TINA 연결관리 패키지의 연결 설정 및 해제 기능 구현

박 준 희[†] · 오 현 주[††] · 김 상 하[†††]

요 약

TINA는 분산처리환경상에서 객체단위로서 망 관리객체와 서비스 관리객체를 공유하는 새로운 개념의 소프트웨어 계층적 개방형 통신망 구조이다. 본 논문에서는 TINA의 관리구조에서 나누고있는 6개의 기능적 영역 중 하나인 연결관리의 연결관리 패키지를, 현재까지 정의되고 규격화된 개념과 원리에 의거해서, 구현하고 시험해 보았다. 구현에 앞서서 TINA의 네가지 구조와 각각의 구조에서 수용하고있는 기본개념, 연결관리의 모데링 개념 등에 대해서 알아본다. 그리고, 구현된 각 연산객체의 기능을 정보객체들의 인터페이스를 통해 설명하고, 연결 설정 및 해제 과정을 연산객체간의 메시지 흐름을 통해 보인다.

Implementation of TINA CM Package Functionning as Connection Setup and Release

Jun-Hee Park[†] · Hyun-Ju Oh[††] · Sang-Ha Kim[†††]

ABSTRACT

TINA is the Open Networking Architecture which as newly introduced software architecture shares the network managements and the service managements on the Distributed Processing Envronment. In this paper, based on the concepts and principles defined up to now, we implement and test the CM(Connection Management) Package included in CM which is one of the six functional areas into which TINA Management Architecture is devided. Before the implementation we learn the basic concepts accepted by the four architectures in TINA, and medelling concept of the CM. The interfaces of eaxh information objects explain the functions of computational objects implemented, and the message flows among the computational objects show the connection setup and release procedures.

1. 서 론

초고속 통신망 기술이 발전함에 따라 멀티미디어 등의 통신 서비스에 대한 사용자의 요구가 날로 증대되고 있다. 또한, 통신 시장의 개방에 따른 통신 사업자의 자유화에 의해 통신망은 점점 복잡해지고 있다. 통신망이 복잡해 질수록 망의 관리 및 새로운 서비스의 도입이 어려워지게 되므로, 망의 원활한 관리와 운용을 위해 이질적인 통신망을 연동할 수 있게하는 일관성있는 통신망 구조가 필요하게 되었다[1].

이러한 추세에 따라 분산 컴퓨팅 개념, 객체 지향 개념 등 기존의 통신 및 컴퓨팅 개념들을 수용하는 개방형 정보 통신망 구조를 위한 컨소시움인 TINA-C

※이 논문은 ETRI에서 충남대학교에 위탁한 연구 결과 중 하나임
† 정 회 원:충남대학교 컴퓨터과학과
†† 정 회 원:충남대학교 컴퓨터과학과
††† 종신회원:충남대학교 컴퓨터과학과
논문접수:1997년 2월 28일, 심사완료:1997년 8월 12일

(Telecommunication Information Networking Architecture-Consortium)가 발족되었다.

TINA-C에서는 통신망에서 폭증하게될 통신 소프트웨어의 개발을 위한 일관된 모델링 방법, 그리고 망 관리에 관련된 구조적 문제의 해결을 위한 개방형 정보 네트워킹의 구조 설계 및 구축, 운용등에 대한 기본적인 개념과 원칙을 제시한다[2, 4].

TINA의 목적은 통신 서비스의 설계/개발/유지보수의 용이성을 위한 논리적 기본 구조 원칙 규정, 멀티미디어/동시성/멀티세션/다중연결의 통신 서비스 지원, 가입자에 의한 통신망 자원의 제어 등에 있다.

TINA-C에서는 상기 목적을 달성하기 위해서 통신 소프트웨어의 설계와 구현, 운용에 적용되는 TINA 구조를 정의하고 있다. TINA구조는 통신 소프트웨어와 전달망의 설계/구현/관리, 그리고 통신자원의 관리에 적용되는 개념과 원리를 정의하기 위한 일반적 구조와 소프트웨어 분리원칙과 계층화를 정의한 계층적 구조로 나누어진다.

TINA 구조는 ODP (Open Distributed Pocessing)의 모델링 개념, OSI Management의 Manager-Agent Paradigm, TMN (Telecommunication Management Network)의 레이어 개념, 그리고 ITU-T 권고안 G.803과 M.3100의 계층화와 분리화 개념 등 기존의 응용 서비스 개발과 망 관리를 위한 개념들을 따르고 있다.

TINA연결관리 구조는 계층적 구조에서 관리계층에 속하고, 일반적 구조에서는 관리 구조에 속하며, 망 구조에서 정의된 망 자원들(NRIM)을 바탕으로 TINA의 응용서비스를 지원하기 위한 기본 원리와 법칙을 정의한다.

TINA연결관리는 연결관리 구조에서 정의한 연결설정, 유지, 해제 절차를 수행하는 일련의 기능들의 집합이다. TINA연결관리 서비스를 위해서 연결관리 구조에서는 계층적으로 컴포넌트를 구성을 하고 있다.

TINA연결관리는 크게 통신 세션부분과 연결관리 패키지 부분으로 나눌 수 있다. 통신 세션부분은 응용서비스의 코아(Core)부분으로 부터 연결설정 요구를 받아서 실제로 망을 관리하는 연결관리 패키지로 넘겨주는 인터페이스 역할을 한다. 연결관리 패키지는 NRIM에서 정의한 객체를 가지고 G.803과 M.3100에서 정의된 개념으로 계층화 및 분리화된 망에서의 연결 설정/유지/해제 기능을 수행한다.

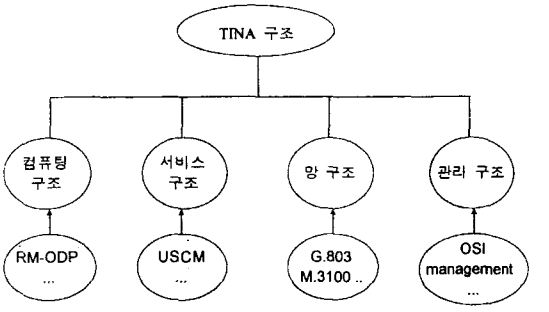
본 논문에서는 TINA연결관리 규격에 준한 연결관리 패키지를 구현하고, 이를 물리적인 망에 적용해 본다.

