SDL을 이용한 MPOA 설계 및 구현
(Design and Implementation of MPOA using SDL)

임 지 영 † 김 희 정 † 임 수 정 † 채 기 준 ††
(Ji-Young Lim) (Hee-Jung Kim) (Soo-Jung Lim) (Kijoon Chae)

이 미 정 ††† 최 길 영 †††† 강 훈 †††††
(Meejung Lee) (Kil-Young Choi) (Hun Kang)

요 약 ATM 포럼에서 제안하여 표준화 한 MPOA(MultiProtocol Over ATM)는 다양한 네트워크 환경에서 ATM 네트워크에게 효과적인 브릿징과 라우팅을 제공하는 프로토콜로써, 서브넷 간에 효율적인 유니캐스트 데이타 전송을 주된 목적으로 하고 있다. 본 논문에서는 ITU에서 표준화한 통신 시스템 개발용 명세 기술 언어인 SDL(Specification and Description Language)을 사용하여 MPOA 구성요소를 구현하였으며, 상위 계층에서 받은 패킷에 대한 주소 해석, Ingress/Egress 캐쉬 관리, 지름길 설정 등에 따른 MPOA 작동 절차를 SDT(SDL Design Tool)에서 제공하고 있는 시뮬레이터를 통하여 검증하였다.

Abstract MPOA proposed and standardized by the ATM Forum is a protocol that provides effective bridging and routing for ATM networks in a diverse network environment. Its primary goal is to transfer unicast data effectively among the subnets. In this paper, MPOA components are implemented using the SDL(Specification and Description Language) which the ITU has standardized for the development of communication systems. In addition, MPOA procedures for various operations such as address translation for packets from upper layers, Ingress/Egress cache management and shortcut configuration, are examined with the help of the SDT(SDL Design Tool) simulator.

1. 서 론

인터넷의 급격한 성장과 실시간 멀티미디어 어플리케이션 사용의 증가 및 네트워크 트래픽 패턴의 변화로인해 기존의 LAN을 기반으로 한 네트워크와는 다른 새로운 하부구조가 요구되고 있다. 이에 따라 실시간 전송, 확장성 및 대역폭에서 월등한 장점을 가진 ATM 기반의 초고속 정보 통신망을 활용하여 통신 하부 구조를 제공받으면서, 기존의 Ethernet, Token Ring, TCP/IP와 같은 다양한 상위 계층의 프로토콜을 유지하는 방안이 여러 표준화 기관에서 연구되고 있다[1].

ATM 포럼에서 정의된 MPOA는 서브넷 경계를 지날 때 라우터를 사용해야 하는 LANE(LAN Emulation)의 단점을 극복하여 ATM 종단간의 지름길을 설정하는 절차를 제공한다[2]. MPOA에서는 IP 주소를 ATM 주소로 해석하기 위하여 IETF에서 제정한 NHRP(Next Hop Resolution Protocol)를 사용한다. 또한 3계층 주소 해석 기능과 데이타 포워드 기능의 물리적인 분리라는 가상 라우터의 개념을 도입하여 네트워크 확장성과 적응성을 증가시키고, 서로 다른 서브넷 간의 SVC 설정을 통하여 라우터를 거치지 않고 데이타

· 본 연구는 한국전자통신연구원 광대역 통신망 연구부 위탁 연구 과제에 의한 것임
† 비 회 원 : 이화여자대학교 컴퓨터학과
jyjim@ewha.ac.kr
hjkim7@lgcit.com
982COG18@ewha.ac.kr
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
kjchae@ewha.ac.kr
††† 정 회 원 : 이화여자대학교 컴퓨터학과 교수
lmj@ewha.ac.kr
†††† 비 회 원 : 한국전자통신연구원 교환전송기술연구소 연구원
kychoi@etri.re.kr
††††† 비 회 원 : 한국아이티벤처투자(주) 상무이사
hkang@korcait.com
논문접수 : 2000년 1월 15일
심사완료 : 2000년 9월 8일

를 포워드 함으로써, 기존 라우터의 hop-by-hop 라우팅 방식 보다 성능을 향상시킨다.

Fore 시스템과 같은 외국 회사에서는 MPOA를 구현하여 제품으로 개발하였으나 아직 국내에서는 제품화된 상태는 아니다. 또한 MPOA에 대한 성능 평가에 대한 연구는 표준안에 입각하여 표준안에서 허용하는 범위 안에서 MPOA 매개변수의 값을 변경하면서 그 성능을 비교한 정도로서 아직 MPOA에 대한 성능 평가에 대한 연구가 많이 이루어진 상태가 아니다[3,4]. 뿐만 아니라 MPOA 구현에 관련된 논문도 부재하다. 본 연구의 목적은 MPOA기반 라우터 장비 개발을 위한 MPOA의 기능인 MPC와 MPS의 구현에 있다. 또한 이의 작동을 테스트하기 위해서 표준안에 따라 MPOA 시스템에서 가능한 시나리오를 작성하여 각 단계의 상태를 표준안과 비교하였다. 본 논문에서는 MPOA의 구성 요소인 MPC(MPOA Client)와 MPS(MPOA Server)의 기능을 명세 기술 언어인 SDL(Specification and Description Language)을 이용하여 구현하였다. 또한 구현한 MPOA 모델이 올바르게 동작하는지 검증하기 위해 SDT (SDL Design Tool)에서 제공하는 시뮬레이터를 통하여 그 동작 과정을 테스트하였다.

MPOA 구현에서 사용한 SDL 언어는 ITU(International Telecommunication Union)가 표준화한 통신 시스템 개발용 명세 기술 언어이다. 네트워크가 점차 복잡해지고 실시간 시스템으로 발전하게 되자 기존의 언어로는 복잡한 네트워크의 동작을 표현하는데 한계를 느끼게 되었다. 이를 극복하고자 복잡한 통신 시스템의 동작을 표현하기 위한 명세 언어와 형식적인 상세 기술의 필요성이 대두되었으며 그 대표적인 언어로서 SDL이 제시되었다[5].

본 논문의 구성은 다음과 같다. 제 2 장과 제 3 장에서는 각각 MPOA의 개요와 SDL의 기본 개념을 설명하고, 제 4 장에서는 SDL을 이용한 MPOA 구현에 대하여 기술한다. 제 5 장에서는 구현한 MPOA 시스템을 4가지 시나리오에 따라 시뮬레이터로 검증한 결과를 보여준다. 마지막으로 제 6 장에서는 결론으로 끝을 맺는다.

2. MPOA의 개요

2.1 MPOA의 기본 개념

MPOA는 다양한 네트워크 환경에서 ATM 네트워크에 효과적인 브리징과 라우팅을 제공하기 위한 프로토콜로서 ATM 포럼에서 제안되고 표준화되었다. ATM 네트워크는 Ethernet, Token Ring, FDDI 등의 기존의 LAN에 비하여 실시간 전송 및 대역폭 면에서 월등한 장점을 가지고 있어 이러한 ATM 네트워크와 이미 널리 퍼져있는 기존 LAN 간의 연동이 요구되었다. 이를 위해 ATM 포럼에서는 다양한 상위 계층 프로토콜을 유지하면서 망계층 라우팅을 제공하는 MPOA를 연구하고 제안, 표준화하게 되었다.

MPOA는 서브넷 간의 유니캐스트 데이타를 효율적으로 전송하는 데에 목적을 두고 다양한 망계층을 수정없이 사용하는 LANE와 3계층 주소 해석에 의해 직접적인 ATM 연결을 제공하는 NHRP를 통합적으로 도입하였다. ATM 포럼의 LANE은 ATM 네트워크 상에서 서브넷 내의 데이타는 효율적으로 전송하는 수단을 제공하지만 서브넷 간의 데이타 전송에는 여전히 라우터를 필요로 한다. 이에 비해 MPOA는 다른 서브넷 간의 지름길 설정에 있어서, 직접적인 연결을 제공하는 NHRP를 사용하여 라우터를 거치지 않고 ATM VCC를 설정함으로써 적용범위가 하나의 ELAN으로 제한되는 LANE의 단점을 보완해 준다.

또한 MPOA는 전통적인 라우터 기반의 네트워크의 기능을 에뮬레이트하면서 hop-by-hop 라우팅의 성능적인 한계점을 제거한 가상 라우터의 개념을 도입하여 확장성과 적응성을 제공하게 되었다. 그림 1은 MPOA 시스템이 MPC(MPOA Client)와 MPS(MPOA Server)에 각각 데이타 포워딩 기능과 3계층 라우팅 계산 기능을 물리적으로 분리시켰음을 보여준다. 즉, 주소 해석과 토폴로지 발견은 라우터에 존재하는 MPS에서 수행하고, ATM 스위치를 통한 트래픽 포워딩은 기존 LAN과 ATM을 연결하는 엣지 디바이스나 ATM 호스트에서 있는 MPC에서 담당한다. 이로써, 라우팅 계산을 수행하기 위해 필요한 디바이스의 수를 줄여 제어성과 확장성을 높이고, 엣지 디바이스에서의 라우팅 계산 기능을 제거하여 복잡성을 줄이게 되었다. 뿐만 아니라 매 홉마다 라우팅 프로세싱을 수행하는 기존의 라우터보다 가상 라우터를 이용하여 ATM 네트워크에서 지름길을 설정함으로써 서브넷 간 솔루션에서 볼 때 우수한 성능을 제공하고, 말단 스테이션 간의 트래픽 지연 시간 또한 감소시킨다.

