리눅스 기반 네트워크 스위치를 위한 GARP/GVRP 개발

노윤정[0] 이원석 김준우 강경태 김지현 신현식
서울대학교 컴퓨터공학부
{partemis,alnova2, kesker, nicola}@cselab.snu.ac.kr, shinhs@snu.ac.kr

Development of GARP/GVRP for network switches based on Linux

Youn Jung Ro[0] Won Seok Lee Junu Kim Kyungtae Kang Jihyun Kim
Heonshik Shin
School of Computer Science and Engineering, Seoul National University

요 약

가상랜(VLAN:Virtual LAN) 기술은 멀티 캐스트 트래픽의 감소 효과가 크고, 효율적인 네트워크 보안 및 관리를 지원하므로, 대부분 최근의 상용 스위치는 가상랜 환경에서의 동작을 지원하고 있다. 가상랜 환경에서 동작하는 스위치에서 가상랜 정보를 주고받기 위하여 사용하는 프로토콜이 GARP/ GVRP(Generic Attribute Registration Protocol/GARP VALN Registration Protocol)이다. 이 프로토콜은 네트워크 환경에서 가상랜 정보가 바뀔 때마다 동적으로 각 스위치의 가상랜 정보를 설정해 줌으로써 네트워크 관리자의 수고를 덜어주며, 효율적으로 가상랜을 관리하는 방법을 지원하는 것이다. 따라서 가상랜 환경에서 동작하는 스위치는 대부분 GARP/GVRP를 내장하고 있으나, 범용 소스가 없이 생산업체 별로 자체 구현한 GARP/GVRP를 각각의 스위치에 탑재하고 있다.

본 논문은 이렇게 가상랜 환경을 지원하는 스위치에 필수적인 프로토콜인 GARP/GVRP를 리눅스 커널에 삽입 가능한 모듈 형태로 개발하였다. 또 개발한 모듈을 기가빗 이더넷 스위치(Gigabit Ethernet Switch)에 이식된 리눅스 커널에 삽입, 동작시키고 공인된 테스트 집합을 이용하여 프로토콜이 정확하게 동작함을 검증하였다.

1. 서론

데이터 링크 계층(Data Link Layer)에서 동작하는 레이어 2 스위치(이하 스위치)는 네트워크를 다수의 세그먼트(segment)로 분할하고 세그먼트 당 사용자의 수를 현저하게 감소시켜 네트워크를 효율적으로 관리하는데 기여하는 바가 크다. 하지만 기존의 스위치는 네트워크의 멀티캐스트 트래픽을 관리하지 못했기 때문에 이를 보완하기 위하여 가상랜(VLAN:Virtual LAN) 기술이 도입되었다.

가상랜은 랜 세그먼트를 물리적인 위치와 관계없이 몇 개의 논리적인 도메인(Logical Domain)으로 분리하여 각 도메인을 하나의 랜 세그먼트로 관리하는 기술이다. 가상랜을 지원하는 환경에서는 각 가상 랜 도메인의 멀티캐스트 트래픽(Multicast Traffic)은 각각의 논리적 도메인 내부에서만 전파된다는 특징이 있다. 이에 따라 가상랜은 멀티캐스트 트래픽을 감소시키고 네트워크의 보안 및 관리를 용이하게 해 주는 효과가 있다.

스위치는 가상랜에서 동적으로 정보를 주고 받기 위해서 GARP/GVRP(Generic Attribute Registration Protocol/GARP VLAN Registration Protocol)을 사용한다. GARP/GVRP는 IEEE802.1d와 IEEE 802.1q에서 그 기본 동작을 정의하고 있지만, 스위치를 제작하는 회사마다 자체 구현하고 있으며 공개된 소스가 거의 없는 것이 현실이다.

본 연구에서는 GARP/GVRP를 리눅스 커널에 삽입 가능한 모듈 형태로 개발하고 검증하려 한다.

