Provider-Side VoD Content Delivery Using OpenFlow Multicast

Provider-Side VoD Content Delivery Using OpenFlow Multicast

by

Bernard Niyonteze

Department of Computer Science and Engineering

Pohang University of Science and Technology

A thesis submitted to the faculty of the Pohang University of Science and Technology in partial fulfilment of the requirements for the degree of Master of Science in the Computer Science and Engineering

Pohang Korea

June 30, 2014

Approved by

Prof. James Won-Ki Hong

Academic advisor

Provider-Side VoD Content Delivery Using OpenFlow Multicast

Bernard Niyonteze

The undersigned have examined this thesis and hereby certify that it is worthy of acceptance for a master's degree from POSTECH

June 30, 2014

Committee Chair James Won-Ki Hong (Seal)

Member Jae-Hyoung Yoo (Seal)

Member HanJun Kim (Seal)

MCSE Bernard Niyonteze, “Provider-Side VoD Content Delivery Using OpenFlow Multicast”

20121007 Department of Computer Science and Engineering, 2014, 54P,

Advisor James Won-Ki Hong, Text in English

ABSTRACT

Internet Protocol Television (IPTV) is a rapidly growing service for the delivery of broadcast TV and other media-rich services over a secure, end-to-end operator managed broadband IP data network. With time IPTV services are growing, new services and high demand of contents in terms of high quality video, timely delivery and time-shifted television places a burden on the IPTV operator. IP based multicast is considered as the most efficient way to deliver data to multiple receivers at the same time with preserving high throughput, and hence, most of the IPTV system exploits IP based multicast protocol to deliver content especially between root node and local nodes (cache of root node). With this protocol, the data transmission paths are determined by multicast tree which is constructed and maintained by each network node in distributed manner. Since each network node has no holistic network view and unaware of the entire network status, therefore, even if some local nodes encounter a drastic packet loss (network status change) due to the constraint of available bandwidth, the multipath tree will still be maintained, and this in turn causes network resource wastage.

OpenFlow is one of the enabling technologies for flexibly managing and controlling the network in a centralized and programmable manner by considering the current network status. Therefore, in this thesis we take advantage of OpenFlow and propose a resource efficient provider-side VoD content delivery method using OpenFlow multicast feature in IPTV network. The proposed OpenFlow based multicast scheme dynamically re-constructs multicast tree with considering the current network status, and conserve the network resource by pruning the multicast tree branches under which the local nodes drop the received packets.

We exploited the group table feature provided by OpenFlow to implement multicast scheme, and implemented a resource efficient VoD content delivery solution on top of OpenFlow controller. The solution contains network status monitoring, multicast tree construction and management functions. To validate the proposed VoD content delivery solution, we constructed a virtual network topology using Mininet network emulator, and to reflect the real-world network structure, we designed the network topology by referring to KT premium network which provides IPTV service. We adopted Ryu as OpenFlow controller, and deployed the proposed solution to it. The experiment result shown that our multicast scheme can conserve network resource, and the multicast tree building procedure only requires around 4 ms.

Contents

List of Figures

List of Tables

I. Introduction

This chapter provides a brief introduction to Internet Protocol Television (IPTV), Content Delivery Network (CDN), Software Defined Network (SDN) and OpenFlow protocol. It also presents the motivation and problem statement. Then it presents research goals and proposed solution.

1.1 Background

According to International Communications Union-Telecommunication (ITU-T) definition, IPTV[1] is defined as multimedia services such as television/video/audio/text/graphical/data delivered over IP based networks managed to provide the required level of Quality of Service (QoS) and Quality of Experience (QoE), security, interactivity and reliability." Television (TV) channels and Video on Demand (VoD) are delivered to the television sets through a broadband connection, instead of being delivered using the conventional cable or broadcasts formats. The video streams are encoded into a series of internet protocol packets and then carried through the public internet means which can be received by anyone who have subscription for the service.

IPTV is generally provided together with the Voice over Internet Protocols (VoIP) and internet access, this service is known as Triple Play service. It is a complete package that allows customers to watch TV, to browse the internet and to make a call using the VoIP. The IPTV service is typically provided by a service provider using a closed network infrastructure. IPTV is not the internet video that simply allows users to watch videos, like movie previews and web-cams, over the internet in a best effort fashion.

IPTV technology offers revenue-generating opportunities for the telecom and cable service providers. From the service provider's perspective, IPTV encompasses the acquisition, processing, and secure delivery of video content over an IP based Network infrastructure.

A typical IPTV network architecture (Figure 1) is composed of Super Headend (SHE) also called root node, core network, Regional Head end (RHE) also called local node, access network, Video Service Office (VSO) and home network. SHE is where common or national channels are acquired. Typically in IPTV system there are two SHEs for resiliency. The functions of SHE include:

- The encoders that encode the received video signal to Moving Picture Experts Group(MPEG)
- Video multiplexers, which are responsible for multiplexing/ demultiplexing the encoded video streams into the format required for the common channel line-up.

The RHE is where local channels are acquired and then added to the common channel line-up to create the channel line-up for a given region. The components within the RHE are typically a subset of those within the SHE. If the national channel line-up needs to be customized for each region, video-multiplexing equipment at the RHE is used. VSO is where the aggregation nodes (and sometimes access nodes) are typically located. The VSO is the closest provider-owned facility to the subscriber. Core network is usually an IP/MPLS network transporting traffic to the access network, access network is used to distribute the IPTV streams to the home network, and finally home network is the communication between digital devices deployed in home and is the point where IPTV stream is terminated and viewed.

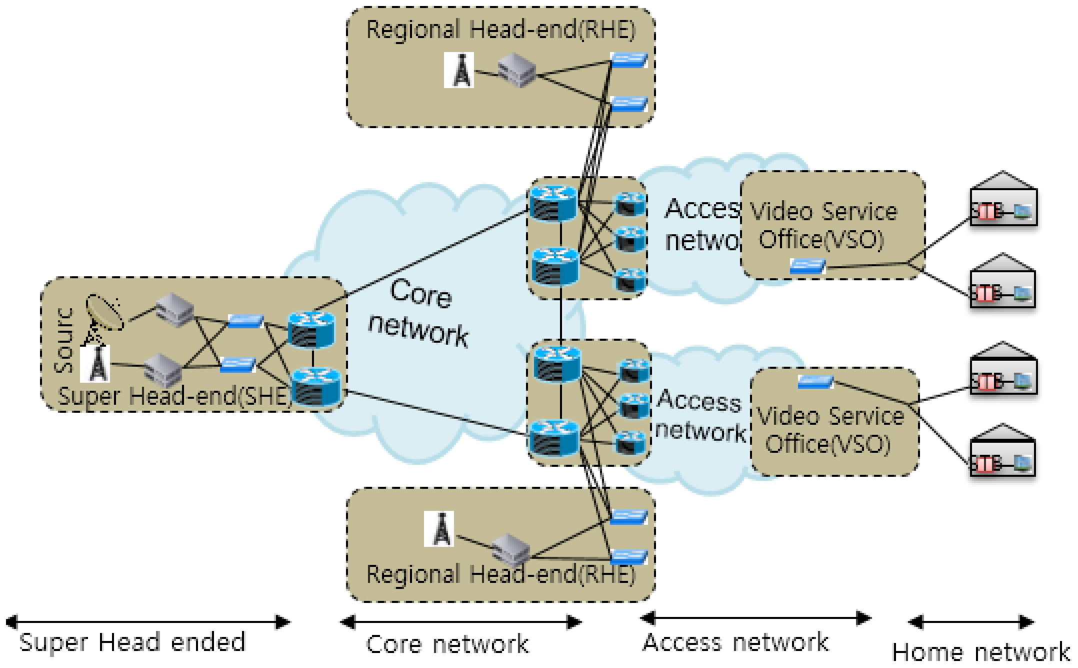

Figure 1.IPTV general Architecture

VoD is a system which allows users to select and watch video content when they choose rather than having to watch at a specific broadcast time [28]. The VoD service is becoming a dominant service in the telecommunication market because it offers great convenience regarding the choice of content items and independent viewing time.

ITPV supports both live TV and the stored videos called VoD service. To receive the IPTV signals a television sets or computer must equipped with a set-top box. Set-top box is an information appliance device that generally contains a TV-tuner input and displays output to a television sets or other display device. Moreover, video contents must be compressed before being delivered through the network. Typically video compression techniques used include MPEG2, MPG4 and H.264 codec.

IPTV uses various communication protocols including Hyper Text Transfer Protocol (HTTP), Internet Group Management Protocol (IGMP) and Real Time Streaming Protocol (RTSP). IPTV operator needs a delivery system to distribute contents to different subscribers in different locations with high quality video and low latency. On the other hand IPTV subscribers are expecting to get high quality of service and low latency as well low cost service from their providers, however to guarantee a high video quality and low latency requires high bandwidth and an efficient way to deliver video contents from source to consumers. To cope with these problems IPTV leverage CDN infrastructures to satisfy requirements.

CDN is a system of servers deployed in different data centers in different geographic locations (Figure 2). The primary goal of CDN is to serve or deliver content to users in different geographic locations with high availability and high speed. CDN rapidly serves web content to multiple users by duplicating the content and directing it to users based on their geographic proximity. CDN decides the data center locations from which the web content will be served to end-user based on numerous factors including content proximity, speed, latency and availability. CDN is used to serve a large part of web content including web objects (text, graphics), downloadable files (media, software), and web applications (e-commerce) and on demand media. In addition, CDN is also heavily used in social network (Facebook) to speed up serving content to the users in different geographic locations. CDN is based on a system of servers with massive storage that places copies from the content library closer to the end user so as to maximize the available bandwidth, and consequently reduces the data access times [24, 25].Topology of CDN follows central-edge architecture, thus it can push the popular video content to the edge nodes, while utilizing the global load balancing and application redirection technologies to ensure that the end-user can watch the requested program without delay.

