이동통신환경에서의 소규모 그룹통신을 위한 XMIP 프로토콜의 구현

박 인 수[†] · 박 용 진[††]

요 약

본 논문에서는 다세션 소규모 멀티캐스트 통신을 위해 제안된 Explicit Multicast 전송방식과 IETF Mobile IP를 효과적으로 결합한 XMIP 프로토콜을 구현하고 검증한다. Xcast 패킷 헤더 내에 목적지 주소들을 소스 노드가 명시하여 Xcast 네트워크로 전송하면 각 Xcast 라우터는 멀티캐스트 트리의 지원없이도 유니캐스트 라우팅 정보만을 기초로 목적지를 향해 경로설정과 전송을 수행한다. XMIP 프로토콜은 이러한 Explicit Multicast 프로토콜의 특징을 상속받아 상태유지의 필요없이 유니캐스트 라우팅 테이블을 기반으로 하므로 전송방식이 명확하고 단순하다. 본 연구에서는 Xcast 네트워크와 IETF Mobile IP의 연동을 함께 고려해서 Mobile IP의 이동 에이젼트인 HA/FA를 수정 보완하여 HA+/FA+로 각각 확장한다. HA+로 전송된 Xcast 패킷은 Mobile IP 바인딩 테이블을 참조하여 각 FA+로 향하는 X-in-X 터널 인터페이스를 통해 전송된다. 이 메커니즘으로 IETF Mobile IP 멀티캐스트 트래픽 집중 문제를 효과적으로 해결할 수 있다. 마지막으로 무선랜 기반 실험망을 구축하고, Xcast 응용으로서 다중 사용자를 위한 인스턴트 메신져를 개발하고 실험하므로써 최종 개발한 XMIP/Xcast 프로토콜의 실효성을 검증한다.

An Implementation of Explicit Multicast with Mobile IP for Small Group Communications in Mobile Networks

PARK, IN-SOO[†] · PARK, YONG-JIN[††]

ABSTRACT

In this paper, we implement and verify XMIP integrating IETF Mobile IP and the Explicit Multicast mechanism for a great number of small group multicast communications. If a source node sends Xcast packets explicitly inserting destination nodes into the headers, each Xcast router decides routes and forwards the packets toward each destination node based on unicast routing table without the support of multicast trees. XMIP is a straightforward and simple multicast mechanism just based on a unicast routing table without maintaining multicast states because of the inheritance from the Explicit Multicast mechanism. This research modifies and extends the functionality of IETF Mobile IP's mobility agents, such as HA/FA to HA+/FA+ respectively, considering interworking with Xcast networks. Xcast packets captured by HA+ are forwarded into X-in-X tunnel interfaces for each FA+ referred to the binding table of HA+. This X-in-X tunneling mechanism can effectively solve the traffic concentration problem of IETF Mobile IP multicast services. Finally WLAN-based testbed is built and a multi-user Instant messenger system is developed as a Xcast application for finally verify the feasibility of the implemented XMIP/Xcast protocols.

키워드 : 멀티캐스트(Multicast), 이동성(Mobility), 소규모 그룹 통신(Small Group Communications)

1. 서 론

1969년, 단 4대의 컴퓨터를 연결한 미국 국방성의 ARPA NET(Advanced Research Projects Agency Network)을 시초로 하여 1991년에 NSFNET(National Science Foundation Network)의 구축 이래로 인터넷은 비약적인 발전을 거듭하여 현재 인터넷 사용자들은 HDTV급의 방송형 멀티미디어 스트리밍과 같은 고수준의 서비스들을 당연하게 요구하고 있다. 이러한 다중 그룹통신 서비스를 지원하기 위해 Steve Deering은 멀티포인트 환경에서의 효율적인 대역폭 사용을 위한 IP 멀티캐스트 라우팅 기법을 제안하였다 [1]. 이후 멀티캐스트에 대한 연구 및 개발은 지속적으로 이루어져 IETF에서는 멀티캐스트 라우팅, 트랜스포트 및 응용계층에 대한 표준화 및 이를 상용망에 도입하려는 활발한 시도가 이루어지고 있다. 그러나, 멀티캐스트 네트워크와 서비스를 확장하는 단계에서 멀티캐스트 주소 할당 문제, 멀티캐스트 패킷을 주소를 기반으로 전송하기 위해 라우팅 정보를 동적

† 정 회 원 : KT 책임연구원
†† 정 회 원 : 한양대학교 공과대학 전자통신컴퓨터공학부 교수
논문접수 : 2004년 9월 10일, 심사완료 : 2005년 3월 10일

으로 유지하는 문제, 그리고 대량의 그룹세션이 형성되는 대형 멀티포인트 환경을 지향하는 경우의 확장성 문제 및 세션 제어 문제 등으로 상업적 도입이 어려운 실정이다 [2]. 1999년 이후에는 이러한 N:N 환경의 ASM(Any Source Multicast)을 해결하기 위해 보다 세분화한 1:N 환경의 SSM (Source Specific Multicast)과 소규모 그룹을 지원하는 Xcast (Explicit Multicast)와 같은 방식들이 활발히 제안되었다 [3][4].

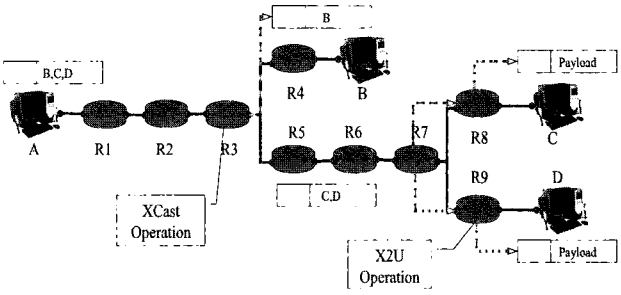
기존의 멀티캐스트 기술은 불특정 다수의 수신자들에게 일방적으로 방송형 (Broadcast) 서비스를 제공하는 유형의 방송형 멀티캐스트와 그룹 멤버의 수가 적고 세션 수가 많은 네로우캐스트형의 소규모 그룹 멀티캐스트 기술이 있다. Xcast 방식은 네로우캐스트형에 속하며, 소규모 그룹을 지원하는데 초점을 맞추어 설계된 프로토콜이다. 즉, Xcast는 기존 멀티캐스트의 장점인 사용 대역폭을 최소화하는 동시에 멀티캐스트 라우팅 정보가 아닌 유니캐스트 IP 라우팅 정보를 참조하기 때문에 멀티캐스트 주소 할당제어 및 멀티캐스트 트리 구조 유지를 위한 지속적이며 오버헤드를 발생시키는 별도의 멀티캐스트 라우팅 프로토콜을 요구하지 않는다. 따라서, 세션 관리 측면에서 뛰어난 확장성을 가지고, 소스에서 모든 클라이언트의 주소를 Xcast 패킷 헤더에 모두 포함시키므로 인증 및 보안 관리가 더욱 손쉽고 안정적이 된다 [4-6]. Xcast의 이와 같은 특징들은 멀티캐스트 기술의 상용화 측면에서 매우 중요한 특징임에 틀림없다. 따라서, Xcast에 이동성 지원 프로토콜인 Mobile IP를 효과적으로 접목함으로써 얻어지는 효과는 매우 고무적이다 [7]. IETF Mobile IP는 Home Agent에서의 멀티캐스트의 전송지원을 명시하고 있으나 현실성이 결여된 방안으로 평가되고 있어 Xcast와 Mobile IP의 접목은 이동 에이젼트들간의 네트워크 사용 효율성의 극대화 및 모바일 그룹통신관리의 용이성을 포함한 많은 긍정적 효과를 기대할 수 있다. 따라서, 본 논문에서는 Mobile IP와 Xcast를 유기적으로 결합시킨 XMIP (Xcast over Mobile IP)를 All-IP기반 이동망에 적용하기 위한 구체적인 방안과 그 유효성을 검증하는데 연구의 초점을 둔다.

논문의 구성은 2장에서 Xcast와 Mobile IP에 대한 기술적 기반을 살펴보고, 3장에서는 XMIP 프로토콜 설계를 제시한다. 4장에서는 3장의 설계를 기초한 XMIP 구현방안을 제시한다. 5장에서는 실험을 위한 응용 프로그램으로 Xcast 기반 인스턴트 메신저 시스템을 구현하고, 6장에서 실험망 구성 및 개발한 XMIP/Xcast 프로토콜에 대한 실효성을 검증한다.

2. 관련 연구

2.1. Xcast 프로토콜 개요

Xcast는 기존의 멀티캐스트 기술과는 달리 IP 헤더와 트랜스포트 계층 헤더 사이에 Xcast 헤더를 삽입하여 헤더 내에 모든 수신자를 명시하여 각 수신자들에게 패킷을 전달하는 방식의 프로토콜이다 [9][11]. 다시 말해서, Xcast에서는 Layer3인 IP 계층 위에 Xcast Layer를 추가하여, 멀티캐스트 트리 정보에 의존하지 않고 유니캐스트 라우팅 정보를 기반으로 라우팅과 패킷전송을 수행한다. 라우팅 도중에는 최소한의 패킷 복제가 일어나게 되며 최종 노드에서는 각각의 수신자들에게 일반적인 유니캐스트 패킷의 형태로 전달되게 되어 Xcast 프로토콜을 이해하지 못하는 수신자도 어떠한 시스템적인 수정없이 Xcast망을 이용한 통신을 할 수 있도록 설계되어 있다. 이를 X2U (Xcast to Unicast) 기술이라 한다. (그림 1)에 Xcast 전송방식을 이해할 수 있는 간단한 시나리오를 도시하였다.

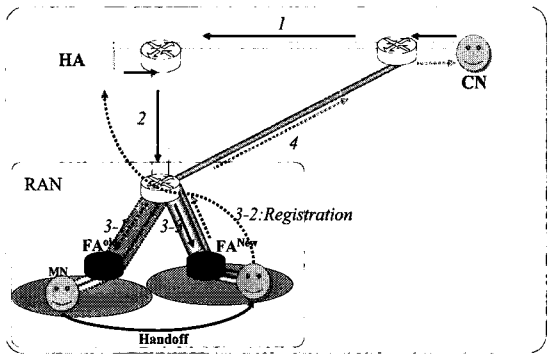


(그림 1) Xcast 서비스 시나리오

따라서, 헤더크기 제한과 그룹주소를 사용하지 않고 IP 라우팅 정보를 기초로 전송하는 특징으로 Xcast 전송방식은 적은 수의 그룹 맴버와 동시에 많은 수의 세션을 가지는 멀티캐스트 응용에 효과적이다.

