액티브 네트워크의 요소 기술

이 한 권* 조 태 경**

목 차

1. 서 론

현재의 인터넷은 웹의 등장과 함께 빠른 속도로 트래픽이 증가하고 있으며 멀티미디어 응용이 추가되면서 기존 네트워크 개발 당시에는 고려하지 않았던 다양한 기능들이 요구되고 있다. 액티브 네트워크는 서로 다른 사용자들의 요구사항을 만족시키기 위한 네트워크 구조를 연구하고자 하는 새로운 시도이다. 오늘날의 네트워크 하부구조는 본질적으로 정적이므로 네트워크에 새로운 기술 및 서비스를 적용시키기 위해서는 많은 시간과 비용이 소요된다. 따라서 기존 네트워크는 다양하고 급격하게 변화하는 응용 서비스를 만족시키지 못하고 있다. 이러한 문제점들을 해결하고자 1994년과 1995년 DARPA(Defense Advanced Research Projects Agency) 연구 회의에서 액티브 네트워크(Active Network)의 개념이 도입되었다.

기존 네트워크의 스위치나 라우터는 단순히 IP 패킷의 헤더만을 보고 경로 배정 테이블에 따라 적절한 인터페이스로 패킷을 포워딩(store-and-forward)하는 식의 단순한 패킷 전달 기능을 수행한다. 반면, 액티브 네트워크는 데이터뿐만 아니라 중간 노드에서 실행되어질 프로그램 코드 또한 패킷 내에 가지고 전송(store-compute-forward)함으로써 적절한 스위치나 라우터에서 프로그램이 실행될 수 있도록 지원한다. 프로그램 코드는 메모리에 적재되고 실행환경(Execution Environment)에서 실행되는 동안 적절한 서비스를 수행한다. 그림 1은 IP 네트워크의 라우터들의 데이터 흐름을 프로세싱하는 과정을 나타내주고 있다.

능동 노드로 프로그램 코드를 주입시키는 방법은 크게 두 가지 구현 방법을 생각할 수 있다[1]. 기존의 패킷 포맷을 그대로 사용하고 프로그램 다운로딩 메커니즘을 별도로 두는 가능한 스위치 접근 방법(programmable switch approach)과 중간노드에서 실행되어질 프로그램을 내장하고 있는 패킷(캡슐)으로 기존 패킷을 대체시키는 캡슐 접근 방법(capsule approach)이 있다. 전자의 경우, 중간노드에서 실행되어질 프로그램의 주입 메커니즘과 패킷 처리 방법을 분리시키는 방법으로 분리 방법(discrete approach)이라고도 부르며, 후자의 경우는 프로그램을 포함하고 있는 캡슐을 기존 일반 패킷과 마찬가지로 중간노드에게 전송하는 방법으로 통합 방법(integrated approach)이라고도 부른다.

