IPv6 네트워크을 위한 라우터 자동 설정 프로토콜†

(A Router Auto-Configuration Protocol(RACP) for IPv6 Networks)

이 완 직*, 허 석 렬**
(Wan-Jik Lee, Seok-Yeol Heo)

요 약 IPv6 프로토콜의 주요 장점 중의 하나는 주소 자동 설정 기능이다. IPv6 호스트들은 이 기능에 의해 자동으로 네트워크 설정이 가능하지만 IPv6 라우터들은 여전히 수동으로 설정되어야 한다. 이러한 문제를 해결하기 위해 본 논문에서는 여러 개의 라우터와 서브 네트워크들로 구성되는 소규모 IPv6 네트워크 상의 모든 라우터들을 자동 설정할 수 있는 새로운 프로토콜을 제안한다. 제안된 라우터 자동설정 프로토콜은 ISP에서 할당된 IPv6 네트워크 프리픽스를 사용하여 사이트 내부의 모든 라우터들의 프리픽스와 라우팅 정보를 자동 생성하고 전달할 수 있는 기능을 가진다. 본 논문에서 제안된 라우터 자동 설정 기능은 소규모 사무실이나 특히 홈 네트워크와 같이 전문 네트워크 관리자가 없는 IPv6 사이트의 네트워크 자동 설정을 위해 많이 활용될 수 있다.

핵심주제어 : 라우터 자동설정, IPv6 네트워크

Abstract Address Auto-configuration capability is one of important advantages of IPv6 protocol. This function enables the IPv6 hosts to configure IPv6 networks automatically, while IPv6 routers still have to be configured manually. To solve this problem, we propose RACP(Router Auto-Configuration Protocol), a new address auto-configuration protocol which configures all routers of a small network consisting of several routers and sub-networks automatically. The RACP protocol can automatically create and deliver IPv6 prefixes and routing informations of all routers on the network by using the network's prefix assigned by ISP. The proposed RACP can be used to set up network automatically for a small IPv6 site such as a small office network, a home network without the assistance of network administrator.

Key Words : Router Auto-configuration, IPv6 Network

1. 서 론

IP 주소 고갈과 무분별한 주소 배정에 따른 라우팅 비용 증가 등을 해결하기 위해 1990년대부터 연구된 IPv6의 표준화는 거의 완료된 상태이다. IPv6 프로토콜은 풍부한 주소 공간 이외 주소 자동 설정 기능, 효율적인 라우팅 테이블 관리, 멀티캐스트(Multicast) 및 애니캐스트(Anycast) 지원, IPSec을 이용한 보안 강화 등의 많은 장점을 가지고 있다.

이러한 장점 중의 하나인 주소 자동 설정 기법

† 이 논문은 부산대학교 자유과제 학술연구비(2년)에 의하여 연구되었음.
* 부산대학교 바이오정보전자공학과
** 부산대학교 바이오정보전자공학과, 교신저자

을 활용하면, IPv6 호스트는 DHCP 등의 외부 서버의 도움 없이 자신의 링크-로컬 주소를 스스로 생성할 수 있으며, 현재 링크에서 동작 중인 IPv6 라우터를 탐지하고, 라우터로부터 글로벌(Global) 프리픽스 주소를 받아서 호스트의 글로벌 IPv6 주소 역시 자동으로 생성할 수 있다. 하지만 네트워크를 구성하고 있는 각 라우터들에 대한 설정은 여전히 네트워크 관리자가 수동으로 설정하여야 한다. 물론 네트워크 관리자가 있는 환경에서는 이와 같은 상황이 크게 문제가 되지 않지만, 홈 네트워크, SOHO 환경, 중·소 규모 기관 등에서는 네트워크 관리자를 따로 두기가 힘들기 때문에 기관 내의 라우터 설정 등이 상당히 힘든 작업이 될 수 있다. 특히 홈 네트워크는 이더넷, WLAN, IEEE1394, Bluetooth, 전력선 등의 여러 물리적인 네트워크이 혼용될 수 있으며, 많은 종류의 장비가 홈 네트워크로 연결될 수 있다. 이런 점을 고려하면 홈 네트워크은 풍부한 주소 공간과 주소 자동 설정 등의 장점을 가진 IPv6 파급에 결정적인 계기가 될 수 있으며 물리적인 네트워크를 연동하기 위해 사용하는 라우터들의 자동 설정 기능도 매우 필요한 기능으로 볼 수 있다.

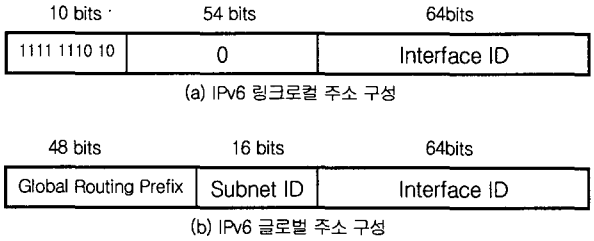
이 점에 착안하여 본 논문에서는 중·소규모 사이트의 모든 IPv6 라우터들을 자동 설정하는 프로토콜(Router Auto-Configuration Protocol: RACP)을 제안한다. 제안된 프로토콜을 활용하면 외부 망과 연결된 외부 라우터 정보를 이용하여 사이트 내의 모든 라우터들의 관계를 트리 형태의 논리적인 계층구조로 형성하고, 모든 라우터들에게 네트워크 프리픽스를 자동으로 전달할 수 있다. 그리고 각 라우터들의 라우팅 정보까지 전달하여 라우팅 프로토콜 등의 설정 없이 자동으로 IPv6 네트워크 연결을 가능하게 한다. 또한 본 논문에서 제안한 프로토콜의 기능을 확인하기 위해 리눅스를 탑재한 PC 라우터 상에서 RACP 기능을 구현하고, 구현된 프로토콜을 IPv6 테스트 베드 상에 동작시켜 각 라우터들 사이의 프리픽스 전달과 라우팅 정보 전달 등을 직접 확인하였다.

본 논문의 2장에서는 IPv6 자동 주소 설정 방법과 관련 연구에 대해 설명하고, 3장에서는 제안한 RACP 프로토콜에 대해 기술한다. 그리고 4장에서는 프로토콜의 구현 및 테스트에 대해 설명하며, 마지막으로 5장에서 결론과 향후 연구에 대해 기술한다.

2. IPv6주소 자동설정 기법 및 관련 연구

2.1 IPv6 호스트 주소 자동 설정

IPv6 호스트는 RFC 2461 Neighbor Discovery for IPv6[5]와 RFC 2462 Stateless Address Autoconfiguration[6]에 의해 다음의 그림 1과 같은 IPv6 링크로컬 주소와 IPv6 글로벌 주소를 생성할 수 있다.



<그림 1> IPv6 주소 구성

링크로컬 주소는 단일 링크에서만 사용이 가능하며, 그림 1의 (a)와 같이 미리 표준으로 정해진 1111111010(FE80::/10) 링크로컬 프리픽스를 가진다. 그리고 글로벌 유니캐스트 주소의 일반적인 구성은 아래와 같다(그림 1의 global routing prefix와 subnet ID의 길이가 각각 48, 16으로 많이 사용되고 있지만, 완전히 고정된 것은 아니다[9]).

- Global Routing Prefix: 일반적으로 TLA(Top Level Aggregator)라고 불리며, 네트워크 제공자(Provider)에 의해 설정된다.
- Subnet ID: 한 사이트 내에서 서브네트를 구별하기 위해 사이트 관리자에 의해 할당된다.
- Interface ID: 하나의 호스트(인터페이스)를 구별하기 위해 사용되며, 일반적으로 네트워크 인터페이스의 MAC 주소에 의해 자동으로 생성된다(사생활 보호 등을 위해 랜덤으로 생성되는 방법을 쓰기도 함).

