ISBN 978-89-6211-377-8 98500

기술문서

# Rendezvous Point에 기반한 Packet Flow의 사상(Mapping) 방법 연구

- IP 멀티캐스트를 위한 오버레이 프록시의 설계 v1.0

JINYONG JO
jiny92@kisti.re.kr
Supercomputing Center.
Korea Institute of Science and Technology Information (KISTI)

# Copyright

# Contents

# Chapter 1

# Overlay Multicast Proxy의 개발

## 1.1 Introduction

본 연구는 IP multicasting을 대행할 overlay proxy 및 controller의 구현을 목적으로 한다. 연구에서는 IP multicasting을 위한 NetFPGA기반 Overlay proxy의 설계 등 멀티캐스트 네트워킹 기반 인프라를 개발하는데 단기 목적이 있으며, Service-enabling overlay(SeeOver) 인프라를 구축 지원하는 것을 최종 목적으로 한다. 구성요소는 데이터평면(data plane)에서 고속 패킷 포워딩(high-speed packet forwarding) 등을 담당할 Proxy 노드, 라우팅 등 제어평면(control plane)의 유지·관리를 위한 Controller, 각 구성요소 간 시그널링을 위한 내부 통신 규약(protocol) 등으로 이루어진다. 본 문서는 데이터평면의 설계 및 구현에 논지를 둔다. 개발 시스템은 접속망(access network)에 연동된 종단 호스트에게 IGMP(Internet group management protocol) 등 멀티캐스트 통신 규약을 제공하며 중계망에서는 Point-to-point방식의 유니캐스트 네트워킹 서비스를 제공함으로써 종단 호스트는 통신 규약이나 기존 협업 소프트웨어 등의 변경없이 다대다 협업 환경에 참여할 수 있다.

### 1.1.1 Problem Definition

IP multicast는 수익모델의 부재, AS간 정책적 차이 및 유지 관리의 문제점 등으로 기술적 장점에도 불구하고 구축 서비스가 제한적으로 진행되어 왔으나 근래에 IPTV등 상용 서비스 모델이 등장하고 e-Science와 같은 과학기술 협업 환경에 요구 등 기술의 확대 적용에 대한 기대가 증가되고 있다. 이러한 시점에서, IP 멀티캐스트 서비스의 부재 및 도메인 간 상호 라우팅 문제 등으로 인해 발생하는 서비스 단절 및 응용 서비스의 저품질화 등과 같은 다양한 문제점들을 극복하고 다자간 망기반 협업이 가능하도록 안정적인 IP 멀티캐스팅 인프라를 제공할 수 있는 기술적 접근이 요구된다.

IP 멀티캐스트의 비신뢰적 전송 등 품질 문제과 인프라 구축 지연 등으로 인한 서비스 부재의 문제를 해결하기 위해 Overcast [1], RON [2], ScatterCast [3], Yoid [4], ECM [5], ALMI [6] 등 다양한 오버레이 멀티캐스트 연구들이 진행되어 왔다. 관련 연구들은 오버레이 구조의 확장성, 연결성, 신뢰성 등의 향상에 초점을 두었고, 응용 계층에서 IP 패킷을 처리하기 때문에 패킷 포워딩 성능이 낮거나, 패킷을 중계를 담당하는 호스트들의 잦은 멀티캐스트 참가 및 탈

퇴(join and leave)로 인해 트리 구성이 안정적이지 못한 단점이 있다. 또한, 오버레이 인프라에 참여하는 호스트 간 메시지 교환을 위해 비표준 통신 규약이나 인프라 종속적인 패킷 헤더 등을 사용해야 하므로 종단 호스트의 응용소프트웨어 또는 시스템의 변경을 필요로 한다. 일례로, 오버레이 프록시(수퍼노드)를 사용하는 ScatterCast는 평균 350Mbps의 패킷 처리율을 보이며 패킷당 처리지연이 200∼600$\mu$s (최대 2∼3 ms) [3]로써 다수의 프록시를 통해 라우팅 될 때 높은 지연변이(Jitter)가 발생하는 등 인프라를 통해 제공되는 서비스의 품질 저하를 예상할 수 있다.

### 1.1.2 Resolution and Significance

본 연구에서는 망기반 다자간 협업 응용의 고품질 서비스가 가능한 멀티캐스트 인프라의 안정적 제공을 위해, 제어평면과 데이터평면을 분리해 각각 멀티캐스트 세션 및 사용자 관리와 고속 패킷포워딩을 담당하게 한다. 데이터평면에서 IP 패킷포워딩을 하드웨어 상에 구현함으로써 처리성능을 향상·안정화시키고, 제어평면을 통해 멀티캐스트 세션에 대한 접근제어를 용이하도록 하는데 최종 목적이 있다.

종단 사용자는 기존 IGMP(Internet group management protocol) 통신규약을 이용해 오버레이 프록시와 상호 연동된다. 프록시는 LAN상에서 2계층 투명 브리징(transparent bridging) 서비스를 제공하며, 프록시들간 IP 패킷 포워딩을 위해 3계층 IP-in-IP 캡슐화 또는 역캡슐화 등이 서비스된다. 기존 IP 프로토콜 스택 및 응용소프트웨어의 변경이 불필요하며 오버레이 백본 내부의 전송 방식들은 사용자에게 보여지지 않으므로, 사용자는 IP 멀티캐스트와 기능적으로 동일한 서비스를 제공받게 된다.

2계층 투명 브리징과 3계층 패킷 포워딩을 위해 4-port 프로그래머블 이더넷 카드인 NetPFGA [7]가 사용된다. 실시간 패킷처리에 부담이 큰 패킷 포워딩과 복제는 규정된 flow table에 기반해 NetFPGA가 담당하고 IGMP 시그널링 등 제어평면과 관련된 프로세스는 시스템 소프트웨어에서 담당하게 함으로써 처리 성능을 높인다.

종단호스트는 스위치(switch)나 허브(hub)에 PC가 연동되는 방법과 동일하게 프록시 노드에 연결된다. 제약 사항으로, NetFPGA 카드의 특성 상 업링크 또는 다운링크가 4 포트로 제한된다는 점을 들 수 있다. 하지만, 개발된 프록시가 2계층 브리징을 지원하고 다른 2계층 스위치와 연동 가능하다는 점에서 포트 제한 문제를 극복할 수 있다.

개발 시스템은 IGMP 등 IP 멀티캐스트 규약을 준수하고 기존 멀티캐스트 응용을 변형없이 수용하며, NetFPGA를 활용해 고성능 패킷 포워딩 및 복제를 수행하므로써 Gbps급 패킷 처리 성능을 보장한다. 또한, NetFPGA 카드가 내장된 시스템의 가격이 $1,500 수준의 저가이므로 비용효율적 프록시의 구축이 가능한 장점이 있다. IP 멀티캐스트 서비스로부터 고립되어 있거나 라우팅 정책 등으로 인해 연결성 문제를 갖고 있는 종단 사용자에게 멀티캐스트 환경을 제공함으로써 다자간 협업을 위한 비용 효율적 네트워킹 서비스를 제공한다.

## 1.2 NetFPGA

NetFPGA는 4개의 1 Gbps 이더넷 포트를 갖는 PCI 카드로써 하드웨어, 게이트웨어(Gateware), 및 소프트웨어로 구성된다. 하드웨어는 Xilinx Virtex-II Pro 50 칩셋에 18 MBit ZBT(Zero-bus turnaround) SDRAM 및 64 Mbytes DDR DRAM을 갖는다. 소프트웨어는 장치드라이버, 유틸리티, 라우터 제어 패키지(라우팅 소프트웨어 및 리눅스 라우팅 테이블 관리 데몬)를 포함하며 게이트웨어는

Verilog HDL 작성된 모듈의 집합으로써 IPv4 라우터 및 4 포트 NIC(network interface card)과 관련된 참조설계(Reference design) 코드를 제공한다.

