능동 네트워크에서의 동적 라우팅 프로토콜

안상현* 김경춘*0 한민호** 나중찬**
* 서울시립대학교 컴퓨터·통계학과
** 한국전자통신연구원 정보보호기술연구본부
ahn@venus.uos.ac.kr, netiv@orgio.net0, {mhhan, njc}@etri.re.kr

Dynamic Routing Protocol in the Active Network

Sanghyun Ahn* Kyeongchoon Kim*0 Min-Ho Han** Jung-Chan Na**
* Dept. of Computer Science & Statistics, University of Seoul
** Information Security Technology Division, ETRI

요 약

기존 능동 네트워크 기술 분야에서 능동 네트워크 라우팅은 정적인 토폴로지 구성을 가정하고 있다. 그렇지만 IP 라우팅 영역에서와 같이 능동 네트워크 영역에서도 능동 패킷의 네트워크 상태에 따른 동적인 라우팅이 이루어져야 한다. 현재까지 SLRP와 TCOM500 PLAN 프로젝트와 같은 프로토콜들이 제시되어 있지만, 두 프로토콜 모두 정적으로 구성되어 있는 토폴로지를 기반으로 하기 때문에 완전하게 동적인 프로토콜은 아니다. 본 논문에서는 좀더 동적이고 기존 프로토콜과의 호환성을 갖춘 능동 네트워크 라우팅 프로토콜로서 AOSPF를 제안한다. AOSPF는 능동 네트워크 라우팅을 위해 새로운 LSA를 추가함으로서 OSPF를 확장한 것이다. 새로운 LSA는 해당 라우터가 능동 라우터임을 명시하며, 이를 위해 LSA 패킷내의 OPTION 필드들 가운데 사용되고 있지 않은 비트 가운데 하나를 능동 라우터 명시 비트로 사용한다. 이러한 개념은 멀티캐스트를 위해 OSPF를 확장한 것으로부터 가져왔다. 본 논문에서는 AOSPF의 동작 과정과 AOSPF를 통해 구성된 능동 네트워크 토폴로지 정보를 기반으로 하는 능동 패킷의 전달 방법에 대해 기술한다.

1. 서 론

능동 네트워크는 능동 패킷에 포함된 코드를 수행할 수 있는 능동 노드들이 서로 지역적 혹은 전역적으로 연결된 네트워크로서 IP 네트워크와 같이 존재할 수는 있지만, 능동 패킷의 처리에 있어서 IP 네트워크와 서로 구별된다. 능동 패킷을 받은 일반 IP 라우터는 능동 패킷 내의 능동 코드를 수행할 필요가 없으며, 단지 IP 헤더에 명시된 목적지로 패킷을 전달할 뿐이다. 능동 노드의 경우는 능동 패킷에 포함된 능동 코드를 해석하고 실행한 후, 경우에 따라 그 결과를 다음 능동 노드로 전달한다. 여기서 우리가 해결해야 할 문제는 다음과 같다. 첫째, 능동 패킷을 다음 능동 노드로 전달할 때 직접 다음 능동 노드로 패킷을 전달할 수 있느냐는 것과 둘째, 패킷을 전달할 다음 능동 노드를 어떤 정보를 기반으로 선택하느냐는 것이다. 위 두가지 문제에 대한 해결책을 찾기에 앞서 우리는 능동 네트워크의 구성 방법에 대해서 알아야 한다. IP 네트워크의 경우 다음 라우터는 물리적으로 직접 연결되어 있는 라우터이지만, 현재 인터넷의 발전 과정으로 미루어 볼 때 기술적, 상업성의 이유로 인터넷의 모든 라우터가 능동 라우터로서 동작할 수 없기 때문에 능동 네트워크 상의 능동 노드는 물리적으로 직접 연결되지 못한다. 이는 능동 네트워크 상에서의 패킷 전달 방법과 라우팅 프로토콜을 복잡하게 만드는 가장 큰 요인이다.