IPv6의 링크로컬 주소는 자동으로 생성되지만, 글로벌 주소를 생성하기 위해서는 라우터가 전송

하는 RA(Router Advertisement)메시지의 프리픽스 정보(Prefix Information) 옵션에 포함된 64비트 길이의 네트워크 프리픽스(global routing prefix + subnet ID) 값과 자신의 interface ID를 결합해야 한다. 따라서 IPv6 호스트의 완전한 글로벌 주소 생성을 위해서는 전체 네트워크의 모든 라우터에게 네트워크 프리픽스 정보를 설정해 주어야 한다. 그러므로 한 사이트 내의 모든 호스트들에게 완전 자동으로 IPv6 주소를 설정해 주기 위해서는 사이트 내의 모든 라우터들에게 자신이 RA메시지로 전송할 네트워크 프리픽스 정보를 설정해 주어야 한다.

2.2 관련 연구

현재까지 라우터의 네트워크 프리픽스 자동 설정에 관해서 몇몇 방법들이 제안된 바가 있다. APD(Automatic Prefix Delegation) 프로토콜[1]과 RA-PD(Route Advertisement - Prefix Delegation) 옵션[2]은 라우터의 네트워크 프리픽스를 자동으로 원격에서 설정하는 방법을 제시하고 있다.

APD 프로토콜은 클라이언트 서버 모델에 기초하여 프리픽스를 필요로 하는 하위 라우터가 자신의 상위 라우터에게 프리픽스를 동적으로 할당받거나 해제하는 기능을 수행한다. 이를 위해 APD는 ICMPv6에 APD 프로토콜을 위한 2개의 메시지를 따로 정의하고 있다. 이와 달리 RA-PD는 라우터가 호스트에게 라우터의 존재와 프리픽스 정보를 전달하기 위해 사용하는 기존의 RA 메시지를 라우터 간으로 확장하는 형식을 사용한다. 즉 RA 메시지에 PD 옵션을 사용하여 이 RA-PD 메시지는 상위의 라우터가 하위 라우터에게 프리픽스 정보를 전달하는 용도로 사용된다. APD 프로토콜은 RA-PD에 비해 동적으로 프리픽스 할당과 해제가 이루어질 수 있으며 하위 라우터가 상위 라우터에게 자신이 선호하는 프리픽스로 협상할 수 있는 장점을 가지지만 새로운 프로토콜을 구현해야 하는 단점을 가진다.

하지만 이 두 방식 모두 상위와 하위 라우터(보통 최하위 단말 라우터)간에만 적용할 수 있다는 큰 단점을 가진다. 즉 이 두 방식 모두 ISP의 라우터가 사용자의 라우터에게 네트워크 프리픽스를 자동으로 할당하는 용도로만 사용할 수 있을 뿐이며 사용자가 여러 개의 라우터를 계층적으로 사용할 경우에는 모두 사용이 불가능하다.

이와는 달리 [3]에서 제안한 NAP(No Administration Protocol) 방식은 본 논문의 목적과 같이 홈네트워크와 같은 소규모 망에서 각 라우터들의 프리픽스를 자동으로 설정하는 것이다. 하지만 이 방식은 OSPF(Open Shortest Path First) 라우팅 프로토콜의 LSA(Link State Advertisement) 메시지 및 LSA 데이터베이스를 이용하는 방식이므로 각 라우터들이 OSPF와 같은 별도의 라우팅 프로토콜을 동작시켜야 한다는 단점을 가진다.

3. RACP 프로토콜 설계

RACP 프로토콜의 기능은 크게 두 가지로 나뉠 수 있다. 그 하나는 사이트 내의 모든 라우터들에게 계층적으로 프리픽스를 자동 전달하고 설정하는 기능이며, 다른 하나는 각 라우터들의 라우팅 정보를 자동 설정하여 RIP, OSPF 등의 라우팅 프로토콜의 도움 없이 자체적으로 라우팅을 가능하도록 하는 것이다. 이러한 기능을 수행하기 위해 RACP는 아래와 같은 형태로 설계되었다.

- 라우터의 데몬(daemon) 프로그램으로 동작한다.
- 라우터의 RA 메시지를 전송하는 프로그램과 연동하여 동작한다.
- 일반 호스트와 동일하게 라우터의 인터페이스도 RA의 프리픽스 정보를 수신하면 이를 이용하여 글로벌 주소를 생성한다. 즉 IPv6 호스트의 주소 자동 생성 기능을 라우터도 수행한다.
- 앞으로 설명될 RACP의 송·수신 관련 타이머들은 지정된 시간의 80%~100% 사이의 랜덤 값을 사용한다. 예를 들어 3초 후 전송일 경우에는 2.4초~3초 사이의 랜덤한 시간에 전송한다. 이는 동일한 시간에 라우터들의 RACP 메시지가 동시에 전송되어 네트워크에 오버헤드가 발생하는 현상을 줄이기 위해서이다.

3.1 RACP 동작 개요

RACP 프로토콜은 네트워크 계층상에 상위 라우터

와 하위 라우터 간에 프리픽스를 요청, 설정, 해제하는 기능을 가진 APD(Automatic Prefix Delegation) 프로토콜[1]을 수정·확장하여 설계하였다.

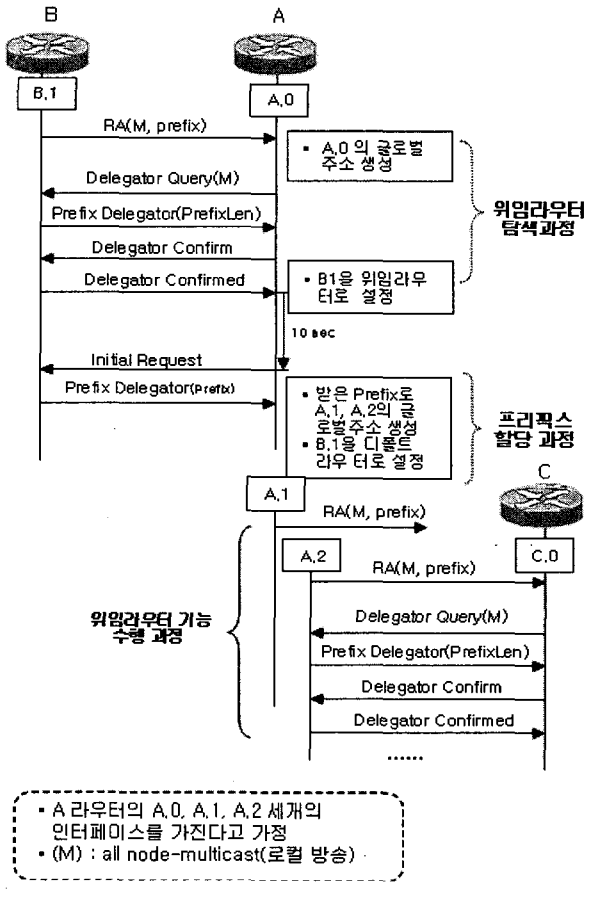
본 논문에서는 APD의 표기법을 따라 프리픽스를 요청하는 하위 라우터를 요청라우터(Requester Router:R/R)로 표현하고, 하위 라우터에게 실제 사용 가능한 프리픽스를 할당하는 라우터를 위임 라우터(Delegator Router: D/R)라고 표현한다. 프로토콜 동작 중에 하나의 라우터는 요청 라우터가 되어 자신의 위임 라우터에게 프리픽스를 할당 받을 수도 있고, 반대로 자신이 위임 라우터가 되어 다른 요청 라우터들에게 프리픽스를 할당해 줄 수도 있다.