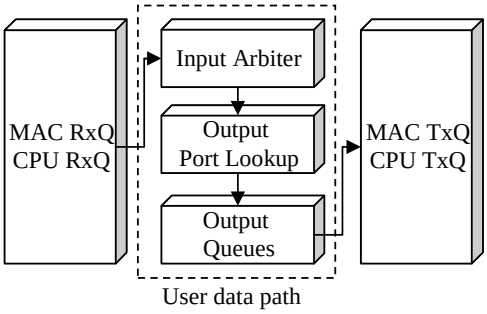


Figure 1.1: Reference pipeline.

참조설계에서 제공하는 참조파이프라인(Reference pipeline)의 데이터경로는 Fig.1.1과 같이 입력중재기(Input arbiter), 출력포트 검색(Output port lookup), 출력 큐(Output queues)로 구성되며 입력중재기는 처리되어야 할 패킷을 담고 있는 수신 큐를 선택하고, 출력포트 검색은 패킷의 출력 큐를 선택하며 출력 큐는 출력 포트가 준비될때까지 패킷을 저장하는 역할을 한다. 오버레이 프록시는 "출력포트 검색" 부분에 2계층 투명 브리징 및 3계층 패킷 포워딩 등의 기능을 구현하고 하드웨어 제어 및 통신규약 서비스를 위한 부분을 사용자 영역에 구현함으로써 완성된다.

## 1.3 Design Principle and Limitations

오버레이 프록시 설계의 기본 원칙은 다음과 같다.

- 종단 호스트의 입장에서 IP protocol stack에 변형이 없어야 한다. 즉, 멀티캐스트를 지원받기 위해서 기존 2계층 및 3계층 통신 프로토콜을 유지해야 한다.
- IP 멀티캐스트를 이용하는 기존 응용프로그램은 소프트웨어 구조의 변경 없이 proxy node와 연동되어야 한다.
- 종단 호스트에서 proxy node와 네트워킹 등을 위해 추가적으로 필요한 설정 변경(MSS 크기 등)이 최소화 되어야 한다.
- 오버레이 네트워크를 구성하는 제어 및 데이터 평면의 모든 규약은 최대한 간소화된 구조를 갖는다.
- 오버레이 네트워크는 RTCP(RTP control protcol) 등 피드백 메시지들을 RTP(real time transport protocol)와 같은 활성 패킷(active packet)과 동등하게 수용해야 한다.
- 멀티캐스트-유니캐스트 브리징(multicast-unicast bridging) 등 기존 IP 멀티캐스트 보완 기술들을 수용할 수 있는 확장성을 지녀야 한다.

추가적으로 현재 구현된 오버레이 프록시의 제한 사항은 다음과 같다.

- IPv4에 대해서만 지원하며 IPv6의 지원은 고려하지 않는다. 단, IPv6 트래픽을 IPv4 터널링 또는 브리징(bridging)할 수 있는 기능 확장은 고려된다.
- 오버레이 네트워크에서는 패킷 단편화(fragmentation) 문제를 처리하지 않는다. IP-in-IP 방식의 사용으로 인해 패킷 단편화가 발생할 경우 IP router가 처리한다.
- IGMPv1과 v2만 지원하며 IGMPv3의 지원은 고려하지 않는다.
- 종단 호스트가 오버레이 네트워크를 활용하기 위해서는 반드시 프록시 노드와 직접 연결되어 있어야 한다. 단, 멀티캐스트-유니캐스트 브리징 서비스 제공을 위한 기능 확장은 고려된다.

## 1.4 Architecture

### 1.4.1 Overview

프록시 노드는 그림 1.2와 같이 크게 고속 패킷 포워딩과 MAC 포워딩 기능을 수행하는 하드웨어 부분과 통신규약 등의 처리를 위한 소프트웨어 부분으로 구분된다.

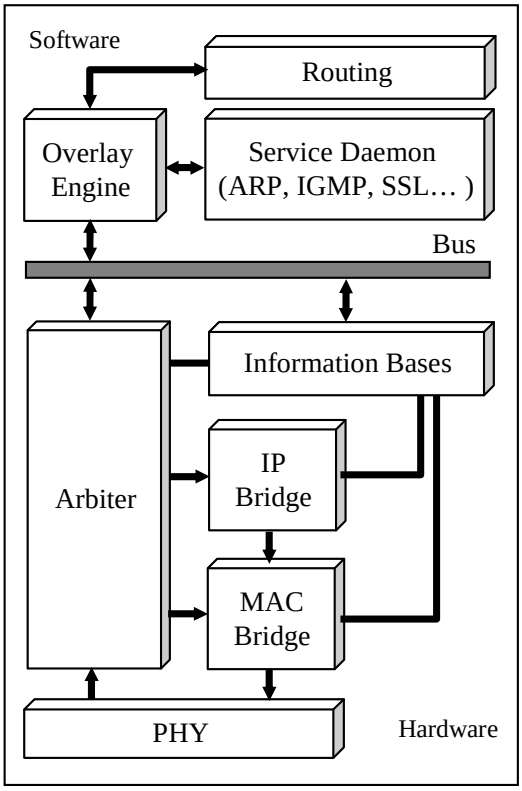


Figure 1.2: System Architecture (Software & Hardware).

하드웨어 부분은 NetFPGA 플랫폼에 Verilog를 이용해 코딩되며 각 블럭별 기능은 다음과 같다.

1. `Arbiter`, 물리계층(`PHY`)을 통해 입력되는 MAC 프레임 및 IP 패킷의 종류를 파악하고, 종류에 따른 처리 블럭을 선택한다. 또한, MAC table을 유지·관리하고 IP checksum 오류를 검사한다. 예를 들어, ARP(Address Resolution Protocol)에 해당하는 MAC 프레임을 수신 받았을 경우에, 프로토콜을 처리하기 위해 소프트웨어(`Software`)부분의 `Overlay Engine`으로 분기시킨다. 또한, `Arbiter`는 `Information Bases`의 Flow table, MAC table, 시스템 상태 정보 등을 갱신하고 `Information Bases`의 정보를 이용해 패킷 및 프레임 입·출력을 관할한다.

2. `MAC Bridge`, 2계층 MAC 프레임의 스위칭(switching) 기능을 담당한다. `Information Bases`가 보유중인 MAC table 읽어 출력 포트 등을 결정하고 MAC 프레임을 생성한다.

3. `IP Bridge`, `Information Bases`이 보유 중인 Flow table을 읽어 IP 패킷을 캡슐화(encapsulation) 또는 역캡슐화(de-encapsulation)한다. 캡슐화는 IP-in-IP 방식이며 캡슐화된 IP 패킷은 point-to-point로 연결된 이웃(neighbor) Proxy에게 전달되며, 역캡슐화되는 IP 패킷은 다운링크에 연동된 종단 호스트들에게 전달된다.

Flow table은 그림 1.2의 `Software` 부분에서 보유하고 있으며 `hardware` 부분의 `Information bases`도 복제된 형태의 flow table을 함께 보유한다. 하드웨어가 갖는 메모리 용량에 한계가 있기때문에 하드웨어 상에 수용할 수 있는 flow table의 수가 제한되기 때문이다. `Software` 영역에서는 시스템 메모리를 사용하기 때문에 다수의 테이블을 저장할 수 있다.

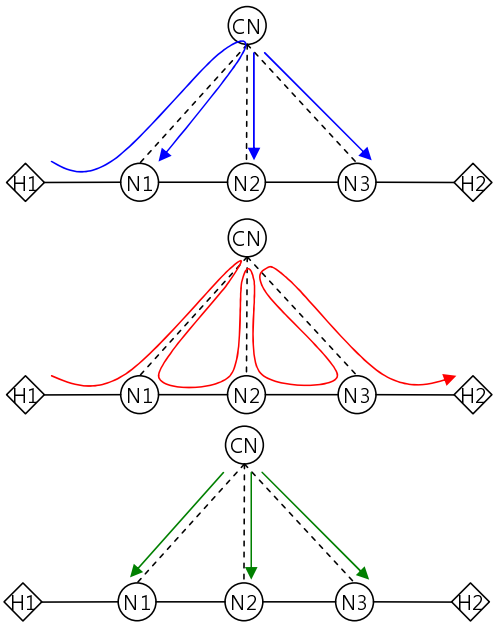


Figure 1.3: 중앙관리(위), 자가제어(가운데), 사용자제어(아래).

