NGN 서비스 구현과 성능측정

준회원 김 형 민*, 정회원 김 화 성*, 최 영 일**, 이 병 선**

Implementation of NGN Service and Performance Measurement

Hyoung-min Kim* *Associate Member*, Hwa-sung Kim*, Young-il Choi**, Byung-sun Lee**
Regular Members

요 약

오늘날 통신네트워크는 음성과 데이터의 통합으로 가는 것과 동시에 새로운 서비스가 요구가 증가됨에 따라서 NGN으로 변화하고 있다 NGN은 Service, Transport, Distributed Processing Environment의 3개 계층으로 구성된다 여기서 Service 계층은 다시 애플리케이션 계층과 Service Component 계층으로 구분할 수 있다 Parlay Group은 third-party 애플리케이션 제공의 실현을 위하여, 애플리케이션과 Service Component Layer사이에 인터페이스로 개방형 Parlay API를 채택하고 있다 Parlay API를 사용함으로써 third-party 업체들은 네트워크 제공자가 제공하는 Service component위에 있는 애플리케이션 레이어 즉 새로운 애플리케이션을 개발을 하게 될 것이다 본 논문에서는 third-party 서비스를 이용한 Third Party Call Control (TPCC) 서비스를 구현 하였다 TPCC서비스는 Parlay API와 Parlay X API를 사용해 각각 구현하였으며 Transport layer에서 시그널링 프로토콜로는 SIP를 사용하였다. 또한 각 구현사항들에 대하여 성능 측정을 실시하여 비교하였다

Key Words Parlay API, Parlay X API, NGN, SIP, Web Service, CORBA,

ABSTRACT

Communication network is in a transition toward the NGN (Next Generation Networks) to accommodate the explosive demand of new services The NGN allows the third-party application provisioning by defining the networks as layers of Services, Distributed Processing Environment and Transport Especially, the Service layer can further be divided into Application and Service Component layer In order to realize the third-party application provisioning, the Parlay Group has adopted an open Parlay API as an interface between the Application and the Service Component layer Using Parlay API, the third parties may develop and deploy the IT-based applications at the Application layer exploiting the service components located within network operators' domain In this paper, we present the implementation details about the Third Party Call Control (TPCC) Service using the third-party service logic based on Parlay API and Parlay X API, when SIP is used as a signaling protocol in Transport layer Also, we compare the performance evaluation of both implementations

Ⅰ. 서 론

통신 네트워크는 반도체와 광 전송 기술 같은 것뿐만 아니라 사용자들의 요구에 의한 새로운 서비스 질의 향상을 가져오면서 빠르게 변화하고 있다 예전에는 사용자들에게 제공되는 서비스들은 네트워크 제공자(Network Operator)들에 의해서 제공되고 있었다 최근 통신 환경이 개방형 네트워크(NGN)로 변화함에 따라 기존에 네트워크 제공자에 의해 독점적으로 제공되는 서비스에서 third-party 업체들이 서비스를 개

* 광운대학교 전자통신공학과 네트워크컴퓨팅 연구실(meruru98@kw ac kr, hwkim@daisy kw ac kr)
** 한국전자통신연구소, 소프트스위치팀({ychoi, bslee}@etri re kr)
논문번호 040147-0408, 접수일자 2004년 4월 8일
※본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 연구결과로 수행되었음

발하고 제공하는 형태로 변화하고 있다.

기존 네트워크 제공자들에 의해서 제공되는 서비스들은 표준과 네트워크의 보안, 호환성이 부족하고 사용자들이 원하는 다양한 서비스들을 빠른 시간 내에 제공하기가 힘들다. 이를 해결하기 위해 third-party 도메인에서 서비스(Application)의 개발이 가능하도록 하는 방법이 있다. 이것은 Application Layer와 Service Component Layer사이에 Open Application Programming Interface (API)를 정의함으로써 가능하다. 이렇게 구성된 API는 네트워크 밖에서 네트워크의 기능을 보여주고 이를 통해 프로그램을 가능하게 한다[1][2].

