리눅스상에서 IPv6 플로우 레이블을 이용한 포워딩의 구현

이주철° 안종석
동국대학교 컴퓨터공학과
{rune, jahn}@dgu.ac.kr

IP Forwarding Engine based on IPv6 Flow label in Linux

Joo-Chul Lee° Jong-Suk Ahn
Dept. of Computer Engineering, Dongguk University

요 약

IPv6는 IPv4의 단점을 보완한 차세대 인터넷 프로토콜이다. IPv6 헤더는 플로우 레이블 필드를 제공하여 레이블 값에 따라 차등적인 서비스를 제공할 수 있는 기반을 갖추었다. 본 논문에서는 이러한 레이블의 용도 외에 레이블을 이용한 포워딩 방법을 제안한다. 레이블을 이용한 포워딩은 기존의 MPLS와 유사한 방식으로 이루어진다. 두 방법의 차이점은, 첫째 MPLS는 속도를 빠르게 하기 위하여 대부분의 스위칭이 2 계층에서 이루어지지만 플로우 레이블을 이용한 방법은 3 계층에서 처리된다. 둘째, 패킷에 플로우 레이블을 기록할 때 기존의 IPv6 헤더 필드를 이용하기 때문에 MPLS에서와 같이 별도의 헤더가 필요치 않다. 또한 레이블과 플로우에 대한 서비스 정보를 매핑시킬 경우 좀더 수월하게 차등적 서비스를 지원할 수 있다. 본 논문에서는 IPv6 레이블 포워딩을 구현해서 실험한 실험 망과 현재 리눅스 커널(2.2.X)상에서 구현되어있는 IPv6의 포워딩 구조, 그리고 구현된 플로우 레이블 포워딩에 대하여 논한다.

1. 서론

인터넷이 처음 생긴 이래로 최근 몇 년간, 인터넷의 사용인구는 폭발적으로 증가해 왔다. 그에 따른 부수적 현상으로 처음엔 충분하리라 여겨졌던 인터넷(IPv4)의 여러 한계점들이 드러나기 시작했다. 최근에 인터넷에 접속되어 있는 호스트의 증가속도를 감안하면 현재의 인터넷이 수용할 수 있는 최대 주소의 개수인 2^{32}를 수년 내에 초과할 것으로 예상되며, 컴퓨터 이외의 지능형 가정기기와 인터넷 활용 통신 기기들도 인터넷에 접속되는 추세를 감안하면 현재의 인터넷은 빠른 속도로 포화상태에 이를 것이다.

이러한 문제를 해결하기 위한 방안이 몇 가지 제시되고 있는데, 부족한 IPv4의 주소 영역을 효율적으로 사용하기 위하여 무계층 주소(CIDR)[4] 개념의 도입, 인트라넷 등 내부의 모든 호스트가 외부와 통신할 필요가 없는 네트워크에서는 임시적 IP 주소를 고유의 IP 주소로 변환해 주는 NAT(Network Address Translation)[5] 등이 그것이다. 하지만 이러한 방법들은 모두 임시 방편적인 것들로 근본적인 해결책은 되지 못한다. 따라서 현재의 인터넷에 존재하는 한계점들을 극복하기 위해서는 궁극적으로 IPv6[1][8]를 사용해야 할 것으로 생각된다.

IPv4와 비교해 볼 때 IPv6는 몇 가지 특징을 갖는데, 그 중 하나로 플로우 레이블이 헤더에 포함된 점을 들 수 있다. 플로우 레이블은 패킷이 지니고 온 레이블 필드의 값에 따라 차별적인 서비스를 제공해주기 위한 용도를 예상하고 고안되었다. 즉 IPv6 라우터로 하여금 같은 플로우 레이블 값을 가진 패킷들에 대하여 특별한 처리를 해주도록 하는 것이다.

본 논문에서는 이러한 레이블의 용도에 추가하여 레이블의 값을 보고 포워딩 할 경로를 정해줄 수 있도록 하였다. 이 방법은 MPLS[6]의 포워딩 방식과 유사한 개념이다. MPLS와 본 논문에서 제안한 방법의 차이점은 MPLS가 2 계층에서의 스위칭을 위해 별도의 헤더[7]를 사용한데 비해, 여기서는 IPv6가 제공하는 플로우 레이블 필드를 활용했다는 점이다.