RACP 프로토콜은 총 10개의 메시지를 사용하여 위임 라우터 탐색 과정, 프리픽스 할당 과정을 거쳐서 프리픽스를 할당받고 이 프리픽스 정보를 이용하여 자신이 관리하는 링크로 RA(프리픽스 정보 포함)를 전송하게 된다. 이런 과정 이후에는 다른 요청 라우터의 위임 라우터 역할을 수행하게 된다. RACP 프로토콜의 각 메시지의 명칭과 의미는 아래의 표 1과 같으며 상세한 메시지 포맷은 3.2절에서 설명한다. 표 2의 코드 열은 실제 메시지 포맷에 메시지 종류를 구별하기 위해 사용되는 코드 값을 나타내며, Delegator Query 메시지는 상위 위임 라우터를 탐색하기 위해 모든 라우터에 대해 링크로컬 멀티캐스트 되어지며, 그 외 다른 메시지는 모두 상대 라우터(위임 또는 요청)로 유니캐스트 전송된다.

<표 1> RACP 메시지 종류 및 역할

메시지 명	의 미	코드
Delegator Query	상위의 위임 라우터를 탐색하기 위해 사용하며, 링크로컬의 모든 라우터로 멀티캐스트 전송됨	0
Prefix Delegator	Delegator Query의 응답으로 자신이 위임 라우터임을 알리며, 요청 라우터에게 현재 링크의 프리픽스 길이를 지정함	1
Delegator Confirm	상위의 위임 라우터를 확정하기 위해 전송됨	2
Delegator Confirmed	Delegator Confirm의 응답으로 자신이 위임 라우터임을 확정하기 위해 요청 라우터로 전송됨	3
Initial Request	상위의 위임 라우터에게 자신이 사용할 프리픽스를 최초로 할당받기 위해 사용함.	4
Prefix Delegated	Initial Request 또는 Renewal Request의 응답으로 요청 라우터에게 할당된 프리픽스를 전달하기 위해 사용함	5
Renewal Request	이미 할당받은 프리픽스를 갱신하기 위해 사용함	6
Prefix Return	할당받아 사용 중인 프리픽스를 위임 라우터에게 반납하기 위해 사용함.	7
Prefix Returned	Prefix Return의 응답으로 프리픽스의 반납을 확인하기 위해 사용함	8
Error Occurred	프로토콜 동작 중에 에러 발생을 통보하기 위해 사용됨 실제 발생한 에러의 종류는 별도의 Error Code 필드로 표시함	9

RACP 프로토콜의 전체 동작 예를 그림 2에 나타내었다. 그림 2에서는 보는 바와 같이 A 라우터의 동작은 크게 위임 라우터 탐색 과정, 프리픽스 할당 과정, 위임 라우터 수행 과정으로 나눌 수 있다. 위임 라우터 탐색 과정은 자신의 인터페이스로 RA 메시지를 보낸 상위 라우터를 탐색하여 자신의 위임 라우터로 설정하는 동작을 수행하며, 프리픽스 할당 과정에서는 위임 라우터로부터 자신이 사용할 대표 프리픽스를 할당받고 그 프리픽스를 라우터 기능 수행 과정은 A 라우터 역시 B 라우 이용하여 자신의 다른 인터페이스(그림 2에서 A.1



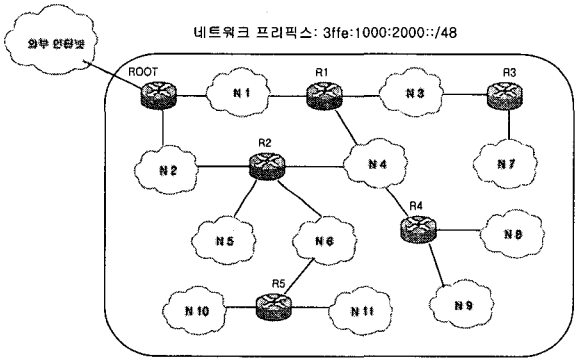
<그림 2> 라우터의 RACP 동작 예시

과 A.2)의 글로벌 주소를 생성한다. 마지막 위임 터처럼 자신의 하위 라우터에게 위임 라우터의 역할을 수행하는 과정이다.

각 과정별 라우터가 수행하는 RACP 프로토콜의 세부 동작 설명은 다음 3.2 절에서 상세히 소개한다.

3.2 RACP 동작 절차

RACP 프로토콜의 동작 절차를 설명하기 위해서 그림 3에 어떤 기관의 가상적인 네트워크 구성도를 나타내었다. 그림 3의 기관은 48비트 길이의 3ffe:1000:2000::/48 네트워크 프리픽스가 ISP에 의해 할당되었다고 가정한다. 그림 2의 네트워크 구성도에서 R2 라우터를 대상으로 RACP 프로토콜 기능을 차례로 설명하면 다음과 같다.



<그림 3> 물리적인 네트워크 구성도

가) 라우터 부팅 후 RACP 프로토콜의 초기 동작

① 라우터의 각 인터페이스들의 글로벌 주소 생성 여부를 탐지한다.

② 만약 글로벌 주소가 생성되었다면 현재 라우터의 프리픽스 설정 방식이 관리자에 의한 수동 설정이라고 판단하고 RACP 프로토콜 동작을 종료한다.

③ 글로벌 주소가 생성된 인터페이스가 없다면 모든 인터페이스에 대해 RA(프리픽스 정보 옵션 포함) 수신을 기다린다.

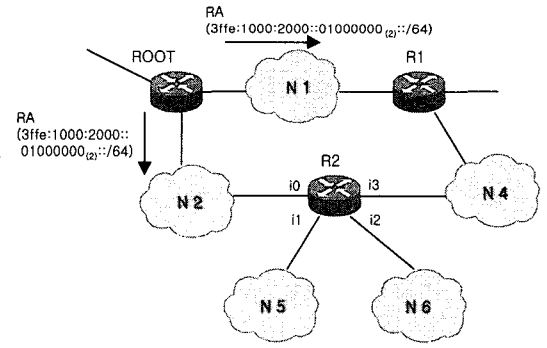
나) 위임 라우터(D/R) 탐색 과정

일단 루트 라우터의 RACP 프로토콜 동작에 의해 그림 4와 같이 프리픽스 정보가 포함된 RA 메시지가 N1, N2 망에 모든 노드(all-node) 멀티캐스트 된다. 그림 4의 프리픽스 주소 값에서 49~56까지의 비트 정보는 설명의 편이를 위하여 16진수가 아닌 2진수로 표시한다. 그리고 실제 루트 라우터에서 N2 망에 할당된 프리픽스 값은 3ffe:1000:2000::$01000000_{(2)}$::/50 이지만, 현재 RA메시지의 프리픽스 정보는 64 비트 길이로 지정되어야 N2 망의 호스트들이 자동으로 글로벌 주소를 생성할 수 있다[5, 6].

그림 4의 상태에서 R2 RACP의 위임 라우터 탐색 과정은 다음과 같다.

① i0 인터페이스로 RA(프리픽스 정보 포함)를 수신하였으므로 i0 인터페이스의 글로벌 주소를 생성한다.

② i0 인터페이스로 Delegator Query 메시지를 멀티캐스트 한다. 만약 i0 인터페이스 이외 i3 인터페이스를 통해서도 RA를 수신하였다면 i3 인터페이스의 글로벌 주소도 생성하고, Delegator Query 메시지를 전송한다. 여러 개의 인터페이스로 Delegator Query 메시지를 전송하였을 경우에는 여러 개의 Prefix Delegator 메시지를 모두 수



<그림 4> R2 라우터의 RACP 프로토콜 초기 동작 상황

신할 수 있도록 3초간 대기한다.