2. 관련 연구

데이터 링크 계층은 논리적인 연결을 관리하는 링크 제어 계층(Logical Link Control)과 MAC 계층으로 다시 나눌 수 있다. 스위치의 구조와 동작을 계층별로 구분해 보면, MAC 계층에서 동작하는 부분인 직접 프레임을 받고 보내는 포트 부분, 포트에서 받은 프레임을 차단, 전달, 전파할지 결정하는 MAC relay entity 부분과 LLC 계층에서 동작하는 Slow Protocol을 위한 특별한 프레임을 처리하는 LLC의 두 부분으로 나뉜다. LLC는 어느 프로토콜에서 사용하는 프레임인지를 프레임의 목적지 주소로부터 판단하고 적당한 프로토콜로 프레임을 전달하는 기능을 한다. GVRP가 사용하는 프레임의 목적지 주소로 01-80-C2-00-00-21가 예약되어 있다.

스위치는 물리적으로 연결된 선을 통해서 전달받은 데이터를 프레임 형태로 바꾸어, 목적지 MAC 주소를 알아내어 적당한 포트로 프레임을 전달하는 것을 기본 동작으로 한다. 이러한 스위칭을 위하여 스위치는 전달(forwarding), 차단(filtering), 전파(flood-ing)의 세 가지 동작을 수행한다. 이에 따라 각 포트 내부에서의 네트워크 통신은 다른 포트로 전달되지 않으므로 스위치의 포트에 따라 랜이 구분되는 것이다[3]. 스위치의 전파 동작은 목적지가 어느 포트에 연결되어 있는 지 알 수 없는 경우 수신된 포트를 제외한 포트에 모두 프레임을 보내는 것으로 멀티캐스트 트래픽을 유발시키는데, 이는 랜 내부의 트래픽을 증가시키는 큰 원인의 하나이다[1].

가상랜 기술은 랜 세그먼트를 포트에 관계없이 논리적인 도메인으로 묶는 것을 지원한다. 각 가상 랜 도메인의 멀티캐스트 트래픽은 도메인 내부에서만 전파되므로, 스위치에서 발생하는 멀티캐스트 트래픽을 줄이고 관리하는 데에 상당히 효과적이다. 따라서 가상랜 기술은 보다 보안을 강화시키고 효율적으로 네트워크를 관리할 수 있게 해 준다.

GVRP는 가상랜 정보를 네트워크에서 공유하기 위해, 각 가상 도메인에 대한 선언과 철회, 등록과 등록취소 정보를 가지고 있으며(GIDTT:GID Transition Table), 이는 GARP(Generic Attribute Registration Protocol)에서 정의한 하부구조(GID:GARP Information Distribution Protocol)를 이용하여 동작한다. GARP는 포트 별 정보를 유지하는 것 이외에도 스위치 내부에서 포트간의 정보 전달의 역할도 담당(GIP:GARP Information Propagation)하고 있다. GVRP는 그 밖에 가상랜 정보를 유지하기 위하여 스위치의 하드웨어와 연동하는 부분과 스위치 외부와 정보를 주고받기 위하여 특수한 프레임(PDU: Protocol Data Unit)을 만들어 내고, LLC로부터 전달 받은 PDU를 해석하여 내부의 정보를 수정하는 부분으로 구성된다. GVRP에서 사용하는 PDU는 01-80-C2-00-00-21의 예약된 멀티캐스트 주소를 목적지 주소로 사용한다.

3. GARP/GVRP의 개발

3.1 GARP/GVRP프로토콜 설계

GARP/GVRP 개발의 근간이 되는 것은 IEEE802.1d의 13장과 IEEE802.q의 11.4장에서 소개하고 있는 예제 코드이다. 그림1은 예제 코드를 기반으로 개발한 GVRP가 리눅스 커널이 포팅된 스위치에 이식하여 동작할 때의 구조를 나타낸 것이다.