레이블을 이용한 포워딩은, 라우팅 엔트리를 결정하기 위해 IP 주소 중 가장 많은 부분이 일치되는 엔트리를 찾는 방법(Longest Prefix Match)을 기본으로 하는 IP 주소검색 포워딩에 비해, 짧은 레이블 값 전체가 일치하는 엔트리를 찾는 방법(Exact Prefix Match)을 원칙으로 하고있어 상대적으로 빠른 속도로 라우팅 테이블을 검색할 수 있다.

또한 기존의 IPv4에서 제공하지 못한 QoS를 플로우 레이블을 사용함으로써 제공할 수 있다. 즉, 레이블을 이용한 포워딩에서는 LDP(Label Distribution Protocol)에 의해 사전에 결정된 경로를 경유하므로 쉽게 QoS를 제공할 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 리눅스 커널2.2.X대에서 구현되어 있는 IPv6의 포워딩 구조에 대해 설명하고, 3절에서는 기존 커널 코드를 수정하여 플로우 레이블을 이용해 포워딩할 수 있도록 구현한 것에 대해 설명한다. 4절에서는 본 구현을 테스트한 실험망과 실험결과에 대하여 소개하고, 마지막으로 5절에서 결론과 향후 계획에 대해 기술하기로 한다.

2. 리눅스에서의 IPv6 포워딩

현재 리눅스 커널[9][10][11] 2.2.X에는 완벽하지는 않지만 IPv6코드가 구현되어 있다. 리눅스상에서의 IPv6 구현 상태는 [12]에서 참조할 수 있다.

2.1 FIB (Forwarding Information dataBase)

리눅스에서 라우팅에 관련된 정보는 FIB(Forwarding Information dataBase) 자료구조로 표현된다. IPv4와 마찬가지로 IPv6도 FIB를 만들어 사용하고 있지만 IPv4의 그것과는 구조가 조금 틀리다. 그림 2-1은 IPv6에서 사용하고 있는 FIB의 구조를 나타내고 있다.

* 이 연구는 '99대학기초연구사업의 지원을 받아 이루어졌습니다.

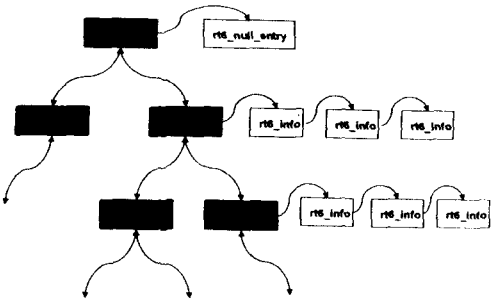

그림 2 - 1 IPv6에서의 FIB 구조

그림 2 -1에서 fib6_node 들의 트리는 밑으로 내려갈수록 프리픽스(prefix)의 길이가 증가하는 식으로 되어있다. 이 트리를 탐색하는 순서는, 먼저 루트노드로부터 프리픽스 값을 하나씩 증가시키며 최대 프리픽스 일치(longest prefix match)를 시도한다. 현 레벨에서의 어느 한 노드와 프리픽스가 일치되면 프리픽스의 다음 비트를 보고 현 노드의 왼쪽과 오른쪽 중 어디로 내려갈 것인가 판단한다. 같은 방법으로 최대 프리픽스 일치를 찾으면, 찾은 fib6_node 의 leaf 필드에 달려있는 rt6_info의 리스트를 차례로 검색한다. 이 리스트는 같은 프리픽스 값을 갖지만 서로 다른 네트워크 장치나 다른 게이트웨이 등을 갖는 엔트리들의 모임이다. 만약 여기서 다음 홉(next hop)이 결정된 엔트리를 찾지 못하면 NDP(Neighbor Discovery Protocol)[2][3] 를 이용하여 다음 홉을 찾은 후 새 엔트리를 삽입하게 된다. 이후 그 패킷의 처리는 찾은 엔트리가 가지고 있는 함수 포인터 input() 에 어떠한 함수의 주소를 가지느냐에 따라 상위 계층으로 올려보낼 것인지 아니면 그대로 포워딩 루틴으로 보낼 것인지 결정된다.