Application Layer와 Service Component Layer사이에 API의 대표적인 예로서 Parlay API가 있다. Parlay Working Group은 새로운 방식의 Open Programmable Network API규격을 제정하고 동시에 이 규격이 상용제품의 구현에 채택될 수 있도록 촉진하는 것을 목표로 하고 있다. 또한 Parlay API는 음성과 영상 그리고 데이터를 네트워크에서 제어하는 통합된 API를 추구한다. 최근 Parlay Working Group은 third-party업체들이 웹 서비스를 이용한 서비스의 개발을 가능하게 하고자 Web Service Working Group과 Parlay X Working Group을 재정하였다. 특히 Parlay X API는 기존의 API보다 더 추상화 시키고 웹 서비스를 사용함으로써 third- party업체들이 서비스를 개발을 더욱 쉽게 할 수 있도록 하였고, 사용자로 하여금 언제 어디서든지 쉽게 서비스를 사용할 수 있도록 하였다. 본 논문에서는 Parlay API와 Parlay X API를 구현하였고, 이와 관련된 서비스를 각각 구현하였다. 그리고 각 서비스들에 관한 성능측정을 실시하였다[3][4].

Ⅱ. Parlay API 와 Parlay X API

2.1 Parlay API

Parlay API는 아래와 같은 두 종류의 open 인터페이스로 이루어져 있다.

- 서비스 인터페이스: 이 인터페이스는 애플리케이션에게 접근 가능하게 하고 네트워크로부터 정보를 가져올 수 있는 기능을 한다. Generic Call Control Service, Generic Messaging Service, Generic User Interaction Service등이 이 인터페이스에 속한다.
- 프레임워크 인터페이스: 이 인터페이스는 서비스 인터페이스를 동작하고 관리하는 보조적인 기능을 지원한다. Authentication, Discovery, Event Notification, Integrity Management, Operation, Administration and Maintenance 이 프레임워크 인터페이스에서 하는 일이다.

위에 언급되었던 두 종류의 인터페이스들을 제공하기 위하여 네트워크에 의해서 제공하는 API를 사용하여 이 인터페이스들이 구현되었다. API는 네트워크 리소스의 컨트롤과 네트워크 리소스의 분리를 가능하게 만든다. 현재의 API는 H.323, SIP[5], INAP와 같은 인터페이스 자원을 사용할 수 있다.

2.2 Parlay X API

Parlay Working Group은 Parlay API 4.0 버전이상이 되면서 웹 서비스를 지원하기 위하여 Web Service Working Group을 재정하였다. Parlay API의 웹 서비스지원은 서비스의 제작과 배포를 쉽게 할 수 있게 되었다. 이때 Web Service Working Group과 같이 재정된 Parlay X Working Group에서는 웹 서비스를 이용한 서비스의 개발을 더욱 원활히 지원하기 위하여 Parlay X API를 발표하였다. Parlay X API는 차세대 네트워크 애플리케이션의 개발자가 telephony나 통신에 대한 정확한 지식을 가지고 있지 않더라도 제작이 가능하게 디자인이 되었다.

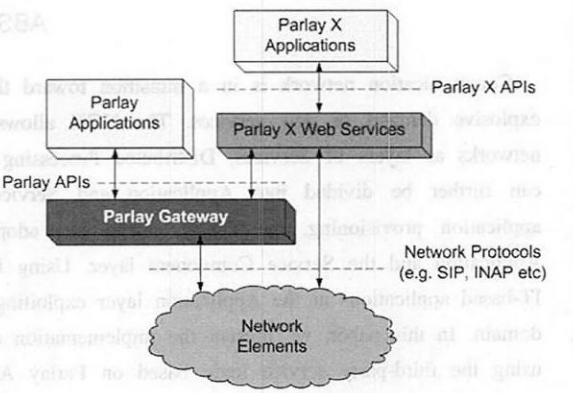


그림 1. Parlay API와 Parlay X API

그림 1은 Parlay X 웹 서비스와 Parlay APIs의 관계를 나타내고 있다. 그림에서 Parlay Gateway는 일반적으로 구현된 Parlay API가 있는 곳이다. 애플리케이션은 CORBA나 웹 서비스를 사용하여 Parlay Gateway를 거쳐 네트워크의 자원을 사용할 수 있다.

Ⅲ. Third-Party에서의 호 제어

3.1 Parlay API 호 제어과정

본 논문에서 설명하는 Parlay API는 자바 언어로 인터페이스를 구현했으며 미들웨어로는 CORBA를 사용하였다. 또한 IDL 컴파일러로는 Java 2 ORB를 사용하였다. Parlay API의 구현 부분은 Parlay Gateway에 존재하며 Application Server(AS)는 물리적으로 떨어진 다른 호스트에 존재한다. Parlay Gateway간의 통신에는 SIP 프로토콜을 사용한다.