다음 장인 제 2장에서는 TINA에서 도입하고 있는 기본적인 개념에 대해서 자세히 알아보고, 제 3장에서는 TINA 연결관리 패키지의 기능과 모델링에 대해서 알아보며, 제 4장에서는 연결관리 패키지를 직접 구현, 적용하고, 제 5장에서 끝을 맺는다.

2. TINA의 기본개념

TINA-C는 개방형 정보 통신망 구조에서의 통신 응용 소프트웨어 개발과 자원 관리를 효율적으로 하기위해 필요한 개념과 원리를 정의하고 제공하기 위해 발족된 컨소시움이다.

TINA-C에서는 통신 소프트웨어의 설계, 구축, 동작등에 필요한 개념과 원칙을 제공하기 위해 TINA 구조를 정의하고 이에 따라 연구를 진행하고 있다. TINA구조는 이러한 개념과 원칙들을 공급하기위해 컴퓨팅 구조, 서비스 구조, 망 구조, 관리 구조의 네가지의 구조로 분할된다. TINA구조에는 TMN, INA, RM-ODP, OSI, OMG 등등, 여러 기구들에서 정의하고 있는 개념들을 포괄적으로 수용하고 있다[1, 2, 4]. (그림 1)은 TINA의 네가지 구조와 수용된 개념에 대해서 간략히 보여주고 있다.



(그림 1) TINA구조와 수용된 개념들
(Fig. 1) TINA Architecture and Concepts

컴퓨팅 구조는 소프트웨어 컴포넌트(연산객체)들 간의 상호작용과 통신방법 등에 대해서 개념과 법칙을 정의한다. 서비스 구조는 사용자에게 제공되고자

하는 서비스와 관련된 소프트웨어 컴포넌트들의 분석, 설계, 동작, 재사용성 등에관한 개념과 원리들을 정의하고, 제공한다. 망 구조는 ITU-T의 G.803과 M.3100을 바탕으로한 추상적인 관점을 이용해서 모든 망 자원들을 일반적인 컴포넌트(객체)로 표현하므로서 이들의 기능과 속성을 정의한다. 관리 구조는 상기한 구조들에 의해 정의된 모든 컴포넌트들의 관리를 위해 필요한 일반적인 관리 개념과 원리를 제공한다.

상기한 네가지 구조의 내용과 포함하고 있는 개념들에 대해서 자세히 알아보면 다음과 같다.

2.1 컴퓨팅 구조

컴퓨팅 구조는 RM-ODP(Refernce Model for Open Distributed Processing) 개념을 근간으로TINA시스템에서 객체지향 소프트웨어를 구축하는데 필요한 모델링 개념을 정의한다. 또한, 객체들을 위치 시키고, 상호간의 통신등을 지원하므로서 분산 투명성을 제공하는 DPE(Distributed Precessing Environment)를 정의한다.

TINA 컴퓨팅 구조의 모델링 단계는 정보모델링, 연산모델링, 그리고 공학모델링으로 구분된다. 정보모델링에서는 정보를 갖는 개체(Entity)들을 객체(Object)로 정의하고, 이들 간의 관계성과 생성 및 소멸을 포함한 객체들의 모든 행동을 지배할 수 있는 제약과 법칙을 정의한다. 정보모델링 결과를 표현하는 표기법으로는 Quasi GDMO-GRM이 사용된다. 연산모델링에서는 서로 상호작용하는 분산 통신 응용체들을 연산개체(Computational Entity)로서 나타낸다. 오퍼레이셔널(Operational) 인터페이스와 스트림(Stream) 인터페이스로써 연산개체 상호간의 관계를 표현하고, 연산개체의 서비스 및 관리 인터페이스를 정의한다. 이러한 인터페이스를 표현하기 위해 OMG IDL (Object Management Group Interface Definition Language)를 확장한 TINA ODL(Object Definition Language)를 사용한다. 공학모델링은 연산모델링을 통해서 표현된 객체들의 망 하부구조 상에서 어떻게 구성하고, 배치할 것인가를 결정한다. 또한, 분산 투명성을 지원하기 위한 객체들의 기능을 정의한다.

2.2 서비스 구조

서비스 구조는 TINA에서의 서비스와 이러한 서비스를 지원하는 환경을 구축하는 방법을 제시한다. TINA에서의 서비스란 통신 서비스, 관리 서비스, 종단 사용자 서비스등 넓은 부분을 포함한다.

정보모델링 관점에서의 서비스 구조는 서비스의 주 기능과 어떻게 관리되는가 등에대한 틀(framework)을 제시한다. 연산모델링 관점에서는 서비스의 기능을 사용자에게 공급하기 위해서 분산 서비스를 구성하는 방법에 대해서 기술한다. 또한, USCM(Universal Service Component Model)을 통한 서비스 모델링을 이용해서, 모든 서비스에서 기본적으로 가지고 있어야할 서비스 컴포넌트로서 UA(User Agent), TA(Terminal Agent), SSM(Service Session Manager), Subscription Manager, CSM(Communication Session Manager)를 정의하고 있다.

2.3 망 구조

망 구조의 목적은 전달방식등의 기술적인 부분에 무관하게 전달망을 일련의 일반적인 개념들로 표현하고, 그 기능 및 속성을 정의하는 것이다. 망 구조에서는 ITU-T의 G.803과 M.3100을 바탕으로 전달망의 자원들을 계층화 및 분리화하고, 이들을 잘 정의된 객체들로서 표현한다. NRIM(Network Resource Information Model)은 이러한 기본 개념을 바탕으로 전달 및 스위치 기술을 정보 규격화한 것으로서, 각각의 망 구성요소들이 어떻게 서로 연관되고, 위상적으로 연결되있으며, 망을 형성하고 있는가를 보여준다.

2.4 관리 구조

관리 구조는 TINA시스템의 개체들을 관리하기 위한 원리를 정의한다. TINA시스템의 관리에는 컴퓨팅 관리와 통신 관리의 두가지 종류가 있다.

컴퓨팅 관리는 연산기와 플랫폼, 그리고 플랫폼에서 실행되는 소프트웨어들에 대한 관리를 포함한다. 이것은 응용체에 따라서 관리가 이루어지므로, 관리 구조에서는 소프트웨어의 설치와 배치, 로드 발란스 등에 대해서만 고려하고 있다.

통신 관리는 TINA에서의 응용체, 즉 전달망을 관리 및 제어하는 소프트웨어와 통신 소프트웨어들과 관련된 응용체들의 컴포넌트를 관리한다. 통신 관리는 TMN(Telecommunication Management Network)

의 개념에 따라 서비스 관리, 망 관리, 망요소 관리로 구분된다.