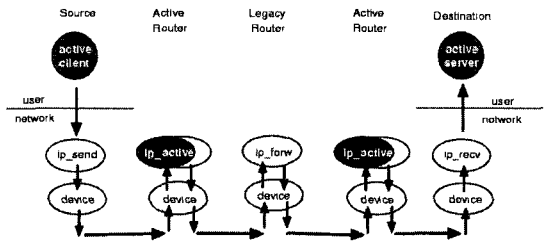


Figure 1. Application-specific processing within the nodes of an Active Network

(그림 1) 액티브 노드들의 어플리케이션 프로세싱

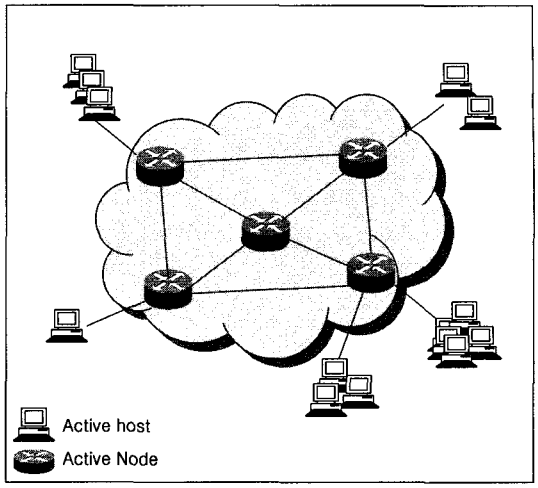
* 상명대학교 대학원 정보통신공학과
** 상명대학교 정보통신공학과 교수

액티브 네트워크 구조는 해결해야 할 많은 이슈들을 포함하고 있다. 능동 노드에서 주입된 프로그램 코드를 실행시켜야 하므로 노드의 성능 문제, 보안 및 안전성 문제 등이 고려되어야 하며, 기존 네트워크와의 호환성뿐만 아니라 프로그램코드가 이동된다는 점을 감안할 때 프로그램 기술 언어에 대한 이슈 또한 고려되어야 할 이슈이다. 현재까지 많은 연구기관과 대학에서 이러한 이슈들을 바탕으로 액티브 네트워크에 대한 연구가 활발히 진행되고 있다. 기존의 액티브 네트워크 연구들은 대부분이 능동노드 구조(노드운영체제, 실행환경)와 능동응용에 초점을 맞추어 진행되고 있으며 패킷 트래픽에 따른 처리 방법이나 능동 토폴로지 구성에 관한 연구는 미흡한 상태이다.

기존 인터넷의 가장 큰 특징 중의 하나가 best-effort방식으로 패킷을 전달한다는 것이다. 모든 IP 패킷은 단일한 특성을 가지는 독립된 패킷으로 고려되어 패킷이 전달되었으며 네트워크 상태에 따라 서로 독립된 IP 패킷을 어떻게 효과적으로 경로를 결정할 것인가에 대한 연구에 초점을 맞추어 진행되어 왔다. 하지만 최근 인터넷의 사용자 증가와 응용의 다양성에 따라, 인터넷의 연구분야는 응용에 따른 패킷의 트래픽 특성을 구별하고 성능을 보장해주기 위한 방법을 모색하는데 관심을 가지게 되었으며 QoS 보장과 고속의 트래픽 처리에 관한 연구분야들이 그것이다. 이러한 인터넷의 발전과정과 마찬가지로 현재까지의 액티브 네트워크 연구 분야도 능동 패킷의 특성을 고려하지 않고 있으며, 능동 패킷의 경로 배정도 정적인 토폴로지 구성에 의존하고 있다.

2. 액티브 네트워크 구조

액티브 네트워크는 패킷 교환 네트워크를 기반으로 하나, 네트워크의 구성요소인 스위치나 라우터가 기존의 단순 패킷 전달 기능이외에 응용이나 사용자가 요구하는 다양한 서비스 제공, 전달상의 효율성 증가를 위해 필요한 기능을 수행하자는데 초점을 맞추고 있다. 즉, 각 노드는 패킷 혹은 프로그램을 저장할 수 있는 저장 장치, 프로세싱 방식, 새로운 서비스 추가나 삭제에 관련된 기술이 제공되는 컴퓨팅 환경을 갖추어 패킷 전달뿐만 아니라 필요한 컴퓨팅이나 프로세싱을 함으로써, 네트워크가 수행해야 할 기능을 능동적이고 효율적으로 처리하자는 것이다[1,2].



