ISBN 978-89-6211-378-5 98500

기술문서

Spanning Tree기반의 그룹 관리, Flow table 정의 및 패킷 포워딩 방법 연구

- IP 멀티캐스트를 위한 오버레이 프록시의 설계 v2.0

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

Copyright

Please do not take it without our permission. All the materials produced here may not be copied, reprinted, published, translated, hosted or otherwise distributed by any means without our written permission.

Contents

Chapter 1

Overview

1.1 Introduction

본 연구는 IP 멀티캐스트를 대행할 오버레이 네트워크(Overlay network)의 설계 및 구현을 목적으로 한다. 점대다(point-to-multipoint) 전송을 위한 고속 패킷 포워딩(high-speed packet forwarding) 등 데이터 평면(data plane)을 NetFPGA 상에 구현하는데 단기 목적이 있으며, Service-enabling Overlay(SeeOver) 인프라를 구축 서비스하는 것을 최종 목적으로 한다. 본 문서는 데이터 평면의 NetFPGA 구현을 위한 세부 설계를 담고있다. SeeOver를 통해 접속망(Access network)에서 연동된 종단 호스트들은 다자간 협업에 참가할 수 있다.

1.1.1 Problem Definition & Resolution

IP multicast는 수익모델의 부재, AS간 정책적 이견 및 유지 관리의 어려움 등으로 기술적 장점에도 불구하고 구축 서비스가 제한적으로 진행되어 왔다. 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, 응용 서비스의 저품질화와 같은 다양한 문제들을 극복하고 다자간 망기반 협업이 가능하도록 안정적인 IP 멀티캐스팅 인프라를 제공할 수 있는 기술적 접근이 요구된다.

IP 멀티캐스트의 비신뢰적 전송 등 품질 문제와 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, 패킷 중계를 담당하는 호스트들의 잦은 멀티캐스트 참가 및 탈퇴(join and leave)로 인해 트리 구성이 안정적이지 못한 단점이 있다. 또한, 오버레이 인프라에 참가하는 호스트 간 메시지 교환을 위해 비표준 통신 규약이나 인프라 종속적인 패킷 헤더 등을 사용해야 하므로 종단 호스트의 응용소프트웨어 또는 시스템 설정의 변경을 필요로 한다. 일례로, 오버레이 프록시(수퍼노드)를 사용하는 ScatterCast는 평균 350Mbps의 패킷 처리율을 보이며 패킷당 처리지연이 200∼600μs (최대 2∼3 ms) [3]로써 다수의 프록시를 통해 라우팅 될 때 높은 지연변이(Jitter)가 발생하는 등 서비스 품질의 저하를 예상할 수 있다.

본 연구를 다음과 같은 기술적 난제를 해결함으로써 종단 사용자에게 안정적인 멀티캐스트 환경과 다자간 협업을 위한 비용 효율적인 네트워킹 서비스

를 제공한다.

1. Ease-of-use: IP 멀티캐스트를 대체하는 기존 오버레이 네트워킹 기술들은 종단 호스트의 설정수정 또는 비표준 통신 규약 등을 이용해야 한다. 또한, 종단 호스트들이 패킷 전달에 직접 참가하기 때문에(참가 및 탈퇴가 자유롭기 때문에) 안정적인 트리(tree)를 제공하지 못했다. SeeOver는 수퍼노드인 오버레이 프록시(Proxy)를 이용하기 때문에 안정적인 트리 제공이 가능하며 IGMP(Internet group management protocol) 등 기존 IP 프로토콜 규약을 준수함으로써 종단 호스트는 시스템 설정의 변경없이 다자간 협업에 참가할 수 있다.

2. Controllability: 기존 IP 멀티캐스트는 호스트의 그룹 참가가 자유롭기 때문에 악성 사용자(malicious users)로 부터 해당 그룹을 보호할 수 없었다(ex, high-volume packet flooding). 본 연구에서는 멀티캐스트 패킷을 플로우테이블(Flow table)에 기반해 포워딩하게 함으로써 각 플로우에 대한 관리성을 높인다. 또한, 멀티캐스트 패킷에 대한 필터링(filtering) 규칙 및 처리 지침(action) 등을 관리서버가 결정하게 함으로써 플로우 처리에 대한 다양한 설정의 조합이 가능한다.

3. Cost-effectiveness: 소프트웨어에 의한 점대다 포워딩은 고성능·고가의 워크스테이션(Workstation)을 활용하지만 패킷 처리율 등에서 낮은 성능을 보인다. 본 연구에서는 PCI(Peripheral component interconnect) 카드인 NetFPGA를 활용해 고속 패킷 포워딩 및 다지점 복제를 수행하고 제어평면과 데이터평면을 분리함으로써 Gbps급 패킷 처리 성능을 보장한다. 또한, NetFPGA 카드가 내장된 시스템의 가격이 $1,500 수준의 저가이므로 비용효율적 프록시의 구축이 가능하며 점진적 배치를 통해 확장성과 안정성을 높일 수 있다.

1.2 Background

1.2.1 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.

참조설계에서 제공하는 참조파이프라인(Reference pipeline)의 데이터경로는 그림 1.1과 같이 입력중재기(Input arbiter), 출력포트 검색(Output port lookup), 출력 큐(Output queues)로 구성되며 입력중재기는 처리되어야 할 패킷을 담고있는 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 소프트웨어 영역에 구현함으로써 완성된다.

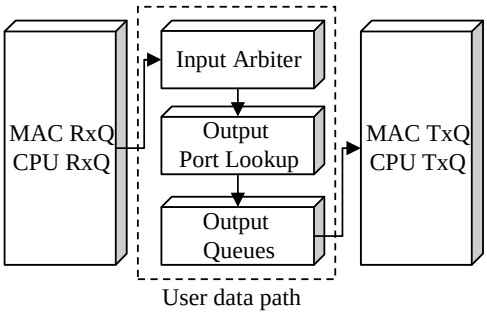

Figure 1.1: Reference pipeline.

1.2.2 Ethertype & Multicast in Layer 2

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.

Table 1.1: Ethernet Type.

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

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 1.2: 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) ARP reply(2), RARP request(3), RARP reply(4)

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, join을 위해서 224.0.0.1(01-00-5e-00-00-01), leave는 224.0.0.2(01-00-5e-00-00-02), report는 multicast group address를 사용한다.

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

IP 멀티캐스트 주소를 IEEE 802 MAC 주소에 사영(mapping)하기 위해서 2계층 프레임은 그림 1.2와 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.

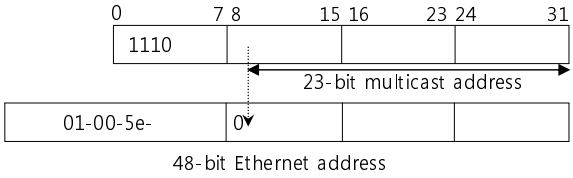

Figure 1.2: Ethernet address for multicast.

Ethernet type이 0x0800인 IP 패킷은 헤더에 프로토콜 번호(protocol number)를 나타내는 필드를 갖는다. 프록시 노드가 처리해야 하는 프로토콜의 종류는 표 1.4와 같다. 임의의 프록시 노드에서 이웃하는 노드로 패킷을 전송할 때는 IP-in-IP 방식의 캡슐화를 이용한다. IP-in-IP의 프로토콜 번호는 0x04이다.

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

Chapter 2

Requirement Spec. & Architecture

2.1 Specification

- 프록시 노드는 업링크와 다운링크를 구분하며 총 1개의 업링크 포트를 갖는다. 관리서버는 스패닝트리(spanning tree)의 유지·관리를 위해 임의의 프록시 노드가 가지는 부모노드(parents node)와 자식노드(child node)에 대한 정보와 노드 간의 연결관계에 대한 정보를 가지고 있다.

- 2계층 테이블(MAC table, Multicast MAC table)과 3계층 테이블(Flow table)이 구분된다. 테이블은 독립된 모듈에 의해 사용되며 3계층 기능은 Enable/Disable 될 수 있다. 2계층에서 모든 멀티캐스트 프레임은 멀티캐스트 MAC 주소 테이블과 수신 포트에 기반해 다운링크 포트로 플러딩된다.

- 프록시 노드는 ARP 및 IGMP 프로토콜을 지원하며, 이웃 프록시 노드 또는 관리서버와 통신하기 위해서 SSL(Secure socket layer)을 탑재한다. ARP, IGMP, SSL 등은 소프트웨어 영역에서 수행되며, 수행 결과로 발생되는 테이블은 소프트웨어 영역과 하드웨어 영역에서 동시 보관한다.

- ARP(Address resolution protocol) 패킷을 통해 MAC learning을 수행한다.

- 프록시 노드는 IGMP snooping(i.e., IGMP join과 leave를 모니터링)을 수행하며 업링크에서 발생하는 IGMP 패킷에 대해서 차단(blocking)한다. IGMP query와 그룹 관리는 프록시 노드에서 담당한다.

- 프록시 노드의 패킷 처리는 IPv4에 한정된다. 단, IPv6 트리팩을 IPv4 터널링 또는 브리징할 수 있는 기능 확장은 고려된다.

- 프록시 노드는 패킷 단편화(fragmentation) 문제를 처리하지 않는다. IP-in-IP 방식 또는 MAC-in-IP 방식의 사용으로 패킷 단편화가 발생할 경우 IP Router가 처리한다. 또한, 프록시 노드는 홉 카운트(hop count) 계산에 어떠한 영향도 주지 않는다.