Flow table을 완성하기 위한 경로 라우팅(routing)은 그림 1.3와 같이 1) 중앙관리식, 2) 자가제어식, 3) 사용자제어식 등 3가지 방법이 사용될 수 있다. 중앙관리식은 관리서버가 네트워크의 모든 토폴로지 지도(topology map)와 상태정보를 가지고 있으며, 프록시의 요청이 있을 경우에 종단간 라우팅 경로를 관리서버가 제공해 주는 방식이다. 진출·입 노드(ingress·egress proxy node)가 새로운 플로우에 대해 엔트리(flow entry) 등록을 요청할 경우, 관리서버는 종단간(프록시 노드) 라우팅 경로를 계산하고 경로 상에 존재하는 프록시 노드들에게 플로우 엔트리를 전달해 플로우 테이블에 기록하게 한다.

자가제어식은 프록시 노드가 자체라우팅 기능을 탑재하고 있어서 분산·독립적으로 라우팅을 수행한다. 자가제어식의 경우 관리서버는 라우팅 정보를 제공해주지 않지만 내부 정책에 의해 프록시 노드들을 그룹핑(grouping)하는 역할을 한다. 각 프록시 노드들은 스스로 엔트리를 구성해 플로우 테이블에 기록한 후에 관리서버에게 해당 엔트리를 통보한다. 관리서버는 프록시 노드가 보낸 플로우 엔트리를 사후 승인하는 방식을 취한다. 승인하는 과정 또는 플로우 엔트리가 설정된 이후에도 관리서버는 각 프록시 노드들에게 플로우 엔트리에 대한 변경을 요구할 수 있다(3가지 방식 모두에 해당된다.).

사용자제어식은 특정 라우팅 알고리즘을 이용하지 않고 사용자가 수동으로 라우팅 경로를 설정하는 방식으로, 프록시 노드는 사용자에 의해 정의된 플로우 엔트리를 이용해 IP 패킷을 포워딩한다. 사용자는 경로 정보와 경로 상에 존재하는 프록시 노드들이 가져야 할 플로우 엔트리를 관리서버에 저장한다. 관리 서버는 사용자의 요청에 따라 기 기술된 플로우 엔트리를 프록시 노드들에게 전달한다.

중앙관리식 및 자가제어식에서 사용되는 라우팅은 OSPF(Open shortest path first)등 기존 최적화(optimal) 알고리즘을 사용한다. 모든 방법에서 관리서버는 "패킷을 처리하기 위해 프록시 노드가 취해야 할 지침"을 나타내는 `Action`에 대해서, 임의의 시간에 각 프록시 노드들에게 명령할 수 있다.

소프트웨어 부분의 `Overlay engine`은 `Arbiter`가 제공하는 하드웨어 레지스터를 읽기 및 쓰기(read/write)함으로써 하드웨어와 스프트웨어 간 통신 인터페이스를 제공한다. 또한, 하드웨어의 요구에 따라 패킷을 개별 `Service daemon`으로 전달하거나 그 역(서비스데몬의 요청에 따라 패킷을 하드웨어에 전달)의 기능을 수행한다. 예를 들어, 하드웨어로부터 ARP(address resolution protocol) 요청을 받았을 경우에 ARP `Service daemon`으로 패킷을 전달한다. 프록시 노드가 지원하는 `Service daemon`은 ARP, IGMP 및 SSL(Secure Socket Layer)이다. IGMP는 `query`의 발생과 `report`에 대한 처리를 포함하며 상위 라우터에서 발생하는 IGMP `query` 메시지는 프록시 노드에의해 필터링된다. SSL은 네트워크 계층의 암호화 방식으로 Controller와 프록시 노드가 통신할 때 사용된다.

### 1.4.2 Formats in Layer 2 & Layer 3

2계층 MAC 프레임이 제공하는 Ethernet type은 다음과 표.1.1과 같이 정의되어 있다. 프록시 노드는 MAC 프레임의 Ethernet type을 읽고, IP와 ARP에 대해서 3계층 처리를 수행하며 기타 2계층 MAC 프레임들에 대해서는 브리징 서비스를 제공한다.

2계층 ARP 프레임은 Ethernet header 및 ARP header를 포함하는 Ethernet ARP fields로 구분된다. ARP header는 hardware type, protocol type, hardware length, protocol length 및 operation field로 구분되며 각 필드는 표.1.2과 같이 정의된다. 참고로 RARP(Reverse Address Resolution Protocol)는 물리주소를 논리주소로 바꿔주는 역할을 한다.

Table 1.1: Ethernet Type.

| Type(Hex) | Name |
|---|---|
| 0x00F1 | NetBIOS |
| 0x0800 | IP |
| 0x0806 | ARP |
| 0x8035 | RARP |
| 0x809B | AppleTalk |
| 0x8100 | VLAN |
| 0x8137 | IPX |

Table 1.2: ARP structure.

| Field name | size(byte) | Valuse |
|---|---|---|
| hardware type | 2 | 1 |
| protocol type | 2 | 0x0800 |
| hardware length | 1 | Ethernet의 경우 6 |
| protocol length | 1 | IP의 경우 4 |
| operation | 2 | ARP request(1)<br>ARP reply(2),<br>RARP request(3),<br>RARP reply(4) |

프록시 노드는 2계층 및 3계층 서비스를 위해 다음과 같은 부가적인 기능을 수행한다.

1. 물리적으로 연동된 업링크와 다운링크를 구분한다.
2. Unicast MAC 테이블을 제공한다.
3. Multicast MAC 테이블을 제공한다. Multicast MAC 테이블은 Unicast MAC 테이블과 통합해서 사용될 수 있으며 `Information bases`에 저장된다.
4. Multicast용 Ethernet 프레임을 처리하며 멀티캐스트 패킷 필터링을 포함한다.

3계층 멀티캐스트는 224.0.0.0~239.255.255.255(Class D)의 IPv4 주소대역을 사용한다. 224.0.0.0~224.0.0.255는 멀티캐스트 통신을 위한 제어 대역으로 각 멀티캐스트 주소의 역할은 표.1.3와 같다. 예를 들어, 멀티캐스트 주소 224.0.0.1은 임의의 호스트가 속한 서브넷(subnet)상의 모든 호스트들에게 멀티캐스트 패킷을 송신할 때 사용된다. IGMP 패킷의 경우, `join`을 위해서 224.0.0.1(01-00-5e-00-00-01), `leave`는 224.0.0.2(01-00-5e-00-00-02), `report`는 multicast group address를 사용한다.

Class D 멀티캐스트 주소를 IEEE 802 MAC 주소에 맵핑(mapping)하기 위해서 2계층 프레임은 그림.1.4와 같이 `01-00-5E-`로 시작되는 주소를 갖는다. `01-00-5E-`로 시작되는 2계층 멀티캐스트 주소는 ARP를 필요로 하지 않는다. 3계층 멀티캐스트 주소 중 하위 23비트가 3계층 이더넷 주소의 하위 23비트로 복사되어 MAC 프레임을 구성한다.

Ethernet type이 0x0800인 IP 패킷은 헤더에 protocol 번호를 나타내는 필드를 갖는다. 프록시 노드가 처리해야 하는 프로토콜의 종류는 표.1.4와 같다. 프

Table 1.3: 멀티캐스트 제어 주소.

| IP address | roles |
|---|---|
| 224.0.0.1 | All systems on this subnet |
| 224.0.0.2 | All routers on this subnet |
| 224.0.0.4 | All DVMRP routers |
| 224.0.0.5 | All OSPF routers |
| 224.0.0.6 | All OSPF designated routers |
| 224.0.0.9 | All RIP2 routers |
| 224.0.0.13 | All PIM routers |
| 224.0.0.15 | All CBT routers |

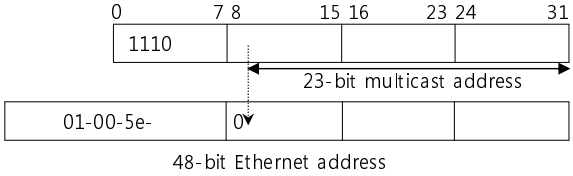


Figure 1.4: Ethernet address for multicast.