③ 만약 3초 동안 Prefix Delegator가 하나도 수신되지 않는다면, 3초 간격으로 최대 2번까지 Delegator Query 메시지를 재전송한다. 재전송했음에도 Prefix Delegator를 수신하지 못한다면 이는 수동 설정 방식으로 판단하고 RACP 동작 수행을 중지한다. Delegator Query 메시지 뿐만 아니라 모든 응답을 필요로 하는 메시지들은 모두 이와 같은 재전송 과정을 수행한다.

④ i0 인터페이스로 루트 라우터의 Prefix Delegator

(PrefixLen=50) 메시지를 수신한다. 이때 PrefixLen 값은 현재 위임 라우터가 인터페이스로 할당된 프리픽스 길이를 의미한다. 루트 라우터의 내부에서 N2 망은 3ffe: 1000:2000::$01000000_{(2)}$::/50으로 정의되었으므로 PrefixLen 값은 50이 된다.

⑤ 만약 여러 인터페이스로부터 Prefix Delegator 메시지를 수신하였다면, 이 PrefixLen 값이 가장 작은 위임 라우터를 선택한다. 만약 PrefixLen 값이 동일할 경우에는 Prefix Delegator 메시지의 각 위임 라우터의 주소를 비교하여 주소 값이 작은 것을 실제 위임 라우터로 선택한다.

⑥ 위임 라우터(루트 라우터)에게 Delegator Confirm 메시지를 전송한다.

⑦ 루트 라우터로부터 Delegator Confirmed 메시지를 수신한다.

다) 프리픽스 할당 과정

① 위임 라우터가 하위의 다른 모든 요청 라우터를 파악할 수 있도록 10초간 대기한다.

② 루트 라우터를 자신의 위임 라우터로 설정하였으므로 루트 라우터에게 Initial Request 정보를 전송한다.

③ 위임 라우터(루트 라우터)로부터 Prefix Delegated 메시지를 수신한다. 이 메시지에는 RACP의 프리픽스 정보 옵션(prefix= 3ffe:1000:2000:$01010000_{(2)}$::/52 및 프리픽스 생존시간)이 포함된다. 프리픽스 생존시간은 IPv6 주소 생존시간과 동일하게 현재 설정한 프리픽스의 생존시간을 명시한다. (현재 RACP 설계에서 프리픽스 생존시간은 루트 라우터의 설정값에 의해 지정될 수도 있고, 별도로 지정된 값이 없으면 3600초를 디폴트 값으로 지정한다.) 이 52 비트 길이의 프리픽스 값은 R2의 대표 프리픽스로서 이 값을 이용하여 R2의 각 인터페이스 별 프리픽스들과 하위 요청 라우터들의 프리픽스 정보를 생성하게 된다.

④ 각 인터페이스 별 프리픽스를 생성한다. 그림 3과 같이 R2는 이미 주소가 생성된 i0 인터페이스를 제외하고 3개의 인터페이스를 가지므로 이들 인터페이스의 프리픽스를 아래와 같이 설정한다.

- i1: prefix = 3ffe:1000:2000:$01010100_{(2)}$::/54
- i2: prefix = 3ffe:1000:2000:$01011000_{(2)}$::/54
- i3: prefix = 3ffe:1000:2000:$01011100_{(2)}$::/54

총 3개의 인터페이스 프리픽스가 필요하므로 53, 54 두 개의 비트를 더 할당하여 01, 10, 11 값을 각각 i1, i2, i3에게 할당하였다. 두 비트로 할당할 수 있는 00 값은 R2의 대표 프리픽스 값과 구별되지 않으므로 사용할 수 없다.

라) 디폴트 라우팅 정보 설정

루트로부터 수신한 Prefix Delegated 메시지에는 AR(Automatic Routing) 플래그 비트가 있다.(3.3절 참조) 이 값이 1로 설정되면 라우터들의 라우팅 정보가 자동으로 설정되어야 하므로 아래와 같은 디폴트 라우터를 위임 라우터인 루트 라우터 주소로 지정한다.

- 목적지 ::/0 에 대해 i0 인터페이스로 루트 라우터의 주소

마) RA 메시지 전송

① 다)항의 인터페이스 별 프리픽스 정보를 이용해 아래와 같은 RA 메시지와 프리픽스 정보 옵션을 생성한다.

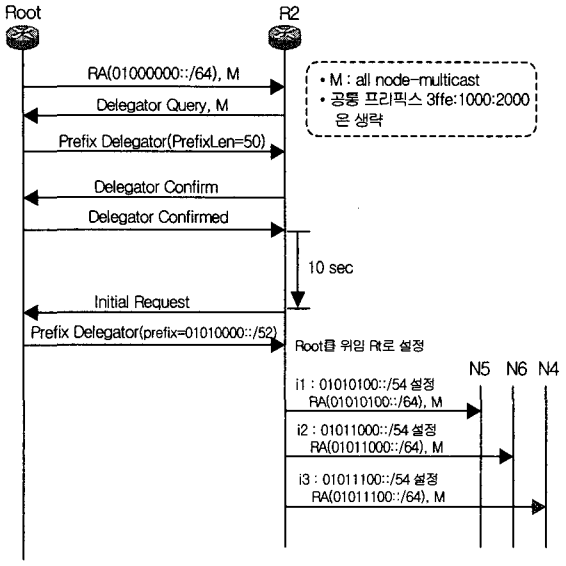
- i1: RA(prefix=3ffe:1000:2000:$01010100_{(2)}$::/64)
- i2: RA(prefix=3ffe:1000:2000:$01011000_{(2)}$::/64)
- i3: RA(prefix=3ffe:1000:2000:$01011100_{(2)}$::/64)

② RA 전송 전에 먼저 각 인터페이스에 다른 라우터의 RA 메시지를 수신하였는가를 살펴봐야 한다. 각 네트워크에 두 개 이상의 라우터가 있을 경우, 미묘한 시간차에 의해 하나의 망에 두 가지의 RA 메시지가 전송될 수 있다. 예를 들어 그림 3에서 R1이 먼저 N4 망으로 RA를 전송할 수가 있다. 이런 경우 R2는 i3 인터페이스로 RA 전송을 중지하고, i3 인터페이스의 글로벌 주소는 R1의 RA 메시지에 의해 생성되어야 한다.

③ RA를 전송하는 각 인터페이스 i1, i2, i3의 글로벌 주소를 생성한다.

④ RA를 받지 않은 인터페이스로 RA(프리픽스 정보 포함)를 전송한다. 빠른 시간 내에 하위 요청 라우터 파악을 위해 처음 3초 간격으로 두 번 전송한다. 이 후에는 30초 간격으로 주기적으로 전송한다.

그림 5는 현재까지 R2 라우터에서 수행된 RACP 동작을 나타낸 것이다.



<그림 5> R2의 RACP 프로토콜 동작 과정

그림 5와 같이 R2는 요청 라우터로서의 동작을 수행하고, 그림 3의 N5, N6, N4 망의 관장하는 라우터가 된다. 또한 R2는 하위 R5, R4에 대해 위임 라우터로서의 동작도 수행해야 한다.

바) 위임 라우터 기능 수행

① RA를 전송한 각 인터페이스에 대해 Delegator Query 메시지를 기다린다.

② Delegator Query 메시지에 대해 PrefixLen =54로 설정하여 Prefix Delegator 메시지를 송신한다.

③ Delegator Confirm 메시지를 수신한 후, Delegator Confirmed 메시지로 응답한다.

④ ③번까지의 동작으로 아래와 같이 하위 요청 라우터를 파악한다.

- i1: Delegator Query 메시지를 수신하지 못함.
- i2: R5가 요청 라우터가 됨.
- i3: R4가 요청 라우터가 됨.