- IGMP v1과 v2를 지원한다. IGMP v3는 설계에 반영한다.

- 프록시가 제공하는 기능을 활용하기 위해서 종단 호스트는 반드시 프록시 노드와 직접 연결되어 있어야 한다.
- 다운링크 1 포트 당 다수의 종단 호스트가 연결될 수 있으며, 업링크 및 다운링크에 연결된 모든 시스템은 동일 VLAN을 갖는 것으로 간주한다. 멀티캐스트 패킷은 동일 서브넷에서 플러딩된다.
- IGMP join에 대한 패킷 플로우(packet flow)의 구분은 Class D 주소를 기준으로 한다. IGMP join으로 인해 프록시 노드에 설정되는 플로우테이블은 동일한 그룹 주소를 사용할 경우 포트번호가 상이하더라도 동일한 패킷 플로우로 간주한다.

2.2 Architecture

2.2.1 Overview

SeeOver는 그림 2.1와 같이 구성된다[1]. 제어평면(control plane)과 데이터평면(data plane)이 분리되어 있으며 프록시 노드는 데이터평면에서 멀티캐스트 패킷의 점대점(point-to-point) 또는 점대다(point-to-multipoint) 포워딩을 담당한다. 중앙관리서버(Coordinator)는 제어평면에서 프록시 노드의 관리, 스패닝 트리의 구성 등을 담당한다. 프록시 노드에 연동된 종단 호스트의 멀티캐스트 그룹 참가/탈퇴(join/leave) 상태 등에 따라 멀티캐스트 그룹 g에 대한 스패닝 트리가 구성된다.

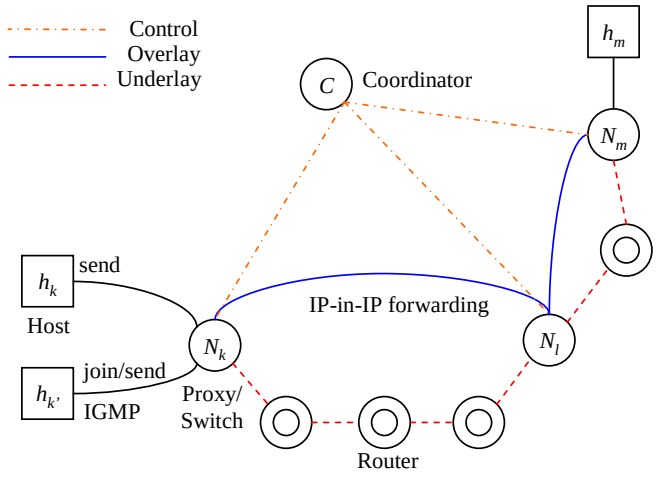

Figure 2.1: Data and Control plane.

프록시 노드는 2계층 스위치의 기능을 수행하며, IGMP snooping을 통해 종단 호스트의 멀티캐스트 참가 및 탈퇴를 모니터링한다. 멀티캐스트 패킷은 프록시 노드에서 IP-in-IP 캡슐화되어 이웃 노드에게 전달된다. 목적지 프록시 노드는 최종적으로 캡슐화된 IP 패킷을 역캡슐화한 후 그룹에 참가한 종단 호스트들에게 플러딩한다.

[1]MAC-in-IP 서비스를 위한 구성은 본 문서에 포함하지 않고 멀티캐스트 서비스를 위한 IP-in-IP 서비스 구성만 설명한다. 플로우테이블의 RULES와 ACTION을 통해 MAC-in-IP 서비스를 제공 가능하다.

요약하면, 종단 호스트와 프록시 노드 간에는 IGMP 규약을 이용하며 프록시 노드는 IGMP snooping을 통해 종단 호스트의 멀티캐스트 참가·탈퇴를 관리한다. 멀티캐스트 패킷의 목적지 전송을 위해서 프록시 노드 간에는 IP-in-IP 서비스를 제공한다. 프록시 노드가 제공한 정보를 바탕으로 관리서버는 스패닝트리를 구성하고 패킷 플로우에 대한 처리 방법을 프록시 노드에게 전달한다. 프록시 노드는 관리서버가 제공한 플로우테이블에 기반해 해당 패킷 플로우를 고속 포워딩한다.

2.2.2 Proxy Node

프록시 노드는 그림 2.2와 같이 크게 IP 패킷 및 MAC 프레임 포워딩 기능을 수행하는 하드웨어 부분과 통신규약 등의 처리를 위한 소프트웨어 부분으로 구분된다.

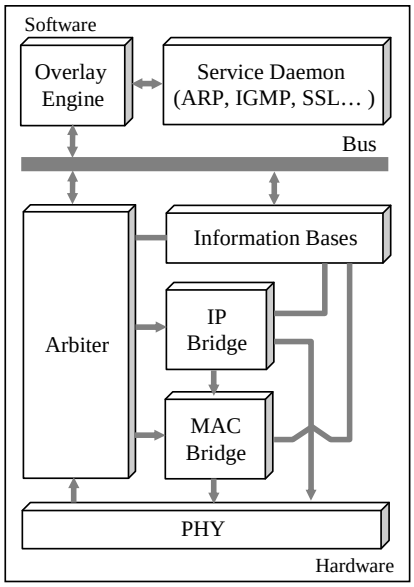

Figure 2.2: System Architecture (Software & Hardware).

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

1. `Arbiter`, 물리계층(`PHY`)을 통해 입력되는 MAC 프레임을 통해 프레임 및 패킷의 종류를 파악하고, 종류에 따른 처리 블럭을 선택한다. 또한, MAC table을 유지·관리하고 IP checksum 오류를 검사한다. 예를 들어, ARP에 해당하는 MAC 프레임을 수신 받았을 경우에, MAC table의 관리를 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 총괄한다.

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`가 보유 중인 Flow table을 읽어 IP 패킷을 캡슐화(encapsulation) 또는 역캡슐화(de-encapsulation)한다. 캡슐화는 IP-in-IP 방식이며 캡슐화된 IP 패킷은 점대점으로 연결된 이웃(neighbor) 프록시 노드에게 전달되며, 역캡슐화되는 IP 패킷은 다운링크 포트에 연결된 종단 호스트들에게 플러딩된다.

2.2.3 MAC frame & IP packet processing

그림 2.3는 2계층에서 MAC 프레임을 기준으로 패킷(또는 프레임)을 구분해 처리하는 과정을 보여준다. 먼저, 수신된 프레임의 목적지 MAC 주소를 바탕으로 워크플로우(workflow)가 분기된다. 브로드캐스트 주소를 가질 경우 헤더의 Ethernet type 필드값을 읽어 ARP 패킷 여부를 확인한다. ARP 패킷은 소프트웨어 영역(`CPU`)으로 옮겨진 후 그림 2.4과 같이 처리된다[2]. 일반 브로드캐스트 패킷은 플러딩한다.

목적지 MAC 주소가 일반 유니캐스트 주소(Ethernet type은 IP)일 경우에는 해당 프록시 노드의 MAC 주소와 수신한 프레임의 MAC 주소를 비교한다. 두 주소가 일치할 경우에는 IP 패킷의 프로토콜 번호를 읽어 IP-in-IP 캡슐화(0x04) 여부를 파악한다. 캡슐화되지 않은 패킷은 소프트웨어 영역으로 옮겨 처리한다. 캡슐화된 패킷일 경우에 플로우테이블을 검색해 해당되는 RULES를 찾고 ACTION을 수행한다. RULES가 플로우테이블에 존재하지 않을 경우 오류이므로 소프트웨어에서 예외처리(`Error handling`)한다. RULES는 패킷 플로우를 필터링하는 규칙이며 ACTION은 해당 플로우를 처리하는 규정(예, discard, forward 등)이다.

목적지 MAC 주소가 IP 멀티캐스트일 경우에는 수신된 패킷의 입력 포트를 파악한다. 업링크를 통해 입력되는 멀티캐스트 프레임은 처리하지 않고 버린다(`Discard`). 다운링크 포트에서 수신된 프레임은 IGMP 패킷을 포함하고 있을 경우 소프트웨어가 처리하고 일반 Class D 주소를 가질 경우에는 플로우테이블을 검색한다. 플로우테이블에 RULES가 존재하면 해당 ACTION을 수행한다. RULES가 존재하지 않을 경우에는 소프트웨어가 관리서버로부터 RULES를 받고 플로우테이블을 갱신한다.

그림 2.4는 ARP 패킷을 소프트웨어(`CPU`)에서 처리하는 절차를 보여준다. ARP 패킷의 Ethernet Type은 `0x0806`이다. 임의의 호스트는 자신의 MAC 테이블이 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 2.4에서는 해당 프록시 노드가 패킷 송신자로써 ARP `request` 패킷을 발생시키는 과정은 생략되어 있다(즉, 타 호스트가 발생시킨 ARP `request`의 처리 과정만 포함하고 있다.).

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 임시 저장된 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.

멀티캐스트 MAC 주소 테이블은 {Port, Sender MAC, Multicast MAC}을

[2]NetFPGA에서 직접 처리 가능하다.