록시 노드에서 이웃하는 proxy로 패킷을 전송할 때, IP-in-IP 방식의 캡슐화를 이용한다. IP-in-IP의 프로토콜 번호는 0x04이지만 프록시 노드는 실험 목적으로 관리되고 있는 프로토콜 번호 0xFD(253)번을 활용한다. 0xFE(254)는 오버레이 네트워크의 제어용 패킷 헤더의 프로토콜 번호로 할당되어 사용한다.

Table 1.4: IP Protocol Number.

| Hex | Keyword | Protocol | Reference |
|---|---|---|---|
| 0x02 | IGMP | Group Management | RFC 1112 |
| 0x04 | IP | IP in IP | RFC 2003 |
| 0x06 | TCP | TCP | RFC 793 |
| 0x11 | UDP | UDP | RFC 768 |
| 0xFD | | | experimental |
| 0xFE | | | experimental |

오버레이 네트워크에서 프록시 노드가 처리해야하는 데이터 패킷의 포맷은 표 1.5와 같다. Overlay protocol header는 총 20바이트로써 IP header와 동일한 구조를 가지며 하위 protocol data unit을 캡슐화하고 있다. 즉, 기존 IP 패킷을 IP header와 동일한 구조를 갖는 Overlay protocol header가 IP-in-IP 방식으로 캡슐화하고 있다. 프로토콜 번호 253 또는 254는 Overlay protocol header에 설정되는 필드값이다.

### 1.4.3 Module Header & Flow Table

모듈헤더는 NetFPGA 내부 모듈 간에 패킷 데이터를 주고 받기위해 필요한 정보를 나타낸다. 모듈헤더는 MAC 프레임이 입력(수신)되었을 때, 2계층 MAC 정보 및 3계층 IP 정보를 순차적으로 읽어 구성된다. 모듈헤더가 포함하는 정보는 표1.6과 같다. 아래 첨자 $_{op}$는 overlay protocol header, $_{ip}$는 IP header를 의

Table 1.5: Packet Format.

| Header | Size(bytes) |
|---|---|
| Overlay protocol header | 20 |
| IP header | 20 |
| UDP or TCP header | |
| Payload | variable |

미하고 모듈헤더를 나타내기 위해 위첨자 $^{mo}$를 이용한다. 예를 들어, $Src_{op}^{mo}$는 모듈헤더가 가지고 있는 overlay protocol header의 source address를 의미한다. 모듈헤더 $Src_{op}^{mo}$ 및 $Des_{op}^{mo}$의 초기값은 0.0.0.0이며 IP 헤더의 프로토콜 번호가 253일 경우에만 두 값을 읽는다.

Table 1.6: Module Header.

| abbreviation | keyword |
|---|---|
| $Src_{mac}$ | MAC Source address |
| $Des_{mac}$ | MAC destination address |
| $Des_{op}$ | Destination address |
| $Ptl_{op}$ | Protocol number |
| $Src_{ip}$ | Source address |
| $Des_{ip}$ | Destination address |
| $Ptl_{ip}$ | Protocol number |
| $sPort_{ip}$ | Source port |
| $dPort_{ip}$ | Destination port |

Flow table의 형태는 다음 표 1.7과 같이 Openflow 스위치 [8]에서 규정하고 있는 flow table과 유사하다. Flow table은 크게 플로우 여과 규칙(filtering rule)과 여과된 플로우에 대해 프록시 노드가 취해야 할 행동 지침(action) 및 여과된 플로우에 대한 패킷 통계(statistics)로 구분된다. 패킷 통계는 노드별 정보와 플로우별(또는 그룹별) 정보로 구분된다. 노드별 정보는 프록시에서 처리한 총 바이트, 총 패킷수, 총 플로우(또는 그룹) 수를 기록하며, 플루우별 정보는 해당 플로우에 대해 프록시 노드가 처리한 총 바이트, 총 패킷수 및 활성화 시간을 기록한다.

Table 1.7: Flow table.

| filtering rules | action | statistics |
|---|---|---|

다음 그림 1.5는 flow table이 업데이트 되는 절차의 예를 보여준다. 프록시 노드는 IP 패킷을 수신하면 플로우 테이블에 해당 엔트리가 존재하는지 확인한다. 엔트리의 존재 확인을 쉽게하고 추후 확장을 고려해 TCAM(Ternary content addressable memory)을 이용하며, $(Src_{ip}, Des_{ip})$을 갖는 2-tuple이 여과 규칙으로 사용된다. 플로우 테이블에 해당 엔트리(즉, $(Src_{ip}, Des_{ip}) == (Src^{fl}, Grp^{fl})$)가 존재할 경우, `action` 필드의 규정에 따라 패킷을 처리한다. 플로우 엔트리가 존재하지 않을 경우에는 수신된 패킷의 헤더부분(MAC 헤더 포함)을 Coordinator에게 전달하며 Coordinator는 내부 규정에 기반해 `filtering rule`과 `action`를 결정하고 프록시 노드에게 전달한다. 프록시 노드는 전달받은 플로우 엔트리를 플로우 테이블에 추가하고, 이 후에 동일한 $(Src_{ip}, Des_{ip})$를

갖는 패킷 플로우가 수신되면 해당 `action`을 수행한다.

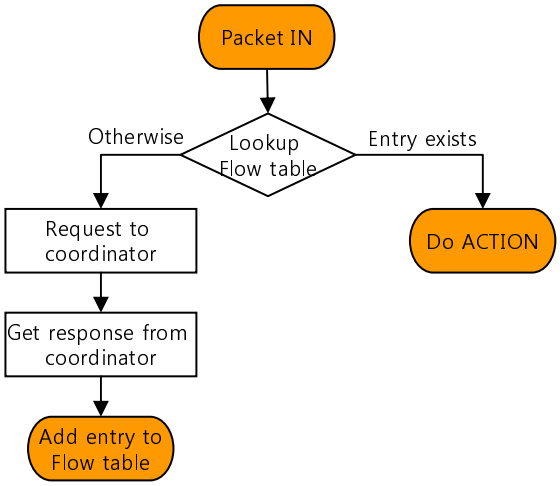


Figure 1.5: Flow table update.

`action`은 가변크기(variable size)를 갖는 필드로써, `action`이 `normal`일 경우에 {ACT=`normal`, $SRC_{op}$=192.168.10.10, $DES_{op}$=192.168.12.10}와 같이 표기될 수 있다. 중앙관리식 및 사용자제어식에서는 `action` 필드가 Coordinator에 의해 정의되며, 자가제어식의 경우는, 예를 들어, {ACT=`normal`}만 Coordinator가 정의하고 나머지 필드는 포워딩 테이블(forwarding table)을 이용해 프록시 노드에 의해 자가 정의된다. 현재 사용가능한 `action`은 `discard`와 `normal`이다. `statistics`는 해당 플로우 엔트리에 대해 프록시 노드가 처리한 총 바이트 수, 패킷 수 및 활성화 시간(timer)을 갖는다. 활성화 시간은 해당 플로우를 마지막으로 처리한 시간을 나타내며, 멀티캐스트 세션의 소멸(expire)을 확인하기 위해서 사용된다.

정리하자면, flow table은 고속 패킷 포워딩을 목적으로 NetFPGA에 구현된 `Arbiter`가 참조하며 {filtering rules, action, statistics}로 구분된다. `filtering rule`은 ($Src^{fl}$(source address), $Grp^{fl}$(multicast group address)), `action`은 ($Act^{fl}$(action), $Cnt^{fl}$(current node address), $Nxt^{fl}$(next hop address))와 같이 구성될 수 있으며, `statistics`는 ($Bytes^{fl}$(total bytes), $Packets^{fl}$(total number of packets), $Timer^{fl}$)로 구성된다[1].

### 1.4.4 Frame and Packet Classification

프록시 노드는 2계층 투명브리징(transparent bridging)과 3계층 IP 서비스를 동시에 수행하기 때문에 MAC 프레임의 Ethernet type 및 IP 패킷의 규약번호(protocol number)에 따라 처리 절차가 구분된다.