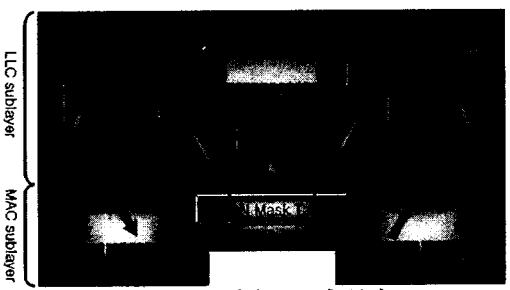

그림1.GVRP 내부 구조와 동작

LLC는 GVRP로 GVRP PDU를 넘겨주는 부분이며 실제 테스트 과정에서 LLC는 리눅스 커널의 디바이스 드라이버 형태로 동작한다. GVRP Participant는 스위치 디바이스 드라이버와 연동하는 하부 구조로 GVF(GVRP PDU Formatting)와 VFDB(VLAN Active Filtering Database interface)를 정의한다. 스위치 디바이스 드라이버의 LLC에서 PDU를 받고 보내는 부분이 GVF이며 VLAN Mask Table을 설정하는 부분과 연동하는 것이 VFDB이다. VFDB VLAN Mask Table의 mask/unmask 설정을 바꾸어 데이터 프레임을 처리하는 동작을 바꾼다. GVRP 내부의 동작을 정의하고 있는 부분이 GARP이며, 표준에서 GARP는 GID(GARP Information Distribution Protocol)과 GIP(GARP Information Propagation)의 하부 프로토콜로 구현되어 있다. GID는 포트 마다 하나씩 존재해서, 각 포트의 상태를 저장하고, 이벤트를 처리한다. 그리고 다른 포트로 전달해야 할 상태 변화가 일어났을 때에 GIP에서 정의된 함수를 실행시킨다. GIP는 포트들 사이의 상태 변화를 다른 포트로 전달하는 역할을 수행하는데, 포트의 개수에 관계없이 전체 GVRP Application에 하나만 존재한다. GVD(GVRP VLAN Database)는 GID에서 이용하는 색인을 실제 가상랜 도메인의 ID를 가리키는 값으로 대응시키는 가상랜 DB를 정의하고, 이 DB를 다루기 위한 함수들을 기술한다. 상용 스위치는 대부분 가상랜 도메인을 문자열로 정의할 수 있도록 지원한다. 따라서 본 연구에서는 문자열로 정보를 저장하기 위해서 DB구조체를 확장하고 이를 위한 함수들도 정의하였다. 본 연구에서 개발한 GVD는 스위치의 MAC Relay Entity 내부의 포트에 기본적으로 선언된 가상 랜 도메인 정보와 연동하여 가상 랜 도메인을 특정한 용도로 예약하는 작업도 지원하고 있다.

3.2 리눅스 커널에 의존적인 부분

GVRP Participant의 내부에서 리눅스 커널에 의존적인 부분은 크게 메모리 접근 부분과, 커널 타이머의 사용부분, 커널 메시지 출력부분으로 나눌수 있다.

메모리 접근은 <linux/slab.h>에서 정의된 kmalloc(), kfree()함수를 사용하여 구현하였다. 메시지 출력은 printk()함수를 이용하여 간단하게 작성하였다.

<linux/timer.h>에서 정의하고 있는 리눅스 커널 내부에서 사용되는 타이머 함수는 unsigned long형의 인자 하나만 전달 받을 수 있는 문제가 있다. 이 때문에, Jiq_data라는 구조체를 선언하여 구조체 안에 함수의 인자들과 함수 자체를 저장하기 위한 공간을 만들어 구조체의 주소값을 unsigned long형의 인자로 전달하도록 함수를 수정하였다. 프로토콜에서는 타이머가 포트마다 4개(join timer, hold timer, leave timer, leaveall timer)씩 있고 각 타이머가 완료되었을 때 호출하는 함수는 바뀌지 않는다. 또한 타이머가 완료될 때까지 타이머를 위한 데이터가 할당 받은 메모리 공간을 유지하기 위해 이후 설명할 gid 구조체에 Jiq_data와 타이머 구조체를 포함시켰다.

3.3 디바이스 드라이버와의 연동

GVRP Participant에서 디바이스 드라이버와 연동하는 부분은 GVF와 VFDB의 두 부분으로 나뉘어 각각 LLC와 VLAN Mask Table과 연동하여 동작한다.

LLC는 PDU의 목적지 MAC주소가 01-80-c2-00-00-21인 경우 GVRP PDU임을 인식하고 앞의 16bytes를 제거하는 대신 프레임을 입력 받은 1byte 크기의 포트 번호를 앞에 붙여 GVRP participant의 GVF로 넘겨준다.

