리눅스 기반 IPv6 라우터 APD(Automatic Prefix Delegation) 프로토콜의 구현

허석렬*, 이완직*, 박정수**

Implementation of IPv6 Router APD(Automatic Prefix Delegation) Protocol based on Linux

Seok-Yeol Heo, Wan-Jik Lee, Jeong-Soo Park

요 약

IPv6에서는 호스트의 주소 재지정 뿐만 아니라 라우터에 대해서도 주소 재지정 메커니즘을 제공한다. IPv6에서는 기존 망 환경 외에도 Mobile-IP나 Ad-hoc 네트워크 상에서 라우터의 주소를 손쉽게 관리할 수 있는 메커니즘이 절대적으로 필요하다. 이러한 방법으로는 RR(Router Renumbering)이나 APDP(Automatic Prefix Delegation Protocol), DHCPv6 등이 있다.

본 논문에서는 라우터 주소 재지정 기법 중에서 APDP 기법을 리눅스 커널 2.4.x 기반 환경에서 구현하였다. 구현된 프로그램은 기존의 radvd와 통합하여 구성하였으며, 리눅스를 이용한 PC-라우터로 테스트베드를 구성하여 동작을 검증하였다.

Ⅰ. 서론

IPv6에서는 호스트의 IP 주소를 DHCP와 같은 서버를 이용하여 주소를 생성하거나, 호스트 스스로 주소를 생성한다. 전자의 방법을 상태 보존형 주소 자동 설정(Stateful auto-configuration) 방법이라고 하고 후자의 방법을 비상태형 주소 자동 설정(Stateless auto-configuration) 방법이라고 한다. 서버를 이용하는 방법은 호스트 측에서 DHCP 서버에 주소를 요청하면 서버에서 할당 가능한 주소 중 하나를 호스트 측에 할당하는 것이다. 따라서 서버는 대규모 데이터베이스를 갖추어야 하며 엄격한 관리가 요구된다. 반면 비상태형 주소 자동 설정 방식은 호스트가 자신의 인터페이스 ID정보와 라우터로부터 획득한 프리픽스 정보 또는 well-known 프리픽스 정보를 이용하여 스스로 주소를 생성한다. 따라서, 호스트가 자신의 주소에 대하여 생성 및 할당을 책임진다[1][3][5].

호스트가 자동으로 생성하는 주소 중에서 링크-로컬 주소는 DAD(Duplicate Address Detection)과정을 통해서 생성되지만 사이트-로컬이나 글로벌 주소를 획득하기 위해서는 라우터로부터 프리픽스와 같은 주소 설정에 필요한 정보를 획득한다. 라우터는 호스트의 주소 생성에 필요한 정보를 주기적으로 RA 메시지를 보내거나 호스트의 RS 메시지에 대한 응답으로 RA 메시지를 보냄으로써 전달한다.

호스트의 주소 재지정은 DAD 프로토콜의 RS와 RA메시지를 이용해서 이루어진다. 따라서 호스트는 라우터가 주기적으로 또는 요구에 의해서 이루어지는 메시지 교환을 통해서 주소 재지정을 자동적으로 수행할 수 있다. 호스트의 경우처럼 라우터의 경우에도 주소를 자동으로 설정할 수 있는 메커니즘이 필요한데, 라우터의 경우에는 RR(Router Renumbering)이나 APDP (Automatic Prefix Delegation Protocol), DHCPv6를 이용해서 자동적으로 라우터의 프리픽스를 재지정 한다.

본 논문에서는 라우터 주소 재지정 기법중에서 APDP 기법을 리눅스를 기반으로 하는 네트워크 환경에서 구현하였다. 라우터는 리눅스를 이용하여 PC-라우터로 구성하였으며 구현 프로그램은 기존의 radvd[1)]와 통합하여 구성하였다. 구현된 프로그램은 PC-라우터로 구성된 테스트베드에서 검증하였다.

본 논문의 2장에서는 IPv6의 주소 자동 설정 기법에 관한 내용을 기술하였고, 3장에서는 APD 프로토콜을 구현하기 위한 기능설계를 기술하였다. 4장에서는 구현된 APD 프로토콜의 기능별 동작 상세를 설명하였고 마지막으로 5장에서 결론을 지었다.