2.2. IETF Mobile IP

Mobile IP의 개념은 이동 단말이 원래 등록된 Home Network의 특정 라우터 및 이동단말이 방문한 네트워크의 특정 라우터에 IP 이동성을 처리해 줄 수 있는 기능을 추가하여 단말이 현재 IP 주소를 유지한 상태로 통신이 이루어지도록 하는 것이다 [8]. 이러한 이동성 처리 라우터 중 Home Network의 라우터를 HA (Home Agent)라고 하며 이동 단말의 원래 IP 주소와 현재 방문한 네트워크의 주소를 저장하고 있다가 이 이동 단말에 대한 패킷이 도착하게 되면 이를 이동 단말 대신 받은 후 이동 단말의 현재 위치로 터널링 기법을 이용하여 전송하게 된다. 이를 (그림 2)에 도시한다.



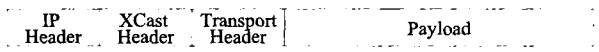
(그림 2) Mobile IP의 기본 동작 방식

이동 단말은 자신이 홈 네트워크가 아닌 방문 네트워크로 이동했음을 찾아내기 위해 FA(Foreign Agent)로부터 방송되는 Agent Advertisement 메시지를 이용한다. 이동 단말은 방문 네트워크로 들어온 것을 판단하게 되면 방문한 네트워크의 임시 주소를 할당받게 되는데 방문 망의 이동성 처리 가능 라우터인 FA의 주소를 할당받거나 단말 자신이 DHCP 혹은 PPP 프로토콜 등에 따라 임시주소를 할당받게 된다. 이 주소는 HA가 인터셉트한 패킷을 터널링하여 전달하는데 사용된다. 이 임시 주소를 CoA(Care of Address)라고 하며 이 주소는 HA로부터 만들어지는 터널의 종착점이 된다. HA에서 터널 종착점 (FA 혹은 이동단말)까지의 터널사이에서 패킷들은 IP-in-IP 메커니즘에 의해 캡슐화되어 전송된다.

3. XMIP 프로토콜 설계

3.1. Xcast 프로토콜

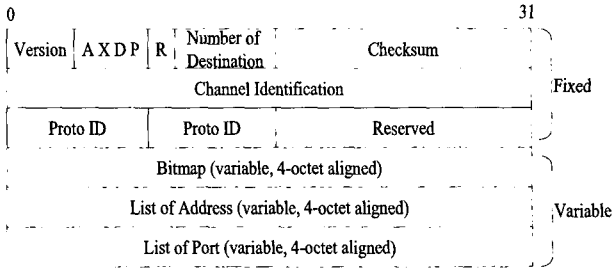
Xcast 패킷의 IP헤더의 목적지 주소는 Xcast 용으로 특별하게 할당된 주소를 사용한다. 이 주소는 멀티캐스트 주소 영역에 존재하며 Xcast 라우팅을 위해서 단 하나의 주소만 사용하게 된다. 따라서 모든 Xcast 패킷의 IP헤더의 목적지 주소는 동일하게 된다. 이 주소를 AXR (All-Xcast-Routers) 주소라고 하며 고정된 값이다. 기존의 멀티캐스트 라우팅 프로토콜과는 달리 Xcast 패킷의 라우팅은 IP 헤더의 목적지 주소와는 상관없이 Xcast 헤더 내부의 목적지 주소 리스트에 의해 이루어진다.



(그림 3) Xcast 패킷 구조

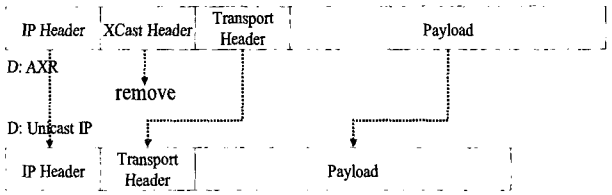
Xcast 헤더는 IP 헤더와 트랜스포터 헤더 사이에 위치하게 된다. 이때 IP 헤더 부분의 source 주소에는 패킷을 보내는 호스트의 주소가 들어 가며 목적지 주소에는 고정된 AXR 주소 값이 들어간다. IP 헤더의 프로토콜 번호는 Xcast를 의미하는 IPPROTO_XCAST 값이 되는데 현재 IANA(Internet Assigned Number Authority)에서 IPPROTO_XCAST 값을 정하지 않고 있기 때문에 구현에서는 현재 정의되지 않은 값을 임시로 사용한다[4]. Xcast 패킷은 다음과 같은 프로토콜 인캡슐레이션 구조를 가진다. Xcast 프로토콜 헤더는 IP 헤더와 트랜스포트 헤더의 사이에 위치하는 3.5 layer 프로토콜로 설계되어 있다. 그러나, 구현 상에서 IPv4/IPv6의 IP Option 헤더로서 설계될 수도 있다 [10].

Xcast 헤더의 구조를 (그림 3)과 (그림 4)에 보인다. Xcast 헤더는 크게 두 부분으로 나눌 수 있다. 즉, 고정된 길이를 가진 부분과 가변 하는 길이를 가진 부분으로 나눌 수 있는데 Xcast 헤더의 앞부분 12 바이트는 모든 Xcast 헤더에 공통적으로 포함되는 헤더이다. 그 이후에 나오는 비트맵과 목적지 주소 리스트, 포트 리스트는 길이가 변할 수 있다.



(그림 4) Xcast 헤더 구조

Xcast 프로토콜은 목적지 주소 리스트 내부의 주소들에 대한 next hop을 알아보는 과정에서 별도의 라우팅 테이블을 참조하지 않고 유니캐스트 라우팅 테이블을 사용한다. Xcast 라우터는 목적지 주소의 리스트 내부의 주소에 한 개의 주소만 남거나 자신에게 수신자가 직접 연결되어 있는 경우, 즉 수신자에 대해 경계 라우터 (Edge Router) 역할을 하는 경우에 Xcast 패킷을 수신자의 주소를 목적지 주소로 가지는 유니캐스트 패킷으로 변환시킨다. (그림 5)에 X2U (Xcast to Unicast) 과정을 도시한다. 따라서 최종 노드의 수신자는 Xcast 패킷이 아닌 보통의 유니캐스트 패킷을 받게 된다. 비록 Xcast 라우터가 경계 라우터가 아닌 경우에도 Xcast 패킷의 목적지 주소 리스트에 하나의 주소만 남아 있는 경우에 Xcast 패킷을 유니캐스트 패킷으로 변환하게 되면 그 이후의 라우팅은 Xcast 라우팅이 아니라 보통의 유니캐스트 라우팅이 된다. 이 경우 Xcast 라우팅에 의한 통신 오버헤드를 줄일 수 있다.



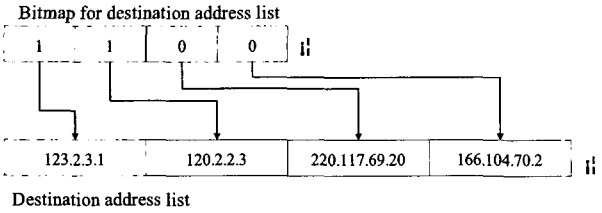
(그림 5) X2U 처리 동작

또한, 이후에 Xcast 라우터가 존재하지 않는 경우에도 미리 X2U 동작을 수행하게 되면 최종 노드까지 패킷 전달이 가능하다. Xcast 라우터는 자신의 다음 노드가 Xcast 라우터인지 여부를 확인한 후, 만일 아니라면 비록 주소 리스트에 남은 주소가 한 개가 아닐지라도 X2U 동작을 수행하여 목적지까지 패킷을 전달 할 수 있는데 이러한 경우의 X2U 동작을 특별히 premature X2U라고 한다.

Xcast 라우터가 다음 노드의 라우터가 Xcast 라우터인지의 여부를 알기 위해서는 Xcast 패킷을 생성한 후 주소 리스트에 라우터 자신의 주소를 포함시켜 다음 노드로 보낸다. 만일 이때 다음 노드로부터 응답이 온다면 다음 노드의 라우터는 Xcast 라우팅 기능을 수행할 수 있는 것이고 그렇지 않다면 Xcast 라우터가 아닌 것을 알 수 있다.

Xcast 라우팅에서 next hop의 수만큼 패킷을 복제하고 각각의 주소 리스트를 수정한다. 만일 이러한 수정이 없을

경우 중복된 패킷이 발생하게 된다. 리스트의 내용을 실제로 수정하는 것은 많은 오버헤드를 발생시킬 수 있다. 따라서, 이러한 부담을 줄이기 위해서 Xcast는 비트맵을 도입하여 적은 연산으로 목적지 주소리스트를 수정하는 효과를 낸다. (그림 6)에 개념적 도표를 제시한다.



(그림 6) 비트맵과 목적지 주소 리스트와의 관계

목적지 주소 리스트에 들어가는 주소들은 각각에 해당하는 비트맵이 존재한다. 목적지 주소 각각에 대한 비트맵은 한 비트이며 0 또는 1의 값이 들어간다. 이 비트맵의 값이 1이면 유효한 주소이고 0인 경우 무효한 주소로서, 주소 리스트에 존재하지 않는 것과 동일한 효과를 가진다. 목적지 주소 리스트의 주소들을 수정할 때 실제로 주소를 삭제하는 것이 아니라 간단하게 한 비트의 비트맵의 값을 0으로 바꾸어 주는 것으로 주소의 삭제 동작을 수행할 수 있다.

선택 사항으로서 목적지 주소의 포트 정보를 포함시킬 수 있다. 역시 목적지 주소 리스트와 같이 리스트의 형태로 Xcast 헤더에 포함된다. 포트 리스트가 포함된 경우에는 최종적으로 유니캐스트 패킷으로 변환될 때 목적지 포트 번호를 목적지 별로 다르게 설정하는 것이 가능하다. 기존의 멀티캐스트 패킷이 단일 포트만을 가지는 것과 차이가 있다.

3.2. XMIP (Xcast over Mobile IP) 프로토콜

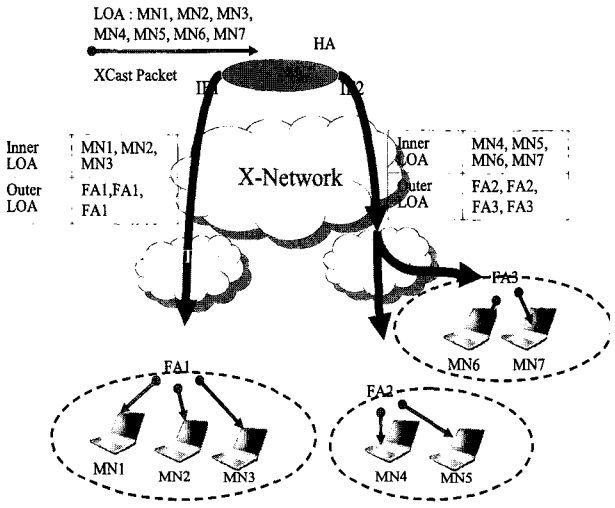
XMIP란 Xcast와 Mobile IP가 공존하는 모바일 멀티캐스트 네트워크에서 Xcast 트래픽을 표준 Mobile IP에 기초하여 효과적으로 모바일 노드들로 전송하기 위한 Xcast 프로토콜과 Mobile IP 프로토콜이 결합된 프로토콜이다 [7]. 따라서, 효과적으로 Xcast 프로토콜을 적용하기 위해 표준 Mobile IP 프로토콜 구성 요소 (HA, FA, MN) 등에 새로운 기능 모듈이 발전적 방향으로 추가될 것이다. 결과적으로 Mobile IP는 Xcast 통신을 이용하여 네트워크 자원의 최소화와 Mobile IP에 의한 트래픽 폭주 방지 효과 등의 긍정적 효과를 얻게 된다.