그림 2는 call setup과정을 UML 시퀀스 다이어그램으로 나타낸 그림이다. 시퀀스 다이어그램에서 밖의 점선 박스는 third-party 서비스 제공자에 의해서 동작하는 AS와 네트워크 제공자가 제공하는 Parlay Gateway를 나타낸다. 그 안의 작은 회색의 박스들은 AS와 Parlay Gateway에 각각 있으며 Parlay API 인터페이스의 구현된 객체들을 이다. 이 객체들은 물리적으로 떨어진 AS와 Parlay Gateway를 연결한다. AS는 서버-클라이언트 모델에서 클라이언트의 역할을 하며 third-party 업체들에 의해서 제공되는 애플리케이션 로직을 실행시키며 Parlay Gateway와 연결하는 역할을 한다. Parlay Gateway에 구현되어 있는 서비스 인터페이스는 IpCall과 IpCallControlManager의 두 개의 인터페이스로 클래스로 구성되어 있다. AS에 구현되어 있는 애플리케이션 인터페이스로는 IpAppCall과 IpAppCall- ControlManger 두 개의 클래스 인터페이스로 구성되어 있다.

그림 2는 Parlay API 3.0 스펙의 시퀀스 다이어그램을 변형시킨 그림이다. Parlay Group에서 발표되는 API는 인터페이스 자원으로서 SIP, H.323, INAP와 같은 여러 가지 자원을 사용할 수 있게 발표되기 때문에 특정 인터페이스 자원과 Parlay API의 관계가 나와 있지 않는다. 하지만 구현하기 위해서는 Parlay Gateway부분의 SIP메시지와 Parlay API의 관계가 필요한 부분이므로 시퀀스 다이어그램을 수정 하였다. 본 논문에서 구현한 Incoming Call Routing 서비스의 시퀀스 다이어그램이 그림 2에 보이고 있다.

3.1 Parlay API 호 제어과정

본 논문에서 설명하는 Parlay X API역시 자바로 구현하였고, SOAP서버로서 AXIS를 사용하였다. Parlay X API는 웹 서비스를 사용하기 때문에 Parlay API와 같은 복잡한 과정을 거치지 않는다. 또한 Parlay API보다 더욱 추상화 되어있기 같은 서비스를

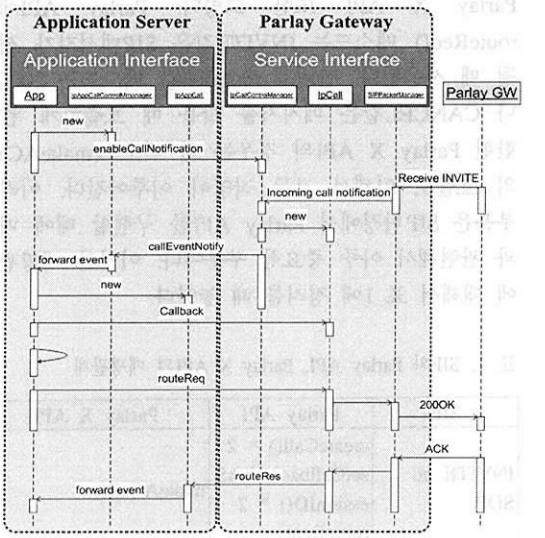


그림 2. Parlay API에서의 호 설정 과정 (메시지가 들어왔을 경우)

비교 했을 때 AS와 교환하는 메시지의 개수가 현저히 적다. 그림 3은 Parlay X API의 Third Party Call[6]에 나와 있는 Call API통해 call set-up하는 과정을 나타낸 시퀀스 다이어그램이다. makeACall()의 호출을 통해 call set-up과정이 일어나는 것을 볼 수 있다.