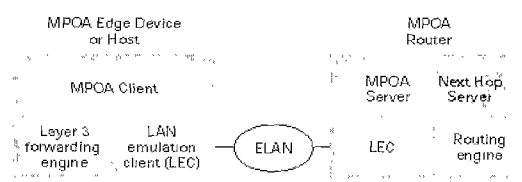

그림 1 MPOA 시스템의 구성요소

2.2 동작 방식

MPOA 모델은 패킷 포워딩을 수행하는 MPC와 라우팅 정보를 제공하는 MPS로 이루어진다. MPC는 올바른 포워딩 결정을 위해서 기존의 LAN 세그먼트로부터 온 패킷의 목적지 MAC 주소를 조사한다. 만약, 패킷이 같은 서브넷 내의 호스트를 목적지로 한다면 LANE을 사용하여 목적지로 데이타를 브리징한다. 그러나 목적지 MAC 주소가 MPOA 라우터 인터페이스의 MAC 주소이면 MPC는 그 패킷의 3계층 목적지 주소 필드를 검사하여,주소 해석 요청을 통해 MPS로부터 정보를 받거나 MPC 자신의 내부에 있는 캐쉬 엔트리 정보를 이용하여 3계층 목적지에 대한 ATM 주소를 결정한다. 캐쉬 엔트리의 정보가 없는 경우에는 정해진 기간 내에 특정 3계층 주소를 가진 패킷 수가 정해진 값을 초과하게 되면 MPC가 Ingress MPC로 되어 Ingress MPS에게 MPOA 주소 해석 요청을 보내게 된다(그림 2의 step 1). 주소 해석 요청을 받은 Ingress MPS가 목적지에 대한 ATM 주소를 모를 때에는, NHRP 주소 해석 요청으로 변환하여 다른 MPS나 라우터에게 주소 해석 요청을 전달한다(step 2). NHRP 주소 해석 요청을 받은 Egress MPS는 같은 서브넷 내에서 자신이 관리하고 있는 MPC 중 목적지 MPC가 존재하면 해당 목적지 MPC(Egress MPC)에게 MPOA 캐쉬 저장 요청을 보내 지름길 설정 시에 필요한 QoS 파라미터를 포함한 정보를 얻는다(step 3). 목적지 MPC는 ATM 주소를 포함하여 주소 해석 요청의 근원지에게 포워드한다(step 4, step 5). 상대방 ATM 주소를 알게되면 목적지와의 직접적인 VC를 설정하여 이로서 라우터 없이 서브넷 간의 커뮤니케이션이 가능하게 된다[6].

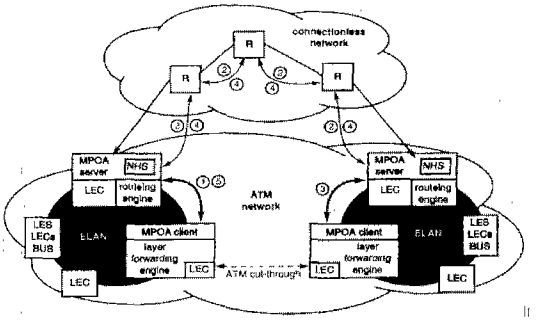

그림 2 지름길 전송을 위한 주소 해석 과정

3. SDL의 개요

SDL은 시스템을 명세화하고 기술하는 표준화된 언어로서 통신 분야에서 여러 동시 작업과 상호 동작이 중요시되는 사건 중심의 실시간 시스템을 기술하는데 적합하다. SDL을 이용하여 프로토콜을 구현할 경우 보다 편리하고 빠른 시간 안에 구현이 가능하며, 구현 내역을 명확하게 표시할 수 있고, 명시한 사실이 바뀌더라도 간편하게 변경 내역을 수정할 수 있다. 계층 구조를 그래픽으로 표현함으로써 타 언어로 구현하는 것보다 쉽게 이해되며, 구현된 시스템에 대한 단계적인 분석이 가능하다. 또한 시뮬레이션과 검증을 위한 시뮬레이터와 검증기를 SDT내에 포함하고 있어 별도로 진행 과정을 표시하는 기능을 추가하지 않아도 시뮬레이션 및 검증이 용이하다.

그림 3은 SDL로 구현된 시스템의 구성요소들이 계층적으로 모듈화됨을 보여준다. SDL의 시스템 구조는 복잡한 프로세스의 동작을 블록 안에 포함시켜 보다 간단하게 표현한다. 한 시스템은 하나 이상의 블록과 이들을 연결해주는 채널로 이루어지며 이러한 블록의 내부는 다시 여러 프로세스들과 신호로 구성된다. 프로세스는 자신의 동작을 수행하고 자신의 지역 속성을 갖는 동적인 개체로서 다른 프로세스들과 시그널을 통해 정보를 상호 교류한다. 또한 프로세스는 프로시저를 호출할 수 있다.

전문화(specialization)와 상속(inheritance) 개념이 블록, 프로세스, 데이타 타입 등 SDL 개념 전반에서 사용될 수 있게 함으로써 객체 지향 설계를 지원한다. 이러한 객체 지향 설계는 압축된 시스템의 설계를 가능하게 하며, 구성 요소를 재 사용할 수 있게 하여 시스템 유지에 필요한 수고를 줄인다[7].

본 구현에서는 SDL을 위한 상용 도구로 Telelogic사에서 개발한 SDT를 SUN Solaris 워크스테이션에 설치하여 사용하였다. SDT는 정보를 처리하는 분리된 도구들이 고도로 통합된 소프트웨어 도구이다. SDT에서 제공하는 SDL 에디터는 Z.100에서 표준화한 그래픽적인 표기방식으로 시스템을 명세화하고 기술할 수 있게 하며, SDL 시뮬레이터는 사용자가 자신이 명세화한 시스템의 동작을 이해하고 디버깅할 수 있게 한다.

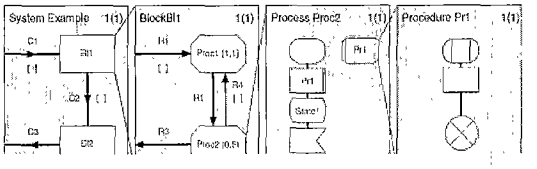

그림 3 SDL 시스템의 계층 구조

4. SDL을 이용한 MPOA 구현

4장에서는 SDL의 계층구조인 시스템 레벨, 블록 레벨, 프로세스 레벨의 특징을 살펴보고 실제로 MPOA 시스템을 구현하는데 있어서의 각 레벨별 역할을 살펴본다.

시스템 레벨에서는 MPOA 목적지 주소 해석 메커니즘을 수행하는 4개의 주요 시스템 엔터티인 Ingress_MPC, Ingress_MPS, Egress_MPS, Egress_MPC들을 블록으로 정의하고 이들간 주고 받는 시그널을 보여주므로써 블록을 통해 MPC와 MPS로 이루어진 MPOA 시스템 전체를 개괄적으로 보여준다. 블록 레벨에서는 MPC와 MPS가 시스템으로 들어가는 인그레스 측면과 시스템에서 나오는 이그레스 측면에 따라 각각 수행하는 기능들을 프로세스로 분류한 것을 보여준다. 마지막으로 프로세스 레벨에서는 실제로 MPC와 MPS의 기능을 수행하기 위한 동작이 상세하게 명시되는 부분으로 플로우 차트처럼 처리 동작의 흐름을 직관적으로 볼 수 있다.

4.1 시스템 레벨

SDL의 시스템레벨은 최상위 레벨로서 시스템의 내용을 개괄적으로 보여줌으로써 전체 시스템에 대한 이해를 도와준다. 시스템은 블록 또는 외부환경과 커뮤니케이션 패스인 채널을 통하여 통신하는 블록들의 집합으로 표현되며 채널은 무한대의 FIFO 큐를 가지고 있어 채널을 통하여 전송되는 시그널을 저장한다. 시그널은 정수형이나 문자형과 같은 간단한 타입에서부터 배열구조를 가지고 있는 사용자 정의 타입까지 다양한 종류의 데이타를 가지고 있는 정보의 패킷이다. 병렬적으로 수행되는 SDL 시스템의 각 프로세스들은 그 수행의 특성상 전역 변수를 가지지 못하기 때문에 각 레벨에서의 구성요소 간 혹은 구성요소와 외부환경 간에 분산 시그널을 통하여 정보를 교환함으로써 서로간의 커뮤니케이션과 데이타의 공유가 가능하다. 따라서 MPOA 시스템을 설정하기 위한 각 블록 간의 제어프레임의 교환과 상대방 목적지로의 데이타 프레임의 전달은 패킷의 필드 값을 매개변수로 가지는 시그널 메시지를 채널 상에서 전송함으로써 이루어진다. 채널을 통해서 교환되는 시그널은 그림 4에서와 같이 텍스트 심볼 내에 시그널 이름과 시그널이 전송하는 전달인자의 데이타 타입을 기입하여 선언하며 채널 위에 각괄호('[ ]') 내에 명시된다. 또 시스템 레벨에서는 전체 구현할 시스템에서 사용되는 데이타 타입이 정의되는데 SDL은 정수형, 불리안 연산, 실수형, 문자형, time 형과 같은 간단한 타입에서부터 배열, struct, powerset과 같은 포괄적인 타입까지 다양한 형태의 데이타 타입을 지원하며 그 표현방식

