http://dx.doi.org/10.3745/KIPSTC.2011.18C.6.413

서비스 오버레이 네트워크의 구성을 위한 NetFPGA 기반의 IP 서비스 게이트웨이

조 진 용[†] · 이 소 연[††] · 공 정 욱[†] · 김 종 원[†††]

요 약

오버레이 네트워크는 인터넷 연결성을 손상시키지 않고 새로운 네트워킹 기능을 제공할 수 있는 현실적인 방안이다. 본 논문은 서비스 오버레이의 용이한 구성을 지원하는 사용자 정의 네트워크 서비스를 소개하고 구성된 오버레이 네트워크 상에서 고속 데이터 평면 처리를 수행하는 서비스 게이트웨이의 하드웨어 및 소프트웨어 구조를 살펴본다. 또한, 멀티캐스트 응용 지원을 위한 서비스 오버레이 네트워크의 생성 과정을 예시한다. 국가과학기술연구망에 구축된 실험환경에서 서비스 게이트웨이의 성능 평가를 수행하고 수백 Mbps급 멀티미디어 서비스의 적용 가능성을 확인한다.

키워드 : 사용자 정의 네트워크, 프로그램 가능 오버레이 네트워크, NetFPGA

A NetFPGA-based IP Service Gateway for the Composition of Service Overlay Networks

Jinyong Jo[†] · Soyeon Lee[††] · JongUk Kong[†] · JongWon Kim[†††]

ABSTRACT

Overlay network is a ready-to-use solution to enable new network functionality with existing Internet connectivity intact. This paper introduces a network service which helps users easily compose their own service overlay networks through software-defined networks. We look into the structure of service gateway which enables 1 Gbps packet processing on composed overlay networks. We also provide examples for the way of composing service overlay for support multicast applications. Experiment results carried over the KREONET (Korea Research Environment Open NETwork) show the forwarding performance of the service gateway.

Keywords : Software-Defined Network, Programmable Overlay Network, NetFPGA

1. 서 론

향후 영상 데이터가 전체 인터넷 트래픽의 90% 이상을 점유할 것으로 예상된다[1]. IP 멀티캐스트는 영상 트래픽의 다지점 분배를 위한 효율적인 자원 활용 방안이다. 하지만, 프로토콜의 높은 복잡도로 인한 네트워크 관리의 문제와 활용에 대한 사용자의 요구 부족 등으로 인해 광범위한 네트워크 배치(network deployment)에 실패했다.

본 논문은 "P2P나 iPhoneTM의 앱스토어(App store)와 같은 사용자 기반의 기술 개발 모델을 멀티캐스트에 적용하면?" 이라는 물음에서 출발한다. 두 모델은 클라우드 서비스의 PaaS(Platform as a Service)나 IaaS(Infrastructure as a Service)와 개념 상 유사한 서비스 인프라를 필요로 한다. 공급자는 기본 인프라를 제공하며 수요자는 스스로의 요구에 따라 기술을 개발하고 결과물을 활용해 수익을 창출한다.

사용자에 의해 네트워크 기술이 혁신되기 위해서 서비스 플랫폼은 다양한 종단 응용들을 수용하고 유연한 기술 검증이 가능한 구조를 가져야 한다. 오버레이 네트워크는 기존 인터넷 환경을 기반으로 신규 네트워크 서비스를 용이하게 개발할 수 있는 장점이 있다. 하지만, 기존의 오버레이 네트워크들은 특정 응용에 종속적으로 동작하며 호스트 PC를 활용해 구축되기 때문에 가용성 및 성능에 한계를 가지고 있다.

본 논문은 IP 멀티캐스트 응용들의 지원과 신규 네트워크 기술 검증을 위한 Teamworks(TEstbed for Advanced Multicast netWORK Services)를 소개하고 데이터 평면 처리를 수행하는 서비스 게이트웨이의 구조를 살펴본다.

† 정 회 원 : 한국과학기술정보연구원 선임연구원
†† 정 회 원 : 한국과학기술정보연구원 연구원
††† 종신회원 : 광주과학기술원 정보통신공학부 교수
논문접수 : 2011년 6월 28일
수 정 일 : 1차 2011년 9월 16일, 2차 2011년 10월 17일
심사완료 : 2011년 10월 24일

Teamworks는 사용자 정의 네트워크(software-defined network)에 기초한 서비스 플랫폼이다. 패킷 플로우 단위의 서비스 오버레이를 구성하기 때문에 다수의 종단 응용들을 동시에 수용할 수 있는 장점을 가지고 있다. 또한, 플로우 처리를 담당하는 데이터 평면을 NetFPGA[2] 하드웨어를 이용해 구성함으로써 시스템 병목을 최소화했다.

본 연구는 사용자 정의 네트워크를 통해 오버레이 네트워크의 실시간 구성이 가능한 최초의 통합 플랫폼이라는 점에서 의의를 갖는다. 특히, NetFPGA 하드웨어 가속에 의한 패킷 플로우의 처리는 1:1 및 1:N 데이터 통신을 지원하고 수백 Mbps의 패킷 처리 속도를 갖기 때문에 고대역폭을 요구하는 다양한 멀티미디어 응용의 서비스에 활용될 수 있을 것으로 기대한다.

본 논문의 2장에서는 유사 연구들을 조사하고, 3장에서는 프레임워크를 총람하며 4장에서 서비스 게이트웨이의 구성 요소들에 대해서 자세히 살펴본다. 5장에서 활용 예시를 통해 프레임워크의 적용 방법을 설명하고 6장에서 성능 평가를 수행한다. 마지막으로, 7장에서 결론을 맺는다.

2. 관련 연구

미래 인터넷 기술 검증을 위한 다양한 플랫폼과 새로운 네트워크 및 서비스 구조에 대한 연구들이 활발히 진행 중이다. 본 절에서는 제안된 프레임워크와 유사한 기능적 구조를 갖는 연구들을 소개한다.

RMCP(Reliable Multicast Control Protocol)는 응용계층 멀티캐스트를 관리하기 위한 제어 프로토콜이다[3]. 멀티캐스트 프로토콜의 복잡도를 줄이기 위해 IP 멀티캐스트의 제어 평면을 부분적으로 분리했다. 제어 프로토콜을 적용할 프레임워크는 중앙제어기와 중앙제어기의 통제를 받는 오버레이 노드로 구성된다. 프레임워크의 구성 요소가 Teamworks와 동일하지만 적용 분야가 멀티캐스트 응용의 제어 프로토콜 처리에 한정되는 한계가 있다. 중앙제어기 및 오버레이 노드 등은 구현되어 있지 않고 개념 모델만 제공된다.

OpenFlow는 2계층 이더넷 스위치에서 얻어진 개별 트래픽 플로우를 사용자 정의에 의해 처리하는 특징을 갖는다[4]. 제어 평면이 분리되어 있고 개별 플로우를 처리하는 특징으로 인해 신규 프로토콜 및 서비스 검증을 위한 테스트베드로 활용되고 있다. 제어 평면은 중앙제어기인 NOX (Network Operating System)[5]에 의해 관리되며 스위치는 사용자에 의해 정의된 정책에 따라 패킷 플로우를 처리한다. 프레임워크는 중앙제어기, 제어 프로토콜 및 스위치로 구성된다.