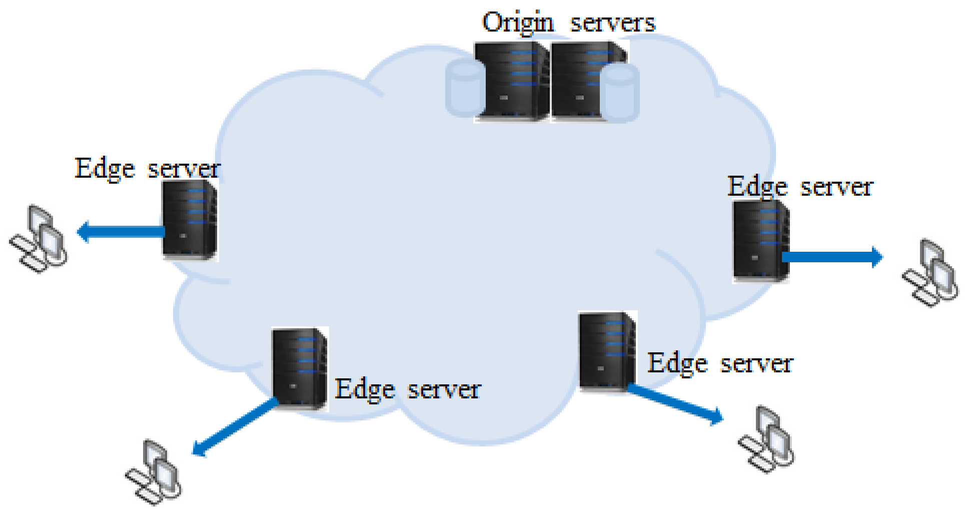

Figure 2. Content delivery Network Architecture

In general, CDNs deploy or outsource infrastructure across the Internet to allow origin content provider's servers to reach end customers. Service provider's infrastructures might differ in many different aspects, (e.g. the geographical coverage, dedicated content type or Internet service), according to Service Level Agreements, or deployment policies. One of the main infrastructure components is the delivery server, also known as edge or local server upon which customers connect to retrieve content. Moreover CDN allows the optimization of the network use through a distribution of the content delivery servers in the physical network, and the optimization of the storage resources through a popularity-based distribution of the content on the servers.

However, IPTV still faces several problems, one problem is that the multicast tree cannot dynamically adjust according to the current state of the network. Therefore, delivery of video content over IP requires a flexible architecture that changes dynamically according to the current network state to optimize utilization of network resources. To meet these requirements, OpenFlow [17, 33] is a competitive candidate that offers flexibility to control and direct the flow through its programmability and dynamicity characteristics. Moreover, OpenFlow Controller manages the topology of the entire network through Link layer Discovery Protocol (LLDP) and builds different multicast forwarding tree depending on the network conditions.

SDN is an emerging paradigm in networking research and industry areas to deal with the appearing demands for flexible and agile network control [2, 29, 30]. DN is defined as an approach to networking in which control plane is decoupled from data plane and tasks related to control plane are offloaded to separate piece of software called controller. SDN is claimed to have a more flexibility than the legacy networks and provides abilities to speed up innovation [2].

SDN brings several advantages compared to traditional network. SDN, as programmable approach, promises to bring new level of flexibility, management and control to networking. By decoupling network logic, policies from underlying switching hardware, SDN brings new flexibility into the networking environment in which logic, policies can be defined, changed and modified on regular basis, all from a central location, and propagated throughout the network. In addition, network logic at the controller level is truly programmable, meaning that the new services can be defined, innovations can be introduced and the network can be customized to satisfy the need of the organization.

OpenFlow [17, 33] is the first standard communication interface defined between the controller and forwarding layers of SDN architecture [3, 27]. OpenFlow allows direct access to and manipulation of the forwarding plane of the network devices such as switches and routers, both physical and virtual. OpenFlow is considered as the most successful approach for realization of SDN.

1.2 Motivation and Problem Statement

IPTV is growing service for the delivery of broadcast TV and other VoD services over managed broadband IP data networks. As on-demand content libraries expand and advertisements insertion increases it brings more challenges to service provider how to manage a huge amount of content as well as to optimize network resource utilization. Therefore network operators need an efficient way to manage and use resources. IPTV system uses IP multicast to transmit content to multiple receivers. This benefits from the bandwidth efficient of multicast technology.

Although the IP multicast has been put into use for a long time, there are still some problems in IP multicast forwarding. Since the most of IP multicast applications are based on UDP protocol, which uses best effort delivery and lacks the congestion avoidance windowing mechanism of TCP, multicast packets may get dropped in the intermediate switch or at the edge node with a cross traffic. However, in existing IP based multicast scheme such packet dropping event cannot trigger the multicast tree re-construction, and as a consequence all intermediate switches between affected node and sender will continuously send packets which would finally get dropped at the end. Therefore, to conserve the network resource, it would be better to prune the branch of multicast tree under the congested link by reconstructing multicast tree.

1.3 Research Goals and Approach

Considering the problem of the IP multicast for content delivery system, in this thesis we propose a resource efficient VoD content delivery solution using OpenFlow multicast. In this section, we enumerate our research goals to solve the current IP based multicast issues and explain briefly the proposed approach to achieve our goals. The research goals of our proposed solution are the following:

- Propose an provider-side VoD content delivery service architecture using OpenFlow multicast
- Reduce the network resource wastage in VoD content delivery by dynamically reconstructing the multicast tree through monitoring the network status
- Implement the proposed solution as an OpenFlow application
- Validate our proposed solution by deploying the application into emulated network

The proposed approach is based on the exploitation of OpenFlow features where we have used group table feature appeared in OpenFlow specification version 1.1[18] to implement multicast.

1.4 Thesis Outline

The organization of this thesis is as follows: Chapter II describes several related work about Content Delivery Networks, IP multicast and Software Defined Networking and OpenFlow Technologies. Chapter III presents IPTV Network Architecture and details description of its components, case study of KT IPTV network architecture. Thereafter, Chapter IV describes the method and implementation details of the proposed solution for content delivery in IPTV networks. Chapter V validates and evaluates the performance of the proposed method. Finally, Chapter VI presents thesis conclusion together with the summary, contributions of the thesis and future work.

II. Related Work

In this chapter, we discuss several research work done on content delivery network technologies and introduce Software Defined Network technologies and OpenFlow protocol. Then, we discuss the advantages of SDN and OpenFlow technologies and how IPTV content delivery can benefits these technologies.

2.1 Content Delivery Network

CDN refers to large network of servers deployed across multiple networks in several data centers, often geographically distributed. CDNs improve the network performance and offer fast and reliable applications and service by distributing contents to cache servers located close to users [24, 25]. Most of the current state-of-the-art multimedia streaming applications (e.g., live and on demand streaming) over the Internet rely heavily on CDN.

A. Azgin et Al. [5] proposed the Cooperative Weighted Fair Queuing technique which fairly and efficiently allocates the peers' resources to ongoing session. They used cooperative transmission strategies to support timely and efficiently delivery of on-demand content to end users. Cooperative transmission strategies suggest that users who have access to the requested content cooperatively transmit to the targeted users to minimize the servicing requirement at the server side. The main objective is to maximize the servicing network capacity by reducing overhead at the server, and to improve the operational lifetime of the network by distributing resources to session peers as fairly as possible.

M. Othman Othman and K. Okamura [6] proposed content Anycasting, which aims to improve the scalability of the network and reduce the congestion overhead at the server side by enabling the client to serve other clients. In this approach, when the server reaches a certain threshold it redirects the request to the clients. The redirection is based on the destination addresses and content identification.

Agrawal et.al [7] present an IPTV distribution model. Their objective is to develop a general framework for planning an IPTV service deployment. They look at ways to maximize the numbers of subscribers that can be added to a given infrastructure in different scenarios. However, all above research works, authors mainly focused on scalability of the system to increase the number of customers who can access the content from the edge server, but not content delivery time in core networks.

P. McDonagh et al. [14] proposed how OpenFlow technology can be used in conjunction with Video Flow management to manage IPTV delivery network. Video flow management helps to indicate the service quality degradation and OpenFlow assists to locate the source of the problem. OpenFlow leverage the measurement of video inspection and combine with OpenFlow statistics to make better decision in routing to assure the quality of service.

D. Chang et al. [12] proposed an SDN-based content delivery framework, which support name-based routing and caching. They describe how the proposed method support flow management with contents names where the contents are mapped to IP addresses by a controller and the flow matching is performed on a switch with the mapped IP address. They also explain how the SDN enhance the efficiency of content delivery. By using SDN structures and properties the route of content delivery is dynamically selected or modified based on the network information and also show how cached contents are managed in centralized or decentralized manner.

Several works and research have been done on different server scheduling techniques and proxy caching strategies and combinations of the two for video on demand systems and content distribution networks [8, 9, 10, 11]. These works are similar to our work by dealing with maximizing bandwidth requirements for media content distribution and investigating the trade-off between network bandwidth and cache storage resources. But our work is different in that it considers provider-side VoD content delivery based on the current network state.

2.2 IP Multicast

Multicast is the delivery of information to multiple destinations simultaneously using the most efficient strategy to deliver the message over each link of the network only once and only creates copies when the link to the destinations split [32]. Multicast over a network allows sender to distribute data to all interested parties while minimizing the use of network resources. Multicast is proved to be a useful and cost effective alternative to traditional methods of sending large multimedia files across network.

Y. Yu et Al [16] proposed a multicast mechanism based on OpenFlow. They separate data and control plane by shifting the multicast management to remote centralized controller. The demonstration shows that the proposed solution compared with traditional multicast yield a good performance, remove burden of multicast routers and increases the router's packet forward speed. This work is similar to our work in a way that it studied multicast using OpenFlow. However, our work is fundamentally different in that it considers VoD content delivery in between super head enf and local nodes whereas their work study the feasibility of multicast in OpenFlow in general.

N. Khaing et Al. [15] proposed the extension of IP multicast service model to support multicast addressing and filtering for large-scale multicast applications. Data are filtered by middleware before passing it to the application, and in addressing, data are routed only to those have expressed their interest. However, in this work the main objective is scalability.