#### Transparent Bridging

2계층 투명브리징은 IP 및 ARP 패킷을 예외로 한다. 즉, 프록시 노드는 IP와 ARP 이외의 패킷들에 대해서 입출력 버퍼를 갖는 dummy hub와 같은 역할

[1]flow table의 field 내용 중 MAC 및 port 정보에 대한 부분은 현재 구현된 버전에 포함되지 않는다. 보다 세밀한 플로우의 제어를 위해서는 해당 필드들이 수용되어야 한다.

을 수행한다. 2계층에서 멀티캐스트 패킷은 플러딩(flooding)되며 MAC 테이블은 aging(특정시간마다 MAC entry를 refresh하는 작업)을 통해 유지된다. 그림1.6는 프록시 노드에서 MAC 프레임이 수신되었을 때 해당 프레임이 처리되는 절차를 보여준다.

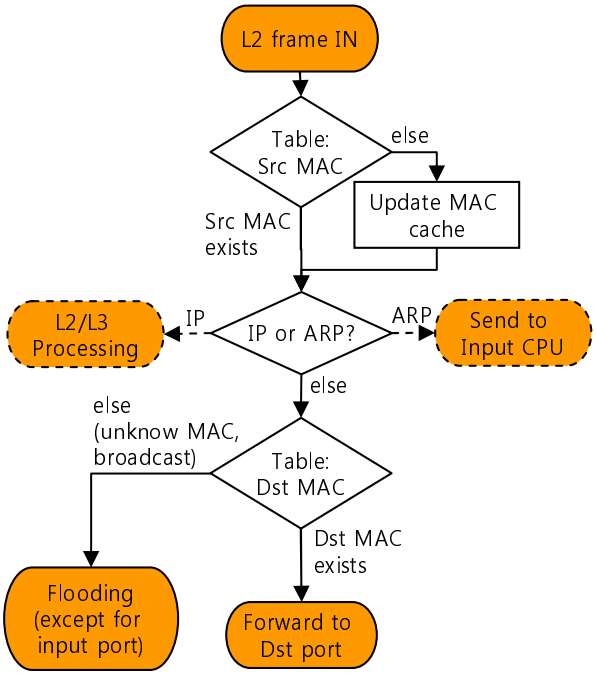


Figure 1.6: Transparent bridging in proxy node.

MAC 프레임이 수신되면 프록시 노드는 먼저 MAC 테이블을 검사한다. MAC 프레임을 송신한 시스템의 MAC 주소(`Src MAC`)가 프록시 노드의 MAC 테이블에 존재하면 프레임의 `Ethernet Type` 필드를 읽어 프레임의 종류를 파악한다. MAC 테이블에 수신받은 프레임의 MAC 정보가 존재하지 않을 경우, 프록시 노드는 호스트 MAC과 입력 포트를 MAC 테이블에 추가한다. 해당 프레임의 `Ethernet Type`이 ARP나 IP이면, 프레임을 프록시 노드의 하드웨어 영역(NetFPGA)에서 CPU 영역으로 이동시켜 3계층 처리를 수행한다. ARP나 IP 패킷이 아닐 경우 MAC 프레임의 목적지 주소(`Des MAC`)를 읽고 목적지 주소가 프록시 노드의 MAC 테이블에 존재할 경우 해당 포트를 통해 송신한다. 목적지 주소가 MAC 테이블에 존재하지 않으면 입력포트를 제외한 나머지 포트로 플러딩한다.

### ARP Processing

그림 1.7는 ARP 패킷을 3계층에서 처리하는 절차를 보여준다. ARP 패킷의 `Ethernet Type`은 `0x0806`이며 프록시 노드에 의해 3계층에서 처리된다. 임의의 호스트(프록시 노드가 활성 송신자 -active sender-일 경우 포함)는 자신의 MAC 테이블에 목적지 IP에 대한 MAC 주소를 가지고 있지 않을 경우 ARP `request` 패킷을 발생시키며, 수신받은 ARP `request` 패킷의 IP 주소가 자신의 주소와 동일한 호스트는 ARP `reply` 패킷을 피드백(feedback)한다. 그림 1.7에

서는 해당 프록시 노드가 활성 송신자로써 ARP `request` 패킷을 발생시키는 과정은 생략되어 있다(즉, 타 호스트가 발생시킨 ARP `request`의 처리 과정만 포함하고 있다.).

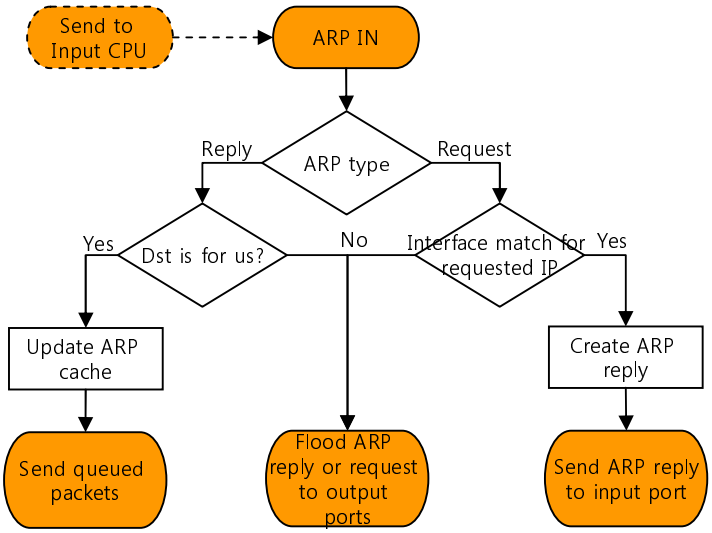


Figure 1.7: ARP processing in Layer 3.

ARP 패킷의 `operation` 필드를 통해 ARP 타입이 구분된다. 임의의 프록시 노드는 자신이 발생시킨 ARP `request` 패킷에 대한 ARP `reply` 패킷을 수신했을 경우에 ARP 테이블을 갱신하고 `reply` 패킷이 보고한 MAC 주소를 이용해 큐에 있는 활성 패킷을 전송한다. ARP `request` 패킷을 수신했을 경우에는, ARP 패킷에 의해 요청된 IP의 MAC 주소(Target IP 주소)가 자신의 인터페이스 주소와 일치하는지 확인한다. 참고로, ARP 패킷의 `Ethernet ARP` 필드는 송신자 {MAC 주소, 송신자 IP 주소, Target MAC 주소, Target IP 주소}를 포함한다. 일치할 경우, ARP `reply` 패킷을 발생시켜 해당 ARP `request` 패킷이 수신된 포트로 송신한다.

### IP processing

프록시 노드에서 처리하는 IP 패킷은 IGMP(Internet group management protocol), TCP 및 UDP 등 3가지 종류이다. IP 헤더의 규약번호(`Protocol number`)를 통해 구분한다.

