecting errors. Also, since it uses out-of-band probes, it cannot detect failures that affect in-band data. At end-hosts, existing monitoring tools often focus on a specific type of information, require access to VMs or use too much of a resource. For example, HONE [104] and SNAP [121] Handles performance diagnoses by monitoring TCP-level statistics.

To summarize the challenges with the work above, the focus is towards generic network failure or focused on specific tcp related diagnostics that are not actionable for video flows.

As we focus on video production workflow, literature has broad coverage on the topic of Video transmission over reliable and unreliable networks. The study in [57] provides an FEC based solution to handle bursty errors in the network. with focus on FEC based solutions implemented at the end-point level. The work in [33, 42, 78] research video quality monitoring towards the end users by utilizing the fact that video delivery is wrapped with HTTP headers using TCP transport protocol. This approach is not applicable to our use case as the video is transported using UDP as a transport protocol. The solutions above focus on end point monitoring and thus lack scalability as well as ability to locate a network problem timely. In contrast, *MediaFlow* framework provides specific and actionable insight that is relevant to video production application by using in-network monitoring approach and focus on monitoring RTP statistics.

CHAPTER 3

MULTICAST OFFLINE PROBLEM FORMULATION

In this chapter, we formulate the multicast tree offline optimization in a multi-spine fabric. For System modeling, we consider a fabric carrying traffic for video production system where the underlying fabric is CLOS fabric with L leafs and S spines. Video sources such as a video camera (1 $<vs <$VS) are attached to Leafs (1 $<l <$L) and generate video content that is transported using multicast trees. The video receivers such as video editing stations (1 $<vr <$VR) are attached to one of the leafs and subscribe to multicast streams. As this system is based on CLOS fabric, it can be modeled as a directed graph with V vertices and E edges. Each vertex could be an end-point or leaf/spine. Each edge represents a network link between two vertices. The edge capacity represents available bandwidth between the vertices. For simplicity, we will assume that all multicast streams require same throughput of 1 unit. We assume that the demand matrix for sources and destinations for each of the trees is known in advance. We can use integer linear programming (ILP) to optimize the distribution of the trees to increase overall fabric utilization while maintaining the constraints of not exceeding available bandwidth on each of the links

Let X_{gs} be the state of the path between the source of the multicast group and one of the spines. The state is equal to 1 if the path is utilized for a group or equal to 0 if the link is not utilized for that group. Similarly, let Y_{rs} be the state of the path between a receiver of the multicast group and one of the spines with the state is equal to 1 if the path is utilized or equal to 0 if the link is not utilized for that group.

With the notations defined above, we mathematically formulate the offline optimization function as:

$$minimize \sum_{g} \sum_{s} X_{gs} \tag{3.1}$$

s.t.

$$\sum_{g} \sum_{s} X_{gs} \leqslant BW, \forall l \in [L], \forall s \in [S] \tag{3.2}$$

$$\sum_{r} \sum_{s} Y_{rs} \leqslant BW, \forall l \in [L], \forall s \in [S] \tag{3.3}$$

$$\sum_{s} X_{gs} \geqslant 1, \forall l \in [L], \forall g \in [G] \tag{3.4}$$

$$\sum_{s} Y_{rs} = 1, \forall l \in [1, L], \forall r \in [R] \tag{3.5}$$

$$Y_{rs} - X_{gs} \leqslant 0, \forall s \in [S], \forall r \in [R] \tag{3.6}$$

$$X_{gs} \geqslant 0, \forall s \in [S], \forall g \in [G] \tag{3.7}$$

$$Y_{rs} \geqslant 0, \forall s \in [S], \forall r \in [R] \tag{3.8}$$

The objective is to minimize the total number of uplinks from the source leafs X_{gs} while creating the multicast trees to achieve the desired connectivity between multicast sources and destinations. The constraints in equation (3.2) and equation (3.3) ensure that the total number of flows X_{gs} or Y_{rs} mapped through an edge do not exceed the total bandwidth of that edge BW. Constraint set equation (3.4) ensures that the source is mapped to one of the spines while equation (3.5) ensures that total links towards a receiver is 1. Constraint set equation (3.6) ensures that the receiver is mapped to a spine where the needed source is mapped to. i.e., ensure that ILP takes into consideration the source to spine mapping when selecting the destination to spine mapping.

Theorem 1. *The problem represented by equation (3.1) is NP-complete.*

Lemma 1. *The problem represented by equation (3.1) is NP.*

Proof. The number of constraints is polynomial in terms of the number of sources, number of receivers per source and number of spines. Given any solution for our

problem, we can check the solution's feasibility in polynomial time; then, the problem is NP. □

Lemma 2. *The problem represented by 3.1 is NP-Hard.*

Proof. To prove that the problem is NP-Hard, we reduce the bin packing problem, which is NP-hard [54] to a special case of our problem. In the bin packing problem, we have a set of items $G = \{1,2,...,N\}$, in which each item has volume Z_n , where $n \in G$. All items must be packed into a finite number of bins $(b_1,b_2,...,b_B)$, each of volume V_B in a way that minimizes the number of bins used. The reduction steps are as follows:

- The b-th bin in the bin packing problem is mapped to the j-th multicast tree in our problem, where the volume V_B for each bin is mapped to the maximum throughput capacity of a spine.
- The n-th item is mapped to the i-th multicast group, where the volume for each item n is mapped to the fan-out requirement for each of the multicast groups.
- All spines in the fabric have the same maximum forwarding capacity P.

If there exists a solution to the bin packing problem with cost C, then the selected bins will represent the spines that are selected for a group, and the items in each bin will represent the multicast groups that will be covered by the fabric and the total cost of our problem is C. This concludes our proof that equation (3.1) is NP-Complete. □

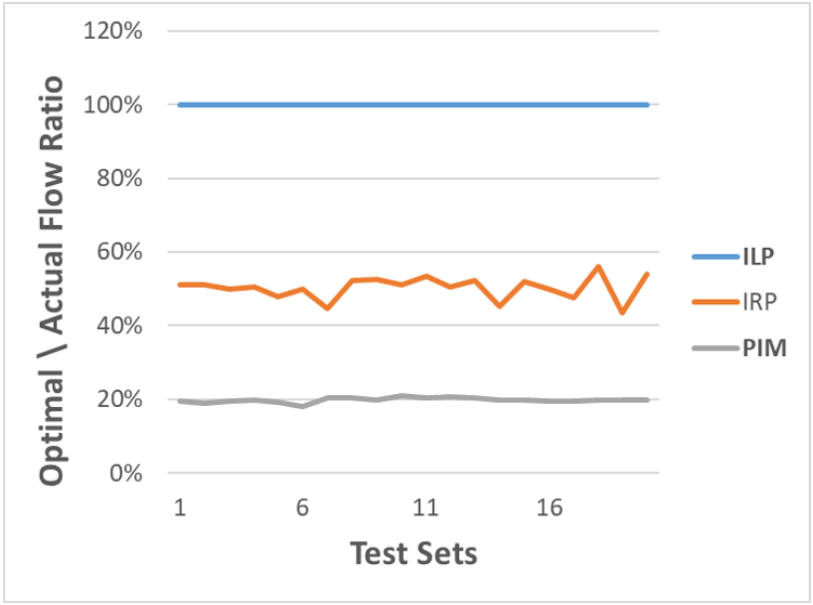


Figure 3.1 Optimal to actual flow count ratio for the same number of flows: Lower utilization reflects less efficient multicast trees.

Figure 3.1 shows Optimal to Actual flow count ratio for the same number of flows across the fabric where the performance of PIM is being compared to that

of the offline optimization algorithm. PIM shows much lower ratio leading to poor fabric utilization compared to optimized multicast tree placement. This observation can be explained by the random nature of PIM traffic hashing across multiple spines which leads to multicast tree going over multiple spines to carry the same multicast group leading to sub-optimal multicast touring and poorer fabric utilization.

Multicast offline optimization assumes that multicast demand matrix is known ahead of time which is not realistic for many of the workflows that rely on multicast to carry traffic. Much of the existing research, reviewed in Chapter 2, assumes full knowledge of demand matrix. As such, it utilizes offline optimization techniques for building multicast trees. In real world deployments of multicast, demand matrix is not known ahead of time and multicast requests arrival is done in an online fashion.

To address these limitations with offline optimization, the next two chapters will discuss two proposed algorithms that optimize multicast tree creation assuming online arrival of multicast requests with unknown demand matrix.

CHAPTER 4

IRP: INTELLIGENT RENDEZVOUS POINT FOR MULTICAST CONTROL PLANE

In this chapter, we focus on multicast online optimization problem specific to media data center requirements[1].

Intelligent Rendezvous Point (*iRP*) Algorithm is a greedy online algorithm for multicast tree optimization in a Clos fabric. The algorithm assumes online arrival of multicast flows and does not assume full knowledge of the demand matrix. The core concept of the algorithm is to select the least utilized link from the short list of available links that can service the request.

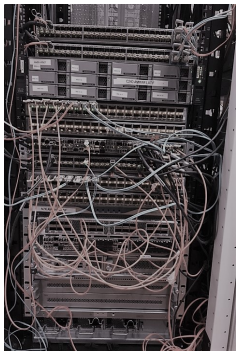
4.1 iRP System Architecture

iRP system architecture is shown in Figure 4.1. *iRP* SDN controller is the central point for multicast control plane and is responsible for setting up the required source trees between source and destinations for a multicast group. *iRP* does not mandate the implementation of special protocols such as openflow, and it is designed to work with existing IP Fabrics that are utilizing known routing protocols (OSPF, BGP, etc.). Leafs are responsible for forwarding IGMP signaling to *iRP* controller for further processing.

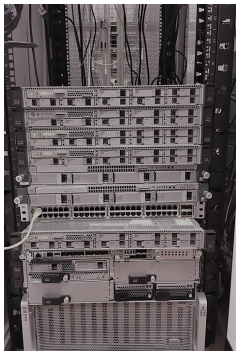
iRP Controller creates a central view of the available bandwidth in each link and connectivity between leafs. *iRP* controller creates source trees with the intention to maximize the overall served bandwidth in the fabric while preventing over-subscription of links. For multicast signaling inside the fabric, the system utilizes REST APIs to communicate signaling between the switches and *iRP* controller. *iRP* protocol is described below:

[1]The work has been published in [60]

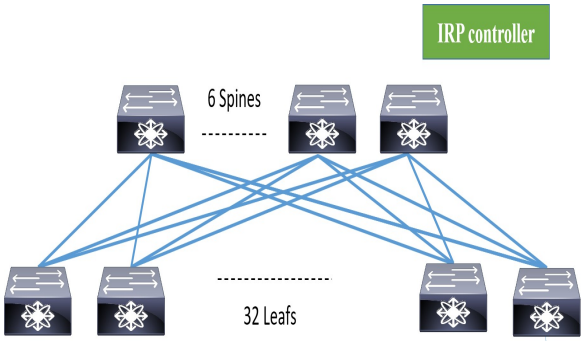
- *iRP* controller is made aware of the bandwidth requirements for a multicast flow. This can be done through API calls by the operator and using default value for a catch-all entry. This metadata information about flow bandwidth requirements will be used by *iRP* controller in the bandwidth calculation and setup of source multicast trees. This bandwidth metadata can also be used to police the flow to avoid mis-configuration or rogue end points that are not adhering to flow bandwidth requirements.
- Host IGMP request is intercepted by the first-hop leaf switch. The request is then forwarded to iRP controller
- In addition to IGMP signaling, source trees for multicast flows can be pre-provisioned in the fabric through API calls to iRT controller. This is useful for end points that do not support IGMP signaling natively but still require access to multicast flows.
- DC fabric is built as a fully meshed spine leaf fabric where a leaf would be connected to all spines in the fabric using one or more links. It is possible that even though fabric is set up as fully meshed, at the runtime some of the links or nodes are operationally down. *iRP* controller is resilient enough to handle link or node failures.
- iRP controller utilizes lazy bandwidth allocation approach for efficient use of fabric bandwidth. As iRP controller is aware of every sender and interested receivers.



(a) Data center fabric



(b) Sources,compute nodes and iRP controller



(c) *iRP* system architecture

Figure 4.1 *iRP* and *LiRP* system racks.

When *iRP* receives multicast requests, it runs *iRP* (line 1) against the multicast request and then instructs relevant spine and leafs to establish multicast tree accordingly using API calls to the relevant switches.

4.2 iRP Algorithm Details

Inspired by existing online routing algorithms such as in [8, 44, 44], iRP evolves the link cost based on its utilization and prefers links with lower utilization. In addition, iRP algorithm design considers the structured nature of clos fabric. Compared to mentioned algorithms, iRP simplifies source tree formation by focusing on minimizing the branching of the multicast tree. iRP also simplifies the decision process of assigning an uplink by always selecting the uplink with most bandwidth within a set of links.

Pseudocode for iRP Algorithms is shown in Algorithm 1. In line 2, the algorithm checks if the multicast group is already provisioned in the fabric. In line 3, we check if there is enough bandwidth from existing assigned spine to the receiver leaf and if true, we proceed with the same selected spine and continue with setting up the path in line 19. If the above is not true, algorithm tries to pick a viable spine from list of available spines by confirming bandwidth availability on the spines both to the receiver leaf and from sender leaf in line 15 and line 16. The list of compliant spines is parsed and the spine with the highest bandwidth is selected in line 20. similar approach is taken for selecting the link with most available bandwidth in line 21 and then the tree is setup across the fabric in line 25.

iRP algorithm executes in polynomial time $\mathcal{O}(S)$ where S is the number of spines in the fabric. As a comparison, time complexity for recent SDN algorithms are proportional to the complete fabric size (all leafs and spines). In [44], time complexity is $\mathcal{O}(K^{\epsilon}log(n))$ where n is the network size, K is multicast request members, and $0 < \epsilon < 1$. Complexity function in [21] is $\mathcal{O}(|V|^2|D|^2)$ where V is number of vertexes in the fabric and D is the number of multicast destinations. As $S << n$ in a Clos fabric, iRP complexity is much lower and has faster execution time compared to [44, 21] without utilizing openflow.

Algorithm 1 iRP Algorithm Pseudocode

Inputs

Input $Spines[]$, $Leafs[]$ ▷ list of Spines and Leafs
Input $SourceLeaf$, $RecieverLeaf$ ▷ Leafs where Source and Receivers are connected
Input $Bandwidth$ ▷ Required Bandwidth
Input $GroupSpine$ ▷ If the multicast group assigned to a Spine, list Spine here

Steps

```
procedure iRP
    if GroupSpine ≠ Empty then
        if Spine to RecieverLeaf Bandwidth > Bandwidth then
            SelectedSpines[] ← Spine
        else
            SelectedSpines[] ← SpinesList
    else
        SelectedSpines[] ← SpinesList
    if SelectedSpines[] = 0 then
        return Unavailable BW Error Code
    else
        SetupTree                                   ▷ Setup needed tree
function SpineList                                  ▷ shortlist spines with BW
    for each S in Spines[] do
        if S to RecieverLeaf Bandwidth > Bandwidth then
            if SourceLeaf to S Bandwidth > B then
                SelectedSpines[] ← Spine
    return SelectedSpines[]
function SetupTree                                  ▷ establish source tree
    SelectedSpine ← SpinewithmostBWtoRecieverLeaf
    ReceiverLink ← linkwithmostBWtoRecieverLeaf
    SourceLink ← linkwithmostBWfromSourceLeaf
    RecieverLink ← RecieverLink − Bandwidth
    SourceLink ← SourceLink − Bandwidth
    Setup Source Tree
    return
```

4.3 System Implementation

iRP system testing was done initially using an IP fabric with two spines and four leafs. The setup was then expanded to 32 Leafs and 6 Spines to reflect testing conditions with a scalable fabric. The switches used to build the fabric are commercially available data center switches that run traditional unicast routing protocol to build routing tables and reachability in the fabric. *iRP* system does not require switches to support openflow or other specialized SDN protocols. The leafs are Cisco's Nexus 93180 switches running NxOS 9.2(1) firmware and the spines are Cisco Nexus 9236 switches running same firmware [106]. Each leaf is connected to all spines using 100Gbps fiber links. For multicast sources and destinations, the setup contains 30 nodes to each leaf for a total of 960 nodes. Commercial grade video nodes that generate uncompressed video streams are used in this setup. During the testing phase, we introduce multicast senders and receivers gradually through automation using python code and capturing testing results accordingly. *iRP* controller software is running inside a VM using ESXi hypervisor running on Cisco UCS C240-M4 server x86 based server. A picture of the actual racks that contain IP fabric as well as server equipment is in Figure 4.1

4.4 Experimental Results

In this section, we compare the performance results of traditional *PIM* SM to *iRP* controller system in a real folded clos fabric using commercial off the shelf (COTS) Cisco switches. All tests are done using two test sets, one test set using fabric running *PIM* control plane. The same test suite is then run against the fabric running *iRP* controller system.

In Figure 4.2, we compare standard deviation (SD) of multicast flow distribution across available equal cost multi paths (ECMP) on leafs and spines. We selected SD as a parameter to show the evenness of distribution of the flows between available ECMP paths where larger SD indicates uneven load balancing between ECMP links. Test sets for *PIM* on both Spine1 and Spine2 shows high SD values while the same

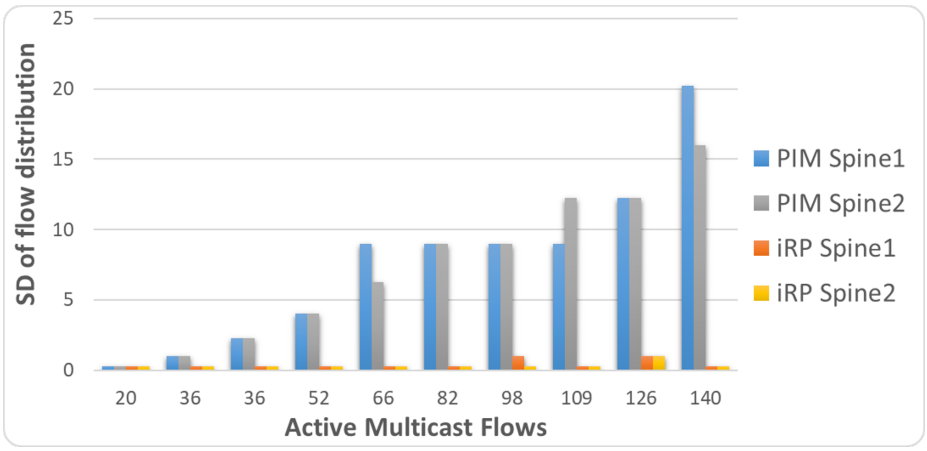


Figure 4.2 Standard deviation to show evenness of flow distribution across ECMP links between spines and leafs in Clos fabric.

tests with iRP system show consistently negligible and close to 0 SD values which confirms iRP ability to evenly distribute flows across ECMP links.

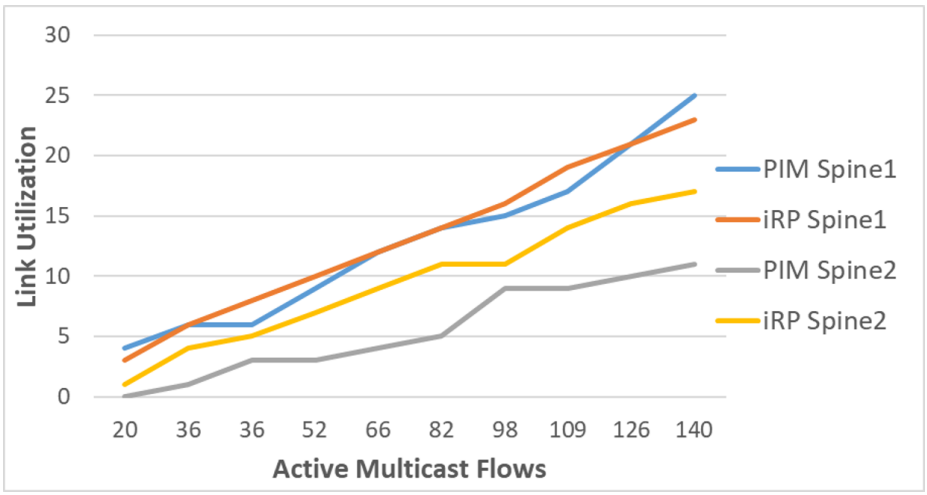


Figure 4.3 Comparing highest link utilization amongst ECMP links between PIM and iRP for the same number of active flows.