OpenFlow는 사용자 정의 네트워크를 이용하고 스위치가 개별 패킷 플로우를 처리한다는 점에서 본 연구와 매우 유사하다. 사용자 정의 네트워크는 전화망에서 시작된 서비스이고 개별 패킷 플로우의 처리는 보안 장비 등에서 이미 사용되는 기술들이다. OpenFlow는 플로우 단위의 네트워크 가상화 구조를 가지며 스위치들 간에 2계층 연동이 요구된다. 하지만, 제안된 프레임워크는 오버레이 노드들 간에 IP 터널링을 이용해 논리링크가 구성되며 오버레이 네트워크 상에서 1:1 및 1:N 데이터 통신을 수행하는 특징을 갖는다.

프로그램 가능 오버레이 라우터는 네트워크 서비스 제공자가 신규 네트워크 서비스를 실험 또는 제공하기 위해 구성하는 오버레이 네트워크이다[6]. 기존 IP 라우터에 SRL(Service Routing Layer)이 추가되어 오버레이 네트워크의 기능을 수행한다. 프로그램이 가능한 오버레이 네트워크를 구성할 수 있다는 점이 제안된 프레임워크와 유사하다. 하지만, DHT(Distributed Hash Table)를 이용해 라우팅을 수행하는 등 제어 평면이 분산되어 있고 서비스 제공자 중심의 오버레이 네트워크로써 종단 사용자들의 프레임워크 접근이 제한되는 차이점을 갖는다.

〈표 1〉 프레임워크의 비교

연구 내용	제어 평면	지원계층	사용자 정의	논리 링크
RMCP	중앙	IP 멀티캐스트	-	○
OpenFlow	중앙	이더넷 이상	○	-
프로그램 가능 오버레이 라우터	분산	-	-	○
Teamworks	중앙	IP 이상	○	○

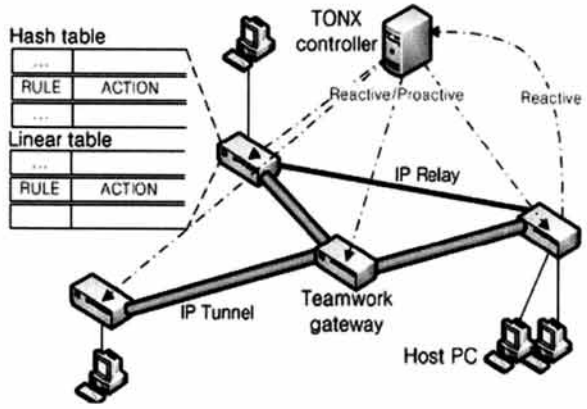
관련 연구들을 <표 1>에 비교 정리한다. Teamworks는 기술 개선의 대상이 IP 멀티캐스트이고 오버레이 네트워크를 통해 멀티캐스트 서비스를 제공한다는 점에서 RMCP와 유사하다. 또한, 사용자 정의 네트워크로써 제어 평면과 데이터 평면이 분리되어 있고 플로우 단위의 가상화 네트워크를 제공한다는 점에서 OpenFlow와 매우 유사한 구조를 가지고 있다. Teamworks는 중앙제어기, 프로토콜 및 서비스 게이트웨이를 활용하고 사용자 정의에 의해 논리링크를 구성하는 오버레이 네트워크로써, 기존 인터넷 환경을 활용해 IP 멀티캐스트 프로토콜의 처리가 가능하다는 점에서 OpenFlow 등과 상이하다. 또한, 패킷 플로우의 처리 등을 위한 메시지 프로토콜, 오버레이 네트워크에 특화된 소프트웨어 및 하드웨어 구조 등에서 관련 연구와의 차별성을 갖는다.

3. 프레임워크 개요

Teamworks는 사용자 정의 네트워크로써 (그림 1)과 같이 중앙제어기인 TONX(Teamworks Overlay-Network OS), 서비스 게이트웨이(Teamworks gateway) 및 메시지 프로토콜인 TWCP(TeamWorks Control Protocol)로 구성된다. 중앙제어기는 개별 패킷 플로우에 대한 사용자 정책을 관리하며 게이트웨이는 중앙제어기에 정의된 플로우 처리정책에 따라 논리링크를 구성하고 패킷 플로우를 고속 처리한

다. 서비스 게이트웨이는 오버레이 노드의 역할을 수행하는 2계층 스위치로써 다수의 종단 사용자들을 수용할 수 있다.

Teamworks는 세션 관리를 용이하게 하기 위해서 플로우 단위의 패킷 처리를 수행한다. 하나 이상의 동일한 특성을 갖는 패킷들의 집합을 플로우로 정의한다. 세션 단위의 자원 분할을 통해 플로우를 처리함으로써 각 논리 영역은 가상화 네트워크로 활용된다.



(그림 1) 프레임워크 개요

사용자에 의해 정의된 플로우 처리정책은 순향성 또는 반응성 메시지를 통해 게이트웨이에게 할당된다. 반응성 메시지 처리는 게이트웨이의 명시적인 요청에 의해 플로우 처리정책을 할당하는 방식이다. 게이트웨이는 테이블에 등록되지 않은 신규 패킷이 수신되면 중앙제어기에게 플로우 처리정책을 요청한다. 순향성 메시지 처리는 게이트웨이의 명시적 요청이 없는 상태에서 게이트웨이들에게 플로우 처리정책을 할당하는 방법이다. 플로우 처리정책을 할당받은 게이트웨이는 해당 정책을 내부 테이블에 저장한 후 대응되는 패킷 플로우가 입력될 때 저장된 정책을 적용한다. 메시지 프로토콜은 UDP(User Datagram Protocol) 또는 TCP (Transmission Control Protocol)를 기반으로 동작한다.

Teamworks의 중앙 제어형 구조는 확장성과 유연성 간의 이익 상쇄를 내포한다. 중앙제어기가 SPOF(Single Point Of Failure)이기 때문에 프레임워크의 신뢰도가 다소 낮고, 게이트웨이들이 송신하는 메시지들이 중앙제어기에 집중되므로 확장성의 문제가 존재한다. 하지만, 트래픽 및 플로우 모니터링 등에 필요한 높은 정밀도와 간결한 프레임워크를 유지함으로써 관리의 용이성을 확보할 수 있다. 또한, 플로우 단위의 가상화가 가능하므로 다수의 네트워크 서비스를 동시에 지원할 수 있는 등 높은 유연성을 갖는다.

Teamworks는 오버레이 네트워크의 규모를 실감형 응용을 이용하는 수십~수백 호스트 이하로 제한하고 기존 인터넷 프로토콜의 처리와 플로우 관리를 게이트웨이와 중앙제어기에 분산시킴으로써 확장성을 유지한다. 확장성 확보를 위한 추가적인 고려 사항으로, 네트워크 뷰(network-wide view)가 동기화[12]된 다수의 제어기를 사용하거나, 중앙제어기의 일부 기능을 스위치에 이식[13]하는 방법 등을 고려할 수 있다.

게이트웨이는 중앙제어기가 할당한 플로우 처리정책에 기초해 해당되는 플로우들을 1Gbps 회선속도로 처리한다. 게이트웨이는 NetFPGA 하드웨어부인 FLOW(Flow-based Overlay sWitch)와 FLOW를 제어하고 내부 테이블의 유지관리를 담당하는 SPONGE(Software to Process Overlay-Networked flows crossing service GatEway) 소프트웨어로 구성된다. 중앙제어기로부터 플로우 처리정책이 할당되면 게이트웨이는 해쉬 및 선형 테이블에 처리정책을 저장한 후 해당 플로우가 입력되면 저장된 처리정책을 적용한다.