패킷 전달 방법에 있어서는 물리적으로 연결되어 있지 않은 다음 능동 노드로 패킷이 전달되는 것을 보장하기 위한 추가적인 방법이 필요하다. 본 논문에서는 이를 위해 능동 패킷에 추가적인 IP 헤더를 캡슐화[6]하여 전달하는 IP 터널링 방법을 사용하도록 제안한다. IP 터널링 방법은 현재 Mobile IP[7]와 멀티캐스트[8]와 같은 많은 프로토콜에서 사용되고 있는 방법이다. 다음으로 다음 능동 노드를 선택하는 과정에 있어서는 능동 패킷의 처리 결과와 능동 네트워크 토폴로지 정보를 고려해야 한다. 만약 능동 패킷에 라우팅 정보가 포함되거나 능동 패킷의 처리 결과를 통해 라우팅 정보를 얻을 수 있다면, 이 라우팅 정보를 통해 다음 능동 노드를 선택할 수 있지만, 그렇지 못한 경우는 능동 노드 자신이 유지하고 있는 라우팅 정보를 기반으로 다음 능동 노드를 결정해야 한다. 만약, 인터넷의 모든 라우터가 능동 라우터로서 동작한다면 기존 인터넷 라우팅 프로토콜들에 의해서 만들어진 라우팅 정보를 그대로 사용할 수 있지만, 능동 네트워크가 서로 물리적으로 연결되어 있지 못하기 때문에 능동 노드들로만 구성된 능동 네트워크 토폴로지 정보를 유지하고 능동 패킷의 라우팅을 위한 능동 라우팅 테이블을 구성해야 한다. 즉, 새로운 능동 네트워크 라우팅 프로토콜이 요구된다. 능동 네트워크 라우팅 프로토콜은 일반적인 인터넷 라우팅 프로토콜보다 좀더 복잡하다. 이웃한 능동 노드들이 물리적으로 직접 연결되어 있지 못하기 때문에 이웃한 노드에 대한 최신의 정보(이웃한 노드들의 리스트, 이웃한 노드와의 링크 상태)를 유지하고 그에 따른 대응이 어렵기 때문이다.

현재 대부분의 능동 네트워크 기술에서 능동 네트워크 토폴로지는 정적으로 구성하거나 혹은 구성되어 있다는 것을 가정하고 있다. 그렇지만 정적인 구성 방법은 구현은 용이하지만 관리자의 많은 노력을 수반하며 특히 네트워크의 동적인 상황을 반영하지 못하며, 능동 네트워크 상의 특정 능동노드를 이용해 세션을 설정하는 능동응용[4]에게는 상황에 따른 정확한 네트워크 토폴로지를 제공해 주지 못하는 단점이 있다. 이러한 문제점을 해결하기 위해 현재까지 제안된 라우팅 프로토콜들로서는 SLRP(Service Layer Routing Protocol)[1]와 TCOM500 PLAN 프로젝트[2] 등이 있다. 두 프로토콜 모두 좀더 동적인 라우팅을 위해 설계되었지만, 두 프로토콜 모두 정적인 토폴로지가 구성되어 있는 상태(즉, 이웃한 능동 노드에 대한 정보가 정적으로 설정되어 있는 상태)를 기반으로 동작하기 때문에 완전하게 동적인 프로토콜이라고는 할 수 없다.

본 논문에서는 동적인 능동 네트워크 라우팅 프로토콜로서 AOSPF(Active OSPF)를 제안한다. 2장에서는 기존의 능동 네트워크 라우팅 프로토콜인 SLRP와 TCOM500 PLAN 프로젝트를 알아보고, 3장에서는 AOSPF의 기본 개념과 토폴로지 구성방법, 4장에서는 능동 패킷의 라우팅 과정, 5장에서는 향후 연구 방향에 대해 기술한다.