Figure 4.3 compares highest link utilization for PIM and iRP on Spine S1 for a similar set of multicast flows. iRP system shows around 50% lower utilization than PIM based system for the same number of multicast streams. The lower link utilization with the same set of multicast flows leads to higher fabric multicast capacity.

Figure 4.4 shows the total number of tree segments as the number of multicast flows increases for both PIM and iRP based systems. For the same number of flows

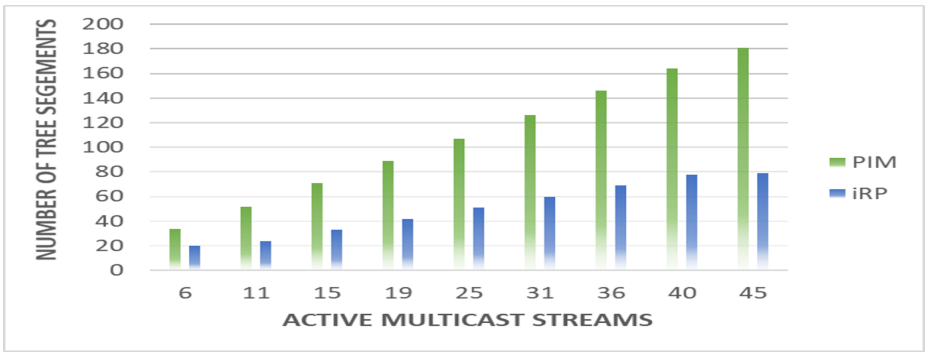


Figure 4.4 Total source tree segments in the fabric: higher number of segments for the same number of flows indicate branching closer to the source leading to more fabric bandwidth utilization.

and distribution of receivers, the lower number of tree segments indicates a more efficient source tree structure in that the multicast branching is happening closer to the receivers. In Figure 4.4, we clearly see the difference in the total number of source tree segments between PIM and iRP. Total segments in iRP is lower by about 50% compared to PIM for the same fabric and the same number of multicast groups running in the fabric which increases overall multicast capacity in the fabric. In Chapter 2, we discussed how PIM lacks awareness of existing multicast flows as well as bandwidth requirements for the flow when building or modifying source tree structure and how that leads to inefficient tree structure.

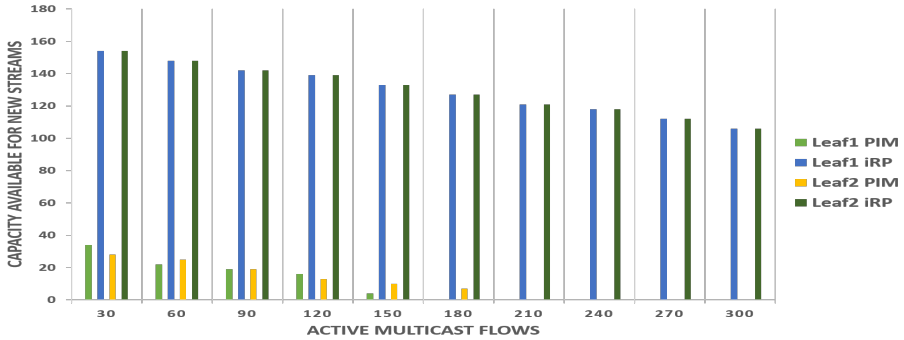


Figure 4.5 Comparing PIM and iRP systems for availability of a link with needed bandwidth for future streams.

The test results in Figure 4.5 compare PIM and iRP systems for the number of multicast streams that was forwarded by Leaf1 before starting to oversubscribe one or more of the uplinks to the spines. Due to the random nature of hash-based flow

distribution, PIM shows lower number of flows before the system gets into blocking state. This is because while there is enough capacity in the system overall, hash based load balancing in PIM would lead to some links getting oversubscribed while others having available capacity. iRP based system offers higher flow capacity before the system reaches the blocking state. This is because of the deterministic nature of flow distribution over ECMP links and bandwidth aware admission control offered by iRP system. Figure 4.5 shows an increase of 70% or more in guaranteed available system capacity compared to PIM.

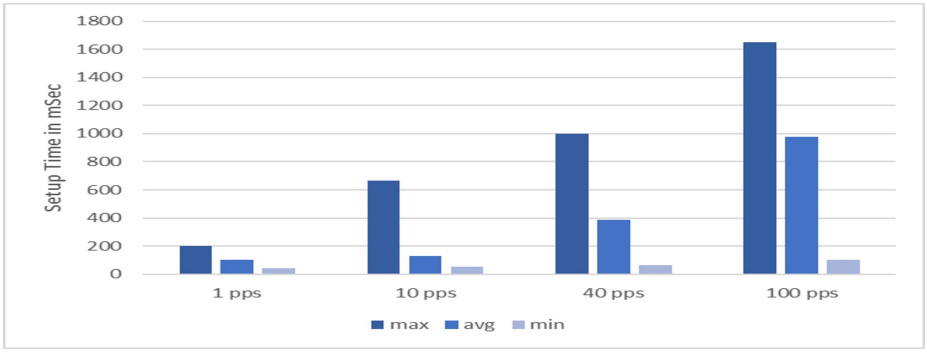


Figure 4.6 iRP system multicast flow setup time.

As an SDN controller system , iRP's system scalability is an essential part of analyzing its performance. Figure 4.6 shows testing results for multicast path setup time utilizing single iRP controller. Results in Figure 4.6 are based on testing with a single iRP instance with about 100 requests per second which is sufficient for Media DC use case. Testing system hardware details are described in Section 4.3. Scaling up SDN controller system capacity to handle increased flow requests is a topic that has been studied extensively ([56, 43, 120]). We note that path calculation portion of iRP is stateless thus scaling of the system can be achieved using distributed system approach by running multiple instances with a shared database backend such as $Onix$ [53].

Our test results confirm the observations that were discussed in Chapter 2. Even distribution of flows over ECMP routes as well as intelligent placement of source

trees, both offered by iRP system, translate to a significant increase in the number of multicast flows that can be allowed through the fabric without over-subscription for the same system bandwidth.

CHAPTER 5

DIRP: DISTRIBUTED INTELLIGENT RENDEZVOUS POINT FOR MULTICAST CONTROL PLANE

In this chapter, we focus on multicast online optimization problem with flow setup delay as an additional constraint [1].

Path setup time is important for real time applications such as Video conferencing or IPTV applications where the receiver needs to receive needed content quickly after it sends a request for such content. Many of the proposed algorithms to address PIM limitations such as iRP in Chapter 4 as well as [66, 22, 22] and [47] utilize a centralized decision making with an SDN controller for bandwidth calculation and setting up the path. One of the main limitations to centralized systems is the higher delay in path setup time compared to distributed systems such as PIM as well as scaling of the system to handle thousands of flows.

In this chapter, We provide the following contributions:

1. Propose $DiRP$ Algorithm for creation of efficient multicast trees for online group arrivals to ensure optimal tree creation as well as ensure multicast flow bandwidth requirements are met in a deterministic way.
2. Implement the proposed algorithms using COTS switches and utilizing traditional routing protocols based on Cisco Nexus DC switch family.
3. Present test results using HD Video flows comparing $DiRP$ with PIM and iRP algorithms results. Test results confirm that DiRP algorithm lowers by 10x the observed path setup time as compared to iRP. This is done while maintaining accurate admission of multicast flows based on bandwidth requirements.

5.1 DiRP Algorithm

In this section, we provide an overview of proposed $DiRP$ algorithm. $DiRP$ algorithm is based on a simple idea of using a deterministic hash function to define which spine

[1] The work has been published in [62]

to choose for multicast routing. The hash function takes as an input source host IP address and multicast group. The output of the hash function is a list of spines to send traffic to or receive traffic from in ascending order. This hash function is then implemented on all Leafs in the fabric as part of the spine selection process. Algorithm 3 provide the pseudocode for DiRP algorithm.

Algorithm 2 DiRP Algorithm Pseudocode

Inputs

Input $Spines[]$ ▷ list of Spines
Input $SourceIP$, $MulticastGroup$ ▷ Source IP and Multicast Group Address
Input $Bandwidth$ ▷ Required Bandwidth

Steps

function SpineHash(S,G) ▷ hash function for spines
    $HashedSpines[] \leftarrow Hash(S, G)$
    return SelectedSpines[]
procedure *DiRP* On Source Leaf
    if GroupSpine ≠ Empty **then**
        $SelectedSpines[] \leftarrow Spine$
    else
        $SelectedSpines[] \leftarrow$ SpineHash(S, G)
    while Spine is empty **do**
        select spine in SelectedSpines if there is BW to spine
    if SelectedSpines[] = 0 **then**
        return Unavailable BW Error Code
    elseSend stream to *spine*
procedure *DiRP* On Receiver Leaf
    if GroupSpine ≠ Empty **then**
        $SelectedSpines[] \leftarrow Spine$
    else
        $SelectedSpines[] \leftarrow$ SpineHash(S, G)
    while Spine is empty **do**
        select spine in SelectedSpines if there is BW to spine
    if SelectedSpines[] = 0 **then**
        return Unavailable BW Error Code
    elseSend PIM Join to *spine*
procedure *DiRP* On Spine(PIM Join message)
    if Spine BW to sender ≠ 0 **then**
        Spine establishes link to Source Leaf
    else
        Spine send PIM ECMP redirect message to Leaf

Figure 5.1 shows an example of *DiRP* in a fabric with three spines. When L_5 receives traffic from source S_1, it would calculate a hash function using the associated source IP address and multicast group to arrive at a preferred *spine*. L_5 will use local bandwidth calculation to choose the best interface to send traffic to the *spine* that has enough bandwidth. In the example, we show that the hash function is hashing towards S_3. When S_3 starts receiving traffic from L_5, It will reduce utilized bandwidth from available bandwidth to keep account of link utilization on its own links.

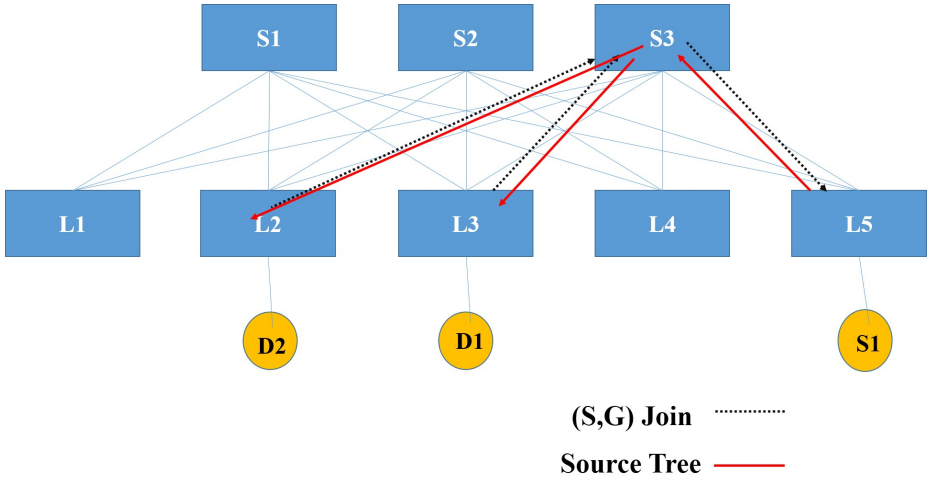


Figure 5.1 *DiRP* example.

When L_3 gets an IGMP join from receiver D_1, L_3 will use the same hash algorithm to determine which *spine* to request traffic from, and update utilized bandwidth for link utilization. Both S_1 and D_1 will be calculating the same hash result independently, and hence reach the same *spine* for establishing a flow.

If the available bandwidth between a spine and a leaf cannot accommodate the bandwidth requirements of an incoming flow, the sender leaf shall calculate the hash for next best *spine* from the list produced by the hash algorithm and send traffic over to the selected spine .

At this point, when a receiver arrives, it shall still hash to original *spine* and request traffic from it. Since the *spine* will not have available bandwidth to reach the requested source, it will send a $PIM\ ECMP$ redirect to the receiver leaf. Receiving

a *PIM ECMP* redirect message, the receiver leaf shall trigger a recalculation of RPF towards RP address, where it'd look for the next best *spine* and since the hash algorithms shall be the same on both sender Leaf and receiver Leaf, they would again end up on the same *spine*.

A port on a switch sees the incoming traffic and chose a link to send the traffic to the Spine. Receivers that are interested, join the source using regular multicast protocols like IGMP. When a receiver shows interest in a traffic, it also looks for an uplink with available bandwidth and requests the traffic from the primary spine 2.

In summary, *DiRP* algorithm utilizes a deterministic hash algorithm to ensure that all nodes through the network attempt to send or receive multicast traffic through same *spine* leading to synchronized tree setup mechanism that is distributed to allow for faster setup time. This algorithm allows the system to benefit from distributed architecture to scale the capacity to process join and leave message using the capacity of the whole fabric when compared to a centralized architecture where the control could become the processing bottleneck.

5.2 Experimental Results

In this section, we compare the performance results of *DiRP* with *PIM* and *IRP* systems in a real folded Clos fabric using Commercial Off the Shelve (COTS) Cisco switches.

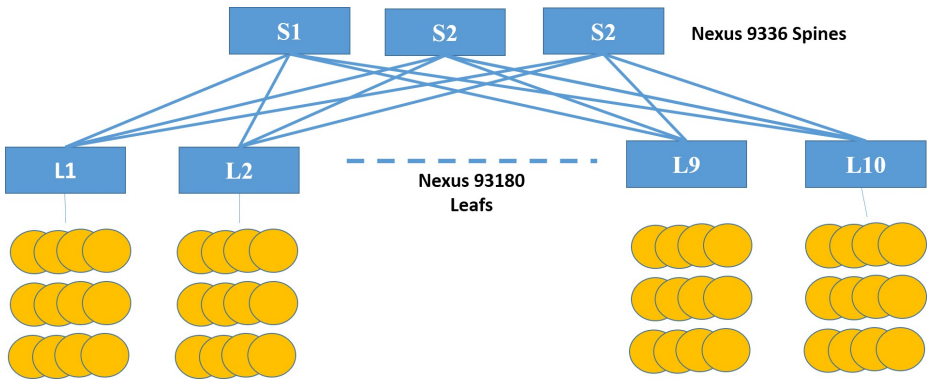


Figure 5.2 *DiRP* System.

System testing was done using an IP fabric with 10 Leafs and 3 Spines. The switches used to build the fabric are commercially available Cisco Nexus data center switches. The leafs are utilizing Cisco Nexus 93180 switches running NxOS 9.2(1) firmware and the spines are using Cisco Nexus 9336 Switches running same firmware[106]. Each of the leafs are connected to each of the spines using 6x100Gbps fiber links as shown in Figure 5.2. The setup utilized a combination of Commercial grade video nodes that generate Uncompressed HD Video streams as well as traffic generators. During the testing phase, we introduce multicast senders and receivers gradually through automation using python code and capturing testing results accordingly. A picture of the actual racks that contain IP Fabric as well as server equipment is in Figure 5.3.

In Figure 5.4, we compare path setup time between PIM, iRP and $DiRP$ for request arrival rate of 40 requests per second evenly distributed. The figure shows that $DiRP$ has much lower path setup time compared to centralized systems like iRP. This is expected as centralized systems require multicast requests to be sent to the central controller for further processing. This path usually adds tens of milliseconds to the processing time. processing the request locally on the leafs cut down the overall processing time without compromising the bandwidth awareness when setting up the path.

Same observations can be seen in Figure 5.5 and in Figure 5.6 where the processing time for $DiRP$ is much lower compared to iRP.

In Figure 5.7, we compare standard deviation (SD) of multicast flow distribution across available equal cost multi paths (ECMP) on leafs and spines. We selected SD as a parameter to show the evenness of distribution of the flows between available ECMP paths where larger SD indicates uneven load balancing between ECMP links. Test sets for PIM on both Spine1 and Spine2 show high SD values while the same tests with $DiRP$ and iRP system show consistently negligible and close to 0 SD

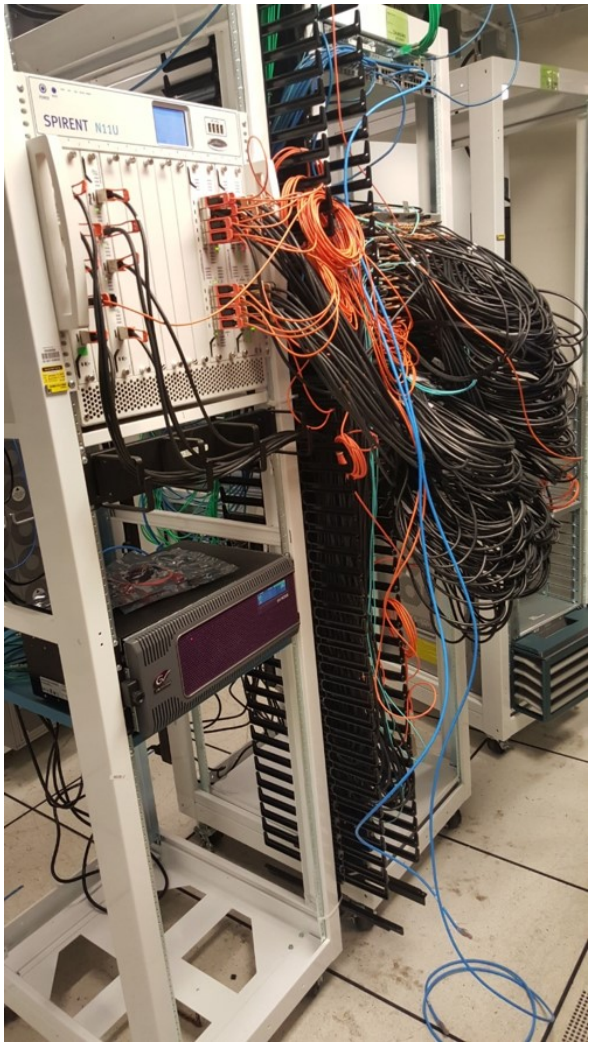


Figure 5.3 DiRP testing rack.

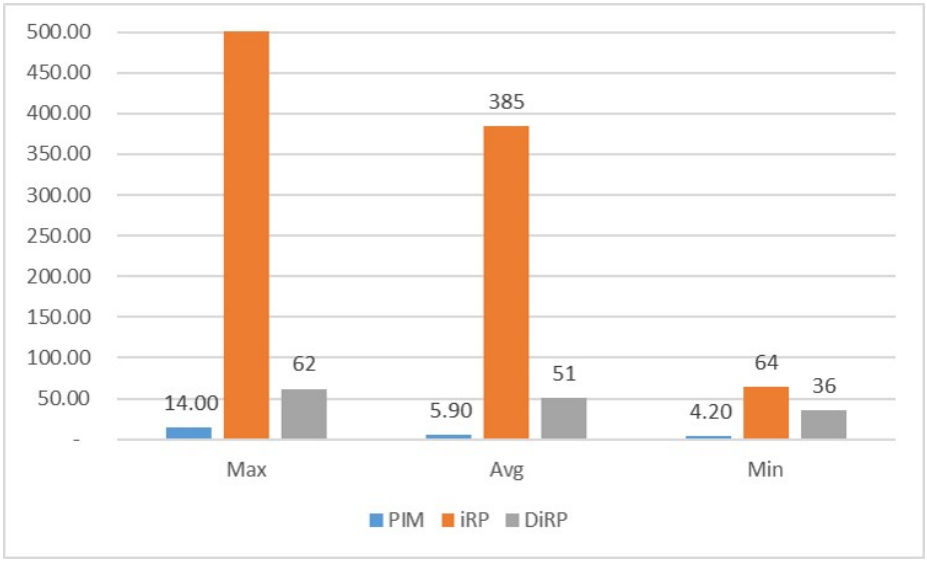


Figure 5.4 Maximum flow setup time for 40 pps IGMP arrival rate.

values. These observations confirm that *DiRP* can evenly distribute flows across ECMP links leading to higher fabric capacity for multicast streams.