Xcast를 지원하는 HA는 자신에게 등록한 MN의 Xcast 패킷을 인터셉트한다. HA는 Xcast 패킷을 가로채어 자신에게 등록한 MN로 터널링하여 전송한다. 일반적인 Xcast 라우터들은 Xcast 패킷 내의 LoA필드 값에 의해 라우팅을 수행하지만 Xcast-capable HA의 경우는 다음 메커니즘에 따라 X-in-X 터널링 기능을 수행한다.

1) HA는 Xcast 패킷내의 주소들의 CoA 바인딩 엔트리를 검색한다.
2) CoA에 대한 next forwarding hop을 결정한다.
3) HA는 next-hop에 따라 CoA set으로 분류한다.
4) ①~③동작을 수행한 후 수신된 Xcast를 Next Hop 인터페이스에 따라서 다중 복제하여 인캡슐레이션한다.

- 이 때, 각 터널 헤더의 LoA에는 분류된 CoA가 들어간다.
- 내부 패킷 헤더의 LoA에는 원래의 수신된 Xcast 패킷에 있던 home address들이 들어가게 된다. 각 주소들의 터널 헤더의 CoA에 바인딩되어 있다.

즉, HA는 mobility binding information에 기초하여 X-in-X 인캡슐레이션을 수행한다. MN이 자신의 홈 주소를 사용해서 Xcast session에 join하고 있다면 MN이 home network에서 떠나있는 상태에서도 Xcast 패킷은 HA까지 도달 할 수 있다. (그림 7)에 간단한 예를 들어, XMIP의 동작원리를 설명한다.



(그림 7) XMIP 터널링 시나리오

(그림 7)에서 HA가 관리하는 MN은 총 7개로 모두 FA CoA를 사용하며 3개의 방문 네트워크들로 분산되어 있고, HA에서 각 방문 네트워크들이 모두 Xcast 라우팅을 지원하는 네트워크라고 가정한다.

HA는 각 MN들이 등록한 CoA 바인딩 정보를 참조하여 CoA에 해당하는 next hop 별로 분류하며, 각 MN의 바인딩 정보는 <표 1>에서 보인다. (그림 7)에서 HA로부터 외부망으로의 왼쪽 인터페이스를 IF1, 오른쪽을 IF2 이라 표기한다.

CoA의 next hop에 따라 분류하면, FA1은 IF1에, FA2, FA3는 IF2에 속함을 알 수 있다. 두개의 next hop이 존재하기 때문에 수신된 하나의 패킷을 두번 복제한다. 그리고 각각의 주소들을 next hop에 해당하는 subset 주소들로 변경해 준다. 이렇게 복제되고 수정된 패킷들은 다시 CoA 주소들을 기초로한 Xcast 헤더로 덧입히게 된다. 최종적으로 총 2개의 X-in-X 패킷이 생성된다. 결과적으로 이중으로 형성된 패킷 헤더내 LOA 구조는 <표 2>와 <표 3>에서 기술하는 바와 같이 정해진다.

〈표 1〉 X-in-X 패킷 전송을 위한 라우팅 정보

Home Addr	CoA	Next hop
MN1	FA1	IF1
MN2	FA1	IF1
MN3	FA1	IF1
MN4	FA2	IF2
MN5	FA2	IF2
MN6	FA3	IF2
MN7	FA3	IF2

각각의 패킷은 해당하는 next hop으로 포워딩 된다. 즉, outer header에 정의된 바에 따라 Xcast 라우팅된다. 각 패킷의 최종 FA에 도달하게 되면, 각 FA는 X-in-X 터널 구조를 해체하여(디캡슐레이션), 내부의 본래 Xcast 패킷을 복원하고 이를 자신의 서브넷에서 이동 중인 모바일 노드들에 전달한다.

〈표 2〉 Packet 1

Inner header Dest.	MN1, MN2, MN3
Outer header Dest.	FA1, FA1, FA1

〈표 3〉 Packet 2

Inner header Dest.	MN4, MN5, MN6, MN7
Outer header Dest.	FA2, FA2, FA3, FA3

3.2.1. HA+ 설계

HA는 적어도 한 개의 mobility binding이 존재하고 있다면 그 패킷이 자신에게 등록한 MN의 홈 주소를 가지고 있는가 아닌가를 판단하기 위해서 일단 모든 Xcast 패킷을 받고, 홈 주소와 바인딩되어 있는 CoA에 기초하여 라우팅을 수행한다. HA는 CoA들의 적당한 next hop들을 결정하고 그 다음에는 next hop들에 기초하여 MN들의 홈 주소를 분류한다.

만일 HA가 IETF Mobile IP에서 정의한 simultaneous binding을 지원한다면 모든 바인딩된 CoA를 포함해야 한다. 결과적으로 동일한 홈 주소를 LoA 필드에 가진 복수개의 Xcast 패킷이 생성될 수도 있다. 만일 next hop에 단 한 개의 주소만 있는 경우라면 HA는 그 주소로 X2U를 수행할 수도 있다. HA는 자신에게 등록하지 않은 주소들에 대해서는 일반적인 Xcast 라우팅을 수행해야 한다. 관리적인 측면에서 HA는 들어오는 모든 Xcast 패킷에 대하여 X2U를 수행하도록 설정할 수 있다. 이것은 최종 수신 노드들이 광범위하게 (super-sparsely) 흩어져 있거나 중간 노드들을 형성하는 라우터들이 Xcast를 지원하지 못하는 경우에 유용하다.

HA는 들어오는 Xcast 패킷들을 Xcast 패킷으로 X-in-X 인캡슐레이션해야 한다. 이때 터널 패킷의 주소영역에는 내부 패킷의 LoA들에 해당하는 CoA들이 들어가게 된다. 터널 헤더의 LoA에는 FA CoA나, co-located CoA가 들어갈 수 있다. 여러 개의 MN들이 동일한 FA CoA를 공유하는 경우도 있을 수 있기 때문에 내부 패킷에는 여러 개의 주소가 담겨 있지만 터널 Xcast 패킷에는 한 개의 주소만 있는 경우도 생길 수 있다. 이러한 경우에 HA는 전송하기 전에 터널 헤더에 대해 X2U(Xcast-in-Unicast)를 수행할 수도 있다.

3.2.2. FA+ 설계

자신의 CoA로 들어온 encapsulated Xcast datagram에 대하여 FA는 그 내부 Xcast 패킷 헤더에 자신의 visitor list에 등록한 주소가 있는지 확인해야 한다. 만일 해당되지 않는 주소가 있다면 해당 주소들은 "0"으로 셋팅되고 이후의 내부 Xcast 패킷 처리에 있어서는 무시된다. 해당되는 주소들에 대해서는 IETF Mobile IP의 FA가 정의된 방식대로 MN으로 패킷을 전송하게 된다. FA가 라우팅 경로 중간에 위치할 수도 있기 때문에 터널 헤더에서 자신의 CoA가 아닌 주소들에 대해서는 보통의 Xcast 라우팅을 수행한다.

3.2.3. MN와 Xcast 프로토콜

3.2.3.1. 패킷의 수신

MN가 홈 네트워크로부터 떠나있는 경우에도 MN은 Xcast 패킷을 받는다. 만일 FA CoA를 사용하는 경우라면 패킷은 FA를 통해서 포워딩 될 것이다. co-located CoA를 사용하는 경우에는 터널헤더에는 co-located CoA, 내부 LoA엔 MN의 홈 주소가 들어있는 패킷을 받게 된다. 자신의 CoA나 홈 주소가 아닌 주소들에 대해서는 아무런 통고 없이 0로 overwrite되고 이후의 처리에서는 무시되어야 한다.

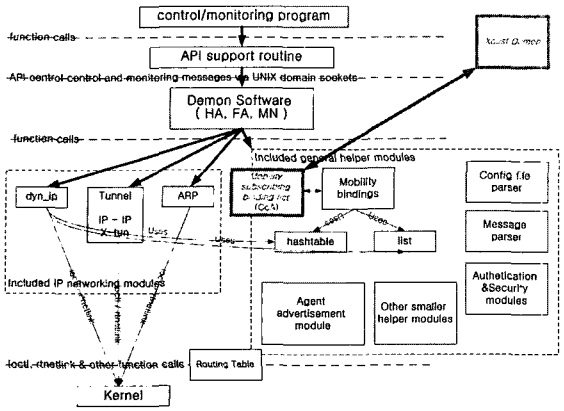
3.2.3.2. 패킷의 송신

외부 네트워크에서 MN은 Mobile IP 모듈에서 결정된 디폴트 라우터를 사용한다. 그러나 만일 Xcast 모듈이 다른 특정한 게이트웨이 라우터를 지정한 경우라면 그것이 더 높은 우선순위를 가지고 디폴트 게이트웨이 라우터로 사용된다. 만일 Mobile IP 등록 시에 reverse tunneling도 등록되었다면 MN은 Xcast 패킷을 X-in-U 형태로 HA에게 인캡슐레이션하여 전송할 수 있다.

4. XMIP 프로토콜 구현

4.1. 전체 구성도

XMIP는 Xcast와 Mobile IP를 적용한 기술이다. Xcast의 인코딩 부분은 이동 에이전트에서 담당하게 된다. 즉, Xcast 패킷을 MN에 보낼 때, HA+와 FA+에서 재구성하여 보내게 된다. 이때 참조하는 것은 MN에 도달할 Xcast 패킷의 경계 라우터 또는 branching node이다. 즉, 이동 에이전트에 바인딩되어 있는 MN의 CoA가 되는 것이다.