서비스 게이트웨이들은 IP 변환(IP translation) 또는 IP-in-IP[7] 터널링(IP tunneling)을 통해 데이터그램을 상호 교환한다. IP 변환은 주소 변환을 통해 점대점(point-to-point) 중계되는 패킷 전달 방식이다. IP-in-IP 터널링은 단편화(fragmentation) 문제를 발생시킬 수 있지만, 원 패킷의 헤더 정보를 손상시키지 않고 패킷 경로를 임의 설정할 수 있는 장점이 있다. 게이트웨이가 패킷 단편화와 재조합을 지원하지 않기 때문에 종단 호스트의 MSS(Maximum Segment Size)를 설정해 단편화의 문제를 해결한다.

서비스 게이트웨이가 인식하는 플로우는 IP 데이터 패킷에 한정된다. 기존 2계층 프로토콜인 ARP(Address Resolution Protocol)의 처리나 3계층 IGMP(Internet Group Management Protocol) 프로토콜 등은 중앙제어기의 처리정책에 영향을 받지 않는다. 즉, IP 이외의 2계층 프레임 및 제어 목적의 3계층 패킷들은 SPONGE를 통해 스위칭되거나 프로토콜 서비스가 제공된다. 사용자는 IP 패킷에 대해 추가적인 여과 규칙을 정의할 수 있으며 여과된 IP 패킷은 플로우 처리정책을 적용받지 않는다. 게이트웨이가 처리하는 패킷 플로우의 범위를 좁힘으로써 메시지 교환을 위한 네트워크 부하 및 중앙제어기의 처리 부하를 감소시킬 수 있다. 네트워크 및 중앙제어기의 부하 감소는 프레임워크의 확장성을 높인다.

4. 서비스 게이트웨이의 구성 요소

사용자는 서비스 게이트웨이, 중앙제어기 및 메시지 프로토콜을 이용해 오버레이 네트워크를 구성할 수 있다. 본 논문에서는 데이터 평면의 핵심 구성요소인 서비스 게이트웨이를 중심으로 내부 구조를 살펴본다.

4.1 플로우 테이블

플로우 테이블은 개별 패킷 플로우들의 처리정책을 저장한다. 개별 플로우를 처리함으로써 논리적인 자원 분할이 가능하다. 플로우 처리정책은 사용자에 의해 정의되며 중앙제어기가 게이트웨이에게 할당한다. 게이트웨이는 할당된 플로우 처리정책을 수행한다. SPONGE와 FLOW는 플로우 테이블의 각 항목에 저장된 정책에 따라 패킷 플로우를 처리한다.

플로우 테이블은 정합하는 방법에 따라 Exact Match를 위한 해쉬 테이블과 Wildcard Match를 위한 선형 테이블로

구분된다. 해쉬 테이블은 개별 패킷 플로우에 대한 독립된 처리정책을 유지하며 선형 테이블은 동일한 처리정책을 갖는 패킷 플로우들의 집합을 플로우 항목으로 유지한다. 개별 플로우 항목은 정합규칙, 계수정보, 처리규칙으로 구분된다. 게이트웨이의 하드웨어부인 FLOW는 정합규칙에 따라 수신된 플로우를 판별하고 처리규칙을 적용한 후 계수정보를 갱신한다.

〈표 2〉 플로우의 정합규칙

F_ID	In Port	IP			TCP/UDP	
		SA	DA	Protocol	SRC	DST

<표 2>는 정합규칙의 구조를 보여준다. 정합규칙은 F_ID(Flow IDentifier), 입력 포트, TCP/IP 5-tuple 등 총 7-tuple로 구성되며 각 플로우 항목은 총 144 비트를 필요로 한다. 정합규칙은 패킷의 원 송신자와 논리링크 상의 송신자를 구분할 수 있도록 설계되었다. 입력 포트(In Port)는 물리 MAC 포트, CPU 포트, 오버레이 네트워킹을 위한 논리 포트를 포함한다. 32 비트 F_ID는 논리링크로 연결된 패킷 송신자의 IP 주소로 설정된다. 계수정보는 플로우 단위의 모니터링 정보를 제공하고 만료된 플로우를 관리하는데 이용된다. 계수정보에 할당된 공간은 총 64 비트이다. 마지막으로, 처리규칙은 정합되는 패킷 플로우의 처리 방법을 지시한다.

처리규칙은 플로우 처리를 위해 부가적으로 필요한 정보(예, MAC 주소, 출력포트, 이웃 게이트웨이 IP 주소 등)를 포함하며 오버레이 네트워크 상에서 패킷 플로우의 다지점 전송이 가능한 구조로 설계되어 있다. 처리규칙의 저장을 위해 총 96 비트가 필요하다.

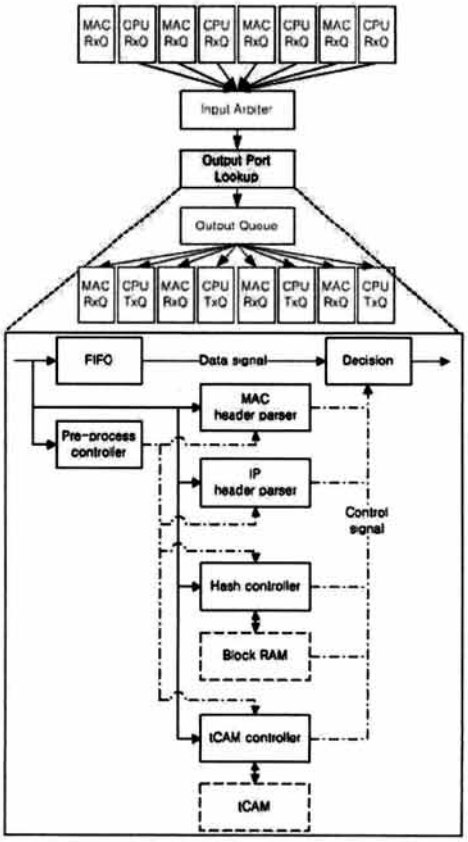
〈표 3〉 기초 요소: 처리규칙의 정의

Primitives	Description
BYPASS	수신되는 IP 또는 IP-in-IP 패킷을 처리규칙에 정의된 출력포트로 포워딩
DISCARD	수신된 패킷의 폐기
POINT	수신되는 IP 또는 IP-in-IP 패킷을 처리규칙에 정의된 이웃 게이트웨이에 전송
WRAP	수신되는 IP 패킷을 IP-in-IP 캡슐화
CLONE	수신되는 IP 또는 IP-in-IP 패킷을 처리규칙에 정의된 이웃 게이트웨이들에게 전송
PEEL	수신되는 IP-in-IP 패킷을 IP 역 캡슐화 한 후 처리규칙에 정의된 출력포트로 포워딩