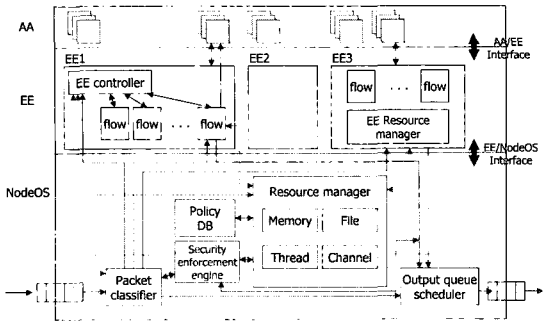
(그림 2) 액티브 네트워크 구성도

그림 2는 액티브 네트워크 구성도를 나타낸다. 이 구조는 일반적인 네트워크 구조와 동일하나 수행하는 기능면에서는 많은 차이점을 보인다. 먼저, 사용자나 응용이 위치하는 액티브 호스트들은 네트워크를 통하여 서로 연결된다. 각 스위치나 라우터는 액티브 노드라 칭하고, 상호 연결된 액티브 노드들은 하나의 액티브 네트워크를 구성하게 된다. 노드 간에는 액티브 네트워킹을 인식하는 통신 프로토콜과 포맷이 정의되어야 하며, 이에 따라 동작하게 된다.

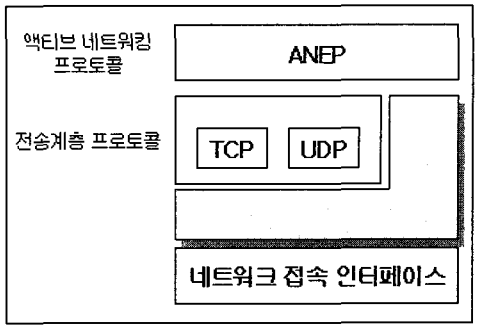
액티브 네트워킹을 실현하기 위해 필요한 세부 기술은 액티브 노드 구조를 비롯하여 통신 프로토콜 및 패킷 포맷, 프로그램을 다른 노드로 전송하는 기법, 프로그램 실행 환경, 각 노드가 프로그램을 로딩하여 수행 하는 방식, 공통의 프로그래밍 모델과 언어 등 여러 분야로 구성된다.

2.1 액티브 노드 구조

능동노드의 구조는 크게 노드운영체제(Node Operating System), 실행환경(Execution Environment), 능동응용(Active Application)으로 구분된다.



(그림 3) 액티브 노드 구조



(그림 4) 액티브 프로토콜 스택

능동응용은 능동 패킷을 처리하기 위한 실행코드이다. 사용되는 횟수가 작거나 실행코드의 크기가 작은 경우는 능동 패킷의 데이터에 포함되어 전달되고, 자주 사용되거나 실행코드의 크기가 큰 경우는 능동코드를 전달하는 패킷을 통해 능동 노드에 기 적재되는 특성을 가진다. 예를 들어, 실행코드를 위해 필요한 라이브러리의 경우 기 적재되어야 한다.

실행환경은 능동 패킷을 받고 실행코드를 통해 패킷을 처리한 후 결과를 다른 응용이나 노드운영체제로 전달하는 기능을 수행한다. 실행환경은 일종의 가상머신(virtual machine)과 같다.

노드운영체제는 다른 능동노드로부터의 능동 패킷을 받아서 이를 적절한 실행환경으로 분배하고, 실행환경에서 해당 능동패킷을 처리하기 위해 필요한 노드 자원을 관리(할당/제어/반환)하며, 실행환경으로부터의 결과를 다른 노드로 전달하는 기능을 수행한다. 그림 3은 능동 노드의 구조이다.

2.2 통신 프로토콜 및 패킷 포맷

모든 노드가 액티브 네트워킹을 지원하는 공통의 프로토콜과 패킷 포맷을 인식하여야 하며, 특정 응용이 액티브 네트워크상에서 동작하기 위해서는 응용 프로토콜과 패킷 포캣이 정의되어야 한다. 액티브 네트워크에서 사용 할 수 있는 프로토콜 스택은 여러 옵션이 존재하나, 일반적으로 현재 많이 구축되어 있는 IP(IPv4, IPv6)를 이용하는 것을 기본으로 하고 있다. 그림 4에서 나타나는 바와 같이 IP를 기반으로 하여 TCP 혹은 UDP를 통하거나 혹은 직접 액티브 네트워크 프로토콜을 사용할 수 있다.