* 밀양대학교 컴퓨터정보통신공학부
** 전자통신연구원 차세대인터넷표준연구팀

1) 리눅스용 라우터 프로그램으로 리눅스 호스트를 라우터로 동작시킬 때 사용하는 프로그램

Ⅱ. IPv6 주소 자동설정 기법

1. IPv6 호스트 주소 자동설정

호스트에 IPv6 기반의 주소를 자동으로 할당하는 Address Auto-configuration 방법은 Stateful Auto-configuration 메커니즘과 Stateless Auto- configuration 메커니즘 두 가지로 구분된다.

• Stateful Auto-configuration

Stateful Auto-configuration 메커니즘에서 호스트는 DHCP 서버로부터 주소를 비롯한 모든 네트워크 정보를 제공받으며, DHCP 서버가 주소 할당에 대한 관리를 제공한다. 또한 각 호스트는 이러한 DHCP 서버를 찾기 위한 DHCP 요청(solicit) 메시지를 보내며 이를 받은 DHCP 서버는 응답 메시지를 통해서 호스트의 주소설정에 관련된 모든 네트워크 정보를 제공한다.

DHCPv6는 IPv6용 DHCP 프로토콜로서 Stateful Auto-configuration을 지원한다. DHCP는 IP 주소, 라우팅 정보, 운영체제 설치 정보, 디렉토리 서비스 정보 등을 각 네트워크 노드 사이의 로컬구성 파일에 분산시키기 보다는, 소수의 DHCP 서버상에 이러한 정보와 네트워크 자원들의 관리를 집중함으로써 유지비용을 줄일 수 있도록 하는 메커니즘이다[6].

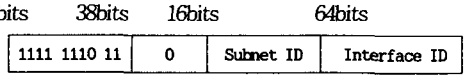
• Stateless Auto-configuration[3]

Stateless Auto-configuration 메커니즘은 호스트 스스로 주소를 생성한다. 링크-로컬 주소에 대해서는 자신의 인터페이스 ID정보와 링크-로컬 prefix 정보를 결합한 뒤 DAD(Duplicate Address Detection)를 수행함으로써 주소를 생성한다. 사이트/글로벌 주소를 생성하기 위해서는 라우터가 자신의 링크 내에 있는 호스트들에게 주소 프리픽스 방송을 하고, 각 호스트는 이를 받아 자신의 인터페이스 식별자와 결합하여 IPv6 사이트/글로벌 주소를 생성한다.

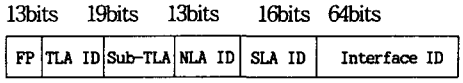
링크-로컬 주소는 단일 링크 내에서만 사용가능하며, 경계 라우터가 외부 망으로 전파되는 것을 차단한다. 링크-로컬 주소는 라우터가 존재하지 않거나 DHCP 서버가 존재할 경우에도 사용 가능하다. 링크-로컬 주소는 라우터의 도움 없이 주소 자동 설정 방식에 의해 획득 될 수 있다.

IPv6 주소의 앞 부분에 위치하는 비트 집합을 프리픽스(prefix)라 하며 그림 1.a의 1111111010(FE80::/10)은 링크-로컬용으로 미리 정해진 프리픽스이다. 일반적으로 링크-로컬 주소를 생성하는 방법은 well-known 프리픽스 정보에 자신의 인터페이스 ID 정보를 붙여서 생성한다. 이렇게 만들어진 링크-로컬 주소는 링크 내에서 주소의 유일성을 검증 받지 않았기 때문에 임시 주소(tentative address)라 불리며 중복 주소 검출(DAD : Duplicate Address Detection)과정을 통해서 유일성을 검증한다.

사이트-로컬이나 글로벌 주소를 설정하고자 하는 호스트는 라우터에게 RS 메시지를 보내거나 라우터가 주기적으로 보내는 RS 메시지로부터 주소 설정에 필요한 정보를 얻을 수 있다. 호스트는 RA 메시지의 유효성 등을 검사한 후 자신의 인터페이스 ID와 수신한 프리픽스 정보를 이용하여 그림 1.b와 같은 주소 구조에 따라 사이트-로컬 또는 글로벌 주소를 생성한다.