패킷 플로우의 처리규칙은 <표 3>과 같이 BYPASS, DISCARD, POINT, WRAP, CLONE, PEEL 등 총 6가지 프리미티브들의 조합으로 구성된다. BYPASS는 입력된 패킷 플로우를 1개 이상의 출력포트로 포워딩한다. SPONGE와 연계해 2계층 스위치의 기능을 수행할 수 있다. DISCARD는 수신된 패킷 플로우를 폐기하는 규칙으로 트래픽 감시나 보안 관련 기능에 적합하다. 단위시간 당 트래픽의 총량이 많은 UDP 플로우의 기본 처리규칙이며 인증되지 않은 멀티캐스트 패킷의 그룹 내 유입을 방지하는데 사용될 수 있다. POINT는 IP 변환을 통해 패킷 플로우를 점대점 전송하기 위해 사용된다. POINT를 통해 전송할 경우, 패킷 플로우가 IP-in-IP 캡슐화(encapsulation)되지 않기 때문에 프레임 단편화의 문제가 발생하지 않는다. 멀티캐스트-유니캐스트 연동에 이용된다. 마지막으로, 오버레이 네트워크 상에서 IP-in-IP 터널을 구성하고 패킷을 전달하기 위해서 WRAP, CLONE, PEEL이 사용된다. WRAP은 IP 패킷의 캡슐화, CLONE은 수신된 IP 또는 IP-in-IP 패킷의 복제, PEEL은 IP 패킷의 역 캡슐화(de-encapsulation)를 수행한다.

4.2 하드웨어 기반 플로우 처리

서비스 게이트웨이의 하드웨어부인 FLOW는 1Gbps 회선 속도로 패킷 플로우의 처리정책을 실행한다. 사용자는 플로우 처리정책을 정의하며, 중앙제어기를 통해 정의된 정책을 게이트웨이에게 할당한다. 할당된 처리정책은 게이트웨이 제어 소프트웨어인 SPONGE가 관리하며, 정책의 실행은 FLOW가 담당한다. SPONGE는 플로우 처리정책을 FLOW에 등록하거나 삭제한다. FLOW는 수신되는 패킷의 헤더 정보와 입력포트 등 메타데이터를 이용해 정합규칙을 조합한 후 플로우 테이블에 저장 중인 항목들과 비교해 정합여부를 판단한다.



(그림 2) 출력포트 검색 모듈의 설계

PlanetLab[8]과 같이 PC 기반의 오버레이 노드들은 낮은 패킷 처리 성능으로 인해, 요구 대역폭이 높은 인터넷 응용의 서비스 제공에 한계가 있다. FLOW는 NetFPGA 하드웨어를 이용해 오버레이 노드의 성능병목을 최소화했다. NetFPGA는 4개의 1Gbit/s 입출력 포트를 갖는 FPGA(Filed Programmable Gate Array) 기반의 프로그램이 가능한 네트워크 인터페이스로써 Xilinx Vertex II Pro XC2VP50를 포함하고 있다. FLOW는 IPv4 Reference Router를 기반으로 구현되었으며 OpenFlow 스위치가 참조되었다.

(그림 2)는 IPv4 Reference Router의 출력포트 검색(Output Port Lookup) 모듈에 구현된 FLOW의 세부 모듈들을 보여준다. FLOW에 이더넷 프레임이 수신되면 선처리제어기(Pre-process controller)는 프레임의 계층별 세그먼트를 구분하고, 연계된 세부모듈로 제어 신호를 보내 해당 세그먼트의 헤더 정보를 해석하게 한다. 헤더해석기(Header parser)는 이더넷 헤더와 IP 헤더의 세부 필드 정보를 얻는다. 조합된 세부 필드 정보는 플로우 처리정책 중 정합규칙의 일치 여부를 확인하는데 이용된다.

플로우의 정합 여부는 해쉬 제어기와 TCAM(Tenary CAM) 제어기에 의해서 확인된다. Exact Match를 위한 플로우 항목들은 Block RAM(BRAM)에 저장되며 Wildcard Match 항목들은 TCAM에 저장된다. Exact Match에는 2개의 다항식을 갖는 CRC-32(Cyclic Redundancy Check-32) 기반의 해쉬 함수가 이용된다. Decision 모듈은 수신된 패킷에 대한 처리 방법을 결정한다. 수신 패킷이 테이블에 보유중인 플로우 항목의 정합규칙과 일치할 때는 해당 플로우 처리정책의 처리규칙이 실행된다. 수신된 2계층 프레임의 이더넷 type이 IP가 아니거나 플로우 처리정책이 테이블에 존재하지 않을 경우에는 SPONGE로 이동된 후 스위칭되거나 프로토콜 서비스를 제공받는다.

플로우의 처리정책을 저장하기 위해 해쉬 테이블과 선형 테이블이 사용된다. 해쉬 테이블은 총 4,096개의 플로우 처리정책을 수용하고 선형 테이블은 16 항목을 저장할 수 있다. 한 개의 플로우 항목을 읽기 위해 최소 4 클럭 사이클이 필요한 64 비트 SRAM(Static RAM)에 비해 1 클럭 사이클에 읽기가 가능한 BRAM을 활용함으로써 상태 기계를 간소화 했다. FLOW는 오버레이 네트워크에서 논리링크로 연결된 이웃 노드들의 정보를 저장하기 위해 동등(peer) 테이블을 유지한다. 동등 테이블은 레지스터를 이용해 구성되었으며 총 32개의 이웃 노드들에 대한 IP 정보를 갖는다.

NetFPGA의 한정된 하드웨어 자원으로 인해 각 테이블이 수용할 수 있는 플로우 항목들의 수가 제한된다. 이는 게이트웨이들의 네트워크 배치 시 가용성 문제를 유발시킬 수 있다. 특히, 다수의 패킷 플로우들이 동시에 게이트웨이로 유입될 경우 해쉬 충돌확률이 높아질 가능성이 있다. 하지만, 오버레이 네트워크의 특성 상 연동되는 종단 호스트의 수는 제한적이다. 즉,게이트웨이로 유입되는 패킷 플로우의 총 수는 2/3계층 스위치에 비해 상대적으로 적다. 또한, 부족한 자원을 효과적으로 활용하기 위해 Cukoo hashing[9]등을 적용하면 충돌 문제에 대처할 수 있다.

FLOW는 수신된 패킷의 다지점 복제와 전송을 위해서 동등 테이블을 이용한다. 동등 테이블은 논리링크로 연결된 이웃하는 게이트웨이들의 IP 정보를 저장한다. 동등 테이블의 색인 정보는 플로우 처리정책 중 처리규칙의 인접 지점 비트맵과 1:1로 대응된다. 인접 지점 비트맵은 32 비트이며 최대 16개의 이웃하는 게이트웨이들을 지칭할 수 있다. 즉, 게이트웨이는 수신된 하나의 패킷을 복제해 최대 16개 이웃하는 게이트웨이들에게 전송할 수 있다. IP 멀티캐스트 라우터의 노드 차수가 도메인 내부에서는 평균 1.3~1.6이며, 도메인 간에는 2.1~2.5[10]인 점을 감안하면 게이트웨이는 6배 이상의 노드 차수를 갖는다.

〈표 4〉 XC2VP50 하드웨어 활용도

Resources	Hardware Utilization	Utilization Percentage
Slices	18,023 out of 23,616	76%
4-input LUTS	23,242 out of 47,232	49%
Flip Flops	19,788 out of 47,232	41%

Xilinx™ ISE(Integrated Software Environment)에 의해 보고된 XC2VP50 하드웨어의 활용도는 <표 4>와 같다.