2. 기존 연구

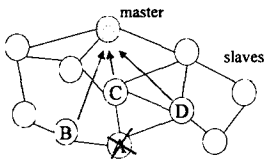
PLAN(Programming Language for Active Networks)[3]은 정적인 경로배정에 기반한 아주 간단한 경로배정 방법을 사용한다. SLRP와 TCOM500 PLAN 프로젝트 모두 PLAN을 위한 좀더 유연성 있는 경로배정 방법을 제공하기 위해 제시된 프로토콜들이다. 두 프로토콜 모두 미리 설정된 토폴로지를 기반으로 동작하며, 노드간 링크 정보에 기반한 링크 상태 프로토콜(link state protocol)이다. 두 프로토콜은 토폴로지 정보를 저장하고, 경로를 계산하는 방식에 차이점이 있다. SLRP는 중앙 집중화 경로배정 방법을 사용하며, TCOM500 PLAN 프로젝트는 분산 라우팅 방법을 사용한다.

SLRP의 경우 능동노드들은 주·종 관계를 이루며 우선 주노드는 자신의 하위 노드로부터의 이웃한 능동노드들에 대한 링크정보를 수집해서 전체 네트워크 토폴로지 구성 데이터베이스(정보)를 유지한다. 이 정보를 기반으로 Dijkstra's 알고리즘을 사용하여 각 하위 노드들 각각에 대한 최단 경로 트리(shortest path tree)를 계산하고, 포워딩 테이블(forwarding table)을 만들어 각 하위 노드들에게 전달한다. 하위 노드들은 주기적으로 이웃한 능동노드들과의 링크의 존재 유무를 검사하

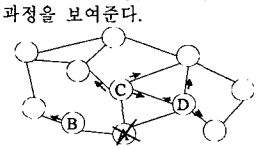
고, 변경된 링크 정보를 주노드이게 전달하며, 주노드로부터의 포워딩 테이블에 의존하여 능동 패킷의 경로배정을 결정한다.

TCOM500 PLAN 프로젝트의 경우 능동노드들은 서로 대등한 관계로 동작한다. 링크 상태 정보를 교환하기 위해 OSPF[5]로부터 프로토콜의 기본 개념을 가져왔으며, 동작 방식은 기존 OSPF 프로토콜과 같다. 차이점은 링크 상태 정보 패킷을 교환하는 노드가 PLAN이 동작하는 능동노드가 되며, 비능동노드는 고려되지 않는다는 것이다. 각 노드는 다른 노드로부터의 링크 상태 정보를 통해 경로배정을 결정한다.

[그림 1]과 [그림 2]는 하나의 노드가 고장났을 때 SLRP와 TCOM500 PLAN 프로젝트의 처리 과정을 보여준다.



[그림 1] SLRP



[그림 2] TCOM500 PLAN 프로젝트

그림에서 노드 A가 고장이 났을 때, SLRP의 경우 이웃한 노드 B, C, D가 A와의 링크가 고장이 났음을 주노드에게 알려주는 것을 볼 수 있고, TCOM500 PLAN 프로젝트에서는 이웃한 노드 B, C, D가 A와의 링크 정보를 네트워크의 다른 노드들에게 알려주는 것을 볼 수 있다. 이러한 과정을 통해 두 프로토콜은 좀더 동적인 경로배정 방법을 제공하긴 하지만, 여전히 몇 가지 문제점을 안고 있다.

SLRP의 경우, 주노드의 고장시 심각한 결과를 초래하게 되며 SLRP가 이러한 상황에 대처할 수 있는 기법들을 제시하고는 있지만 주노드의 복구시까지 네트워크의 상황에 대처하지 못하는 것은 필연적이다. 또한 주종관계를 사용하기 때문에 분산환경에서 동작하는 링크 상태 프로토콜보다 네트워크의 상태 변경에 대한 응답시간도 문제가 된다. 더욱 중요한 문제는 두 프로토콜 모두 PLAN 네트워크 토폴로지가 정적으로 설정되어 있는 상태를 기반으로 동작한다는 것이다. 즉, 두 프로토콜 모두 PLAN의 정적인 라우팅 방법에 대한 유연성을 좀더 증가시키기 위한 방법일 뿐, 완전하게 동적인 프로토콜은 아니다.