2.3 Software Defined Networking and OpenFlow

Software Defined Networking is an emerging paradigm in the networking research and industry areas to cope with the recently appearing demands for flexible and agile network control [17, 33]. SDN is a new approach to designing, building and managing network. The basic concept is that SDN separates the network's control plane and forwarding plane to make it easier to optimize each plane [2, 26]. In this environment, a controller acts as the brains, providing abstract, centralized view of the overall network. Through the controller, network administrator can quickly and easily make and push out decisions on how the underlying systems (switches, routers) of the forwarding plane will handle the traffic. The most common protocol used in SDN networks to facilitate the communication between the controller and the switches is currently OpenFlow.

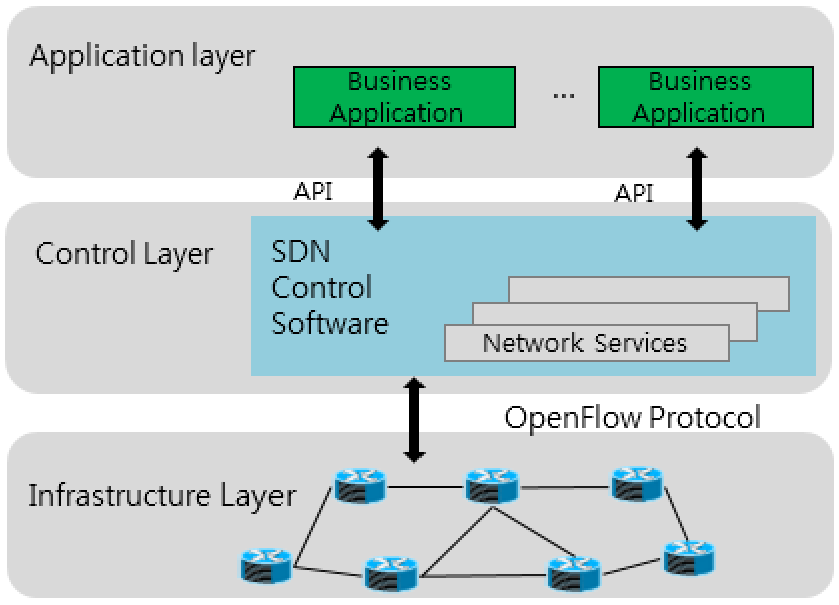

Figure 3. Software network architecture

SDN architecture consists of three layers (see Figure 3), from the bottom infrastructure layer, also called the data plane which comprises the forwarding network elements. In the middle, we find the control layer, also called control plane. It is responsible for programming and managing the forwarding plane. The control plane uses information provided by the forwarding plane and defines network operation and routing. It comprises one or more software controllers that communicate with the forwarding network elements through standardized interfaces, which are referred to as southbound interfaces. The application layer contains network applications that can introduce new network features, such as security, forwarding schemes or assist the control layer in the network configuration. Network intelligence is logically centralized in DN controllers, which maintain a global view of the network. As a result, the network appears to the application and policy engines as a single, logical switch.

On the other hand, with SDN, enterprises and carriers gain vendor-independent control over the entire network from a single logical point, which greatly simplifies the network design and operation. SDN also simplifies the network devices themselves, since they no longer need to understand and process thousands of protocols standard but merely accept instructions from the SDN controllers. Moreover, network operators and administrators can programmatically configure this simplified network abstraction rather than having to write tens of thousands line of code to configure and to scatter among thousands of devices.

OpenFlow [17, 33] is considered as one of the standards communication interface between the control plane and forwarding plane of SDN architecture. Currently, OpenFlow protocol is considered as a key enabler for SDN. OpenFlow allows direct access to and manipulation of the forwarding plane of network devices such as switches and routers, both physical and virtual.

OpenFlow uses the concept of flows to identify network traffic based on pre-defined match rules that can be statically or dynamically programmed by the SDN control software. It also allows IT to define how traffic should flow through network devices based on parameters such as usage patterns, applications, and cloud resources. Because OpenFlow allows the network to be programmed on a per-flow basis, OpenFlow-based SDN architecture provides extremely granular control, enabling the network to respond to real-time changes at the application, user, and session levels.

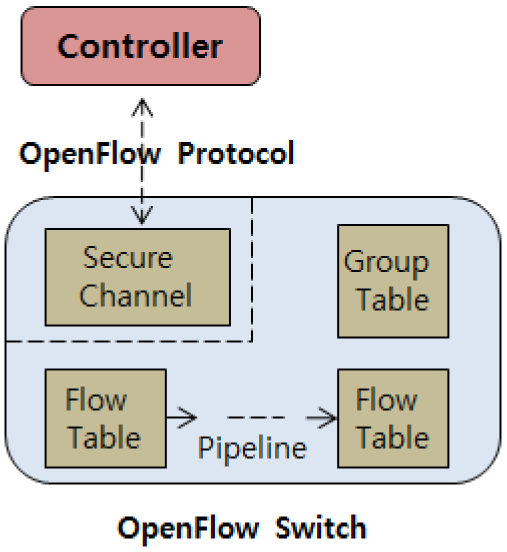

Figure 4. Component of OpenFlow Switch

OpenFlow controller is an application that manages flow control in SDN environment. OpenFlow controller serves as a sort of operating system (OS) for the network. All communication between applications and devices have to go through the controller.

An OpenFlow switch consists of at least three parts as shown in Figure 4. [17,33]:

(1) One or more Flow tables and group table, which perform packet lookups, forwarding and tell the switch how to process the flow

(2) A secure channel that connects switch to the controller, allowing commands and packets to be send between controller and the switch

(3) An OpenFlow protocol, which provides an open and standard way for a controller to communicate with a switch.

Using the OpenFlow Protocol, a controller can add, update, and delete flow entries in flow tables both reactively and proactively. By specifying a standard interface (the OpenFlow Protocol) through which entries in the Flow table can be defined externally, the OpenFlow switch avoids the need for operators to program the switch. Implementation is based on the Application Programming Interfaces (API), which allow modification of the Flow tables that represent the forwarding decisions of a switch.

All packets processed by the forwarder are compared against flow entries in the Flow table. For each packet matched the flow entry received by the forwarder, the appropriate actions (forward, drop, etc.) will be taken. Flow entries may forward packets to one or more OpenFlow ports. If no match entry is found, the switch sends the packet to the controller and controller determines how to handle the packets and add additional flow entries to switch's flow table.

With the emerging of SDN technology, many OpenFlow controllers are coming out (e.g., Beacon [19], Trema [20], Floodlight [21], Pox [22] and Ryu [23]). Ryu is a component-based software defined network framework with well-defined API that makes it easy for developers to create new network management and control application. We chose to use Ryu controller because it supports various protocols for managing network devices, such as OpenFlow, Netconf,[35], OF-Config[31], in addition Ryu supports fully OpenFlow version 1.0, 1.2, 1.3, 1.4.

From OpenFlow specification 1.1 version and later, OpenFlow introduces group table as a new feature that supports more complex forwarding behaviors which are possibly applied to a set of flows. Group table contains group entries (Table 1) and each entry contains a list of actions buckets. The action in one or more buckets is applied to packet sent to the group. A group table entry may be performed if a flow table entry uses an appropriate instruction that refers to its group identifier. In particular, multiple flow table entries can point to the same group identifier, so that the group table entry is performed for multiple flows. Group table entry contains a group type, a counter field and field for actions buckets. The Counters are used for collecting statistics about packets that are processed by this group. A single action bucket contains a set of actions that may be executed, depending on the group type. There are possibly multiple action buckets for a group table entry. The group types defines which of them are applied. There are four group types in group table including all, select, indirect and fast failover but we only explain the group type all that is our interest in this work. Group type "all" is used to implement broadcast and multicast. Packets of this group are processed by all action buckets. The actions of each bucket are applied to the packet consecutively.

Table 1.Group table entries for OpenFlow 1.1 and later

Group Identifier	Group Type	Counters	Actions Buckets

OpenFlow-based SDN technologies enable IT to address the high-bandwidth, dynamic nature of today's applications and adapts the network to ever changing business needs, and significantly reduce operation and management complexity. Benefits to enterprises include the following:

- **Reduced complexity through automation**: OpenFlow-based SDN offers a flexible network automation and management control framework, which makes it possible to develop tools that automate many management tasks. As a result, automation reduces operational overhead and decrease significantly error introduced by network operator.

- **Centralized control of multi-vendor environment**: SDN can control many OpenFlow-enabled network device (switch, router) from any vendor. Instead of managing groups of devices from individual vendors, network Operator can use SDN-based orchestration and management tools to quickly deploy, configure and update devices across the entire network.
- **Better user experience**: Through centralized control management and availability of network state information at application level, SDN infrastructure can adapt to dynamic user needs. A typical example, for video service user select a resolution setting, which the network may or may not be able to support, resulting in delays and interruptions that degrade user experience and annoys users. However, with OpenFlow-based SDN, video application would be able to detect available bandwidth in network in real time and automatically adjust the video resolution accordingly.

More specific benefit of OpenFlow-based in IPTV service is the following:

OpenFlow in IPTV offers comprehensive and flexible way to manage the IPTV multicast group. In traditional multicast each router takes part in routing decision, routers have to exchange information and update their routing tables always when new multicast member joins or leaves[4]. This process takes quite a longtime to exchange and update routing table for each router. As result, it introduces the latency to build multicast tree. Therefore, it is useful to adopt a mechanism that has a full knowledge of the network topology. OpenFlow controller is a competitive candidate which uses centralized approach with entire network view. When a multicast group member want joins, immediately OpenFlow controller inserts flow entries into concerned OpenFlow switches and send content to the dedicated nodes. On the other hand, when a multicast group member leaves multicast group OpenFlow controller deletes flow entries from the switches. This provides the flexibility in management of multicast tree and reduces significantly the complexity of exchanging message between routers.

III. IPTV Content Delivery Network Architecture

In this chapter, we illustrate the components of IPTV network architecture, IPTV service and operation and we will also discuss briefly IPTV transport protocols.

3.1 Case Study: KT IPTV

This part presents core components that consist IPTV system architecture. First, we briefly introduce KT, then we illustrate functional blocks of KT IPTV network architecture.

KT stands for Korean Telecom, KT is a South Korean integrated wired/wireless telecommunication service provider. KT focus on information & communication business, and it has the largest portion of the South Korean telephone and high-speed Internet market.