그림 1.8는 프록시 노드가 IGMP 및 TCP/UDP 패킷을 처리하는 과정을 보여준다. 프록시 노드가 2계층 MAC 프레임을 수신하면 목적지 MAC 주소를 통해 통신방식(멀티캐스트, 유니캐스트 또는 브로드캐스트)이 결정된다. 브로드캐스트용 MAC 프레임이 도착하면 2계층에서 플러딩을 수행하고, 유니캐스트 프레임에 대해서는 목적지 주소에 따라 처리 방식이 달라진다. 목적지 MAC 주소가 프록시 노드가 아닐 경우에는 2계층에서 플러딩 또는 선택적 포워딩이 행해지며, 프록시 노드일 때는 IP 패킷 헤더의 규약번호에 따라 처리된다. 즉, 규약번호가 253(프록시 노드 간 패킷 전송)일 때는 플로우 테이블을 검색해 해당 엔트리가 존재하면 `action`을 수행하고, 존재하지 않으면 역캡슐화하거나 해당 프레임을 버린다(discard). 멀티캐스트용 MAC 주소를 갖는 프레임일 경우에

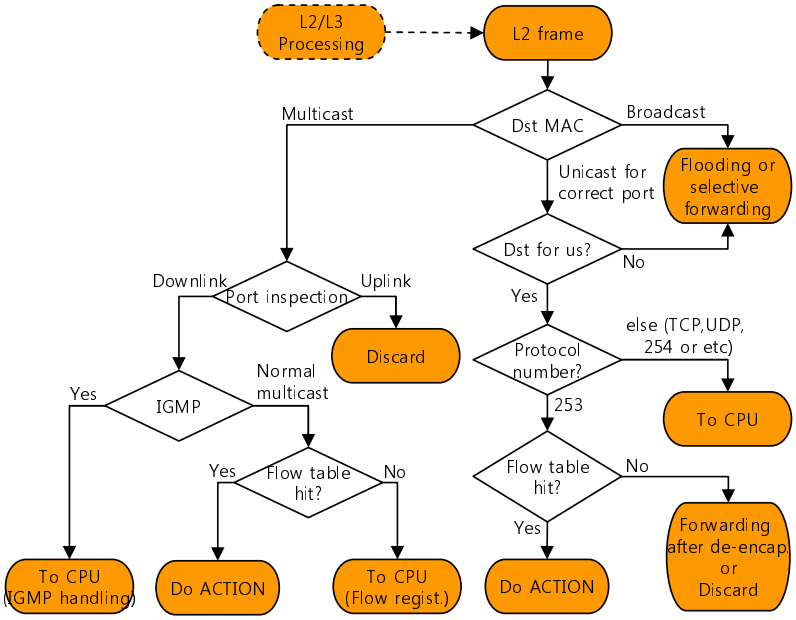


Figure 1.8: IP processing in Layer 2.

는 업링크와 다운링크를 구분해 처리한다[2]. 이는 종단 호스트와 연동되는 첫번째 라우터(first-hop router)가 발생시키는 IGMP query 메시지를 차단하기 위해서이다(프록시 노드도 IGMP 서비스를 제공하기 때문에 충돌이 발생할 수 있다.). 업링크 포트를 통해 수신받은 멀티캐스트 패킷은 버려진다. 다운링크 포트를 통해 수신된 IGMP(멀티캐스트) 패킷은 소프트웨어 영역으로 이동된 후 서비스 데몬이 처리하며, 일반 멀티캐스트 패킷을 수신받았을 경우에는 플로우 테이블을 검색한다. 플로우 테이블에서 해당 엔트리가 검색되면 정의된 `action`에 따라 패킷이 처리되고 존재하지 않을 경우에는 `Overlay engine`에게 이동시켜 플로우 엔트리를 정의한다.

### 1.4.5 Packet Processing in NetFPGA

그림 1.9은 그림 1.8의 일부분으로 일반 멀티캐스트 패킷에 대한 처리 절차를 구체적으로 보여준다. 다운링크 포트를 통해 수신된 일반 멀티캐스트 프레임에 대해서 프록시 노드는 해당 프레임의 모듈헤더를 구성하기 위해서 2계층/3계층 헤더 정보를 수집하고 수집된 정보를 바탕으로 프레임을 처리한다. 먼저 IP 헤더의 규약번호를 읽어(253일 경우에 IP-in-IP 캡슐화 된다.) $Des_{ip}$의 위치를 파악한 후, $Des_{ip}$가 Class D 멀티캐스트 주소인지를 확인한다. 해당 멀티캐스트 프레임을 종단 호스트가 송신했을 경우에는 오버레이 규약 헤더(overlay protocol header) 정보를 가지고 있지 않기 때문에, 즉, IP-in-IP 캡슐화되지 않은 프레임을 수신했기 때문에 프록시 노드는 $Src_{op}^{mo}$ 및 $Des_{op}^{mo}$를 0.0.0.0으로 설정한다. IP 헤더 또는 오버레이 규약 헤더의 규약번호를 읽어 해당 패킷이 종단 호스트나 임의의 시스템으로부터 수신받은 것인지 또는 인접 프록시 노드

[2]multicast-unicast 브리징을 위해서는 업링크를 통해 수신되는 멀티캐스트 패킷이 프록시 노드에 의해 추가적으로 처리되어야 한다.

로부터 수신받은 것인지를 판단할 수 있다.

MAC 프레임의 목적지 주소가 멀티캐스트이면(즉, `01-00-5E-`), ($Src_{ip}^{mo}$, $Des_{ip}^{mo}$)를 `filtering rules`로 갖는 엔트리를 플로우 테이블에서 검색한다. 검색된 엔트리의 `action`에 따라 해당 프레임을 처리한다. 검색된 엔트리가 0일 경우에는 Coordinator에게 플로우 엔트리 등록(`register`)을 요청한다. 등록 방식은 앞서 설명한 중앙관리식, 자가제어식, 또는 사용자제어식에 따른다.

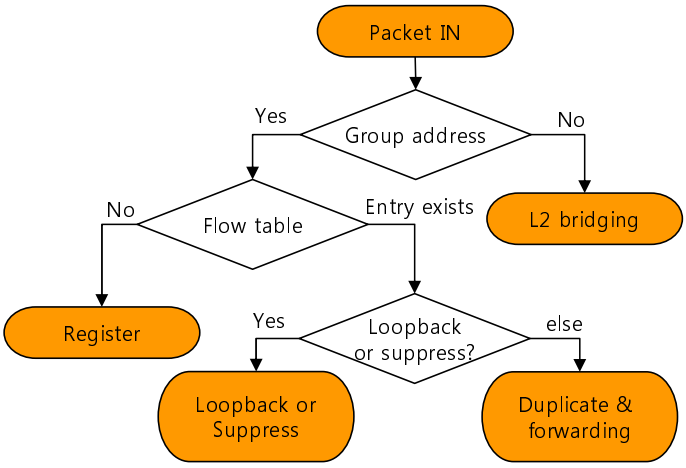


Figure 1.9: Proxy gets a frame or packet.

임의의 멀티캐스트 세션에 참가한 종단 호스트가 동시에 멀티캐스트 패킷을 송신할 경우, 해당 송신 패킷은 Rendezvous 지점까지 트래픽이 흐르는 것을 막기위해 오버레이 네트워크를 통해 이동되지 않고 진출·입 노드(ingress·egress proxy node)에서 루프백(loopback)되어야 한다. 루프백 여부를 판단하기 위해서 프록시 노드는 ($Src_{op}^{mo}$가 $Cnt^{fl}$)와 동일한지를 검사한다(플로우 엔트리가 이미 존재했을 때를 가정한다.).

본 문서에서 `suppression`은, 멀티캐스트 트래픽이 Rendezvous 지점과 임의의 프록시 노드간에 루프되는 것을 방지하기 위해서 해당 패킷을 필터링하는 과정을 의미한다. 그림 1.10과 같이 `suppression`이 없을 경우 멀티캐스트 트래픽이 Rendezvous 지점(`N2`)과 프록시 노드(`N5`) 사이에서 루프되지만(붉은 점선), `suppression`되면 Rendezvous 지점을 거치지 않고 프록시 노드 `N5`에서 `N7`으로 직접 전달된다(파란 실선). 참고로, 호스트가 특정 멀티캐스트 그룹에 참가(`join`)하면 진출·입 노드의 플로우 엔트리는 ($Cnt^{fl}$,$Nxt^{fl}$)쌍을 (0.0.0.0, 0.0.0.0)로 갖는다.

다음 유사코드(pseudo code)는 프록시 노드가 루프백과 `suppression`을 처리하는 방법을 보여준다.

**if** $Src_{op}^{mo} == 0.0.0.0$ **then**
  **if** $Cnt^{fl} == Src_{op}^{mo}$ **then**
    loop back
  **end if**
**else**
  **if** $Cnt^{fl} != 0.0.0.0$ **then**
    **if** $(Src_{op}^{mo} == Nxt^{fl})\&\&(Des_{op}^{mo} == Cnt^{fl})$ **then**
      suppression

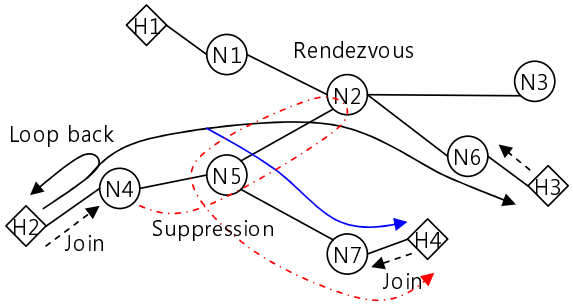


Figure 1.10: Suppression and loop back.