3. AOSPF 개관

우리는 다음과 같은 목적을 두고 능동 네트워크 라우팅 프로토콜을 설계하였다.

첫째, 완전하게 동적인 프로토콜을 설계한다. 앞서 두가지 기존 연구는 초기 네트워크 토폴로지가 정적으로 설정되어 있음을 가정하였지만, 우리가 제시하는 프로토콜은 이웃한 노드에 대한 정보가 없는 상태에서부터 시작한다. 둘째, 능동응용에게 능동 네트워크의 토폴로지 정보를 제공한다. 셋째, 기존 네트워크 프로토콜을 활용함으로써 프로토콜 개발 시간과 노력을 단축시킨다.

위와 같은 목적을 위해 설계한 능동 네트워크 라우팅 프로토콜이 AOSPF(Active OSPF)이다. AOSPF의 가장 주요한 개념은 기존 OSPF 프로토콜을 그대로 사용하고, 추가적으로 새로운 LSA(Link State Advertisement) 메시지를 정의한다는 것이다. 새로운 LSA 메시지는 자신이 능동노드임을 알려주는 메시지로서 우리는 이를 ALSA(Active LSA)라고 명명한다. 이 개념은 OSPF에서 자신이 멀티캐스트가 가능한 라우터임을 명시한 LSA를 사용하는 것으로부터 가져왔다. OSPF에서는 멀티캐스트를 지원하기 위해 OPTION 필드에 해당 라우터가 멀티캐스트 기능을 가진 라우터임을 명시한 비트(MC 비트)[5]를 정의하고 있다. 그리고 각 라우터에서는 패킷 생성시 자신이 인식하지 못하는 비트들에 대해서는 "0" 값을 설정하며, 또한 전송 받은 패킷 내의 OPTION 필드에 설정된 비트들중 인식하지 못하는 비트들은 무시한다. 따라서, 멀티캐스트 라우터들은 멀티캐스트를 지원하지 않는 라우터들의 패킷 처리에 영향을 주지 않으면서 서로 패킷을 교환할 수 있다.

현재 OSPF[5]는 OPTION 필드내의 다섯 개의 비트의 사용을 지정하고 있으며, 나머지 세개의 비트는 사용하고 있지 않다. 우리는 이중 하나의 비트(상위 2번째 비트)를 능동 라우터임을 명시하도록 제안한다.

[그림 3]은 새로운 OPTION 필드를 보여준다. 다섯 개의 비트(E, MC, N/P, EA, DC)는 기존에 정의된 비트들이며, AN 비트는 새로 정의한 비트이다.

*	AN	DC	EA	N/P	MC	E	*

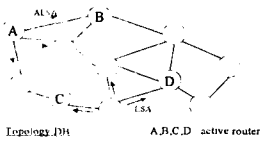
[그림 3] 새로운 option 필드

능동노드를 호스트와 라우터로 구분해서 ALSA의 동작과정을 좀더 자세히 살펴보면 다음과 같다.

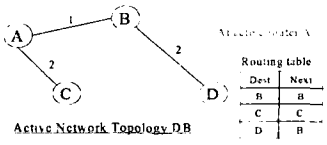
호스트의 경우 가장 가까이에 있는 능동 라우터를 찾는 과정을 수행한다. 이 과정에는 기존에 존재하는 기술들을 적용시킬 수 있다. 예를 들어, 호스트로부터의 확장 링 검색(expanding ring search) 방법이나 능동 라우터가 자신이 기본 라우터(default router)로서 동작하고자 하는 메시지를 일정한 TTL 값을 가지고 광고하는 방법이 있다. 우리는 이 과정에 대한 프로토콜은 아직 결정하지 못했지만 앞서 설명한 방법들도 충분히 만족할만한 결과를 얻을 수 있을 것으로 생각한다.