⑤ 파악된 요청 라우터들의 대표 주소를 계산한다. 현재 RACP에서는 새로운 라우터들의 추가를 대비하여 최소 하나 이상의 여분을 두도록 설계하였다. 따라서 i2 인터페이스에는 R5 라우터 하나만 파악되었지만 i2인터페이스의 프리픽스(54비트 길이)에 요청 라우터를 위해 55, 56 두 개의 비트를 할당하여 아래와 같이 대표주소를 계산한다. R4 라우터에도 동일하게 적용한다.

- R5: 3ffe:1000:2000:01011001$_{(2)}$::/56
- R6: 3ffe:1000:2000:01011101$_{(2)}$::/56

⑥ R5와 R4의 Initial Request를 수신한다.

⑦ 아래와 같은 형태로 Prefix Delegated 메시지를 송신한다.

- R5: Prefix Delegator(프리픽스 정보 옵션: prefix= 3ffe:1000:2000:01011001$_{(2)}$::/56, 프리픽스 생존 시간)
- R4: Prefix Delegator(프리픽스 정보 옵션: prefix= 3ffe:1000:2000:01011101$_{(2)}$::/56, 프리픽스 생존 시간)

사) 라우팅 정보 설정

R5, R4에게 Prefix Delegator 메시지로 프리픽스를 할당해주었으므로 이에 대한 라우팅 정보를 아래와 같이 설정한다.

- R5: 3ffe:1000:2000:01011001$_{(2)}$::/56에 대해 인터페이스 i2로 R5의 주소
- R4: 3ffe:1000:2000:01011101$_{(2)}$::/56에 대해 인터페이스 i3로 R4의 주소

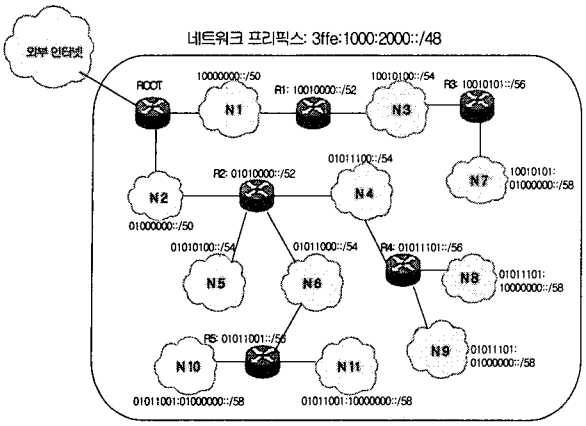
지금까지 R2의 요청 라우터로서의 동작 및 위임 라우터로서의 동작을 살펴보았다. R2와 같이 전체 네트워크에서의 라우터들이 RACP 동작을 수행하면 그림 6과 같은 형태로 전체 네트워크의 프리픽스들이 설정될 것이다. 그림 6에서 표현한 각 프리픽스들은 공통 프리픽스 3ffe:1000:2000::48은 생략하고 49비트 이후의 값들만 이진수로 표현한 것이다. 또 각 라우터들에게 표시된 프리픽스 값은 라우터들에게 할당된 대표 프리픽스를 나타낸 것이다.

이후 각 라우터들은 이미 설정된 프리픽스들의 갱신 과정과 라우터 종료 시에 프리픽스 반환 동작을 수행해야 한다.

아) 프리픽스 갱신(Renewal) 과정

현재 할당받았거나 할당해준 프리픽스들에게는 생존시간이 있다. 따라서 생존 시간이 경과되기 전에 프리픽스를 계속 사용하기 위해서는 아래와 같은 갱신 과정이 필요하다.

① 루트 라우터로부터 할당받은 3ffe:1000; 2000:01010000$_{(2)}$::/52에 대해 프리픽스 생존시간의 2/3가 경과 되면 루트 라우터로 Renewal Request 메시지를 송신한다.



<그림 6> RACP 프로토콜에 의한 전체 네트워크 프리픽스 할당 형태

② 송신한 Renewal Request 메시지의 응답으로 Prefix Delegated 메시지를 수신하면 이 메시지에 지정된 프리픽스 생존시간으로 현재 할당된 인터페이스들의 생존시간을 다시 갱신한다.

③ 만약 재전송을 했음에도 불구하고 Renewal Request 메시지를 수신하지 못한 경우에는 현재 남은 시간 동안만 프리픽스 활용이 가능하다.

자) 프리픽스 반납 과정

관리자에 의해 RACP 데몬이 중지되거나 라우터가 종료될 경우에는 현재 라우터에 할당된 프리픽스를 위임 라우터에게 반납해야 한다.

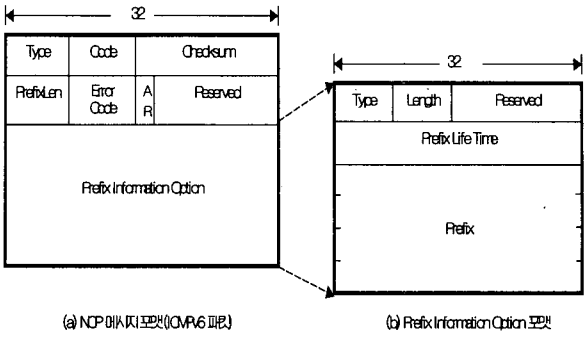
① 루트 라우터로부터 할당받은 3ffe:1000:2000:01010000$_{(2)}$::/52에 대해 위임 라우터인 루트 라우터에게 Prefix Return 메시지를 송신한다.

② Prefix Return 메시지의 응답으로 Prefix Returned 메시지를 수신하면 현재 각 인터페이스에 RA(프리픽스 정보, 생존시간=0)을 전송하여 프리픽스를 해제한다.

③ Prefix Return 응답이 없어 Prefix Return 메시지를 재전송을 했음에도 불구하고 응답이 없을 경우에도 ②과 동일하게 각 인터페이스에 할당된 프리픽스를 해제한다.

3.3 RACP 메시지 포맷

RACP 메시지 포맷은 그림 7과 같이 설계하였다. 그림 7의 (a)와 같이 RACP 메시지는 ICMPv6 메시지 중 하나의 Type을 RACP 용도로 정의해야 한다(현재 구현에는 IETF에서 아직 할당하지 않은 번호 141번을 사용하였다).



<그림 7> RACP 메시지 포맷

- Type: ICMPv6 패킷에서 RACP 메시지임을 표시하기 위해 사용
- Code: RACP의 메시지 종류를 표시함(표 1 참조)
- Checksum: ICMPv6 체크섬 값
- PrefixLen: Prefix Delegator 메시지에서 위임 라우터가 지정한 링크의 프리픽스 길이
- ErrorCode: RACP 동작 시 발생하는 에러의 종류를 표시함 No Prefix(0), Authentication Required(1), Unknown Field(2)를 명시
- AR: 한 비트의 플래그로서 Automatic Routing을 명시, AR 비트가 설정되면 RACP 자체에서 라우팅 정보를 설정하도록 동작함

그림 7의 (b)는 RACP 메시지의 옵션 중에 프리픽스 정보(Prefix Information) 옵션을 보여준다. 현재 설계에는 프리픽스 정보 옵션만 정의되어 있으며 type 값은 1로 표시한다. 프리픽스 정보 옵션은 위임 라우터와 옵션 라우터 사이에 교환되는 프리픽스 정보를 표시한다. Prefix Delegated, Renewal Request, Prefix Return, Prefix Returned 메시지만 이 옵션이 적용될 수 있다. 프리픽스 정보 옵션 필드의 설명은 다음과 같다.