(a) 사이트-로컬 주소 형식



(b) 글로벌 주소 형식

그림 1. 사이트-로컬 및 글로벌 주소 형식

다음 그림 2에 위에서 설명한 Stateless Auto-configuration 메커니즘을 이용한 링크-로컬 및 사이트 로컬, 글로벌 주소 생성을 위한 과정을 보였다.

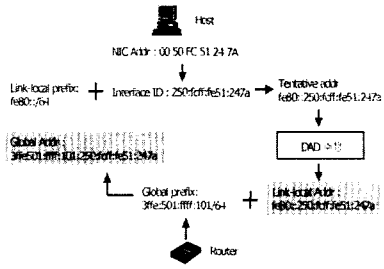


그림 2. Stateless Auto-configuration 예

2. IPv6 라우터 프리픽스 설정

IPv6 호스트는 현재 네트워크 상의 라우터에서 전송 받은 프리픽스 주소부분과 자신의 로컬 주소 부분(NIC 주소로 생성)을 결합하여 유일한 IPv6 주소를 자동 생성하는 기능을 가지고 있다. 하지만 라우터에게 할당되는 프리픽스 주소는 주로 네트워크 관리자에 의해서 설정되는데 현재 RFC 2894의 RR 프로토콜은 이들 라우터의 프리픽스 주소 부분을 원격에서 갱신, 수정할 수 있는 기능을 가진 프로토콜이다.

• PC-라우터 prefix 수동 설정

IPv6를 지원하는 PC-라우터는 리눅스와 Free BSD를 상에서 설치되어 응용 레벨에서 동작하며 raw IPv6 소켓을 사용하여 라우터 동작을 수행한다. 리눅스는 radvd와 관련 파일로 구성되어 있고 Free-BSD는 rtadvd와 관련 파일로 구성되어 있다. 이 프로그램들은 시스템 내부의 설정 파일(리눅스의 경우 /etc/radvd.conf, Free-BSD의 경우 /etc/rc.conf 파일)을 참조하여 라우터의 프리픽스와 관련된 내용들을 설정하고 RA(prefix_ information_opt) 메시지를 멀티캐스트 함으로써 네트워크 상의 호스트들에게 프리픽스를 제공한다.

라우터 프리픽스의 수동설정은 네트워크 상의 라우터들이 계층 구조를 가지는 환경이나 네트워크의 토폴로지가 동적으로 변화하는 환경에서는 운영의 어려움이 있기 때문에 라우터의 프리픽스를 자동으로 설정하는 메커니즘이 필요하다.

• 라우터 prefix 자동 설정

네트워크의 구성이 동적으로 변화하거나 관리 상의 이유로 사용중인 라우터의 프리픽스를 바꿀 필요가 있을 때, 라우터의 프리픽스를 관리자가 직접 수동으로 설정하는 것은 네트워크 변화에 대해 적응성이 떨어지며 일관성을 유지하기가 어렵다.

라우터의 프리픽스를 자동설정 하는 것은 (단말) 라우터들의 프리픽스를 생성하는 것뿐만 아니라 프리픽스의 삭제, 변경과 이미 사용중인 프리픽스의 재지정(prefix-lifetime 등을 고려) 등을 포함한다. 이러한 라우터 프리픽스 자동설정 메커니즘을 통하여 ISP업체들은 원격 라우터 설정과 관리가 가능하며 Ad-hoc, 홈 네트워크 구축에 활용 될 수 있다.

라우터 프리픽스 자동 설정 메커니즘은 라우터의 프리픽스 주소 부분을 원격에서 갱신, 수정할 수 있는 기능을 가진 프로토콜이다. 현재, 이와 같은 기능을 갖는 프로토콜은 Router Renumbering(RFC 2894)와 APDP(Automatic Prefix Delegation Protocol), RA-PD-option(Router Advertisement Prefix Delegation Option) 세 가지로 구분할 수 있다. 이 중에서 RR은 라우터 프리픽스 재설정(변경)에 초점을 맞춘 프로토콜이며, APDP는 클라이언트/서버 모델을 기반으로 프리픽스 요청, 해제 등의 기능을 가지고 있으며 RA-PD-option은 확장된 RA 메시지를 이용하여 라우터 프리픽스와 관련된 정보를 전달하는 방법이다.[1][4][6]