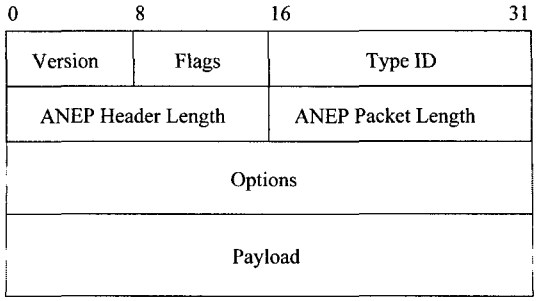
액티브 네트워킹의 패킷 포맷으로서는 ANEP(Active Network Encapsulation Protocol)과 SAPF(Simple Active Packet Format)이 있다.

2.2.1 ANEP

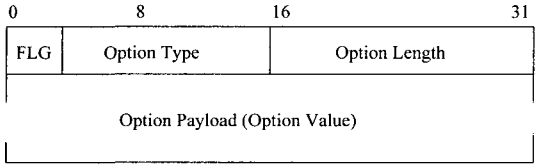
액티브 네트워크의 능동노드에서는 다양한 언어로 작성된 프로그램을 로딩하고 실행시킬 수 있으며, 이러한 프로그램들은 능동 패킷의 페이로드에 실려 전송된다. ANEP는 그림 5와 같은 헤더 구조를 갖으며, 프로그램은 ANEP에 의해 명시된 실행환경 안에서 실행되어 처리되어진다.

ANEP 구조는 다음과 같은 필드들을 갖는다.

- Version : 현재 사용 중인 ANEP 헤더의 종류
- Flags : 능동 노드가 Type ID를 인식하지 못할 때 패킷을 어떻게 처리할 것인지에 대해서 명시. 필드 값이 0이면 디폴트 라우팅 메커니즘을 사용, 1이면 패킷을 폐기.
- Type ID : 패킷이 수행될 실행환경 값.



(그림 5) ANEP 헤더의 형식



(그림 6) 옵션 필드 형식

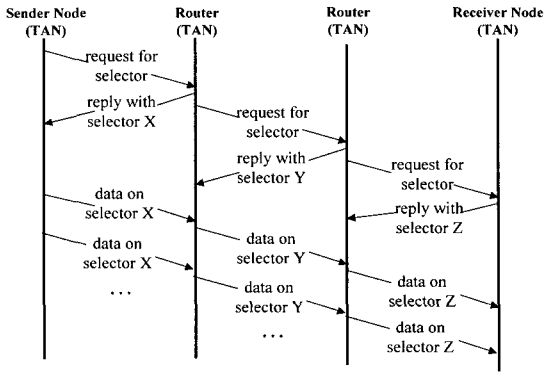
- ANEP Header Length : 32비트 워드 단위로 ANEP 헤더의 길이.
- ANEP Packet Length : 패킷 전체 길이를 옥텟 단위로 표수.
- Option : 소스 식별자, 목적지 식별자, 무결성 체크섬, N/N 인증 표시.

옵션 필드의 형식은 그림 6과 같으며 현재 제공되는 옵션 타입으로는 소스 식별자, 목적지 식별자, 무결성 체크섬, N/N인증 등이 있다. 2비트 플래그 필드의 첫 번째 비트는 옵션 타입이 어떤 특정 Type ID에서만 의미를 가지는지 명시하고 두 번째 비트는 해당 옵션 타입을 인식하지 못할 때 취해야 할 동작을 명시한다. 0일 경우에는 옵션을 무시하고 헤더를 처리하며, 1일 경우에는 패킷을 폐기한다.