- Type: 프리픽스 정보 옵션을 표시(=1)
- Length: 프리픽스 정보 옵션의 전체 길이를 바이트 단위로 표시
- Prefix Life Time: 프리픽스의 생존 시간을 초 단위로 명시
- Prefix: 실제 할당된 프리픽스 주소를 명시

4. RACP 프로토콜 구현 및 테스트

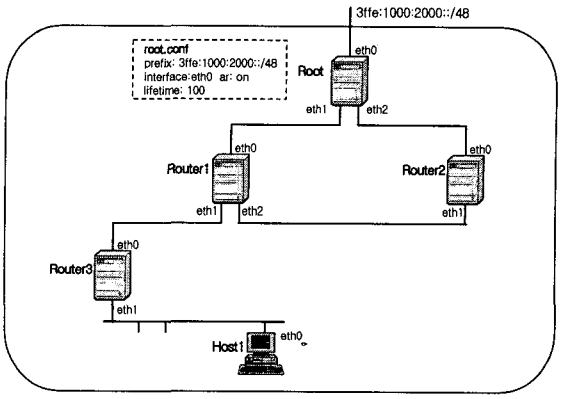
4.1 RACP 프로토콜 구현 환경 및 특징

본 논문에서 제안한 RACP 프로토콜을 구현하기 위한 시스템으로 Linux O/S가 탑재된 PC를 사용하였다. 구현에 사용된 PC들은 2개 이상의 NIC(Network Interface Card)를 설치하고, 시스템 설정을 IPv6 라우터로 지정하였다. 사용된 리눅스의 커널 버전은 2.4.18을 사용하였다. 프로토콜 구현을 위해 사용한 언어는 C 언어이며, ICMP 형태의 RACP, RA, RS 메시지 전송을 위해 IPv6 Raw Socket 인터페이스를 사용하였다.

그리고 RACP 구현은 리눅스 라우터에서 RA 메시지를 전송하기 위해 사용되는 Radvd 프로그램 기능을 포함하고 있다. Radvd 프로그램은 IPv6 네트워크에서 리눅스 라우터를 수동 설정시킬 때 사용되는 라우터 전용 프로그램이다. Radvd 프로그램은 프리픽스 값, 생존 시간, RA 전송인터페이스, 전송 주기 등의 정보가 기록된 설정파일(radvd.conf)을 읽어서 라우터의 지정된 인터페이스로 RA 메시지를 전송하는 프로그램이다. 또한 기존 radvd 프로그램과 호환성을 유지하도록 구현하였기 때문에 이 RACP 프로그램을 사용하여 기존의 수동 설정과 자동 설정(RACP)이 모두 가능하다. 구현된 RACP 프로그램은 우선 radvd 설정파일을 찾아서 이 파일이 존재하면 기존의 radvd 프로그램 기능만 수행하고, 이 파일이 존재하지 않으면 RACP 동작을 수행한다.

구현 및 테스트에 사용한 환경은 그림 8과 같다.

그림 8과 같이 총 4대의 PC 라우터와 하나의 호스트를 사용하여 구현 환경을 구축하였다. 다른 라우터와 달리 루트 라우터에는 root.conf라는 RACP 설정파일이 기록되어 있다. 비록 RACP가 라우터의 모든 과정을 자동으로 설정하지만, 루트 라우터의 동작은 일반 라우터의 동작과 약간의 차이가 있다. (상위 위임 라우터가 없고 외부와 연결되는 인터페이스 주소는 수동으로 설정된다.) 따라서 적어도 이 라우터가 루트 라우터이며 기관 전체에 할당된 프리픽스 정보를 명시해 줄 필요가 있으므로 이러한 역할을 현재의 구현에서는 root.conf 파일의 존재 유무로 결정한다. Root.conf 파일로 RACP의 설정할 수 있는 항목 및 내용은 표 2와 같다.



<그림 8> RACP 구현 및 테스트 환경

<표 2> root.conf 파일의 설정 항목 및 의미

항 목	의 미
prefix	사이트 전체에 할당된 프리픽스 및 프리픽스 길이를 지정한다.
interface	외부 ISP와 연결된 인터페이스를 지정
ar	RACP의 자동 라우팅의 on/off를 지정한다.(디폴트=on)
lifetime	각 라우터가 사용하는 프리픽스의 생존시간을 초 단위로 지정한다. (디폴트=3600초)

표 2와 같이 현재의 구현에서는 prefix와 interface 항목은 필수적으로 설정해 주어야 한다. 만약 RACP를 동작하는 기관과 ISP 간의 연결에 2장에서 언급한 APD[1] 또는 RA-PD[2] 와 같은 자동 설정 프로토콜을 활용한다면 이러한 프로토콜 기능도 루트 라우터에 내장하여 RACP와 연동할 필요가 있다.

4.2 RACP 프로토콜 동작 및 테스트

구현된 프로토콜의 모든 기능은 기본적으로 그림 8과 같은 환경에서 여러 형태의 테스트 시나리오를 통하여 테스트되고 디버깅되었다. 아래의 그림 9는 그림 8의 Router1의 RACP 동작 중에 출력되는 디버깅 정보를 캡쳐한 내용이다.

그림 8에서 Router1은 Root 설정파일(root.conf)이 없으므로 우선 Delegator Query 메시지를 이용하여 위임 라우터를 탐색하는 과정을 수행한다. 결

과적으로 Root 라우터를 위임 라우터로 설정하여 프리픽스(3ffe:1000:2000:5000::/52)를 할당받게 되고, 그 이후는 위임 라우터가 되어서 자신이 할당받은 프리픽스를 이용하여 요청 라우터인 Router3에게 프리픽스를 할당해 주는 과정을 볼 수 있다.