일반적인 관리 개념에는 두가지 원칙이 있다. 하나는 관리 문제를 여러개의 구분된 영역으로 기능적 분류를 하는 것이다. TINA에서는 OSI의 System Management에서 정의한 5개의 기능적 영역을 바탕으로 6개의 기능적 영역을 정의하고 있다(그림 2). TINA에서는 TINA의 논리 틀 구조에서 정립한 객체 모델링 방법을 이용하고 있다.

Accounting Management	Connection Management	Fault Management	Performance Management	Resource Management	Security Management

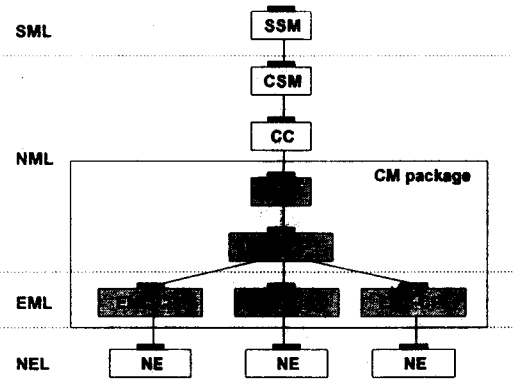
(그림 2) TINA 관리 기능 영역
(Fig. 2) TINA Management Functional Areas

3. TINA의 연결관리

TINA연결관리는 관리구조에서 정의된 6개의 기능적 영역 중 하나로서 다른 기능들과 함께 TINA 통신 서비스를 지원한다. TINA-C에서는 6개의 기능들을 하나하나 정의 및 규격화 해나가고 있다. 연결관리는 모든 통신 서비스의 기본이 되는 망에서의 연결 설정, 유지, 해제 기능을 수행하는 부분이므로 현재 핵심적인 부분은 규격화 작업을 거의 마친 상태이다.

TINA에서 연결관리기능의 의무는 분산 응용체 간의 통신 서비스를 지원하기 위한 연결 설정의 요구를 만족시키는 것이다[1, 5]. 연결관리 구조는 통신세션 관리부분과 연결관리 패키지(CM Package) 부분으로 이루어진다. TINA 연결관리에서는 TMN(Telecommunication Management Network) 레이어의 다섯 개의 기능 중에서 NML(Network Management Layer) 기능과 EML(Element Management Layer)기능을 포함하고 있다. NML 기능은 EML에 의해서 표현된 폭 넓은 부위의 모든 망 자원들의 관리를 책임진다. 또한, 피어 NML 기능들과 기능적인 연합을 이루며 상호작용한다. EML 기능은 망 요소 계층의 기능을 관리한다. EML에서는 부분망(Subnetwork)으로 망 자원을 그룹지어 놓고, 각각의 부분망을 하나의 EML 기능이 관리한다. 일반적으로 EML에서의 부분망(Subnetwork)이란 하나의 망 장비가 될 수도 있고, 새로운 서비스를 제공하기 위한 설비의 집합이 될 수도 있다. (그림 2-3)에서는 연결관리 컴포넌트와 TMN 레이어, 그리고 연결관리 패키지를 보여준다. 본 논문에서는 라우팅 기능이 필요한 연결관리 패키지 부분에 대해서 자세히 다룬다.

앞에서 기술한 바와 같이 TINA구조에는 여러가지 기존의 개념들을 수용하고 있다. TINA연결관리 기능을 설계하고, 구현하는 과정에도 TINA구조의 모든 개념이 사용된다.



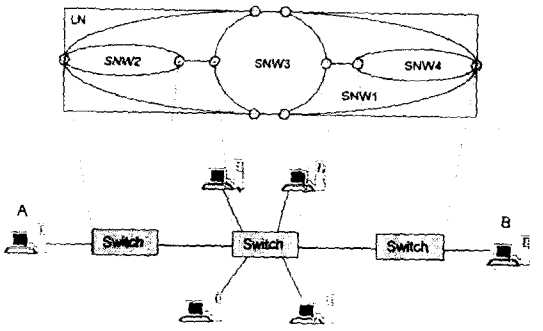
(그림 3) 연결관리 컴포넌트와 TMN 레이어
(Fig. 3) CM Components and TMN layer

3.1 연결관리의 정보모델

연결관리의 정보모델은 망 구조의 NRIM에 크게 의존한다. NRIM은 G.803의 계층화(Layering) 및 분리화(Partitioning) 개념과 M.3100의 TMN 객체 클래스 정의 개념을 수용한다. NRIM에서는 망의 정적인 자원을 기술하기 위한 객체들과 망에서의 연결과 같은 동적인 자원을 표현하기 위한 객체들을 정의하고 있다. 이러한 객체들은 물리적인 망의 자원들을 논리적으로 표현하기 위한 수단으로서 연결관리의 모든 기능은 이 객체들을 대상으로, 혹은 이 객체들에 의해서 일어난다.

NRIM에서 정보모델링에 의해 정의된, 연결관리 패키지에서 사용되는 객체들은 정적 관리객체(Static Managed Object)와 동적 관리객체(Dynamic Managed Object)로 구분된다. 전자는 망의 정적인 자원을 표현하는 객체로서, 연결의 설정이 요구되기 전에 이미

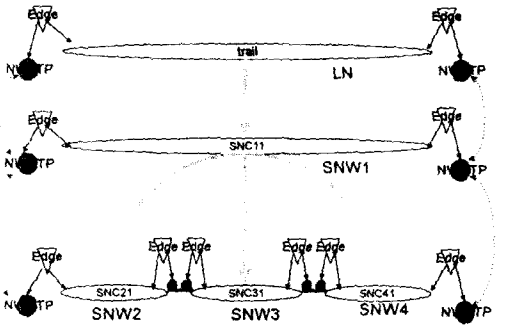
존재하고 연결이 끝난 후에도 지속적으로 존재하는 객체이다. 이 객체의 생성은 망의 형성때(Configuration time) 일어나고, 망의 형상(Configuration)이 변할때 삭제(혹은 생성) 작업이 일어난다. (그림 4)에 물리적인 망의 구성과 이에 맞는 정적 관리객체들을 보이고 있다.