2.2.2 SAPF

ANEP 헤더가 가변의 옵션 필드를 가지고 있으며 특정 실행환경을 위해 명시된 패킷 헤더들도 가변길이 필드들을 가지고 있으므로 ANEP 헤더를 이용하여 실행환경을 분류하는 방법은 느리고 복잡하다는 단점을 지닌다. SAPF는 헤더 정보를 줄임으로써 패킷 포워딩을 위한 기본 기능과 실행환경 분류를 위한 시간을 최소화하고자 제안되었다. 실제로 참고문헌 [9]에 의하면 ANEP 대신 SAPF를 사용하여 능동 패킷을 처리하였을 경우 30%정도 처리시간이 빨라진다고 한다.

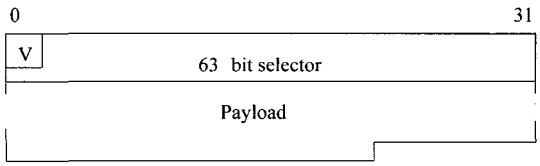
SAPF는 첫 번째 데이터가 전송되기 전에 이웃 TAN (Tags for Active Networking)노드들 사이에 셀렉터(selector)를 설정하게 되는데 그림 7은 그 과정을 보여준다. 송신자는 데이터 전송 전에 상태 레코드(state record)를 생성하고 셀렉터를 위해 다운 스트림 TAN 라우터에게 실렉터 요청을 한다. 셀렉터 요청 패킷을 받은 TAN 라우터는 상태 레코드를 생성하고 라우터 안에서 유일한



(그림 7) TAN 노드의 셀렉터 설정

셀렉터를 선택하여 요청에 응답한다. 송신자는 셀렉터에 대한 정보를 수신하게 되면, 상태를 저장하고 셀렉터를 이용하여 데이터를 보내기 시작한다.

셀렉터를 이용한 SAPF 헤더는 그림 8과 같으며 헤더의 길이가 고정되어 있으므로 신속하게 패킷 분류 작업을 수행할 수 있다.



(그림 8) SAPF 헤더

2.3 프로그램 로딩 및 프로세싱 방식

각 액티브 노드는 응용의 요청에 따라 프로그램을 저장, 수행 및 관리할 수 있는데, 이때 프로그램을 로딩하고 처리하는 방법에 따라 분리방식, 통합방식 그리고 이 두 가지를 이용하는 혼합방식이 있다.[2]

2.3.1 분리방식 (Discrete approach)

프로그램이 데이터와 분리되어 이미 노드에 상주되어 있는 경우이다. 이 방식대로 호스트는 수행되기를 원하는 프로그램 식별자와 필요한 데이터를 저장한 패킷을 전송하고, 노드는 프로그램 식별자에 해당하는 프로그램을 로딩하여 전달받은 데이터를 프로세싱 하며, 결과물로 얻어진 변형된 데이터를 다음 연결된 노

드에게 전송한다. 이 방식은 노드에 이미 이식되어 있는 프로그램에 대해서만 적용 가능하므로 새로운 프로그램 이식 및 수행을 동적으로 할 수 없다는 단점을 갖고 있다. 이 방식을 적용시킨 예로써 Active IP와 SwitchWare에 대한 연구가 있다.

2.3.2 통합/캡슐 방식 (integrated/capsule approach)

프로그램이 미리 저장되지 않는 경우이며, 노드가 프로그램과 데이터를 실은 패킷을 전달한다. 각 노드는 전송받은 프로그램과 데이터를 자신의 실행 환경에서 수행한 다음, 얻어진 데이터를 연결된 다음 노드에게 전달한다. 이 방식은 전달해야 하는 프로그램 량이 큰 경우나 네트워크 상에서 트래픽 문제, 패킷 분실시 재전송 문제 등에 인하여 효율성이 저하될 수 있다. MIT에서 수행중인 ANTS 프로젝트와 팬실베니아 대학의 PLANet이 이 방식을 적용하였다.

2.3.3 혼합방식

패킷 전달 과정에서 발생하는 지연이나 분실에 따른 비효율성을 제거하기 위하여 위의 두 가지 방식을 이용할 수 있다. 즉, 공통적으로 사용하는 프로그램은 미리 노드에 이식하고, 각 노드가 사용 할 특정 프로그램은 패킷에 실어 전송하는 방법이다. 호스트가 보내는 데이터는 노드에 이미 저장되어 있던지 데이터와 함께 보내지는 프로그램에 의해 처리되어진다.