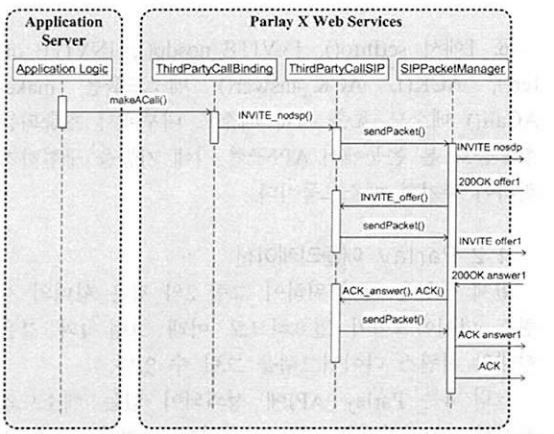


그림 3. Parlay X API의 호 설정 및 서비스 연결 과정[3]

Ⅳ. Third-Party 애플리케이션의 설계

4.1 SIP 매핑관계

Parlay Gateway에서 서비스 인터페이스를 구현할 때에 SIP의 요청 또는 SIP 응답 메시지를 Parlay API의 인터페이스의 메소드로 변경을 해야 한다.

Parlay X API 또한 그렇다. Parlay API에서 routeReq() 메소드는 INVITE같은 SIP메시지가 생성될 때 사용된다. 반대로 routeRes() 메소드는 200OK나 CANCEL같은 메시지를 받을 때 호출하게 된다. 한편 Parlay X API의 경우는 한 메소드(makeACall)의 호출로 인해서 모든 작업이 이루어진다. 이러한 부분은 SIP환경에서 Parlay API를 구현할 때에 매핑과 관련해서 아주 중요한 부분이다. 이러한 매핑관계에 대해서 표 1에 정리를 해 놓았다.

표 1. SIP와 Parlay API, Parlay X API의 매핑관계

SIP	Parlay API	Parlay X API
INVITE no SDP	createCall() * 2 setCallbackWithSessionID() * 2 routeReq()	makeACall()
200OK offer1	routeRes()	이벤트처리 (INVITE_offer()함수 호출)
INVITE offer1	routeReq()	INVITE_offer()
200OK answer	routeRes()	이벤트처리 (ACK(), ACK_answer() 함수 호출)
ACK	routeReq()	ACK()
ACK answer1	routeReq()	ACK_answer()

표 1에서 setInfo(), INVITE_nosdp(), INVITE_offer(), ACK(), ACK_answer() 메소드들은 makeACall() 메소드 호출 시에 메소드 내부에서 호출되는 함수들로 본 논문에서 API구현 시에 기능을 구현하기 위하여 추가한 메소드들이다.

4.2 Parlay 애플리케이션

실제 구현을 하기 위하여 그림 2와 같은 형태의 시퀀스 다이어그램이 필요하므로 아래 그림 4와 같은 형태의 시퀀스 다이어그램을 그릴 수 있다.

그림 4는 Parlay API에 정의되어 있는 메소드와 적절한 SIP메시지를 매핑을 통하여 그려질 수 있다. 메시지를 보내기 위해서는 routeReq()를 통하여 정보를 전달하고 메시지를 받은 경우는 routeRes()를 통하여 애플리케이션 로직에게 정보를 전달하게 된다. 단, 보내는 메시지의 종류를 구분하기 위해서는 각 메소드마다 들어있는 정보필드에 값을 다르게 넣어 줌으로서 구분하게 된다. 하지만 같은 애플리케이션을 구현한다 하더라도 항상 같은 시퀀스 다이어그램이 나

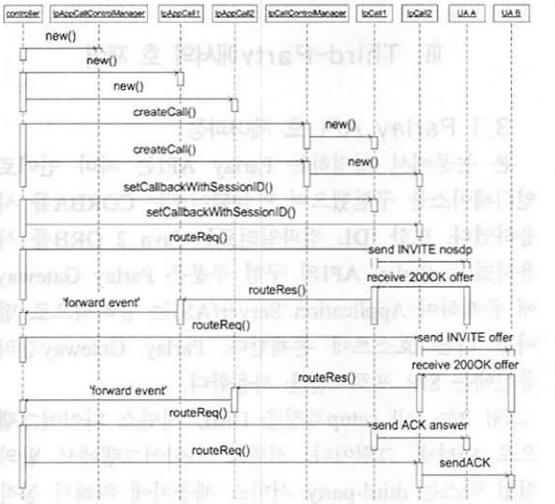


그림 4. Third-party Call Control 서비스의 시퀀스 다이어그램

오는 것은 아니다. 그림 4도 마찬가지로 multiparty call control과 call leg등을 사용하여 다른 시퀀스 다이어그램을 그려 구현할 수도 있다.