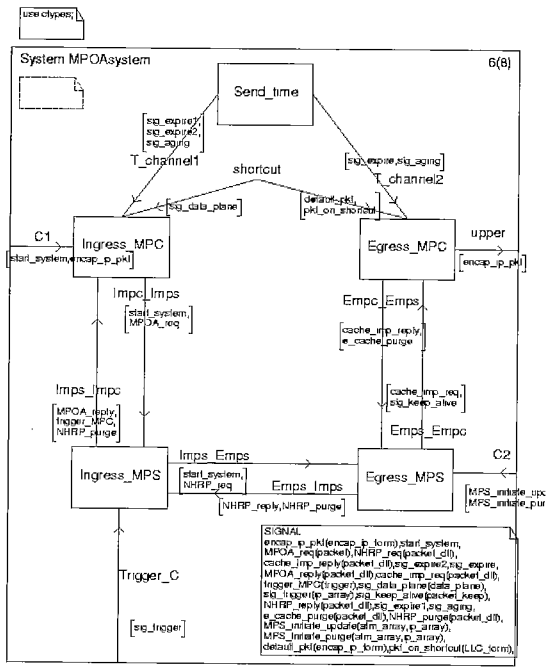

그림 4 시스템 레벨 : MPOA system

은 C 언어나 Pascal 언어와 유사하다. 또한 객체 지향 프로그래밍의 근간이 되는 추상적인 데이타 타입을 지원하여 특정한 데이타 구조를 가지지 않고 대신에 프로그래머가 값이나 데이타 타입에서 허용되는 연산자 그리고 연산자가 수행해야 할 방정식 등을 직접 정의 할 수 있게 한다[7]. 그림 5와 같이 'newtype'과 'endnewtype'이라는 키워드를 이용하면 여러 필드로 이루어진 사용자 정의 데이타 타입을 만들 수 있으며 이 데이타 타입을 실제로 쓰고자 할 때에는 실제 시스템의 동작을 명세하는 프로세스 레벨에서

DCL MPOA_req_pkt packet;

이라고 선언하여 사용한다.

본 구현에서 MPOA 시스템 구성요소들이 주고받는 제어 메시지들의 자료구조는 'packet', 'packet_dll', 'packet_keep', 'trigger' 그리고 'data_plane'등의 사용자 정의 데이타 타입이며 이들은 'fixed_header', 'com_header', 'cies', 'tag_ext', 'serv' 등의 필드로 구성된다. 'packet' 데이타 타입은 MPOA 주소 해석 요청 메시지에서 사용되는 타입이며, 'packet_dll'은 MPOA 주소해석 응답, MPOA 캐쉬저장 요청, MPOA 캐쉬 저장 응답, MPOA Egress 캐쉬 제거 요청 그리고 MPOA Egress 캐쉬 제거 응답에서 사용되는 타입의

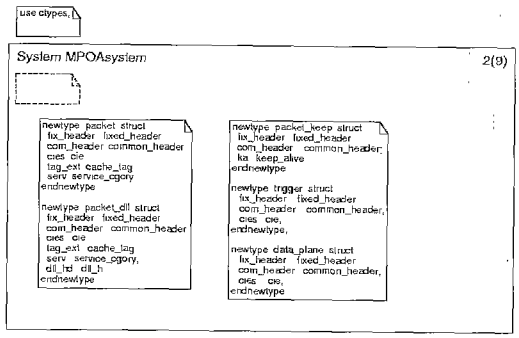

그림 5 시스템 레벨 : 자료형 선언

자료구조이다. 'packet_keep' 데이타 타입은 MPOA Keep_Alive 메시지에, 'trigger'는 MPOA Trigger 메시지에서 사용된다. 마지막으로 'data_plane'은 데이타 플레인으로 NHRP 제거 메시지를 전송할 때 사용된다. MPOA는 모든 제어 메시지에 대해 기본적으로 NHRP에서 정의한 LLC 인캡슐레이션을 사용한다[8].

MPOA 시스템의 논리적인 구성 요소 MPC와 MPS의 기본 작동 원리인 주소해석 과정은 MPOA 시스템으로 들어가는 측면인 Ingress 규칙과 MPOA 시스템에서 나오는 측면인 Egress 규칙으로 나누어 생각할 수 있다. 본 구현에서는 가장 상위 레벨인 시스템 레벨을 목적지 주소 해석 메커니즘을 수행하는 4개의 주요 블록인 Ingress_MPC 블록, Ingress_MPS 블록, Egress_MPS 블록, Egress_MPC 블록과 MPOA 시스템 내의 타이머 동작과 관리를 위한 Send_time 블록으로 구성하였음을 그림 4를 통해 볼 수 있다.

Ingress_MPC 블록은 상위계층으로부터 받은 패킷(C1 채널을 통하여 온 encap_ip_pkt 시그널)이 ELAN상의 MPS가 있는 라우터로 포워드되는 것임을 발견하면, LANE에 의해 이미 설정되어 있는 길보다 지름길 전송을 하는 것이 더 유리한지를 체크한다. 지름길 전송이 더 유리하다고 결정되면, 서로 다른 ELAN에 있는 목적지에 대한 지름길 설정을 요청하기 위하여 같은 ELAN 내에 있는 Ingress_MPS에게 Impc_Imps 채널을 통하여 MPOA 주소 해석 요청인 MPOA_req 시그널을 보낸다. MPOA_req를 받은 Ingress_MPS 블럭은 목적지가 로컬인 경우 직접 응답하고, 그렇지 않은 경우 MPOA_req를 NHRP 주소 해석 요청인 NHRP_req 시그널로 바꾸어 Imps_Emps 채널을 통해 Egress_MPS 블록에게 전송한다. Egress_MPS는 자신이 관리하는 MPC에 대한 NHRP_req가 도착하면 NHRP_req를 MPOA 캐쉬 저장 요청에 해당하는 cache_imp_req 시그널로 바꾸어 해당 Egress_MPC에게 전송한다. 요청을 받은 Egress_MPC는 자신의 자원을 살핀 후에 MPOA 캐쉬 저장 요청을 받아들일 것인지에 대한 응답(cache_imp_reply)을 Empc_Emps 채널을 통하여 보낸다. 응답을 받은 Egress_MPS는 그 메시지를 다시 NHRP 주소 해석 응답인 NHRP_reply 시그널로 바꾸어 Ingress_MPS에게 보내고, Ingress_MPS는 MPOA 주소 해석 응답인 MPOA_reply 시그널로 바꾸어 Ingress_MPC에게 포워드한다.

다섯 번째 블럭인 Send_time은 Ingress_MPC와 Egress_MPC 블럭에서 시간의 흐름에 따라 수행되어야 하는 프로세스의 동작을 가능하게 한다. 매 시간마다 주기적으로 sig_expire 시그널과 sig_aging 시그널을 보냄으로써 MPOA 요청 후에 일정 시간 내에 응답이 오지 않으면 다시 요청하는 제시도 절차, Ingress, Egress 캐쉬 엔트리 내의 holding time을 일정하게 줄여 나가는 캐쉬 에이징 기법, Egress 캐쉬 엔트리를 부여한 MPS가 잘 작동중임을 나타내는 Keep-Alive 프로토콜의 수행을 가능하게 한다.

4.2 블록 레벨

시스템의 동작은 원칙적으로 프로세스에서 명시되지만, 개발자에 의해서 그 프로세스를 단위로 묶어 보다 편리하게 관리할 수 있도록 만든 것이 블록이다. 블록의 내부는 채널을 통해 다시 더 낮은 수준의 블록의 관계를 나타내는 하향식 순환적 표현 방법을 사용하거나, 시그널을 통해 프로세스들의 동작을 표현한다. 그러나 블록과 프로세스가 같은 다이어그램 내에서 혼합되어 쓰여지는 것은 허용되지 않으며, 시스템 레벨에서는 프로세스 다이어그램이 포함될 수 없다.

목적지 주소 해석 메커니즘을 수행하는 4개의 주요 블록인 Ingress_MPC, Ingress_MPS, Egress_MPS, Egress_MPC는 블록속에 있는 프로세스들을 통해 주어진 주요 작업들을 처리하고 블록내에서는 프로세스들의 관계와 흐름을 나타냄으로써 전체적인 처리과정을 좀 더 쉽게 볼 수 있다.

(1) Ingress MPC 블록(그림 6)

Ingress_MPC 블록은 'make_pkt', 'search_entry', 'make_entry', 'threshold_check', 'retry', 'receive_MPOA_Reply', 'I_Aging' 등의 프로세스들과 프로세스 간의 통신을 가능하게 하는 R1, R2, R3 등의 시그널루트로 이루어진다. Ingress_MPC 블록 내에서 상세화되고 있는 프로세스들의 작업 동작은 크게 임계치를 넘거나 Trigger에 의한 주소해석 요청, Ingress 캐쉬 엔트리 에이징, 그리고 데이타 전송으로 나누어 질 수 있다.