라우터의 경우 다른 라우터와 LSA를 교환함으로써 네트워크 토폴로지 정보를 구성하는 과정을 수행한다. 능동노드로서 동작하는 라우터는 LSA에 자신이 능동노드임을 명시한 ALSA를 생성하며, 이 ALSA 메시지는 능동노드에서만 특별한 처리 과정을 거친다. 비능동노드(일반 라우터)의 ALSA에 대한 처리 방식은 일반 LSA의 처리과정을 따른다. 즉, ALSA에 명시된 능동노드 태그는 무시한다. 능동노드로 동작하는 라우터는 추가적으로 능동 네트워크 토폴로지 데이터베이스(Active Network Topology DB)를 유지한다. 능동노드로 동작하는 라우터는 비능동 라우터로부터의 LSA와 능동 라우터로부터의 ALSA를 기반으로 능동 라우터와 비능동 라우터가 같이 존재하는 전체 네트워크 토폴로지 정보(Topology DB)를 얻을 수 있고, 이 중에서 능동 라우터에 대한 정보만 추출해서 능동 네트워크 토폴로지 데이터베이스를 구성할 수 있다. 이 정보를 기반으로 Dijkstra 알고리즘을 사용하여 능동 패킷을 전달하기 위한 능동 라우팅 테이블을 구성할 수 있다.

[그림 4]은 능동/비능동 라우터에서의 동작과정과 이 과정을 통한 전체 네트워크 토폴로지 데이터베이스를 보여준다. 그리고 [그림 5]는 [그림 4]으로부터 능동노드들에 대한 정보만 추출해서 얻은 능동 네트워크 토폴로지와 능동 라우팅 테이블을 보여준다.



[그림 4] 능동/비능동 라우터 동작 과정



[그림 5] 능동 네트워크 토폴로지와 능동 라우팅 테이블

4. 능동 패킷 라우팅 방식

능동 패킷의 라우팅은 위의 과정을 통해 얻은 능동 네트워크 토폴로지 데이터베이스와 능동 라우팅 테이블을 통해 이루어진다. 그리고 인터넷의 모든 라우터가 능동노드가 아니기 때문에 능동노드간의 데이터 전송은 터널링(tunneling)을 통해 이루어져야 한다. 만약 일반 IP 라우팅 프로토콜에 의해 송신지에서 목적지까지의 라우팅이 이루어진다면 능동 패킷의 경로는 패킷마다 서로 다른 경로를 통해 전달될 가능성이 커지게 된다. 이 상황은 다음과 같은 능동응용에 있어서 심각한 문제를 유발한다. 능동 네트워크 토폴로지 정보를 기반으로 몇몇 특정 능동노드들 세션에 포함한 능동응용의 경우, 반드시 같은 세션이 속한 능동 패킷은 해당 능동노드를 순서대로 거쳐가야 한다. 혹은 이전 능동 패킷의 결과가 다음 능동 패킷의 처리과정에 영향을 주는 능동응용의 경우도 이전 능동 패킷의 경로를 통해 전달되어야 한다. 또한 같은 세션에 속한 능동 패킷이 서로 다른 능동노드를 거쳐 간다면, 각 능동노드마다 능동 코드 요청 과정을 수반하게 되는 상황이 발생한다.

우리는 능동 패킷의 라우팅 과정을 호스트의 동작 방식과 라우터의 동작 방식으로 구별해서 기술한다.

4.1 능동 호스트

능동 호스트들은 직접적으로 능동 네트워크 토폴로지의 구성에 참여하지 않으며, 가장 가까이 위치한 능동 라우터를 기본 능동 라우터로서 사용하는 정보만 유지하기 때문에 능동 패킷의 라우팅이 쉬운 편이다.