Ⅲ. APD 프로토콜 기능설계

1. APDP(Automatic Prefix Delegation Protocol)

프리픽스 자동 위임 프로토콜은 IPv6 네트워크 프리픽스(network prefix)의 자동 위임을 위한 프로토콜이다. 즉, 요청 라우터(Requesting router)가 특정 크기의 프리픽스를 위임 라우터(Delegation Router)에게 요청하는 기능과 자신이 지원할 수 있는 라우팅 프로토콜을 알려주어서 위임 라우터가 서로간에 지원 가능한 라우터를 결정할 수 있도록 돕는 기능을 수행한다. 현재, 이 APDP를 지원하기 위해서 그림 6과 그림 7과 같은 새로운 두 가지 형식의 ICMP 메시지를 정의하고 있으며 이들은 프리픽스 위임을 요청하기 위해 사용되는 프리픽스 요청(Prefix Request)과 새로운 프리픽스 및 생존기간, 라우팅 프로토콜 정보 또는 에러정보를 공지하기 위해 사용되는 프리픽스 위임(Prefix Delegation) 메시지이다.[2]

이 두 가지 메시지를 이용하여, 구체적으로 Code 필드에 의해 지시되는 값에 따라 다음과 같은 6가지 기능을 수행한다. 이 중에서 두 가지 기능, 즉 인증과 확인 및 프리픽스 위임은 위임 라우터에 의해 개시되고 나머지 기능은 요청 라우터에 의해 개시된다.

- **Delegator Query** : 요청 라우터가 모든 위임 라우터를 대상으로 링크-로컬 멀티캐스트 주소로 위임 라우터를 찾기 위한 메시지를 전송하는 기능이다.
- **Initial Request** : 위임자 query에 의해 찾아진 위임 라우터에게 프리픽스와 라우팅 프로토콜 정보를 요청하는 기능이다.
- **Authentication and Authorization** : 초기 요청, 프리픽스 재 활성화 및 반환 과정에서 요청한 메시지에 대한 인증과 확인 기능이며, 인증은 요청자와 위임자간에 보호연관(Security Association)이 미리 존재해야 한다. 확인 기능은 요청한 프리픽스의 길이가 관리정책에 부합 하는 지를 검사하는 기능이다.
- **Prefix Delegation** : 인증과 확인 과정을 통과한 요청에 따라 프리픽스를 할당해 주는 기능이다.
- **Prefix Refresh** : 프리픽스 위임 과정에서 제공되는 프리픽스의 생존기간은 제한되는데, 이 생존기간이 지나기 전에 프리픽스 재활성화 메시지를 보내서 계속해서 프리픽스를 사용하겠다는 것을 알리는 기능이다.
- **Prefix Return** : 더 이상 프리픽스를 사용하지 않을 경우 프리픽스를 위임 라우터에게 돌려주는 기능이다.

2. 프로토콜 구현 환경 및 고려사항

• 구현환경

본 논문의 APD 프로토콜 구현은 그림 3과 같은 환경에서 수행되었다.

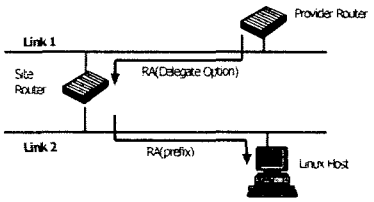


그림 3. APD 프로토콜의 구현 환경

본 논문의 구현에서 전체 시스템(라우터, 호스트)들은 모두 펜티엄급의 PC를 사용하였으며, 라우터들은 2개 이상의 NIC를 설치하고, 라우터로 설정하였다. 또한 프로토콜 구현을 위해 사용한 언어는 C 언어이며, ICMP 형태의 APDP, RA, RS 패킷들의 전송을 위해 IPv6 Raw Socket 인터페이스를 사용하였다. APD 프로토콜 구현은 Automatic Prefix Delegation Protocol for Internet Protocol Version 6(IPv6) <draft-haberman-ipngwg- auto-prefix-0.2.txt> 문서에 준하여 수행되었다.

3. APD 프로토콜 상세 설계

3.1 R/R 프로그램 상세 구현 설계

앞에서 설명한 바와 같이 R/R 프로그램은 APDP 처리 부분의 기능을 기존의 radvd 프로그램에 첨가하여 수정한 형태로 구현하였으며, 일부 코드들은 기존의 radvd 프로그램 코드를 그대로 사용하였다. 구현 설계에는 주요 자료구조, 전체 프로토콜 상태 천이, 각 APD 프로토콜 기능별 구현 상세 등이 포함된다.