2.2 리눅스에서의 포워딩 흐름

리눅스상에서 들어온 패킷을 전달하는 과정은 그림2-2와 같다.

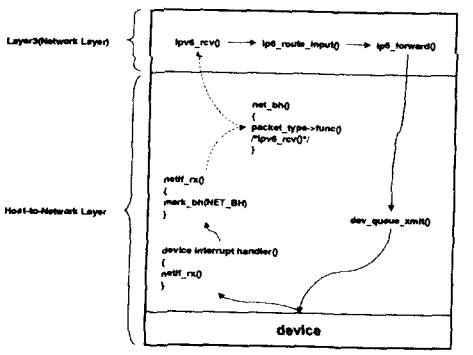

그림 2 - 2 리눅스에서의 패킷전달과정

처음 디바이스에서 패킷을 수신하면, 인터럽트가 발생하게 되고, 인터럽트 핸들링 루틴에서는 netif_rx() 를 호출하게 된다. netif_rx()에서는 소켓버퍼의 리스트에 새로 들어온 패킷을 달아주고 기본적인 처리만을 한 후 네트워크 하반부처리(Network Bottom Half)에 나머지 처리를 등록시키고 종료하게 된다. 이 후 스케줄러가 호출되고 네트워크 하반부처리 루틴인 net_bh() 가 호출되는데 여기서는 새로 들어온 패킷이 어느 프로토콜에 속하는지 판단한 후 적절히 등록된 함수(IPv6인 경우는 ipv6_rcv()) 를 호출하게 된다. ipv6_rcv() 에서는 홉 당 선택 헤더(hop-by-hop option header) 를 처리하고, 새로 들어온 패킷의 경로설정을 하기위해 ip6_route_input() 을 호출한다. ip6_route_input() 에서 다음 홉이 결정된 패킷은 상위 계층으로 올려지거나 ip6_forward() 로 넘겨져서 다음 라우터(혹은 호스트)로 전송된다.

3. 플로우 레이블을 이용한 포워딩 구현

레이블을 이용한 포워딩을 하기 위해서는 사전에 레이블 스위칭을 하는 라우터끼리 각각의 경로에 어떤 레이블을 사용할 것인지 결정하는 과정(LDP)이 필요하다. 하지만 이번 구현에서는 사용자가 수동으로 레이블 정보를 넣어주도록 하였고, LDP 기능은 차후에 추가로 구현할 예정이다.

3.1. 커널 자료구조

레이블 스위칭을 하기 위해 수정한 커널은 두개의 버전으로 나뉜다. 첫째는 레이블 스위칭을 하는 도매인의 입구에 존재하는 진입/진출(Ingress/Egress) 라우터의 커널이고, 두 번째는 도매인 내부에서 레이블을 이용한 스위칭을 행하는 라우터용 커널이다.

진입/진출 라우터용 커널은 레이블 스위칭 도매인에 들어오는 패킷에 레이블을 할당하는 일과, 나가는 패킷의 레이블을 0로 만들어주는 역할을 한다. 따라서 진입 패킷일 때는 기존 IPv6 의 FIB구조를 그대로 이용하여 rt6_info 에 저장되어 있는 레이블 정보를 가져오고, 진출 패킷일 때는 레이블의 해쉬테이블의 엔트리로 하여금 레이블에 해당하는 rt6_info 의 포인터를 가지게 하여 레이블을 보고 라우팅 정보를 바로 가져올 수 있게 하였다. 이 구조를 그림 3 -1에 나타내었다.