GVF에서는 PDU를 효과적으로 다루기 위하여 Gvf 구조체와 Gvf_msg 구조체를 정의하여 사용하였다. Gvf 구조체는 GVRP PDU를 구조체의 형태로 바꾼 것에 port_no를 추가한 것으로 디바이스 드라이버의 LLC에서 넘겨 받은 PDU를 필드 별로 접근하기에 적당한 구조이다. Gvf_msg는 하나의 GVRP attribute를 저장하기 위한 구조체이다. attribute 길이는 0x04와 0x02만 가능한데 이 길이에 따라서 특정 가상랜 도메인에 JoinEmpty, JoinIn, LeaveEmpty, Empty 이벤트를 발생시킬 것인지 아니면 모든 가상 랜 도메인에 대한 LeaveAll 이벤트가 발생한 것인지를 알 수 있다.

VFDB(VLAN Filtering Database)는 GVRP의 내부 포트의 상태 변화에 따라 스위치 디바이스 드라이버에 접근하여 VLAN Mask Table의 내용을 설정한다.

3.4 GARP/GVRP의 내부 동작

GID는 도메인에 따른 포트의 상태 정보를 유지하고 이벤트를 처리하기 위한 함수들을 정의한다. GID는 포트

마다 gid구조체를 정의하고 있다. 이 구조체에서는 가상랜 도메인 정보를 저장하기 위한 gid_machine을 최대 가상랜 도메인 개수만큼 가지고 있다. Gid_machine은 등록(registration) 정보를 저장하고 있는 등록 테이블(registrar table)과 선언(declaration)정보를 저장하고 있는 신청 테이블(applicant table)로 구성된다. 각 등록 테이블과 신청테이블의 상태를 정의하고, 그 상태의 변화를 테이블로 정의하고 있는 부분이 GIDTT(GID Transtition Table)이다.

GIDTT는 각 테이블의 상태에서 이벤트가 발생하였을 때 어떤 상태로 전이 되는지에 대한 정보 역시 테이블로 갖고 있으며, 이전 상태와 발생한 이벤트로 테이블의 색인을 찾아 쉽게 다음 상태를 알아낼 수 있다.

포트마다 존재하는 GID는 다른 포트로 정보를 전달하기 위하여 GIP(GARP Information Propagation)를 사용한다.

3.5 가상랜 DB의 정의와 구성

GVD는 GID에서 사용하는 인덱스를 실제 가상 랜 도메인의 ID를 가리키는 값으로 연결시키는 역할만을 수행한다. 본 연구에서는 상용 스위치에서 일반적으로 사용되는 몇 가지의 기능을 추가 시켰다[9][10]. 가상 랜 DB는 Gvd구조체를 이용하여 정의 된다. 이 Gvd 구조체에서 identifier 변수가 있는데 이는 GVRP의 고유 기능을 위하여 정의된 것이며, 그 밖에 가상 랜 도메인을 문자열로 이름 붙일 수 있는 기능을 추가하기 위한 name 필드와 사용 중인지를 나타내는 used bit, 그리고 특정 용도로 가상랜 도메인을 예약하기 위한 reserved 필드가 있다. 표준에서 가상랜 ID는 12비트로 정의할 수 있으며 0과 4095는 사용할 수 없다. GVRP에서 Max_VLans 변수를 두어 Max_Vlans 범위에서 가상랜 ID를 사용하도록 하였기 때문에 Gvd 구조체를 생성할 때 Max_VLans - 2 만큼의 엔트리를 위한 공간을 할당받고, used필드를 이용하여 사용 중인지 아닌지의 여부만 표시하였다. 이 방법은 연결리스트에 비해서 메모리를 많이 사용하지만, 구조체 접근 방법이 간단하여 함수 구현을 편하게 해주는 장점이 있다. 그 밖에 reserved 필드의 값이 0이 아닌 경우 이 가상랜 도메인의 used필드를 늘1로 설정하여 포트별로 기본적으로 소속된 가상랜 도메인인 PVID(Port VLAN id)를 설정할 수 있도록 하였다.