(그림 8) XMIP 전체 구성도

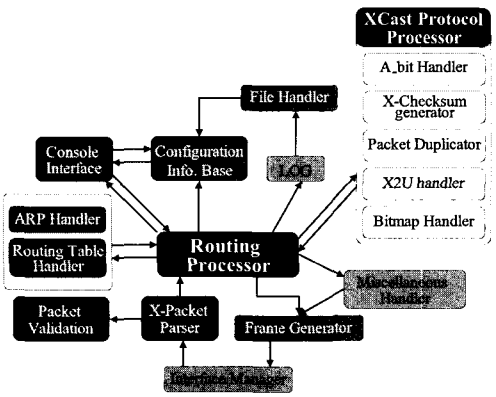
XMIP 를 구현하기 위한 컴포넌트 로서 크게 두 가지 컴포넌트가 존재한다. 바로 HA+ 와 FA+인데, HA+는 MN의 Mobile IP 바인딩 정보를 기반으로 수신된 Xcast 패킷의 터널링을 수행한다. FA+ 에서는 터널링 되어 들어온 Xcast 패킷에 대해 터널 해제를 수행하고 자신이 관리하고 있는 MN들에게 터널 헤더가 제거된 원래 Xcast 패킷을 전달하는 역할을 한다. XMIP에서의 Xcast 패킷의 터널링은 크게 X-in-X 방식과 X-in-U 두 가지 방식으로 동작할 수 있다. co-located CoA를 쓰는 MN들은 터널 패킷을 해체할 FA+가 존재하지 않기 때문에 자기 자신이 그 역할까지 수행해야 한다. XMIP의 동작은 기본적으로 Xcast 망을 가정하고 있다. 따라서, XMIP 동작을 위해서는 중간 코어망 부분에 Xcast router를 먼저 설치 해야 한다.

4.2. Xcast 라우터 구현

4.2.1 Xcast 라우터의 구조

Xcast 라우터의 구조를 (그림 9)에 도시한다.

1) Console interface: 콘솔 모니터링 툴과의 인터페이스를 제공한다. 내부적인 구현은 named pipe 기반으로 되어 있다.
2) File handler: Xcast 컨피규레이션 파일을 읽어들이고 파싱하는 역할을 한다. Xrouter의 컨피규레이션 파일은 Xrouter. cfg이다.



(그림 9) Xcast 라우터 구조

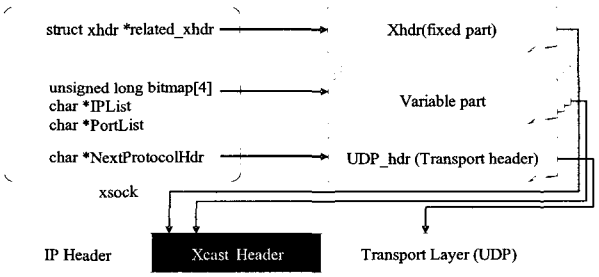
3) Configuration information base: Xrouter의 컨피규레이션 사항들을 저장하는 모듈이다.
4) ARP handler: 이 구현체에서는 XoE를 구현하고 있다. 따라서 next hop으로 패킷을 보낼 때 멀티캐스트 MAC 주소를 사용하지 않는다. Next hop의 MAC주소를 알기 위해 ARP 캐시를 참조한다. 이 부분을 담당하는 모듈이다.
5) Routing table handler: 유니캐스트 라우팅 테이블을 사용한다. 본 연구에서는 라우팅 정보를 얻기 위해 proc 파일에 접근하여 파싱하는 방식을 취하고 있다.
6) Routing processor: 들어온 패킷에 대해 라우팅을 수행해 준다. next hop별로 LoA를 분리 하고 개수만큼 복제하며 적절한 인터페이스로 내보내주는 역할을 한다.
7) Xprotocol processor: Xcast의 기본적인 동작을 맡는 부분. 라우팅을 할 경우 비트맵의 조작이나 각종 파라미터를 얻는 부분등을 이 모듈에 의존하고 있다.
8) Miscellaneous handler: 라우팅 한 패킷의 TTL값 처리와 체크섬의 재계산 등의 동작을 한다.
9) Packet validation: 수신된 Xpacket이 유효한지, 비트맵은 유효한지를 판단한다.
10) Xpacket parser: 수신된 Xpacket에서 각종 비트 플래그 값을 얻어낸다.
11) Frame generator: 이 구현은 XoE를 적용하고 있다. 이 부분이 XoE를 구현한 전송엔진으로 ARP 캐시와 밀접한 연관성을 가진다.

4.2.2. Xcast 라우터 처리 과정

Xcast 라우터 개발시 주요 상수에 대한 정의는 다음과 같다.

```
#define IPPROTO_XCAST 200
#define AXRADDR     "224.2.2.1"
#define MAXDEST   127
```

또한 주요 구조체인 xsock의 구조를 (그림 10)에 도시한다.



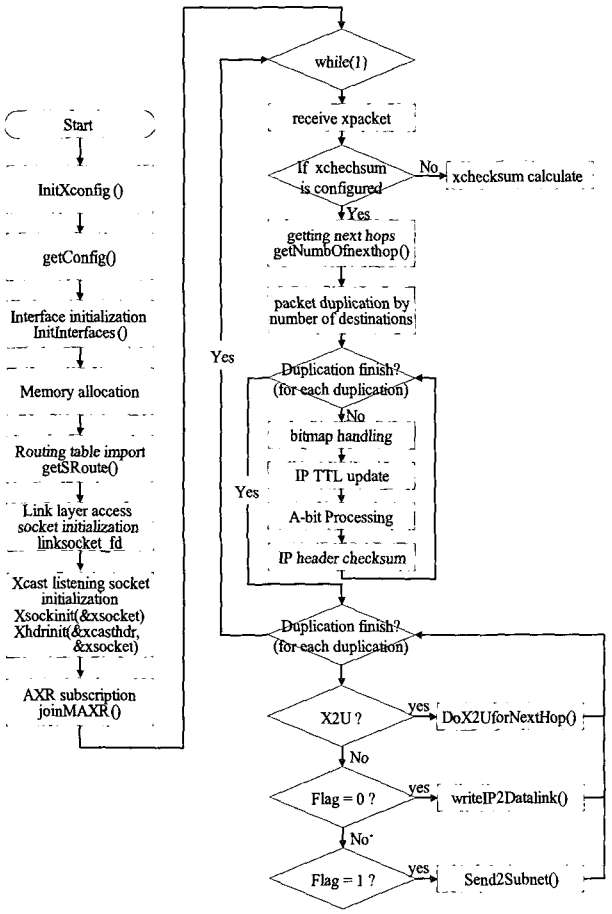
(그림 10) 구조체와 패킷간 매핑 관계

(그림 11)에 Xcast 라우터의 주요 기능 흐름도를 도시한다. 라우터가 처음으로 시동되면 일단 라우팅에 관련된 여러 가지 정보를 얻어오는 동작이 이루어진다. 이때, InitXconfig()와

getConfig()를 통해서 컨피규레이션 파일의 내용을 읽어 온다. 설정 사항을 읽어 들인 후에는 IniInterfaces() 루틴을 통해 현재 라우터의 인터페이스 정보를 얻어 오며 getSRoute()를 사용해서 현재의 라우팅 테이블 정보를 메모리에 복제해 넣는다. 정보를 얻어오는 작업을 마친 이후에는 라우팅에 필요한 raw socket과 구조체들을 초기화 하는 작업을 진행한다. 이 때, Xsockinit() 함수와 Xhdrinit() 함수가 사용된다. Xcast 패킷을 받기 위해서 AXR 주소에 join하는 동작이 필요한데, 이때 자신이 가진 모든 인터페이스에 대해 AXR join 작업을 수행한다.

초기화 작업을 모두 마친 후에는 메인 루프에 진입하게 되는데, 이 과정에서 만일 Xcast 패킷을 수신하게 되면 주소 리스트를 읽어 들여 next hop별로 주소를 분류하고 그 수만큼 패킷 복제를 하게 된다. 이후에는 복제된 패킷 각각에 대해 비트맵을 수정하게 되고 IP 헤더의 TTL 값을 변경한다. 만일 flag들이 설정되어 있다면 그에 대한 처리를 해주게 된다.

Xcast 라우터는 코어 라우터 혹은 경계 라우터로 동작할 수 있는데 만일 core router로서 동작 할 경우에는 next hop의 MAC 주소를 사용해서 프레임을 만든 후 전달되며, 경계 라우터로 동작하는 경우에는 자신의 서브넷에 존재하는 수신자들의 MAC 주소를 얻어내어 프레임을 만든다.



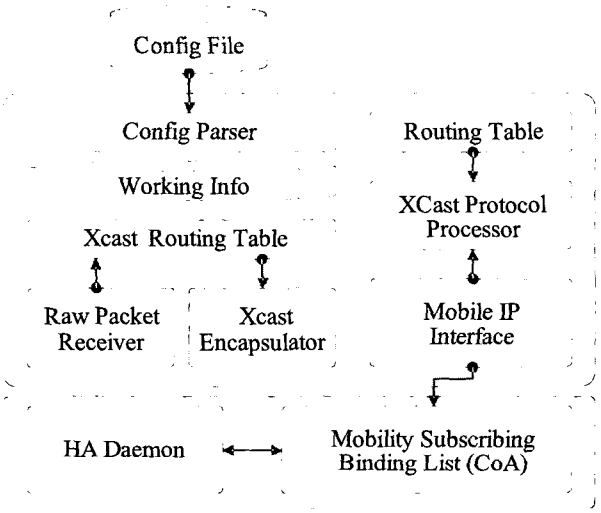
(그림 11) Xcast 라우터 주요 흐름도

4.3. HA+ 구현

HA+는 XMIP 구현을 위해 기존 HA에 덧붙어 동작하는 별도의 데몬으로 구현 되어있다. 직접 Mobile IP 코드에 내장되어 있지는 않지만 Mobile IP 코드와 상호작용하면서 얻어낸 정보를 기반으로 Xpacket의 터널링을 수행한다. HA+의 의미는 기존 HA의 기능을 확장한다는 의미와 기존 HA의 대체가 아닌 덧붙여 동작하는 프로그램이라는 의미를 모두 포함한다. Mobile IP 터널링과 상관없는 트래픽에 대해서는 일반적인 라우터의 역할을 해 주어야 하기 때문에 기본적으로 Xrouter가 가지는 모든 기능성을 가지고 있으며 거기에 터널링을 위한 추가적인 기능을 가지고 있다. 따라서 기존의 구현된 Xrouter 구현체를 기반으로 작업했으며 필요한 기능을 덧붙이고 확장하는 형식으로 구현 하였다.

4.3.1. HA+의 구조

HA+의 기본 구조를 (그림 12)에 도시한다.