Originally founded in 1981 as public corporation, KT actively led Korea's transition to the information era and played a key role in promoting the growth of Korea into globally recognized IT superpower [1]. In 2009, KT merged with its subsidiary Korea Telecom Freetel (KTF), paving the way to the convergence of fixed and mobile network. Since then it constantly seeks new business area, such as media, virtual goods etc.

KT IPTV network architecture presents the following functional blocks including Root node, Core Network, Local Nodes, Access Network and Home Network (Figure 5).

[1]Source: http://en.wikipedia.org/wiki/KT_Corporation

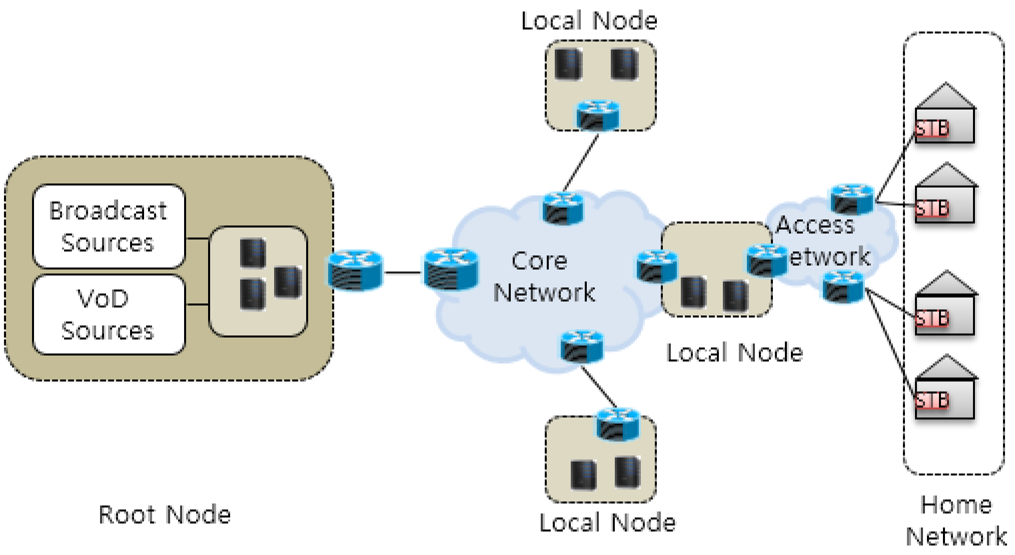

Figure 5. KT IPTV system Architecture

Root node captures broadcast information coming from an antenna or a satellite dish, it is the source that contains contents before being distributed to local nodes. The core Network is the central network portion of a communication system. The core network primarily provides interconnection and transfer between content sources (such as root node) and the edge access network. Local nodes are small data centers distributed in different regions close to customers. Usually popular content are stored in local nodes in order to reduce the congestion at root node and to reduce the network latency. The Access network allows individual subscribers or devices to connect to the core network. IPTV access networks can be DSL, cable, wireless or optical lines. Home network consists of set-top box (STB) which is an IPTV Consumer Device (IPTV CD) that allows users to access IPTV services. STB is also the end point in the home network where the television set is connected.

3.2 KT IPTV Service and Operation

KT IPTV offers mainly two services including TV Channel and VoD services. VoD contents are divided into two categories: Hot VoD content and Cold VoD content. Hot VoD contents are popular contents which are stored at local nodes close to customers in order to avoid network congestion at root node whereas cold VoD contents are stored in root node. All customer requests are redirected to the root node to initiate a content delivery session (Figure 6). Root node analyses a client’s location, media availability and CDN server’s load and then redirects the request to the appropriate local node (Figure 6).

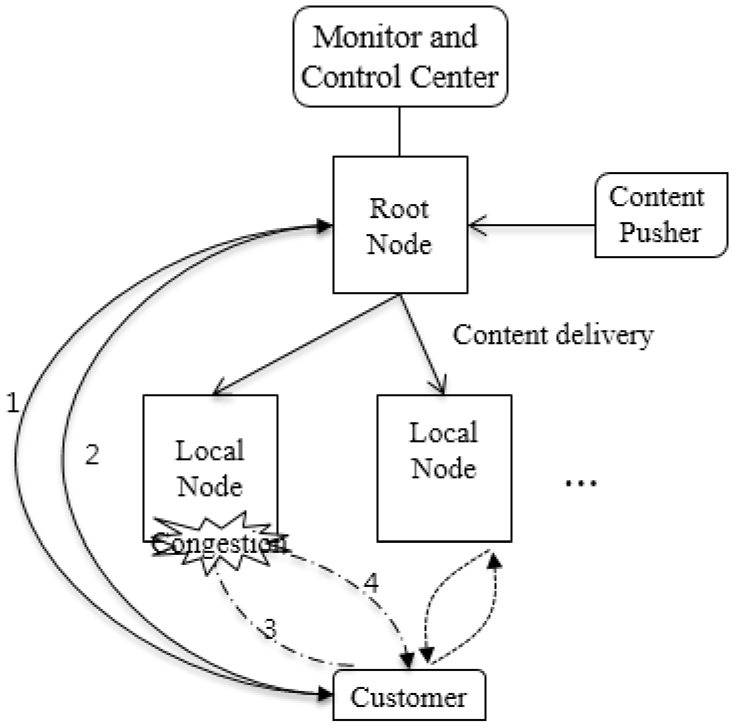

Figure 6. KT IPTV Content Request Operation

1. Customer sends request to root node
2. Root node returns url that indicates a local node containing the requested content
3. Customer sends his request to the indicated local node
4. Local node reply with the content

   If local node is congested, root node redirects the customer to another local node

3.3 IPTV Transport Protocol

In this part we will not discuss the details of each transport protocol but we outline some important factors to select protocols. There are different protocols that can be used for the delivery of IPTV Content. The type of protocol that is used depends on the number of factors. First of all the type of video service is important: live television broadcast have different requirement compare with Video On demand service. Secondly, when the content is transmitted to multiple users simultaneously some protocol allow efficiency delivery by using broadcast or multicast technique. Lastly, the delay or latency requirement of IPTV application are also important factors to select appropriate protocols. The following is the list of IPTV transport protocols:

- Transport Control Protocol
- User Datagram Protocol
- Data congestion Control Protocol
- Real time transport protocols
- Real time streaming protocol

IV. Proposed Solution: IPTV Content Delivery Using SDN

In this chapter we present detail explanations of the proposed IPTV Content Delivery Service using OpenFlow Multicast. First, we present OpenFlow-based IPTV Network Architecture (Figure 7), then we describe the overall system architecture of IPTV content delivery service. Thereafter, we present details components that consists Content Delivery Manager and their functionalities.

4.1 OpenFlow-based IPTV network Architecture

In this architecture (Figure 7), content arrives via satellite or terrestrial links, then encapsulated into IP packet at the root node then forwarded to the edge router and passed through core network to local nodes. OpenFlow Controller has a complete knowledge of the entire network (Figure 7). The centralized architecture allows OpenFlow controller to easily manage, make decision and apply actions quickly to forwarding devices in IPTV network.

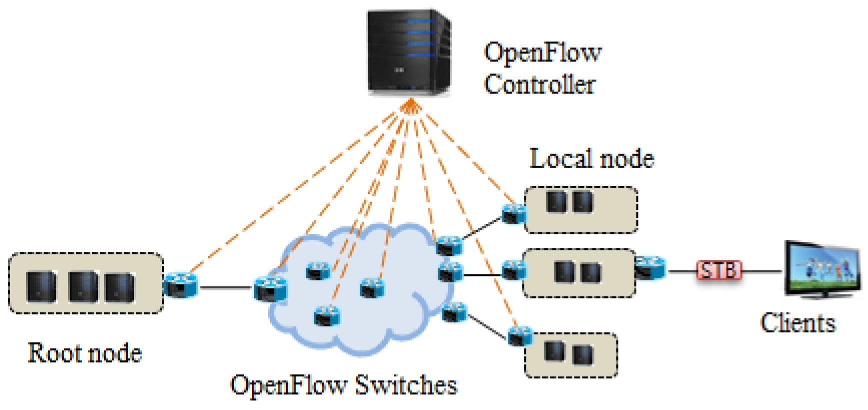

Figure 7. OpenFlow-based IPTV network architecture

4.2 Content Delivery System Architecture

Architecture of the proposed content delivery service consists of three main components including IPTV network, RYU SDN controller and Content Delivery Manager (Figure 8).

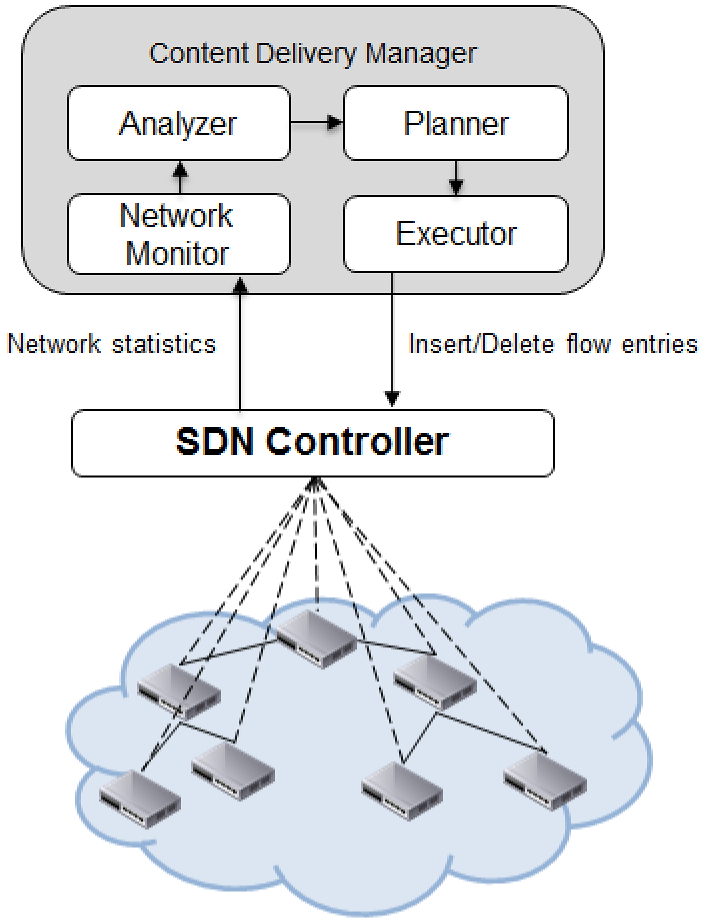

Figure 8. Content Delivery System Architecture

OpenFlow-based IPTV network contains several servers and OpenFlow-enabled switches. This network comprises root node and several local nodes often called edge servers. Root node is the distribution point of video contents towards local nodes via core network. Upon request, OpenFlow-enabled switches report network statistics to the Content Delivery Manager, Content Delivery Manager analyzes the statistics and based on the current network state execute commands to notify controller to update network switches.