(그림 4) 물리적인 망과 정적 관리객체
(Fig. 4) Physical Resources and Static Management Objects

LN은 계층망(Layer Network)을 표현한다. 전달되는 자료의 특성에 따라 망은 여러개의 계층망으로 구분될 수 있다. 예를들면 ATM VP계층과 ATM VC계층으로의 구분이 있다. SNW는 subnetwork을 표현하는 객체이다. Subnetwork은 G.803의 분리화 개념에 따라 분리된 망의 단편이다. SNW는 크게는 계층망에서 부터 작게는 하나의 스위치 이하 까지도 표현한다. TP(nwttp, nwctp)와 POOL(nwtppool, link temination point)은 SNW와 LN의 종단점과 종담점의 묶음(Ports)을 표현하는 객체이며, TL(Topological Link)과 connection은 SNW간, 또는 LN간의 연결을 표현하는 객체이다.

동적 관리객체는 연결을 표현하는 객체들로서 연결 설정시에 생성되고, 해제시에 삭제되는 객체들이다. 이 객체들이 생성되고, 정적 관리객체들과 바인드되면서 연결 설정작업이 진행된다. (그림 5)는 (그림 4)에 구성된 망의 예제에서 두 호스트 A, B간의 점 대 점 연결이 설정된 경우 생성되는 동적 관리객체를 계층별로 보여주고 있다.



(그림 5) 동적 관리객체 예제
(Fig. 5) Example : Dynamic Management Objects

Trail은 LN의 관점에서 연결을 표현한다. 또한, trail은 연결관리 패키지의 최상위의 연결을 표현한다. 즉, 연결관리 패키지의 클라이언트는 trail생성을 요구하므로써 연결관리 패키지의 서비스를 받게된다. SNC는 하나의 SNW안의 연결을 표현한다. 하나의 SNW는 여러개의 SNW로 다시 분할될 수 있으므로, 하나의 SNC역시 여러개의 SNC로 분할될 수 있다. TC (Tandem Connection)는 하나의 trail을 생성해 나가는 과정에서, 하나 이상의 SNC를 하나의 연결 객체로 표현하기 위한 객체이다. Edge는 trail, SNC, TC를 정적 관리객체인 nwttp나 nwctp와 바인딩 시키는 객체이다. 즉, 모든 연결에는 연결의 종단점이 있어야 하므로, 연결을 표현하는 객체에도 연결의 끝점을 표현하는 객체가 필요하다. Edge는 위에서 나열한 연결을 표현하는 객체들과 물리적인 망에서의 종단점을 연결시켜 주는 역할을 하며, 동적인 개념을 가지고 여러가지 연결 설정, 유지 및 해제 과정에서 발생하는 연결의 동적인 특성을 해결한다.

3.2 연결관리의 연산모델

전달망은 여러가지 전달 자료의 특성에따라 다양한 계층망으로 구성된다. 이러한 계층망들은 하나의 LNC(Layer Network Coordinator)와 여러개의 CP (Connection Performer)들로 구성된 연결관리 패키지들에 의해 제어된다. 연결관리의 연산모델에서는 연결관리에 필요한 컴포넌트들을 정의하고 있다. 본 논문에서는 연결관리 패키지에서 필요한 연산객체, 즉 소프트웨어 컴포넌트에 대해서 알아본다.

연결관리 패키지에서 정의된 연산객체는 LNC (Layer Network Coordinator)와 CP(Connection Performer)이다. CP는 TMN 레이어의 두가지 기능(EML과 NML)을 포함하고 있다. NML기능을 포함하고 있는 CP를 NML-CP, EML기능을 포함하고 있는 CP를 EML-CP로 구분하고 있다. 각 연산객체의 기능은 다음과 같다.

LNC는 하나의 계층망을 제어하기위한 연산객체이다. LNC는 자신의 계층망에서 trail을 생성할 의무를 갖는다. 하나의 계층망에는 하나의 LNC가 존재하므로, LNC는 한 계층망의 유일한 접근경로가 된다. 두개 이상의 계층망을 경유하는 trail의 경우는 페더레이션(Federation) 기능이 필요하게 된다. LNC는 LN, trail, TPPool, TP(nwttp), edge 등의 정보객체를 갖고 있으며 이들을 이용해서 연결에 관련된 작업을 한다.

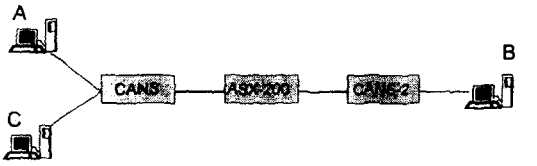
CP는 하나의 SNW(Subnetwork)를 관리하며, SNW의 종단점 간의 연결을 설정한다. 이 연결을 SNC (Subnetwork Connection)이라고 한다. CP는 SNC를 설정, 수정 및 해제하는 기능을 갖는다. CP는 NML-CP(Network Management Layer Connection Performer)와 EML-CP(Element Management Layer Connection Performer)로 구분된다. EML-CP는 NE(Network Element)에 접근해서 망 요소의 관리기능까지 수행한다. CP에는 SNW, SNC, TP(nwctp), TPPool, edge등의 정보객체를 갖고 있으며 이들을 이용해서 연결에 관련된 작업을 한다.

4. TINA연결관리 패키지의 구현

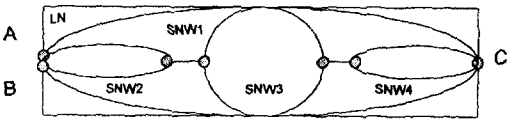
4.1 구현 범위 및 환경

본 논문에서 구현한 연결관리 패키지는 (그림 6)과 같은 하드웨어 플랫폼을 기반으로 개발되었다. (그림 7)은 (그림 6)의 하드웨어 플랫폼을 TINA의 NR1M에서 정의한 논리적인 망 구조로 대응시킨 것이다. 그림에서 보듯이 구현할 대상이되는 망 구조는 하나의 계층망과 네개의 SNW로 구성되어 있다. 그러므로, 본 구현에서는 두개 이상의 계층망 사이에서 필요한 계층망 간의 Federation 기능과, 정보객체 TC (Tandem Connection)등을 정의하지 않는다.

또한, 연산객체 NML-CP가 갖는 라우팅 기능은 자신의 SNC가 경유하는 하위 SNW들의 NWCTP를 직



(그림 6) 망의 하드웨어 플랫폼
(Fig 6) Hardware Platform



(그림 7) 망의 논리적 구조
(Fig. 7) Logical Architecture

접찾아서 연결절차를 진행하는 정적 라우팅을 이용한다. 이에따라 모든 NWCTP들의 관리는 NML-CP에서 일어나게 되므로 정보객체인 NWTPPOOL과 LTP(Link Termination Point)는 정의하지 않는다.