• 주요 자료 구조 설계

본 R/R 프로그램의 주요 자료 구조는 struct Context, struct Interface, struct Advprefix, struct timer_lst 이렇게 4 가지이며, 이들 중 새로이 추가된 자료 구조는 Context 하나이며, 나머지 자료 구조들은 기존 radvd 프로그램에 사용하던 자료 구조를 본 과제에서 추가 확장한 것들이다. 이들 자료 구조의 정의는 R/R 프로그램의 radvd.h 파일에 있다.

이 들 중요한 자료구조의 형태를 그림 4에 나타내었다.

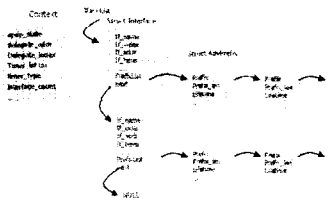


그림 4. 주요 자료구조 형태

• APDP 프로토콜 상태 천이 동작

본 R/R 프로그램에서 APD 프로토콜 동작과 RA를 전송하는 동작을 정확히 수행하기 위해서 프로토콜 상태 천이를 정의하였다. APD 프로토콜에 사용되는 APDP 메시지의 송수신 상태를 정확히 정의하고, 기존 Radvd 프로그램의 여러 동작(radvd.conf 분석 및 처리)과 호환성을 위해서는 전체 프로토콜(Context)에 대한 상태 천이와 각 프리픽스별 상태 천이를 나누었다.

우선 전체적인 프로토콜 상태 천이에서 정의된 상태는 다음과 같다.

CLOSED: 모든 프로토콜 동작을 종료한 상태
DR_SERACHING : 각 인터페이스로 DELEGATE_QUERY 메시지를 전송하여 D/R을 검색하고 있는 상태.
RADVD : D/R 검색에 실패하고, radvd.conf 파일에 의한 수동 RA 전송 동작 상태
DELEGATOR: D/R 검색에 성공하여 APDP 동작을 수행하는 상태
CLOSING : R/R 프로그램 종료를 위해 현재 할당받은 prefix들의 리턴을 수행하고 있는 상태

이와 같은 상태에 따라 정의된 프로토콜 상태 천이도를 다음의 그림 5에 나타내었다.

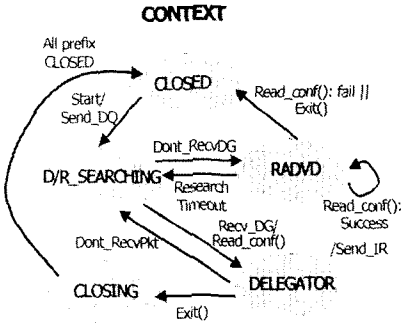


그림 5. R/R의 프로토콜 상태 천이도

3.2 D/R 프로그램 상세 구현 설계

본 절에서는 APD 프로토콜의 D/R 프로그램을 구현하기 위해 상세 설계 내용을 담고 있다. D/R 프로그램은 Prefix pool에 대한 기본 설정 정보를 가지고 있는 파일을 읽고 prefix pool을 초기화하는 부분과 R/R의 요청에 의해서 prefix를 할당하고 관리하며 반환된 prefix를 재 사용이 가능하도록 처리하는 부분으로 구성되어 있다. 각 기능별로 상세한 설계 내용을 다음에 설명한다.

• 주요 자료 구조 설계

D/R이 할당할 프리픽스에 관한 모든 정보를 가지고 있는 prefix_pool.conf 파일 형식이다. Static은 64비트 프리픽스와 생존기간 R/R의 링크-로컬 주소, 인터페이스 인덱스를 가지며, dynamic은 48비트의 프리픽스와 생존기간 인터페이스 인덱스를 가진다.