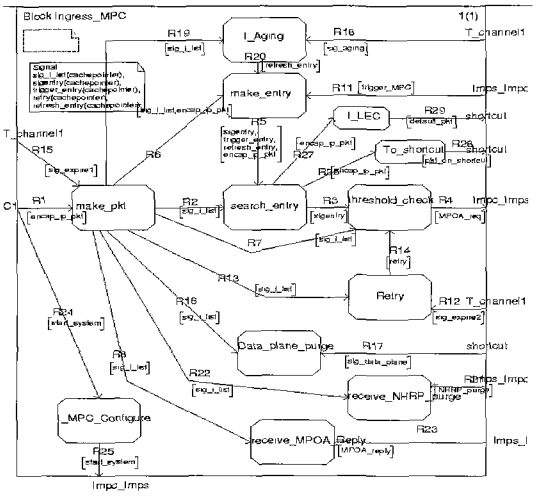

그림 6 블록 레벨 : Ingress_MPC

▪ 임계치를 넘거나 Trigger에 의한 주소해석 요청

MPOA 시스템이 C1 게이트를 통하여 외부환경, 즉 상위계층으로부터 DLL 인캡슐레이트된 IP 패킷 encap_ip_pkt 시그널 패킷을 받으면 이 메시지가 상위계층으로부터 온 것인지 확인한 후에(make_pkt 프로세스에서 수행) 시그널의 전달인자를 통해서 온 정보(목적지의 3계층 주소, Ingress MPS MAC 주소)를 바탕으로 make_entry 프로세스는 <MPS ATM 주소, 3계층 주소>를 키 값으로 하는 캐쉬 엔트리를 생성한다. 이때 기존에 존재하는 캐쉬 엔트리에 대하여 엔트리를 생성한 것이면 지금 생성한 엔트리는 삭제하고 대신에 이전에 있던 엔트리에 대한 카운트 값을 1 증가시킨다(프로세스 search_entry 수행). threshold_check 프로세스에서는 생성된 캐쉬 엔트리의 카운트 값이 정해진 시간(MPC-p2)안에 정해진 임계치(MPC-p1)를 넘는지 조사하여 임계치를 넘게되면 목적지 주소에 대한 지름길 요청을 위해 MPOA 주소 해석 요청을 MPOA_req 시그널로 변환하여 Ingress_MPS 블럭에게 전송한다. MPOA 주소 해석에 대한 응답이 오기 전까지 data 프레임은 I_LEC 블록을 통해 ELAN으로 보내어지게 된다(디폴트 전송). Ingress_MPS에게로 보내어졌던 MPOA 주소 해석에 대한 응답메시지인 MPOA_reply 시그널을 receive_MPOA_Reply 프로세스가 받게 되면 목적지 3계층 주소에 대한 ATM 주소, holding time 등 리턴되는 값으로 캐쉬 엔트리의 내용을 채우게 되어 유효한 정보를 갖게 된다.

▪ Ingress 캐쉬 엔트리 aging

NHRP 캐쉬 제거 요청인 NHRP_purge 시그널을 받아서 Invalid인 상태로 변환시킨(receive_NHRP_purge 프로세스에서 수행) 엔트리를 물리적으로 삭제하고, 주소 해석 응답을 통해 리턴된 holding time을 정기적으로 sig_aging 시그널을 받을 때마다 1씩 줄여 나가면서 Ingress 캐쉬 엔트리를 aging 하게 된다(I_Aging 프로세스에서 수행).

▪ 데이타 전송

MPOA 시스템은 효율적인 유니캐스트 데이타 전송을 위해, 한 목적지로의 전송할 데이타 양이 많지 않을 때는 LEC(그림 6에서 I_LEC 프로세스)를 통한 이미 정해진 경로로 전송하고, 전송량이 많아 목적지로의 지름길 전송이 유리하다고 판단했을 때는 앞에서 설명한 MPOA 목적지 주소 해석 메커니즘과 캐쉬 관리 메커니즘을 통해 지름길을 설정하여 전송한다. <MPS ATM 주소, 3계층 주소> 쌍이 Ingress 캐쉬에 있고 캐쉬 엔트리의 상태가 유효한 경우 패킷의 DLL 인캡슐레이션을 제거하고 적절한 3계층으로 인캡슐레이트한 pkt_on_hortcut 시그널 패킷으로 변환하여 To_Shortcut 프로세스에서 shortcut 채널을 통해 지름길로 전송한다.

(2) Ingress_MPS 블록(그림 7)

이 블록의 주된 기능은 Egress_MPS와 더불어 기존의 NHRP를 확장하여 지름길 전송 시에 필요한 종단간의 ATM 주소를 얻기 위한 목적지 주소 해석 메커니즘을 수행하는 것이다. 목적지로의 디폴트 전송보다 지름길 전송이 이롭다고 판단을 내린 Ingress_MPC 블록으

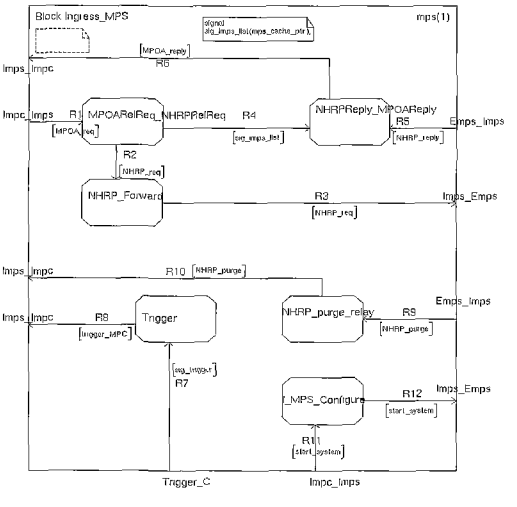

그림 7 블록 레벨 : Ingress_MPS

로부터 Lookup 프로세스에서 MPOA_req를 받았을 때 목적지가 로컬인 경우 직접 응답하고, 그렇지 않은 경우에는 MPOARelReq_NHRPRelReq 프로세스로 포워드한다. MPOARelReq_NHRPRelReq는 MPOA_Req를 NHRP 주소 해석 요청인 NHRP_req 시그널로 변환하여 목적지를 향한 경로인 Imps_Emps 채널로 전송한다. 이 때 NHRP_req 시그널의 매개변수는 새로운 요청 ID 값으로 세팅되고 근원지 프로토콜 주소 또한 Ingress MPS의 주소로 바뀌어, 후에 NHRP 주소 해석 요청에 대한 응답인 NHRP_reply 시그널이 Ingress_MPS까지 잘 돌아왔는지 NHRP_Reply_MPOAReply 프로세스가 확인할 수 있게 한다. NHRP 응답을 받으면 요청 ID와 근원지 프로토콜 주소를 원래의 값으로 복구하여 Ingress_MPC 블록으로 주소해석 응답인 MPOA_reply 시그널을 보낸다.

MPC가 MPOA 시스템 내부로 들어오는 데이타 흐름을 감지하여 MPOA 주소해석 요청을 할 수 있지만 Ingress MPS가 Inbound 데이타 흐름을 감지하여 Ingress MPC에게 MPOA 주소 해석 요청을 개시하게 할 수도 있다. 이 기능을 수행하는 프로세스가 Trigger 프로세스이며 목적지 3계층 주소와 Ingress MPS의 제어 ATM 주소를 포함한 tirgger_MPC 시그널을 Ingress_MPC에게 보냄으로써 이루어진다. trigger후의 과정은 4.2.1 절의 MPOA 주소 해석 요청 시의 과정과 같다.

(3) Egress_MPS 블록(그림 8)

이 블록의 주된 기능은 크게 다음과 같이 나누어 질 수 있다.

▪ Egress 캐쉬 저장 요청

라우터 포워딩 테이블을 조사하여 자신이 관리하는 MPC에 대한 NHRP_req가 도착하면(receive_NHRP_Forward 프로세스에서 수행) NHRP_req를 MPOA 캐쉬 저장 요청인 cache_imp_req로 바꾸어 (NHRP Releq_MPOACacheImpReq 프로세스에서 수행) 시그널 루트 R3을 통하여 Egress_MPC 블록으로 보낸다. 이 때의 cache_imp_req는 Egress MPC가 필요로하는 인캡슐레이션, 상태 유지 정보를 포함한다. NHRP 주소 해석 요청을 MPOA 캐쉬 저장 요청으로 바꾸는 과정은 프로세스 레벨에서 좀 더 자세히 살펴보겠다.

▪ 캐쉬 제거 프로토콜

MPOA 주소 해석 응답과 NHRP 주소 해석 응답, 그리고 성공적인 MPOA 캐쉬 저장 요청에 대한 상태 정보를 자신이 정한 holding time 동안 유지하는 Egress MPS는 유효한 캐쉬 엔트리에 대한 목적지 3계층 주소

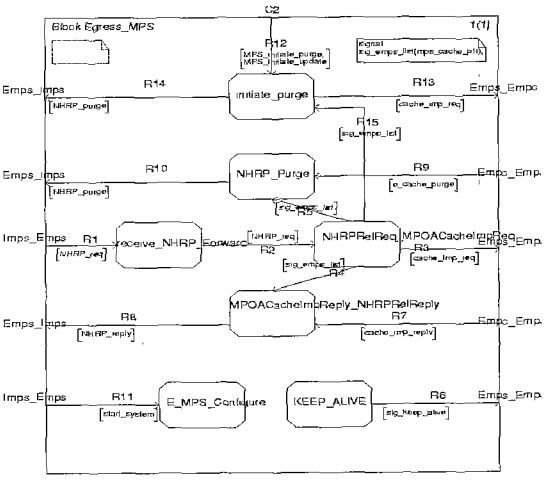

그림 8 블록 레벨 : Egress_MPS

변경을 감지하게 되면 그 엔트리를 수정하거나 제거하는 작업을 해야 한다(initiate_purge 프로세서에서 수행). 엔트리를 수정하기 위해서는 수정된 DLL 헤더를 가진 cache_imp_req를 해당 Egress_MPC 블록에게 전송한다. 엔트리를 제거할 경우에는 주소 해석 요청을 보냈던 근원지들에게 NHRP 제거 요청을 보내기 위해 R10을 통해 NHRP_purge 시그널을 Ingress_MPS 블록에게 보내고, Egress_MPC 블록에게는 holding time 값을 0으로 세팅한 뒤 이전에 사용했던 것과 같은 Egress 캐쉬 ID를 가진 cache_imp_req를 보낸다.