TINA에서 제공하는 하부구조는 TINA DPE이다. 그러나, 아직 이를 상용화시킨 제품이 나오지 않았기 때문에, 본 구현에서는 TINA DPE와 유사한 규격을 정립하고 있는 CORBA를 상용화한 제품인 Orbix (Version 1.3.5)를 하부구조로 사용한다. 이에따라, 인터페이스 정의를 위해서 CORBA의 IDL을 사용한다.

컴포넌트의 개발을 위해서 사용한 시스템은 DEC 3000 (model 800 workstation)이고, 운영체제는 DEC OSF/1 V3.2이다. 또, 사용언어는 IDL과 C++이고, Orbix Library를 필요로한다.

ATM 교환기는 ASX-200 FORE Switch 와 ETRI에서 개발한 CANS를 사용한다. 분산환경에서 개발된 프로그램은 분산환경이 갖추어진 기종간에만 상호작용이 일어날 수 있는 현실에 반에 상기 기술된 스위치들은 아직 분산환경, 즉 미들웨어(본 논문에서는 Orbix)가 갖추어져 있지 않다. 이를 극복하기 위해 연결관리 패키지와 망 장비(Switch) 사이에는 분산환경과 비 분산환경사이의 중간자로서 SA(Switch Agent)를 정의 및 구현하고, 이를 통해 스위치에 명령을 전달한다[6].

연결관리의 최 하단 컴포넌트인 (EML)CP는 SNMP에 따라, 스위치의 MIB(Management Information

Base)를 이용해서 스위치에게 명령을 전달한다.

구현에 관한 내용은 TINA연산객체 중심으로 설명된다.

LNC는 하나의 계층망을 제어하기위한 연산객체이다. LNC는 자신의 계층망에서 trail을 생성할 의무를 갖는다. 하나의 계층망에는 하나의 LNC가 존재하므로, LNC는 한 계층망의 유일한 접근경로가 된다. 두개 이상의 계층망을 경유하는 trail의 경우는 페더레이션 기능과 TC(Tandem Connection)객체가 필요하지만 본 구현에서는 하나의 계층망 만이 존재한다.

LNC에 속하는 정보객체와 그 기능은 다음과 같다.

4.2.1 LN (Layer Network)

LN은 LNC의 인터페이스를 갖는다[3]. 구현 과정의 인터페이스는 Orbix에서 지원하는 CORBA IDL을 사용했으므로 이하 인터페이스의 표기는 CORBA IDL을 이용한다.

```
interface LN {
        void reserve_trail
        (in NWAP nwap_root,
            in NWAP nwap_leaf,
            out Trail_ID_t trail_id,
            out string result)
        void reserve_branch_trail
        (in Trail_ID_t trail_id,
            in NWAP nwap,
            out string result)
        void setup_trail
        (in Trail_ID_t trail_id,
            out string result)
        void setup_branch_trail
        (in Trail_ID_t trail_id,
            in NWAP_ID_t nwap_id,
            out string result)
        void release_trail
        (in Trail_ID_t trail_id,
            out string result)
        void release_branch_trail
        (in Trail_ID_t trail_id,
            in NWAP_ID_t nwap_id,
            out string result)
};
```

LN은 trail을 생성, 관리 및 해제하고, 루트 NWTP를 예약한다. Trail의 생성은 reserve_trail 인터페이스가 호출되면 일어나고, setup_trail이 호출되면 trail객체의 인터페이스 attach_edge를 호출해서 CP의 스위치 셋팅 인터페이스를 호출하도록 한다. LNC의 클라이언트는 trail의 생성을 요구할때 항상 reserve_trail 바로 다음에는 setup_trail을 호출해야 한다. resreve_branch_trail 인터페이스는 하나의 리프 파티, 즉 수신 종단점을 추가하고자 할 경우 호출한다. setup_branch_trail은 reserve_branch_trail에 의해 추가된 연결을 설정할 경우에 호출된다. 점 대 다중점 연결일 경우 하나의 리프 파티가 연결 해제를 원할 경우는 release_branch_trail이 호출된다. 연결을 완전히 종결할 때는 release_trail이 호출된다.

본 구현은 TINA의 연결관리 규격에 따라 trail이하의 연결관리 객체의 인터페이스를 구현하였다. 그러므로, LN은 상위 연산객체 CC와 LNC내부의 trail간의 인터페이스 역할을 한다. 예를들어, CC가 reserve_trail을 호출하면 LN은 trail의 멤버함수 create_edge를 두번(root, leaf)호출하게 된다.

in 파라미터로 사용되는 NWAP는 망의 접근점(Netwrok Access Point)을 표현하는 구조체 로써 연결설정에 필요한 정보를 가지고 있다. reserve_trail의 out 파라미터 trail_id는 나머지 인터페이스 호출시 LN에서 관리되는 trail중 하나를 지칭하는 정보로 사용된다.

4.2.2 Trail

Trail은 계층망에서의 연결 및 망 전체에서의 연결을 표현한다. 본 구현에서는 trail에서 갖는 인터페이스가 모두 LN에서 구현되었다. 그러므로, trail의 인터페이스는 TINA ODL로 표현하지 않으며. Trail은 LNC내부에서만 정의된다. Trail의 멤버함수는 LN에 의해서 호출된다. Trail의 멤버함수는 다음과 같다.

```
void create_edges
   (NWTP_t_c* p_nwttp, char*& result);
void create_branch_edge
```

(NWTP_t_c* p_nwttp, char*& result);
void attach_edge
(NWTP_t_c* p_nwttp, char*& result);
void delete_edge
(NWTP_t_c* p_nwttp, char*& result);
void destroy_trail (char*& result);

Trail의 멤버함수는 TINA-C의 CM 규격서에서 정의한 trail의 인터페이스와 이름과 기능이 거의 일치한다. create_edges는 루트 및 리프 edge를 생성하고, CP에게 SNC 및 edge의 생성을 요구한다. create_branch_edge는 LN의 reserve_branch_trail 인터페이스에 의해 호출되며, CP에게 edge 추가를 요청한다. attach_edge는 setup_trail 인터페이스에 의해 호출되며, 스위치의 셋팅을 위해 CP의 attach_edge 인터페이스를 호출한다. delete_edge는 연결이 설정되어 있는 하나의 리프파티 하나를 제거하는 작업을 한다. LN의 release_branch_trail에서 호출된다. 마지막으로, destroy_trail은 연결 전체를 해제할때 LN의 release_trail에서 호출되는 멤버함수 이다.