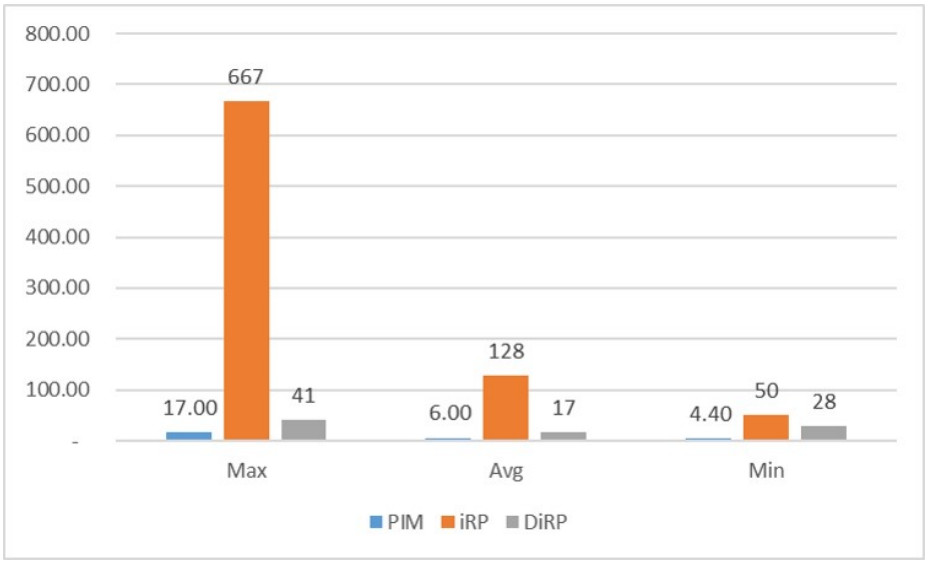


Figure 5.5 Maximum flow setup time for 10 pps IGMP arrival rate.

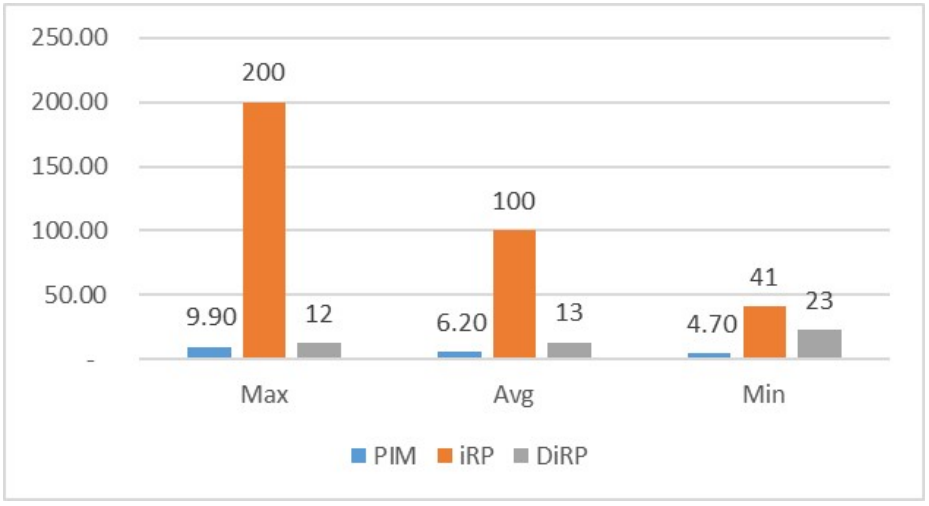


Figure 5.6 Maximum flow setup time for 1 pps IGMP arrival rate.

Our test results confirm the observations that were discussed in Chapter 2. Even distribution of flows over ECMP routes as well as intelligent placement of source trees, both offered by iRP system, translate to significant increase of the number of multicast flows that can be allowed through the fabric without over-subscription for the same system bandwidth.

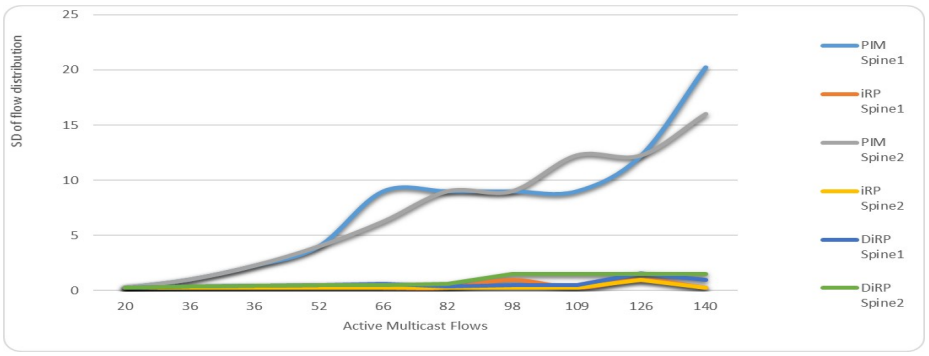


Figure 5.7 Standard deviation to show evenness of flow distribution across ECMP links between spines and leafs in Clos fabric.

5.3 Conclusion

Multicast is widely used to deliver point to multi-point traffic such as real-time audio and video traffic. Multicast protocols such as PIM suffer from performance limitations as it is not bandwidth-aware. In this chapter, we propose $DiRP$ algorithm to enhance multicast capacity in a fabric. Multicast routing in $DiRP$ is handled in a distributed model across the fabric while maintaining bandwidth awareness when making routing decisions. $DiRP$ Algorithm ensures efficient tree formation and load-balancing of multicast flows across ECMP links and implements multicast admission control based on bandwidth availability in the network. We implement $DiRP$ system using fabric built on Cisco Nexus COTS switches. The testing results confirm increased fabric utilization compared to PIM while maintaining similar path setup time as compared to PIM. Test results also confirm that $DiRP$ provides performance that is 60% higher than PIM while maintaining similar path setup delay times.

CHAPTER 6

LIRP: LEARNING-BASED ENHANCED IRP

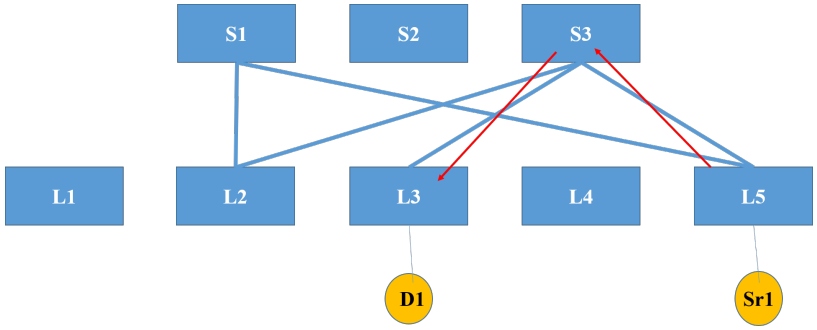
In this chapter, we research the benefits of utilizing machine learning to further optimize multicast tree placement in an online environment [1].

In Chapter 4, we show that iRP algorithm achieves better fabric utilization compared to PIM. However, Figure 3.1 shows that iRP is less efficient than offline optimization. This is expected because iRP does not have foresight or complete knowledge of all the members of a multicast group in advance. That could lead to situations where the spine selected for a group does not have enough bandwidth needed for future members of a multicast group.

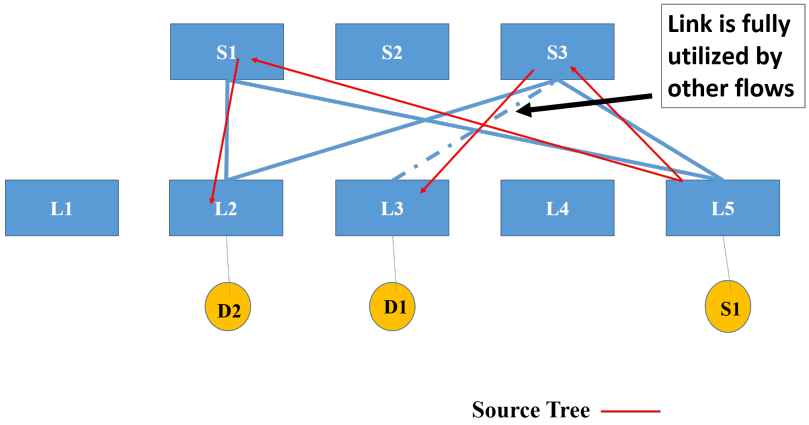
Figure 6.1 provides an example of the impact of local optimization provided by online algorithms such as iRP. In Figure 6.1a, the first receiver D_1 for multicast group Sr_1 arrives into leaf L_3. iRP online algorithm selects spine $S3$ and a multicast tree is established $L5 > S3 > L3$. In Figure 6.1b, second receiver D_2 arrives to leaf L_2. Due to other existing multicast groups, spine $S-3$ does not have enough capacity in its link to L_2. This forces the iRP algorithm to select a second spine to connect the source to D_2 which in this case is $S1$ and the multicast tree now has two paths $L_5 > S3 > L_3$ and $L_5 > S1 > L_2$. This leads to L_5 having to service the same multicast tree through two different uplinks, one towards $S3$ and another towards $S1$. This leads to lower overall fabric capacity and utilization for the same number of multicast groups.

In summary, one of the main challenges for online optimization of multicast traffic is the lack of knowledge of full demand matrix. It is also expected that receivers change group subscriptions in a specific time frame.

[1] The work has been published in [61]



(a) Arrival of initial receiver for group Sr1, multicast tree is established though spine S3 based on available capacity



(b) Arrival of second receiver for group Sr1, L3-S3 Link is already at capacity, tree will need to go through another spine S1 that has capacity towards L2

Figure 6.1 Multicast tree placement problem with online arrival pattern.

In the next few sections, we describe Learning based Intelligent Rendezvous Point *LiRP* Algorithm to incorporate prediction in the optimization. Like *iRP*, *LiRP* algorithm assumes online arrival of multicast flows and utilizes past observations of multicast flow patterns to predict future multicast membership. *LiRP* utilizes time series forecasting to be able to predict the multicast group membership and utilize the information during the path selection phase in order to optimize multicast tree placement.

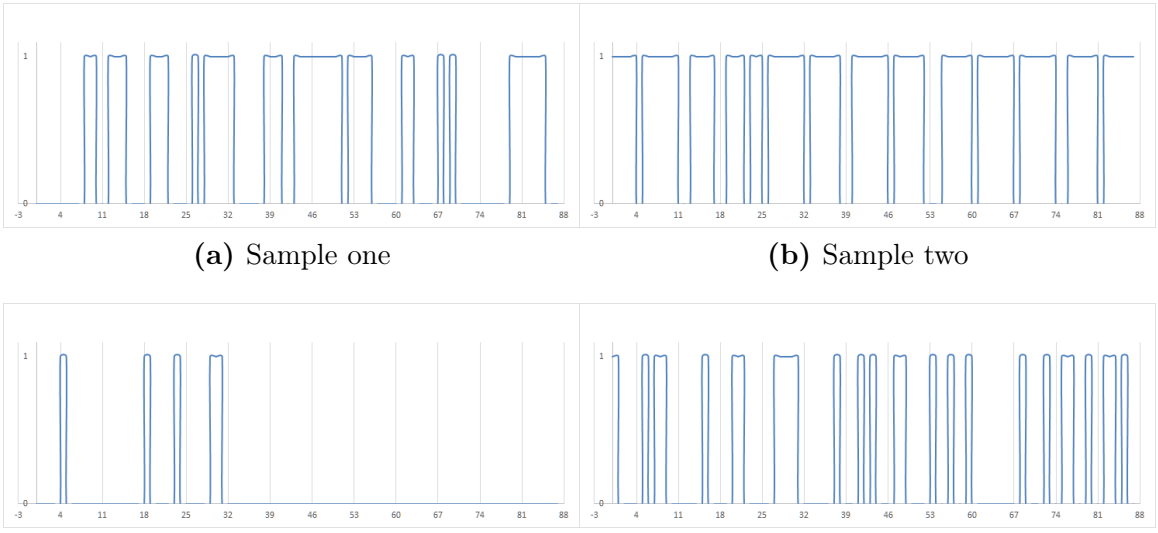
6.1 The Case for *LiRP*

SDN Controller architecture allows for utilization of traffic analytics in forwarding decisions and made it easier to apply machine learning algorithms [118, 115]. The core idea behind *LiRP* algorithm is to utilize machine learning to predict future group memberships based on past occurrences of such membership, This is done through exploiting repetitive traffic patterns in multicast group membership. Once group membership is predicted, that information can be utilized to optimize multicast tree placement while maintaining online arrival of multicast receivers. We start by analyzing control logs captured from a TV production studio to identify traffic patterns that can be analyzed by machine learning to optimize multicast tree formation. The logs being analyzed represent the *takes* that a TV Studio operator would execute as part of his/her workflow.

A *take* represents a control command to route a source (e.g., a TV camera) to a destination (e.g., a production switcher). Figure 6.2 shows four samples of such operator action or takes over 87 day period in an actual TV studio.

Each of the four samples represent a time series of the daily membership of a receiver to a multicast group using true/false binary representation for an 87 day duration. Elements in the time series represent a TV station operator action either connecting (represented by logical 1 or true) or disconnecting (represented by logical 0 or false) a receiver to a group. The *take* would translate to a multicast join hitting the

switch from the receiver and triggering the fabric to create a multicast tree towards the source.



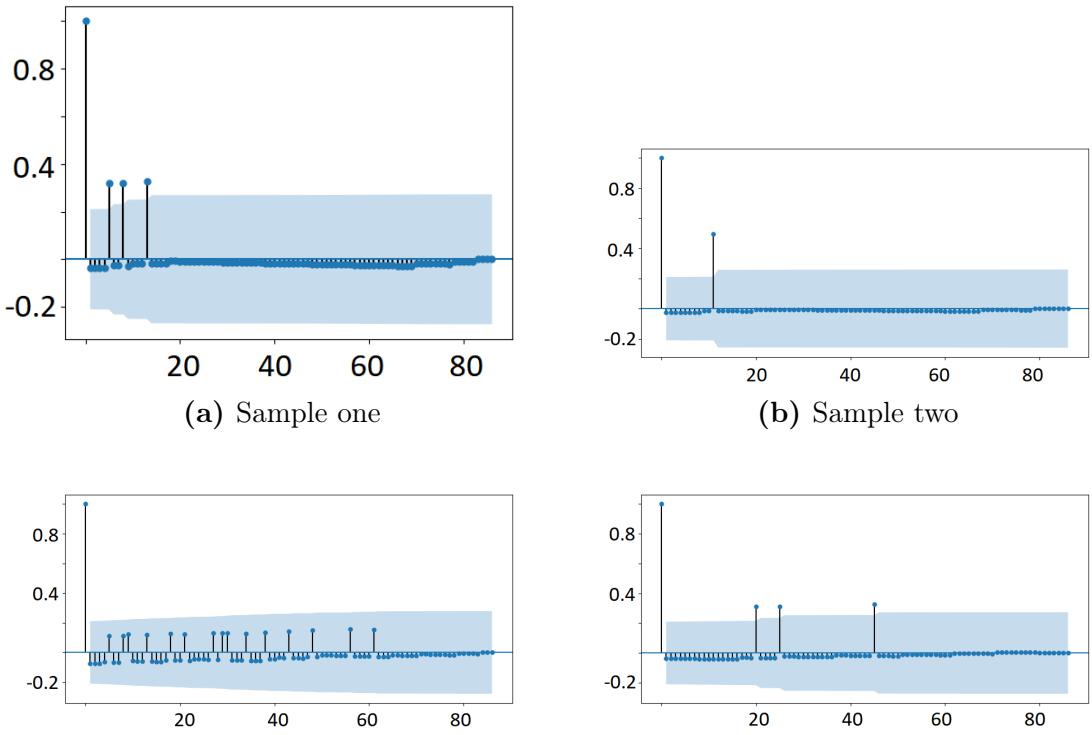
(a) Sample one **(b)** Sample two

(c) Sample three **(d)** Sample four

Figure 6.2 Time series of operator activities in a TV studio over 87 days where "1" represents a take or a receiver joining a group and "0" represents a leave from that multicast group.

These *takes* are delivered in sequence, either manually by an operator or through an automation system that generates the commands based on a schedule. This online behavior means that the multicast group membership is not known or provided ahead of time but rather a single receiver is connected to the source of multicast group in each of these control sequences. To simplify the analysis of the logs, we will assume a 24h time cycle in a studio where the workflow is a cycle of 24 hours which is a reasonable assumption given the daily nature of TV production and program schedule.

To analyze the arrival patterns of these operator commands, we utilize autocorrelation function analysis. Autocorrelation function calculates the correlations between the time series and its shifted copies at different points in time. The autocorrelations are usually calculated for the specific range of lags (shifts) and are expressed in the form of correlogram graph or autocorrelation plot [116].



(a) Sample one

(b) Sample two

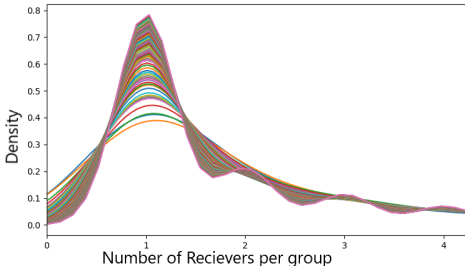
(c) Sample three

(d) Sample four

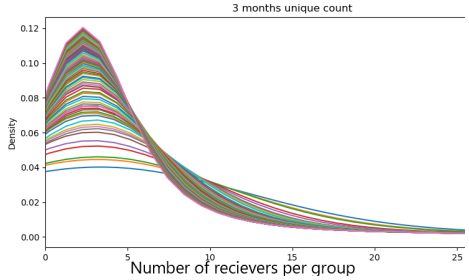
Figure 6.3 Autocorrelation of 4 samples of operator takes over 79 days period.

Figure 6.3 shows correlogram graphs with autocorrelation analysis of four samples of the operator actions represented by time series. Each of these four samples shows strong but unique autocorrelations around specific intervals unique to each receiver. Sample in Figure 6.3a shows a strong autocorrelation around days 5, 7 and 13 which is a reasonable pattern that matches 5 business week days as well as 7 week days. This is not true for all-time series though. Sample in Figure 6.3b shows strong correlation around days 20 and 25. From these samples, we can see that the multicast group membership time series shows clear autocorrelation attributes or strong seasonality. This is expected as the workflow inside the TV studio plant exhibit seasonality itself. Workflows have daily or weekly patterns. Sources and destinations would have clear association patterns that are repeated over time. Another observation is that the seasonality or autocorrelation parameters vary between different time series. i.e., the autocorrelation parameters would vary between samples. This leads us to have to define these parameters per a pair of sender/receiver than generalizing to a group or to the whole fabric.

We use kernel density estimation (KDE) in our statistical analysis of TV studio traffic. KDE is a non-parametric density estimator requiring no assumption that the underlying density function is from a parametric family. KDE will learn the shape of the density from the data automatically [20]. Figure 6.4a shows KDE of multicast tree size (number of unique receivers in a group) daily with each line representing a day. The graph shows that majority of the multicast tree sizes has less than four unique receivers. Actual distribution shows that trees with up to two flows constitute to 60% or more of flows. These observations confirm that while the multicast fabric allows for all-to-all type communication, the ratio of receivers to a sender in such fabric is much less that the total number of possible receivers. This allows us to build a fabric that is less likely to be blocking.



(a) Number of unique receivers per tree per day



(b) Total number receivers per tree per day

Figure 6.4 KDE graph representing number of unique receivers per tree per day.

Figure 6.4b shows the same KDE but using number of all receivers that belong to a specific multicast tree throughout the day. If a receiver leaves a tree and joins it back, it will be counted again. This graph shows the maximum number of all receivers as the day progresses. We can observe that for majority of the trees, the maximum number of all receivers including changes is less than 10.

6.2 Predicting Multicast Group Membership Using Neural Networks

As presented in Section 6.1, Multicast groups memberships in media data centers can be represented by time series that exhibits repeated traffic patterns. These traffic patterns have strong autocorrelation characteristics that vary between various time series. These characteristics allow us to use time series forecasting techniques to achieve one step forecasting of time series to predict group membership for a specific day.

Neural Networks (NN) has been used in the literature for time series forecasting [11, 23, 52, 12, 83, 122, 2]. *LiRP* utilizes neural networks supervised learning based on K-fold cross validation to implement one step time series forecasting of multicast group time series. Supervised local learning has been studied in the literature [11, 45, 37]. K-fold cross validation has been studied in the literature for its effectiveness in estimation [3, 31].

Advantages of using NN local learning for time series forecasting are:

Fewer assumptions: Neural Network local learning does not assume a priori knowledge on the process underlying the data. For example, it makes no assumption on the existence of a global function describing the data and no assumption on the properties of the noise. The only available information is represented by a finite set of input/output observations. This feature is relevant in real datasets where problems of missing features, non-stationary and measurement errors make appealing a data-driven and assumption-free approach [12].

Online learning capability: The local learning method can easily deal with online learning tasks where the number of training samples increase with time. In this case, local learning simply adds new points to the dataset and does not need time-consuming re-training when new data become available [12].