▪ Keep-Alive 프로토콜

Egress_MPC에게 캐쉬 엔트리를 부여한 MPS가 잘 작동중임을 나타내어 캐쉬 엔트리에 대한 신뢰성을 보장하기 위하여, Egress_MPS는 캐쉬 엔트리를 부여한 모든 Egress_MPC에게 MPOA Keep-Alive 메시지(sig_keep_alive 시그널)를 보낸다. 이 시그널은 일정 시간(MPS-p1)마다 보내어져야 하고, 이 시그널 안에 명시된 유효 시간인 Keep-Alive Lifetime(MPS-p2) 안에 다음 sig_keep_alive 시그널이 오는지 Egress_MPC는 감시해야 한다. (Egress_MPC 블록의 get_KEEP_ALIVE 프로세스).

(4) Egress_MPC 블록(그림 9)

cache_imp_req를 받은 Egress_MPC는 Egress 캐쉬 엔트리를 생성하기 위해 요청 시그널 내에 포함되어 있는 목적지 3계층 주소, 근원지 ATM 주소와 태그값을 키로 하여 기존에 있던 Egress 캐쉬 엔트리들과 비교한다(search_egress_entry 프로세스). 이미 존재하는 Egress 캐쉬 엔트리에 대한 존재 유무를 확인한 뒤, 캐

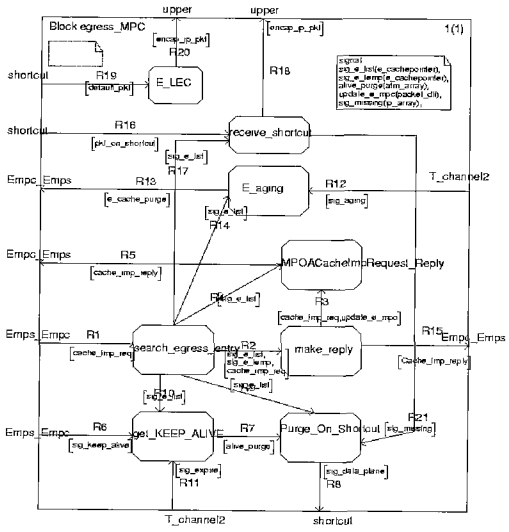

그림 9 블록 레벨 : Egress_MPC

쉬 엔트리가 없는 경우 make_reply 프로세스는 MPOA CacheImpRequest_Reply 프로세스에게 cache_imp_req 시그널을 포워드하고 다시 이 시그널은 MPOA 캐쉬 저장 응답인 cache_imp_reply로 변환되어 Egress_MPS에게 전송된다. 이미 캐쉬 엔트리가 존재하는 경우에는 요청 시그널 내에 있는 holding time에 따라 캐쉬 엔트리에 대한 갱신이나 삭제 여부가 결정된다. holding time이 0이 아닌 경우 해당 엔트리에 대한 갱신을 요청한 경우이므로 cache_imp_req 시그널을 통해 전달 된 수정된 DLL 헤더로 해당 엔트리를 갱신한다. holding time이 0인 캐쉬 저장 요청에 대해서는 캐쉬 엔트리를 삭제하는데, 이 단계에서는 물리적인 삭제는 일어나지 않고 캐쉬 엔트리 상태를 Invalid로 변화시킨다. 물리적 삭제는 추후에 Egress 캐쉬 엔트리를 Aging하는 E_aging 프로세스에서 일어나게 된다. Invalid인 캐쉬를 삭제할 때 E_aging 프로세스는 무효화된 캐쉬 엔트리를 부과한 Egress_MPS에게 MPOA 캐쉬 제거 요청 e_cache_purge 시그널을 전송해야 한다. e_cache_purge 시그널을 받은 Egress_MPS는 자신의 캐쉬 엔트리도 제거하고, Ingress MPC의 캐쉬 제거를 위하여 NHRP_purge 시그널을 전송한다(Egress_MPS 블럭의 NHRP_Purge 프로세스).

Purge_On_Shortcut 프로세스는 특정 상황에서 Egress_MPC가 Ingress_MPC에게 Egress 캐쉬 엔트리가 무효화되었음을 알리기 위해서 데이타 플레인으로 NHRP 제거 요청을 보낼 필요가 있을 때 동작한다. Egress_MPS가 더 이상 작동하지 않을 때, 즉 keep-alive lifetime 내에 Keep-alive 메시지인 sig_keep_alive 시그널을 받지 못했을 때(이 경우 get_KEEP_ALIVE 프로세스가 Purge_On_Shortcut 프로세스에게 alive_purge 시그널을 전송)와 해당 지름길로 MPC가 패킷을 받았으나 Egress 캐쉬에 없을 때(이 경우 receive_shortcut 프로세스가 Purge_On_Shortcut 프로세스에게 sig_missing 시그널을 전송), 이 두 가지 경우에 Egress_MPC는 Ingress_MPC에게 지름길 VCC인 shortcut 채널을 통해 NHRP 제거 요청에 해당하는 sig_data_plane 메시지를 보낸다.

4.3 프로세스 레벨

프로세스는 시스템의 동작이 실제로 명시되는 동적인 개체로서 다른 프로세스들과 시그널을 통해 정보를 상호 교류한다. SDL의 프로세스는 확장된 개념의 유한 오토마타라 할 수 있는데 각 프로세스는 4가지 주요부분 즉, 입력포트, 유한 오토마타, 타이머 그리고 변수 부분으로 구성되어 진다. 입력 포트는 들어오는 시그널을 무한대로 저장할 수 있는 큐를 가지고 있어 항상 새 신호를 받아들일 준비가 되어 있다. 프로세스에 도착한 시그널들은 도착한 순서대로 입력포트에 더해지며 만약 두 시그널이 동시에 도착했다면 시스템이 임의적으로 두 시그널의 순차적 순서를 결정한다. 입력포트에서 처리할 시그널이 없다면 프로세스는 시그널이 들어올 때까지 일정 상태에서 기다리고 있다. 시그널을 처리함으로써 유한 오토마타는 한 상태에서 다른 상태로의 전이가 가능하며 다음 상태로 전이할 동한 프로세스는 출력 시그널 발생, 변수에 대한 연산 처리, 타이머 동작 등을 수행할 수 있다. 상태 전이 동작은 프로세스 다이어그램으로 표현된다. 프로세스 다이어그램은 프로세스들이 모든 상태에서, 모든 가능한 입력에 대하여 어떻게 처리할 것인지를 상세화하는 프로세스 그래프이다. 각 상태는 전이를 유발하는 입력 기호를 먼저 명세한 뒤 다음 상태로 들어가기 전까지의 연속된 동작을 명세한다. 프로세스 다이어그램은 플로우 차트처럼 처리 작업에 따라 다양한 프로세스 기호를 제공하여 프로세스의 동작을 표현한다. 이상에서 살펴본 프로세스의 개념을 바탕으로 본 MPOA 시스템에서의 대표적인 기능을 수행하는 프로세스 2가지를 예로 살펴보기로 한다.

(1) 프로세스 : threshold_check(그림 10)

threshold_check는 상위 계층으로부터 받은 메시지에 대하여 캐쉬 엔트리를 생성한 뒤 캐쉬 엔트리가 임계값을 초과하면 지름길 설정을 위해 Ingres MPS에게 MPOA 주소 해석 요청을 하는 Ingress_MPC 블록 내

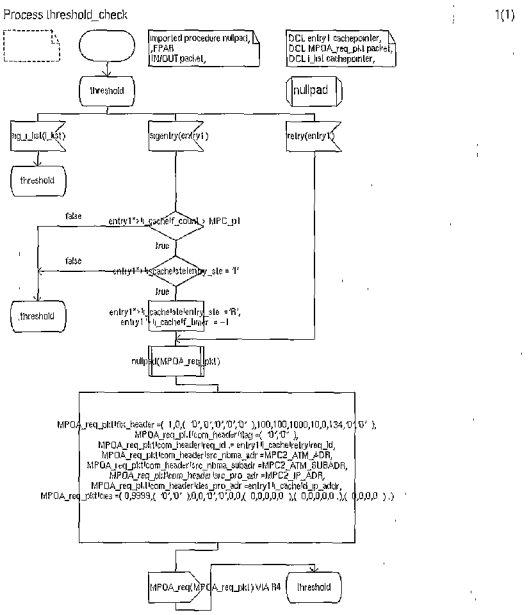

그림 10 프로세스 다이어 그램 : threshold_check

의 프로세스이다. 프로세스 다이어그램에서의 첫 번째 심볼은 시작기호이다. 시스템이 처음 시작할 때, 혹은 시스템 수행 중에 프로세스가 만들어지면 시작 전이가 수행된다. 따라서, 그림 12의 "threshold_check" 프로세스는 시스템이 처음 시작될 때, "threshold" 상태에서 시그널 입력을 기다리게 된다. 이 상태에서 다른 프로세스로부터 "sig_i_list", "sigentry", "retry" 시그널 중 sigentry 시그널을 받게 되면, 다음의 작업을 연속적으로 수행한다.