(그림 12) HA+ 구조

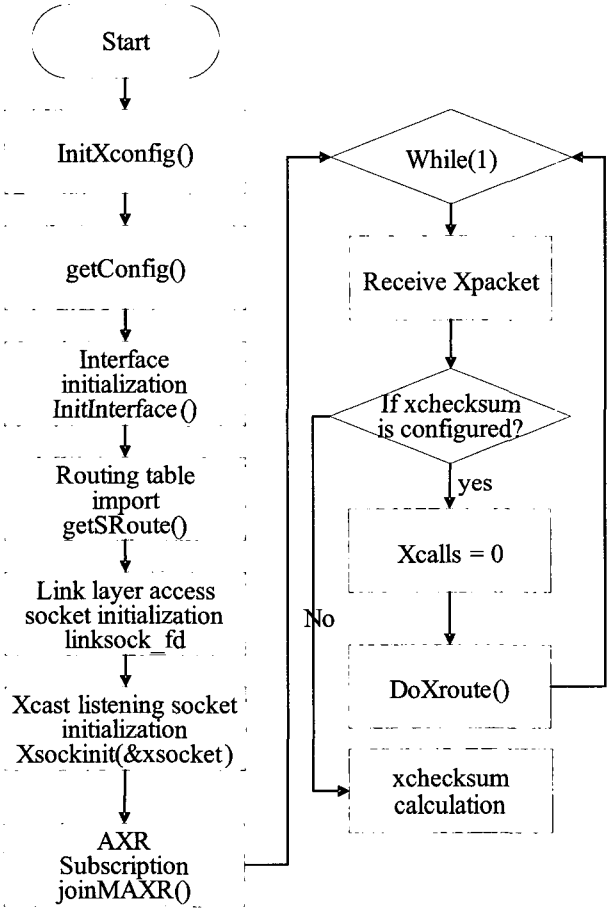
각 모듈들의 기능은 다음과 같다.

1) config parser: Configuration file로부터 HA+ 의 설정 사항들을 읽어 내부의 저장소로 저장하는 역할을 한다.
2) Working information: 현재 HA+의 동작 상태, 처리한 패킷수 등의 작업에서 발생 하는 로그 처리 기능 등을 가진다.
3) Routing table: 유니캐스트 라우팅 테이블을 그대로 사용한다. 본 연구에서는 라우팅 정보를 얻기 위해 proc 파일에 접근하여 파싱하는 방식을 취하고 있다.
4) Xcast protocol processor: Xcast의 기본적 동작을 담당한 모듈이며, 라우팅을 할 경우 비트맵 관리와 각종 파라미터를 얻는 부분 등을 이 모듈에 의존하고 있다.
5) Mobile IP Interface: Mobile IP와 연동하기 위한 부분이며, Mobile IP에서 제공하는 라이브러리를 사용하여 접근한다.

6) Xcast routing module: 들어온 패킷에 대해 라우팅을 수행한다.
7) Raw socket receiver: 기본적으로 Xcast 패킷을 받기 위해 raw socket을 사용한다.
8) Xcast encapsulator: Xcast 터널링을 수행하는 모듈이며, 전송 정책에 따라서 X-in-X 혹은 X-in-U 인캡슐레이션을 수행한다.
9) HA Daemon: Mobile IP의 HA 데몬이다.
10) Mobility Subscribing Binding List: CoA의 바인딩 리스트를 HA 데몬에서 Script File을 이용하여 참조한다.

4.3.2. HA+ 처리 과정

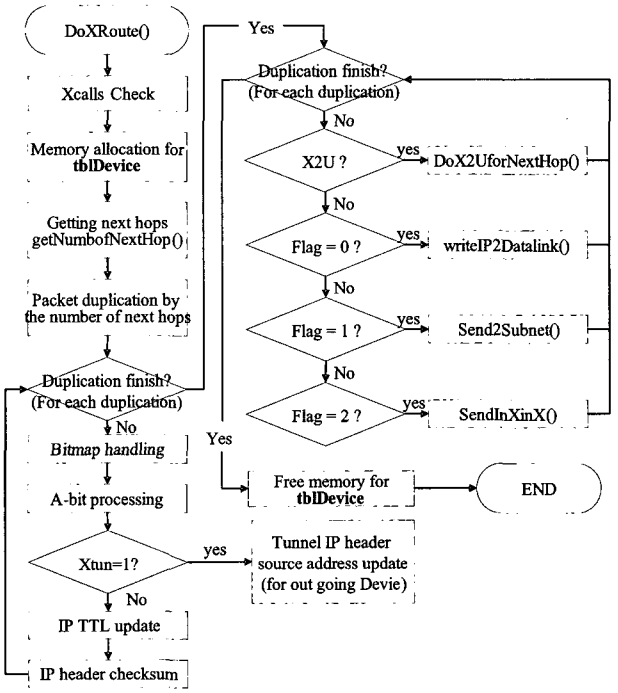
HA+의 동작 구현을 위한 프로시져를 (그림 13)과 (그림 14)에 도시한다.



(그림 13) HA+ 주요 흐름도

1) Next hop의 수를 확인한다. 외부에 있는 MN들은 별도의 next hop으로 간주한다.
2) Next hop의 수만큼 패킷을 복제한다.
3) 복제한 패킷에 대해 next hop으로 보내준다. 네트워크가 이더넷의 경우엔 XoE (Xcast over Ethernet) 메카니즘이 적용된다.
4) 외부의 MN들로만 구성된 패킷 복제본에 대해 전송 정책에 따라 X-in-X 터널링을 수행한다.
5) Encapsulated 패킷의 터널 헤더의 LoA부분을 목적지로 지정된 CoA 리스트로 채워준다.
6) 최종 패킷을 다시 Xcast 라우팅 모듈로 전달하면 터널 헤더에 대해 Xcast 라우팅을 수행한다.

CoA를 찾는 경우, 만일 XTIB가 설정이 되어 정적인 (Static) 엔트리가 존재한다면 우선적으로 참조된다. 만일 static TIB설정이 되어 있지 않는다면, dynamic TIB 참조로 넘어가서 Mobile IP와 연계하여 CoA를 얻어 낸다. 또한, HA+는 Mobile IP와 상관없는 주소에 대해서는 일반적인 Xcast 라우팅을 수행한다. 따라서 기본적인 Xcast 라우터의 기능을 가지고 있다.



(그림 14) DoXroute() 흐름도

4.3.3. Mobile IP Interworking

위에 설명되었던 바와 같이 Mobile IP에서의 바인딩된 MN의 CoA를 얻어내야 한다. 그래서, Xcast 패킷에 인코딩하여 보내게 된다. 바인딩 리스트는 HA 데몬에 있는 HA의 binding status tool을 참조하여 Xcast에 이용할 수 있는 또 다른 데몬을 구성하였다. 처음에 HA 데몬에 초기화 하고, 바인딩된 MN의 리스트와 주소를 얻는다.

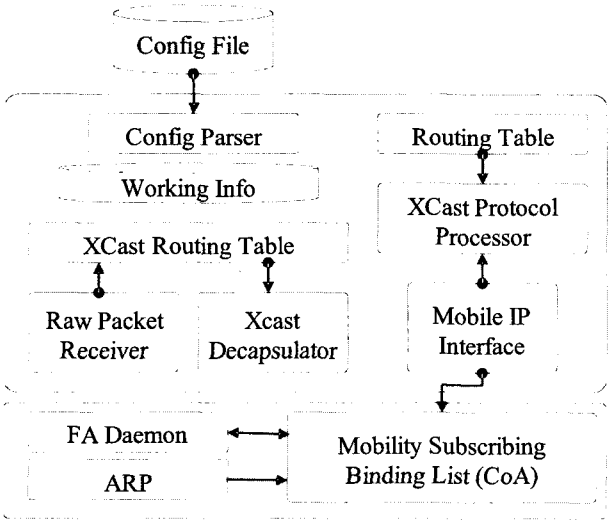
4.4. FA+ 구현

FA+는 XMIP 구현을 위해 기존 FA에 덧붙어 동작하는 별도의 데몬으로 구현 되어있다. Mobile IP 코드에 내장되어 있지는 않지만 Mobile IP 코드와 상호작용하면서 얻어낸 정보를 기반으로 Xpacket의 디캡슐레이션을 수행한다. FA+

의 핵심 동작은 X-in-X 형태로 들어온 터널링 패킷을 해제하는 것이며, 이 역할을 해주기 위해 기존의 Xrouter의 동작의 일부를 수정하였다. 구체적으로 자신의 서브넷으로 패킷을 뿌려주는 함수인 (그림 11)의 Send2Subnet() 부분을 수정하였다. 나머지 부분은 기존 라우터와 거의 유사한 동작을 한다. Mobile IP와 상호작용하는 부분도 이 함수에 집중되어 있다.

4.4.1 FA+의 구조

FA+의 시스템 구조를 (그림 15)에 도시한다.

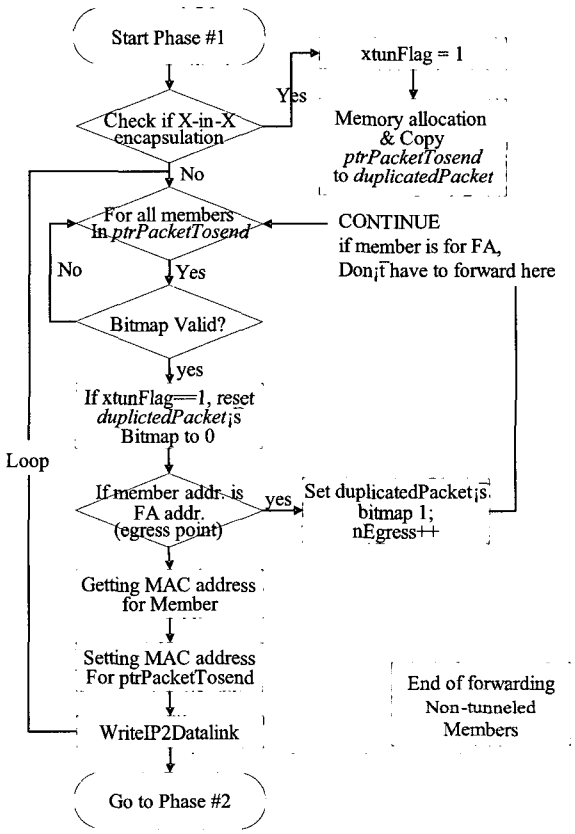


(그림 15) FA+ 구조

1) config parser: Configuration file로부터 FA+ 의 설정사항들을 읽어 내부의 저장소로 저장하는 역할을 한다.
2) Working information: 현재 FA+의 동작 상태 및 처리한 패킷의 수 등 운용 시 발생하는 로그 기록 등을 가지고 있다.
3) Routing table: 유니캐스트 라우팅 테이블을 그대로 사용한다.
4) Xcast protocol processor: Xcast의 기본적인 동작을 맡는 부분이다.
5) Mobile IP Interface: Mobile IP와 연동하기 위한 부분이며, Mobile IP에서 제공하는 라이브러리를 사용하여 접근하고 있다. 이 인터페이스를 통해서 FA+는 visiting list 정보를 얻어 올 수 있다.
6) Xcast routing table: 들어온 패킷에 대해 라우팅을 수행해 준다.
7) Raw socket receiver: 기본적으로 Xcast 패킷을 받기 위해 raw socket을 사용한다.
8) Xcast decapsulator: Xcast 터널 해제를 수행하는 모듈이다.
9) FA Daemon: Mobile IP의 FA 데몬이다.
10) Mobility Subscribing Binding List: CoA의 바인딩 리스트를 FA 데몬에서 Script File을 이용하여 참조한다.
11) ARP: FA에서의 Static한 ARP를 참조하여 패킷을 보낸다.