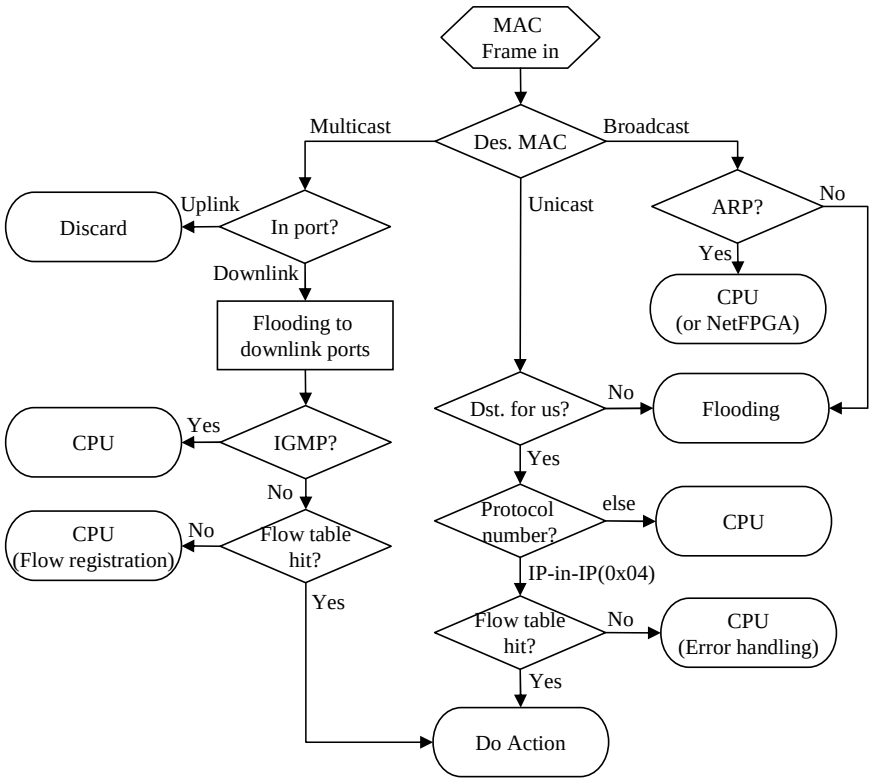

Figure 2.3: Frame processing in Layer 2.

기본 tuples로 갖으며 확장가능하다. 그림 2.5은 프록시 노드에서 IGMP 패킷을 처리하고 멀티캐스트 MAC 주소 테이블을 유지하는 과정을 보여준다.

프록시 노드는 지속적으로 IGMP snooping을 수행한다. 검색된 IGMP join/leave/report 메시지를 바탕으로 멀티캐스트 MAC 테이블을 유지한다. 먼저, 프록시 노드가 IGMP join이나 report 메시지를 필터링했을 경우에, 멀티캐스트 MAC 주소 테이블을 검색한다. 이미 존재하는 엔트리에 대해서는 해당 엔트리를 유지하며 존재하지 않을 경우에는 멀티캐스트 MAC 주소 테이블에 추가한 후, 관리서버에게 플로우엔트리의 등록을 요청하며 관리서버로 부터 받을 플로우엔트리를 로컬 플로우테이블에 추가한다. 멀티캐스트 MAC 주소 테이블에 추가하는 과정은 플로우테이블에 추가한 직후 또는 관리서버에게 플로우 등록을 요청하기 직전에 삽입 가능하다(그림 2.5은 직전에 추가하는 예).

IGMP leave 메시일 경우에는 멀티캐스트 MAC 주소 테이블에서 삭제한 후, 관리서버에게 보고하고 관리서버가 보낸 RULES에 기반해 플로우테이블을 갱신한다. IGMP join과 마찬가지로 MAC 주소 테이블에서 삭제하는 과정은 플로우테이블을 갱신한 직후에 수행할 수 있다.

2.2.4 Signalling: Coordinator-Proxy

다음 그림 2.6은 관리서버와 프록시 노드 간의 시그널링과 관련된 절차를 보여준다. 프록시 노드는 2계층에서 플로우테이블에 등록되지 않은 멀티캐스트 패킷(multicast or IGMP packet)을 수신하면 관리서버(C)에게 플로우엔트리을

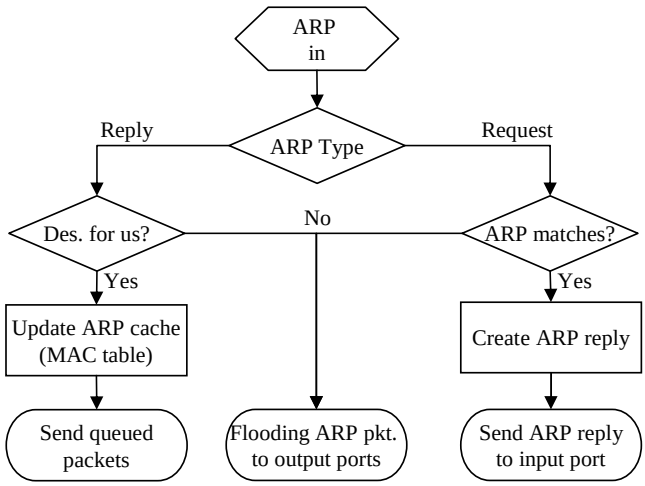

Figure 2.4: ARP processing.

요청한다(h_α `request`). 서버는 자신의 토폴로지 지도에 기반해 이웃한 노드를 선택하고 플로우의 처리지침(ACTION)을 결정해 프록시 노드에게 전달한다(`C response`). 관리서버로부터 플로우 처리지침을 전달받은 프록시 노드는 해당 엔트리(entry)를 플로우테이블에 기록한다(h_α `register`). 이후 수신되는 동일 플로우에 대해서는 플로우테이블을 참조해 포워딩한다.

호스트(h_α)의 `join` 또는 `send` 여부에 따라 관리서버가 결정하는 명령이 구분된다.

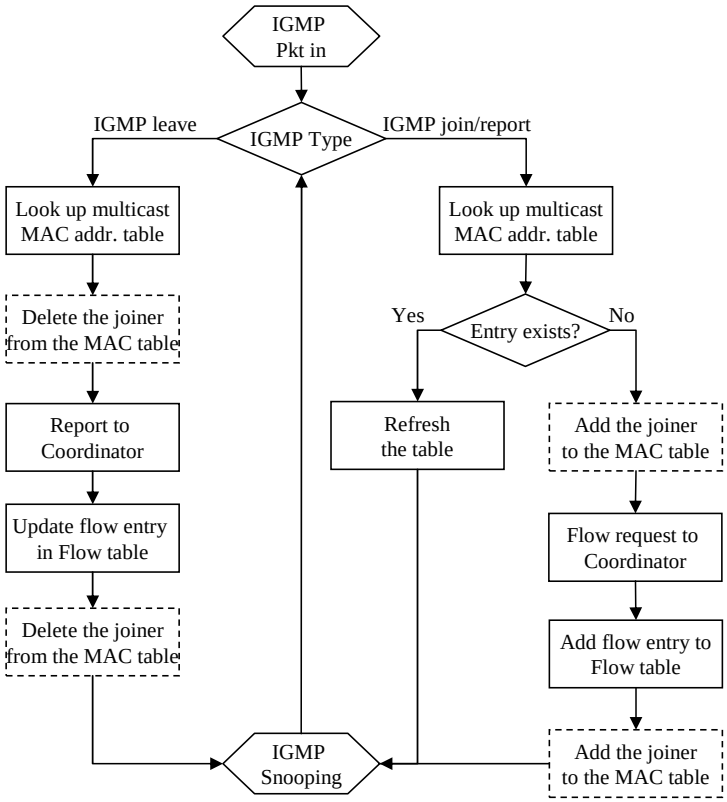

Figure 2.5: IGMP processing.

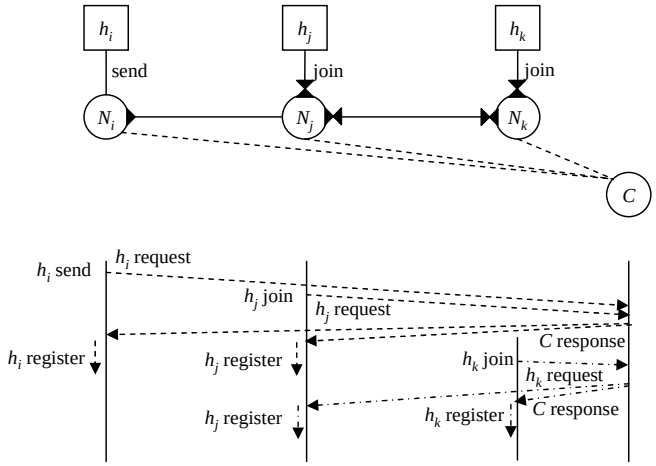

Figure 2.6: Signalling.

Chapter 3

Packet Forwarding & Group Management

3.1 Term. definition

이후에 사용될 기호와 의미를 정의한다. $\mathfrak{s}(h_i, g)$는 종단 호스트 h_i에서 그룹 g로 멀티캐스트 패킷을 송신하는 것을 의미(그룹에 참가하지 않은 상태에서)하고 $\mathfrak{j}(h_j, g)$는 종단 호스트 h_j에서 IGMP join 메시지를 통해 그룹 g로 참가하는 것을 뜻한다. $\mathfrak{l}(h_k, g)$는 호스트 h_k가 그룹 g에서 IGMP leave메시지를 통해 탈퇴하는 것을 의미한다.