2.4 공통의 프로그래밍 모델

네트워크 프로그램들은 액티브 노드에 이식되어 있는 경우를 제외하고는 연결된 네트워크를 따라 이동하면서 다른 플랫폼 상에 로딩되어 작업을 처리하게 된다. 따라서, 각 프로그램은 어떠한 플랫폼 환경에서도 동일한 방법으로 동작할 수 있도록 코딩되어야 하며, 프로그램이 실린 패킷을 각 노드가 인식할 수 있는 서비스 프리미티브가 정의되어야 하고, 또한 실행되는 노드의 자원을 할당하고 관리할 수 있는 기능이 제공되어야 한다.

- 프로그램 인코딩 방식 : 프로그램을 다른 노드로 전송하고 실행(mobility), 자원에 대한 액세스 관리(safety), 네트워크 효율성(efficiency) 저하방지 기능이 제공되어야 한다.
- 공통의 프리미티브 : 전달받은 패킷의 헤더 처리 기능 (헤더, 패킷내용, 길이정보 변형 등), 실행될 노드의 환경을 액세스할 수 있는 기능(노드주소, 시간과 날짜 점검, 연결 정보 관리 등), 그리고 패킷 전달 기능(전송, 복제, 삭제 등)이 요구된다.
- 자원 정보 교환 방식 : 통신 대역폭, 프로세싱 성능, 저장장치, 라우팅 테이블, 노드의 MIB 정보 등 노드가 관리하는 자원 정보를 서로 교환 및 인식할 수 있는 공통의 모델이 제공되어야 한다.

3. 연구 동향

액티브 네트워크 구조에 대한 연구가 몇몇 대학과 연구기관으로 시작된 이후, 현재는 많은 기관들이 협력하여 액티브 네트워크 구조 세부 기술에 대한 연구를 꾸준히 진행시키고 있다.

액티브 네트워크에 관련된 연구 프로젝트 중에 몇몇 주요 프로젝트의 간략한 내용은 다음과 같다.

- ANTS (Active Node Transfer System) : 액티브 네트워크에 접속할 수 있는 노드의 실행 환경과 네트워크 프로그래밍 모델을 제공해 주는 툴킷으로 MIT에서 개발하였다. Free BSD와 Linux상에서 동작하는 버전이 무상으로 배포되고 있다.
- NetScript : 네트워크 노드에서 프로그램을 생성, 전달, 실행하기 위해 필요한 환경과 기능을 제공하는 시스템이며, 콜럼비아 대학에서 개발하였다.
- PLANet : 펜실베니아 대학에서 정의한 액티브 네트워크 구조로서, 모든 패킷이 프로그램을 실어 보내는 분리 방식을 이용하여 개발되었다. 고유의 언어인 PLAN (Programming Language for Active Networks)을 이용하여 패킷으로 전달되어지는 프로그램을 기술하고, Caml을 이용하여 라우터의 기능을 확장할 수 있도록 한다. 노드간에 통신 패킷 포맷을 정의하였고, IP기반에서 실행된다. 또한, PLAN을 이용하여 프로그램 가능한 스위치

(Switchware)를 개발하였다.