송신지 능동 호스트는 새로운 능동 패킷을 생성하고, 이를 간단히 기본 능동 라우터로 전달하면 된다. 기본 능동 라우터로 능동 패킷을 전달할 때는 앞서 설명한 것과 같이 패킷을 캡슐화(encapsulation)한 후 전달해야 한다.

목적지 능동 호스트는 자신에게 전달된 캡슐화된 능동 패킷으로부터 원래의 능동 패킷을 추출해서 받아들이면 된다.

4.2 능동 라우터

능동 라우터의 경우 직접적으로 능동 네트워크 토폴로지의 구성에 참여하며, 라우터로서의 역할과 송·수신자로서의 역할을 같이 하기 때문에 더 복잡한 처리과정을 수행한다.

우선 라우터로서의 역할은 기본 라우터와 중간 라우터로 구분된다.

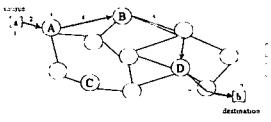
기본 능동 라우터로서 동작하는 경우, 호스트로부터의 캡슐화된 패킷에서 원래의 패킷을 추출하고, 패킷의 경로를 결정한 후 다른 능동 노드로 전달한다. 이때 목적지가 능동 라우터인 경우와 능동 호스트인 경우에 따라 동작이 구분된다. 목적지가 능동 라우터인 경우는 능동 네트워크 토폴로지에 포함되어 있기 때문에(즉, 능동 라우팅 테이블에 목적지 라우터에 대한 항목이 존재한다), 능동 라우팅 테이블에 명시된 다음 능동노드로 패킷을 전달한다. 목적지가 능동 호스트인 경우는 우선 목적지 능동 호스트와 가장 가까이에 위치한 능동 라우터(목적지-인접 능동 라우터)를 찾는 과정이 필요하다. IP 패킷 라우팅의 경우에 있어서 목적지 호스트로 패킷을 전달하기 위해서는 우선 호스트와 같은 서브넷상에 있는 기본 라우터로 패킷을 전달하면 된다. 이후 기본 라우터는 자신의 서브넷에 있는 목적지 호스트로 패킷을 전달함으로서 패킷 전달 과정을 종료한다. 그렇지만 능동 네트워크에서 능동 호스트는 자신의 기본 능동 라우터와 같은 서브넷에 존재하지 않을 수 있기 때문에, IP 라우팅에서 사용한 방법을 그대로 적용할 수 없다. 따라서 능동 호스트로 능동 패킷을 전달하기 위해서는 우선 목적지 능동 호스트와 가장 가까이 위치한 능동 라우터를 찾아서 해당 목적지-인접 능동 라우터에게 능동 패킷을 전달하고, 해당 목적지-인접 능동 라우터가 마지막으로 능동 호스트에게 패킷을 전달하는 방법을 사용해야 한다. 목적지 능동 호스트와 가장 가까이 위치한 능동 라우터는 전체 네트워크 토폴로지 정보를 통해 찾을 수 있다. 그 후 능동 라우팅 테이블을 통해 목적지-인접 능동 라우터에 대한 다음 능동노드로 능동 패킷을 전달한다. 전체 네트워크 토폴로지 정보에서 목적지-인접 능동 라우터를 찾는 과정은 다음과 같다. 우선 목적지 능동 호스트의 IP 주소의 네트워크 주소부분을 찾고, 전체 네트워크 토폴로지 정보를 통해 기본 능동 라우터로부터 해당 네트워크로의 최단 경로를 검색한다. 그 다음 검색된 최단 경로 상에 있는 능동 라우터 중에 목적지 능동 호스트와 가장 가까이 위치한 능동 라우터를 목적지-인접 능동 라우터로 설정한다. 이 경우에도 마찬가지로 능동 패킷은 캡슐화된 후 전달되어야 한다. 목적지가 능동 라우터인 경우는 다음 능동 라우터로 터널링하기 위해 필요한 IP 헤더로 캡슐화하며, 이때 IP 헤더의 송신지 주소는 능동 라우터 자신이 되며, 목적지 주소는 다음 능동 라우터의 주소가 된다. 목적지가 능동 호스트인 경우는 다음 능동 라우터로 터널링하기 위해 필요한 IP 헤더로 캡슐화하기 전에 목적지-인접 능동 라우터로의 터널링하기 위해 필요한 IP 헤더로 캡슐화한다. 이때 IP 헤더의 송신지 주소는 기본 능동 라우터인 라우터 자신의 주소가 되고 목적지 주소는 목적지-인접 능동 라우터의 주소가 된다.