We will start NN Analysis of multicast time series by modeling it as a univariate time series:

$$s_t = g(t) + \phi_t \tag{6.1}$$

The model consists of a systematic part $g(t)$, also called signal or trend, which is a deterministic function of time and a stochastic sequence and a residual term ϕ_t or noise. More specifically to a univariate time series, the model can be constructed as :

$$s_t = f(s_{t-l}, s_{t-l-1}, \ldots, s_{t-l-n+1}) + \phi_t \tag{6.2}$$

where $f(s_{t-l}, s_{t-l-1}, \ldots, s_{t-l-n+1})$ is the embedding vector which is a finite window of the time series l called the lag time and n (order) is the number of past windows that are being taken into consideration for future prediction.

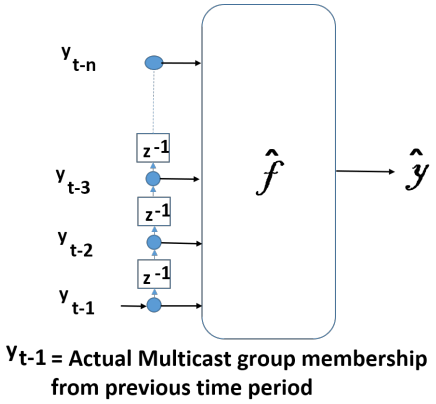
LiRP utilizes the historical data of multicast group membership in a TV studio that were discussed earlier. The historical data is used to create the required training and testing sets for input into K-fold cross validation algorithm. For the forecasting, the training set is derived by the historical series by creating the [(N-n-1) x n] input data matrix M. We have selected $n = 14$ and $N = 30$ based on expected patterns of two weeks and a month.

$$M = \begin{bmatrix} s_{N-1} & s_{N-2} & \cdots & s_{N-n-1} \\ s_{N-2} & s_{N-3} & \cdots & s_{N-n-2} \\ \vdots & \vdots & \vdots & \vdots \\ s_n & s_{n-1} & \cdots & s_1 \end{bmatrix} \tag{6.3}$$

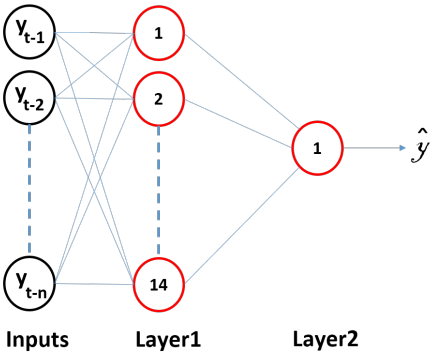
Matrix M shown in (6.3) represents the training set that will be used as an input to machine learning one step forecasting. Matrix Y in (6.4) represents the output of the training set.

$$Y = \begin{bmatrix} y_N \\ y_{N-1} \\ \vdots \\ y_{n+1} \end{bmatrix} \tag{6.4}$$

In *LiRP*, We utilize a multi-layer perceptron (MLP) model [6] with two layers as show in Figure 6.5b. Where the group memberships for the last 14 days get fed into MLP model with two layers and the output is the predicted multicast group



(a) *LiRP* approximation function



(b) Deep neural network

Figure 6.5 One-step forecasting. The approximation function $\hat{f}$ returns the prediction of the value of the time series at time t_{+1} as a function of the n previous values (the rectangular box represents a unit delay operator, i.e., $y_{t-1} = z^{-1} \cdot y_t$).

membership that feeds into *LiRP* algorithm line 3. The prediction for a multicast group is done for a specific time duration. We assume a time epoch of 24 hours.

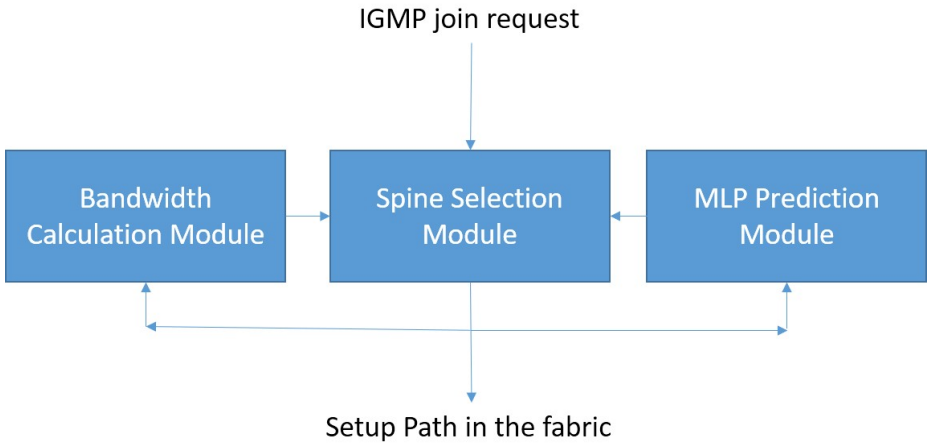


Figure 6.6 *LiRP* system modules.

6.3 Algorithm and System Architecture

LiRP algorithm adds a neural networks module in *iRP* controller and utilized for multicast group prediction information to influence the placement of multicast groups. The additional *NN* module is implemented inside the *LiRP* controller as shown in the system architecture diagram in Figure 6.6. *LiRP* introduces a link utilization virtual weight to track anticipated utilization of the link based on anticipated multicast receivers. *LiRP* pseudocode is shown in Algorithm 3. *LiRP* algorithm augments *iRP* through the inclusion of group prediction in line 5 and virtual trackers of link utilization in line 6 as inputs. Group prediction in line 5 is the output of *MLP* function reviewed in the previous section and illustrated in Figure 6.5.

In *LiRP*, the spine selection is influenced by both the available bandwidth to accommodate the existing multicast request as well as virtual utilization of the links that takes into account the anticipated demand from future receivers that we expect to belong to the same group but did not receive the request yet. This is reflected in the algorithm line 20 and line 21. As the receivers arrive in online fashion, the spine selection will be influenced by virtual utilization of links rather than only actual utilization leading to better selection of spines for a new tree being setup.

6.4 System Implementation

LiRP system testing was done using an IP fabric with 16 leafs and 4 spines. The test bed was then scaled up to 32 leafs and 6 spines to reflect testing conditions with a scalable fabric. The switches used to build the fabric are commercially available data center switches that run traditional unicast routing protocols to build routing tables and reachability in the fabric. The system does not require switches to support OpenFlow or other specialized SDN protocols. The leafs are Cisco's Nexus 93180 switches running NxOS 9.2(1) firmware and the spines are Cisco Nexus 9236 switches running same firmware [106] as shown in Figure 4.1. The leafs are connected to the spines using 6x100Gbps fiber links distributed evenly between the spines. For

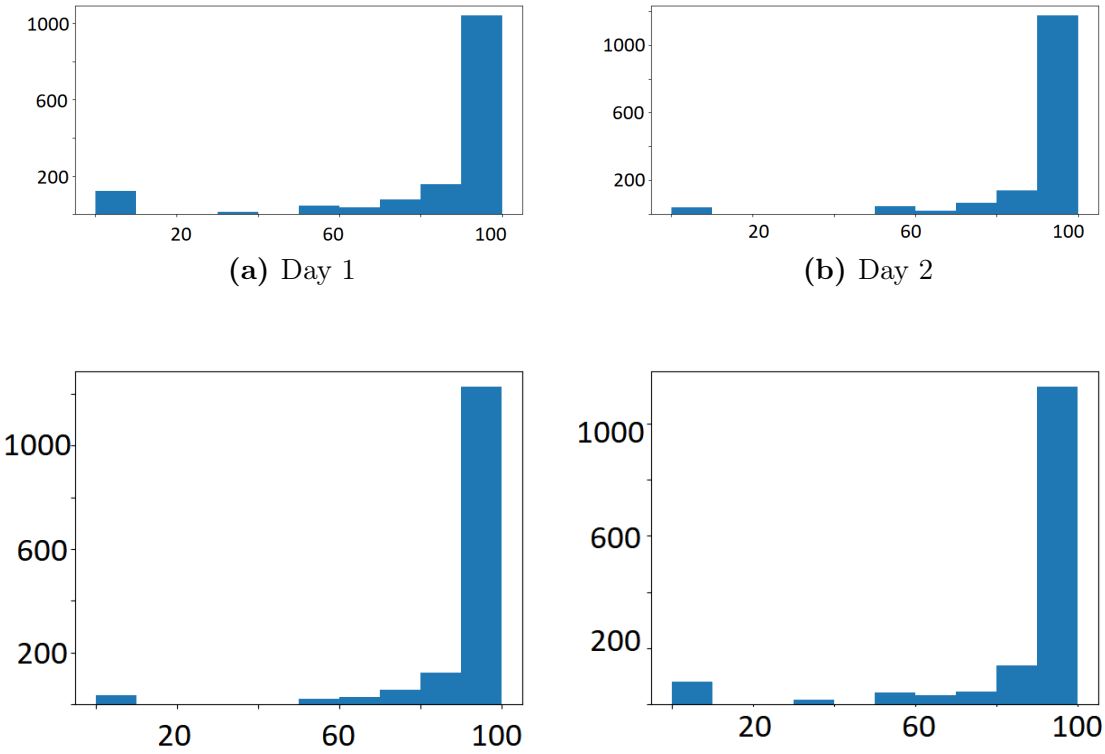
Algorithm 3 LiRP Algorithm Pseudocode

Inputs

Input $Spines[]$, $Leafs[]$
Input $SourceLeaf$, $RecieverLeaf$
Input $Bandwidth$
Input $GroupSpine$
Input $PredictGroup$ ▷ ML Predicted Group Membership
Input $AnticipatedLoad$ ▷ Anticipated Link Load Tracker

Steps

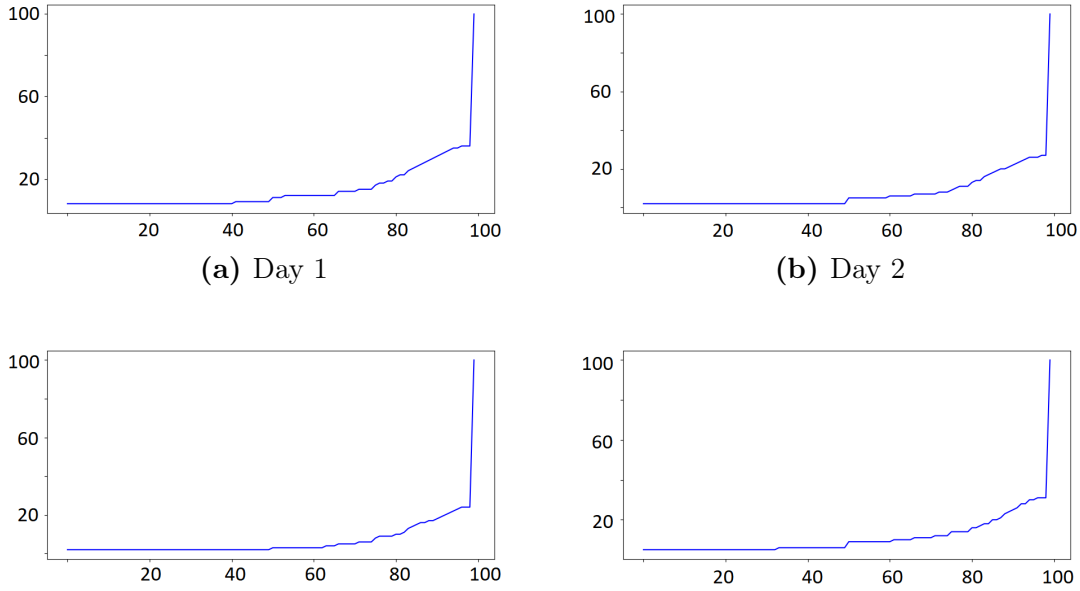
procedure LiRP
    if GroupSpine ≠ Empty **then**
        if Spine to RecieverLeaf Bandwidth > Bandwidth **then**
            $SelectedSpines[] \leftarrow Spine$
        else
            $SelectedSpines[] \leftarrow$ SpinesList
    else
        $SelectedSpines[] \leftarrow$ SpinesList
    if SelectedSpines[] = 0 **then**
        return Unavailable BW Error Code
    else
        SetupTree
function SpineList ▷ shortlist spines with BW
    for each S in Spines[] **do**
        if S to RecieverLeaf Bandwidth > Bandwidth **then**
            if SourceLeaf to S Bandwidth > B **then**
                $SelectedSpines[] \leftarrow Spine$
    return SelectedSpines[]
function SetupTree ▷ establish source tree
    $SelectedSpine \leftarrow Spinewithmostvirtual BWtoRecieverLeaf$
    $ReceiverLink \leftarrow linkwithmostvirtual BWtoRecievLeaf$
    $SourceLink \leftarrow linkwithmostvirtual BWfromSourceLeaf$
    $RecieverLink \leftarrow RecieverLink - Bandwidth$
    $SourceLink \leftarrow SourceLink - Bandwidth$
    $RecieverLink \leftarrow RecieverLink - virtualBandwidth$ for all anticipated receivers
    $SourceLink \leftarrow SourceLink - virtualBandwidth$ for all anticipated receivers
    Setup Source Tree
    return



(a) Day 1 **(b)** Day 2

(c) Day 3 **(d)** Day 4

Figure 6.7 Histogram showing forecasting accuracy - Neural Network one-step forecasting in 4 different days from TV studio traces - Y axis represent number of groups in a bin.



(a) Day 1 **(b)** Day 2

(c) Day 3 **(d)** Day 4

Figure 6.8 CDF showing forecasting accuracy - Neural Network one-step forecasting in 4 different days from TV studio traces - Y axis represent percent accuracy of prediction.

multicast sources and destinations, the setup contains 30 nodes connected to each leaf for a total of 960 nodes serving both as sources and as destinations. The setup utilizes a combination of commercial grade video nodes that generate Uncompressed HD video streams as well as traffic generators. During the testing phase, we introduce multicast senders and receivers gradually through automation using python code and we capture testing results accordingly. *LiRP* controller software is running inside a VM which is running on top of ESXi hypervisor on Cisco UCS C240-M4 server x86 based server with Intel Xeon Processor E5-2699v4 and 256G of memory. A picture of the actual racks that contain IP Fabric as well as server equipment is in Figure 4.1.

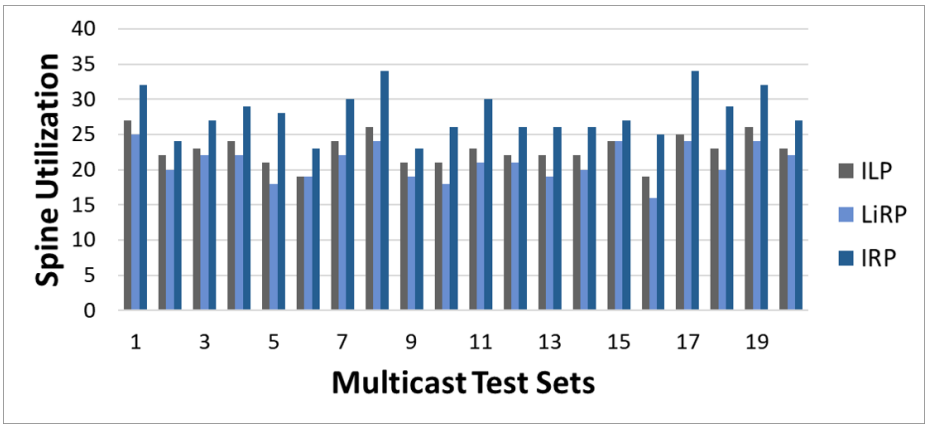
6.5 Experimental Results

We will start by analyzing accuracy of *NN* prediction algorithm utilized. For the analysis, we utilize actual traffic patterns captured in a TV studio and applying these patterns as training and testing sets in the *NN* algorithm used. Figure 6.7 shows

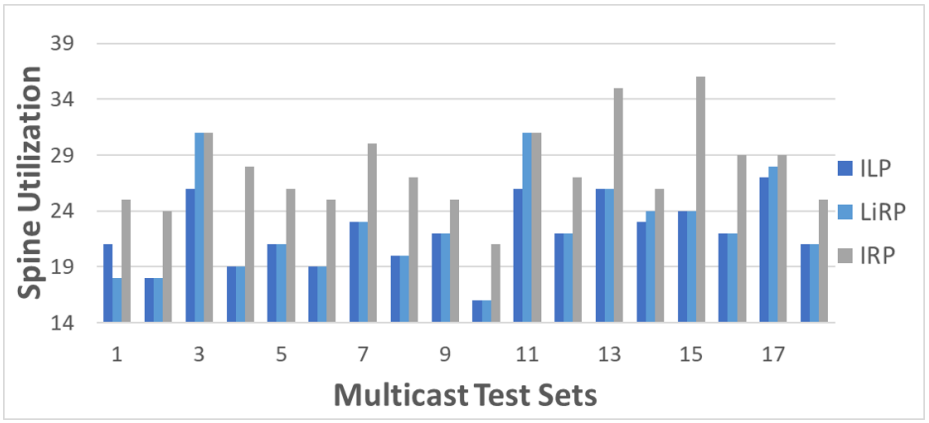
one-step forecasting histogram and Figure 6.8 shows one-step forecasting empirical CDF. The results utilize real TV studio traces results of the one step forecasting NN algorithm discussed in Section 6.1. The results show very high accuracy of group membership prediction for a certain receiver where more than 70% of the group membership is 100% accurate and the group with 0% accuracy is a small percentage. This confirms the intuition that the multicast traffic patterns are repetitive enough to allow for highly reliable one step forecasting.

Once prediction algorithm performance is reviewed, we move to compare the multicast tree setup performance results of Multicast PIM , ILP, IRP and $LiRP$ systems in a real folded Clos fabric using COTS switches from Cisco systems. All tests are done using four test sets, one test set using the Fabric with PIM control plane. The same test suite is then run against a fabric running iRP and $LiRP$ controller systems. Offline optimization ILP results are obtained utilizing a simulator built using Python. To increase the scale of the test, we generate traffic patterns that are like the TV studio traces observed with expanded set of sources and receivers. The test sets generated vary in the fan-out size per group which is a uniformly distributed variable between one and the max number of leafs. The accuracy of group membership prediction matches patterns seen in captures from real media DCs described in section 6.1 while scaling out to match the number of leafs in the system setup.

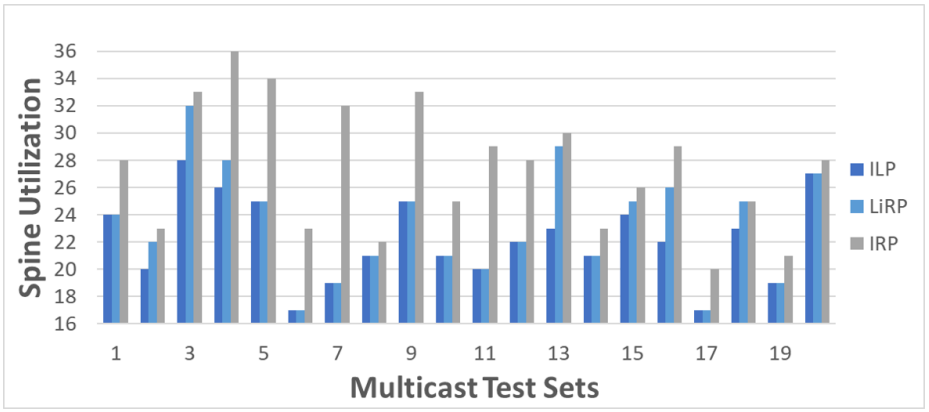
In Figure 6.9, we compare average spine utilization for the same number of multicast groups serviced using iRP, $LiRP$ and ILP. Higher spine utilization indicates less optimized multicast tree positioning leading to lower fabric capacity. Figure 6.9a provides results assuming 100% accurate prediction by $LiRP$ while Figure 6.9b and Figure 6.9c provides the same results with 10% and 33% $LiRP$ prediction error. ILP providing a benchmark for the highest fabric capacity through ability to optimize by having access to full demand matrix. $LiRP$'s performance is very close to that of ILP while maintaining online arrival pattern of multicast



(a) 100% accurate ML prediction of group membership



(b) 90% accurate ML prediction of group membership



(c) 67% accurate ML prediction of group membership

Figure 6.9 Average spine utilization for same number of groups among iRP, $LiRP$ and ILP with various prediction rates.

requests and offering higher fabric utilization compared to both PIM and iRP. $LiRP$ is showing up to 40% improvement over iRP.

