Provider-Side VoD Content Delivery Using OpenFlow Multicast

- Master Thesis Defense -

Bernard Niyonteze
Email: bernard@postech.ac.kr

Supervisor: Prof. James Won-Ki Hong
Co-Supervisor: Prof. Jae-Hyoung Yoo

Department of Computer Science and Engineering
POSTECH, Korea

June 30, 2014
Outline

- Introduction
- Problem Statement
- Related Work
- Proposed Method
- Validation
- Conclusion & Future Work
INTRODUCTION
Introduction

❖ What is IPTV
  • Internet Protocol Television (IPTV), technology that delivers video or TV broadcasts over the IP network

❖ IPTV Services
  • Live Broadcast
    • Stream content to multiple clients
    • High priority (not delay-tolerable)
  • Video on Demand (VoD)
    • Send content upon customer request
    • Lower priority (relatively delay-tolerable)
IPTV Network Architecture

Research Scope

Source
Super Head-end (SHE) (Root Node)

Content source

Core network

Access network

Home network

Access network

Regional Head-end (RHE) (Local Node)

Video Service Office (VSO)

Video Service Office (VSO) (Local Node)

STB

Office (VSO)

Office (VSO)
Problem Statement

- IP multicast presents a waste of network resources in VoD content delivery
  - Inefficient to manage multicast tree
    - Source transmits packets continuously even if edge links are congested
    - The receiver drops packets but core links are still occupied by multicast traffic

Multicast-based VoD content delivery

Root Node

Local Nodes

Cannot receive packets
  - Link congestion or failure
  - Violate management policy
Research Goals

- **Propose multicast-based VoD delivery service using Software-Defined Networking (SDN)**
  - Provider-side efficient content delivery
  - Dynamically adjust multicast tree while monitoring link utilizations
    - Implement control loop at SDN controller
    - Resolve wasted network resources

- **Validate proposed method**
  - Emulating proposed method on Mininet with 20 nodes and 80 switches
    - Approximately ½ scale of a Korean IPTV service provider with 5 million subscribers
  - Comparing number of active links with IP-based multicast
  - Detecting link status and pruning unnecessary links
Related Work (1/2)

❖ **Content Delivery Networks**
  - Content Anycast & P2P (APAN, 2010)
    - Content can be stored in multiple nodes (server as well as client)
    - Servers redirect requests to clients
  - C-flow (ICOIN, 2014)
    - Use OpenFlow to enhance content delivery using dynamic re-routing, parallel transmission and cache management

❖ **IP Multicast**
  - IP Multicast Content Delivery System (APSITT, 2005)
    - Extension of IP multicast service model
    - Support multicast addressing and filtering
Related Work (2/2)

♦ Software-Defined Networking (SDN)
  ♦ Separate the control plane from the data plane

Traditional networks

SDN Networks

Controller

SDN

OpenFlow Protocol

Flow Table

<table>
<thead>
<tr>
<th>Flow entry</th>
<th>match field</th>
<th>counter</th>
<th>action</th>
<th>priority</th>
<th>Timeout...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Actions (Instructions)
1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields
6. etc

Niyonteze Bernard, POSTECH Master Thesis Defense
PROPOSED METHOD
Proposed Method

meteorological

Architecture and control loop

- **Proposed Method**
- **Architecture and control loop**

**Diagram:**
- SDN Controller
  - Analyzer
  - Planner
  - Network Monitor
  - Executor
- Network statistics
- Insert/Delete flow entries
- Monitor link Utilization
- Calculate available nodes to receive VoD
- Calculate multicast tree
  - Is tree changed?
    - No
    - Yes
  - Update flow entries
Multicast Tree Calculation

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 ∈ topology do
3.   for port ∈ GetAllPorts(switch) do
4.     /* Insert the uplink flows */
5.     descendantNodes = GetDescendants(switch, port)
6.     if sender_node ∈ descendantNodes then
7.       AddEntryToMCTree(new_ge, switch, in_port = port);
8.     else if available_nodes ∩ descendantNodes ≠ Ø then
9.       AddEntryToMCTree(new_ge, switch, out_port = port);
10.    /* Insert the downlink flows */
11. /* Different part of current and new group entries */
12. Δ ← Diff(new_ge, current_ge);
13. /* Update group tables of switches */
14. UpdateToGroupTable(Δ);
15. current_ge ← new_ge;

- Obtained by link utilization data from network monitor
- Find nodes filtered by given switch and port
- Actually send update messages to switches
Example – Low Throughput Caused by Cross Traffic

- **Cross traffic**
  - Interfere with VoD traffic
    - Increase packet loss probability
  - To avoid packet loss
    - Serve higher prioritized traffic first
    - VoD distribution is controllable traffic

Cross Traffic Occurs

Unable to receive content
Example – Low Throughput Caused by Cross Traffic

[Diagram showing network nodes and processes involved in network monitoring and control.]