Trail은 CP인터페이스의 호출을 위해 최 상위 CP, 본 구현에서는 NML-CP, 의 서버이름과 SNW 마커 이름을 연결설정 작업이 시작되기 전에 알고 있어야 한다. 이 정보는 trail이 LN에 의해서 생성되면서 trail의 속성에 셋팅된다. LN은 망의 컨피그레이션 작업때 이와같은 정보를 알게된다.

4.2.3 Edge

Edge는 NWTP를 대신해서 trail과 SNC와 같은 동적연결의 종단점을 동적으로 표현하는 객체이다. 그러므로, edge는 NWTP를 바인드하는 기능을 갖는다. Edge는 각 연산객체 내부의 동적 연결객체의 종단점을 나타내기 때문에 연산객체간의 인터페이스로 정의되지 않는다. 단, 상위 연산객체의 edge는 하위 연산객체의 edge와 대표관계(delegation)가 성립된다. 대표관계의 정보를 보관하기 위해서 edge는 대표되는edge의 ID를 저장한다. Edge의 존재는 edge이 가리키는 곳에 연결이 설정중 이거나 설정이 되있음을 뜻하고, 이후의 모든 설정 및 해제 작업(attach, delete, destroy)은 대표관계의 edge ID를 인터페이스의 파라미터로 넘겨주면서 일어난다. 다음은 edge의 주요 멤버함수 이다.

void delegation_edge(Edge_ID_t edge_id);
void bind_edge(NWTP_t_c *nwtp);
void delete_edge( );
void attach_edge( );

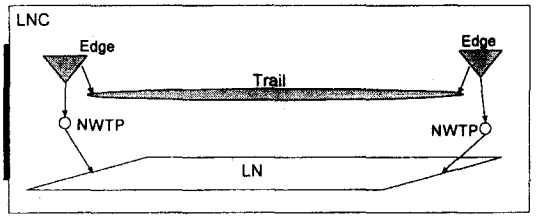
4.2.4 NWTP

NWTP는 각 연산객체가 갖는 망(LN or SNW)의 종단점을 표현하는 객체이다. 본 구현의 최 하위 CP, EML-CP, 에서는 NWTP가 ATM의 VPI/VCI와 대응된다. EML-CP의 NWTP는 스위치를 셋팅하는 기능이 필요하다. NWTP는 trail의 NWTTP와 SNC의 NWCTP에 유전되는 상위 클래스로써 각 연산객체에 포함되는 정적객체인 LN 혹은 SNW에 포함되므로 인터페이스로 정의하지 않는다. NWTP도 edge와 같이 상위객체와 하위객체간에 대표관계(delegation)가 있다. 이 관계는 망의 컨피그레이션 과정에서 각 연산객체의 망이 생성되면서 결정되고, 정보가 저장된다.

NWTP의 주요 기능은 다음 멤버함수에 나타내었다.

void reserve_nwtp (void);
void activate_nwtp (void);
void release_nwtp (void);

(그림 8)에서는 LNC에 속하는 정보객체를 도시하고 있다.



(그림 8) LNC와 정보객체
(Fig. 8) LNC and Information Objects

4.3 Connection Performer

CP는 하나의 SNW(Subnetwork)를 관리하며, SNW

의 종단점 간의 연결을 설정한다. 이 연결을 SNC (Subnetwork Connection)이라고 한다. CP는 SNC를 설정, 수정 및 해제하는 기능을 갖는다. CP는 NML-CP (Network Management Layer Connection Performer)와 EML-CP(Element Management Layer Connection Performer)로 구분된다. EML-CP는 NE(Network Element)의 에이전트에 접근해서 망 요소의 관리기능까지 수행한다. 본 구현에서는 하나의 NML-CP와 세 개의 EML-CP를 정의한다. CP의 인터페이스는 SNW와 SNC가 갖는다.

CP에 속하는 정보객체와 그 기능은 다음과 같다.

4.3.1 SNW (Subnetwork)

SNW의 인터페이스는 다음과 같다.

```
interface SNW {
        void create_snc
        (in NWTP_ID_t nwctp_root_id,
            out SNC_ID_t snc_id,
            out Edge_ID_t root_edge_id,
            out string result)
        void release_snc
        (in SNC_ID_t snc_id,
            out string result)
};
```

SNW는 SNC를 생성, 관리 및 소멸하는 기능을 갖는다. CP 연산객체에 SNW는 단 하나만 존재한다. SNW의 인스턴스가 생성되면서 SNW의 NWTP들이 생성된다. SNW에 속하는 NWTP등의 컨피그레이션 정보는 화일 형태로 이미 저장되어 있고, 이 화일들은 SNW의 인스턴스가 생성되는 과정에서 읽혀진다. create_snc는 상위 CP의 SNC혹은 LNC의 trail에서 호출하는 인터페이스이다. 이 인터페이스는 SNC를 생성하고 생성한 SNC의 create_edges인터페이스를 호출해서 루트 edge를 생성하게 한다. 그리고, 생성된 SNC와 edge의 ID를 리턴한다. release_snc 인터페이스는 SNC의 ID를 받아서 SNC를 삭제하는 기능을 수행한다.

4.3.2 SNC (Subnetwork Connection)

SNC는 SNW에서의 동적인 연결을 나타내는 객체이다. 인터페이스는 다음과 같다.

```
interface SNC {
        void create_edges
        (in NWTP_ID_t nwctp_id,
            out Edge_ID_t edge_id,
            out string result);
        void create_branch_edge
        (in NWTP_ID_t nwctp_id,
            out Edge_ID_t edge_id,
            out string result);
        void attach_edge
        (in Edge_ID_t edge_id,
            out string result)
        void delete_edge
        (in Edge_ID_t edge_id,
            out string result)
        void destroy_snc
        (out string result)
};
```

본 논문에서 구현한 SNC의 인터페이스는 TINA-C의 CM 규격서에서 정의한 SNC의 인터페이스와 이름 및 기능이 일치한다. create_edges는 루트 및 리프 edge를 만들고, 자신이 NML-CP의 SNC일 경우는 하위 CP, 본 구현에서는 EML-CP, 의 인터페이스를 호출한다. create_branch_edge는 리프 파티의 추가시에 호출되는 인터페이스로써 create_edge에서 리프 edge를 생성할 경우와 같은 작업을 한다. attach_edge는 생성되어 있는 SNC의 한 리프 edge부터 루트 edge까지의 동적 객체들을의 상태를 BUSY로 만들어준다. 이때, 자신이 EML-CP의 SNC일 경우는 스위치의 셋팅 작업을 한다.