Our test results confirm the observations that were discussed in Chapter 2 that Machine Learning and Neural networks can be utilized to learn and apply learning towards multicast placement optimization. Even distribution of flows over ECMP routes as well as intelligent placement of source trees, both offered by $LiRP$ system, translate into significant increase of the number of multicast flows that can be allowed through the fabric without over-subscription for the same system bandwidth.

CHAPTER 7

MEDIAFLOW : PROFESSIONAL MEDIA IN-NETWORK VIDEO FLOW MULTICAST MONITORING

In live media production environment (such as TV studios and sports venues), IP networks are utilized to carry live video using multicast for point to multi-point delivery. TV production traffic patterns are unique due to ultra high bandwidth requirements and high sensitivity to packet loss that cause video impairments. Existing network monitoring tools are either reactive by design or perform generic monitoring of flows with no insights into video domain. In this chapter, we introduce $MediaFlow$, a robust system for active network monitoring and reporting of video quality for thousands of flows simultaneously using a fraction of the cost of traditional monitoring solutions. To the best of authors knowledge, $Mediaflow$ is the first of its kind to offer active in-network monitoring of video flow quality. We implement $MediaFlow$ using data center switches and our testing results confirm that $MediaFlow$ reduces video error detection and correction time from minutes to milliseconds as compared to current state-of-the-art methods. $MediaFlow$ is able to detect and report on integrity of video flows at a granularity of 100 mSec at line rate for thousands of flows. The system increases video monitoring scale by a thousand fold compared to edge monitoring solutions.

7.1 Introduction

As discussed in Chapter 1, packet-based transport networks are widely utilized in media industry for professional content production [107] such as in TV stations, newscasts and live sports events. This trend is supported by the rapid increase in bandwidth availability, increased reliability as well as steep decline in IP transport price per bit [14]. This is in-line with industry projections (Cisco's Network Visual

Index study [107]) where 82% of all IP traffic is forecasted to be video traffic by 2022 with annual growth rate of 34%.

In professional media production, a video flow from a sender might be consumed by multiple receivers leading to the need for point-to-multipoint communications [69, 99, 102]. Multicast family of protocols offers an efficient mechanism for delivering point-to-multi-point communications for multimedia and real time video delivery [34]. Reliable delivery in packet networks is often based on the re-transmission at the cost of longer end-to-end delay. This approach does not work for real-time video applications as they require low latency for delivery (e.g., news and sports content). Such applications utilize multicast for point to multi-point delivery that does not easily support automatic error recovery. State of the art video delivery methods utilize unreliable transport protocols such as RTP (real-time transport Protocol) [93]. As Multicast is unidirectional in nature, RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio and video. RFC 6792 [86] describes the current framework for video quality monitoring using RTP metrics. Figure 7.1 represents RFC 6792 framework which relies on edge monitoring of video flow quality leading to expensive specialized monitoring tools that do not scale well.

Edge monitoring [89] has been utilized to monitor video quality. However, in-network monitoring of video quality received no attention due to limitations in switching hardware that can only support sampled flow capture. Sampling of video flows for monitoring leads to information loss and is not able to detect granular per flow information. New generations of programmable DC ASICs (Application Specific Integrated Circuits) are made available that are capable of deeper packet inspection without impacting throughput of the switch. A popular example of such ASICs is the Tofino ASIC [111] that has implemented P4 [13] programmable capabilities. Cisco's Cloudscale ASICs [28] offer capabilities for packet header inspections that is utilized

in this work. Cisco's ASICs are widely deployed in DC switching environments with varying estimates of more than 50% market share [29].

To the best of authors knowledge, *MediaFlow* is the first system to implement in-network monitoring of video flows using RTP sequence tracking to generate insight into the quality of the video domain.

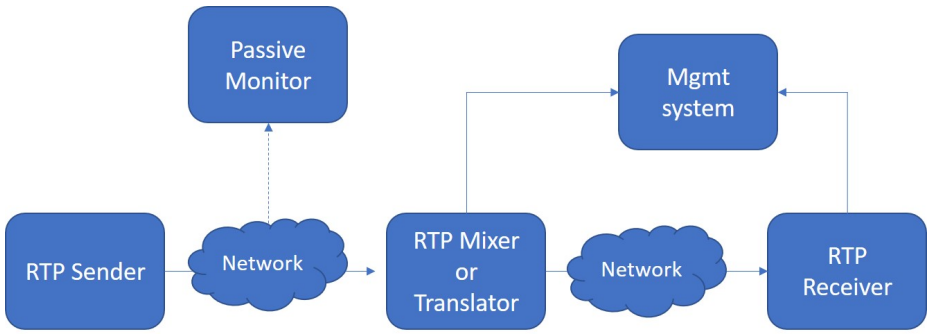


Figure 7.1 RFC 6792 : Example showing RTP monitoring framework.

To this end, we design and implement *MediaFlow*, a robust system for real-time monitoring and reporting of video quality for thousands of flows at a fraction of the cost of existing network monitoring solutions. *Mediaflow* is first in its kind to utilize unique packet processing features in DC switching ASICs and combine that with a controller implementation to achieve scalable real-time monitoring and reporting of media flows. The unique implementation achieves this mentoring scale at a fraction of the cost of traditional edge monitoring solutions [17, 70, 51, 114].

In this chapter, we provide the following contributions:

1. We introduce a novel video flow quality metric that tracks video quality per flow on each edge in the fabric.
2. We develop *MediaFlow*, an active in-switch video quality monitoring algorithm that can identify packet loss impacting a particular video flow and reroute traffic to recover.
3. We implement *MediaFlow* system using COTS switches. We present test results using HD Video flows comparing *MediaFlow* results against existing monitoring solutions. Test results confirm that *MediaFlow* Algorithm is able to lower detection and recovery measurements by up to 99% while ensuring accurate admission of multicast flows based on bandwidth requirements.

7.2 The Case for *MediaFlow*

TV and broadcast video production is one of the last domains that is not fully IP enabled and where Serial Digital Interface (SDI) [101] continues to be widely deployed as a transport protocol. This work is inspired by the technology transition happening in the uncompressed Video transport technologies from legacy non-IP format based on Serial Digital Interface (SDI) [101] to transport based on IP and the emergence of new set of standards, namely SMPTE ST2110 suite of protocols that target Professional Media Over Managed IP Networks [81]. The work in [30] and [55] provide an overview of the live video industry current challenges, activities and forward looking direction to migrate to IP based video flows.

SMPTE ST2110 protocol [81] provides a pathway to carry uncompressed video flows over IP networks. The migration to IP for broadcasters and content providers brings great business value as it unlocks productivity and allows to benefit from economies of scale that IP and Ethernet offer. However, such architecture comes with risk as operations and monitoring of such infrastructure is not well defined.

Professional video content originates from the capture point (a professional video camera) and goes through various stages of processing and storage (e.g., professional video editing software) that lead to the creation of a final product in the form of a sports game, live news or an episode of a popular show.

In such workflow, video quality is of the highest importance as video is the final product being produced and packet loss is not tolerated well as it impacts the quality of the created video. Figure 7.2 represents a contribution network use case for a live sports production in a sports venue such as a stadium where the sports content is processed locally and then sent back to Media DC for further processing. Video traffic is carried using multicast with very high bandwidth requirements driven by video format used and can be in uncompressed or compressed format. The processed (yet still live) content would then be sent over wide area IP networks (WAN) to central

video production studios for final production before sending out to consumers. The WAN network is usually managed by a third party service provider and the content provider might not have visibility nor control on the WAN network. From the WAN, live video traffic makes it to the central production facility (TV studio) for final processing before sending the sports show to air to be watched by consumers. Video flows continue to be carried using multicast and are usually in uncompressed format with throughput rates described in Table 7.1.

There are multiple places in the described production chain where video quality could be impacted. Problems can be outside the network such as errors during camera capture or artifacts due to video compression and transmission link imperfections [57, 95, 109]. Operating IP networks in such environments is very challenging as the network is the first to be blamed for imperfections seen in video quality. Network operators need deep network visibility and an efficient way to quickly and effectively locate video problems and their root causes if packets are being dropped from video streams or if the video quality issues are due to other factors [86, 110]. The deterioration of video quality is often observed on the receiver side as pixelation and impurities in the reconstructed video images.

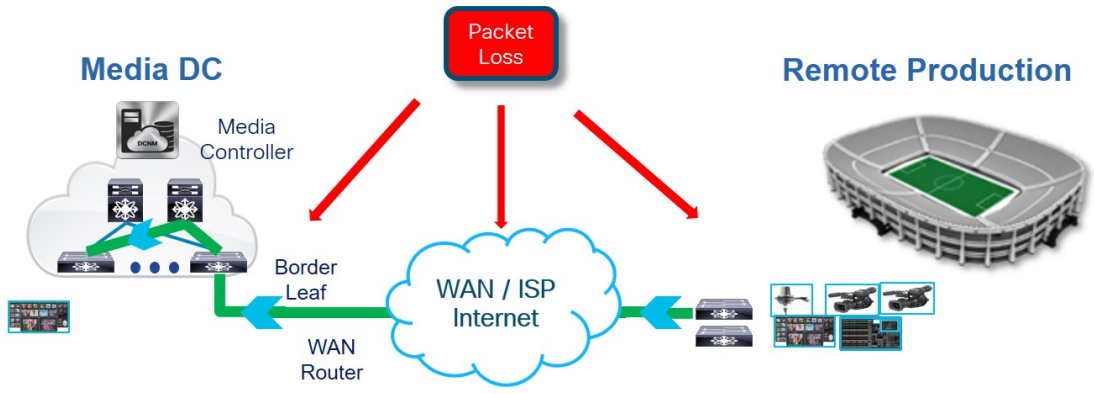


Figure 7.2 Packet loss in different parts of the production chain.

Existing commercial monitoring solutions are based on reactive selective monitoring where the impact on a video flow needs to be detected visually by an

operator using a visual monitoring station. The monitoring station monitors a subset of the flows based on level of business impact of the video flow (to-air video signals are clearly more important than others). When the operator notices a visual artifact, the impacted flow is then cloned to another deep inspection station to analyze the health of video flow and try to take corrective measure. This reactive approach takes long time to execute, is prone to errors and does not scale to thousands of flows.

State-of-the-art solutions like [123, 75, 67] are not focused on video quality monitoring but rather a generic packet flow error tracking which do not provide actionable video-related insight. To address scalability issues in tracking flows, they utilize either sampling approach or selective routing of flows to a monitoring station based on a trigger. Both approaches are clearly not adequate to proactively locate and rectify network issues affecting video flows at scale.

Mean time to detect (MTTD) and Mean time to repair (MTTR) are key operational metrics for network operators. MTTD represents the time it takes for an operator to detect an error in a particular video flow, while MTTR is the total time needed to recover the flow from errors.

As shown in Figure 7.2, The issue with Video QoS monitoring is more challenging when parts of the network is not managed by the organization (WAN circuit, VPN circuit, etc). This is a popular deployment model especially in live sports production environment at sports leagues to transport the production video signal from the sports venue (stadium, ball park or field) back to the Media DC for further processing before sending the final video content out. The impact of packet drops in or between switches can not be correlated to the video flows going through the link or switch. This leads to a very high MTTD and MTTR resulting in video and audio artifacts in high quality productions ultimately impacting customer quality of experience (QoE). In real-time video production and delivery domain, sub-second measurements are the expected norm. State of art MTTD and MTTR

are in magnitudes of minutes which do not satisfy business requirements of the video application real time monitoring and high availability.

7.3 Definitions and Problem Description

Most of the work in the literature is focused on the delivery of an already produced and final content to end users [33, 78, 5, 10]. In contrast, the work in this chapter is focused on professional multimedia content generation such as in TV studios, News production and Sports production where handling live video traffic puts demand on the fabric to achieve high availability and low latency for the traffic flows going through the fabric.

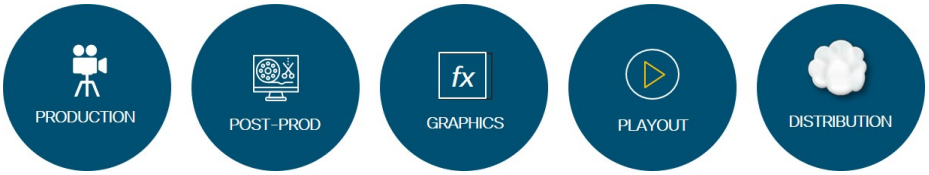


Figure 7.3 Media functions in a live broadcast facility.

In a video production environment, SMPTE ST2110 flows can be carried within the production facility (intra-facility) but also in between geographically separated production locations (inter-facility). The workflow requires the content to be transferred between many video processing end points that perform a media function (which is very similar to a network function concept) as depicted in Figure 7.3.

Table 7.1 Video Data Rates In Broadcast Facility

VIDEO FORMAT	DATA RATES in Mbps [101]
Standard Definition SD-SDI	270
High Definition HD-SDI 1080i	1485
3G-SDI 1080p	2970

The traffic patterns of uncompressed video within a broadcast studio (TV stations, sports venues, etc.) consist of thousands of video flows that are very large in throughput requirements as they carry uncompressed video. Table 7.1 shows the rates for Uncompressed video which could be 1.5Gbps, 3Gbps, 12Gbps and growing as video

formats are evolving [99]. Carrying uncompressed video format allows professional video environments to achieve high quality video distribution while meeting strict latency requirements [112, 88, 59].

This means that three to six of these video streams can fill up a 10G link. Such Video flows do not tolerate packet loss as packet loss severely impacts the delivered video quality leading to image pixelation or total loss of image. Another characterization of the uncompressed video flows is that they are long lived in nature and utilize best-effort UDP transport protocol for delivery.

Within this context, $MediaFlow$ focus is on minimizing the detection time (MTTD) and resolution (MTTR) time of packet loss affecting video flows.

$$MTTD = T_r + T_s + T_d \tag{7.1}$$

To calculate MTTD we measure the time interval from the start of the network fault to the time the fault is reported and localized properly when an operator gets an alarm identifying the location of network errors. Equation (7.1) describes the components included in MTTD calculation where T_r is the reporting time for human eye to detect an issue with a video on monitoring screens, T_s is the time it takes to route the impacted flow to an edge device for monitoring and T_d is the time needed for edge device to analyze and detect video flow issues. Of these values, T_r is the dominant factor as it is usually human driven. $MediaFlow$ reduces MTTD significantly because it utilizes in-network monitoring leading to both T_r and T_s being almost zero.

$$MTTR = MTTD + T_l + T_{reroute} \tag{7.2}$$

MTTR represents the total time interval from the start of the network fault to the time it is successfully mitigated. Equation (7.2) extends (7.1) where the additional parameters are T_l which is the time needed to localize the link where the packets

were lost in the fabric and $T_{reroute}$ is the time needed to reroute an impacted flow around a problematic link. Similar to MTTD, $MediaFlow$ reduces MTTR in orders of magnitude because in addition to lower MTTR, T_l is significantly lower and very close to zero.

Typical multicast flow scale varies depending on the facility sizing and could range from a couple of hundreds of flows to hundreds of thousands of flows. Authors own measurements of multicast flows in a major live sports production facility (based in North America) show 24,100 active multicast flows at any one point in time during video production cycles. These flows represent a mix of video, audio and ancillary data as described in SMPTE ST2110 protocol [81]. A typical Inter- and Intra- facilities system architecture is depicted in Figure 7.2.

For System modeling, we consider a fabric carrying traffic for video production system where the underlying fabric is CLOS fabric with L leafs and S spines (Figure 7.4). Video sources such as a video camera ($1 < vs < VS$) are attached to Leafs ($1 < l < L$) and generate video content that is transported using multicast trees. The video receivers such as video editing stations ($1 < vr < VR$) are attached to one of the leafs and subscribe to multicast streams. Production facilities might be spread geographically and would be interconnected via wide area network (WAN) links connecting border leafs. Such facilities would vary in size depending on production requirements and might consist of a single switch or a larger CLOS fabric.

Let X_{gs} be the state of the path between the source of the multicast group and one of the spines. The state is equal to 1 if the path is utilized for a group and zero otherwise. Similarly, let Y_{rs} be the state of the path between a receiver of the multicast group and one of the spines with the state is equal to 1 if the path is utilized and zero otherwise.

As this system is based on CLOS fabric, it can be modeled as a directed graph with V vertices and E edges. Each vertex could be an end-point or leaf/spine. Each

edge represents a network link between two vertices. The edge capacity represents available bandwidth between the vertices. Clos fabric modeling tracks edge utilization U_t where $0 < U_t < U_{max}$ and U_{max} is maximum link utilization.

Without loss of generality, we assume that all multicast streams require same throughput of 1 unit. We assume that the demand matrix for sources and destinations for each of the trees is known in advance. We assume online arrival and departure of video flows as this reflects a typical professional video production environment where flows are routed through the facility depending on the actual needs of the producers at that particular time.

We also introduce flow state metric F_f to track video flow quality (or lack of it) in a particular edge. For simplicity, we set F_f to zero when errors were reported on that edge and we set F_f to 1 when no errors are reported for that flow on that particular edge.

We formulate the off-line problem of optimizing the distribution of the trees to increase overall fabric utilization while satisfying the constraints of not exceeding available bandwidth on each of the links using integer linear programming (ILP) as follows:

$$minimize \sum_{g} \sum_{s} X_{gs} \tag{7.3}$$

s.t.

$$\sum_{g} \sum_{s} X_{gs} \leqslant BW, \forall l \in [L], \forall s \in [S] \tag{7.4}$$

$$\sum_{r} \sum_{s} Y_{rs} \leqslant BW, \forall l \in [L], \forall s \in [S] \tag{7.5}$$

$$F_{rs} = 1, \forall s \in [S], \forall r \in [R] \tag{7.6}$$

$$\sum_{s} X_{gs} \geqslant 1, \forall l \in [L], \forall g \in [G] \tag{7.7}$$

$$\sum_{s} Y_{rs} = 1, \forall l \in [1, L], \forall r \in [R] \tag{7.8}$$

$$Y_{rs} - X_{gs} \leqslant 0, \forall s \in [S], \forall r \in [R] \tag{7.9}$$

$$X_{gs} \geqslant 0, \forall s \in [S], \forall g \in [G] \tag{7.10}$$

$$Y_{rs} \geqslant 0, \forall s \in [S], \forall r \in [R] \tag{7.11}$$

The optimization objective is to minimize the total number of uplinks from the source leafs X_{gs} when creating the multicast trees to achieve the desired connectivity between multicast sources and destinations while tracking both bandwidth and flow error constraints. The constraints in (7.4) and (7.5) ensure that the total number of flows X_{gs} or Y_{rs} mapped through an edge do not exceed the total bandwidth of that edge BW. Constraint set (7.6) tracks the flow error reporting status per edge and ensures edges with reported errors are not utilized. Constraint set (7.7) ensures that the source is mapped to one of the spines while (7.8) ensures that total links towards a receiver is one. Constraint set (7.9) ensures that the receiver is mapped to a spine where the needed source is mapped to. i.e., ensure that ILP takes into consideration the source to spine mapping when selecting the destination to spine mapping.

The above optimization problem assumes that multicast demand matrix is known ahead of time which is not realistic for many of the workflows that rely on multicast to carry traffic. Much of the existing research reviewed in Chapter 2 assume full knowledge of demand matrix and thus utilizes offline optimization techniques for building multicast trees. In real world deployments of multicast, demand matrix is not known ahead of time and multicast requests arrive in an online fashion.

In the next section, we discuss *MediaFlow* algorithm that optimizes multicast tree creation assuming online arrival of multicast requests with unknown demand matrix while taking into consideration network errors that affect video flow quality.

7.4 MediaFlow Algorithm

In this section, we describe Professional Media In-Network Video Flow Multicast Monitoring *MediaFlow* Algorithm. *MediaFlow* algorithm extends our work in Chapter 4 to incorporate video error detection and recovery in the optimization. *MediaFlow* algorithm assumes online arrival of multicast flows.