4.4.2. FA+ 처리 과정

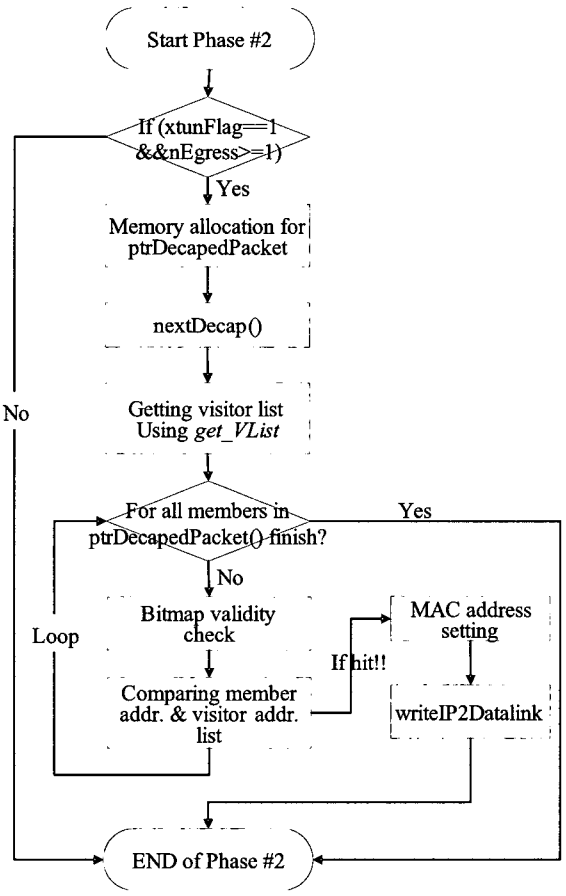
(그림 16)과 (그림 17)에 FA+의 주요 기능 흐름도를 도시한다.



(그림 16) FA+ 주요 흐름도-I

1) Xcast 패킷이 들어면, 자신의 visiting list를 조사한다.
2) 만일 visiting list에 존재하는 주소를 발견할 경우 자신의 subnet으로 패킷을 전송하여 주고 해당되는 엔트리의 비트맵은 0으로 마킹한다.
3) visiting list 처리가 완료된 이후 Xcast 라우팅 모듈로 보내진다.
4) Xcast 패킷이 터널링 되어있는지 여부를 조사한다. 만일 그러한 경우, LoA가 자기 자신으로 향해 있는 것들을 별도의 next hop으로 취급하여 분류한다.
5) FA 자신과 무관한 패킷들에 대해서는 일반적인 Xcast 라우팅을 수행한다.
6) tunneling egress point가 FA+로 되어 있는 주소들에 대해서는 일단 터널 해제를 수행한다. 비트맵 상속도 동시에 수행된다.

7) 터널 해제 후 남아 있는 주소들과 visiting list 정보를 다시 비교한다.
8) 해당되는 엔트리를 발견할 경우 자신의 서브넷으로 전송해 준다. 만일 발견하지 못한다면 패킷을 제거한다.



(그림 17) FA+ 주요 흐름도-II

4.4.3 Mobile IP Interworking

FA+ 역시 Mobile IP의 status tool을 참조하여 들어온 Xcast 패킷이 자신의 visiting list에 있는 지 확인한다. 그리고, ARP 모듈을 참조하여 처리할 패킷의 출력 인터페이스를 결정한다.

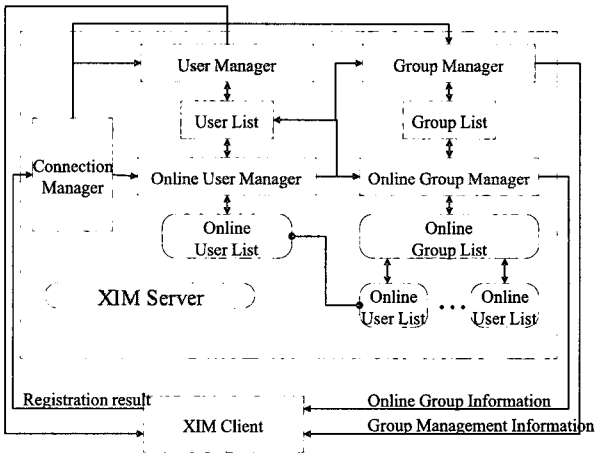
5. XIM 응용 개발

5.1. XIM 개요

응용프로그램인 XIM (XMIP Instant Messenger) 개발 목적은 무선랜기반의 All-IP 망을 통해 끊김없는 멀티캐스트 서비스가 XMIP 프토로콜에 의해 효과적으로 제공될 수 있으며 그 성능도 실제 사용자들이 이동성을 느끼지 못하는 수준까지 제공할 수 있음을 입증하는 것이다.

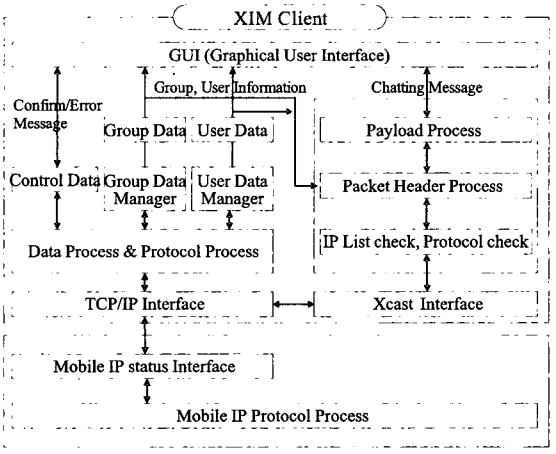
XIM 서버는 Microsoft의 Visual C로 개발하여 Windows 2000 환경에서 동작하며, 기본적인 XIM 사용자 정보관리 및 Xcast 그룹 세션을 관리한다. 필요시 모든 온라인 상태의 클라이언트들에게 긴급공지를 Xcast할 수 있는 기능을 가진다. 서버에서는 클라이언트들간 메시지 송수신에 간섭하지 않으며 오직 온라인 클라이언트들과 그들이 속한 세션 정보들을 관리하여 새로 갱신되는 사용자 정보 (Nickname, IP 주소 및 Port 번호 등)를 클라이언트들에게 전송하는 관리시스템 역할만을 수행한다. 실제적인 사용자 메시지는 클라이언트들 사이에서 Xcasting된다.

5.2. XIM 설계 및 개발



(그림 18) XIM Server 구조

클라이언트들은 유니캐스트 방식을 이용하여, 처음 서버에 접속하여 등록과 소속 그룹을 정한다. 그리고, 해당 그룹에 대한 세션 정보를 수신하여 사용자 메시지 전송 시에 그룹에 속한 모든 클라이언트들의 주소를 Xcast 패킷의 LoA에 삽입하여 망에 전송한다. 이는 각 목적지 클라이언트 노드들이 속한 HA+로 보내어 지고, HA+에서는 클라이언트들의 CoA를 확인하고 다시 X-in-X 터널을 통해 해당 FA들로 보낸다. 새로운 사용자가 그룹에 등록되거나 탈퇴하면 그 정보를 가입된 모든 클라이언트들에게 전송되어, 송신자의 Xcast LoA에서 새로이 삽입하거나 제거하여 동적인 그룹통신이 이루어지도록 설계한다.

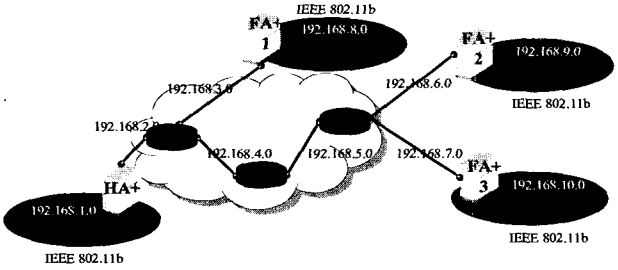


(그림 19) XIM Client 구조

6. 실험 및 결과

6.1. 실험 환경

6.1.1. 실험망 설계



(그림 20) 실험망 구성도

(그림 20)과 같이 1대의 HA+와 3대의 FA+, 4대의 MN, 3대의 Xcast 라우터 그리고 4대의 AP를 이용하여 실험망을 구성하였다. 구축한 실험망은 본 논문의 연구 주제인 XMIP의 특성에 맞게 분기점을 2곳 이상 만들어서 Xcast 패킷이 효율적으로 전송되는 지 검증할 수 있다. 그리고, 4대의 MN이 각각 홈 네트워크와 방문 네트워크에 위치시켰다.

6.1.2. 실험 파라미터

<표 4>에 본 연구의 실험에서 사용하는 장비, 프로토콜 및 주요 요소 들에 대한 규격을 제시한다.