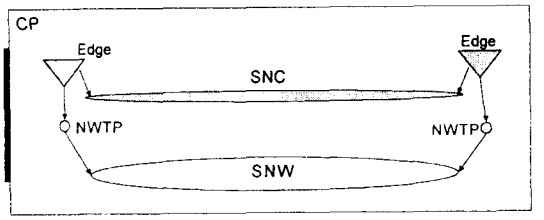
delete_edge는 여러개의 리프 파티중 하나를 삭제할 경우 호출되는 인터페이스이다. NML-CP의 delete_edge일 경우는 삭제되는 여녁陜에따라 하위 CP에 destroy_snc나 delete_edge 인터페이스를 호출하게된다. destroy_snc는 CP의 SNC를 완전히 삭제하고자할 때 호출되는 인터페이스이다. NML-CP의 경우는 하위 SNC에게 반복적으로 destroy_snc를 호출한다. EML-

CP의 경우는 스위치에 셋팅되있던 연결을 해제한다.

4.3.3 Edge와 NWTP

CP에 속하는 edge와 NWTP는 LNC의 그것과 같은 기능을 하므로 동일한 객체를 사용한다.

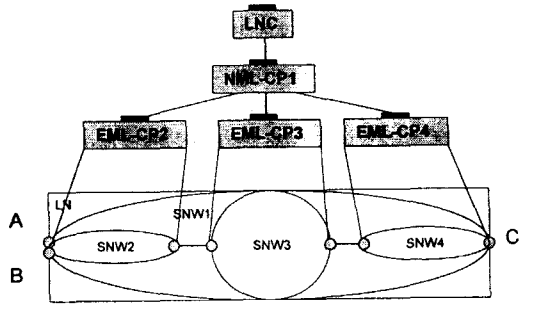
(그림 9)는 CP의 정보객체를 보이고 있다.



(그림 9) CP와 정보객체
(Fig. 9) CP and Information Objects

4.4 연결설정 및 해제과정 시나리오

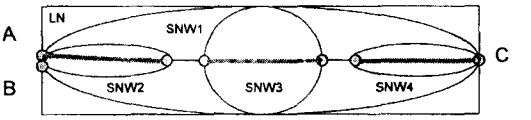
이 단원에서는 앞에서 구현된 연결관리 컴포넌트의 연결 설정 및 해제 기능을 실제로 물리적인 망에 적용해 보고, 그 과정을 보인다. 망의 구조에 따른 연산객체는 (그림10)에서 보인다.



(그림 10) 망 구조와 연산객체
(Fig. 10) Network Architecture and Computational Objects

4.4.1 시나리오

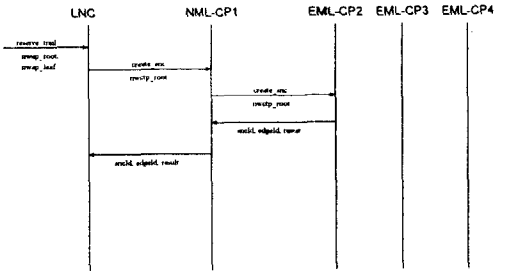
연결관리 패키지의 클라이언트는 (그림 11)과 같은 연결 설정을 요구한다. A와C의 연결 설정을 위해서는 (그림 5)와 같은 동적객체들이 생성되어야 하며, 각 동적객체들의 생성을 위해서 연산객체들의 인터페이스를 통한 메시지의 흐름이 필요하다.



(그림 11) 연결설정 예제 (A→C)
(Fig. 11) Connection Setup example

4.4.2 연결관리 패키지의 메시지 흐름

(그림 11)의 연결을 위한 연결관리 패키지의 연산객체간의 메시지 흐름은 (그림 12, 13, 14, 15)와 같다. 모든 메시지들은 각 연산객체가 갖고있는 인터페이스의 파라미터로서 전달된다. 연결 설정을 위한 각각의 과정은 다음과 같이 설명될 수 있다.



(그림 12) 메시지 흐름 (reserve_trail 1)
(Fig. 12) Message Flow (reserve_trail 1)

A. reserve_trail

◎ LNC Client : trail 생성 요구(reserve_trail).

① LNC : trail 생성, root edge 생성, SNC 생성요구(create_snc).

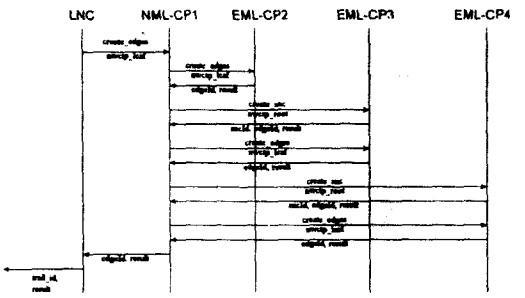
② NML-CP1 : SNC 생성, root edge 생성, Router를 통해 root edge의 Subnetwork (EML-CP2) search, SNC 생성요구(create_snc).

③ EML-CP2 : SNC 생성, root edge 생성, 생성된 sncId, root edge Id 리턴.

④ NML-CP1 : 리턴된 sncId, root edge Id 저장, 생성된 sncId, root edge Id 리턴.

⑤ LNC : 리턴된 sncId, root edge Id 저장, leaf edge 생성, edge 생성 요구(create_edges)

⑥ NML-CP1 : leaf edge 생성, 라우팅(EML-CP2, EML-CP3, EML-CP4 search), EML-CP2에 leaf edge 생성 요구.



(그림 13) 메시지 흐름 (reserve_trail 2)
(Fig. 13) Message Flow (reserve_trail 2)

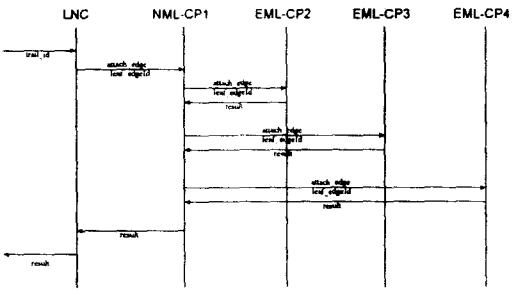
⑦EML-CP2 : leaf edge 생성, leaf edge Id 리턴.