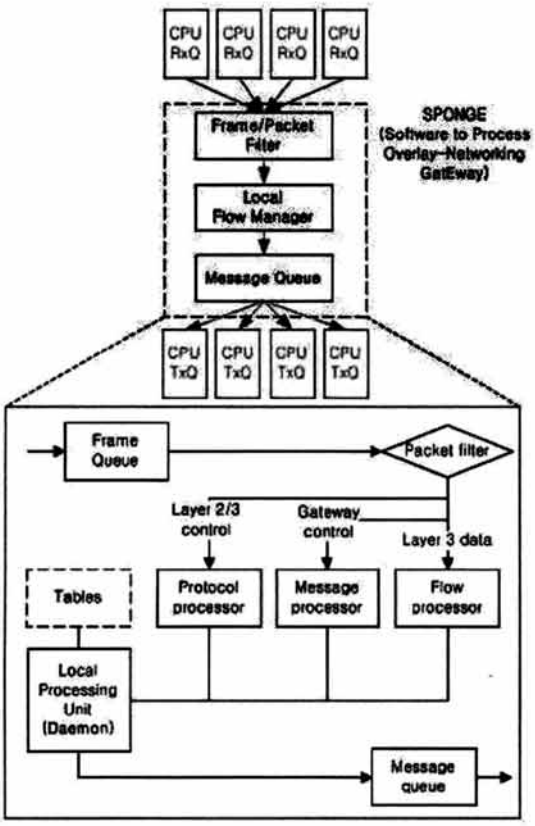
4.3 소프트웨어 기반 플로우 처리

게이트웨이 제어 소프트웨어인 SPONGE의 주 역할은 FLOW와 중앙제어기 사이에서 상호 인터페이스를 제공하는 것이다. 제어 소프트웨어의 기능으로는 FLOW가 전달한 2계층 프레임의 분석 및 스위칭, 인터넷 프로토콜 서비스의 제공, 플로우 테이블의 유지 관리, FLOW 제어 및 TWCP 프로토콜의 처리 등이다.

(그림 3)은 SPONGE의 내부 구조도를 보여준다. LPU(Local Processing Unit)는 SPONGE의 주 프로세스로써 FLOW가 CPU queue를 통해 전달한 2계층 프레임의 처리를 총괄 관리한다. 관리하는 부 프로세스로 메시지 처리기(Message processor), 프로토콜 처리기(Protocol processor) 및 플로우 처리기(Flow processor)를 갖는다.

2계층 이더넷 프레임이 FLOW를 통해 CPU 수신버퍼에 입력되면 패킷 필터(Packet filter)는 제어 패킷과 데이터 패킷을 분리한다. 기존 인터넷 프로토콜(예, ARP, IGMP, ICMP 등)의 서비스를 위해 필요한 패킷들과 TWCP 메시지들이 제어 패킷으로 구분된다. 여과 규칙에 정의된 IP 패킷들은 제어 패킷으로 분류된다.

메시지 처리기는 TWCP 프로토콜을 해석하거나 신규 메시지를 생성한다. SPONGE와 중앙제어기 간 메시지 교환을 위해 LwIP(Lightweight TCP/IP stack)[11] 기반의 TCP/UDP 프로토콜이 이용된다. 프로토콜 처리기는 ARP나 IGMP와 같은 2/3 계층 인터넷 프로토콜을 제공한다. 게이트웨이 내부에서 인터넷 프로토콜 서비스를 제공함으로써



(그림 3) 제어 소프트웨어의 내부 구조도

네트워크 및 중앙제어기의 처리 부하를 낮출 수 있다. 플로우 처리기는 테이블에 저장된 플로우 항목들을 관리한다. 테이블에 등록되지 않은 신규 플로우가 입력되면 2계층 프레임 헤더의 목적지 주소로부터 128 바이트를 분리한 후 중앙제어기에게 해당 플로우의 처리정책을 요청한다.

인터넷 프로토콜들을 지원하기 위해서 SPONGE는 IGMP 및 ARP 테이블을 보유한다. 또한, 플로우 항목 등의 관리를 위해서 해쉬 테이블, 선형 테이블 및 동등 테이블을 관리한다. 서비스 게이트웨이의 하드웨어 부인 FLOW는 소프트웨어 부인 SPONGE가 관리하는 해쉬 테이블과 동등 테이블의 복사본을 갖는다. 또한, FLOW는 선형 테이블의 진부분집합을 하드웨어에 유지한다. SPONGE는 사용자에 의해 정의된 플로우 처리정책의 적용 및 갱신을 담당한다.

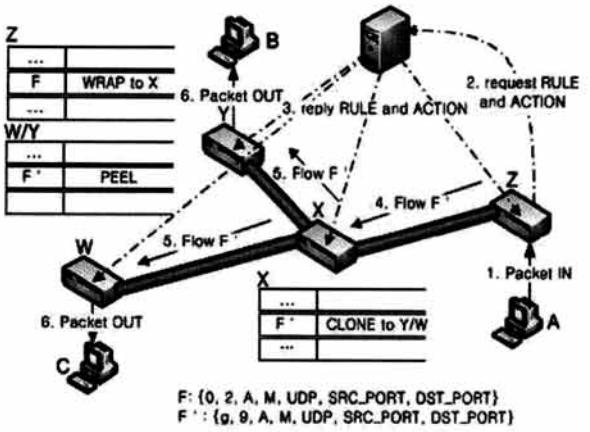
하드웨어 자원의 제약으로 인해 FLOW가 선형 테이블에 저장할 수 있는 플로우 항목의 수는 16개로 제한된다. 문제 해결을 위해 SPONGE는 선형 테이블을 정적 Wildcard Match가 요구되는 항목들과 동적 Wildcard Match를 위한 항목들로 구분해 관리한다. 정적 Wildcard Match는 FLOW의 선형 테이블에 등록되어 "don't care" 정합을 수행하는 방법이다. 동적 Wildcard Match가 요구되는 플로우 항목은 SPONGE의 선형 테이블에 임시 저장된 후 해당되는 패킷 플로우가 수신되면 FLOW의 해쉬 테이블에 재등록되어 Exact Match가 수행된다. 동적 Wildcard Match는 서비스 게이트웨이 내부에서 처리정책의 복제를 수행하므로 중앙제어기와의 메시지 교환 횟수를 줄일 수 있다.