〈표 4〉 실험 파라미터

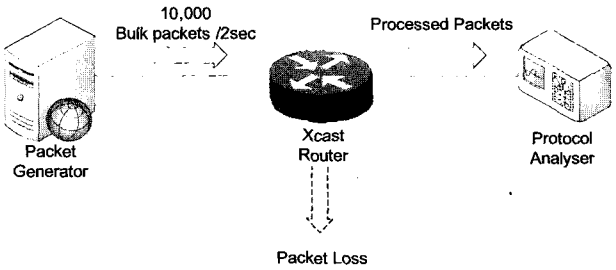
항 목	규 격
Mobility Agent 4대	Pentium 4 1.6GHz
라우터 3대	Pentium 3 866MHz
XIM 클라이언트 4대 (Linux 노트북2대/MS Window 노트북 2대)	Pentium 3 800MHz
XIM 서버 1대	HA와 시스템 공유
무선 네트워크 대역폭	11Mbps (IEEE802.11b)
무선랜 카드/AP	Orinoco WLAN Card/AP
Mobile IP	IETF RFC2002 준수
XMIP	IETF XMIP Draft [7]
보장 핸드오프지연	<1.5 sec
서버간 연결 링크	Ethernet 100BaseT

6.2. 실험 결과

6.2.1. Xcast 라우터 성능 실험

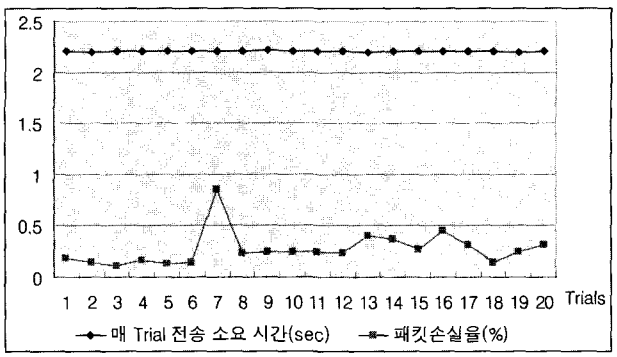
라우터 성능 실험을 위해서 일정 수의 패킷을 고속 전송하여 라우터에서 발생하는 패킷 손실율과 평균적인 패킷 처리 시간을 측정한다. Xcast 라우터를 중간 노드로서 지정하고 패킷 생성기로부터 Xcast 패킷을 생성하여 라우터로 전송하면 Xcast 라우팅 및 헤더처리가 된 패킷을 프로토콜 분석 시스템이 수신하는 형태의 간단한 망을 (그림 21)과 같이 구성한다.

패킷 생성기에서는 Xcast 패킷의 주소 리스트에 프로토콜 분석 시스템의 IP 주소와 포트번호를 삽입한다. 실험 패킷은 IP 헤더 부분의 20 바이트와 Xcast 헤더부분 20 바이트, 그리고 payload부분 1400 바이트의 총 1440바이트 길이의 실험용 패킷이다. 실험 방법은 해당 Null 패킷 10,000개를 강제적으로 Xcast 라우터를 통해 프로토콜 분석 시스템을 대상으로 전송한다. 전송방식은 raw socket을 통해 Xcast 패킷을 IP에 실어 전송하므로 전송율 제어, 흐름제어 및 에러제어 등은 적용되지 않다. 따라서, 운영체제를 포함한 시스템과 네트워크의 성능이 보장하는 최대한의 속도로 전송하게 된다. 실험 시 Xcast 라우팅과 무관한 Xcast 옵션 기능은 비활성화하여 실험한다.



(그림 21) Xcast 라우터 성능실험

프로토콜 분석 시스템에서는 Xcast 라우터에서 처리된 패킷을 수신 후 Ethereal을 통해 분석한다. 이때 라우터의 처리 능력을 넘어서 패킷을 수신 하면 오버플로우 현상이 발생하는데, 이러한 Xcast 패킷 처리용량을 Xcast 라우터의 성능 지표로 삼는다. 이는 본 실험과 동일한 전송율의 일반 IP 패킷 전송 시 전송 실패율이 거의 0%이며, 프로토콜 분석 시스템의 수신에 따른 패킷 손실은 거의 없다는 사실을 사전 실험을 통해 검증했다. 동일한 전송 실험을 20회 반복한 결과를 (그림 22)에 도시한다.

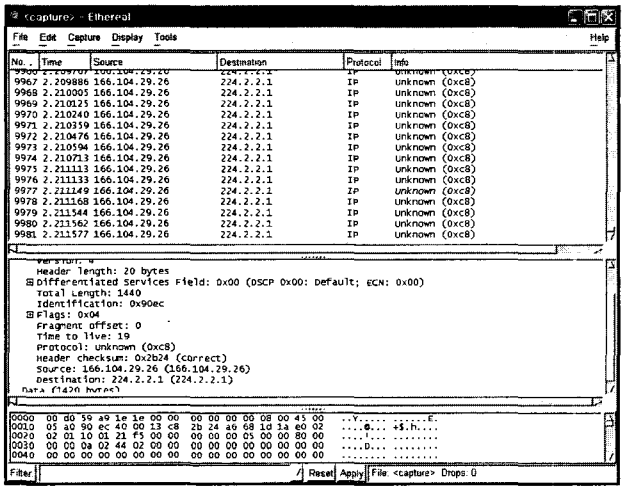


(그림 22) 라우터 성능 실험 결과

10,000개의 입력 패킷에 대해 평균적으로 27.4개의 패킷 손실로 약 0.27% 패킷손실율이 발생하였다. 또한, 프로토콜 분석 시스템에서 9972.6개의 패킷을 수신하는데 평균 2.21초의 시간이 소요되었다.

따라서, 본 논문에서 구현한 Xcast 라우터의 throughput은 Intel P3 866Mhz CPU기반의 리눅스 커널 2.4.17버전을

사용할 경우 52.2Mbps 급의 Xcast 스트림을 0.27% 패킷 손실율 내에서 안정적으로 처리할 수 있는 수준이었다. 참고로서 100Mbps급 고속 이더넷의 실측된 throughput은 표준 전송속도의 70~80% 정도인 것을 고려하면 개발된 PC기반 Xcast 라우터 시스템의 최대 대역폭 중 65%~75% 정도를 사용하고 있다고 볼 수 있다. 또한, 본 Xcast 라우터의 성능은 인터넷 전화 응용의 요구사항인 5~20%의 패킷 손실 허용 및 최대 20kbps 정도의 미디어 대역폭 요구사항을 고려하면, 2,600회선정도의 그룹통신형의 VoIP 응용에도 적용할 수 있는 처리량이다 [12]. 본 연구는 100Mbps급의 이더넷과 리눅스기반 PC급 라우터를 이용한 소프트웨어적 라우팅에 의존하는 형태이므로 실제 망에서 실시간 운영체제 및 하드웨어기반 라우팅을 적용할 경우 성능은 크게 향상될 수 있으리라 사료된다. 그러나, 이는 추가적 Xcast 헤더 옵션없이 단일 수신자만을 가정하였기 때문에 절대적인 성능이라고 단언할 수는 없으며 수신자 수나 Xcast 옵션 처리 여부에 따라 다소간 처리 능력이 달라질 수 있다.

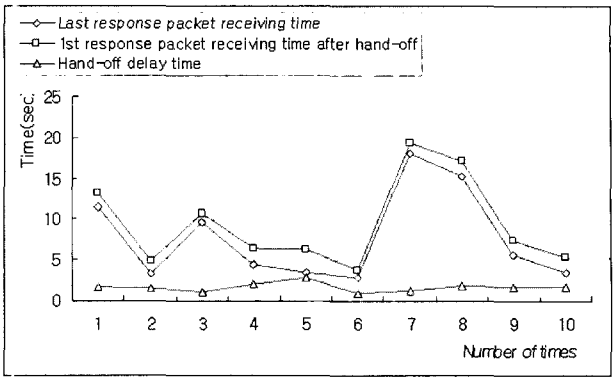


(그림 23) Ethereal의 패킷 캡처 예시

본 연구에서는 Xcast 라우터 성능 실험 이외에도 헤더 옵션 처리 및 분기능력 등의 Xcast 기능성 실험들을 수행하였으며, 이로부터 얻어진 결과들에 의해 Xcast 프로토콜 기능이 설계된 대로 정확히 수행됨을 확인할 수 있었다. 이 실험 결과는 지면을 통해 시각적인 표현이 힘든 관계로 본 절에서는 생략하고 2.2절의 XMIP와 2.3절의 XIM 실험을 통해 결과를 제시하도록 한다.

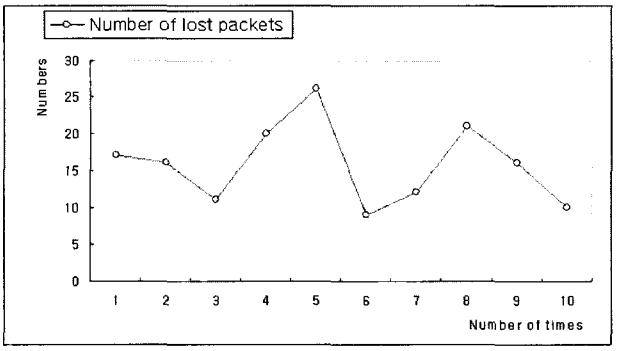
6.2.2. XMIP 프로토콜 실험

실험의 주요 목표는 XMIP망을 통한 Xcast 패킷의 안정적 모바일 멀티캐스팅이다. 다시 말해서, 송신 노드로부터 HA+로 전송된 Xcast 패킷을 HA+가 각 FA+들로 X-in-X 터널을 통해 원본 Xcast 패킷을 전송하여 종단 이동단말까지 안정적으로 전송하되는 것을 확인한다. 아래 (그림 24)는 테스트망에서 XMIP에 의해 핸드오프 시 발생되는 전송지연시간을 보인다.



(그림 24) XMIP망에서의 핸드오프 지연 시간

결론적으로, 평균 핸드오프 전송지연시간은 약 1.5초로 매우 고른 핸드오프 지연처리가 이루어 졌다. 측정방법은 시스템 타이머를 이용하여 패킷이 단절 후 수신이 재개되는 시간을 측정하는 방법을 사용하였다. (그림 20)의 실험망에서 송신노드와 수신노드의 종단간 전송지연은 망의 구조상 최대 10 hop이 가능하다. 즉, 소스 노드가 FA+2에 존재하고 수신노드가 FA+3에 위치할 경우 최대 전송 지연이 발생할 수 있다. 그러나, 앞서 5.1절에서 계산된 Xcast 라우터의 throughput을 기준으로하면 각 라우터의 패킷 처리 지연은 약 0.2msec/packet가 소요되며 망의 크기가 작아 propagation delay 역시 매우 작은 값이므로 전체 전송 지연, 즉 Propagation delay + Packet Processing Delay + Queueing Delay, 는 무시할만하다. 실험적으로도 전송 지연시간은 수 msec 내의 실시간성이 유지되었다.



(그림 25) 핸드오프에 따른 패킷 손실량

핸드오프처리에 대한 다른 측정방법으로 전송율이 150Kbps 급 Xcast 트래픽을 발생시켰을 때, 핸드오프에 따른 XMIP 망에서의 패킷 손실량을 측정하였다. 이를 (그림 25)에 도시한다. 이는 (그림 24)의 핸드오프 평균시간인 1.5초에 근접한 결과를 역으로 추적할 수 있는 패킷 손실량이었다. 미디어 측면에서 분석해보면 150Kbps 대역의 QCIF급 15fps정도의 VoD가 가능한데, 본 연구의 XMIP망에서 평균 16개의 패킷이 핸드오프 시 손실된다고 가정하면 180Kbit의 데이터 손실이며, 대략 영상 20frame정도가 손실되는 것으로 추정된다. 따라서, 수신 버퍼 메커니즘을 도입한 멀티미디어 플

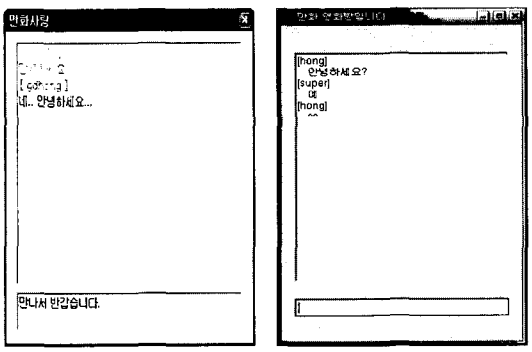
레이어의 경우 그 끊어짐을 심하게 느낄 수 없을 만큼의 데이터량이다.