- Smart Packets : 액티브 네트워크 기술을 망관리 시스템에 적용한 프로젝트로 BBN에서 개발하였다. Sprocket과 Spanner라는 고유의 프로그래밍 언어를 사용하였으며, ANEP을 이용하고 망 관리를 위한 패킷 포맷을 추가로 정의하였다.
- ANEP 프로토타입 : 3기관에서 다른 프로젝트 하에서 구현되었으며, ANTS와 마찬가지로 현재 무상으로 배포되고 있다.
  - BBN Technologies : Smart Packets 프로젝트를 수행하면서 구현된 것으로, IP위에 바로 ANEP을 구축한다. 개발 언어는 C와 자신들이 개발한 전용 언어를 이용하여 기존의 커널을 수정하여 개발하였다.
  - ISI(Information Science Institute) : Active Signalling 프로젝트에서 Java 1.1개발 언어를 이용하여 개발한 것으로, 프로토콜로써는 기존의 IP와 UDP를 사용한다.
  - 펜실베니아 대학 : Switchware 프로젝트를 수행하면서 ISI와 마찬가지로 개발 언어는 Java 1.1, 프로토콜 스택으로는 IP와 UDP를 이용하여 개발하였다.

4. 적용 사례

앞 절에서 살펴본 액티브 네트워크의 기본 개념과 목적을 어느 분야에 어떻게 적용하느냐가 매우 중요한 사항이다. 본 절에서는 실제로 액티브 네트워킹 기술을 적용한 예를 살펴보기로 한다.

4.1 Active Reliable Multicast

일반적으로 인터넷상에서의 Reliable Multicast는 패킷 전송에 있어 신뢰성을 보장하지 못하기 때문에 각 호스트는 전달받지 못한 패킷에 대해 재전송 요구를 하고, 서버는 해당하는 패킷을 전달하게 된다. 따라서, 기존의 Reliable Multicast의 가장 큰 문제는 각 호스트마다 요청하는 NACK의 폭주라 할 수 있다. 이 문제를 해결하기 위해 서버 중심의 재전송이 아닌 각 액티브 노드가 수행할 수 있는 로컬 재전송 기법을 제안하고 있으며, 구체적인 기능 및 동작 기법은 다음과 같다.

- 패킷 저장 : 각 액티브 노드는 전달받은 패킷을 다음 목적지로 전달하기 전에 자신의 캐쉬 메모리에 임의의 시간(TTL: Time To Live)동안 저장해 둔다. 저장한 패킷은 향후 호스트의 재전송 요청에 응답하기 위함이며, TTL이 지나면 자동 삭제한다.
- NACK제어 : 각 노드는 중복되는 재전송 요청을 점검하여 하나의 재전송 패킷만을 전달한다.
- 패킷 재전송 : 전달받은 패킷이 재전송을 요청하는 경우 자신의 캐쉬 메모리에 해당하는 패킷이 존재하는지를 점검하여, 갖고 있는 경우에는 재전송 요청 패킷을 다음 노드에게 전달한다.

4.2 세션 멀티캐스트 설계

여러 사용자간에 공동작업을 지원하는 응용들의 중요한 요구사항은 그룹 작업에 참여하는 모든 사용자간에 신속하고 정확한 정보전달을 제공하는 멀티캐스트 기능이다. 이전의 응용 계층에서 사용되는 멀티캐스팅 기능은 거의 다 공동작업 서비스를 제공하는 서버 중심의 전달 방식을 채택하고 있었다. 이 방식은 멀티캐스트되는 모든 정보가 일단 서버로 전송되고, 서버가 모든 참여자에게 재전송하는 방식으로, 정보 전송 시 지연이 많고 특히 전달하는 정보량이 많은 경우 또는 실시간적 전달이 필요한 경우에는 효율성이 급격히 저하됨을 나타낸다.

세션 멀티캐스트는 그룹 작업에 참가하고 있는 사용자 정보(세션 정보)를 저장, 갱신 및 관리하며, 멀티캐스트 수행시 전달되는 정보를 저장하고 세션 정보를 참조하여 전달시킴으로써, 전체적인 정보전송의 효율성 증가를 목적으로 하고 있다. 이를 위하여, 각 액티브 노드는 멀티캐스트를 위하여 세션 정보를 관리한다. 세션 정보는 세션의 식별자, 세션을 생성한 멤버의 식별자, 세션에 가담하고 있는 참가자 명단(노드 식별자)으로 구성되며, 다른 제어 정보와 더불어 제어 채널을 이용하여 전달된다. 또한, 참가자 정보가 신속하게 전달되도록 멀티캐스트 트리 구성(AMT: Active

Multicast Tree) 알고리즘을 이용한다. AMT는 각 노드의 멀티캐스트 용량(Multicast capacity)과 링크 거리/비용(link cost/distance)을 고려하여 새로운 참가자를 멀티캐스트 트리에 포함시키기에 가장 짧고 저렴한 링크를 구성하여 준다.