프리픽스 pool을 구성하는 노드의 기본자료 구조로서 프리픽스와 관련된 모든 정보를 정의하고 있다. 프리픽스의 길이와 생존기간, 할당 인터페이스, 정적인 프리픽스를 할당할 경우에 필요한 R/R의 링크-로컬 주소, 할당될 때의 시간을 저장하는 타임스탬프, 노드의 할당 상태를 나타내는 상태 필드로 구성되어 있다.

```
struct prefix_addr_list {
        struct in6_addr prefix_addr;   //할당할 프리픽스
        uint8_t length;                //프리픽스 길이
        uint32_t lifetime;             //생존기간
        struct in6_addr llocal_addr;   //RR의 링크-로컬 주소
        int if_index;                  //인터페이스 인덱스
        struct timeval tv;     // last used time_stamp
        int state;   // USED, RETURNED,FREE
        struct     prefix_addr_list *next;
        struct     prefix_addr_list *down;
};
```

D/R의 NIC에 관련된 정보를 저장하고 있는 자료 구조이다. 인터페이스는 ipv6 주소와 정수 값을 갖는 인덱스로 구성된다.

```
struct Interface {
  char Name[IFNAMSIZ];      /* interface name */
  struct in6_addr   if_addr;    /* ipv6 localaddr */
  int if_index;     /* interface index */
  struct Interface  *next;
};
```

Ⅳ. APD 프로토콜 구현

1. APD R/R 프로토콜 기능별 동작 상세

R/R 프로그램은 주요한 동작을 다음과 같이 모두 4가지로 나누어 정의하였다.

Delegator Query 동작 : R/R 프로그램의 초기 동작으로써 D/R을 검색하는 동작
Initial Request 동작 :D/R 검색 후, D/R 에게 새로운 프리픽스를 할당받는 기능
Renewal Request 동작 :이미 사용하던 프리픽스에 대한 재사용(연장) 요청을 수행하는 기능
Prefix Return 동작:R/R 프로그램의 종료시에는 현재 사용하고 있는 모든 프리픽스를 반환하는 기능

그리고 4가지 동작에서 발생할 수 있는 모든 상황을 정의하여 프로그램의 수행 시 처리할 수 있도록 설계하였다. 이 중에서 Delegator Query 동작에 대한 동작 설계를 예를 들어 아래에 설명하였다.

• Delegator Query 동작 설계

R/R 프로그램의 초기 동작으로써 D/R을 검색하는 동작이다. 가장 일반적인 상태에 대한 타임 플로우는 다음과 같다.

CASE 1) 정상적인 D/R 검색 성공(하나의 D/R만 검색)

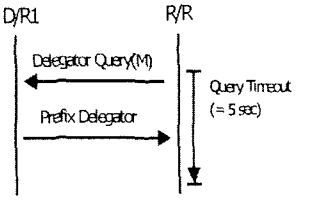


그림 6. 정상적인 D/R 검색 성공

정상적으로 D/R 검색하고 이를 완료하는 동작이다. 또 다른 D/R 존재할 수 있으므로 5초 동안 가능한 모든 D/R의 Prefix Delegator 메시지를 수신한다.

CASE 2) 두 개 이상의 D/R 검색

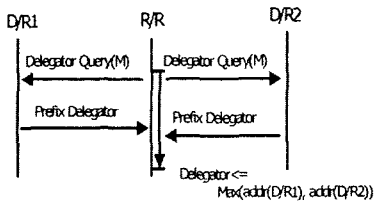


그림 7. 두 개 이상의 D/R 검색

두 개 이상의 D/R이 검색되는 상황이다. 이 상황에서는 새로운 D/R이 검색될 때 마다 기존의 D/R 링크 주소와 새로운 D/R의 링크 주소를 비교하여 주소 값이 더 큰 D/R의 주소를 프로토콜 Context에 저장하도록 한다. 이 상황에 대한 처리는 프로토콜 draft 상세에 준해서 구현하였다.

CASE 3) D/R 검색 실패

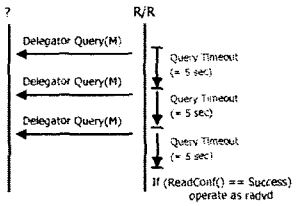


그림 8. D/R 검색 실패

D/R의 검색에 실패하는 상황이다. 그림 8에서는 표현되지 않았지만 실제 구현에서는 R/R의 모든 인터페이스로 Delegator Query 메시지를 멀티캐스트 한다. 각 5초 동안 세번의 전송에 대한 응답이 없으면 D/R 검색이 실패하였다고 판단한다. 이후 기존 Radvd 프로그램과 같이 내부의 radvd.conf라는 설정 파일을 읽는 동작을 수행한다. 만약 radvd.conf 파일이 존재하고 그 분석이 성공적이라면 기존의 radvd 동작 상태를 정의한 RADVD 상태가 되고 radvd.conf를 읽는 동작이 실패한다면 R/R 프로그램을 종료하도록 구현하였다.