Content Delivery Manager which is a fundamental component of our system periodically collects network statistics information, analyzes statistics, plan and executes commands by notifying SDN Controller to insert/ delete flow entries in network switches[27].

4.3 Content Delivery Manager

Content Delivery Manager consists of four modules: Network Monitor, Analyzer, Planner and Executor. Network Monitor periodically collects statistics from network devices, analyzer is responsible to analyze statistics and detect some changes and to send information about changes that happens to the planner module which is responsible to build multicast tree and manage multicast group. Planner sends message that instructs an executor what do to, then executor updates flow entries in network switches through SDN controller.

To implement IPTV Content Delivery using OpenFlow multicast we exploited OpenFlow group table features appeared in OpenFlow specification 1.1version [18].

Group table is used to specify actions to a group of flows. Group table enables executions of multiple actions from single group for multiple flow entries from different tables. Group table have different group types including “all” which is used to implement broadcast and multicast. Packets of this group are processed by all actions buckets. The actions of each bucket are applied to the packet consecutively. For example, a group table entry with type all contains two actions buckets. The first bucket consists of the action “forward to Port 1”. The second bucket consists of the actions “forward to Port 2” then the switch sends the packet to both port 1 and port 2.

In our system implementation we collect network statistics periodically for time interval of different values, 1, 2, 3 seconds, upon result from the analysis of network statistics information, Content Delivery Manager Updates flow entries in network switches. For example Content Delivery Manager after analyzing network statistics and found out that links connecting to local nodes 1 and 4 are congested due to cross traffic in such a way that they cannot receive the video content from root node. Content Delivery Manager sends information to the OpenFlow controller to delete flow entries of switches connecting to local nodes 1 and 4, then local nodes leave multicast group ((Figure 9). On the other hand if local nodes become active (Figure 10) to receive content, automatically OpenFlow controller inserts flow entries into switches and then local nodes to join a multicast group and start to receive contents.

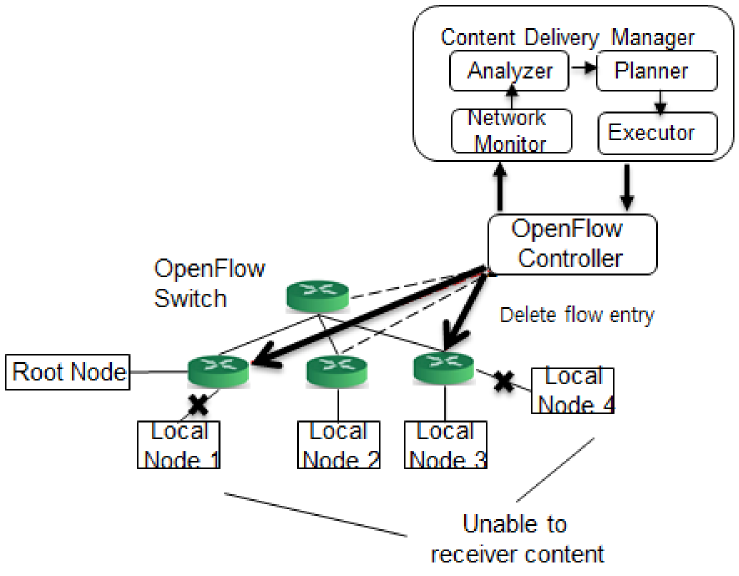

Figure 9. Local nodes 1 and 4 cannot to receive content

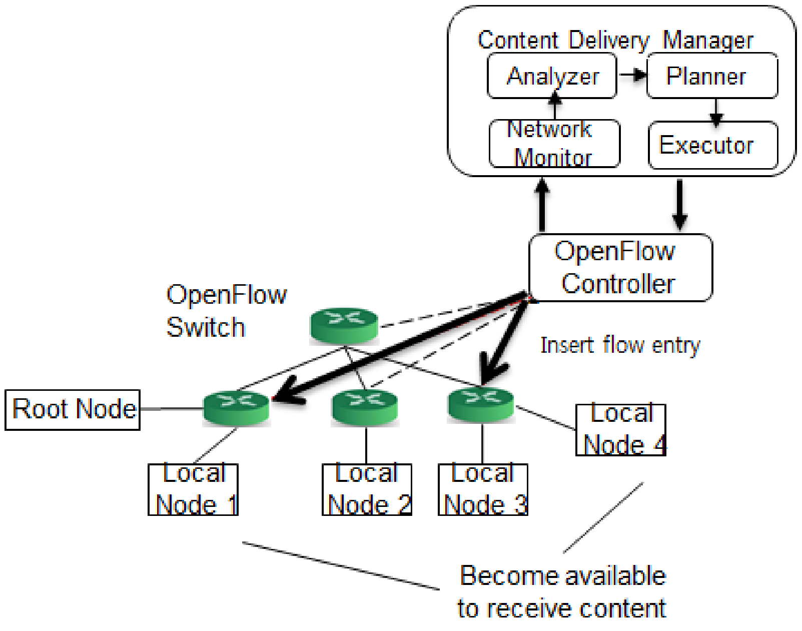

Figure 10. Local nodes 1 and 4 become available to receive content

4.4 VoD Traffic Model Description

Figure 11 depicts a streaming traffic model that sends three multicast connections concurrently without cross traffic. In this model link capacity accommodates three multicast connections with a threshold to avoid congestion. We assume that a cross traffic has a high priority than streaming traffic. We artificially generated and injected cross traffic into network in the middle of VoD traffic transmission. As the link capacity cannot handle all traffics at the same time because of limited bandwidth capacity, one multicast connection is disabled and left the multicast group as shown in Figure 12. When cross traffic terminates the disabled multicast connection can join again multicast group and resumes the VoD traffic. In ideal case it can immediately join multicast group but it presents a small delay after cross traffic ends.

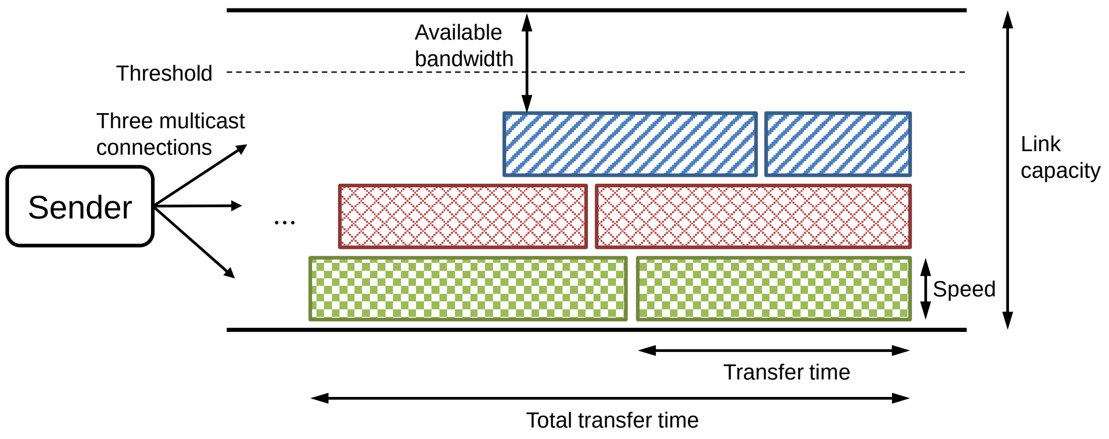

Figure 11.Traffic model

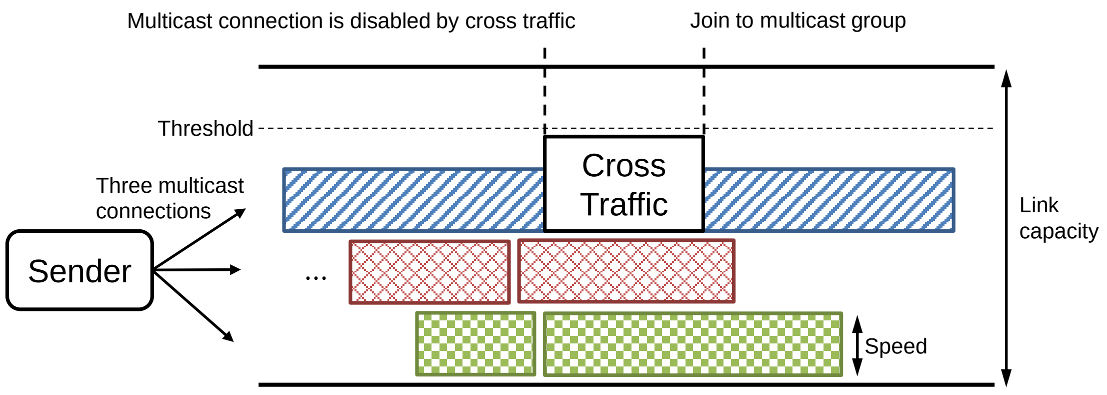
Multicast connection is disabled by cross traffic
Join to multicast group
Threshold
Three multicast connections
Cross Traffic
Link capacity
Sender
...
Speed

Figure 12.VoD traffic with cross traffic in controlled scenario

4. 5 Algorithm on rebuilding the multicast tree

In following section, we will describe the details on multicast tree rebuilding algorithm. The pseudo code of the proposed algorithm has been shown in Figure 13. The algorithm resides inside Planner component in the proposed control loop, and takes four arguments as input, and those are as follow:

Current Multicast Group Entries (current_ge): denotes a set of group entries which stored in OpenFlow switches before rebuilding the multicast tree. Based on these group entries inside switches, the multicast tree is constructed. Each group entry specifies which ports (out_port) are used to forward the outgoing flows, and which ports (in_port) are used to accept the incoming flows.