WAN 환경에서 게이트웨이가 반응성 메시지를 통해 패킷 플로우의 처리정책을 할당 받는데 필요한 시간은 LAN 환경에서 보다 크다. 신규 패킷 플로우의 수신 게이트웨이가 g_i이고 중앙제어기까지의 메시지 전송 시간을 t_i ($t_i = RTT/2$, RTT는 왕복 시간)라고 하면, 이동 경로에 n개의 게이트웨이들이 존재하는 패킷 플로우는 경로 설정에 최대 $t_i + t_m$의 시간이 필요하다. t_m은 $\max(t_i, \ldots, t_{n-i})$이다. 결과적으로, 게이트웨이 g_i는 최소 $2t_i$ 시간 동안 플로우 처리정책을 할당받지 못한다. 또한, 패킷의 이동 경로상에 위치한 게이트웨이 g_j가 $t_j - t_i > \sum_{k=i}^{j-1} t_{e_k}$ $(j \le n)$인 상황이 발생할 경우, 플로우 처리정책을 중앙제어기로부터 할당받기 전에 해당 패킷이 수신된다. t_{e_k}는 게이트웨이 g_k에서 오버레이 경로 상의 다음 홉 g_{k+1}까지의 패킷 전송 시간이다. 수신되는 패킷의 도착 간격 t_a가 $2t_i$보다 작을 경우에 플로우 처리정책을 할당받기 위해 게이트웨이는 중앙제어기에게 $\lfloor 2t_i/t_a \rfloor$ 개의 메시지를 전송해야 한다. 문제 해결을 위해 SPONGE는 플로우에 대한 기본 처리정책을 유지한다. 예를 들어, UDP 플로우는 중앙제어기로부터 처리정책을 할당 받을 때까지 폐기(DISCARD)를 수행한다.

5. 플로우 처리 시나리오

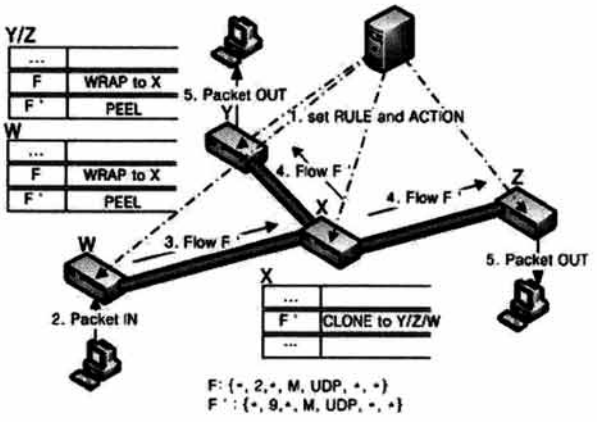
본 절에서는 중앙제어기와 서비스 게이트웨이 사이의 TWCP 메시지 교환, 플로우 항목의 테이블 저장 및 해당 패킷 플로우의 처리 절차 등을 살펴본다. 멀티캐스트 스패닝 트리와 루트기반 멀티캐스트 $1:N$ 트리의 생성 과정을 예시한다. 종단 호스트의 집합 $H:\{A,B,C\}$, 게이트웨이의 집합 $G:\{W,X,Y,Z\}$ 및 중앙제어기로 네트워크 환경이 구성된다. N은 G의 차수 의해서 정의되며 $N \le |G|$ 이다. 집합 G에 속한 게이트웨이들은 등록된 플로우 항목을 갖지 않는 초기 상태이다.

(그림 4)는 반응성 메시지 처리와 Exact Match를 통한 루트기반 멀티캐스트 $1:N$ 트리의 구성 방법을 보여준다. 먼저, $h \in H$인 임의의 호스트 h는 특정 멀티캐스트 그룹 M에 참가를 요청하고 중앙제어기는 해당 그룹 정보를 유지한다. 종단 호스트 A가 F로 명시된 패킷 플로우를 발생시키면 게이트웨이 Z는 중앙제어기에게 정합규칙과 처리규칙의 할당을 요청한다. 중앙제어기는 기 정의된 플로우 관리정책에 따라 경로 상의 모든 게이트웨이들에게 플로우 F 또는 F'에 대한 정합규칙과 처리규칙을 전달한다. F'은 F를 바탕으로 게이트웨이가 재정의한 플로우 항목이다. $\exists g_i, g_j \in G, (g_i \neq g_j)$에 대해 g_i에서 g_j로의 방향성 논리링크를 e_{ij}라고 하면, 게이트웨이 g_i에서 g_j로 이동하는 패킷 플로우 F'의 F_ID는 e_{ij}에 의해서 정의된다. 즉, e_{zx}와 e_{xz}에 의해 정의되는 F'은 서로 다른 F_ID를 갖는다.



(그림 4) 루트기반 멀티캐스트 $1:N$ 트리

F와 F'으로 정의된 패킷 플로우의 처리정책은 SPONGE와 FLOW의 해쉬 테이블에 각각 저장된다. 게이트웨이로 유입되는 플로우 F와 F'은 플로우 테이블에 저장된 항목의 처리규칙에 따라 처리된다. F'이 목적지 호스트와 연결된 출력 게이트웨이에 도착하면 역 캡슐화한 후 지정된 물리 포트로 출력한다. 하나의 멀티캐스트 세션 내에서 $N:N$ 데이터 통신을 위해서는 최대 N개의 루트기반 멀티캐스트 $1:N$ 트리가 요구된다.



(그림 5) 멀티캐스트 스패닝 트리의 구성

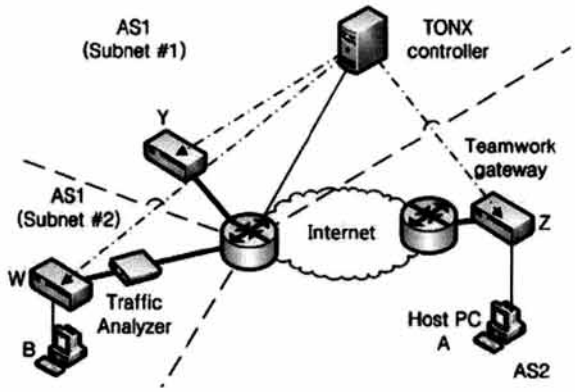
(그림 5)는 순향성 메시지 처리와 Wildcard Match를 통해 멀티캐스트 스패닝 트리를 구성하는 방법을 보여준다. 중앙제어기는 임의의 플로우 F나 F'에 대해서 오버레이 네트워크를 구성하고 플로우 처리정책을 관련된 게이트웨이들에게 선 할당한다. 할당된 플로우 처리정책은 게이트웨이의 SPONGE와 FLOW가 유지하는 선형 테이블들에 각각 저장된다. 서비스 게이트웨이가 플로우 F나 F'을 처리하는 절차는 루트기반 멀티캐스트 $1:N$ 트리의 예와 동일하다.

Exact Match를 이용한 루트기반 멀티캐스트 $1:N$ 트리에서 $N:N$ ($N \leq |G|$) 통신을 위해서 게이트웨이는 최대 $2N$개의 플로우 항목을 유지해야 한다. 하지만, Wildcard Match를 이용하면 최대 2개의 플로우 항목으로 멀티캐스트 스패닝 트리를 구성할 수 있다. Wildcard Match를 이용할 경우, 게이트웨이가 처리해야 하는 플로우 F'은 F_ID에 의해 구분될 필요가 없다. 즉, 게이트웨이들은 동일한 플로우 정합규칙 F'을 공유한다.

(그림 5)의 게이트웨이 X와 같이 차수 D가 $D \geq 2$인 게이트웨이는 오버레이 네트워크 상에서 라우팅 루프를 방지함으로써 테이블에 저장되는 플로우 항목의 개수를 줄일 수 있다. FLOW가 라우팅 루프를 방지한다. 게이트웨이 X는 W가 전달한 플로우 F'에 대해서 "W, Y, Z로 복제"를 수행해야 한다. 하지만, 라우팅 루프의 방지를 위해서 오버레이 네트워크 상에서 송신 게이트웨이인 W로는 패킷을 보내지 않고 Y, Z 로만 전송한다.