**end if**
**end if**
**end if**

추가적으로 프록시 노드들 간의 패킷 송·수신은 IP-in-IP 캡슐화를 이용하기 때문에 진출·입 노드는 멀티캐스트 패킷을 캡슐화 또는 역캡슐화하는 기능이 필요하다. 위 유사코드를 확장해 프록시 노드가 패킷을 처리하는 전체적인 방법을 정의한다.

**if** $Src_{op}^{mo} == 0.0.0.0$ **then**
  **if** $Cur^{fl} == Src_{op}^{mo}$ **then**
    loop back to downlinks
  **else**
    encapsulation, duplication and forwarding
  **end if**
**else**
  **if** $Cnt^{fl} != 0.0.0.0$ **then**
    **if** $(Src_{op}^{mo} == Nxt^{fl})\&\&(Des_{op}^{mo} == Cnt^{fl})$ **then**
      suppression
    **else**
      duplication and forwarding
    **end if**
  **else**
    de-encapsulation and multicast to downlinks
  **end if**
**end if**

### 1.4.6 Example Scenario

다음 그림1.11는 중앙관리서버(CN)에서 라우팅 경로설정과 관련된 태스크(task)를 수행한다고 가정했을 때 종단 호스트에서의 멀티캐스트 join 절차와 경로설정 과정을 보여준다. 이 때 각 proxy 노드에서 갖는 flow table의 설정값을 살펴본다.

호스트 H2가 IGMPv2를 이용해 특정 멀티캐스트 그룹에 join하면 proxy node N5는 H2에 대한 flow table을 $\{group\ addr., 0.0.0.0, 0.0.0.0, expires\}$로 설정한다. 앞서 설명되었듯이 0.0.0.0은 local loopback을 위해서 사용되는 IP 주소이다. Flow table의 Action 필드는 생략하고 설명한다. expires는 멀티캐

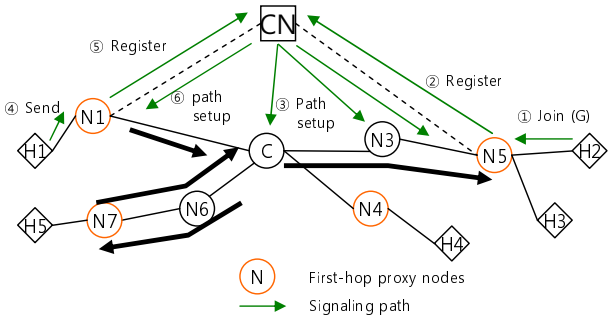


Figure 1.11: Example Scenario #1.

스트 세션의 active 상태 여부를 판단하며 설정된 값을 초과하는 시간동안 해당 멀티캐스트 그룹패킷을 받지 못하면 flow table에서 해당 entry를 삭제한다. 즉, proxy node에서 묵시적인 multicast 세션 탈퇴(leave)를 위해서 활용된다. N5는 중앙관리서버에게 호스트 H2의 join을 알리며 중앙관리서버는 선택된 Rendezvous(또는 core, C)로 부터 요청을 한 N5까지의 경로를 설정할 수 있도록 Rendezvous C에게 경로 설정을 명령한다. 경로설정과 관련해서 1) 중앙관리서버에서 경로 설정을 완료한 후 경로상에 있는 모든 proxy node들에게 해당 flow table을 전달하는 방법과 2) Rendezvous에게 first-hop proxy (예, N5)의 주소를 주고 경로상의 proxy들이 자체 라우팅 프로토콜을 이용해 flow table을 작성하는 분산형 방법이 있다. 그림1.11과의 일관성을 유지하기 위해 전자의 경우를 가정한다.

$\{C, N3, N5\}$ 까지의 경로가 설정되었다고 가정하면 C는 $\{group\ addr., C, N3, expires\}$, N3는 $\{group\ addr., N3, N5, expires\}$, N5는 $\{group\ addr., 0.0.0.0, 0.0.0.0, expires\}$을 flow table로 갖게 된다. 예를 들어, $\{group\ addr., C, N3, expires\}$는 `group addr.`를 갖는 packet이 proxy C에 도착하면 proxy N3로 전송하라는 의미이다. 동일한 방법으로, 호스트 H5가 멀티캐스트 그룹에 join하면 Rendezvous C는 $\{group\ addr., C, N6, expires\}$, N6는 $\{group\ addr., N6, N7, expires\}$, N7은 $\{group\ addr., 0.0.0.0, 0.0.0.0, expires\}$을 flow table로 설정한다.

멀티캐스트 패킷의 송신과 관련된 설명이 이어진다. 호스트 H1이 멀티캐스트 그룹으로 패킷을 송신한다고 가정하면 proxy $\{N1, C\}$ 사이에 라우팅 경로가 설정된다(중앙관리식 또는 분산형). proxy N1은 멀티캐스트 패킷을 받게 되면 중앙관리서버 CN에 접속해 Rendezvous C에 관련된 정보를 얻어온 후 라우팅 경로를 계산하고 flow table을 설정한다. 라우팅 경로는 중앙관리형 또는 분산형으로 계산한다. 이때, N1에 설정된 flow table entry는 $\{group\ addr., N1, C\}$이다.

IGMP를 이용한 멀티캐스트 join/leave는 multicast packet의 수신과 관련된 문제이며 active한 멀티캐스트 패킷을 송신하는 과정은 멀티캐스트 join을 필요로 하지 않는다. 예를 들어, 이미 멀티캐스트 그룹에 join하고 있는 호스트 H5가 멀티캐스트 패킷을 송신하고자 한다면 경로 $\{N7, N6, C\}$가 설정되어야 하며 join과정에서 만들어진 flow table과 구분되어져야 한다. 멀티캐스트 패킷을 송신할 때, proxy N7은 $\{group\ addr., N7, N6, expires\}$, N6는 $\{group\ addr., N6, C, expires\}$를 flow table entry로 갖는다. 멀티캐스트 join과정에서 만들어진 N7의 flow table entry는 $\{group\ addr., 0.0.0.0, 0.0.0.0, expires\}$이다.

정리하면, H1, H2, H5가 멀티캐스트 그룹에 join하고, H1과 H5가 멀티캐스트 패킷을 송신한다고 가정했을 때 네트워크 상의 proxy node들이 가지는 flow

table entry는 다음 표1.8과 같다.

Table 1.8: Flow Table Entries (example scenario #1).

| Node | Group | Cur. hop | Nxt. hop | Expires |
|---|---|---|---|---|
| C | Group addr. | C | N3 | expires |
| | Group addr. | C | N6 | expires |
| N7 | Group addr. | N7 | N6 | expires |
| | Group addr. | 0.0.0.0 | 0.0.0.0 | expires |
| N6 | Group addr. | N6 | N7 | expires |
| | Group addr. | N6 | C | expires |
| N5 | Group addr. | 0.0.0.0 | 0.0.0.0 | expires |

Suppression을 설명하기 위해 그림1.11을 확장한다. 그림1.12는 그림1.11과 비교해 하나의 proxy node N8과 호스트 H6를 추가적으로 갖는다.

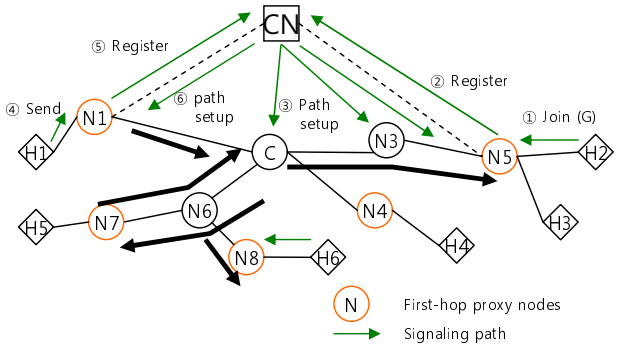


Figure 1.12: Example Scenario #2.