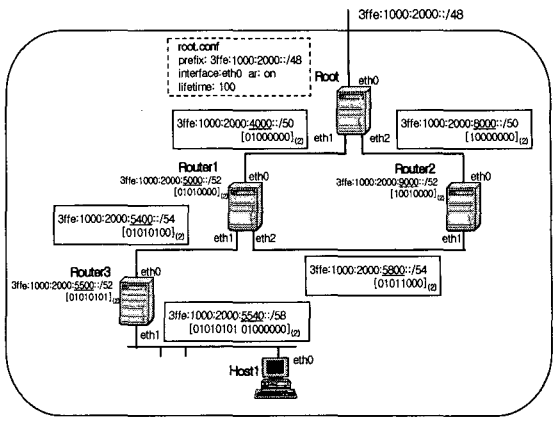
```
 1 [Mar. 17 15:52:18] racp:
 2 version 0.7.1 started
 3 [Mar. 17 15:50:16] racp: Cannot open rootconf file: root.conf -> racp_router mode
 4 [Mar. 17 15:52:18] racp: Recv RA prefix: 3ffe:1000:2000:4::: len:64
 5 [Mar. 17 15:52:18] racp: make_interface(): eth0 ifaceC2> : 3ffe:1000:2000:4000:250:fcff:feee:76c5
 6 [Mar. 17 15:52:18] racp: sending DELEGATOR QUERY on eth0 & 2 if_idx
 7 [Mar. 17 15:52:18] racp: process_racp(): Received DELEGATOR_QUERY !!!
 8 [Mar. 17 15:52:18] racp: process_racp(): But it is my DQ pkt -> Discard !!!
 9 [Mar. 17 15:52:18] racp: process_racp(): PREFIX_DELEGATOR recv !!!!!
10 [Mar. 17 15:52:18] racp: process_pd(): Received a PREFIX_DELEGATOR with a new Delegator through 2 indxx, prefixlen: 50
11 [Mar. 17 15:52:21] racp: Recv RA prefix: 3ffe:1000:2000:400C::: len:64
12 [Mar. 17 15:52:28] racp: racp_timer(): : State <- racp_DELEGATOR_confirm
13 [Mar. 17 15:52:28] racp: send_dc(): Send DELEGATOR_CONFIRM
14 [Mar. 17 15:52:28] racp: process_racp(): DELEGATOR_CONFIRMED recv !!!!!
15 [Mar. 17 15:52:28] racp: process_dced() : State <- racp_DELEGATOR_CONFIRMED
16 [Mar. 17 15:52:28] racp: send_rr(): sending INITIAL REQUEST to Delegator[fe80::210:5aff:fe84:5307]
17 [Mar. 17 15:52:28] racp: process_racp(): PREFIX_DELEGATED recv !!!!!
18 [Mar. 17 15:52:28] racp: process_pded(): PREFIX recvd 3ffe:1000:2000:5000::  len:52  LT:10G  AR
19 [Mar. 17 15:52:28] racp: process_pded(): eth1 get PREFIX: 3ffe:1000:2000:5400:: len:54
20 [Mar. 17 15:52:28] racp: make_interface(): eth1 ifaceC3> : 3ffe:1000:2000:5400:250:fcff:feee:76bc
21 [Mar. 17 15:52:28] racp: process_pded(): eth2 get PREFIX: 3ffe:1000:2000:5800:: len:54
22 [Mar. 17 15:52:28] racp: make_interface(): eth2 ifaceC4> : 3ffe:1000:2000:5800:20e:2eff:fe0a:134c
23 [Mar. 17 15:52:28] racp: kickoff(): eth2 Send RA
24 [Mar. 17 15:52:28] racp: kickoff(): eth1 Send RA
25 [Mar. 17 15:52:28] racp: process_pded(): State <- racp_DELEGATED, Prefix: 3ffe:1000:2000:5000::
26 [Mar. 17 15:52:28] racp: process_pded(): /sbin/ip -6 route add ::/0 via fe80::210:5aff:fe84:5307 dev eth0
27 [Mar. 17 15:52:28] racp: process_pded(): Set REFRESH Timer:      85.00
28 [Mar. 17 15:52:28] racp: process_racp(): Received DELEGATOR_QUERY !!!
29 [Mar. 17 15:52:28] racp: process_dq(): Received DELEGATOR QUERY peer addr: fe80::250:fcff:fe51:2490
30 [Mar. 17 15:52:28] racp: send_pd(): Send PREFIX DELEGATOR
31 [Mar. 17 15:52:31] racp: kickoff(): eth2 Send RA
32 [Mar. 17 15:52:31] racp: kickoff(): eth1 Send RA
33 [Mar. 17 15:52:37] racp: process_racp(): Received INITIAL REQUEST !!!
34 [Mar. 17 15:52:37] racp: send_pded(): Send PREFIX DELEGATED[Prefix: 3ffe:1000:2000:5500::]
35 [Mar. 17 15:52:28] racp: process_pded(): /sbin/ip -6 route add 3ffe:1000:2000:5500::/56 via fe80::250:fcff:fe51:2490 dev eth1
36 [Mar. 17 15:52:57] racp: kickoff(): eth2 Send RA
37 [Mar. 17 15:52:57] racp: kickoff(): eth1 Send RA
38 [Mar. 17 15:53:53] racp: send_rr(): sending RENEWAL REQUEST on 2 if_idx
39 [Mar. 17 15:53:53] racp: process_racp(): PREFIX_DELEGATED recv !!!!!
40 [Mar. 17 15:53:53] racp: process_pded(): PREFIX recvd 3ffe:1000:2000:5000::  len:52  LT:10G  AR
41 [Mar. 17 15:53:53] racp: process_pded(): State <- racp_DELEGATED, Prefix: 3ffe:1000:2000:5000::
42 [Mar. 17 15:53:53] racp: process_pded(): Set PREFIXSH Timer:      85.00
43 [Mar. 17 15:53:58] racp: kickoff(): eth2 Send RA
44 [Mar. 17 15:53:58] racp: kickoff(): eth1 Send RA
45 [Mar. 17 15:54:07] racp: process_racp(): Received RENEWAL REQUEST !!!
46 [Mar. 17 15:54:07] racp: send_pded(): Send PREFIX DELEGATED[Prefix: 3ffe:1000:2000:5500::]
47 [Mar. 17 15:54:07] racp: process_rr(): Send PREFIX DELEGATED !!!
48 [Mar. 17 15:55:16] racp: signt_handler called
49 [Mar. 17 15:55:16] racp: send_pr(): sending PREFIX RETURN on 2 if_idx
50 [Mar. 17 15:55:16] racp: signt_handler called && DELEGATOR exist0
51 [Mar. 17 15:55:16] racp: process_racp(): PREFIX_RETURNED recv !!!!!
52 [Mar. 17 15:55:16] racp: process_pded(): /sbin/ip -6 route del ::/0 via fe80::210:5aff:fe84:5307 dev eth0
53 [Mar. 17 15:55:16] racp: process_pded(): recv a RETURN pkt and Set CLOSING Timer
54 [Mar. 17 15:55:16] racp: process_pded(): /sbin/ip -6 route del 3ffe:1000:2000:5500::/56 via fe80::250:fcff:fe51:2490 dev eth1
55 [Mar. 17 15:55:16] racp: kickoff(): eth2 Send RA w/ LT=0
56 [Mar. 17 15:55:16] racp: kickoff(): eth1 Send RA w/ LT=0
```

<그림 9> Router1 RACP 동작 출력

또한 Renewal 타이머에 의해 프리픽스 갱신 과정도 볼 수 있으며 RACP 종료 시의 프리픽스 반납 과정도 볼 수 있다(38~42행). 프리픽스 반납 동작은 리눅스의 SIGINT, SIGKILL, SIGTERM 등의 시그널을 이용하여 구현하였으며 프리픽스 반납을 수행한 후 RACP를 종료하도록 하였다.

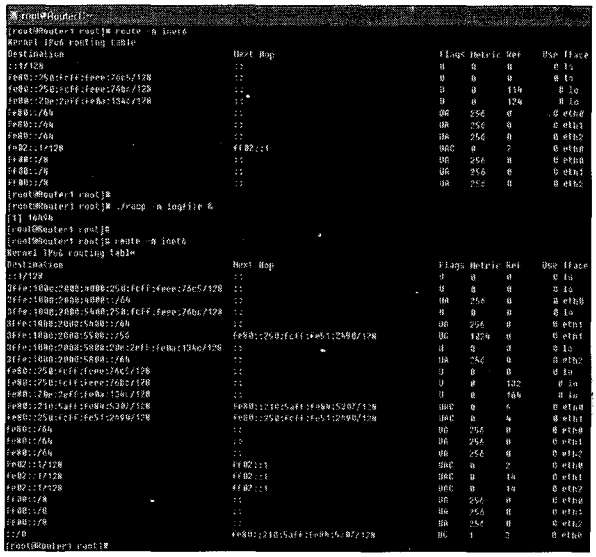
그림 8의 모든 라우터가 RACP 프로토콜을 수행하여 프리픽스를 할당 받은 상태를 그림 10에 나타내었다. 3장에서 언급한 RACP 설계 규격과 동일한 결과를 볼 수 있으며 다만 Router1의 eth2와 Router2의 eth1이 동일하게 연결된 링크의 경우에는 RACP 내부 타이머 정책에 따라 Router1에 의해 프리픽스가 할당될 수도 있고 Router2에 의해 프리픽스가 할당될 수도 있다.