Available Local Nodes (available_nodes): stores a list of local nodes whose available bandwidth exceeds the pre-defined threshold so that guarantee the multicast traffic to be received without any loss.

Sender Node (sender_node): denotes the node reside in head end IPTV network. Node that, this is the only sender in the multicast tree, there is only outgoing traffic and incoming traffic from sender node.

Topology (topology): denotes a network topology which contains a set of switches and links between switches. Topology information can be retrieve from the topology discovery application which runs on top of OpenFlow controller.

Algorithm 1: OpenFlow based Multicast Tree Building Algorithm

```
input : Current Multicast Group Entries: $current\_ge$
        Available Local Nodes: $available\_nodes$
        Sender Node: $sender\_node$
        Topology: $topology$
1  Initialize $new\_ge$;
   /* Build a new group table instance                    */
2  for $switch \in topology$ do
3      for $port \in$ GetAllPorts($switch$) do
           /* Insert the uplink flows                      */
4          $descendantNodes =$ GetDescendants($switch, port$);
5          if $sender\_node \in descendantNodes$ then
6              AddEntryToMCTree($new\_ge, switch, in\_port = port$);
           /* Insert the downlink flows                    */
7          else if $available\_nodes \bigcap descendantNodes \neq \emptyset$ then
8              AddEntryToMCTree($new\_ge, switch, out\_port = port$);
   /* Different part of current and new group entries      */
9  $\Delta \leftarrow$ Diff($new\_ge, current\_ge$);
   /* Update group tables of switches                      */
10 UpdateToGroupTable($\Delta$);
11 $current\_ge \leftarrow new\_ge$;
```

Figure 13 OpenFlow based Multicast Tree Building Algorithm

The algorithm is mainly comprise of two parts which are 1) calculating the multipath tree by building a new group table instance, 2) comparing the new multipath tree with current multicast tree to obtain the diff part only. By using this diff part we can minimize the number of switches whose group table should be updated to reflect multicast tree changes.

The multicast tree calculation can be further divided into two parts. Since each switch has to deal with both uplink and downlink flow, therefore, we need to specify two directional flows respectively. The sender node should be treated differently, and at least one port of switches on the path between the sender node and core switch should be occupied for transmitting all incoming traffic. Switches which are not on the path between sender node and core switch, should only need to deal with the outgoing traffic. By considering this, we design the multicast tree calculation logic as shown in pseudo code line2-line8.

At first the algorithm traverses the entire topology to obtain all switches, and for each switch it tries to obtain all ports (line2-3). Once a switch and a port in the switch have been specified, the algorithm tries to get all descendant nodes in the sub-tree which is directly connected to the port. And then, we search through all the descendant nodes to find out whether they contain the sender node. If they subsume the sender node, we will add a group entry which is comprised of a pair of switch id and in port number to match over incoming flows (line5-6). If the descendant nodes contain at least one available node, and then we will add a group entry which is comprised of a pair of switch id and output port number to match over the outgoing flows (line 7-8). By going through the above procedures, we can build a new multicast tree (group entries) in accordance with the currently available nodes. Since by using the OpenFlow we can partially change the multicast tree by updating the corresponding switches' group table, thus we try to fetch the different part between new and current multicast group entries, and we call this delta (line 9). Finally, we update the corresponding switches' group table by issuing the group table update message through invoking UpdateToGroupTable method with delta as input parameter.

V. Validation

In this chapter, we describe how we implemented and validated the proposed VoD content delivery service for IPTV using OpenFlow Multicast. First we present testbed environment in which we describe an IPTV network topology, and explain the implementation details of the proposed approach. The detailed analysis results are shown in following section.

5.1 Testbed Construction

In this section, we present the testbed environment for validating the proposed approach. First we describe an IPTV network topology (Figure 14) used for running our experiment; second, we describe the detailed implementation architecture.

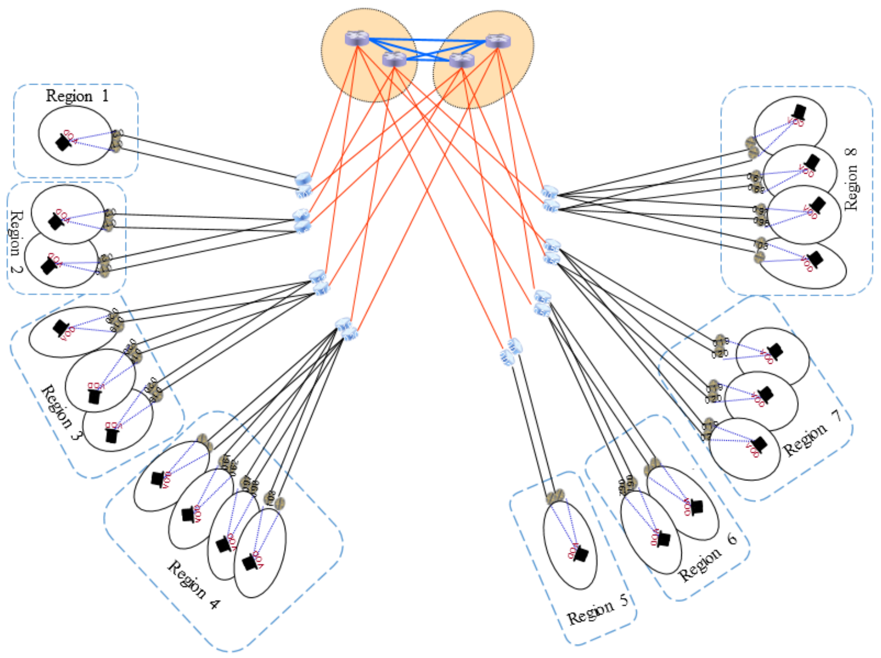

Figure 14. An IPTV Network topology used for running our experiment

The network topology is divided into 8 regions, switches and local nodes reside in each region. A region binds to a geographically separated area in which a certain number of subscribers use the IPTV services. Since the number of subscribers would be varied from area to area, hence, to simulate and accommodate such variation, we assigned different numbers switches and local nodes in each region. For example, there was only one node with two switches in region 1, and two nodes with four switches in region 2 and so on. Overall, with this parameter setup, there were 80 switches, and 20 local nodes in this network topology.

To emulate the network topology, we adopted a well-known network emulator - Mininet [26], in which a set of virtual hosts and virtual switches reside. A virtual host was emulated as a single OS process, whereas, a virtual switch was realized using OpenVSwitch (OVS) [34]. Note that, the exploited OVS supported the OpenFlow protocol up to 1.3 and was compatible with the most of OpenFlow controllers. To the best of our knowledge, Ryu [23] supported almost all features of OpenFlow 1.1, therefore, we adopted Ryu as the OpenFlow controller, and implemented the proposed approach as a Ryu application. Although the resulting solution was relying on specific controller and virtual switch solution, however, since OpenFlow provides vender-agnostic interface, our solution can be easily deployed using any OpenFlow based hardware and software switches.

We call the resulting solution as Content Delivery Manager (CDM), which contains four main components as we have shown in previous chapter. Network Monitor periodically collects network statistics from the forwarding devices (switches) in polling manner, and forwards collected statistics to analyzer component. Analyzer analyzes the statistics, detects changes and sends information to the planner. In our case, we check whether there are any switches change their status either from busy to idle or from idle to busy. As long as any status change occurs, analyzer notifies this to the planner. The planner plans how to handle the content delivery to improve the transmission performance, and in our context, the planner calculates and rebuilds the multicast tree with considering the switch status. Since in the proposed scheme, we do not need to build the entire multicast tree, therefore only a differentiated multicast sub-tree is calculated and resulted as the output of the planner. The executor finally converts the sub-tree into a set of OpenFlow commander and issues to the corresponding switches.

5.2 Experiment Result

To evaluate the proposed approach, we exploited the testbed introduced in previous section. To simulate the video streaming traffic, we artificially generated the UDP traffic using iPerf program with 20 MBps speed. For the sake of simplicity, we generated one streaming traffic, and it was initiated from node 1 which located in region 1, and multi-casted to all residual nodes through intermediate switches. We performed the traffic measurement on node 8 located in region 4, for the purpose of validating whether the OpenFlow based multicast works properly. Moreover, to verify the proposed approach, we artificially generated and injected the cross traffic to node 8 at around 9 second with 22 MBps speed. Note that the threshold value that we chose to trigger the proposed algorithm was around 35 MBps, and the total amount of link capacity was 100 MBps.

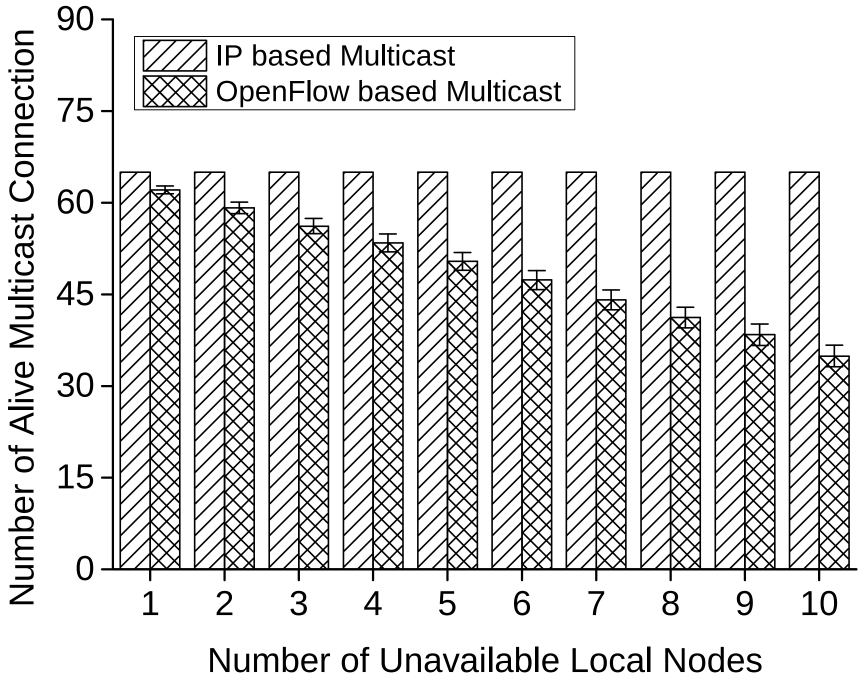

Figure 15.Comparison on the number of alive connection

In the first experiment, we compared the total number of alive connections in the multicast network topology by choosing different multicast schemes. For the comparison purpose, we chose the IP based multicast as the base line approach, and varied the number of unavailable local nodes by generating the cross traffic to randomly selected local nodes. We performed 100 tries for each number of unavailable local nodes and the experiment result has been shown in Figure 15 .