다른 프로세스에서 생성된 캐쉬 엔트리 entry1을 매개변수로 포함한 sigentry 시그 널을 전송한다. 캐쉬 엔트리의 내용 중 같은 목적지로 향하는 데이타의 수를 나타내는 f_count 필드 값과 상수로 정의된 임계치 MPC_p1을 비교하여 카운트 값이 임계 치를 넘고 이 엔트리의 상태가 Invalid 상태이면, entry1의 엔트리 상태를 "R" 즉 , reserved 상태로 변환시키고 필드 내 타이머 값을 음수 값으로 지정하여 아직 주소해석 응답을 받지 않은 캐쉬 엔트리를 에이징하는 것을 방지한 뒤, "nullpad" 프로시져를 호출한다. 이 프로시져는 MPOA_req 시그널을 만들 때 기본적으로 이 시그널의 전달인자인 MPOA_req_pkt에 들어가는 모든 값을 디폴트 값인 0으로 세팅하는 프로시져이다. 다음으로 MPOA 주소 해석 요청을 보내기 위해 목적지의 3계층 주소, 자신의 ATM 주소, 요청 ID, 플래그, 등을 삽입하여 MPOA_req_pkt 매개변수를 만든다. 그리고, 끝으로 MPOA_req_pkt를 매개변수로 가지는 MPOA_req 시그널을 R4 시그널 루트를 통해 다른 블록내의 프로세스에게 전송한다.

시그널 전송 작업이 끝나면 "threshold" 상태가 되므로 다시 다이어그램의 처음 상태로 돌아가서 3개의 시그널 중 하나가 입력되기를 기다린다. 기대하는 시간 내에 MPOA 주소 해석 응답을 받지 못한 Retry 프로세스로부터 retry 시그널을 입력받았을 경우에는 임계치를 검사할 필요없이 바로 MPOA_req_pkt 전달인자를 만들어 MPOA_req 시그널을 전송한다.

이 프로세스 다이어그램에서 사용되는 변수는 텍스트 심볼 내에서 "DCL"이라는 키워드를 이용하여, 사용하고자 하는 변수명, 시스템 레벨에서 정의된 변수형 순서로 선언된다. 이 때 변수형은 정수, 실수 등 시스템이 지원하는 변수형 외에도 시스템 레벨에서 "NEWTYPE" 키워드로 사용자가 정의하였던 변수형을 사용할 수도 있다. 프로세스 다이어그램에서 입력 신호의 근원지는 정의될 필요가 없는 데 반해 출력 신호의 목적지는 출력 심볼 내 혹은 텍스트 확장 심볼 내에서 반드시 정의되어야 한다. 이 때 목적지는 "Via 절" 혹은 "To 절"을 이용하여 지정되며 "Via 절"은 목적지로 갈 시그널 루트를 지정할 때 사용되고 "To 절"은 목적지 프로세스 이름을 지정할 때 사용된다.

(2) 프로세스 : NHRPRelReq_MPOACacheImpReq (그림 11)

NHRPRelReq_MPOACacheImpReq 프로세스는 Egress_MPS 블록 내에서 수행되는 프로세스로서 다른 MPS가 보낸 NHRP 주소 해석 요청이 자신이 서브하는 목적지인 경우 MPOA 캐쉬 저장 요청으로 변환하여 Egress_MPC 블록에게 전송하는 작업을 수행한다. NHRP 주소 해석 요청인 NHRP_req_pkt 매개변수를 포함한 NHRP_req 시그널을 입 력받으면 NHRP_req 시그널을 통해서 전달되는 정보가 이미 이전의 패킷에 의해 Egress MPS 캐쉬 엔트리에 내에 저장되어 있는지 확인한다. 이전에 생성된 캐쉬 엔트리가 없는 경우 Save_In_EMPS_Cache 프로시져를 호출하여 NHRP_req에 대한 새로운 캐쉬 엔트리를 만들어 저장한다. 여기에서 생성된 캐쉬 엔트리는 후에 NHRP 주소 해석 요청에 대한 응답을 위해 사용된다. 라우터 포워딩 테이블을 조사하여 3계층 목적지 주소가 자신이 알고 있는 LEC들 중 하나인지 알아본다. 목적지 주소와 연관된

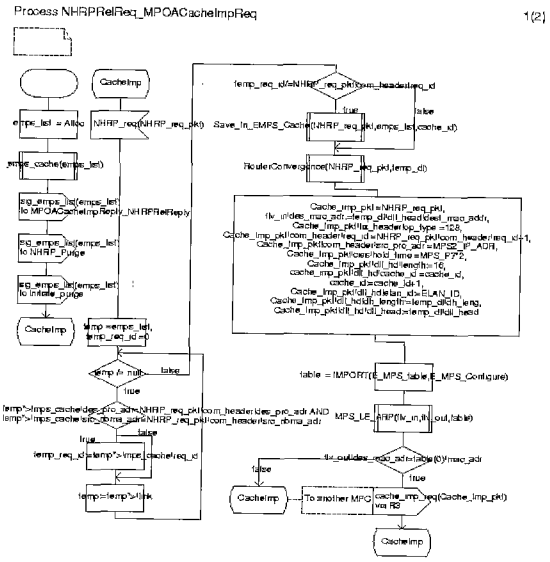

그림 11 프로세스 다이어그램 : NHRPRelReq_MPOA CacheImpReq

LEC를 찾았다면 RouterConvergence 프로시져를 통해 temp_dl 파라미터를 리턴 받아 목적지로 보낼 프레임의 DLL 헤더를 결정한다. NHRP_req_pkt 전달인자를 바탕으로하여 Cache_imp_pkt 전달인자를 가지는 MPOA 캐쉬 저장 요청 패킷을 만든다. temp_dl 파라미터를 통해 전달 받은 DLL 헤더 안에 담긴 목적지 MAC 주소를 가지고 LE_ARP 제어 프레임에 포함된 장치형 TLV를 통해 LEC로부터 MPC의 ATM 주소를 알아낸다. MPOA 캐쉬 저장 요청 패킷인 cache_imp_req 시그널을 R3를 통해 Egress_MPC 블록에게 전송한다.

5. 시나리오를 통한 검증

5장에서는 SDL 언어로 구현한 MPOA 시스템이 올바르게 동작하는지 검증하고 그 동작 방식을 살펴보기 위해 ATM 포럼에서 표준화한 기능을 모두 포함하도록 조합한 4가지 시나리오를 정의하고 SDT에서 제공하는 시뮬레이션 도구로 시뮬레이션한 결과를 보여준다. 본 구현에서는 MPOA의 주요 기능과 MPOA 구성요소 간의 동작을 모두 포함하는 4가지 시나리오를 정의하였으며 시나리오에 따른 진행 상태를 표준안과 비교하여 올바른지 검사하였다. 본 논문에서는 SDT에서 제공하는 MSC (Message Sequence Chart)를 이용해 시나리오별 진행 과정을 직관적으로 보여준다. MSC는 SDL 시스템의 동적인 움직임을 명확히 보여주며, 타이머 객체를 두어 시간을 고려하여 SDL 시스템의 시뮬레이션과 무결성 검증을 위해 사용되는 언어이다. MSC는 각 레벨별로 객체들의 통신 상태를 보여줄 수 있으며 본 논문에서는 결과를 전체적으로 보여주기 위해 블록 객체들 사이의 통신을 결과 자료로 첨부하였다.

첫 번째 시나리오에서는 MPOA 시스템의 가장 주된 기능인 3계층 주소 해석에 의한 지름길 설정 과정을, 두 번째 시나리오에서는 MPS로부터 Keep-alive 메시지 전송을 받지 못하였을 때 시작되는 데이타 플레인 제거 과정을 보여준다. 세 번째 시나리오에서는 Egress MPS에서 시작하는 캐쉬 엔트리 갱신과 제거, Egress MPC 타이머에 의한 캐쉬 엔트리 에이징의 경우를 보여준다. 마지막 시나리오에서는 액티브 캐쉬 엔트리에 대한 refresh 메커니즘을 시뮬레이션하였다. 이 구현 시나리오를 테스트하기 위한 가상 MPOA 시스템은 두 개의 ELAN을 가정하였고 각 ELAN마다 MPS와 MPC 호스트, MPC 엣지 디바이스를 두어 시뮬레이션에 필요한 시스템을 구성하였다. 그림 12에서 이 MPOA 시스템의 구성도를 보여준다.