그룹 g에 대해 노드 N_i와 N_j의 양방향성(bi-directional) 연결관계를 나타낼 때는 η_{i-j}로 표기한다. 연결관계는 그룹 g에 대해 이웃하는 두 노드의 참가 여부를 나타낸다. 그룹 g에 대해 N_i가 N_j로의 연결설정을 갖지 못할 경우, 연결관계를 $\eta_{\bar{i}-\bar{j}}$로 표기된다. 그룹 g에 대해서 이웃한 두 노드 N_i와 N_j가 단방향성(uni-directional) 연결관계를 가지고 있을 때는 $\eta_{\bar{i}-j}$ 또는 $\eta_{i-\bar{j}}$ 표시한다.

η_{i-j} 관계일 경우 노드 N_i와 N_j간에 그룹 g에 속한 멀티캐스트 패킷은 양방향 전송될 수 있다. $\eta_{i-\bar{j}}$ 관계하에서는 그룹 g에 대해 노드 N_i에서 N_j로 멀티캐스트 패킷이 단방향 전송될 수 있다. 그림 3.1에서 N_i와 N_j의 관계는 $\eta_{i-\bar{j}}$이고 N_j와 N_k의 관계는 η_{j-k}이다. 또한, 프록시 노드 N_j와 N_k는 하나 이상의 양방향성 η 관계를 포함하고 있고 N_i는 단방향성 η 관계만 포함하고 있다.

노드 N_α를 통해 그룹 g에 참가하고 있는 종단 호스트들의 수를 κ_α라 하고, 노드 N_α가 다른 프록시 노드들과 가지고 있는 연결관계 η의 수를 μ_α로 정의한다. 관리서버는 그룹 g에 대한 스패닝트리 정보를 가지고 있다. 스패닝트리를 구성하는 모든 프록시 노드들은 집합 G_t의 원소이며, 집합 G_p는 집합 G_t의 부분집합($G_t \supseteq G_p$)으로 그룹 g에 대해 $\kappa_i \geq 1$인 임의의 프록시 노드 i 들로 구성된 집합이다.

3.2 Flow table

모든 플로우테이블은 {RULES, ACTION, STATISTICS}로 구분된다. 플로우테이블의 RULES 부분은 $p-type$ tuples로 정의된다. $p-type$ RULES는 $p-type$과 $m-type$ ACTION으로 구분된다. 먼저, $p-type$ RULES는(표 3.1 참조) Openflow 스위치의 플로우테이블과 유사하게 구성되며 총 12-tuples를 갖는다. RULES는 {입력 포트(IN_{port}), IP, MAC in IP(MAC_{in-IP}), IP in IP(IP_{in-IP}),

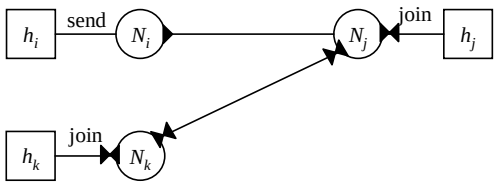

Figure 3.1: Join relation.

TCP/UDP 포트(TCP)}로 구성된다. S_{addr}와 D_{addr}는 각각 송신 및 목적지 IP 주소이며 P_{num}은 IP 헤더의 플로토콜 번호(protocol number)이다. S_{mac}과 D_{mac}은 송신 및 목적지 MAC 주소이며 E_{type}은 MAC 프레임의 이더넷 타입(Ethernet type)이다. 마지막으로 S_{port}와 D_{port}는 TCP/UDP의 송신 및 목적지 포트번호이다.

MAC_{in-IP}와 IP_{in-IP}는 IP 패킷이 각각 MAC-in-IP 또는 IP-in-IP 캡슐화되었을 때만 값을 갖는다. 예를 들어, 2계층 프레임, {MAC', IP', TCP},이 MAC-in-IP 캡슐화될 때 해당 IP 패킷은 2계층에서 {MAC, IP, {MAC', IP', TCP}}를 가지게 된다. 이때 RULES의 MAC_{in-IP}는 앞 예의 MAC'에 해당되고 IP_{in-IP}는 IP'이 된다. 동일한 방법으로 2계층 프레임이 IP-in-IP 캡슐화되면 2계층에서 {MAC, IP, {IP', TCP}}로 나타나며 MAC_{in-IP}는 null에 해당하고 IP_{in-IP}는 IP'이 된다.

Table 3.1: Flow table: $p-type$ RULES.

IN_{port}	IP			MAC_{in-IP}			IP_{in-IP}			TCP	
	S_{addr}	D_{addr}	P_{num}	S_{mac}	D_{mac}	E_{type}	S_{addr}	D_{addr}	P_{num}	S_{port}	D_{port}

Table 3.2: Flow table: $p-type$ ACTION.

		Reserved(8bit)	Action(8bit)	Mask(16bit)
MAC_{in-IP}		S_{mac}(MSB)		
		S_{mac}(LSB)		D_{mac}(MSB)
		D_{mac}(LSB)		
IP_{in-IP}		S_{addr}		
		D_{addr}		
IP_{in-IP}	IP	P_{num}		P_{num}
IP		S_{addr}		
		D_{addr}		
TCP/UDP		S_{port}		D_{port}

표 3.2는 $p-type$ RULES에 의한 패킷 플로우의 처리 지침(ACTION)을 나타낸다. `Action`(8bit) 필드 중, MSB(Most Significant Bit) 3비트는 활용하지 않으며 하위 5비트를 사용한다. 하위 5비트의 용도는 표 3.3에 정의되어 있다. 1번 비트(`modif.`)는 플로우테이블의 `Mask` 비트에 기초해 헤더 필드({IP, MAC_{in-IP}, IP_{in-IP}, TCP})의 변경을 결정한다. 0번 비트(`popul.`)는 $m-type$ ACTION의 추가 수행 여부를 결정하며 '1'로 설정되어 있을 경우 $m-type$ ACTION 테이

블을 읽어 패킷을 처리한다. $m-type$ 테이블은 패킷의 다지점 복제를 위해 활용된다. 비트는 각 (4,3,2), `modif.`, `popul.`로 분류(grouping)되며 실행 순서는 좌에서 우로 진행된다. 즉, `modif.`은 `popul.` 이전에 일어난다.

`Mask`(16bit) 필드는 `Action` 필드의 '1'번 비트(`modif.`)가 1로 설정되었을 때 실행되고, 각 비트가 지정하는 $p-type$ ACTION의 필드값을 수정한다. MSB 6비트는 사용되지 않고 나머지 10비트만 사용되며, 9번 비트부터 내림차순으로 표 3.2의 {S_{mac}, D_{mac}, S_{addr}, D_{addr}, P_{num}(`in-IP`), P_{num}(`IP`), S_{addr}, D_{addr}, S_{port}, D_{port} }를 지시한다. 예를 들어, $p-type$ ACTION 테이블이 갖는 `Action`필드의 `modif.` 비트가 1로 설정되어 있고 `Mask` 값이 이진수 0000000000000011(0x0003)을 가지면, 수신된 패킷 플로우의 TCP 또는 UDP 헤더의 출발지 포트번호와 목적지 포트번호를 각각 $p-type$ ACTION 테이블의 S_{port}와 D_{port}로 변경하라는 의미이다.

3.2.1 Examples

그룹 g를 갖는 멀티캐스트 서비스를 제공하기 위해 프록시 노드는 다음과 같은 3 종류의 플로우테이블을 가질 수 있다. 아래는 하나의 예이며, 패킷 플로우가 갖는 속성에 따라 RULES를 다르게 설정할 수 있다. S^{IP}_{addr}은 $p-type$ RULES의 IP 필드에 해당하는 송신지 주소이며 g는 IP 또는 IP_{in-IP} 필드의 D_{addr}이다. 위첨자는 $p-type$ RULES의 해당 필드를 나타낸다.

1. 노드 N_α가 종단 호스트 h_α가 보낸 IGMP join 메시지를 받았을 때, 플로우테이블의 RULES는 (*, (S^{IP}_{addr}, g, *), (*,*,*), (*,*,*), (*,*))로 설정될 수 있다. 관리서버는 그룹 g에 대해 IGMP join된 패킷 플로우에 대한 필터링 RULES를 결정한다. 이후 호스트 h_α가 그룹 g에 보내는 IP 멀티캐스트 패킷은 위 RULES를 이용해 필터링된다. 프록시 노드에 수신된 IGMP 패킷을 기준으로 결정된다.

2. 호스트 h_α가 그룹 g에 참가하지 않고 멀티캐스트 패킷을 송신할 때, 플로우테이블의 RULES는 (*, (S^{IP}_{addr}, g, p^{IP}_{num}), (*,*,*), (*,*,*), (*,*))를 갖는다. 유니캐스트 전송을 위한 플로우테이블의 RULES와 같다. 프록시 노드에 수신된 IP 멀티캐스트 패킷을 기준으로 결정된다.

3. 프록시 노드와 노드 사이의 방향성 연결관계(멀티캐스트 패킷을 송·수신)를 나타내기 위한 플로우테이블의 RULES는 (*, (*,*,*), (*,*,*), (*,g, $P^{IP_{in-IP}}_{num}$), (*,*))이 가능하다. 프록시 노드에 수신된 IGMP 또는 IP 멀티캐스트 패킷을 기준으로 결정된다.