MediaFlow algorithm consists of two main components, (i) In-network error detection of video flows and (ii) Video flow error recovery utilizing SDN concepts to reroute video flows around areas where the error or loss occurs. Both components are described in the following sections:

7.4.1 Video Flow Error Detection

For Video Flow Error Detection, *MediaFlow* algorithm utilizes the switch ability to report on gaps in rtp flow sequences as described in Section 7.1. *MediaFlow* algorithm utilizes real time video error detection capabilities available in Cisco's Nexus 9000 switching platform [79] namely the Media Flow Analytic capabilities [36]. These capabilities rely on Cisco's Cloudscale ASICs that are able to read first 128 bytes of packet headers, monitor RTP sequences and report on any gaps in sequences. ASIC provides a unique ability to monitor video flows inline as they cross the network and does that at scale. ASIC is widely deployed in enterprise networks however *MediaFlow* is the first system to utilize the above unique ASIC ability.

iRP algorithm, described in Chapter 5, is able to achieve better fabric utilization compared to the standard, namely *PIM*. However, *iRP* focuses on bandwidth and distance for defining flow path and does not track video flow quality as part of its metrics.

MediaFlow algorithm extends *iRP* algorithm by adding a module for flow state metric tracking as shown in line 1 of Algorithm 4. The module introduces a new metric F_f to report flow state on edges and track in *MediaFlow* controller through complete visibility into the fabric. *MediaFlow* monitors flow errors reported by all switches in the fabric. This is done by correlating error reporting for a particular stream between the sender switch and receiver switch that are located on both sides of a link. If the upstream switch is not reporting errors for a particular flow while the downstream or receiver switch is reporting errors for the same flow then the algorithm flags the link and flips the value of the F_f metric from 1 to 0 for that particular flow. This would trigger a notification to the operator through notification APIs. An API sample of notifications is described in Table 7.3

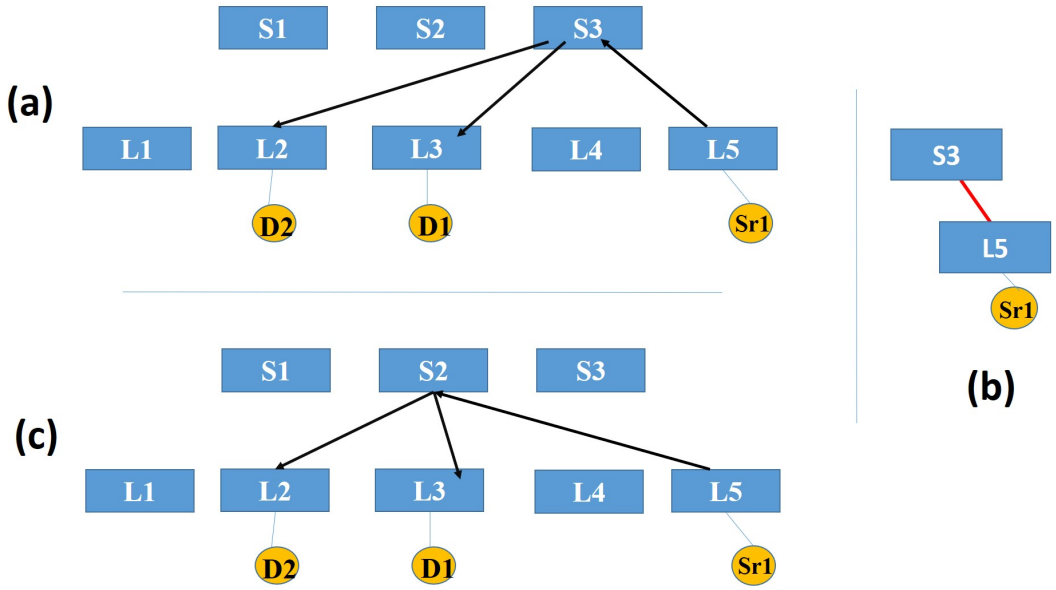


Figure 7.4 Multicast in a CLOS fabric (a) Initial flow state without errors. (b) Reporting of RTP flow errors on spine S3. (c) Recalculation Of flow path towards spine S2.

Figure 7.4 provides an example of flow error detection. Figure 7.4 (a), Source $Sr1$ is sending a video flow with two receivers $D1$ and $D2$ utilizing Spine $S3$ in the path. $S3$, $L2$ and $L3$ start reporting RTP errors for this particular flow while $L5$ does not report errors. This allows *MediaFlow* controller to localize the problematic edge

to the edge connecting $L5$ to $S3$. $MediaFlow$ controller would lower metric F_f for this edge to zero indicating an impact to the flow. Algorithm 4 shows the psudeocode for $MediaFlow$ algorithm.

7.4.2 Video Flow Error Recovery

$MediaFlow$ algorithm utilizes a centralized controller to perform multicast path calculation. $MediaFlow$ extends iRP algorithm, by adding capability to reroute traffic based on reported flow state metric F_f on a specific edge. $MediaFlow$ module incorporates the metric in the online multicast path calculation. As described in Section 7.4.1, a change in F_f metric for a particular edge triggers $MediaFlow$ Algorithm to recalculate multicast path.

Multicast flow state includes the list of edges and vertices representing the fabric. $MediaFlow$ starts by checking if there is enough bandwidth in other edges that connect the two vertices (the upstream and downstream switches). It is likely that there is another edge that has enough bandwidth to handle the needed bandwidth for the flow. The algorithm would move the flow to the identified link to recover service. the selected edge bandwidth metric would be reduced to reflect the added bandwidth utilization of the new flow. This would be the least impactful option to the rest of the flows and the overall topology. The calculation has local scope (to the uplink switch) and thus has minimal execution time and minimal complexity.

If there is not enough bandwidth in any of the remaining links between the two vertices identified (the spine and the leaf), $MediaFlow$ would go through a full algorithm cycle to move the flow to another vertex (spine). It will pick a viable spine from list of available spines by confirming bandwidth availability on the spines both to the receiver leaf and from sender leaf in line 19 and line 20. The list of compliant spines will then be parsed and the spine with the highest bandwidth will be selected in line 24. The same is done for selecting the link with most available bandwidth in line 25 and then the tree is setup across the fabric in line 29. The algorithm will result

Algorithm 4 MediaFlow Algorithm

Inputs

Input $Spines[]$, $Leafs[]$ ▷ list of Spines and Leafs
Input $SourceLeaf$, $RecieverLeaf$ ▷ Leafs where Source and Receivers are connected
Input $Bandwidth$ ▷ Required Bandwidth
Input $GroupSpine$ ▷ If the multicast group assigned to a Spine, list Spine here
Input $FlowState$ ▷ Tracks flow state metric

Steps

procedure FlowStateUpdate
    if New $FlowState \neq$ Current $FlowState$ **then**
        Current $FlowState \leftarrow$ New $FlowState$ ▷ Update for each edge
        return $UpstreamSwitch$, $DownstreamSwtich$ ▷ return upstream and downstream switches with error link
procedure MediaFlow
    if GroupSpine $\neq$ Empty **then**
        if Spine to RecieverLeaf Bandwidth $>$ Bandwidth **then**
            $SelectedSpines[] \leftarrow Spine$
        else
            $SelectedSpines[] \leftarrow$ SpinesList
    else
        $SelectedSpines[] \leftarrow$ SpinesList
    if SelectedSpines[] = 0 **then**
        return Unavailable BW Error Code
    else
        SetupTree ▷ Setup needed tree
function SpineList ▷ shortlist spines with BW
    for each S in Spines[] **do**
        if S to RecieverLeaf Bandwidth $>$ Bandwidth **then**
            if SourceLeaf to S Bandwidth $>$ B **then**
                $SelectedSpines[] \leftarrow Spine$
    return SelectedSpines[]
function SetupTree ▷ establish source tree
    $SelectedSpine \leftarrow SpinewithmostBWtoRecieverLeaf$
    $ReceiverLink \leftarrow linkwithmostBWtoRecieverLeaf$
    $SourceLink \leftarrow linkwithmostBWfromSourceLeaf$
    $RecieverLink \leftarrow RecieverLink - Bandwidth$
    $SourceLink \leftarrow SourceLink - Bandwidth$
    Setup Source Tree
    return

in moving the flow to another spine and is done based on the available bandwidth and flow state metrics with the intention to bypass the impacted edge. The calculation is done from the impact point onward towards the receivers to minimize the impact on other receivers who are not affected by that particular edge.

In line 5, algorithm checks availability of alternate local links with available bandwidth to accommodate the impacted flow. If a link is found, flow is moved to the new link in line 23. If the above is not true, the algorithm tries to pick a viable spine from the list of available spines by confirming bandwidth availability on the spines both to the receiver leaf and from sender leaf in line 19 and line 20. The list of compliant spines will then be parsed and the spine with the highest bandwidth will be selected in line 24. The Same is done for selecting the link with the most available bandwidth in line 25 and then the tree is setup across the fabric in line 29.

Continuing the example in Figure 7.4 from previous section, and following the reporting by $S3$, $L2$ and $L3$ of RTP errors for this particular flow, $MediaFlow$ controller would lower metric F_f for the problematic edge connecting $L5$ to $S3$ to 0 indicating an impact to the flow. $MediaFlow$ algorithm would recalculate a new path for the impacted flow utilizing available metrics for all edges including available bandwidth and F_f for that particular flow as described in line 4. Spine $S2$ would now be hosting the impacted flow and forwarding to both receivers.

$MediaFlow$ algorithm executes in polynomial time $\mathcal{O}(S + L)$ where S is the number of spines and L is the number of leafs in the fabric. This is because $MediaFlow$ algorithm takes advantage of CLOS fabric structure where leafs are connected to all spines thus $MediaFlow$ decision is to select a new link in one of the spines as the path criteria. As a comparison, time complexity for offline optimization is proportional to the complete fabric size (all leafs and spines). In [44], time complexity is $\mathcal{O}(K^{\epsilon} log(n))$ where n is the network size, K is multicast request members, and $0 < \epsilon < 1$. The complexity of the algorithm in [21] is $\mathcal{O}(|V|^2|D|^2)$ where

V is number of vertexes in the fabric and D is the number of multicast destinations. As $S << n$ in a Clos fabric, *MediaFlow* complexity is much lower and has faster execution time compared to [44, 21] without utilizing specialized protocols such as OpenFlow that is required in the algorithms above. OpenFlow does not have wide commercial adoption despite its life span of more than ten years [82] which limits the possibility of deploying such algorithms in actual enterprise networks. We show performance and testing results in Section 7.6.

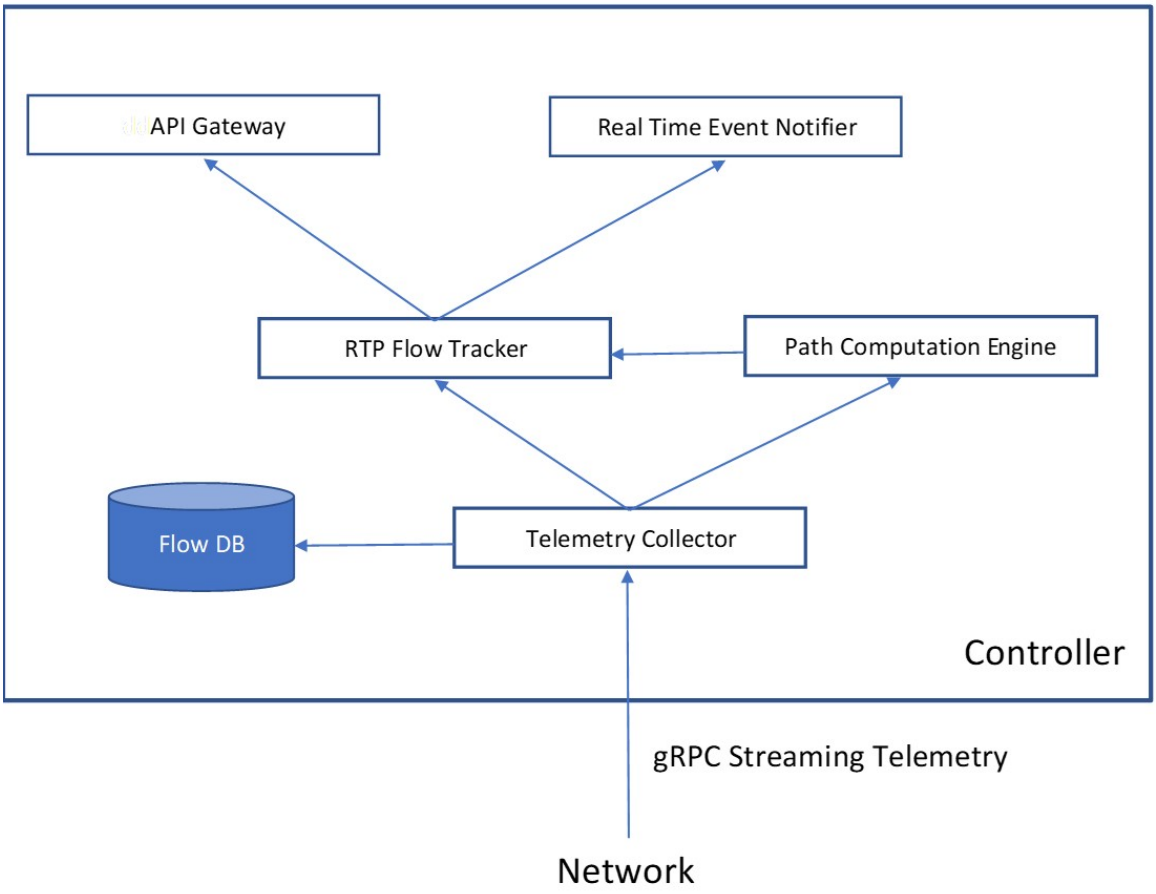


Figure 7.5 MediaFlow controller architecture.

MediaFlow controller internal architecture is shown in Figure 7.5. The controller stores flow connectivity, statistics and failures/drop information into persistent database (ElasticSearch). *MediaFlow* controller's API Gateway module

exposes flow, RTP traffic information–statistics, and packet drops via HTTP REST interface. Flow life-cycle events as well as any drops/failures can be received in real time fashion by subscribing to AMQP channel hosted by the controller.

MediaFlow Controller exposes HTTP(s) REST API to provide insight into media flow traffic integrity for external systems. Structure of the main APIs is shown in Table 7.2. *MediaFlow* controller tracks end to end flow in the network and actively monitors them for any glitches. If there is a packet loss, *MediaFlow* controller identifies affected flows and pinpoints node/device and involved incoming interface where drops are observed. Information can be retrieved via REST API as well as client can subscribe to real time notifications of video flow status. Structure of notification message is shown in Table 7.3.

Table 7.2 MediaFlow Controller Sample External API

API	DESCRIPTION
GET /rtp/fabrics/all/active	Returns active flows using 3 tuples of stream, traffic counters and incoming interface
GET /rtp/fabrics/all/error	Returns flows with packet drops using 5 tuples of stream, traffic counters, incoming interface, loss start time and packet drop count

7.5 System Implementation

MediaFlow system architecture is shown in Figure 4.1. Following SDN principles, *MediaFlow* algorithm utilizes a centralized controller approach to collect RTP flow statistics generated by switches in the network. Collection is done using streaming telemetry process running on network nodes using gRPC protocol [39]. The statistics are received by *MediaFlow* SDN controller that analyzes RTP flow statistics of each flow to identify and localize network locations where a particular video flow starts to experience errors.

MediaFlow SDN controller is the central point for multicast control plane and is responsible for tracking flow metrics and setting up the required source trees between

Table 7.3 MediaFlow Controller Sample Notification Payload

KEY	VALUE
message-id	Message Identifier
delivery-mode	Sync or Async
Event	Switch name and Event (add or delete)
Operation	Real Time Notification
Operation-Status	Success or Failure
Sent-At	Date and Time
Severity	Severity level Critical or Information
Type	Switch
User	Internal
content-encoding	UTF-8
content-type	application/json
Payload	*"Switch":"node-name", "id":"fault-id", "LossStart":12:18:19PST Apr 06 2020, "SourceIP":"10.33.56.10", "DestinationIP":"232.200.255.29", "LossStart":"12:18:19 PST Apr 06 2020", "PacketLoss":"450959", "SourcePort:33002, "DestinationPort":31002, "BitsPerSec":246998, "Protocol":17, "PacketCount":674537, "IFName":"Ethernet1/8/2", "status":"created"*

source and destinations for a multicast group. Streaming telemetry is utilized to stream video flow state changes from each switch to the controller. *MediaFlow* controller utilizes API calls to drive multicast flow stitching through the fabric driven by online arrival of sources and receivers.

MediaFlow SDN controller tracks RTP flow statistics reported by each node and calculates flow state metric F_f for each of the edges in the network where the controller compares the old state and new state of flow state metric reported on a particular edge based on change trigger. The change trigger could be reported by the switches in intervals of 100 uSec.

MediaFlow controller has a built-in gRPC receiver to collect streaming telemetry from all switches in the fabric. Network periodically pushes flow data and statistics and any flow failures/packet drops in real time via telemetry channel using gRPC protocol with JSON encoding. Using link layer discovery protocols and

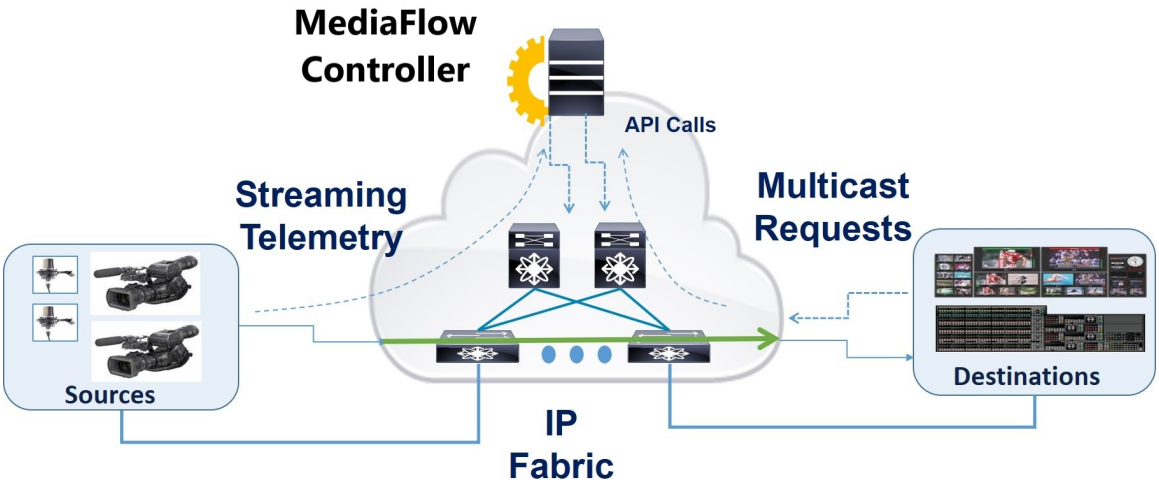


Figure 7.6 MediaFlow system architecture.

collating flow information from all switches, *MediaFlow* controller can establish end to end path for all flows.

MediaFlow controller software is implemented using Python and runs in a Centos 7 VM with 8 cores and 24GB of memoery. The VM is running on top of ESXi 6.5 hypervisor on Cisco UCS C240-M4 x86 based server with Intel Xeon Processor E5-2699v4 and 256G of memory.