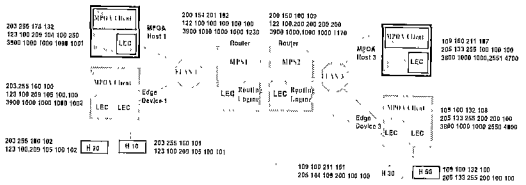

그림 12 구현 시나리오에서 사용된 MPOA 시스템 구성도

5.1 시나리오 1 : 주소 해석에 의한 지름길 설정

(1) 임계치을 넘겨 지름길을 생성하는 경우(그림 13)

Host10의 상위계층인 외부환경으로부터 같은 목적지인 MPOA Host3으로 향하는 IP 패킷이 일정 시간 내에 임계치 이상 들어오면 Edge Device1이 Ingress MPC가 되어 지름길 설정을 위한 MPOA 주소 해석 요청 MPOA_req 시그널을 Ingress MPS인 MPS1에게 전송한다. 그림 15에서 시그널과 함께 괄호 내에 표기된 것은 파라미터를 표시하며, 지면 관계상 실제 MPOA 시스템에서 사용되는 제어 메시지 내의 필드 중 중요한 몇 가지 특정 필드즉, Common 헤더 내의 요청 ID, 근원지 NBMA 주소, 근원지 프로토콜 주소, 목적지 프로토콜 주소만 명시하였다. MPOA_req 시그널을 받은 MPS는 새로운 요청 ID를 만들어 다른 MPS에게 NHRP 주소 해석 요청 NHRP_req 시그널을 보낸다. 이 시그널을 받은 MPS는 자신이 관리하는 목적지로의

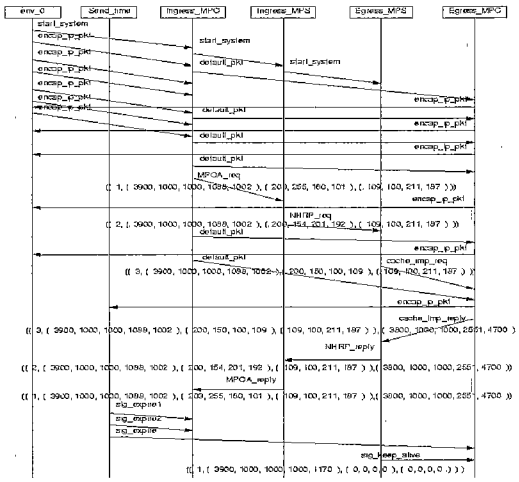

그림 13 임계치를 넘겨 지름길을 생성

주소 해석 요청이므로 MPOA 캐쉬 저장 요청인 cache_imp_req 시그널을 Egress_MPC에게 전송한다. Egress_MPC는 cache_imp_req 시그널 내의 매개 변수로 온 정보들을 자신의 캐쉬 엔트리에 저장한 뒤 MPOA Host3과의 지름길 설정을 위해 필요한 MPC2의 ATM 주소를 포함하는 cache_imp_reply 시그널을 Egress_MPS에게 응답한다. 이 응답을 NHRP_reply 시그널로 바꾸고 다시 MPOA_reply 시그널로 바꾸어 MPC1인 Ingress_MPC에게 전송하며(응답 전송 시에는 시그널 파라미터 중 추가적으로 Egress MPC의 NBMA 주소를 명시하였다) 응답 시에는 요청 할 때와 같은 요청 ID로 바꾸어 주게 된다. 응답을 받게 된 Ingress_MPC는 응답 시그널 파라미터에 포함된 MPC2의 ATM 주소와 그 외 정보를 캐쉬 엔트리에 저장하고 MPOA Host3과의 지름길 VCC를 설정하게 된다. 주소 해석 과정이 수행되는 동안의 데이타 패킷은 default_pkt 시그널 형태로 디폴트 전송된다. 라우터와 LANE을 통한 디폴트 전송은 본 구현 범위에서 제외되어 Ingress MPC와 함께 있는 LEC에게 포워드하고 Egress MPC에서 다시 LEC로부터 받는 과정만 보여주었다. 그림 13은 주소 해석 과정을 시뮬레이션 했을 때 SDT 시뮬레이터에서 보여주는 MSC(Message Sequence Chart)로 MPOA Resolution Reqest, NHRP Resolution Request, MPOA Cache Imposition Request, MPOA Cache Imposition Reply, NGRP Resolution Reply, MPOA Resolution Reply가 연쇄적으로 발생하여 성공적으로 VCC가 설정되는 과정을 보여준다. 또한 Egress_MPS가 Egress_MPC에게 캐쉬를 부여한 뒤에는 자신이 부여한 캐쉬 엔트리의 유효성을 보장하기 위해 주기적으로 sig_keep_alive 시그널(파라미터 중 Sequence Number인 요청 ID, 근원지 NBMA 주소, 근원지 프로토콜 주소, 목적지 프로토콜 주소를 명시)을 전송하여 MPS가 잘 작동중임을 알려준다.

(2) 트리거에 의해 지름길을 생성하는 경우(그림 14)

MPS1이 디폴트 전송되는 데이타 흐름을 보고 Host10을 담당하는 Edge Device 1로부터 MPOA Host 3으로의 지름길 전송이 유리하다고 판단하고(외부 환경으로부터 MPOA Host IP 주소를 매개 인자로 가지는 sig_trigger 시그널을 전송 받음) 트리거하면 (Ingress_MPC에게 trigger_MPC 시그널 전송, 이 때 파라미터로 요청 ID, 근원지인 MPS1의 NBMA 주소, 클라이언트 프로토콜 주소를 표기) Ingress_MPC는 임계치에 상관없이 MPOA 주소 해석 요청을 시작한다. 그 이후의 과정은 앞의 (1)의 과정과 동일하다.

그림 14는 ingress MPS에서 발생한 트리거 요청에 MPOA Resolution 프로세스가 수행되므로써 성공적으로 VCC가 설정되는 과정을 보여준다.

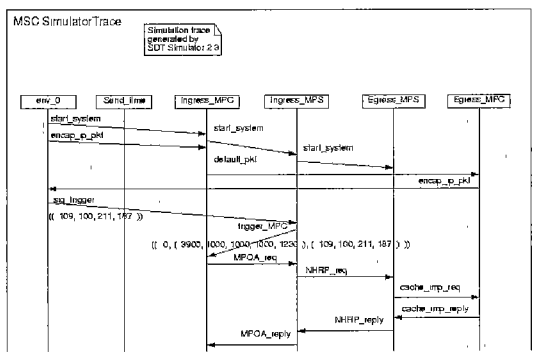

그림 14 트리거에 의한 지름길 생성

5.2 시나리오 2 : 데이타 플레인 제거 (그림 15)

시나리오 1에서와 같이 캐쉬 엔트리가 생성된 후에 일정 시간마다 Egress MPS로부터Keep-Alive 메시지를 이전 메시지를 통해 설정된 메시지 유효 시간 내에 받지 못하거나, 받은 Keep-Alive 메시지 내에 있는 요청 ID가 이전 메시지의 요청 ID와 연속된 숫자가 아닐 때 Egress MPS에게 문제가 생겼다고 가정하고 Egress MPC는 MPS에 의해서 부과되었던 캐쉬 엔트리를 무효화시킨다. 또한 지름길 VC를 통하여 Ingress_MPC에게 캐쉬 제거 요청sig_data_plane 시그널을 보내어 대

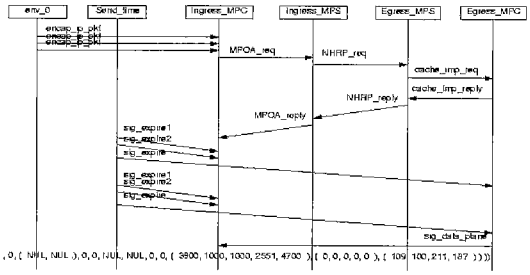

그림 15 데이타 플레인 제거

응되는 캐쉬 엔트리를 무효화시킨다.

그림 15는 Egress MPS로 부터 Keep-Alive 메시지를 받지 못한 Egress MPC가 Ingress MPC에게 직접 캐쉬 제거 요청을 보냄으로써 해당 캐쉬 엔트리를 무효화 시키는 과정을 보여준다.

5.3 시나리오 3 : Egress 캐쉬 엔트리의 갱신, 제거, Aging

(1) Egress MPS가 초기화는 캐쉬 엔트리의 갱신(그림 16)

호스트의 이동이나 라우팅 정보의 변경으로 기존 캐쉬 엔트리의 변경이 요구될 때, 변경된 정보를 가진 MPOA 캐쉬 저장 요청 cache_imp_req를 Egress MPC인 MPOA Host 3에게 전송하여 캐쉬 엔트리를 갱신한다. 그림 16은 MPS_initiate_update 시그널을 외부 환경으로부터 전송 받은 Egress_MPS가 이전에 그 엔트리를 부여할 때 사용한 목적지 3계층 주소, 근원지 ATM 주소, 태그의 키 값과 새로운 DLL 헤더, 0이 아닌 유효 시간을 담은 Cache Imposition Reqest를 보내 Egress_MPC 캐쉬 엔트리 갱신하고 성공적으로 Cache

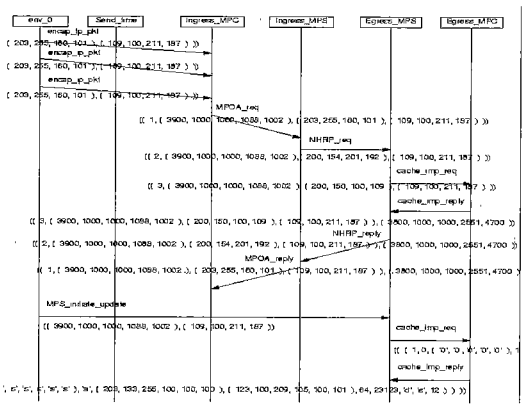

그림 16 Egress MPS에 의한 캐쉬 엔트리 갱신

Imposition Reply를 받는 과정을 보여준다.

(2) Egress MPS가 초기화는 캐쉬 엔트리의 삭제와 Aging(그림 17)

위의 상황과 마찬가지로 토폴로지가 변경되어 기존 엔트리가 무효화되었을 때, Egress MPC에게 0인 유효 시간을 갖는 MPOA 캐쉬 저장 요청을 보내어 캐쉬를 삭제하게 한다. 캐쉬 엔트리가 무효화 된 후에는 일정 시간마다 Send_Time 블록에 의해서 보내어 지는 sig_aging 시그널에 의해서 무효화 된 캐쉬 엔트리는 물리적으로 삭제된다.