2. APD D/R 프로토콜 기능별 동작 상세

• prefix pool 구조

prefix_pool.conf 파일을 읽어서 prefix pool을 초기화 한다. 가로로 링크된 리스트가 초기화된 pool이며 동적 프리픽스가 할당 될 때마다 동적 프리픽스 초기 노드(가로 링크의 48 노드)에서부터 down 링크로 생성되어 할당된다.

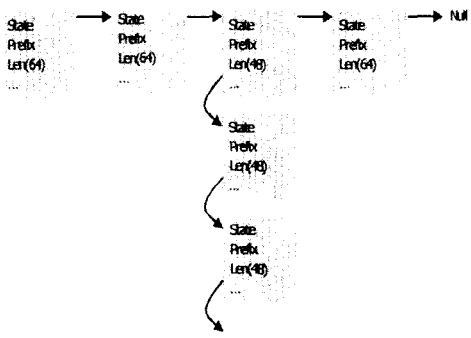


그림 9. 프리픽스 pool 구조

• prefix 할당 방법

구성된 프리픽스 pool을 유지하며 R/R이 프리픽스 요청을 할 때 조건을 검사하여 적합한 프리픽스를 할당하는 알고리즘이다. R/R이 특정 프리픽스를 요구할 때에는 모든 리스트에 대해서 요청한 프리픽스와 같은 노드를 검색하여 프리픽스를 할당하며, 임의의 64비트 프리픽스를 요청할 경우에는 동적 리스트에 대해서만 할당 가능한 프리픽스 검사를 수행한다.

• 타이머에 의한 반환 주소 처리

동적 프리픽스를 할당 받은 R/R이 프리픽스를 반환하면 D/R은 이 프리픽스를 즉시 FREE상태로 만드는 것은 R/R에서 순간적인 power off와 같은 상황이 발생할 경우, 계속 사용중인 프리픽스가 바뀔 수 있기 때문에 바람직 하지 않다. 따라서, 다음과 같이 타이머를 이용하여 반환된 주소를 처리하면 위와 같은 상황을 대비할 수 있다.

3. 프로그램 테스트

본 구현의 프로그램의 데모 환경은 다음과 같다. 데모 시에 R/R 프로그램의 수행을 radvd m stderr 과 같이 수행하여 R/R 프로그램 수행 시에 모든 로그 정보를 화면을 출력시켰다.

그림 10. APDP 테스트 환경

APDP의 D/R 프로그램에 대한 수행 환경 캡처 그림은 다음과 같다.

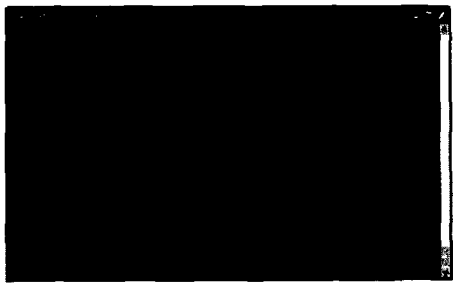


그림 11. APDP D/R 프로그램 수행결과

위의 그림에서 먼저 프로그램 수행 시에 현재 사용 가능한 인터페이스의 정보(인터페이스 이름, 인덱스, 각 인터페이스 IPv6 링크 로컬 주소 등)가 1-2 라인에 출력됨을 알 수 있으며, 그 다음으로 현재 Prefix

Pool 설정 파일을 읽고 Prefix Pool 설정 형태가 출력된다. 현재 모두 6개 Prefix가 설정되었음 알수 있으며, 그 중 마지막 프리픽스는 길이 48의 Dynamic Prefix임을 알 수 있다.

이 후, R/R 프로그램이 전송한 APAP Delegater Query 메시지 수신과 Prefix Delegator 패킷의 송신 로그를 볼 수 있다. 이 후 Dynamic Prefix에서 하나의 Prefix를 R/R에게 할당하는 과정이 출력된다. 이 D/R과 같이 동작 R/R 의 로그 상태는 다음 그림과 같다.