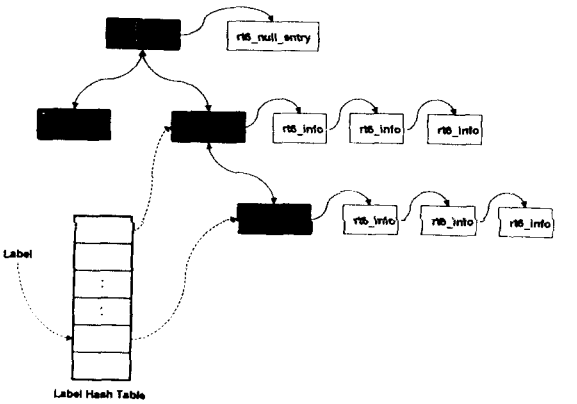

그림 3 -1 진입/진출(In/Egress) 라우터의 FIB 구조

도메인 내부에서 동작하는 커널은 들어오는 패킷의 레이블만 가지고 스위칭을 행한다. 그림 3 -2에 본 구조를 나타내었다.

도메인 내부의 라우터는 들어온 패킷의 레이블을 보고 해쉬테이블의 엔트리를 찾은 후 매치되는 rt6_info 를 찾아 rt6_info 에 저장되어 있는 외부로 나가는 레이블(Outgoing Label)을 다시 플로우 레이블 필드에 저장하여 전송한다

3.2. route 명령의 수정

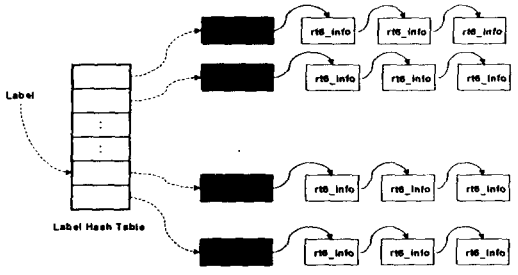

그림 3-2 도메인 내부(Intra Domain) 라우터의 FIB 구조

현재의 구현에서는 LDP 가 구현되어 있지않기 때문에 자동으로 레이블을 분배할 수가 없다. 따라서 수동으로 레이블을 커널 내에 삽입해 주어야 한다. 따라서 기존 route 명령을 수정하여 커널내의 자료구조를 수정할 수 있도록 하였다. 수정된 route 명령의 형식은 다음과 같다.

route -A inet6 add **destination-addr gw gateway-addr il input-label ol outgoing-label dev device-name**

il 옵션 다음에는 패킷이 라우터에 들어올 때 갖고 들어오는 레이블을 나타내며, ol은 il과 매핑되어 있는 레이블을 나타낸다. 즉, ol 은 포워딩 시 il 레이블을 달고 들어온 패킷이 바꿔가질 레이블을 나타낸다.

4. 실험

본 논문에서 사용한 실험 망은, 본래 IPv6 의 플로우 레이블을 이용한 QoS 알고리즘을 개발하기 위해 구축한 테스트베드에서 이루어졌다. 본 망은 서강대와 64k 의 전용선을 통해 연결되어 있다. 전용선은 두 망간의 고속 직렬 통신 (V.35/RS449)을 위해 각각 FDSU 를 사용해 연결되어 있다. 또한 고속 직렬 통신을 위한 동기식 직렬 인터페이스를 제공하기 위해 리눅스 박스에 동기화 보드를 장착하여 FDSU 와 연결하고 있다. 현재 구축된 망의 토폴로지는 그림 4-1과 같다.