그림 17은 MPS_initiate_purge 시그널을 외부환경으로부터 전송 받은 Egress_MPS (MPS2)가 0인 유효 시간을 갖는 Cache Impostion Request를 Egress_MPC에게 보내고, 그와 동시에 Ingress_MPS(MPS1)와 Ingress_MPC(Edge Device 1)에게는 NHRP 캐쉬 제거 요청을 보내어 연관되어 있는 캐쉬 엔트리를 무효화 시키고 Aging에 의해 Ingress_MPC와 Egress_MPC의 캐쉬 엔트리가 삭제되는 과정을 보여준다. 또한 그림 17의 마지막 시그널은 Egress MPC의 캐쉬 엔트리가 삭제되면 이에 대응되는 Ingress MPC의 캐쉬 엔트리도 삭제하기 위해서 Egress MPC는 MPC-Initiated Egress Cache Purge Request를 Egress MPS에게 보내지만 이미 NHRP_purge를 통해 자신이 무효화시킨 캐쉬 엔트리에 대한 제거 요청임을 알고 더 이상 Ingress MPC에게 포워드하지 않는 것을 보여준다.

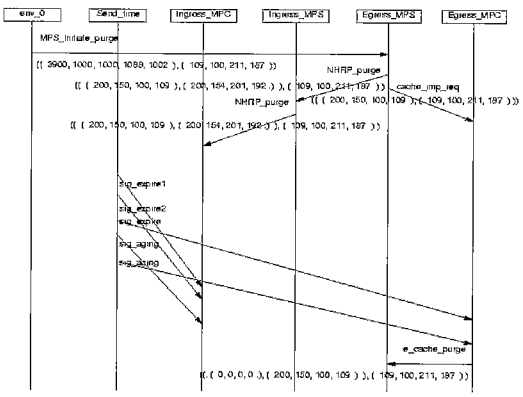

그림 17 Egress MPS에 의한 캐쉬 엔트리 삭제와 Aging

5.4 시나리오 4 : 캐쉬 엔트리의 Refresh(그림 18,19)

캐쉬 엔트리가 생성된 이후에 계속 사용 중인 액티브 캐쉬 엔트리가 aging 되어 캐쉬에서 제거되는 것을 막기 위해 Egress MPC인 Edge Device 1은 holding

time이 초과되기 전에 액티브 캐쉬 엔트리과 같은 3계층 목적지 주소에 대한 MPOA 주소 해석 요청을 시도한다. 본 구현에서는 sig_expire 1과 2를 통해 aging하고 있는 액티브 캐쉬 엔트리의 holding time이 2/3가 초과되면 새로운 요청 ID를 담은 MPOA_req를 보내도록 구현하였다.

그림 18과 19는 이미 VCC가 설정된 엔트리에 대해 일정 시간후 다시 MPOA Resolution Process가 수행되어 엔트리를 액티브 상태로 유지하는 과정을 보여준다.

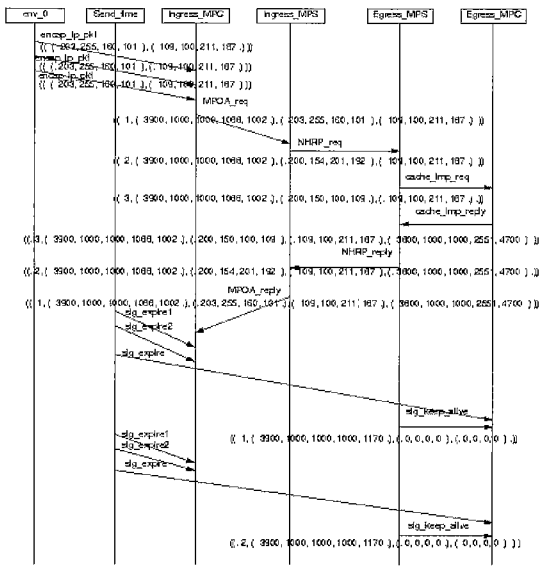

그림 18 액티브 캐쉬 엔트리에 대한 refresh (I)

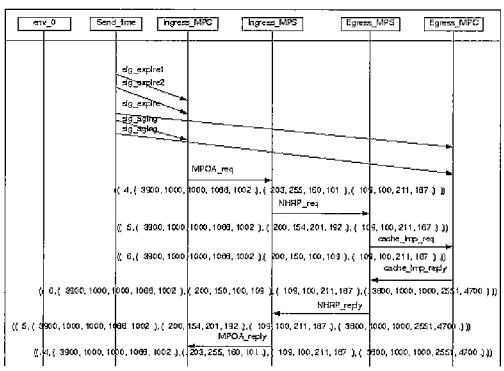

그림 19 액티브 캐쉬 엔트리에 대한 refresh (II)

6. 결 론

본 논문에서는 다양한 네트워크 환경에서 ATM 네트워크에게 효과적인 브릿징과 라우팅을 제공하는 프로토콜로서, 서브넷 간에 효율적인 유니캐스트 데이타 전송을 주된 목적으로 하고 있는 MPOA를 구현하였다. 구현에 있어서는 통신 시스템 개발용 명세 언어인 SDL을 이용하였으며 Ingress/Egress MPC와 Ingress/Egress MPS를 구현하였다. 상위 계층에서 받은 패킷에 대해 주소 해석을 통한 지름길 전송 또는 디폴트 전송, Ingress/Egress 캐쉬 엔트리의 갱신, 삭제와 에이징, 그리고 데이타 플래인 제거에 의한 캐쉬 관리를 구현하였으며 작동상황을 테스트하기 위하여 시나리오를 구성하고 이를 SDL의 시뮬레이션을 통해 검증하였다.

SDL을 이용하여 프로토콜을 구현하여 구현내역을 명확하게 표시할 수 있었고, 명시한 사실이 바뀌더라도 간편하게 변경 내역을 수정할 수 있었으며 계층 구조를 그래픽으로 표현함으로써 타 언어로 구현하는 것보다 쉽게 이해되며, 구현된 시스템에 대한 단계적인 분석이 가능하였다. 또한 시뮬레이션과 검증을 위한 시뮬레이터와 검증기를 SDT내에 포함하고 있어 별도로 진행 과정을 표시하는 기능을 추가하지 않아도 시뮬레이션 및 검증이 용이하다. 따라서 본 논문은 MPOA를 구현하려는 ATM 스위치 개발에 도움이 될 것이다. 또한 향후 계획으로는 실제 ATM 스위치에 구현한 MPOA를 탑재하고 구현한 MPOA를 이용한 3계층 가상랜을 개발하고자 한다.

참 고 문 헌

[1] 김용진, "ATM 공중망에서의 IP 서비스 제공방안", Telecommunications Review 제8권 6호, 11-12월 1998.

[2] The ATM Forum, "Multiprotocol Over ATM Version 1.0 (Letter Ballot)," May 1997.

[3] Hao Che, San-qi Li, "MPOA Flow Classification Design and Analysis," *IEEE INFOCOM '99*, pp. 1497-1504 , Mar. 1999.

[4] Indra Widjaja, Haining Wang, Steve Wright, Amalendu Chatterjee, "Salability Evaluation of Multi-Protocol Over ATM," *IEEE INFOCOM '99*, pp. 1505-1512, Mar. 1999.

[5] Rolv Braek, "SDL Basic," Computer Networks and ISDN System 28, 1996.

[6] D J Newson, D Ginsburg and M T Wilkins, "Next Generation Local Area Networks," BT Tech. Journal Vol. 16, Jan. 1998.

[7] Telelogic SDT 3.2, "SDT Getting Started Part 1 : Tutorials on SDT Tools," Sep. 1997.
[8] James V. Luciani, Dave Katz, David Pscitello, Bruce Cole, "NBMA Next Hop Resolution Protocol (NHRP)," INTERNET-DRAFT <draft-ietf-rolc-nhrp-11.txt>, Sep. 1997.

임 지 영
1994년 1월 이화여자 대학교 전자계산학과 학사. 1996년 2월 이화여자 대학교 전자계산학과 석사. 1996년 ~ 현재 이화여자대학교 컴퓨터학과 박사과정

김 희 정
1998년 2월 이화여자 대학교 컴퓨터학과 학사. 2000년 2월 이화여자 대학교 컴퓨터학과 석사. 2000년 ~ 현재 LG 전자 기술원 정보기술연구소

임 수 정
1998년 2월 이화여자 대학교 컴퓨터학과 학사. 1998년 ~ 현재 이화여자 대학교 컴퓨터학과 석사과정.

채 기 준
정보과학회논문지 : 정보통신
제 27 권 제 2 호 참조

이 미 정
정보과학회논문지 : 정보통신
제 27 권 제 3 호 참조

최 길 영
1985년 2월 경북대학교 전자공학과 학사. 1987년 2월 경북대학교 전자공학과 석사. 1987년 2월 ~ 현재 한국전자통신연구원 교환전송기술연구소. 라우터기술연구부 ATM LAN팀 선임연구원

강 훈
1980년 2월 연세대학교 전자공학과 학사. 1989년 8월 Iowa State University 공학석사. 1993년 8월 Iowa State University 공학박사. 1993년 ~ 1999년 한국전자통신연구원 ATM LAN팀장. 2000년 ~ 현재 한국 아이티 벤처 투자 (주) 상무이사.