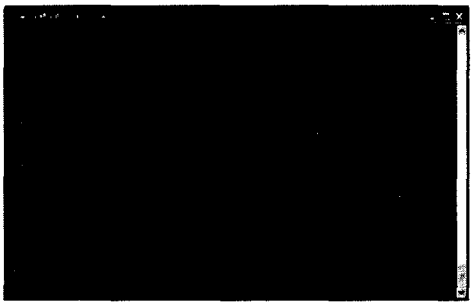


그림 12. APDP R/R 프로그램 수행 결과

위의 그림에서 R/R 전형적인 동작들을 확인할 수 있다. 가장 먼저 각 인터페이스(eth0, eth1)로 Delegator Query 메시지를 전송하고, 그 후 Prefix Delegator 패킷을 수신하여 D/R 검색을 완료하는 동작을 보여준다. 이후 Initial Request 송신 및 Prefix Delegated 패킷 수신(이때 수신 Prefix 관련 정보가 여러 줄에 거쳐 출력된다.)이 성공적으로 수행되고, Renewal Request 전송 및 Prefix Delegated의 수신 등이 연속해서 출력됨을 알 수 있다.

프로그램 수행 시에 현재 사용 가능한 인터페이스의 정보(인터페이스 이름, 인덱스, 각 인터페이스 IPv6 링크 로컬 주소 등)가 1-2 라인에 출력됨을 알 수 있으며, 그 다음으로 현재 Prefix Pool 설정 파일을 읽고 Prefix Pool 설정 형태가 출력된다. 현재 모두 6개 Prefix가 설정되었음 알수 있으며, 그 중 마지막 프리픽스는 길이 48의 Dynamic Prefix임을 알 수 있다.

이 후, R/R 프로그램이 전송한 APAP Delegater Query 메시지 수신과 Prefix Delegator 패킷의 송신 로그를 볼 수 있다. 이 후 Dynamic Prefix에서 하나의 Prefix를 R/R에게 할당하는 과정이 출력된다.

또한 구현 환경에서 R/R에 의해 운영되는 망의 일반 IPv6 호스트에서도 radvdump 프로그램이나, ifconfig 명령을 수행하여, R/R이 보내주는 RA 메시지에 의해 새로운 Prefix를 수신하고, Global 주소의 생성을 확인 할 수 있었다.

V. 결론

IPv6에서는 호스트의 주소 재지정 뿐만 아니라 라우터에 대해서도 주소 재지정 메커니즘을 제공한다. IPv6에서는 기존 망 환경 외에도 mobile-ip나 Ad-hoc 네트워크 상에서 라우터의 주소를 손쉽게 관리할 수 있는 메커니즘이 절대적으로 필요하다. 이러한 방법으로는 RR(Router Renumbering)이나 APDP(Automatic Prefix Delegation *Protocol*), DHCPv6 등이 있다.

본 논문에서는 라우터 주소 재지정 기법 중에서 APDP 기법을 리눅스 커널 2.4.18 기반 환경에서 구현하였다. 라우터는 리눅스를 이용하여 PC-라우터로 구성하였으며 구현 프로그램은 기존의 radvd와 통합하여 구성하였다. 구현된 프로그램은 PC-라우터로 구성된 테스트베드에서 테스트를 통해 검증하였다. 앞으로 구현된 APDP 프로그램을 6BONE 환경에서 테스트할 예정이다.

참고문헌

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

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

[3] S. Thompson and T. Narten, "IPv6 Stateless Address Autoconfiguration,", RFC 2462, December 1998.

[4] M. Crawford, "Router Renumbering for IPv6," RFC 2894, August 2000.

[5] R. Hinden, S. Deering, "IP Version 6 Addressing Architecture," RFC 2373, 1998.

[6] J. Bound, M. Carney, C. Perkins and R. Droms(ed.), "Dynamic Host Configuration Protocol for IPv6(DHCPv6)," draft-ietf-dhc-dhcpv6-21.txt, November 2001.

[7] IPv6 포럼 코리아, "차세대 인터넷 프로토콜," 다성 출판사, 2002.