6. 성능 검증

서비스 게이트웨이는 플로우 기반의 오버레이 네트워크를 구성하고 논리링크들을 통해 패킷 전송을 수행하는 프레임워크의 핵심 구성요소이다. 서비스 게이트웨이에서 성능병목이 발생하면 패킷 손실이 유발되기 때문에 고대역폭을 요구하는 멀티미디어 응용의 서비스 품질이 저하된다. 본 절에서는 서비스 게이트웨이의 플로우 처리 성능을 평가하고 멀티미디어 서비스의 적용 가능성을 확인한다.

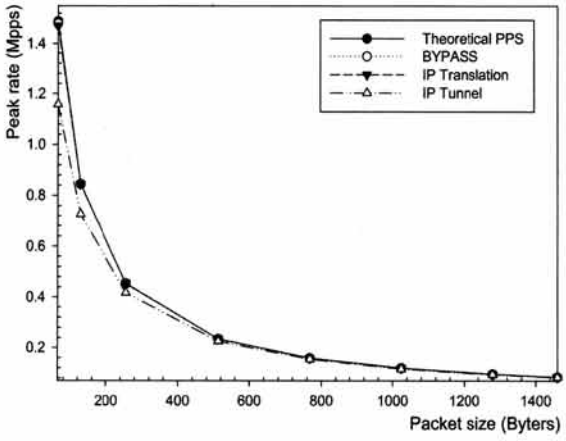


(그림 6) 실험 환경

(그림 6)과 같이 국가과학기술연구망(KREONET)에 실험환경이 구축되었다. 실험 환경은 2개의 자율시스템(Autonomous system)과 3개의 서브넷을 갖도록 구성되었다. 게이트웨이 W에 연결된 호스트 B는 하드웨어 기반의 트래픽 발생기이며 호스트 A는 트래픽 수신기이다. 게이트웨이 Y는 패킷 플로우의 중계를 담당한다.

서비스 게이트웨이의 패킷 처리 및 전송 성능을 검증하기 위해서 자율시스템 AS1에서 로컬 테스트를 수행한다. 중앙제어기는 순향성 메시지를 이용해 게이트웨이 Z에게 WRAP과 CLONE의 조합으로 구성된 플로우 처리규칙을 선 할당한다. 해당 플로우 처리규칙은 가장 복잡한 프리미티브들의 조합으로 FLOW에서 하나의 패킷을 처리하기 위해 총

14 클럭 사이클이 요구된다. 결과적으로, 게이트웨이 Z는 B로부터 입력되는 UDP 패킷 플로우를 복제한 후 처리규칙에 지정된 게이트웨이들에게 1:1 또는 1:N전송한다. 게이트웨이 Z에 연동된 트래픽 분석기(Traffic Analyzer)를 통해 해당 패킷 플로우의 계수 정보 등을 분석한다.



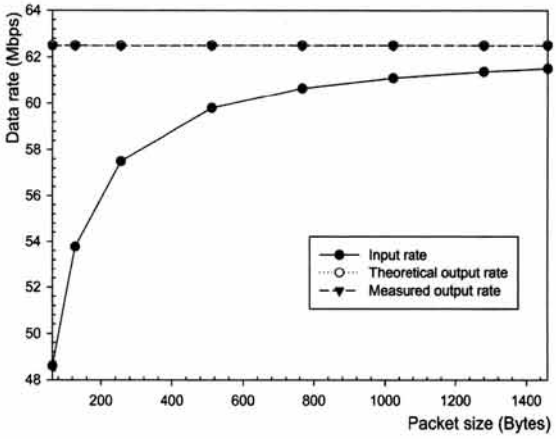
(그림 7) 패킷 크기 별 최대 패킷 처리율

(그림 7)은 게이트웨이에서 수신된 패킷이 1:1 전송될 때 패킷 크기 별 최대 처리율을 측정한 결과이다. 패킷 크기는 이더넷 헤더와 32 비트 CRC 필드를 포함하지만 프레임 프리앰블과 IPG(Inter-Packet Gap)는 제외된다. 프리앰블과 IPG는 총 20 바이트 크기를 갖는다. 그러므로 전송선로 상의 패킷 크기는 20 바이트가 추가된다. 서비스 게이트웨이는 패킷이 수신되면 BYPASS, IP 변환 또는 IP-in-IP 터널링을 수행하고 트래픽 분석기는 최대 처리율을 측정한다.

수신되는 패킷의 처리규칙이 각각 IP 변환 또는 BYPASS일 경우, 최대 패킷 처리율은 이론치(Theoretical PPS(Packets Per Second))에 근접한다. 최대 처리율은 64 바이트 패킷에 대해 1.480 Mpps (757 Mbps)이며 1460 바이트 패킷은 약 0.084 Mpps(969 Mbps)로 측정되었다. 수신되는 패킷 플로우의 처리규칙을 IP-in-IP 터널링으로 변경하면 64 바이트 패킷에 대해 최대 처리율이 1.158 Mpps로 나타났다. IP 터널링은 캡슐화를 위해 24 바이트 패킷 헤더를 추가하므로 캡슐화되지 않은 패킷에 비해 약 0.322 Mpps의 처리율 감소가 나타난다.

오버레이 네트워크를 통해 IP 멀티캐스트를 서비스하기 위해 서비스 게이트웨이는 1:N 전송을 지원한다. 성능 평가를 위해 중앙제어기는 게이트웨이 W의 플로우 처리규칙으로 IP-in-IP 터널링을 할당한다. 게이트웨이 W는 논리링크로 연결된 16개의 서비스 게이트웨이들에게 수신된 플로우를 복제 후 전송한다. (그림 8)은 패킷의 크기 별 복제 성능을 측정한 결과이다.

서비스 게이트웨이에서 복제 후 전송되는 패킷들의 출력 데이터율(Data rate)이 1 Gbps가 되도록 입력 데이터율(Input rate)을 설정했다. 출력되는 패킷은 캡슐화가 되기 때문에 입력되는 패킷에 24 바이트 헤더가 추가된다. 결론적

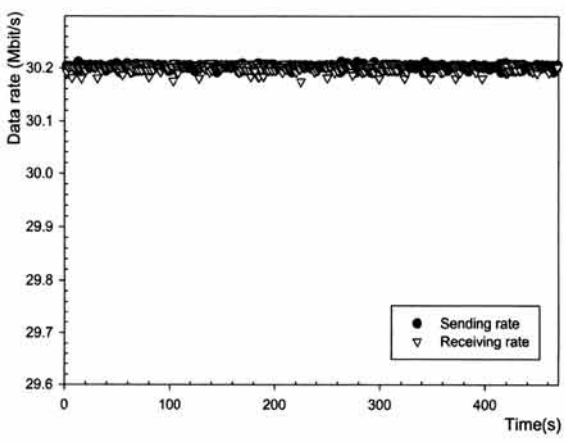


(그림 8) 패킷 복제 성능

으로, 게이트웨이가 16개의 논리링크들로 패킷 플로우를 전송한다면 하나의 논리링크는 이론적으로 최대 62.5 Mbps의 데이터율을 갖게 된다(1 Gbps = 16×62.5 Mbps). (그림 8)에서 측정된 출력 데이터율(Measured output rate)이 이론치(Theoretical output rate)와 일치하는 것을 확인할 수 있다. 즉, 서비스 게이트웨이는 1:N 전송을 위한 패킷 복제 시 성능 감쇄가 발생하지 않는다. (그림 8)의 입력 데이터율과 측정된 출력 데이터율의 차이는 입력되는 패킷에 추가되는 24 바이트 오버헤드로 인해 발생된다.