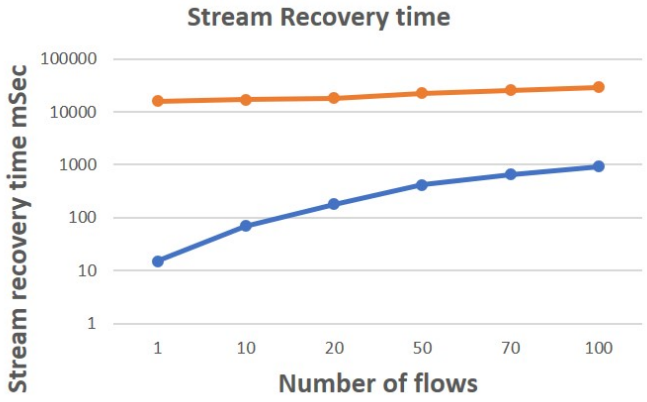
MediaFlow system testing is executed using an IP fabric with 16 leafs and 4 spines. This scale is representative of industry deployments for this use case. The testbed is then scaled up to 32 leafs and 6 spines to reflect testing conditions with a scalable fabric. The switches used to build the fabric are commercially available DC switches that are widely deployed in enterprise DCs. These switches run traditional unicast routing protocols to build routing tables and reachability in the fabric. The system does not require switches to support OpenFlow or other specialized SDN protocols to ensure wider scope of deployment. The leafs are Cisco's Nexus 93180 switches running NxOS 9.3(3) firmware and the spines are Cisco Nexus 9336 switches with 36x100G ports and are also running NxOS 9.3(3) [79] as shown in Figure 7.6. The leafs are connected to the spines using 6x100Gbps fiber links distributed evenly

(a) WAN Captures

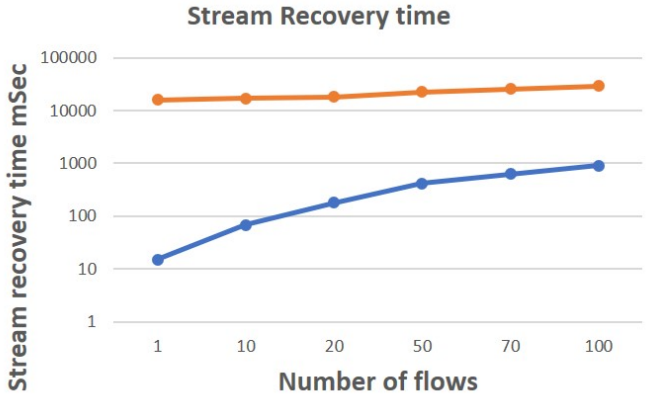
(b) Normal Distribution

(c) Poison Distribution

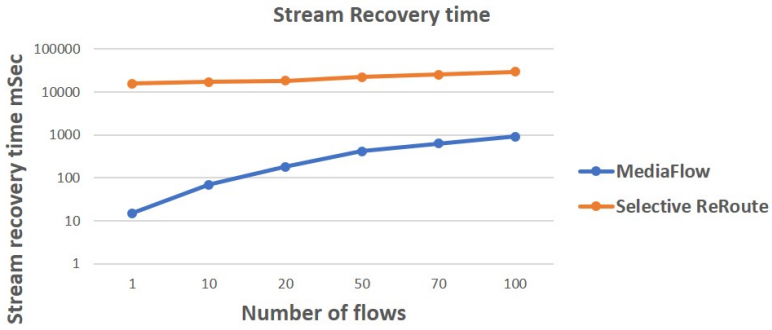
Figure 7.7 Video flow error detection time : Comparing mean time to detection of *MediaFlow* to edge monitoring methods.



(a) WAN Captures



(b) Normal Distribution

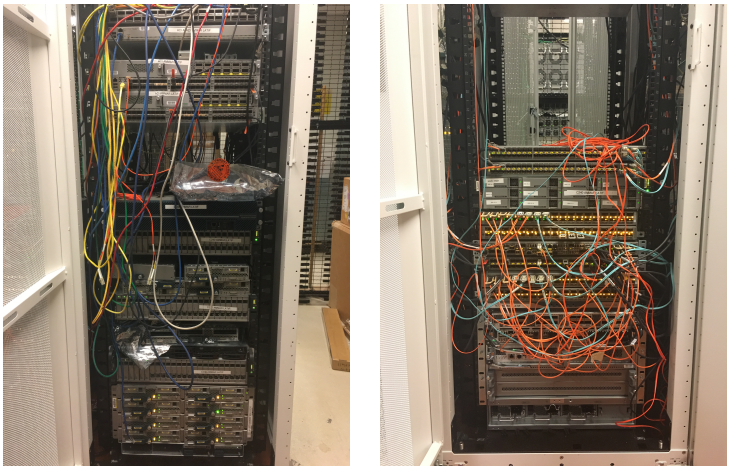


(c) Poison Distribution

Figure 7.8 Video flow stream recovery time : comparing mean time to recover of *MediaFlow* to edge monitoring methods.

between the spines. For multicast sources and destinations, the setup contains 30 nodes connected to each leaf for a total of 960 nodes serving both as sources and destinations. The setup utilizes a combination of commercial grade video nodes that generate Uncompressed HD video streams as well as traffic generators. During the testing phase, we introduce multicast senders and receivers gradually through automation using python code and we capture testing results accordingly. Main testing involves measuring packet loss detection times as well as recovery times. For that, we introduce packet loss by injecting packet loss for particular flows in edges. This is done through introducing traffic shaping applied directly to switch interfaces that would lead to some packet policing and thus packet loss. Figure 7.9 shows the actual racks that contain equipment used in *MediaFlow* testing including IP Fabric as well as server equipment.

7.6 Experimental Testing Results



(a) Nexus Fabric

(b) Sources, compute nodes and MediaFlow Controller

Figure 7.9 *MediaFlow* system testing racks.

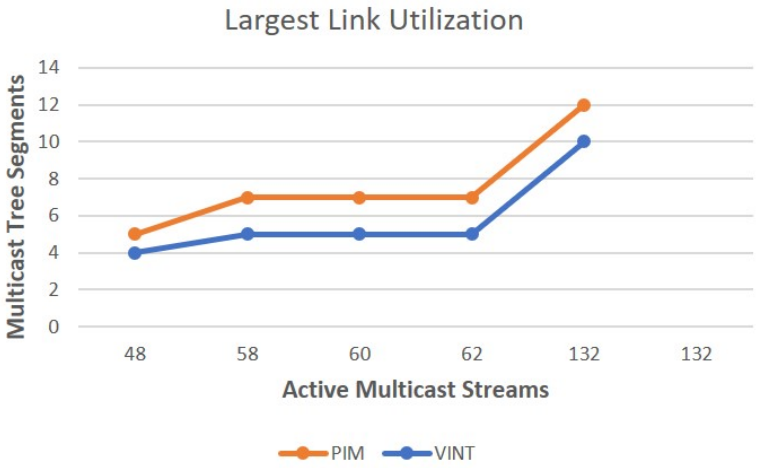
In this section, we compare the performance results of *MediaFlow* with those from state-of-the-art video monitoring solutions which we call *SelectiveReroute* method in the figures. *SelectiveReroute* utilizes reactive selective routing as described

in Chapter 2. To approximate selective routing method, and without loss of generality, we developed a python code that takes a specific multicast flow data from an operator and clones the flow to a monitoring station at the edge of the network. If the video error is confirmed by the station, the flow is rerouted using API calls to the switches.

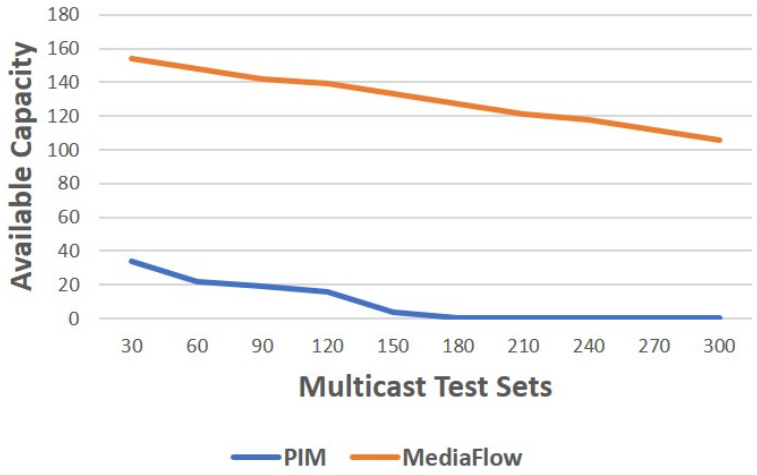
Our testing utilizes three sets of error patterns to induce packet loss errors needed for testing. The first set is based on actual packet traces captured using Multicast video transfer between Amazon Web Services Ohio and North Virginia availability zones to represent traffic that transits unmanaged networks. The other two sets of captures are generated using errors with Gaussian as well as Poison distributions. This is represented in both Figure 7.7 and Figure 7.8.

Figure 7.7 compares error detection time (MTTD) of $MediaFlow$ to selective reroute video monitoring solutions. As discussed in Section 7.1, MTTD component T_r is the reporting time for human eye to detect an issue, T_s is the time it takes to route the impacted flow to an edge device for monitoring and T_d is the time needed for edge device to analyze and detect video flow issues. Using $mediaFlow$, T_r is lowered drastically to 100 uSec and both T_r and T_s will be negligible. As such, MTTD with $MediaFlow$ will be below a second while MTTD without $MediaFlow$ will be in the tens of seconds or more. Testing results confirm the above and show that MTTD for $MediaFlow$ is 100 times lower than that for edge monitoring. This is true for all threetraffic patterns tested.

Figure 7.8 compares flow recovery time (MTTR) of $MediaFlow$ to traditional solutions based on edge monitoring of video quality. The MTTR equation (7.2) extends MTTD calculation in (7.1) by adding two additional parameters: 1) T_l represents the time needed to localize the particular edge where the packets were lost in the fabric. $MediaFlow$ lowers T_l to zero compared to traditional methods which would require tens of seconds. 2) $T_{reroute}$ which represents reroute time for an impacted flow around a problematic link. $MediaFlow$ reduces MTTR by 1000% or

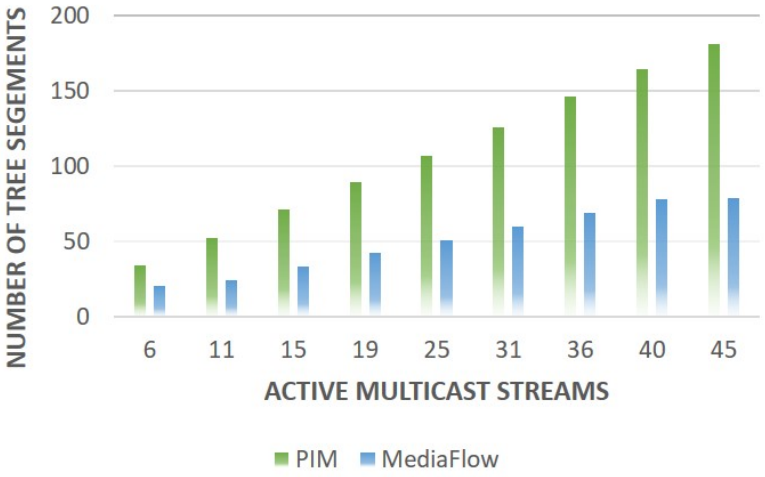


(a) Total source tree segments in the Fabric where higher number of segments for the same number of flows indicate branching closer to the source leading to more fabric bandwidth utilization

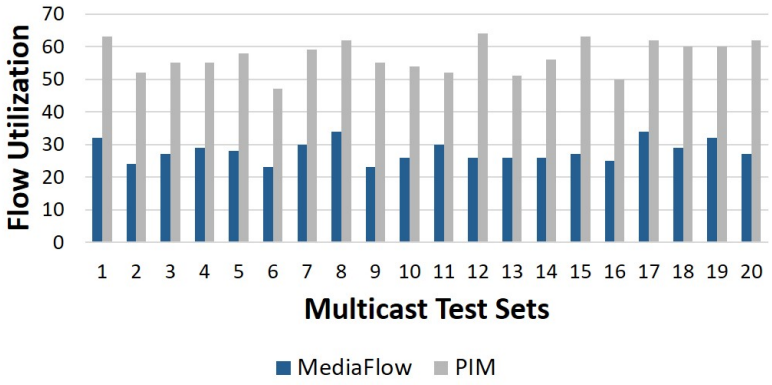


(b) Comparing availability of a link with needed bandwidth for future streams

Figure 7.10 Video flow stream recovery time : Comparing mean time to recover of *MediaFlow* to edge monitoring methods.



(a) Total source tree segments in the Fabric where higher number of segments for the same number of flows indicate branching closer to the source leading to more fabric bandwidth utilization



(b) Standard deviation to show evenness of flow distribution across ECMP links between spines and leafs in clos fabric

Figure 7.11 Video flow stream recovery time : Comparing mean time to recover of *MediaFlow* to edge monitoring methods.

more because in addition to lower MTTR, T_l is significantly lower and very close to zero. Similar to MTTD testing results, testing confirms that MTTR with $MediaFlow$ is orders of magnitude lower than that of edge monitoring for all tested traffic patterns. This observation is explained in equation (7.2).

Key consideration for in-network monitoring advantage is the higher scalability. The high scalability is achieved because video quality is being monitored in-network. A leaf or a spine can monitor up to 32000 flows simultaneously. This is in contrast to 10 to 20 simultaneous flows that can be monitored at the edge using traditional monitoring methods.

Figure 7.12 shows testing results for scale testing where $MediaFlow$ monitoring is able to scale to thousands of flows simultaneously. For actual flow generation we use 100 virtual machines to generate video flows representing actual video flows. Our test results confirm the ability to monitor 1000 simultaneous flows and identify issues in real time. We contrast that with ability of the legacy existing monitoring methods to monitor at the edge for up to 10-20 flows per device which is much lower in scale.

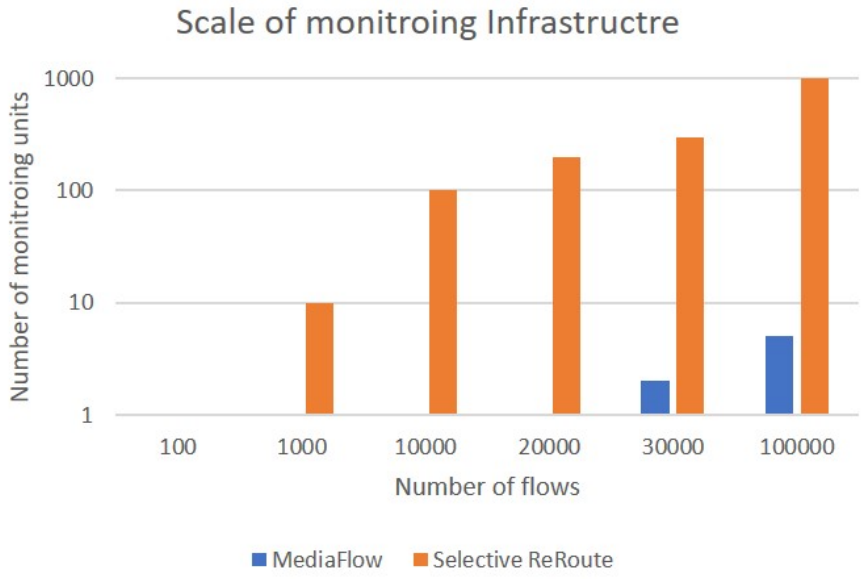


Figure 7.12 Number of edge monitoring devices needed vs. number of switches needed for in-network monitoring based on flow scale.

The test results in Figure 7.10 compare $MediaFlow$ to PIM for scale and fabric capacity. Figure 7.10a shows the total number of tree segments as the number of multicast flows increases for both PIM and $MediaFlow$ based systems. For the same number of flows, total segments in $MediaFlow$ is lower compared to PIM which leads to increased multicast capacity in the fabric. Figure 7.10b compares the number of multicast streams that was forwarded by a leaf before starting to oversubscribe one or more of the uplinks to the spines. PIM shows lower number of flows before the system gets into blocking state.

CHAPTER 8

SUMMARY AND FUTURE DIRECTIONS

Live media production, such as in TV studios or sports production, is being transformed through IP and virtualization adoption. This change is allowing media businesses to adopt workflows for higher video resolutions and better agility in the internet era.

In live video production environments, Multicast is widely used to deliver point to multi-point traffic such as real-time audio and video traffic. Multicast protocols such as $PIM-SM$ suffer from performance limitations as it is not bandwidth aware. In addition, they lack the ability to build multicast trees that can minimize overall traffic inside the network.

In Chapter 4, iRP algorithm is presented which is based on a controller-based multicast routing. iRP algorithm ensures efficient tree formation and load-balancing of multicast flows across links. In addition, iRP algorithm implements multicast admission control based on bandwidth availability in the network. We implement iRP system using fabric built on Cisco Nexus data center switches. Our testing results confirm optimized distribution of flows as well as up to 50% improvement in fabric multicast capacity with efficient tree formation.

Chapter 6 presents $LiRP$ algorithm that increases the efficiency of iRP algorithm by utilizing neural networks to predict multicast group memberships. $LiRP$ utilizes these predictions to further optimize multicast routing in order to increase fabric efficiency. Testing results show that $LiRP$ provides performance that is close to the optimal offline optimization while maintaining online arrival of flows.

In Chapter 5, we present $DiRP$ distributed algorithm to enhance multicast capacity in a fabric. Multicast routing in $DiRP$ is handled in a distributed model across the fabric while maintaining bandwidth awareness when making routing

decisions. $DiRP$ Algorithm ensures efficient tree formation and load-balancing of multicast flows across multi-path links. Also, $DiRP$ implements multicast admission control based on bandwidth availability in the network. We implement $DiRP$ system using fabric built on Cisco Nexus data center switches. Test results confirm that $DiRP$ provides performance that is 60% higher than PIM while maintaining similar path setup delay times.

In Chapter 7, Monitoring and diagnostics of professional media delivery over IP is critical due to very high impact of packet loss on video production. When video flow experiences packet loss, there will be noticeable artifacts in the video image affecting production quality. Video monitoring solutions have focused on monitoring at the edge of the network making MTTD and MTTR very high. In this chapter, we propose $MediaFlow$ algorithm that performs in-network video integrity monitoring at a very high scale. $MediaFlow$ utilizes SDN based controller system to handle multicast routing as well as video quality monitoring. $MediaFlow$ aggregates switch reported video quality and tracks that by introducing flow state metric F_f. The metric is then used, as part of $MediaFlow$ algorithm, to reroute multicast path for a particular video flow around a faulty link to recover impacted video flow. We implement $MediaFlow$ system using fabric built on DC switches and our testing shows reduction in MTTD from minutes to milliseconds and reduction in MTTR from minutes to seconds as compared to state of the art monitoring methods.

8.1 New Realities of Remote Media Production - Global Pandemics Impact on Media Workflows

COVID-19 pandemic has impacted how media companies go about doing their business. On the one hand, consumption of media content has increased by 38% to 41% depending on the demographics [46] with huge focus on online streaming of content. In a new survey by Leichtman Research Group [38], 53% of American adults agreed (selecting 8, 9 or 10 on a 1-10 scale) that they spend more time watching

TV during the pandemic. Just 16% selected 1, 2 or 3 that they disagreed that they were spending more time watching TV. A good example of that is Disney+ streaming offering by The Walt Disney Company which reached 54.5 million subscribers within 6 months of launching [15].

On the other hand, COVID-19 pandemic is already driving a new wave of digitization being introduced to media production and how content is produced. Work-from-home and social-distancing rules are affecting how content production is done with the shift to work from home. Live news production is being done from home using remote production tools. Minimal onsite presence of technical personnel and talent is another challenge that these companies need to handle. Content production workflows is already starting to consume cloud resources as part of the workflow. This is accelerated by the sudden change to workflow imposed by COVID-19 and the need to burst capacity for remote workers to do their function of video editing and content producing.

Current work can be extended to cloud offering as media workflows requirements around end to end capacity management, monitoring and reporting of flow quality. As public clouds start to embrace multicast, our work can be extended to managing multicast streams in a public cloud environments. It can also be extended to establish consistent monitoring and reporting of video streams in the public cloud. The task of monitoring and reporting is especially important as public cloud is a managed service and media customers would require standardized and relevant reporting on their key production workflow.

8.2 Future Direction

With continued focus on digitization of media workflows, some of the future directions for this work are:

1. Multicast optimization in Data Center overlay networks. Overlay networks are utilized to carry VLANs over fully routed fabrics. Multicast optimization in overlay networks is a challenge due to the tree nature of the overlay networks which complicates multicast tree formation.

2. Network optimization techniques for placement of compute nodes to handle video compression algorithms using CPU cycles vs sending uncompressed video and utilize high bandwidth in the network. This is a new topic driven by the advancement of uncompressed video transport protocols such as SMPTE 2110.

3. This work can be expanded to investigate utilizing machine learning for prediction of flow errors based on error patterns. This learning can be utilized to learning how to minimize the impact to video quality and maintain higher service level agreements (SLAs).

4. Cloud production of media workflows is accelerating. This work can be expanded to investigate needed extensions for expansion of multicast into public cloud environments. Areas of focus can be to address the increase in expected error when utilizing a mix of managed and unmanaged networks and how to minimize the impact to video quality.

REFERENCES

[1] A. Adams, J. Nicholas, and W. Siadak. Rfc 3973: Protocol independent multicast-dense mode (pim-dm): Protocol specification (revised). *IETF*, 2005.

[2] N. Ahmed, A. Atiya, N. Gayar, and H. El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621, 2010.

[3] C. Alippi and M. Roveri. Virtual k-fold cross validation: An effective method for accuracy assessment. In *International Joint Conference on Neural Networks*. IEEE, 2010.

[4] M. Alreshoodi, A. Adeyemi-Ejeye, J. Woods, and S. Walker. Fuzzy logic inference system-based hybrid quality prediction model for wireless 4kuhd h. 265-coded video streaming. *IET Networks*, 4(6):296–303, 2015.