- **Root Node**
- **Local Node 1**
- **Local Node 2**
- **Local Node 3**
- **Local Node 4**
- **OpenFlow Switch**
- **Content Delivery Manager**
  - Analyzer
  - Planner
  - Network Monitor
  - Executor

Become available
VALIDATION
Experiment Environment

Software
- Controller: Ryu 3.9
- OpenFlow Switch: OVS 2.1.2
- Network Emulator: Mininet 2.1.0
- Traffic generator
  - VoD traffic: implement in Python
  - Cross traffic: Iperf
- OS: Ubuntu 12.10

Topology
- Approx. ½ scale of a Korean IPTV service provider
- 8 regions
- 1 to 4 subnets in each region
- Number of switches: 80
- Root node (sender): 1
- Local nodes (receiver): 19
Simulation Scenario

- **Simulation parameters:**
  - Threshold: 35 MB/s
  - Generated VoD traffic: 20 MB/s
  - Generated cross traffic: 22 MB/s
  - Scaling down simulation parameters
    - Difficulties in generating high bitrate VoD and cross traffic

- **Traffic generation**
  - Sending VoD traffic from root node to all local nodes - multicast
  - Sending cross traffic from a local node to another local node - unicast

- **Monitoring and actions**
  - Every 1-second, collect byte counts of edge links to find which nodes cannot have enough bandwidth to receive VoD traffic
  - Generate VoD traffic and cross traffic
  - Calculate multicast tree and compare it with previous tree
  - Find added/deleted/modified flow entries (switch id, output ports)
  - Send message to update flow/group entries
Simulation Results (1/2)

- **Number of active links in multicast**
  - Comparing with IP-based multicast
    - OpenFlow multicast shows less number of active links
    - OpenFlow multicast prunes unnecessary links in multicast tree - benefit

- **Calculation time and execution time**
  - Tree calculation time and comparison time
    - 2.18 ms and 1.66 ms respectively
  - Execution time is 0.0418 ms
    - Controller just sending update messages without ACK
Simulation Results (2/2)

- Control loop and detection time

![Graph showing throughput and time intervals for VoD Traffic and Cross Traffic. The graph includes a legend highlighting the difference between VoD Traffic and Cross Traffic, with a note indicating 'delay' in the network response. The x-axis represents time in seconds, ranging from 0 to 24, while the y-axis shows throughput in MB/s, ranging from 0 to 50. The graph visualizes the polling interval for 1 second, with VoD Traffic and Cross Traffic represented by distinct patterns and colors.]
CONCLUSION
Conclusion and Future Work

❖ Contributions

- Design OpenFlow-based content delivery system for service providers
- Dynamically adjust multicast tree
  - Design and implement simple control loop at controller
  - Save wasted link bandwidth
- Validate proposed method using Mininet with real-world-like topology (20 nodes; 80 switches)

❖ Future Work

- Explore other features of OpenFlow to guarantee QoS and QoE
- P2P-based content delivery system on OpenFlow
THANK YOU
References


Traffic Model

VoD Traffic

VoD Traffic with Cross Traffic
\section*{Simulation Results (2/3)}

\subsection*{Detection delay}

Polling interval for 1 Second

Polling interval for 2 Seconds
Simulation Results (3/3)

Detection delay

Time to calculate multicast tree was around 6.4 ms

Polling interval for 3 Seconds

<table>
<thead>
<tr>
<th>Polling Interval Time</th>
<th>1 second</th>
<th>2 seconds</th>
<th>3 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lag time during leaving multicast group</td>
<td>1.3</td>
<td>2.8</td>
<td>4.8</td>
</tr>
<tr>
<td>Lag time during joining multicast group</td>
<td>1.2</td>
<td>2.7</td>
<td>4.7</td>
</tr>
</tbody>
</table>

Detection delay
Two Unavailable Nodes (1/2)

Multicast-based VoD content delivery

Root Node

Local Nodes

Cannot receive packets
- Link congestion or failure
- Violate management policy

Prune three links
Two Unavailable Nodes (2/2)

Multicast-based VoD content delivery

Root Node

Cannot receive packets
- Link congestion or failure
- Violate management policy

Local Nodes

Prune two links