3.6 GARP/GVRP의 동작

GVRP applicant 전체의 구조를 위한 gvr을 초기화과정은 내부의 하부 구조의 생성 및 그 구조의 초기화 과정으로 이루어진다. Gvr_create_gvr() 함수는 garp.h에 정의된 garp구조체와 gip, gvd구조체에 메모리 공간을 할당하고 초기화하는 작업을 수행한다. 반대로 gvr_destroy_gvr() 함수는 각 구조체가 점유한 메모리 공간을 해제한다. 또한 GARP의 내부 구조인 GID와 GIP의 동작을 GVRP와 연동시키기 위한 함수들과 PDU를 송수신하여 GVF를 이용하여 각 attribute의 이벤트 및 값을 GARP로 넘겨주는 함수가 정의되어 있다. GVRP의 전체동작의 흐름을 정리하기 위한 함수들과 시스템 호출과 사용자 인터페이스, 그리고 테스트를 위하여 정의한 함수들은 gvrp.c 파일에 구현하였다.

4. 테스트 집합을 이용한 테스트 결과

구현한 GARP/GVRP의 정확한 동작여부를 검사하기 위하여, 미국의 The University of New Hampshire의 InterOperability 연구실에서 작성한 Bridge Function Consortium GARP/GVRP Frame Format Test Suite를 사용하였다. 이 테스트 집합은 크게 네 가지 테스트 집합으로 구성되어 있으며 각 부분은 또 다시 몇 개의 그룹으로 구성되어 있다.

Interoperation test는 프로토콜의 내부 동작을 테스트하는 것으로 타이머의 동작과 이벤트 발생에 관한 동작, 상태의 천이 등을 검사하기 위한 테스트로 모두 통과하였다.

Frame test는 GARP/GVRP가 PDU를 정확히 만들어 내는지, 혹은 PDU를 받아 정확히 인식하는지를 테스트하는 것으로 역시 모두 통과하였다.

Fdb test는 가상 랜 DB와 관련된 동작을 테스트 하는 것으로 모두 통과하였다.

Multi-operation test는 다수의 이벤트를 담고 있는 PDU를 전달 받았을 경우 프로토콜이 제대로 동작하는지에 대한 테스트이며 역시 모두 통과하였다.

5. 결론

본 연구는 가상랜을 지원하는 스위치의 필수적인 프로토콜인 GARP/GVRP를 리눅스 커널에 삽입할 수 있는 모듈 형태로 개발한 내용을 담고 있다.

최근의 스위치는 데이터 프레임 처리 등의 기본 기능을 모두 하드웨어로 구현하여 속도의 향상을 꾀하고 있다. 하지만 GARP/GVRP와 같은 Slow Protocol의 경우에는 소프트웨어적으로 구현하는 경우가 많고 본 연구에서는 GARP/GVRP의 기능을 소프트웨어 적으로 구현하였다.

개발한 프로토콜은 실험실 환경에서 테스트 집합의 내용을 모두 통과하였지만, 실제로 많은 수의 스테이션과 여러 개의 스위치로 구성된 네트워크에서의 테스트가 부족하였다. 또한 사용자 인터페이스나 그 밖의 표준에서 정의하고 있지는 않지만 유용한 기능의 구현이 만족할 만큼 이루어지지 못하였다. 따라서 앞으로 지속적인 테스트와 프로그램 보완을 해나가야 할 것이다.

참고문헌

[1] Behrouz A. Forouzan, Catherine Ann Coombs, Sophia Chung Fegan, Data Communications and Networking(2nd Update Edition): International Edi, McGraw-Hill,2002

[2] ANSI/IEEE 802.1Q, 1998 Edition

[3] ANSI/IEEE 802.1D,1998 Edition

[4] Rich Seifert, The Switch Book:The Complete Guide to LAN Switching Technology, Wiley Computer Publishing,2000

[5] Gigabit Ethernet Switch Menual,2002

[6] Michael Beck, Harald Bohme, Mirko Dziadzka, Ulrich Kunitz, Robert Magnus & Dirk Verwornet, Linux Kernel Internals Second Edition, Addison-Wesley,1998

[7] Alessandro Rubini & Jonathan Corbet, Linux Device Drivers Second Edition, O' Reilly, 2001

[8] Linux Kernel Version 2.4.19, http://www.kernel.org

[9] Express 520T Switch Manual, Intel, 1999