중간 라우터로서 동작하는 경우는 기본 라우터로서 동작하는 경우보다 좀더 간단하다. 능동 패킷은 캡슐화되어 전달되기 때문에, 중간 라우터는 우선 캡슐화된 패킷에서 원래의 패킷을 추출한다. 그 다음 추출된 패킷의 목적지를 검사함으로써 해당 패킷이 라우터 자신에게 전달되는 패킷인지를 구별한다. 만약 다른 능동 라우터로의 패킷인 경우는 능동 라우팅 테이블을 통해 다음 능동 라우터로 패킷을 전달하며, 이때 패킷은 송신지 주소에 라우터 자신의 주소를 포함하며 목적지 주소에 다음 능동 라우터의 주소를 포함하는 IP 헤더를 사용하여 캡슐화된다. 라우터 자신에게 전달되는 패킷인 경우는 두가지 경우가 있다. 원래의 목적지가 라우터 자신인 경우와 자신이 다른 능동 호스트에 대한 인접한 라우터로서 동작하는 경우가 있다. 이는 추출된 패킷의 IP 헤더내의 프로토콜 타입(Protocol Type) 필드를 검사함으로써 알 수 있다. 만약 이 값이 IP-in-IP(즉, 추출된 패킷의 IP 헤더가 터널링을 위해 사용된 경우로 프로토콜 타입 값이 4이다)라면 해당 능동 패킷은 능동 호스트를 위한 것이며, 자신은 목적지-인접 능동 라우터로서 동작하는 경우가 된다. 그렇지 않은 경우는 능동 라우터 자신을 위한 패킷으로서 능동 라우터 자신이 패킷을 처리하며 더 이상 다른 능동 라우터로 전달하지 않는다. 목적지-인접 능동 라우터로서 동작하는 경우는 원래의 목적지 능동 호스트로 곧바로 터널링해서 패킷을 전달한다. 즉, 캡슐화되는 IP 헤더의 송신지 주소는 라우터 자신의 주소가 되며, 목적지 주소는 원래의 목적지 능동 호스트의 주소가 된다.

능동 라우터가 송신자로서의 역할을 하는 경우는 자신이 기본 능동 라우터가 되며, 앞서 기본 능동 라우터로서의 동작 과정에 따라 능동 패킷을 처리한다.

4.3 능동 패킷 라우팅 동작 예

[그림 6]와 [그림 7]은 송신지와 목적지가 모두 능동 호스트인 경우에 있어서, 각각 능동 패킷이 전달되는 과정과 각 과정에 해당하는 패킷의 형태를 보여준다.