Router1에서 RACP 동작 전과 후의 라우팅 테이블 내용 변화를 그림 11에 나타내었다. 그림 10에서 보는 바와 같이 RACP 동작 후, Router1의



<그림 10> RACP에 의한 프리픽스 할당 결과

eth0 링크로 3ffe:1000:2000:4000::, Router1의 eth1 링크로 3ffe:1000::2000:5400::, Router1의 eth2 링크로 3ffe:1000:2000:5800:: 프리픽스가 할당되었음을 알 수 있다. 그리고 RACP의 또 다른 기능인 자동 라우팅 설정에 의해 Router1의 디폴트 라우터(::/0)는 Root 라우터의 eth1의 링크로컬 주소(fe80::210:5aff:fe84:5307)로 설정되어 있으며 Router3 에게 할당된 프리픽스(3ffe:1000:2000:5500::/56)에 대해서는 Router3의 eth0의 링크로컬 주소(fe80::250:fcff:fe51:2490)로 설정되어 있음을 확인할 수 있다.



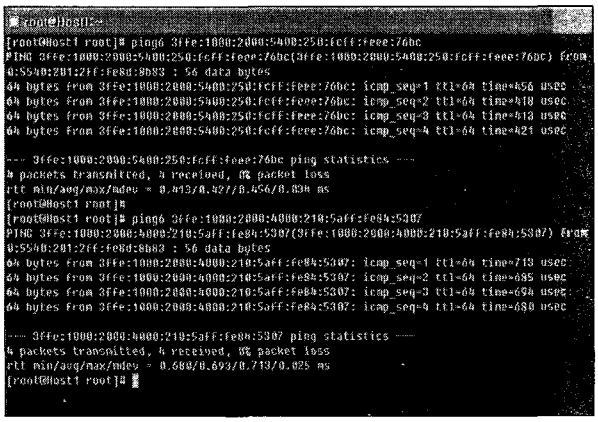
<그림 11> RACP 구동 전·후의 Router1 라우팅 테이블 내용 변화

마지막으로 그림 12에서는 RACP 동작에 의해

전체 네트워크가 설정된 후 Host1에서 Router1의 eth1(3ffe:1000:2000:5400:250:fcff:feee:76bc) 및 Root의 eth1 (3ffe:1000:2000:40000:210:5aff:fe84:5307)으로 ping6가 올바르게 동작됨을 보여주고 있다.

5. 결론 및 향후 연구

본 논문에서는 중·소규모의 사내 망이나 여러 물리적인 망이 연동되는 홈 네트워크 환경 등에서 전체 IPv6 네트워크를 자동 설정하는 기능을 가진 RACP 프로토콜을 제안하였다. RACP 프로토콜은 기존의 APD나 RA-PD 프로토콜에서 상위와 하위 간의 일대일 라우터 정보 전달에 벗어나 전체 네트워크의 라우터 간에 트리 형태의 계층적인 논리 구조를 형성하고, 루트 라우터에 지정된 프리픽스 정보를 하위 계층으로 동적으로 전파하여 전체 네트워크의 모든 라우터에게 자동으로 IPv6 프리픽스와 라우팅 정보를 설정할 수 있는 기능을 가진다. 본 논문에서 제안된 RACP 프로토콜을 활용하면 IPv6의 호스트 주소 자동 설정 기능을 전체 네트워크 자동 설정 기능으로 확장할 수 있다. 다만 RACP 프로토콜은 외부 인터넷이 두 개 이상의 ISP에 의해 연결되는 멀티 홈 사이트에 적용할 수 없으며 라우터와 링크의 추가 및 삭제 등, 내부 토폴로지 변화가 심한 네트워크에 적용하기 힘들다는 제약 조건을 가진다. 하지만 일반적인 홈 네트워크나 사내 망의 경우 내부 토폴로지 변화가 심



<그림 12> Host1에서 Router1 및 Root 라우터로 ping6 수행 결과

토콜의 활용 가능성은 매우 높다고 볼 수 있다. 또한 본 논문에서는 제안된 RACP 프로토콜을 시험하기 위해, 리눅스 운영체제의 PC를 활용하여 가상의 IPv6 망(라우터, 호스트)을 구성하였고, 구성된 시스템에서 프로토콜의 기능을 C언어를 이용하여 직접 구현하고 동작을 확인하였다.

현재 RACP 구현은 자동 네트워크 설정 메커니즘은 충실하게 수행하지만 라우터에서 필요한 보안 기능은 고려되지 않았다. 만약 IPv6 라우터에 악의적인 공격에 의해 잘못된 IPv6 프리픽스 설정 및 해제가 수행된다면 로컬 네트워크의 기능이 마비될 수도 있다. 그러므로 현재 RACP 프로토콜에 RFC 3971에 명시된 SEND(Secure Neighbor Discovery) 기능을 이용하여 라우터들의 인증 등의 보안 기능을 추가하는 것이 필요하다. 또한 현재 구현은 가상의 IPv6 망에서만 테스트된 상태이므로, 실제 현장에서 적용하기 위해서는 많은 라우터와 호스트가 구성된 복잡한 망에서 완벽히 테스트되고 검증할 필요가 있다.

참 고 문 헌

[1] B. Haberman and J. Martin, "Automatic Prefix Del-egation Protocol for Internet Protocol Version 6(IPv6)," draft- haberman-ipngwg-auto-prefix-01.txt, July 2001.

[2] Nathan Lutchansky, "IPv6 Router Advertisement Prefix Delegation Option," draft-lutchann-ipv6-delegate-option-00.txt, February 2002.

[3] Guillaume Chelius, Eric Fleury, "No Administration Protocol(NAP) for IPv6 router-auto-configuration," Proceedings of the 19th AINA, 2005.

[4] A. Conta, and S. Deering, "ICMP for Internet Protocol Version 6 (IPv6)," RFC 2463, December 1998.

[5] T. Narten, E. Nordmark and W. Simpson, "Neighbor Discovery for Internet Protocol Version 6 (IPv6)," RFC 2461, December 1998

[6] S. Thomson and T. Narten, "IPv6 Stateless

Address Autoconfiguration," RFC 2462 December 1998.
[7] J. Bound, M. Carney, C. Prekins and R. Droms(ed.), "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," draft-ietf-dhc-dchp6-21.txt, November 2001.
[8] M. Crawford, "Router Renumbering for IPv6," RFC 2894 August 2002.
[9] R, Hinden, S. Deering, E. Nordmark, "IPv6 Global Unicast Address Format".
[10] S. Deering and B. Hinden, "Internet Protocol Version 6 (IPv6) specification," RFC 2460, 1998.
[11] R. Hinden S. Deering, "IP Version 6 Address Architecture," RFC 2373, 1998.
[12] J. Arkko, J. Kempf, B. Zill, P.Nikander, "SEcure Neighbor Discovery," RFC 3971, 2005.
[13] M. A. Miller, "Implementing IPv6 Second Edtion : Supporting the Next Generation Protocols," M&T Books, 1999.
[14] W. Stevens, "TCP/IP Illustrated V1, The Protocols," Addison-Wesley Publishing, 1994.

이 완 직 (Wan-Jik Lee)

- 1992년 경북대학교 통계학과 졸업
- 1994년 경북대학교 컴퓨터공학과 졸업(공학석사)
- 1997년 경북대학교 컴퓨터공학과 박사수료
- 1997년 3월 ~ 2006년 2월 밀양대학교 정보통신공학부 교수
- 2006년 3월 ~ 현재 부산대학교 바이오정보전자공학과 교수
- 관심분야 : 통신 프로토콜, 프로토콜 구현, 네트워크 보안

허 석 렬 (Seok-Yeol Heo)

- 1986년 경북대학교 전자공학과 졸업
- 1991년 경북대학교 컴퓨터공학과 졸업(공학석사)
- 1993년 경북대학교 컴퓨터공학과 박사수료
- 1992년 3월 ~ 2006년 2월 밀양대학교 컴퓨터공학부 교수
- 2006년 3월 ~ 현재 부산대학교 바이오정보전자공학과 교수
- 관심분야 : RFID/USN, 컴퓨터 네트워크, u-Health