From Figure 15 we can observe that, with IP based multicast, the number of alive multicast connections remained the same, no matter how many number of unavailable local nodes we have. In contrast, since OpenFlow based multicast proportionally prune the unnecessary branches from multicast tree by referring to the network condition, we can observe that the number of alive multicast connections were reduced in accordance with the number of unavailable local nodes increased. More interestingly, the standard deviation of the number of alive multicast connection increased as the number of unavailable local nodes increased. This is because with larger number of unavailable local nodes, there would be much more different ways to prune the multicast tree, and this is mainly determined by the location of unavailable local nodes.

The purpose of the second experiment is to evaluate the performance of planner and executor. The planner deals with building the multicast tree in the case of the availability status of local node changes due to the cross traffic. As we have mentioned in previous chapter, the time for building the multicast tree is mainly contributed by the multicast tree calculation time and the time for obtaining the differentiated part between newly calculated and existing multipath trees. We ran 20 experiments and obtained the time consumed by the planner and the executor respectively, and Figure 16 shows the results on this. The time consumed by the planner was further divided as calculation time and diff time in this figure, and the time consumed by the executor is denoted as execution time. As we can see, the sum of calculation and diff time contributed the most which was around 3.8 ms, whereas executor only required the time which was less than 0.05 ms. Overall, the time for planner and executor was around 4 ms, which is fairly short time for building the multicast group tree.

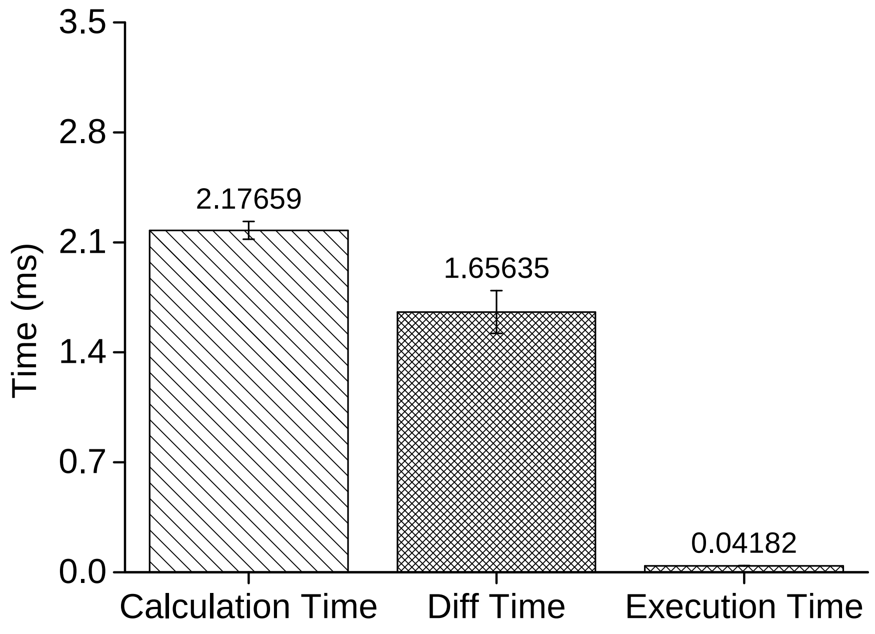

Figure 16. Performance of Planner and Executor

In the following experiment, we evaluated the performance of the entire procedure on the proposed control loop. Since the current version of OpenFlow only supports network metric reporting in polling manner, therefore we implemented the network metric collecting procedure in polling manner with a predefined periodicity. We used the same scenarios as illustrated in Figure 12 but only using one VoD traffic. Three experiments were conducted by varying the polling interval. We set 1 second and 2 seconds respectively for each experiment. The reason why we used different time interval value is to find out the impact of polling interval on the entire control loop procedure.

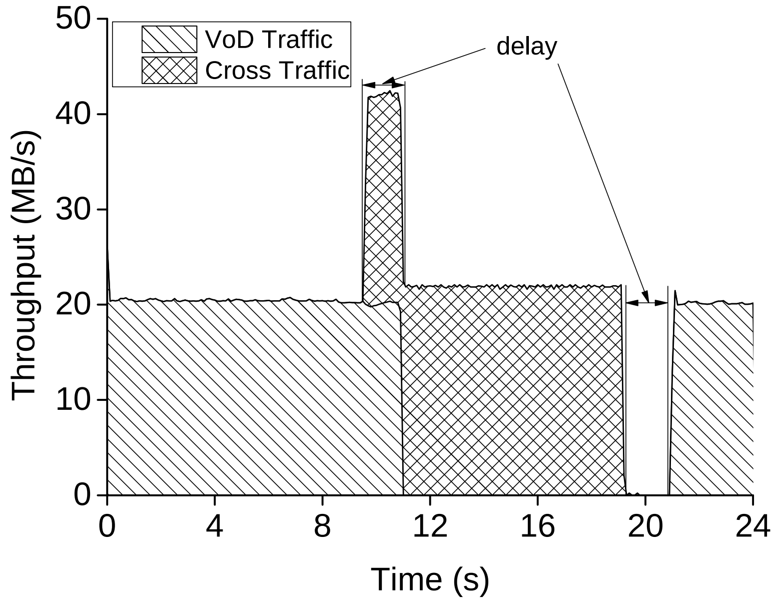

Figure 17. Traffic pattern with 1 second polling interval

The experiment results have been shown in Figure 17 , Figure 18 and Figure 19 . As we can observe from the figures: 1) in all cases, our solution correctly terminated the VoD traffic by forcing the affected local node to leave from the multicast tree, since the sum of VoD traffic and cross traffic exceeded the preconfigured threshold value which was 35 MBps; 2) it also shown that the VoD traffic did not terminate immediately after injecting the cross traffic. There was observable time lag after commencing or terminating the cross traffic. Such consequence was inevitable because the network metric collecting mechanism was implemented in polling manner. Therefore, even if the throughput of a certain link exceeded the threshold value, such event would not be spontaneously reported to CDM, instead, Network Monitor had to sense such event by polling the network metrics with a certain time interval.

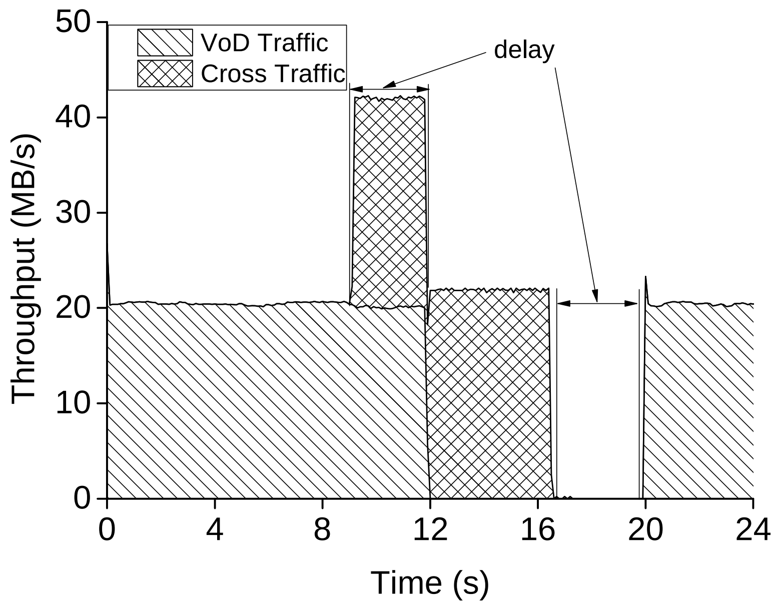

Figure 18. Traffic pattern with 2 seconds polling interval

Moreover, we also measured the lag time during leaving and joining multicast group. Unsurprisingly, as the polling interval increased, the lag time for leaving and joining the multicast group increased accordingly. The detailed results have been shown in Table 2. We also measured how much packets were transmitted during the lag time, and the results have been shown in Table 3.

Although the resulting solution shown lag time in accordance with the polling interval due to the implementation issue, but we have very strong confidence on eliminating such lag time by reporting the status of local node changing event using trap mechanism rather than polling mechanism. However, implementing trap mechanism requires intensive modification on underlying switch as well as OpenFlow protocol which contradicts with the OpenFlow de facto standard.