IP 라우터는 패킷 헤더의 필드값이 유효하지 않거나 CRC 오류가 존재할 경우 해당 패킷을 폐기한다. 또한, 헤더의 필드값은 라우터의 포워딩 성능에 영향을 줄 수 있다. 예를 들어, IP 헤더의 IHL(IP Header Length) 필드값이 5가 아닐 경우 IP 라우터는 처리 부하로 인해 포워딩 성능이 하락한다[14]. 서비스 게이트웨이들을 상호 연결하는 논리링크는 다수의 IP 라우터들과 물리링크로 구성되므로 패킷 헤더의 유효성이 확보되어야 WAN 환경에서 서비스를 제공할 수 있다.

헤더 유효성을 검증하기 위해서 호스트 B는 A로 멀티미디어 패킷을 전송한다. 해당 패킷 플로우는 게이트웨이 W에 의해 IP 캡슐화되고 Y에 의해 중계된다. 게이트웨이 Z



(그림 9) 멀티미디어 응용을 활용한 헤더 유효성 검증

는 역 캡슐화를 수행한 후 최종적으로 호스트 *A*에게 전달된다. (그림 6)의 인터넷 영역은 기간네트워크 구간으로써 백그라운드 트래픽이 존재한다. 기간네트워크에 영향을 주지 않는 범위 내에서 패킷 플로우를 발생시켜 헤더 유효성을 검증했다. 호스트 *B*는 DVTS[15](Digital Video Transport System) 소프트웨어를 활용해 약 30.2 Mbps의 비디오 트래픽을 발생시킨다.

(그림 9)는 호스트 *A*에서 수신된 비디오 트래픽의 데이터율을 1초 간격으로 측정한 결과이다. 캡슐화된 IP 패킷이 기간네트워크를 거쳐 종단 호스트에 도착하는 것을 알 수 있다. 즉, 게이트웨이에 의해 추가된 패킷 헤더는 유효하며 IP 라우터에 의해 정상적으로 처리됨을 확인할 수 있다.

7. 결 론

본 논문은 서비스 오버레이 네트워크의 용이한 구성을 지원하는 사용자 정의 네트워크 서비스를 소개했다. 또한, 하드웨어 가속을 통해 패킷 플로우를 고속 처리를 수행하는 서비스 게이트웨이의 하드웨어 및 소프트웨어 구조를 살펴보았다. 플로우 기반의 오버레이 네트워크를 구성하고 프레임워크의 확장성과 패킷 플로우의 처리 성능을 높임으로써 다양한 성능 요구를 갖는 종단 응용들을 수용할 수 있다. 서비스 게이트웨이는 1 Gbps의 회선속도를 갖는 네트워크 환경에서 이론치에 근접하는 1:1 및 1:N 전송 성능을 보임으로써 고대역폭을 요구하는 멀티미디어 응용의 IP 멀티캐스트에 활용될 수 있을 것으로 기대한다.

참 고 문 헌

[1] Cisco VNI, "Forecast and Methodology, 2009-2014," June, 2010. White paper.

[2] J. Lockwood, N. NcKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, "NetFPGA-An Open Platform for Gigabit-rate Network Switching and Routing," Proc. IEEE International Conference on Microelectronic Systems Education, 2007.

[3] S. Koh, J. Park, J. Min, and K. Park, "Framework of Control Protocol for Relayed Multicast," Proc. Human.society@internet, pp.576-581, 2003.

[4] N. NcKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," ACM SIGCOMM Computer Communication Review 38, April, 2008.

[5] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: toward an operating system for networks," ACM SIGCOM Computer Communication Review, Vol.38, July, 2008.

[6] B. Davie and J. Medved, "A Programmable Overlay Router for Service Provider Innovation," Proc. ACM PRESTO'09, Aug., 2009.

[7] C. Perkins, "IP encapsulation within IP," Internet Engineering Task Force, RFC 2003, 1996.

[8] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: an overlay testbed for broad-coverage services," ACM SIGCOMM Computer Communication Review, Vol.33, July, 2003.

[9] R. Pagh and F. F. Rodler, "Cuckoo hashing," Journal of Algorithm, Vol.51, pp.122-144, May, 2004.

[10] R. C. Chalmers and K. C. Almeroth, "On the topology of multicast trees," IEEE/ACM Trans. on Networking (TON), Vol.11, Feb., 2003.

[11] A. Dunkels, "Design and implementation of the LwIP TCP/IP stack," Feb., 2001. http://www.sics.se/~adam/lwip/doc/lwip.pdf

[12] A. Tootoocian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," Proc. INM/WREN workshop. 2010.

[13] M. Yu, J. Rexford, M. J. Freedman, J. Wang, "Scalable flow-based networking with DIFANE," ACM SIGCOMM Computer Communication Review, Vol.40, Oct., 2010.

[14] P. Fransson and A. Jonsson, End-to-end measurements on performance penalties of IPv4 options, Proc. IEEE Globecom 2004, Dec., 2004.

[15] A. Ogawa, K. Kobayashi, K. Sugiura, O. Nakamura, and J. Murai, "Design and implementation of DV stream over internet," Proc. Internet Workshop IWS'99, pp.255-260, Feb., 1999.

조 진 용

e-mail : jiny92@kisti.re.kr

1999년 전남대학교 컴퓨터공학과(학사)

2002년 광주과학기술원 정보통신공학과 (공학석사)

2007년~현 재 광주과학기술원 정보기전공학부 박사과정

2003년~현 재 한국과학기술정보연구원 선임연구원

관심분야 : 멀티미디어 시스템 및 서비스, 오버레이 네트워크, 사용자 정의 네트워크

이 소 연

e-mail : soyeon.lee@kisti.re.kr

2009년 국민대학교 전자공학과(학사)

2010년~현 재 한국과학기술정보연구원 연구원

관심분야 : 미래인터넷, 서비스 오버레이 네트워크 등

공 정 욱

e-mail : kju@kisti.re.kr

1993년 한국과학기술원 전기 및 전자공학과(학사)

1998년 포항공과대학교 정보통신학과 (공학석사)

2008년 충남대학교 정보통신공학과 박사수료

1993년~2001년 (주)데이콤 중앙연구소 선임연구원

2001년~2002년 (주)맥스웨이브 책임연구원

2002년~현 재 한국과학기술정보연구원 선임연구원

관심분야 : 망 성능 분석, 망 자원 관리 등

김 종 원

e-mail : jongwon@nm.gist.ac.kr

1987년 서울대학교 제어계측공학과(학사)

1989년 서울대학교 제어계측공학과(석사)

1994년 서울대학교 제어계측공학과(박사)

1994년~1999년 공주대학교 전자공학과 조교수

1998년~2001년 Univ. of Southern California Dept. of Engineering - Systems 연구조교수

2001년~현 재 광주과학기술원 정보통신공학부 교수

관심분야 : Networked Media Systems and Protocols focusing "Dynamic Composition of Immersive Media-oriented Services over the Wire/Wireless IP Convergence Networks"