호스트 H5가 멀티캐스트 패킷을 송신한다고 가정한다. proxy node N6가 suppression을 수행하지 않는다면 호스트 H6는 $\{N7, N6, C, N6, N8\}$의 경로를 통해 호스트 H5가 송신한 패킷을 수신하게 된다. 이와 같은 경로 설정은 불필요한 네트워크 대역폭 소모 및 Rendezvous의 부하 증가의 문제점을 발생시키며 proxy node에서의 패킷 forwarding 규약에 위반된다. Suppression은 경로 $\{N7, N6, N8\}$를 통해 호스트 H6가 H5의 패킷을 수신하게 하고 그룹 내 기타 호스트들이 H5가 송신한 멀티캐스트 패킷들을 수신할 수 있도록 경로 $\{N7, N6, C\}$로 트래픽을 분기해 준다.

마지막으로, 패킷 수준에서 처리 절차를 살펴본다.

### Packet Processing in Proxy N7

호스트 H5가 멀티캐스트 패킷을 송신하면 proxy node N7은 그림 1.13와 같이 IP-in-IP encapsulation되지 않은 패킷을 수신하게 된다. 이하 IP 패킷의 형태는 전체 header 필드 중 주소 영역만을 표기한다.

proxy node N7은 다음 그림1.14과 같은 flow table을 가지고 있다. Group addr.에 대하여 N6로 전송하거나 downlink로 loopback한다. flow table(그림1.14)에 기초해, N6로 전송하기 위해서는 IP-in-IP encapsulation을 수행해 그림1.15과 같은 패킷을 생성한다.

| H5 | Group addr. |
|---|---|
| UDP/TCP header | |
| Payload | |

Figure 1.13: Packet sent by *H5*.

| Group addr. | N7 | N6 | expires |
|---|---|---|---|
| Group addr. | 0.0.0.0 | 0.0.0.0 | expires |

Figure 1.14: Flow table at proxy node *N7*.

또한, loopback 시키기 위해서 그림 1.16와 같이 수신받은 패킷을 수정하지 않고 downlink 방향으로 멀티캐스팅한다.

### Packet Processing in Proxy `N6`

Proxy node `N7`이 IP-in-IP encapsulation을 통해 전송한 패킷이 proxy `N6`에 도착했을 때 `N6`의 flow table을 이용해 패킷 처리하는 과정을 설명한다.

proxy node `N6`에 도착한 패킷의 형태와 `N6`의 flow table은 다음 그림1.17과 같다. $\{group\ addr., N6, N7, expires\}$은 패킷 처리 규약에 의해 suppression된다.

수신받은 패킷의 overlay protocol header는 flow table entry 값에 따라 변경된다. Entry $\{group\ addr., N6, N7, expires\}$는 `N7`로 부터 송신된 패킷의 overlay protocol header $(N7, N6)$를 $(N6, C)$로 수정해 Rendezvous `C`로 송신한다. 또한, $(N6, N8)$으로 수정해 `N8`로 송신한다. 다음 그림1.18는 수정된 패킷의 형태이다.

### Packet Processing in Rendezvous `C`

Proxy node `N6`가 송신한 패킷이 Rendezvous `C`에 도착하면 Rendezvous는 flow table의 entry 중 $\{group\ addr., C, N6, expires\}$를 패킷 처리 규약에 의해 suppression한다. Suppression된 flow entry를 제외하고 Rendezvous `C`는 수신받은 패킷의 overlay protocol header를 $(C, N3)$로 수정해 `N3`에게 전달한다. Rendezvous `C`의 flow table과 수정된 패킷의 형태는 다음 그림1.19와 같다.

| N7 | N6 |
|---|---|
| H5 | Group addr. |
| UDP/TCP header | |
| Payload | |

Figure 1.15: Outgoing packet format to *N6*.

| H5 | Group addr. |
|---|---|
| UDP/TCP header | |
| Payload | |

Figure 1.16: Packet looped back to *H5*.

| N7 | N6 |
|---|---|
| H5 | Group addr. |
| UDP/TCP header | |
| Payload | |

| Group addr. | N6 | N7 | expires |
|---|---|---|---|
| Group addr. | N6 | C | expires |
| Group addr. | N6 | N8 | expires |

Figure 1.17: Incoming packet and flow table at `N6`.

## 1.5 Discussion

### 1.5.1 Connection Release

호스트가 IGMP leave 메시지를 보내거나 오류로 인해 멀티캐스트 세션이 해제되었을 경우에 각 proxy node들은 flow table에서 해당 entry를 삭제해야 한다. 또한, 멀티캐스트 패킷 송신에 의해서 경로 설정된 table entry도 삭제되어야 한다.

IGMP leave 메시지에 의해 명시적으로 table entry를 삭제하는 방법으로는 1) 역경로 추적(back tracking) 해제, 2) 관리서버 기반 해제 및 3) 복합형(hybrid) 해제 등이 있을 수 있다. 패킷 송신에 의해 설정된 entry는 `expires` 타이머에 의해 해제된다. 먼저, 역경로 추적에 의한 방법은 proxy 노드가 flow table에 존재하지 않는 멀티캐스트 주소를 갖는 패킷을 수신할 경우에 송신 노드에게 prune message를 보내고, prune message를 수신한 proxy node는 flow table에서 해당 entry를 삭제한다. 두번째, 관리서버 기반 해제법은 종단 호스트가 IGMP 탈퇴 메시지를 명시적으로 발생시키는 경우, 메시지를 받은 proxy node는 해당 사항을 관리서버에게 알리고 관리서버가 경로상에 있는 모든 proxy node들에게 prune message를 보내는 방법이다. 관리서버가 모든 멀티캐스트 flow에 대한 end-to-end path 정보를 가지고 있어야 하는 문제가 있다. 복합형 해제법은

| N6 | C |
|---|---|
| H5 | Group addr. |
| UDP/TCP header | |
| Payload | |

| N6 | N8 |
|---|---|
| H5 | Group addr. |
| UDP/TCP header | |
| Payload | |

Figure 1.18: Outgoing packet format at `N6`.

| Group addr. | C | N3 | expires |
|---|---|---|---|
| Group addr. | C | N6 | expires |

| C | N3 |
|---|---|
| H5 | Group addr. |
| UDP/TCP header | |
| Payload | |

Figure 1.19: Flow table and outgoing packet at Rendezvous C.

송신 호스트부터 Rendezvous까지는 관리서버가 해제하고(또는, 송신 호스트에 연동된 first-hop proxy가 prune message를 Rendezvous까지 보내는 방법도 있다.) Rendezvous 부터 수신 호스트까지는 역경로 추적을 통해 해제하는 방식이다.

# Bibliography


[1] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O. Jr, "Overcast: reliable multicasting with an overlay network," in *Proc. of the 4th conference on Symposium on Operating System Design & Implementation*, vol. 4, 2000.

[2] D. G. Andersen, N. Feamster, S. Bauer, and H. Balakrishana, "Resilient overlay networks," in *Proc. of 18th ACM SOSP*, Oct. 2001.

[3] Y. Chawathe, "Scattercast: An architecture for Internet broadcast distribution as an infrastructure service," *Ph.D. thesis, Dept. of EECS, UC Berkeley*, Dec. 2000.

[4] P. Francis, "Yoid: Extending the Internet multicast architecture," *white paper http://www.aciri.org/yoid/*, 2000.

[5] K. Lakshminarayanan, A. Rao, I. Stoica, and S. Shenker, "End-host controlled multicast routing," *Elsevier Computer Networks, Special Issue on Overlay Distribution Structures and their Applications*, 2005.

[6] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An application level multicast infrastructure," in *Proc. of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*, vol. 3, 2001.

[7] G. Watson, N. McKeown, and M. Casado, "NetFPGA: A tool for network reseach and education," in *Proc. of the 2nd workshop on Architectural Research using FPGA platforms*, 2006.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus network," *ACM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.