⑧ NML-CP1 : 리턴된 leaf edge Id를 라우트 리스트에 저장, EML-CP3, EML-CP4에게 create_snc, create_edge를 순차적으로 요구.

⑨EML-CP3, EML-CP4 : EML-CP2의 ③, ⑦과 같은 절차.

⑩NML-CP1 : 리턴된 leaf edge Id를 라우트 리스트에 저장, ⑥에서 생성한 leaf edge Id 리턴.

⑪LNC : 리턴된 leaf edge Id 저장, trail Id 리턴.



(그림 14) 메시지 흐름 (setup_trail)
(Fig. 14) Message Flow (setup_trail)

B. setup_trail

◎LNC Client : trail Id 를 주며 trail setup 요구.

①LNC : edge, nwttp의 status를 BUSY로 변환, NML-CP1의 attach_edge 인터페이스 호출.

②NML-CP1 : edge, nwctp의 status를 BUSY로 변환, 라우트 리스트를 따라가면서 EML-CP들에게 attach_edge 요구.

③EML-CP2, 3, 4 : edge, nwctp의 status를 BUSY로 변환, SNMP MIB를 이용해서 스위치 셋팅 작업.

C. Release_trail

◎LNC Client : trail Id를 주며 trail 해제 요구(release_trail).

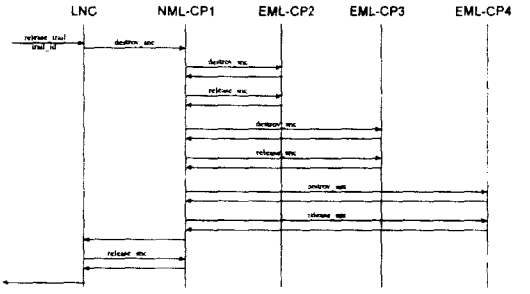
①LNC : NML-CP1에게 destroy_snc 요구.

②NML-CP1 : 라우트 리스트를 따라가며 하위 컴포넌트들에게 destroy_snc 요구.

③EML-CP2, 3, 4 : SNMP MIB를 이용해서 스위치 해제 작업, edge와 SNC 를 삭제(free)하고, nwctp의 상태를 IDLE로 변경.

④NML-CP1 : edge와 SNC를 삭제(free)하고, nwctp의 상태를 IDLE로 변경, 라우트 리스트 삭제.

⑤LNC : edge와 trail을 삭제(free)하고, nwttp의 상태를 IDLE로 변경.



(그림 15) 메시지 흐름 (release_trail)
(Fig. 15) Message Flow (release_trail)

5. 결 론

분산 컴퓨팅 개념, 객체 지향 개념 등 기존의 통신 및 컴퓨팅 개념들을 수용하는 개방형 정보 통신망 구조를 위한 컨소시움인 TINA-C 는 시간이 갈수록 복잡해지고 다양해지는 통신망의 원할한 관리와 운용을 위해 이질적인 통신망을 연동할 수 있게하는 일관성있는 통신망 구조를 제시하고 있다.

본 논문에서는 이러한 개방형 정보통신망 구조에서의 연결관리 기능을 구현 시험하였다. 연결관리 기능을 구현하기 위해서 OMG CORBA의 상용제품인 IONA社의 Orbix (ver 1.3.5)를 하부구조로 사용하였고, 망의 스위치는 ATM 스위치(Fore, CANS)를 사용

했다. 본 논문에서 구현된 연결관리 패키지는 TINA 연결관리 규격과 망 자원의 정보모델을 충실히 따라, 연결이라는 동적요소의 표현과 기능은 동적 관리객체가 담당하고, 망 구조의 표현과 기능은 정적 관리객체를 이용했다. TINA 연결관리에서의 중요한 기능인 라우팅 기능은 정적 라우팅을 원칙으로 했다. 또한, 본 논문에서 구현한 컴포넌트들은 시험될 망의 구조에서 필요한 기능 및 객체들만을 정의하고 구현되었다. 아직 구현되지 않은 기능 및 객체들은 향후에도 계속해서 연구, 설계, 구현될 예정이다.

참 고 문 헌

[1] "TINA 연결관리를 위한 동적 멀티캐스팅 알고리즘", 박준희, 충남대학교 석사학위논문, 1997년 2월.

[2] "An Overview of the TINA", G. Nilsson 외 2인, TINA Conference Proceeding, pp1-12, February, 1995.

[3] "An Implementation of TINA Service and Connection Management Architectures on ATM Network", D. S. Yun 외 1인, TINA Conference Proceeding, pp134-144, September, 1996.

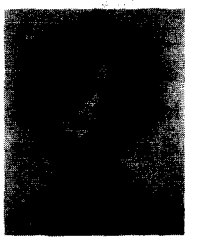
[4] "The TINA-C: Towards Networking Telecommunications Information Services", F. Dupuy 외 2인, ISS '95 Contribution, June, 1994.

[5] "TINA-C Connection Management Components", J. Bloem 외 3인, TINA Conference Proceeding, pp485-494, February, 1995.

[6] "분산환경을 지원하지 않는 ATM 스위치를 위한 중개자의 설계 및 구현", 박준희 외 3인, 한국정보과학회, 학술발표논문집, 제23권 2호, pp. 993-996, Oct., 1996.

박 준 희

1995년 충남대학교 자연과학대학 컴퓨터과학과(학사)
1997년 충남대학교 대학원 컴퓨터과학과(석사)
1997년~현재 시스템 공학 연구소 분산 컴퓨팅연구실 연구원
관심분야: 컴퓨터 네트워크, 분산 컴퓨팅, 병렬 컴퓨팅, 고속 LAN 프로토콜



오 현 주

1991년 충남대학교 자연과학대학 전산학과(이학사)
1991년~현재 전자통신연구원 통신망구조연구실 선임연구원
1996년~현재 충남대학교 대학원 컴퓨터과학과 석사과정 재학중
관심분야: 분산처리 시스템, 이동통신

김 상 하

1980년 서울대학교 화학과(이학사)
1984년 University of Houston (화학과 석사)
1989년 University of Houston (전산학과 박사)
1989년 HNSX Supercomputers Inc.(자문위원)
1992년 KIST/SERI(선임연구원)
1992년~현재 충남대학교 컴퓨터과학과 부교수 재직
관심분야: 컴퓨터 네트워크, 분산 시스템, 광대역/신호 통신망, 이동 통신, 분산 운영체제