5. 결 론

액티브 네트워크는 기존 통신 개념이나 구조가 갖고 있는 비효율성을 해결하자는 취지에서 새로운 통신 패러다임을 제안하고 있다. 네트워크를 구성하는 각 노드의 기능이 단순 패킷 전달에서 프로그램을 직접 수행하여 전달하고, 서비스 요구에 따라 다양한 컴퓨팅을 실행하는 등 네트워크가 능동적이고 유연한 구조나 기능을 제공하자는 것이다. 또한, 통신망에서 새로운 서비스를 보다 신속하고 경제적으로 도입하고 망자원을 보다 적절하게 활용할 수 있도록 해준다. 액티브 네트워크를 이용하여 보안에 활용할 경우 사이버 공격자의 실제 위치를 역추적 할 수 있고, 네트워크에 대한 접근을 원천적으로 차단 및 고립화 시킬 수도 있다.

현재 대부분의 통신망은 액티브 네트워크 기술을 적용하여 전송속도와 성능의 향상을 꾀하고 있다. 따라서 이러한 통신 인프라를 활용할 수 있는 다양한 멀티미디어 서비스의 개발이 지속적으로 이루어져야 할 것이다.

참 고 문 헌

[1] David L. Tennehouse, Jonathan M. Smith, W. David Sincoskie, Gary J. Minden, "A Survey of Active Network Research", IEEE Communications Magazine, January 1997.

[2] David L. Tennehouse and David J. Wetherall, "Towards and Active Network Architecture", Proc. Multimedia Comp. And Networkin, MMCN 96, San Jose, CA, SPIE, January 1996.

[3] Biswas J, et al, "The IEEE P1520 Standards Initiative for Programmable Network Interfaces", IEEE Communications, Special Issue on Programmable Networks, Vol. 36, No 10, October 1998.

[4] L. K. Calvert, "Architectural Framework for Active Networks, Version 1", Active Network Working Group, Draft, July 1999.

[5] D. Alexander, et. al., "Active Network Encapsulation Protocol(ANEP)," http://www.cis.upenn.edu/~switchware/ ANEP/docs/ANEP.txt, 1997.

[6] B. Schwartz, et. al, "Smart packets for Active Networks," BBN Technology, Jan. 1998.

[7] G. Alex, et. al., "A Flexible IP Active Networks Architecture," IWAN 2000 Conference, 2000.

[8] D. Decasper and C. Tschudin, "Simple Active Packet Format (SAPF) ," 1998

[9] D. Alexander, et. al., "The SwitchWare Active Network Architecture," IEEE Network, May/June 1998.

[11] M. Hicks, et.al, "PLANet: An Active Internetwork," IEEE INFOCOM, 1999.

[12] David J. Wetherall, John V. Guttag and David L. Tennenhouse, "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols", IEEE OPENARCH'98, San Francisco, CA, April 1998.

◐ 저 자 소 개 ◑

이 한 권

1998년 3월 ~현재 : 상명대학교 정보통신학과
관심분야 : 초고속정보통신망, BcN, 홈네트워크

조 태 경

1984년 2월 : 한양대학교 전자통신공학과 (공학사)
1986년 2월 : 한양대학교 전자통신공학과 (공학석사)
2001년 9월 : 한양대학교 전자통신공학과 (공학박사)
2003년 9월~현재 : 상명대학교 정보통신공학과 교수
관심분야 : 초고속정보통신망, e-Learning