[그림 6] 능동 패킷 전달 과정

[그림 7] 패킷 형식

(1) 송신지 능동 호스트에서 패킷을 생성한다. (2) 능동 패킷을 기본 능동 라우터(A)로 전달한다. 이때 패킷은 목적지 주소가 기본 능동 라우터의 주소를 포함한 IP 헤더로 캡슐화되어 전송된다. (3) 기본 능동 라우터는 능동 패킷의 목적지 능동 호스트에 대한 목적지-인접 능동 라우터로서 D를 선택하고, 능동 라우팅 테이블에서 다음 능동 라우터(B)를 검색한다. (4) 다음 능동 라우터로 전송되는 패킷은 우선 목적지-인접 능동 라우터로의 터널링을 위해 캡슐화된 후, 다음 능동 라우터로의 터널링을 위한 캡슐화를 해서 전달한다. [그림 7]의 4번은 두 번의 캡슐화 과정을 거친 후의 패킷 형식을 보여준다. (5) 중간 능동 라우터로 동작하는 B는 능동 패킷을 받고 자신의 능동 라우팅 테이블에서 D에 대한 다음 능동 라우터로서 D를 선택하고 다시 캡슐화한 후 전송한다. (6) 목적지-인접 능동 라우터 D는 캡슐화된 패킷을 추출하는 과정에서 자신이 목적지-인접 능동 라우터임을 인식하고 원래의 능동 패킷을 추출한 후 원래의 목적지인 능동 호스트로 터널링을 통해 패킷을 전송한다. (7) 목적지 능동 호스트는 패킷의 목적지가 자신임을 인식하고 패킷을 처리한 후, 더 이상 패킷을 다른 곳으로 전달하지 않는다.

위에서 기술한 라우팅 과정은 능동 패킷에 추가적인 라우팅 정보를 포함시키지 않은 경우, 즉, 패킷이 각 노드에 존재하는 능동 라우팅 테이블 기반으로 전달되는 경우, 에 해당한다. 이러한 라우팅 방식을 hop-by-hop 라우팅이라 하며, 능동 패킷 라우팅 방식들중 기본 방식이다. 만약 능동 패킷에 source 라우팅으로 명시되었다면 위의 라우팅 과정은 달라지게 된다. 능동 토폴로지 정보를 기반으로 몇몇 특정 능동노드를 이용해 세션을 설정하는 경우, 패킷은 세션에 참여하는 능동 노드를 경유해야 하며 이 경우에는 source 라우팅이 적합하다. Source 라우팅 경로가 패킷에 명시된 경우, 혹은 실행환경에서 해당 패킷 플로우에 대한 추가적인 라우팅 정보를 유지하는 경우는 명시된 경로를 통해 패킷이 전달된다.

5. 향후 연구 방향

본 논문에서는 능동 네트워크 라우팅 프로토콜로서 AOSPF를 제안했다. AOSPF는 동적이며 기존 프로토콜(OSPF)과의 호환성을 갖춘 프로토콜로서, OSPF 기반이기 때문에 OSPF가 지닌 제약사항을 가진다. OSPF는 확장성(scalability)에 제약을 받기 때문에 도메인 내부 라우팅 프로토콜(intra-domain routing protocol)로 사용되며, AOSPF도 OSPF와 마찬가지로 도메인 내부에서 사용되어야 한다. 따라서 BGP와 같은 도메인간 라우팅 프로토콜(inter-domain routing protocol)로서 사용될 수 있는 새로운 프로토콜을 정의할 필요가 있다.

감사의 글

본 연구는 2001년도 한국전자통신연구원 정보보호기술본부 위탁연구과제에 의한 것임.

참고문헌

[1] Jonathan T. Moore and Michael Hick, "A Service Layer Routing Protocol for PLAN," November 1997.
[2] Javier Alvarez and Jessica Kornbium, and Erick Messing, "Simulating Link State Routing in an Active Network." TCOM500 PLAN Project
[3] Pankaj Kakkar, "The Specification of PLAN," July 1999.
[4] Sumi Choi et al., "Configuring Sessions in Programming Networks," Proc. IEEE INFOCOM, 2001
[5] John Moy. "OSPF version 2," RFC 1247, July 1991
[6] Charles Perkins. "IP Encapsulation within IP," RFC 2003, October 1996.
[7] Charles Perkins. "IP Mobility Support," RFC 2002, October 1996.
[8] Eriksson, Hans, "MBone: The Multicast Backbone," Communications of the ACM, August 1994, Vol.37, pp.54-60