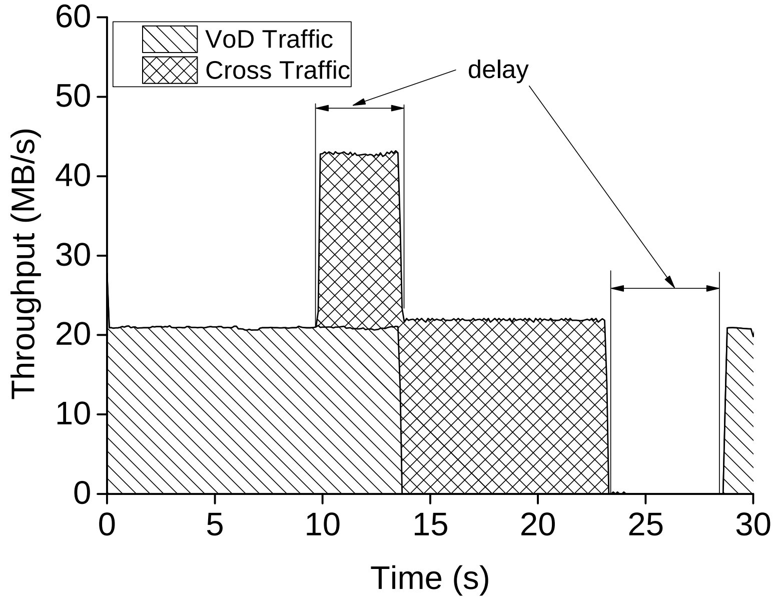

Figure 19. Traffic pattern with 3 seconds polling interval

j

Table 2. Detection delays

	Polling Interval Time		
	1second	2 seconds	3 seconds
Lag time during leaving multicast group	1.3	2.8	4.8
Lag time during joining multicast group	1.2	2.7	4.7

Table 3. Sum of packets transmitted during the overlap time and detection delays

	Polling Interval Time		
	1 Second	2 Seconds	3 Seconds
Detection delays(s)	1.3	2.8	4.8
Sum of packets transmitted(MB)	28057	56884	80778

VI. Conclusion

This chapter presents the summary of the thesis, lists key contributions of the thesis and discuss future work.

6.1 Summary

In this thesis we proposed provider-side VoD content delivery solution. We focused on the VoD content delivery between root node which resides in super head end and local nodes reside in regional head end in IPTV network. The proposed solution is based on multicast protocol, since the IP based multicast suffers from the network resource wastage issue, we adopted the OpenFlow to realize the multicast and further enhance it by exploiting control-loop based management mechanism. We implemented the proposed content delivery solution and named as Content Delivery Manager (CDM) which consists of four components including Network Monitor, Analyzer, Planner and Executor. Network Monitor is responsible to collect periodically network statistic, Analyzer is in charge of analyzing the statistics and detecting changes then informs Planner about changes. Planner based on the current network state rebuilds the multicast tree and sends instructions to Executor. Executor executes instruction from Planner to update flow entries of corresponding switches. We exploited group features to realize multicast, and adopted Ryu SDN controller to deploy the CDM. We performed extensive experiments using virtual network topology and the experiment results shown that OpenFlow based multicast can save network resources compared to IP based multicast.

6.2 Thesis Contributions

The contributions of this thesis are the following:

- Propose a content delivery solution by utilizing the OpenFlow
- Design and implement an OpenFlow based multicast scheme using control loop based management mechanism
- The OpenFlow based multicast can reduce the network resource wastage compared to IP based multicast
- Deploy the solution into emulated network environment and perform validation

6.3 Future Work

In this work we explored only multicast mode of video content delivery using OpenFlow. However, P2P is also important aspect we will consider to extend our future work. As some local nodes may not receive all delivered content because of link congestion, to deliver missing part of content we plan to study how to implement P2P using OpenFlow technology and local node with full content can serve neighboring nodes that need missing part of content. For example one local node experienced congestion in the middle of video content delivery. Our system will remove this local node from multicast group, then after congestion, local node joins again the multicast group. There is a part of video content missing, during the congestion time, this is where P2P can be very useful. This missing part can be obtained from neighboring node instead of making request to the root to avoid congestion. Therefore, our future work will study the feasibility of P2P based content delivery using OpenFlow.

The quality of service (QoS) and quality of experience (QoE) are the most important factors of IPTV service that may attract more customers for IPTV service providers. In our future work we will extend our work to improve both QoE and QoS by exploiting other feature of OpenFlow. Meter table is one of the OpenFlow features we will exploit to implement QoS and QoE in our future.

References

[1] T. Plevyak and V. Salim, “Next Generation Telecommunications Networks, Services, and Management.” JohnWiley & Sons, 2011.

[2] “Software Defined Networking” https://www.opennetworking.org/sdn-resources/sdn-definition

[3] “Software-Defined Networking: The new norm for Networks.” White paper, Open Networking Foundation papers/wp-sdn-newnorm.pdf

[4] A.S. Thyagarajan and S.E Deering, “Hierarchical distance-vector multicast routing for the mbone”, SIGCOMM comput. Commun.Re., vol 25,pp.60-66, October 1995

[5] A. Azgin, G. AlRegib and Y. Altunbasak , “Cooperative On-Demand Delivery for IPTV Networks”, Global Communication Conference (GLOBECOM),2012 IEEE

[6] M. Othman Othman and K. Okamura, “Improvement of content server with Content anycasting Using OpenFlow” In Proceedings of the Asia-Pacific Advanced Network, 2010

[7] D. Agrawal, M.S. Beigi, C.Bisdikian, L. Kang-Won , “Planning and Managing the IPTV Service Deployment. In proceeding of 10 the IFIP/IEEE International Symposium on Integrated Network management IM ‘2007, Munich, Germany May 2007

[8] D. Eager, M. Ferris and M. Vernon, “Optimized Caching for on –Demand Data Delivery.” In Proceedings of Multimedia Computing and Network (MMCN 199), San Jose, California January 1999

[9] S. Ramesh, I. Rhee and K. Guo, Multicast with Cache (Mcache) “An Adaptive Zero- Delay Video-on-Demand Service”. In Proceedings of IEEE INFOCOM 2001, Anchorage, Alaska April 2001

[10] C. Venkatramani, O. Verscheure, P. Brossard, K.W. Lee, “Optimal proxy management for multimedia streaming in content distribution networks”. In Proceedings of NOSSDAV 2002, Miami, USA (2002).

[11] B. Wang, S. Sen, M. Adler and D.Towsley. “Optimal proxy cache allocation for efficient streaming media distribution”. In Proceedings of INFOCOM 2002, USA, 2002 SND and OpenFlow

[12] D. Chang, M. Kwak, N.Choi, T.Kwon and Y.Choi “C-flow: An efficient content delivery framework with OpenFlow”, ICOIN, 2014

[13] E. Hilmi Egilmez , S. Tahsin Dane, K. Tolga Bagci and A. Murat Tekalp, “OpenQoS: An OpenFlow Controller Design for Multimedia Delivery with End-to-End Quality of Service over Software-Defined Networks”

[14] P. McDonagh, C. Olariu, A. Hava, and C. Thorpe, “Enabling IPTV Service Assurance using OpenFlow” International Conference on Advanced Information Network and Application Workshops, 2013

[15] N. New Khaing , T. Phyu and T. Thu Naing ,“IP multicast Content Delivery System for Large Scale Applications” Information and Telecommunication technologies, 2005. APSITT 2005 Proceeding 6th Asia Pacific Symposium

[16] Y. Yu, Q. Zhen, L. Xin, C. Shanzhi,” OFM: A novel Multicast Mechanism Based on OpenFlow”, Advances in information Sciences and Service Sciences, May, 2012

[17] Software-Defined Networking: “The new norms for networks”. White paper, Open Network Foundation, April 13, 2012

[18] “OpenFlow Switch Specification Version 1.1.0.” Open Network Foundation, Feb.28, 2011

[19] “Beacon: A java-based OpenFlow control platform.” October 2011, http://www.beaconcontroller.net

[20] “About Trema Controller.” http://trema.github.io/trema/

[21] “Floodlight SDN Controller.” https://github.com/floodlight/floodlight

[22] “Pox SDN Controller.” https://github.com/noxrepo/pox

[23] “Build SDN Agilely: RYU Controller”, Component-based software defined networking framework.” ttps://github.com/osrg/ryu

[24] F. Douglis, and M. F. Kaashoek, “Scalable Internet Services,” IEEE Internet Computing, Vol. 5, No. 4, 2001, pp. 36-37

[25] G. Pallis, and A. Vakali, “Insight and Perspectives for Content Delivery Networks,” Communications of the ACM, Vol. 49, No. 1, ACM Press, NY, USA, pp. 101-106, January 2006

[26] “Mininet: An instant Virtual Network on your Laptop (or other PC).” http://mininet.org/

[27] N. McKeown, T. Anderson, H. Balakrishnan, G.Parulkar, L.Petterson, J.Rexford, S.Shenker and J.Turner, “OpenFlow: Enabling Innovation in Campus Networks” ACM SIGCOMM April, 2008

[28] W.Simpson and H.Greefied, “IPTV and Internet Video: Expanding the Reach of Television Broadcasting”, Focal Press, UK, 2009

[29] W.Braun and M Menth “Software Defined Networking Using OpenFlow: Protocols, Applications and Architecture Design Choices”, Future Internet, 12 May 2014

[30] N. Feamster, J. Rexford and E. Zegura “The Road to SDN”, ACM, December 30, 2013.

[31] Open Networking Foundation: “OF-CONFIG 1.2 OpenFlow Management and Configuration Protocol”, 2014

[32] A. Nagata, Y.Tsukiji and M. Tsuru, “Delivering a File by Multipath-Multicast on OpenFlow Networks. INCoS, 2013 5th International Conference.

[33] “OpenFlow: The next generation in networking interoperability”, white paper, May 2011

[34] Nicira Networks, “Open vSwitch: Production Quality, multilayer Open Virtual Switch”, http://openvswitch.org

[35] R. Enns, M. Bjorklund, J. Schoenwaelder and A. Bierman,”Request for Comments: 6241, Network Configuration Protocol (NETCONFIG)”, Internet Engineering Task Force, June 2011

Acknowledgement

First of all, I would like to express deeply my sincere gratitude to my supervisor Prof. James Won-Ki Hong for his advice, patience, understanding, support and provide me with opportunity to do my study and research under his guidance. My sincere gratitude also goes to Prof. Jae Hyoung Yoo for his advice, guidance, motivation and encouragement.

My many thanks go to Mr. Jea Yoo Chung and Mr. Jian Li for their support, effort and compassion. This work wouldn't have succeeded without their support. I also thank my fellow labmates in DPNM Lab, POSTECH, Yoon Seon Han, TaeYeol Jeong and Jong-Hwan Hyun, for their support in various ways and for the good time we were working together for two year.

Special thanks to the Korean Government for giving the opportunity and providing me with the scholarship to study and do research in POSTECH. I must also acknowledge Korea Telecom Company for its support to provide with me IPTV network information and helpful explanation.

Last but not the least, I would like to thank my family, especially my Mother for her love and support. To my beloved father who passed away last year I dedicate this thesis.