Table 3.3: $p-type$ ACTION: `Action` definition.

4	3	2	1 (`modif.`)	0 (`popul.`)	Description
1	1	1	0	0	discard the packet
0	0	0	0/1	0	do nothing or modif., only
0	0	0	0	1	lookup $m-$`type` table
0	0	1	0/1	0/1	IP-in-IP encap., wo/w modif.
0	1	0	0/1	0	IP-in-IP de-encap., wo/w modif.
0	1	1	0/1	0/1	MAC-in-IP encap., wo/w modif.
1	0	0	0/1	0	MAC-in-IP de-encap., wo/w modif.

(*, (S^{IP}_{addr}, g, p^{IP}_{num}), (*,*,*), (*,*,*), (*,*))에 대한 ACTION은(즉, 그룹 g에 참가하지 않은 호스트가 패킷을 송신할 때) 프록시 노드 N_α가 이웃한 노드와 단방향성 연결관계를 가지게 되므로 $m-type$ ACTION 테이블을 추가적으로 읽어 점대점 또는 점대다 전송을 수행한다.

유니캐스트 전송(점대점)을 위해서는, $p-type$ ACTION의 `Action` 필드가, 예를 들어, 이진수 00000110(0x06)으로 설정되고, `Mask` 필드가 이진수 0000000000011100(0x001C)로 설정될 수 있다. 플로우테이블에 속한 임의의 RULES로 사상되는 패킷이 프록시 노드로 수신되면 해당 패킷을 IP-in-IP 캡슐화 하고 캡슐화된 IP 헤더의 출발지, 목적지 및 프로토콜 번호를 ACTION에 따라 변경한 후 송신한다. 점대다 전송은 아래 RULES (*, (*, g, p^{IP}_{num}), (*,*,*), (*,*,*), (*,*))과 동일한 방법으로 처리된다.

RULES (*, (*, g, p^{IP}_{num}), (*,*,*), (*,*,*), (*,*))에 대한 $p-type$ ACTION의(즉, 그룹 g에 참가한 호스트가 패킷을 송신할 때) `Action` 필드값은 이진수 00000111(0x07)을 가질 수 있으며 `Mask`로 이진수 0000000000011000(0x0018)가 설정될 수 있다. 즉, IP-in-IP 캡슐화를 수행한 후에 캡슐화한 IP 헤더의 출발지 주소와 프로토콜 번호를 $p-type$ ACTION 필드의 S_{addr}와 P_{num} 값으로 변경하고 $m-type$ ACTION 테이블을 읽어 이후 명령을 수행한다.

RULES (*, (*,*,*), (*,*,*), (*,g, $P^{IP_{in-IP}}_{num}$), (*,*))에 대한 $p-type$ ACTION은(즉, 해당 프록시 노드가 이웃한 노드와 연결관계를 가지고 있을 때) 상기 RULES (*, (*, g, p^{IP}_{num}), (*,*,*), (*,*,*), (*,*))와 동일한 방법으로 패킷 플로우가 처리된다. 프록시 노드와 노드사이에는 in-IP 캡슐화가 필요하지 않으므로 $p-type$ ACTION의 `Action` 필드는 이진수 00000011(0x03)을 갖고, `Mask`는 이진수 0000000000001000(0x0008)로 설정될 수 있다. 즉, 수신받은 패킷이 캡슐화되어 있는 상태이므로 IP 헤더의 출발지 주소를 $p-type$ ACTION의 IP 필드에 속한 S_{addr}로 수정한 후 $m-type$ 테이블을 읽어 점대다 전송을 수행한다.

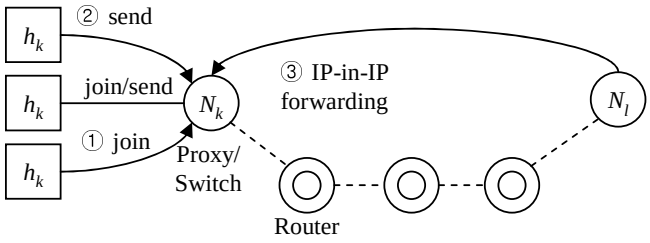

Figure 3.2: Example: Flow matching.

Table 3.4: Flow table and incoming packet (Fig.10).

Flow table: $p-type$ RULES in N_k	
①	(*,(S^{IP}_{addr},g,*),(*,*,*),(*,*,*),(*,*))
②	(*,(S^{IP}_{addr},g,p^{IP}_{num}),(*,*,*),(*,*,*),(*,*))
③	(*,(*,*,*),(*,*,*),(*,g,$p^{IP_{in-IP}}_{num}$),(*,*))
Packet to N_k	
①	(4,(*,g,p^{IP}_{num}),(×,×,×),(×,×,×),(*,*))
②	(2,(S^{IP}_{addr},g,17),(×,×,×),(×,×,×),(20,30))
③	(1,(N_l,N_k,4),(×,×,×),(h_α,g,$p^{IP_{in-IP}}_{num}$),(20,30))

그림 3.2과 표 3.4은 노드 N_k에서 수신된 IP 패킷과 플로우테이블 간의 사상(mapping)관계를 보여준다. 노드 N_k가 가지고 있는 $p-type$ RULES가 표 3.4의 상단(Flow table: $p-type$ RULES in N_k)과 같이 설정되어 있다고 가정하자. 호스트 h_k가 그룹 g에 참가할 경우, 프록시 노드 N_k에 도착하는 IGMP join 메시지는 표 하단(Packet to N_k)의 ①과 같은 정보를 의미한다. 또한, 호스트 h_k가 그룹 g에 참가하지 않고 멀티캐스트 패킷을 N_k로 보낸다면 표 하단의 ②와 같은 정보를 가지고 있다. 마지막으로, 이웃한 프록시 노드 N_l에서 노드 N_k로 전송한 IP 패킷은 ③과 같은 패킷 정보를 가지고 있다. 표에서 프로토콜 번호 4는 IP-in-IP이며 17은 UDP(User datagram protocol)이다.

표 하단의 ①과 같은 정보를 포함하는 IP 패킷이 프록시 노드 N_k에 도착하면, N_k는 플로우테이블을 검색해 필터링된 RULES (*, (S^{IP}_{addr}, g, *), (*,*,*), (*,*,*), (*,*))를 얻게된다. 필터링은 최적 정합(best match)한 RULES를 반환한다. 예를 들어, 표 상단과 같은 $p-type$ 플로우테이블이 존재하고 하단의 ②와 같은 정보를 갖는 패킷이 프록시 노드에 도착했다고 가정하자. 해당 패킷은 플로우테이블의 RULES (*, (S^{IP}_{addr}, g, *), (*,*,*), (*,*,*), (*,*))과 RULES (*, (S^{IP}_{addr}, g, p^{IP}_{num}), (*,*,*), (*,*,*), (*,*)) 등 두 종류의 후보군이 있지만, 최적 정합하는 (*, (S^{IP}_{addr}, g, p^{IP}_{num}), (*,*,*), (*,*,*), (*,*))가 최종적으로 반환된다. 즉, 정합되는 필드가 각각 2개와 3개이므로, 많은 정합 필드를 갖는 RULES가 반환된다.

표 3.5는 $m-type$ ACTION 테이블의 형태를 보여준다. G_{addr}은 그룹 주소이며 IN_{port} Filter와 함께 $m-type$의 RULES 역할을 한다. $m-type$의 G_{addr}은 해당 그룹에 속한 플로우 집합의 대표 주소이다. 즉, 그룹 G_{addr}에 속한 플로우는 {IN_{port} Filter, Action, OUT_{port}, N_{addr}} tuples를 갖는다.

IN_{port} Filter는 해당 패킷 플로우의 $m-type$ ACTION 테이블이 갖는 `Action` 필드의 수행 여부를 결정한다. $p-type$ RULES의 IN_{port} 값과 $m-type$ ACTION의 IN_{port} Filter 값을 AND 연산 시킨 값이 0이 아닐 경우 $m-type$ ACTION 테이블의 `Action`을 수행한다. $m-type$ ACTION 테이블의 `Action` 필드값이 0x0으로 설정되어야 할 경우 IN_{port} Filter 값이 이진수 0001로 설정된다. 즉, 업링크 포트(G0/1)를 통해 수신된 패킷만 처리된다.

$m-type$ ACTION의 `Action` 필드값이 0x1로 설정되여야 할 경우에는 IN_{port} Filter 값이 이진수 1111이 된다. 즉, 모든 물리 포트로 부터 수신된 패킷에 대해서 $m-type$ ACTION의 `Action`을 수행한다. OUT_{port}는 bitmask로써 각 물리 포트와 bit의 위치를 1:1로 사영한 값이다. 상위 4비트를 사용하지 않으며 MSE부터 {G0/4, G0/3, G0/2, G0/1}를 의미한다. $m-type$ ACTION에 의해 수정된 패킷이 출력되어야 할 물리 포트이다. G0/1는 업링크 포트이다. 예를 들어, 임의의 패킷 플로우에 대한 $m-type$ ACTION의 OUT_{port}가 이진수 1110(0xF)로 설정되었을 경우는 다운링크 포트로 플러딩된다. 즉, 해당 패킷은 NetFPGA의 물리 포트 {G0/4, G0/3, G0/2}로 출력된다.

Table 3.5: Flow table: $m-type$ ACTION.