이러한 실험 결과는 테스트망이 무선랜을 기반으로 구축되어 있으므로 하드 핸드오프 (hard handoff)에 기초하였기 때문에 발생한다. 만일 cellular 네트워크처럼 소프트 핸드오프를 지원할 경우에는 패킷 손실이 거의 발생하지 않을 것이며, 더불어 무선 네트워크 인터페이스로부터의 핸드오프 트리거(Trigger) 이벤트 지원이 있다면 보다 효율적인 핸드오프처리가 이루어 질 수 있다. 그러나, 본 연구에서와 같이, 무선랜기반으로 망이 구성될 경우에는 모든 핸드오프 처리가 Mobile IP 프로토콜에 의존할 수 밖에 없다. 따라서, Mobile IP 프로토콜이 가능한 빠른 핸드오프를 처리하도록 하고 반면에 지나친 컨트롤 오버헤드를 발생되지 않도록 조율(tradeoff)해야 한다.

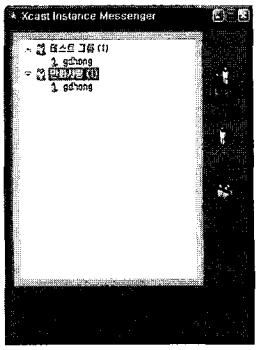
6.2.3. XIM 실험 결과

본 연구에서는 XMIP를 기반으로한 이동환경에서의 멀티유저 메시저 응용으로 XIM을 개발하였다. 실제로 XIM은 XMIP 프로토콜과는 직접적인 관련성이 없으며, 오히려 Mobile IP와 Xcast를 기반으로 한다. 앞서 기술한 바와 같이 XMIP 프로토콜은 중복된 패킷의 전송을 최소화하기 위해 HA+와 FA+ 노드들간에 X-tunnel을 구성하여 X-in-X 전송방식으로 FA+까지 Xcast 패킷을 전송한다. 따라서, Xcast 터널 해체는 FA+에서 수행하고, XIM 클라이언트는 Xcast 처리와 Mobile IP 처리를 수행한다.

실험과정은 XIM 서버를 실행시켜 클라이언트들들이 등록, 인증 및 세션생성 단계를 수행하도록 한다. 이때, 클라이언트와 서버간은 Xcast가 아닌 TCP 기반 통신이 이루어진다. 생성된 세션은 서버에서 생성자가 삭제하기 전까지 유지되며, 다른 XIM 클라이언트는 모든 세션정보를 서버로부터 수신하여 자신이 원하는 세션에 가입하게 된다. 결과적으로 동일한 세션에 참가하는 모든 클라이언트들은 서로의 IP와 Port 정보를 공유하게 된다. 이후 사용자 데이터 즉 메시징 데이터는 동일 세션에 참여하는 모든 사용자들의 어드레스(주소, 포트)로 Xcasting 된다. 이동 중인 노드의 경우엔 HA+를 거쳐 XMIP 프로토콜을 이용하게 되고 홈 네트워크에 존재하는 노드는 직접 Xcast 패킷을 수신한다.



(a) MS 메신저 실행 (b) Linux 메신저 실행



(c) XIM 클라이언트 표준 UI

(그림 26) XIM 응용프로그램 (MS/Linux)

결론적으로, XIM 클라이언트 시스템들로부터 생성된 Xcast 패킷들이 테스트망에 분산되어 이동 중인 다른 클라이언트들까지 XMIP/Xcast 네트워크를 통해 패킷 손실없이 안정적으로 전송됨을 확인하였다. 스트리밍 서비스와는 달리 메시징 서비스의 경우 사용자의 행위 패턴이 일정하지 않고 idle time이 상대적으로 길어서 핸드오프에 따른 패킷 손실이 거의 일어나지 않았다. 그리고, 앞서 5.2절에서 언급한 바와 같이 종단간 전송 시간은 무시할 수 있는 정도이므로 실시간 통신이 가능하였다.

하드 핸드오프 기반의 이동망에서 핸드오프에 따른 패킷 손실이 발생하더라도 응용의 미디어 속성, 데이터의 전송량 및 핸드오프 지연시간 등을 고려하여 데이터링크 계측, 전송 계층 및 응용계층 등 다양한 계층에서의 안정적 전송방식을 고려할 경우 완벽하게 해결할 수 있을 것이다. 그러나, XMIP망에서의 안정적 전송관련 문제는 향후 연구과제로 남겨 두도록 한다.

7. 결 론

기존 멀티캐스트 라우팅 기술들은 대부분 구현되어 오버레이 네트워크의 형태로서 실제 응용 서비스와 연동하여 사용되고 있으며 네트워크 자원의 효율적 사용 측면에서의 유효성 이 입증되고 있다. 그러나, 많은 멀티캐스트 라우팅 기술들은 확장성, 멀티캐스트 트리 유지를 위한 오버헤드, 멀티캐스트 그룹 주소 할당 문제, 그룹세션 제어 문제, 상용화를 위한 AAA 제공 문제 등에서 심각한 한계를 드러내고 있다.

최근 3GPP등에서 중요하게 다루고 있는 POC(PTT over Cellular)로 대표되는 그룹형 VoIP 서비스가 바로 핵심적인 차세대 모바일 통신 서비스가 될 수 있는데, 이러한 엄청난 수의 소규모 그룹 세션이 생성될 수 있는 킬러 서비스의 출현 가능성이 높은 상황에서 기존 멀티캐스트 기술에 비해 향상된 그룹통신 서비스 기능과 이동환경에 신속히 적응할 수 있는 새로운 멀티캐스트 기술의 개발은 매우 시급한 문제가 아닐 수 없다. 따라서, 본 연구에서 수행한 XMIP 기술의 개발과 모바일 환경에서의 실험을 통한 실용성 검증은 매우 중요한 의미를 가진다. 이에 본 논문에서는 IETF에 표

준화 Draft 문서로서 제안된 Xcast와 XMIP 프로토콜을 기반으로 시스템 설계과정을 거쳐 완벽한 시스템으로 구현하였다 [4-7]. XMIP 구현측면에서 Xcast와 Mobile IP를 각각 독립된 프로세스 형태로 구성하여 상호 의존성을 최소화하였고, Mobility Agent들의 바인딩 정보를 Xcast 노드들과 공유할 수 있는 기반을 마련함으로써 상호 운용 측면에서 효율성을 높였다. 이는 향후 독립적으로 진화할 두 기술간에 부드러운 접합점으로 그 활용성을 더욱 높일 수 있을 것이다. 마지막으로 All-IP 기반 모바일 실험망을 구성하고, 실험적인 그룹 메시징 시스템인 XIM 시스템의 구현 및 XMIP망과의 연동실험을 통해 XMIP 및 Xcast 프로토콜에 대한 실증적 검증을 완료하였다.

참 고 문 헌

[1] Steve Deering, "Multicast Routing in a Datagram Internetwork," PhD thesis, Dec., 1991.

[2] C. Diot et al., "Deployment Issues for the IP Multicast Service and Architecture," IEEE Network Magazine, Jan., 2000.

[3] H. Holbrook et al., "Source-Specific Multicast for IP, <draft-ietf-holbrook-ssm-arch-00.txt>, 2000.

[4] R. Boivie et al., "Explicit Multicast(Xcast) Basic Specification," <draft-ooms-xcast-basic-spec-00. txt>, 2000.

[5] J. Lee and M. Shin, "Explicit Multicast Tunneling," <draft-lee-xcast-tunneling-00.txt> Nov., 2001.

[6] J. Lee and M. Shin, "Explicit Multicast over Ethernet," <draft-lee-xcast-ethernet-01.txt>, Sep., 2001.

[7] J. Lee and M. Shin, "Explicit Multicast over Mobile IP (XMIP)," <draft-lee-xcast-mobileip-00.txt>, Nov., 2001.

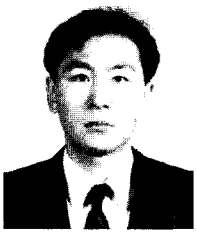
[8] C. Perkins, "IP Mobility Support," IETF RFC 2002, Oct., 1996.

[9] M. Shin et al., "Multicast delivery using explicit multicast over IPv6 networks," IEEE Communications Letters, Feb., 2003.

[10] S. Egger and T. Braun, "Multicast for Small Conferences: A Scalable Multicast Mechanism Based on IPv6," IEEE Communications Magazine, Jan., 2004.

[11] M. Shin et al., "Explicit Multicast Extension (Xcast+) for Efficient Multicast Packet Delivery," ETRI Journal, Dec., 2001.

[12] A. Croll and E. Packman, "Managing Bandwidth Deploying QOS in Enterprise Networks", page 95, Prentice Hall PTR, 2000.



박 인 수

e-mail : ispark@nclab.hanyang.ac.kr

1986년 한양대학교 전자공학과(학사)

1990년 한양대학교 대학원 전자공학과 (공학석사)

1990년~현재 KT 책임연구원

관심분야 : Mobile IP, 3GPP IP Multimedia Service, Multicast, QoS, Wireless Ad-hoc Network

박 용 진

e-mail : park@nclab.hanyang.ac.kr

1969년 일본 와세다대학교 전자통신학부 (학사)

1971년 일본 와세다대학교 전자통신학부 (공학석사)

1978년 일본 와세다대학교 전자통신학부 (공학박사)

1978년 한양대학교 공과대학 교수

1983년 Univ. of Illinois, Urbana 객원교수

1991년 Univ. of Kent, Canterbury 객원교수

1998년 일본 와세다대학교 객원교수

1998년~2000년 APAN 사무총장

1999년~2000년 IEEE 서울지부장

2003년 IEEE Region 10 총무이사

2004년 한국정보과학회 회장

현 재 한양대학교 공과대학 전자통신컴퓨터공학부 교수

관심분야 : High-Quality Multimedia Application, P2P Application, Mobile Agent, NGI, Grid Computing, Wireless Ad-hoc Nework, Ubiquitous Computing