[5] D. Ammar and M. Varela. Qoe-aware routing for video streaming over wired networks. In *The 23rd International Symposium on Quality of Service*. IEEE, 2015.

[6] J. Anderson. *An introduction to neural networks*. MIT press, Cambridge, MA, USA, 1995.

[7] B. Arzani, S. Ciraci, L. Chamon, Y. Zhu, H. Liu, J. Padhye, B. Loo, and G. Outhred. 007: Democratically finding the cause of packet drops. In *The 15th Symposium on Networked Systems Design and Implementation*. USENIX, 2018.

[8] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *34th Annual Foundations of Computer Science*. IEEE, 1993.

[9] Awerbuch B and T. Singh. Online algorithms for selective multicast and maximal dense trees. In *Symposium on Theory of Computing*. ACM, 1997.

[10] A. A. Barakabitze, L. Sun, I. Mkwawa, and E. Ifeachor. A novel qoe-centric sdn-based multipath routing approach for multimedia services over 5g networks. In *International Conference on Communications*. IEEE, 2018.

[11] G. Bontempi, M. Birattari, and H. Bersini. Lazy learners at work: the lazy learning toolbox. In *The 7th European Congress on Intelligent Techniques and Soft Computing*. EUFIT, 1999.

[12] G. Bontempi, S. Taieb, Y. Le Borgne, et al. Machine learning strategies for time series forecasting. In *European Business Intelligence Summer School*, 2012.

[13] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, 2014.

[14] B. Boudreau. Outlook for IP Transit Prices in 2018. https://blog.telegeography.com/outlook-for-ip-transit-prices-in-2018, 2018. [Online; accessed 30-Oct-2019].

[15] CA: The Walt Disney Company Burbank. Q2 2020 earnings call transcript. https://thewaltdisneycompany.com/disneys-q2-fy20-earnings-results-webcast/, 2020. [Online; accessed 19-May-2020].

[16] Y. Cai, L. Wei, H. Ou, V. Arya, and S. Jethwani. RFC 6754: Protocol Independent Multicast Equal-Cost Multipath (ECMP) Redirect. *IETF*, 2012.

[17] G. Calvigioni, R. Aparicio-Pardo, L. Sassatelli, J. Leguay, P. Medagliani, and S. Paris. Quality of experience-based routing of video traffic for overlay and isp networks. In *INFOCOM - IEEE Conference on Computer Communications*, 2018.

[18] H. Castro, A. P. Alves, C. Serrão, and B. Caraway. A new paradigm for content producers. *IEEE MultiMedia*, 17(2):90–93, 2010.

[19] S. Chen, O. Gunluk, and B. Yener. The multicast packing problem. *IEEE Transactions on networking*, 8(3):311–318, 2000.

[20] Y. Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics and Epidemiology*, 1(1):161–187, 2017.

[21] S. Chiang, J. Kuo, S. Shen, D. Yang, and W. Chen. Online multicast traffic engineering for software-defined networks. In *Annual International Conference on Computer Communications*. IEEE, 2018.

[22] A. Craig, B. Nandy, I. Lambadaris, and P. Ashwood-Smith. Load balancing for multicast traffic in sdn using real-time link cost modification. In *International Conference on Communications*. IEEE, 2015.

[23] S. Crone, M. Hibon, and K. Nikolopoulos. Advances in forecasting with neural networks: Empirical evidence from the nn3 competition on time series prediction. *International Journal of forecasting*, 27(3):635–660, 2011.

[24] Q. Dai and R. Lehnert. Impact of packet loss on the perceived video quality. In *Second International Conference on Evolving Internet*. IEEE, 2010.

[25] Q. Dai and R. Lehnert. Impact of packet loss on the perceived video quality. In *Second International Conference on Evolving Internet*. IEEE, 2010.

[26] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[27] S. Dhesikan and P. Kathail. Broadcast and network integration using the network control (sdn) api. In *Annual Technical Conference and Exhibition*. SMPTE, 2016.

[28] D. Dhillon. Cisco Cloud Scale ASIC switches. https://blogs.cisco.com/datacenter/cisco-cloud-scale-asic-switches-get-a-2-year-advantage-over-merchant-silicon-switches, 2016. [Online; accessed 01-Nov-2019].

[29] D. Dhillon. Switch & Router Revenues Set a New Record; Cisco Market Share Still Over 50%. https://www.srgresearch.com/inproceedingss/switch-router-revenues-set-new-record-cisco-market-share-still-over-50, 2019. [Online; accessed 02-11-2019].

[30] T Edwards and M. Bany. Elementary flows for live ip production. *SMPTE Motion Imaging Journal*, 125(2):24–29, 2016.

[31] B. Efron. Estimating the error rate of a prediction rule: improvement on cross-validation. *Journal of the American statistical association*, 78(382):316–331, 1983.

[32] A. El Saddik. Digital twins: The convergence of multimedia technologies. *IEEE MultiMedia*, 25(2):87–92, 2018.

[33] M. Ellis, D. Pezaros, T. Kypraios, and C. Perkins. Modelling packet loss in rtp-based streaming video for residential users. In *the 37th Annual Conference on Local Computer Networks*. IEEE, 2012.

[34] L. Elmstedt. *Multicast Routing for Multimedia Applications*. PhD thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 1994.

[35] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, R. Parekh, and Z. Zhang. Rfc 2362: Protocol independent multicast-sparse mode (pim-sm): Protocol specification (revised). *IETF*, 1998.

[36] Nexus 9000 Series NXOS IP Fabric for Media Solution Guide. https://www.cisco.com/c/en/us/td/docs/switches/datacenter/, 2020. [Online; accessed 20-Jan-2020].

[37] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics, New York, NY, USA, 2001.

[38] Leichtman Research Group. Over half of adults report watching more tv since the pandemic. https://www.leichtmanresearch.com/over-half-of-adults-report-watching-more-tv-since-the-pandemic/, 2020. [Online; accessed 26-May-2020].

[39] A gRPC. high performance, open-source universal RPC framework, 2018.

[40] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, et al. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *Conference on Special Interest Group on Data Communication*. ACM, 2015.

[41] Z. Guo, J. Duan, and Y.Yang. On-line multicast scheduling with bounded congestion in fat-tree data center networks. *IEEE Journal on Selected Areas in Communications*, 32(1):102–115, 2014.

[42] J. He and W. Song. Towards smart routing: Exploiting user context for video delivery in mobile networks. In *the 11th International Conference on Mobile Ad Hoc and Sensor Systems*, 2014.

[43] B. Heller, R. Sherwood, and N. McKeown. The controller placement problem. In *First workshop on Hot topics in software defined networks*. ACM, 2012.

[44] M. Huang, W. Liang, Z. Xu, W. Xu, S. Guo, and Y. Xu. Dynamic routing for network throughput maximization in software-defined networks. In *Annual International Conference on Computer Communications*. IEEE, 2016.

[45] T. Ikeguchi and K. Aihara. Prediction of chaotic time series with noise. *Transactions on fundamentals of electronics, communications and computer sciences*, 78(10):1291–1298, 1995.

[46] Global Web Index. Coronavirus research report impact on media consumption. https://www.globalwebindex.com/coronavirus, 2020. [Online; accessed 19-May-2020].

[47] A. Iyer, P. Kumar, and V. Mann. Avalanche: Data center multicast using software defined networking. In *The 6th International Conference on Communication Systems and Networks*. IEEE, 2014.

[48] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte. Real time network policy checking using header space analysis. In *The 10th Symposium on Networked Systems Design and Implementation*. USENIX, 2013.

[49] A. Khandelwaland, R. Agarwal, and I. Stoica. Confluo: Distributed monitoring and diagnosis stack for high-speed networks. In *The 16th Symposium on Networked Systems Design and Implementation*. USENIX, 2019.

[50] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and B. Godfrey. Veriflow: Verifying network-wide invariants in real time. In *The 10th Symposium on Networked Systems Design and Implementation*. USENIX, 2013.

[51] J. Kleinrouweler, B. Meixner, and P. Cesar. Improving video quality in crowded networks using a dane. In *The 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2017.

[52] D. Kline. *Methods for multi-step time series forecasting neural networks*. IGI Global, Hershey, PA, USA, 2004.

[53] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, et al. Onix: A distributed control platform for large-scale production networks. In *The 9th conference on Operating systems design and implementation*. USENIX, 2010.

[54] R. Korf. A new algorithm for optimal bin packing. In *The 18th National Conference on Artificial Intelligence*. The Association for the Advancement of Artificial Intelligence, 2002.

[55] A. Kovalick. Design Elements for Core IP Media Infrastructures. *SMPTE Motion Imaging Journal*, 125(2):16–23, 2016.

[56] D. Kreutz, F. Ramos, P. Verissimo, C. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[57] E. Kurdoglu, Y. Liu, and Y. Wang. Perceptual quality maximization for video calls with packet losses by optimizing fec, frame rate, and quantization. *IEEE Transactions on Multimedia*, 20(7):1876–1887, 2017.

[58] T. Lakshman and U. Madhow. The performance of tcp/ip for networks with high bandwidth-delay products and random loss. *IEEE Transactions on Networking*, 5(3):336–350, 1997.

[59] J. Lapierre and M. Al-Habbal. Bridging the Gap Between Software and SMPTE ST 2110. In *Annual Technical Conference*. SMPTE, 2018.

[60] A. Latif, P. Kathail, S. Vishwarupe, S. Dhesikan, and A. Khreishah. irp: intelligent rendezvous point for multicast control plane. In *38th Sarnoff Symposium*. IEEE, 2017.

[61] A. Latif, P. Kathail, S. Vishwarupe, S. Dhesikan, A. Khreishah, and Y. Jararweh. Multicast optimization for clos fabric in media data centers. *IEEE Transactions on Network and Service Management*, 16(4):1855–1868, 2019.

[62] A. Latif, R. Paul, R.Chhibber, A. Singh, R. Parameswaran, A. Khreishah, and Y. Jararweh. Dirp: Distributed intelligent rendezvous point for multicast control plane. In *The 9th International Green and Sustainable Computing Conference*. IEEE, 2018.

[63] M. Lee, Y. Li, X. Huang, Y. Chen, T. Hou, and C. Hsu. Robust multipath multicast routing algorithms for videos in software-defined networks. In *The 22nd International Symposium of Quality of Service*. IEEE, 2014.

[64] D. Li, M. Xu, Y. Liu, X. Xie, Y. Cui, J. Wang, and G. Chen. Reliable multicast in data center networks. *IEEE Transactions on Computers*, 63(8):2011–2024, 2014.

[65] X. Li, M. Ammar, and S. Paul. Video multicast over the internet. *IEEE network*, 13(2):46–60, 1999.

[66] X. Li and M. Freedman. Scaling ip multicast on datacenter topologies. In *The 9th conference on Emerging networking experiments and technologies*. ACM, 2013.

[67] Y. Li, R. Miao, C. Kim, and M. Yu. Flowradar: A better netflow for data centers. In *The 13th Symposium on Networked Systems Design and Implementation*. USENIX, 2016.

[68] X. Lin and L. Ni. Multicast communication in multicomputer networks. *IEEE transactions on Parallel and Distributed Systems*, 4(10):1105–1117, 1993.

[69] G. Magnaye, J. Poh, M. Aung, and T. Sun. SMPTE2022-5/6 High Bit Rate Media Transport over IP Networks with Forward Error Correction on Kintex-7 FPGAs, 2013.

[70] A. Mansy, B. Ver Steeg, and M. Ammar. Sabre: A client based technique for mitigating the buffer bloat effect of adaptive video flows. In *The 4th Multimedia Systems Conference*. ACM, 2013.

[71] A. Matrawy and I. Lambadaris. A survey of congestion control schemes for multicast video applications. *IEEE Communications Surveys & Tutorials*, 5(2), 2003.

[72] A. Matrawy, L. Lambadaris, and C. Huang. On layered video fairness on ip networks. In *GLOBECOM Global Telecommunications Conference*. IEEE, 2001.

[73] 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 Computer Communication Review*, 38(2):69–74, 2008.

[74] J. Mol, D. Epema, and H. Sips. The orchard algorithm: Building multicast trees for p2p video multicasting without free-riding. *IEEE Transactions on Multimedia*, 9(8):1593–1604, 2007.

[75] M. Moshref, M. Yu, R. Govindan, and A. Vahdat. Trumpet: Timely and precise triggers in data centers. In *SIGCOMM Conference*. ACM, 2016.

[76] J. Moy. Rfc1585 mospf: Analysis and experience. *IETF*, 1994.

[77] J. Moy. Rfc 2328: Ospf version 2. *IETF*, 1998.

[78] H. Nam, K. Kim, J. Kim, and H. Schulzrinne. Towards qoe-aware video streaming using sdn. In *Global Communications Conference*. IEEE, 2014.

[79] Cisco Nexus 9300 Series Switches Data Sheet. https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/datasheet-c78-742284.html, 2020. [Online; accessed 10-Jan-2020].

[80] K. Obraczka. Multicast transport protocols: a survey and taxonomy. *IEEE Communications magazine*, 36(1):94–102, 1998.

[81] Society of Motion Picture and Television Engineers. Smpte overview document - professional media over managed ip networks roadmap for the 2110 document suite. In *SMPTE 2110*, 2019.

[82] Special Report: OpenFlow and SDN State of the Union. https://www.opennetworking.org/images/stories/downloads/sdn-resources/special-reports/Special-Report-OpenFlow-and-SDN-State-of-the-Union-B.pdf, 2016. [Online; accessed 15-May-2020].

[83] A. Palit and D. Popovic. *Computational intelligence in time series forecasting: theory and engineering applications*. Springer, Berlin, Germany, 2006.

[84] P. Phaal, S. Panchen, and N. McKee. RFC 3176: sFlow. *IETF*, 2001.

[85] J. Pyun, J. Jung, J. Shim, S. Ko, and S. Park. Packet loss resilience for mpeg-4 video stream over the internet. In *Digest of Technical Papers. International Conference on Consumer Electronics*. IEEE, 2002.

[86] Q. Qu and P. Arden. RFC 6792: Guidelines for Use of the RTP Monitoring Framework. *IETF*, 2012.

[87] M. Ramalho. Intra-and inter-domain multicast routing protocols: A survey and taxonomy. *IEEE Communications Surveys & Tutorials*, 3(1), 2000.

[88] N. Ranasinghe, R. Bangamuarachchi, J. Seneviratne, A. Jayawardane, A. Pasqual, and R. Senarath. SMPTE ST 2110 Compliant Scalable Architecture on FPGA for end to end Uncompressed Professional Video Transport Over IP Networks. In *The 30th International Conference on Application-specific Systems, Architectures and Processors*. IEEE, 2019.

[89] A. Reibman, V. Vaishampayan, and Y. Sermadevi. Quality monitoring of video over a packet network. *IEEE Transactions on Multimedia*, 6(2):327–334, 2004.

[90] G. Robins and A. Zelikovsky. Improved steiner tree approximation in graphs. In *The 11th annual symposium on Discrete algorithms*. ACM-SIAM, 2000.

[91] G. Rouskas and I. Baldine. Multicast routing with end-to-end delay and delay variation constraints. *IEEE Journal on Selected Areas in communications*, 15(3):346–356, 1997.

[92] P. Savola. Rfc 5110: Overview of the internet multicast routing architecture. *IETF*, 2008.

[93] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, et al. RTP: A transport protocol for real-time applications, 1996.

[94] J. Shin and P. C. Cosman. Classification of mpeg-2 transport stream packet loss visibility. In *International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010.

[95] B. Siregar, M. Manik, R. Rahmat, U. Andayani, and F. Fahmi. Implementation of network monitoring and packets capturing using random early detection (red) method. In *International Conference on Communication, Networks and Satellite*, 2017.

[96] Television Standard SMPTE. 292m-1998," bit-serial digital interface for high-definition television systems," society of motion picture and television engineers.

[97] Standard. Smpte 424m-2006: 3 gb/s signal/data serial interface. *Society of Motion Picture and Television Engineers*, 2006.

[98] Standard. 1.5 gb/s signal/data serial interface. *The Society of Motion Picture and Television Engineers*, 2011.

[99] Standard. Transport of high bit rate media signals over ip networks. *The Society of Motion Picture and Television Engineers*, 2012.

[100] Standard. St 2110-21:2017 - smpte standard - professional media over managed ip networks: Traffic shaping and delivery timing for video. *Society of Motion Picture and Television Engineers*, 2017.

[101] SMPTE Standard. For television — serial digital fiber transmission system for smpte 259m, smpte 344m, smpte 292 and smpte 424m signals. In *ST 297:2006*. SMPTE, 2006.

[102] M. Starzak, W. Zabierowski, and A. Napieralski. System for transmitting hdtv over ip networks using open-source software. In *The 11th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics*, 2011.

[103] Newscast Studio. With 4k and ip, nbc 10 and telemundo 62 philadelphia debut upgrade. https://www.newscaststudio.com/2018/10/25/nbc-10-telemundio-62-philadelphia-facility/, 2018. [Online; accessed 14-February-2020].

[104] P. Sun, M. Yu, M. Freedman, J. Rexford, and D. Walker. Hone: Joint host-network traffic management in software-defined networks. *Journal of Network and Systems Management*, 23(2):374–399, 2015.

[105] Cisco Systems. Ip multicast technology overview. https://www.cisco.com/c/en/us/td/docs/, 2001. [Online; accessed 28-February-2017].

[106] Cisco Systems. Nexus 9200 platform switches data sheet. http://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/datasheet-c78-735989.html, 2016. [Online; accessed 14-February-2017].

[107] Cisco Systems. Global mobile data traffic forecast update, 2017–2022. *Visual Networking Index*, 2019.

[108] H. Takahashi et al. An approximate solution for the steiner problem in graphs. *Math. Japonic*, 24(6):573–577, 1980.

[109] S. Tang and P. Alface. Impact of random and burst packet losses on h. 264 scalable video coding. *IEEE Transactions on Multimedia*, 16(8):2256–2269, 2014.

[110] N. Teslic, V. Zlokolica, V. Pekovic, T. Teckan, and M. Temerinac. Packet-loss error detection system for dtv and set-top box functional testing. *IEEE Transactions on Consumer Electronics*, 56(3):1311–1319, 2010.

[111] Second-generation of world's fastest p4-programmable ethernet switch asics. https://www.barefootnetworks.com/products/brief-tofino-2/, 2019. [Online; accessed 10-Dec-2019].

[112] In Praise of Uncompressed Video. https://www.tvtechnology.com/opinions/in-praise-of-uncompressed-video, 2018. [Online; accessed 10-Dec-2019].

[113] K. Vachaspathi, G. Polyzos, and J. Pasquale. Multicast routing for multimedia communication. *IEEE TRANSACTIONS ON NETWORKING*, 1(3):286–292, 1993.

[114] M. Vega, C. Perra, and A. Liotta. Resilience of video streaming services to network impairments. *IEEE Transactions on Broadcasting*, 64(2):220–234, 2018.

[115] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang. Machine learning for networking: Workflow, advances and opportunities. *IEEE Network*, 32(2):92–99, 2018.

[116] W. Wei. Time series analysis. In *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2*, 2006.

[117] W. Xia, P. Zhao, Y. Wen, and H. Xie. A survey on data center networking (dcn): Infrastructure and operations. *IEEE communications surveys & tutorials*, 19(1):640–656, 2016.

[118] J. Xie, R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu. A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges. *IEEE Communications Surveys and Tutorials*, 21(1):393–430, 2018.

[119] D. Yang and W. Liao. Optimal state allocation for multicast communications with explicit multicast forwarding. *IEEE Transactions on Parallel and Distributed Systems*, 19(4):476–488, 2008.

[120] G. Yao, J. Bi, Y. Li, and L. Guo. On the capacitated controller placement problem in software defined networks. *IEEE Communications Letters*, 18(8):1339–1342, 2014.

[121] M. Yu, A. Greenberg, D. Maltz, J. Rexford, L. Yuan, S. Kandula, and C. Kim. Profiling network performance for multi-tier data center applications. In *The 8th Symposium on Networked Systems Design and Implementation*. USENIX, 2011.

[122] G. Zhang, E. Patuwo, and M. Hu. Forecasting with artificial neural networks: The state of the art. *International journal of forecasting, Elsevier*, 14(1):35–62, 1998.

[123] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, and B. Zhaoand others. Packet-level telemetry in large datacenter networks. In *Conference on Special Interest Group on Data Communication*. ACM, 2015.