G_{addr}			
IN_{port} Filter	Action	OUT_{port}	N_{addr}

프록시 노드 N_α의 차수(degree) $d(N_\alpha)$는 그룹 g에 대해서 노드 N_α가 소유하고 있는 $m-type$ tuples의 개수를 나타낸다. 즉, 연결관계에 있는 이웃 프록시 노드들의 수를 n이라고 했을 때, $d(N_\alpha)$은 $n+1$(노드 N_α와 연동되어 있고 그룹 g에 참가하고 있는 종단 호스트들의 집합을 1로 여김)이다.

Table 3.6: $m-type$ `Action` definition.

bit 1	bit 0	Description
0	0	IP-in-IP de-encapsulation
0	1	IP-in-IP forwarding

3.3 Group Management

본 절은 종단 호스트가 멀티캐스트 그룹 g에 참가 및 탈퇴하는 방법과 프록시 노드에서 플로우테이블을 설정하는 방법에 대해서 정의한다. 플로우테이블의 {RULES, ACTION}과 관련된 결정은 관리서버가 담당하며 설정된 플로우테이블에 기초한 패킷 포워딩은 프록시 노드가 수행한다.

먼저, 본 절에서 사용될 표기법을 정의한다. 표 3.1의 $p-type$ RULES가 나타내는 대표 필드 {IN_{port}, IP, MAC_{in-IP}, IP_{in-IP}, TCP}를 {$p(i)$, $p(g_o)$, $p(m_i)$, $p(g_i)$, $p(t)$}로 표기한다. 예를 들어, $p-type$ RULES의 IP 필드에 해당하는 tuples는 (S_{addr}, D_{addr}, P_{num})이고 $p(g_o)$와 동일한 의미로 사용된다. 프록시 노드 N_α에 플로우테이블의 추가, 삭제 등 갱신이 필요할 때는 [add/delete] $f(\alpha) : p(g_o) \rightarrow m(f_w;\beta)$과 같이 표기하며 $p-type$ RULES가 $p(g_o)$ tuples와 관계되고, ACTION($\rightarrow$)은 $m(f_w:\beta)$와 관련됨을 의미한다. $m(f_w:\beta)$은 $m-type$ ACTION 테이블을 I/O하며 "프록시 노드 N_β로 포워딩"하는 명령이다. 추가적으로, $m(f_d)$는 $m-type$ 테이블을 I/O하며 "다운링크 플러딩"을 의미하며 $p(f_w;\beta)$는 $p-type$ 테이블을 I/O하고 "N_β로 포워딩" 을 나타낸다. 사용예는 다음과 같다.

- $f(\alpha) : p(g_o^m) \rightarrow m(f_w;\beta)$: 노드 N_α의 플로우테이블을 설정한다. $p-type$ RULES는 IP(S_{addr}, D_{addr}, P_{num})에 의해 결정된다. g_o^m은 (S_{addr}^{IP},g,*)이다. $m-type$ 테이블을 이용하고 해당 플로우를 "N_β로 포워딩"을 의미한다. 생략된 명령에 따라 테이블에 추가, 삭제, 갱신할 수 있다.

- $f(\alpha) : p(g_o^m) \rightarrow m(f_d)$: 상기 예와 동일하다. $m-type$ 테이블을 이용하고 해당 플로우를 "N_α의 다운링크 포트로 플러딩"을 의미한다.

- $f(\alpha) : p(g_o^u) \rightarrow p(f_w;\beta)$: 노드 N_α의 플로우테이블을 설정한다. $p-type$ RULES는 IP(S_{addr}, D_{addr}, P_{num})에 의해 결정된다. g_o^u는 (S_{addr}^{IP}, g, P_{num}^{IP})이다. $p-type$ 테이블을 이용하고 해당 플로우를 "N_β로 포워딩"을 의미한다.

- $f(\alpha) : p(g_i) \rightarrow m(f_d;\beta)$: 노드 N_α의 플로우테이블을 설정한다. $p-type$ RULES는 IP_{in-IP}(S_{addr}, D_{addr}, P_{num})에 의해 결정된다. $p(g_i)$는 (*,g,*)이다.

- $f(\alpha) : p(g_i) \rightarrow m(f_w)$: 노드 N_α의 플로우테이블을 설정한다. 상기 예와 동일한다. $m-type$ 테이블을 이용하고 해당 플로우를 "N_α의 다운링크 포트로 플러딩"을 의미한다.

3.3.1 Group Join

Join relation among proxy nodes

종단 호스트가 IGMP join 메시지를 발생시킬 때 해당 메시지를 수신한 프록시 노드들이 보고한 내용을 중심으로 관리서버가 멀티캐스트 스패닝트리가 구

성한다. $\exists N_i, N_j \in P$(P는 프록시 노드들의 집합)에 대해서, N_i (h_i와 연결된 프록시 노드)와 N_j가 그룹 g에 대한 $\{\mathfrak{s}(h_i,g),\ \mathfrak{j}(h_j,g)\}$, $\{\mathfrak{j}(h_i,g),\ \mathfrak{s}(h_j,g)\}$ 또는 $\{\mathfrak{j}(h_i,g),\ \mathfrak{j}(h_j,g)\}$ 패킷을 받으면 경로 $\{h_i, N_i, N_j, h_j\}$사이에 초기 트리가 구성된다. 그룹 g에 대해 연결관계를 맺고자 하는 프록시 노드들은 초기 트리가 구성된 이후 트리에 추가될 수 있다. 즉, 적어도 2개의 프록시 노드들이 그룹 g에 대해 서로 연결관계를 맺고 있을 때만 타 프록시 노드들이 스패닝트리에 추가될 수 있다. 기본적으로 종단 호스트에서 발생한 모든 멀티캐스트 프레임들은 프록시 노드에 의해 다운링크 포트로 플러딩된다.

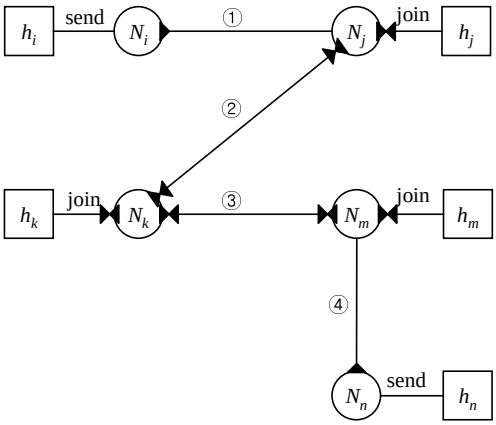

Figure 3.3: Group Join.

그림 3.3과 같이 멀티캐스트 그룹 g를 위한 스패닝트리가 생성된다고 가정하자. 그림의 ①에서 h_i가 $\mathfrak{s}(h_i,g)$를 요청하고 h_j가 $\mathfrak{j}(h_j,g)$를 요청해 그룹 g에 대한 초기 트리가 구성된다. 다대다 통신의 경우 $\mathfrak{j}(h_j,g)$는 묵시적으로 $\mathfrak{s}(h_j,g)$의 가능성을 내포하고 있다. 그룹 g에 대한 초기 트리가 구성된 이후에 해당 그룹에 가입하고자 하는 프록시 노드는 다음과 같은 방법으로 기존 프록시 노드와 연결관계를 맺는다. 연결설정은 항상 한 쌍의 프록시 노드들 사이에서만 이루어지며, 관리서버가 연결관계를 가질 한쌍의 프록시 노드들을 선택한다. 종단 호스트 h_γ가 연결된 프록시 노드 N_γ에 $\mathfrak{j}(h_\gamma,g)$ 또는 $\mathfrak{s}(h_\gamma,g)$ 메시지를 발생시키면, 프록시 N_γ는 관리서버에게 이벤트 발생 사실을 통보한다. 관리서버는 $N_\alpha \in G_t$인 최적의 N_α를 선택해 N_α와 N_γ에게 각각의 RULES와 ACTION을 전달한다[1].

관리서버에서 임의의 노드 N_γ와 연결관계를 맺을 프록시 노드 N_α를 선택한 후 두 노드에게 전달할 플로우테이블은 다음과 같은 연결 설정 알고리즘에 의해 결정된다.

- 연결 설정 알고리즘 ①: 프록시 노드 N_γ가 그룹 g에 참가하고 있는 종단 호스트들이 없는 상황에서, 호스트 h_γ가 IGMP join 메시지를 발생시키면 관리서버는 노드 N_γ에게 "그룹 g에 대해 캡슐화되어 수신된 패킷을 다운링크로 플러딩$(p(g_i) \rightarrow m(f_d))$" 명령과 "그룹 g에 대해 캡슐화되지 않은 패킷에 대해서 노드 N_α로 포워딩$(p(g_o^m) \rightarrow m(f_w;\alpha))$"을 해당 프록시 노드들에게 전달한다. 또한, 프록시 노드 N_α에게 "그룹 g에 대해 캡슐화되어 수신된 패킷을 노드 N_γ로 포워딩$(f(\alpha) : p(g_o^m) \rightarrow m(f_w;\gamma))$" 명령과

[1] 라우팅 및 성능 최적화를 위한 노드 선택 알고리즘은 본 문서에서 포함하지 않는다.