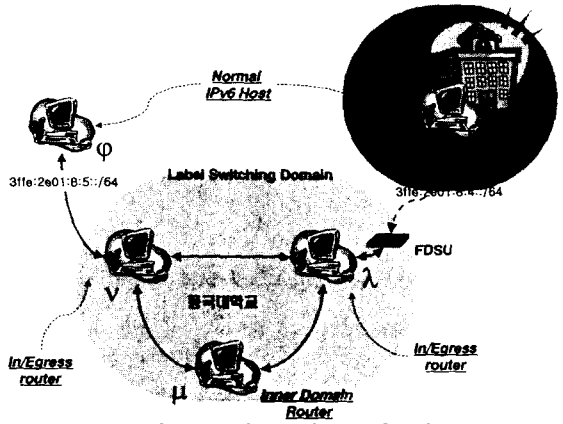

그림 4-1 테스트베드 토폴로지

커널 수정 후 레이블 스위칭을 테스트한 환경은 그림 4-1과 같은 환경에서 수행했다. 서강대와 동국대에 각각 아무런 수정도 하지 않은 IPv6 리눅스 박스(①, ②)를 세팅하고, In/Egress 용 커널은 ③, ⑤에 설치하였다. ④에는 도메인 내부 라우터용 커널을 설치하였다. 실험은 위와 같은 환경에서① - ⑤ - ④ - ③ - ② 와 같은 경로를 이용하도록 레이블을 설정한 뒤 몇 가지 IPv6 용 응용(ping, ftp, telnet) 을 ①과 ② 사이에서 수행시켜서 올바른 수행을 테스트하였다.

② 의 레이블 테이블

destination	input label	output label	out device
3ffe:2e01:8:5::/64	0	3	eth1
3ffe:2e01:8:4::/64	9	0	ets0 (전용선디바이스)

④ 의 레이블 테이블

destination	input label	output label	out device
3ffe:2e01:8:5::/64	3	7	eth1
3ffe:2e01:8:4::/64	5	9	eth0

⑤ 의 레이블 테이블

destination	input label	output label	out device
3ffe:2e01:8:5::/64	7	0	eth1
3ffe:2e01:8:4::/64	0	5	eth2

테스트 시에 사용한 레이블 테이블의 내용은 위와 같고 IP v6 용 응용프로그램들이 이상 없이 수행됨을 확인할 수 있었다.

5. 결론

레이블을 이용한 스위칭 방법은 기존의 IP 주소를 이용한 방법이 longest prefix match 을 이용하는데 반해 short label exact match를 사용하여 고속의 포워딩 방법을 제공한다. 본 논문에서는 이러한 레이블을 이용한 스위칭 기법을 이용하되, 별도의 헤더가 필요한 MPLS 와는 다르게 IPv6 에서 제공하는 플로우 레이블을 이용하여 레이블 스위칭을 구현해 보았다. 이렇게 함으로써 스위칭을 하기 위해서 3 계층까지 올라가야 한다는 단점이 있지만, 새로운 헤더를 처리하기위한 부담이 없고, 일반적인 플로우 레이블의 기능과 접목시켜 수월하게 특정 플로우에 대한 서비스의 레벨을 결정할 수 있다는 장점도 있다.

현재 구현의 문제점은 LDP 기능이 구현되어 있지않아 수동으로 레이블을 결정해 주어야 한다는 점이다. 이 점의 보완은 차후 구현목표로써, 레이블 스위칭 도메인 내부의 라우터들끼리 동적으로 레이블을 결정할 수 있도록 하는 기능을 추가할 예정이다.

6. 참고문헌

[1] S. Deering and R. Hinden , "RFC-2460 Internet Protocol, Version 6 (IPv6) Specification", 1998

[2] T. Narten, E. Nordmark and W. Simpson, "RFC-2461 Neighbor Discovery for IP Version 6 (IPv6)", 1998

[3] A. Conta and S. Deering, "RFC-2463 Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", 1998

[4] V. Fuller, T.Li, J. Yu and K. Varadhan, "Classless Inter-Domain Routing (CIDR):an Address Assignment and Aggregation Strategy", 1993

[5] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)", 1994

[6] Eric C. Rosen and Arun Viswanathan , "Multiprotocol Label Switching Architecture : draft-ietf-mpls-arch-06.txt", 1999

[7] Eric C. Rosen, Yakov Rekhter, Daniel Tappan, Dino Farinacci and Guy Fedorkow, "MPLS Label Stack Encoding : draft-ietf-mpls-label-encaps-07.txt"

[8] Chiristian Huitema, "Routing in the Internet", PH PTR, 2000

[9] David A Rusling , "The Linux Kernel", GNU book, 1999

[10] Alessandro Rubini , "Linux Device Drivers", O'Reilly, 1998

[11] M Beck, "Linux Kernel Internals", Addison Wesley, 1998

[12] http://www.bieringer.de/linux/IPv6/status/IPv6+ Linux-status-kernel.html