"그룹 g에 대해 캡슐화되지 않고 수신된 패킷을 노드 N_γ로 포워딩"을 플로우테이블에 설정하게 한다. 노드 N_α에 그룹 g에 대한 참가자가 있어야 노드 N_γ이 스패닝트리에 추가 될 수 있으므로, 노드 N_α는 플로우테이블에 $p(g_o^m)$과 $p(g_i)$ 정보를 이미 가지고 있는 상태이다. 따라서, 노드 N_α는 $m(f_w;\gamma)$만 추가하면 된다. 설명을 위해 전체 명령을 기록한다.

- 연결 설정 알고리즘 ②: 프록시 노드 N_γ가 그룹 g에 참가하고 있는 종단 호스트들이 없는 상황에서, 호스트 h_γ가 그룹 g에 참가하지 않고 멀티캐스트 패킷을 송신하면 관리서버는 노드 N_γ에게 "호스트 h_γ가 그룹 g에 대해서 캡슐화하지 않고 보낸 멀티캐스트 패킷을 수신하면 노드 N_α로 포워딩"($p(g_o^u) \rightarrow m(f_w;\alpha)$)을 명령한다. h_γ가 그룹 g에 참가하지 않았으므로 관련된 플로우를 받을 수 없다. 따라서, 이웃한 프록시 노드에는 플로우테이블을 설정하지 않는다.

연결 설정 알고리즘 ①의 경우 연결관계 $\eta_{\gamma-\alpha}$가 성립한다. 관계 성립 후, 호스트 h_γ에서 $\mathfrak{s}(h_\gamma, g)$되는 멀티캐스트 패킷은 플로우테이블의 RULES와 ACTION $p(g_o^m) \rightarrow m(f_w;\alpha)$에 의해 포워딩된다.

if ①: h_γ issues $\mathfrak{j}(h_\gamma, g)$ AND $\kappa_\alpha \geq 1$ in N_α **then**
  set $f(\gamma) : p(g_i) \rightarrow m(f_d)$, $p(g_o^m) \rightarrow m(f_w;\alpha)$
  set $f(\alpha) : p(g_o^m) \rightarrow m(f_w;\gamma)$, $p(g_i) \rightarrow m(f_w;\gamma)$
else
  if ②: (h_γ issues $\mathfrak{s}(h_\gamma, g)$ AND $\kappa_\gamma = 0$ in N_γ) AND $\kappa_\alpha \geq 1$ in N_α **then**
    set $f(\gamma) : p(g_o^u) \rightarrow m(f_w;\alpha)$
  end if
end if

또한, 스패닝트리의 초기 연결 설정 알고리즘은 다음과 같다.

if $\mathfrak{j}(h_\gamma, g)$ AND $\mathfrak{j}(h_\alpha, g)$ **then**
  set $f(\gamma) : p(g_i) \rightarrow m(f_d)$, $p(g_o^m) \rightarrow m(f_w;\alpha)$
  set $f(\alpha) : p(g_i) \rightarrow m(f_d)$, $p(g_o^m) \rightarrow m(f_w;\gamma)$
else
  if $\mathfrak{s}(h_\gamma, g)$ AND $\mathfrak{j}(h_\alpha, g)$ **then**
    set $f(\gamma) : p(g_o^u) \rightarrow m(f_w;\alpha)$
    set $f(\alpha) : p(g_i) \rightarrow m(f_d)$
  end if
end if

전체 토폴로지 지도의 관리, 인접할 노드의 선택, 플로우테이블의 결정 등 제어 평면과 관련된 임무는 관리서버에서 수행한 후 필요에 따라 각 프록시 노드에게 전달한다.

Summary

그림 3.3에 나타난 초기 스패닝트리의 구성, 스패닝트리의 확장 및 연결관계의 설정과 관련된 순서와 방법을 정리하면 다음과 같다.

- 그림의 ①: 관리서버가 초기 연결 설정 알고리즘을 수행한다. 관리서버는 최초로 수신된 $\mathfrak{j}(h_j, g)$와 $\mathfrak{s}(h_i, g)$에 대해서 연결 설정하며 알고리즘에 따라 플로우테이블을 구성한 후 N_i와 N_j에 전달한다. N_i와 N_j가 각각 $\mathfrak{s}(h_i, g)$, $\mathfrak{j}(h_j, g)$에 의한 요청을 했으므로 초기 연결 설정 알고리즘에 의해 N_i는 $f(i) : p(g_o^u) \rightarrow m(f_w; j)$를 받고 N_j는 $f(j) : p(g_i) \rightarrow m(f_d)$를 받는다. 이때, N_i와 N_j는 관계 $\eta_{i-\bar{j}}$를 갖는다.

- 그림의 ②: 새로운 호스트 h_k가 연동된 프록시 노드 N_k를 통해 그룹 g에 참가하기 위해 j(h_k, g)(IGMP join 메시지)를 보낸다. 노드 N_k는 관리서버에게 h_k의 그룹 g에 대한 참가요청을 알린다. 관리서버는 그룹 g에 대한 네트워크 토폴로지 지도를 통해 N_k가 연동되어야 할 프록시 노드를 선택한다. 노드의 선택은 지리적 위치, AS번호, 노드 degree 등의 정보를 이용하고[2], 관리서버가 $N_\alpha \in G_p$인 모든 프록시 노드들 중에 최적의 노드를 선택한다. 그림 3.3의 ①에서 $\{N_i, N_j\} \in G_t$이고 $\{N_j\} \in P_p$이므로 관리서버는 G_p에 속한 N_j를 리턴하고, 연결 설정 알고리즘을 통해 N_k와 N_j에 플로우테이블이 설정된다. N_k와 N_j는 η_{k-j}의 관계가 성립된다.

- 그림의 ③: 호스트 h_m이 N_m에게 j(h_m, g)를 보내면, N_m은 관리서버에게 플로우테이블을 요청한다. 관리서버는 $\{N_i, N_j, N_k\} \in G_t$이고 $\{N_j, N_k\} \in G_p$의 관계를 가지고 있으므로 N_m이 그룹 g에 대해서 연결되어야 할 프록시 노드는 $\{N_j, N_k\} \in G_p$에서 선택되어진다. N_k가 선택되어졌다고 하자. 플로우테이블은 연결 설정 알고리즘에 의해 결정되어 각 프록시 노드에게 전달된다. N_m은 η_{k-m}의 관계를 갖는다.

- 그림의 ④: 호스트 h_n이 그룹 g에 참가하지 않은 상태에서 $\mathfrak{s}(h_n, g)$한다면, 관리서버는 $\{N_j, N_k, N_m\} \in G_p$ 중 한 노드를 선택해 플로우테이블을 전달함으로써 프록시 노드 N_n과 연결관계를 맺는다. N_m이 선택되었다고 하자. h_n이 그룹 g에 참가하지 않았으므로 N_n은 $\eta_{n-\bar{m}}$의 관계를 갖는다. $\mathfrak{s}(h, g)$에 대해서 $\eta_{n-\bar{m}}$의 연결관계를 맺는 이유는 프록시 노드 N_m에서 그룹 g에 대해 참가하지 않고 있는 노드 N_n에 멀티캐스트 트래픽이 전송되는 것을 막기 위해서이다.

Join relation between a proxy node and end hosts

NetFPGA가 총 4포트의 기가비트 이더넷 인터페이스를 가지고 있으므로, 업링크 1포트를 제외한 나머지 3포트를 통해 종단 호스트가 연결될 수 있다. 다운링크 포트가 2계층 스위치와 연동된다면 다수의 호스트를 수용할 수 있다. 따라서, $\exists N_\alpha \in \{G_t - G_p\}$인 프록시 노드에도 새로운 호스트 $h_{\alpha'}$이 j$(h_{\alpha'}, g)$를 할 수 있다. 다음 그림 3.4는 그림 3.3에서 $\{h_{i'}, h_{j'}, h_{k'}\}$가 새롭게 j$(h, g)$ 또는 $\mathfrak{s}(h, z)$한 토폴로지 지도이다.

종단 호스트가 그룹 g에 참가하고 있는 N_j와 N_k는 각 신규 호스트 $h_{j'}$과 $h_{k'}$의 $\mathfrak{s}(h_{j'}, g)$나 j$(h_{k'}, g)$에 대해서 테이블의 갱신을 필요로 하지 않는다[3]. $\mathfrak{s}(h_{j'}, g)$나 j$(h_{k'}, g)$에 대한 요청이 관리서버로 전달된다. 관리서버에 정의된 규정에 따라 그룹 g에 대해 참가하지 않고 그룹 g에 송신하는 $h_{j'}$의 패킷 플로우를 차단(blocking)할 수 있다.

그룹 g에 참가하고 있는 종단 호스트가 없는 N_i에 신규 호스트 $h_{i'}$이 j$(h_{i'}, g)$ 메시지를 보낼때(IGMP join), 노드 N_i와 인접한 노드 N_j는 연결관계를 $\eta_{i-\bar{j}}$에서 η_{i-j}로 변경해야 한다.

이웃하는 두 프록시 노드 N_γ와 N_α에서 종단 호스트의 그룹 g에 대한 상태 변화에 따른 상태 갱신(status update) 알고리즘은 다음과 같다. 먼저, 종단 호스트 $h_{\gamma'}$이 프록시 노드 N_γ에게 j$(h_{\gamma'}, g)$ 요청을 했다고 가정하다. 호스트 상태 변화에 의해 플로우테이블이 변경되어야 할 기타 경우는 존재하지 않는다. 프록시 노드 N_γ가 그룹 g에 참가하고 있는 종단 호스트를 가지고 있지 않을 경우에만 플로우테이블의 갱신이 필요하다. 관리서버는 N_γ에 "다운링크 포트로 플러딩"$(p(g_i) \rightarrow m(f_d))$, N_α에게는 "그룹 g를 갖는 패킷 플로우를 N_γ로 포워

[2]노드 선택과 관련된 부분은 본 문서에서 다루지 않는다.
[3]관리서버에 의해 $\mathfrak{s}(h_{j'}, g)$ 등의 요청이 거절되면 플로우테이블이 갱신되어야 한다.

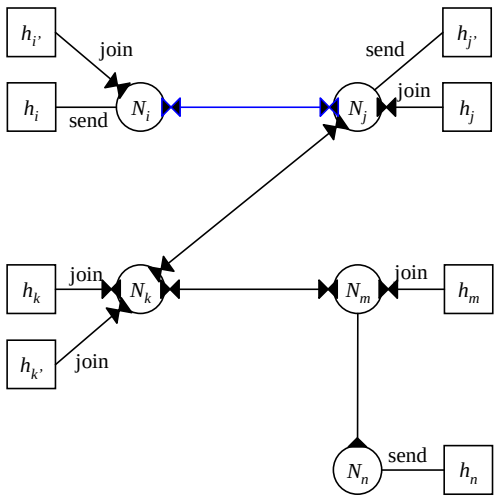

Figure 3.4: Status Update.

딩"($p(g_o^m) \rightarrow m(f_w;\gamma)$, $p(g_i) \rightarrow m(f_w;\gamma)$)과 관련된 플로우테이블을 각각 전달한다.

if $h_{\gamma'}$ issues j$(h_{\gamma'}, g)$ **then**
  if $\kappa_\gamma = 0$ **then**
    set $f(\alpha) : p(g_o^m) \rightarrow m(f_w;\gamma)$, $p(g_i) \rightarrow m(f_w;\gamma)$
    set $f(\gamma) : p(g_i) \rightarrow m(f_d)$
  end if
end if

3.3.2 Group Leave

그룹의 탈퇴는 IGMP leave 메시지 또는 타이머를 통해 이루어진다. 임의의 패킷 플로우에 대한 플로우테이블은 {RULES, ACTION, STATISTICS}를 유지하며 STATISTICS에 타이머를 포함하고 있다. 즉, 동일한 RULES를 갖는 패킷 플로우가 특정시간 이상 프록시 노드에서 처리되지 않을 경우에 타이머가 만료(expire)된다. 프록시 노드는 만료된 RULES에 속한 패킷 플로우를 묵시적 그룹 탈퇴로 간주하고 해당 {RULES, ACTION, STATISTICS}를 플로우테이블에서 삭제한다. 타이머에 의한 탈퇴 트리거링(triggering) 시 관리서버에게 그룹 탈퇴가 보고된다. 프록시 노드에서 발생하는 그룹 참가 및 탈퇴 등과 관련된 정보는 모두 관리서버에게 전달되므로, 관리서버는 그룹 g에 속한 모든 프록시 노드들 간의 연결관계를 알고 있다. 본 문서에서는 타이머에 기반한 그룹 탈퇴를 다루지 않는다.

그림 3.5은 그림 3.4에서 호스트 $\{h_{i'}, h_j, h_{k'}, h_m\}$이 그룹 g에서 탈퇴했을 때 최종적으로 만들어지는 토폴로지 지도이다. 먼저, 종단 호스트의 잦은 그룹 이탈(churn-in/churn-out)로 인한 테이블 갱신 요구를 막기 위해 그룹 탈퇴 요청은 관리서버에서 인위적으로 지연 처리한다. 즉, 타이머나 IGMP leave 메시지에 의한 그룹 탈퇴는 관리서버에게 요청된 후 일정시간이 경과된 후에 처리된다. 종단 호스트 h_α가 그룹 탈퇴를 요청할 경우 연동된 프록시 노드 N_α는 관리서버에게 플로우테이블의 갱신을 요청한다. 관리서버는 요청한 노드 N_α의

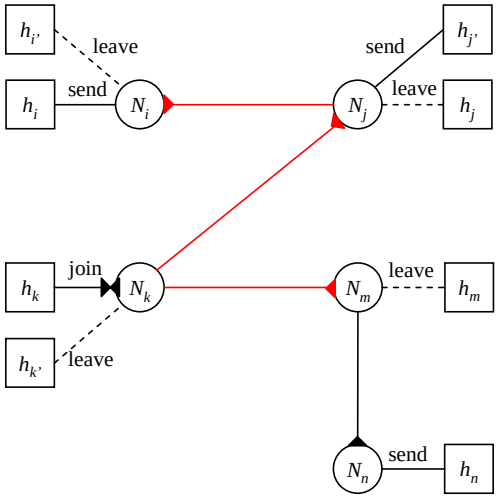

Figure 3.5: Group Leave.

κ_α와 μ_α를 바탕으로 그룹 g에 대한 스패닝트리를 갱신한다.

요청된 시점에서 $\kappa_\alpha = 1$이고(즉, 그룹 탈퇴를 요청한 호스트를 제외하고 해당 그룹에 참여하고 있는 종단 호스트가 존재하지 않을 경우) $\mu_\alpha = 1$일 경우, 이웃한 노드 N_γ로 단방향성 연결 관계을 맺는다. $\kappa_\alpha = 1$이고 $\mu_\alpha \geq 2$일 경우, 노드 N_α는 이웃하는 노드들과의 연결관계는 유지하지만 그룹 g에 해당하는 패킷 플로우들이 다운링크 포트로 플러딩되는 것을 막는다.

프록시 노드 N_α에서 그룹 탈퇴 알고리즘은 다음과 같다[4]. N_γ를 노드 N_α의 이웃 프록시 노드라고 가정한다.

if h_α issues $\mathfrak{l}(h_\alpha, g)$ **then**
  if $\kappa_\alpha = 1$ **then**
    if $\mu_\alpha \leq 1$ **then**
      delete $f(\alpha) : p(g_i) \rightarrow m(f_d)$
      delete $f(\gamma) : p(g_o^m)$ or $p(g_i) \rightarrow m(f_w; \alpha)$
    else
      if $\mu_\alpha \geq 2$ **then**
        delete $f(\alpha) : p(g_i) \rightarrow m(f_d)$
      end if
    end if
  end if
end if

그림 3.4의 노드 N_i과 N_j를 예로 들어 설명한다. 먼저, 노드 N_i는 종단 호스트 h_i'의 $\mathfrak{l}(h_{i'}, g)$ 요청 메시지를 관리서버에게 전달하고 관리서버는 노드 N_j의 κ_j와 μ_j가 각각 1개씩이므로 위 알고리즘에 따라 N_j의 $m - type$ ACTION 중 다운링크 플러딩과 관련된 부분을 삭제한다(delete $f(i) : p(g_i) \rightarrow m(f_d)$). 또한, 프록시 노드 N_j에서 노드 N_i로 트래픽이 흐르는 것을 막기 위해, N_i의 $m-type$ 플로우테이블 중 노드 N_j로의 패킷 포워딩 명령을 삭제한다(delete $f(j) : p(g_o^m)$ or $p(g_i) \rightarrow m(f_w; i)$).

[4]Multicast MAC table은 그룹 g에 참가하고 있는 종단 호스트들의 수를 알고 있다.

최종적으로 관리서버는 $\|G_p\| = 0$ (즉, 스패닝트리를 구성하는 모든 프록시 노드들이 그룹 g에 대해서 참가하고 있는 종단호스트를 가지고 있지 않을 때) 인 경우, 집합 G_t에 포함되는 모든 프록시 노드들을 스패닝트리에서 제거하고 해당 노드들에게 그룹 g에 대한 플로우테이블 삭제를 명령한다[5].

3.3.3 Optimization

그룹 g에 대한 토폴로지 지도가 최종적으로 그림 3.6의 파란색과 같이 최적화 될 수 있지만, 본 연구에서는 복잡도와 전송 품질 등을 고려해 그룹 g가 유지되는 동안 최적화를 수행하지 않는다. $N_i \in \{G_t - G_p\}$가 그룹 g에 더 이상 패킷을 보내지 않을 때는 타이머에 기초해 연결을 해제한다.

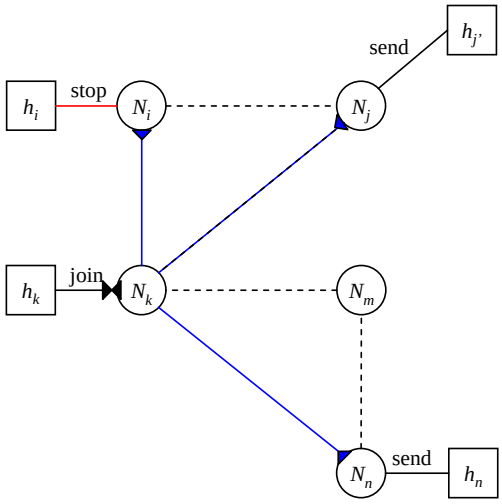

Figure 3.6: Optimization.

[5] 관리서버와 프록시 노드 또는 노드와 노드 간의 통신 프로토콜은 본 문서에서 다루지 않는다.

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.