4.3 Parlay X 애플리케이션

Parlay X API에서는 Third Party Call Control의 기능을 하는 API를 제공하고 있다. 이 API를 통하여 구현을 하게 되면 그림 3과 같은 시퀀스 다이어그램을 그릴 수 있다. 그림 4와 비교하여 메시지 수가 감소하여 단순하게 표현되었다. 하지만 그림 3에서 보는 것처럼 애플리케이션에서는 makeACall함수만을 호출하고 나머지 일은 Parlay X Gateway부분에서 전부 처리하므로 Parlay API에서 언급한 여러 가지 형태의 시퀀스 다이어그램이 나오기는 힘들다. 이것은 애플리케이션을 개발하는 third-party 업체에서 새로운 기능을 추가하거나 세부적인 컨트롤이 필요한 애플리케이션개발을 어렵게 하는 요소이다.

V. 성능측정 및 비교

본 논문에서 구현한 Parlay API와 Parlay X API그리고 각각의 애플리케이션들이 동작 했을 경우 어느 정도의 성능의 차이가 나는지 알기 위하여 성능 측정을 하였다. Parlay API의 경우 Parlay Gateway의 서비스 인터페이스 부분과 AS에 있는 인터페이스 부분으로 나눠 측정하였다. Parlay X API의 경우도 마찬가지로 Gateway측의 인터페이스와 애플리케이션 부분으로 나눠 측정하였다. 세부적으로는 각각의 인터페

이스에 대하여 실제 API라 동작하는 시간인 인터페이스 메소드의 수행시간, CORBA의 등록과 수행시간, 웹 서비스 통과 시간 그리고 Parser의 수행시간을 측정하였다.

성능측정 환경은 PC에서 수행 하였으며 Parlay Gateway와 AS를 물리적으로 떨어진 다른 호스트에 설치하고 각각의 UA도 또한 다른 호스트PC에 위치하여 테스트를 하였다. 성능측정 툴로는 자바에서 지원하는 기본적인 툴들과 Rational Quantify를 사용하여 측정하였다. 각각은 자바로 구현 되었으며 동작 OS는 Windows에서 실행되었다.

표 2. 같은 기능을 하는 메소드들과의 수행시간 비교

	Parlay API		Parlay X API	
1	createCall() * 2 setCallbackWithSessionID() * 2 routeReq() 합계	532.14 550.18 93.68 1,176.00	makeACall()	669.37
2	routeReq()	93.68	INVITE_offer()	29.88
3	routeReq()	93.68	ACK()	18.49
4	routeReq()	93.68	ACK_answer()	12.76
5	routeRes()	113.22	이벤트처리	

표 2는 Parlay API와 Parlay X API의 같은 기능을 하는 메소드들끼리 수행 시간 차이를 나타낸 그래프이다. 표 1과 비교하면서 보면 1번의 경우는 AS와 Gateway간의 연결설정을 맺고 INVITE no SDP SIP 메시지를 보내는 기능을 하는 메소드들이다. Parlay X의 경우는 makeACall()를 한번 호출하면 그 모든 기능이 포함 되지만Parlay의 경우는 createCall(), setCallbackWithSessionID() 메소드를 각각 두 번씩 호출하고 routeReq()를 한번 호출해야 그 기능을 하게 된다. 그러므로 이 메소드를 더한 시간과makeACall의 시간을 비교하게 된다. 2는 INVITE offer1를 3은 ACK, 4는 ACK answer1 SIP메시지를 보내는 기능을 하는 메소드이다. 5번의 경우는 앞서 4장SIP 매핑 관계에서 설명 했듯이 Parlay X API의 경우 내부에서 이벤트 호출을 통하여 바로 다음 메소드를 호출하기 때문에 routeRes()에 대응되는 메소드가 존재하지 않는다.

그림 5는 표 2를 그래프화 한 것이다 x축은 표 2의 번호를 나타낸다. 전체적으로 Parlay API보다는 Parlay X API의 수행시간이 더 짧은 것을 볼 수 있다. 이것은 전체 메시지수가 감소하고 또한 메시지들 중에서 미들웨어를 통과하는 메시지 수를 줄였기 때

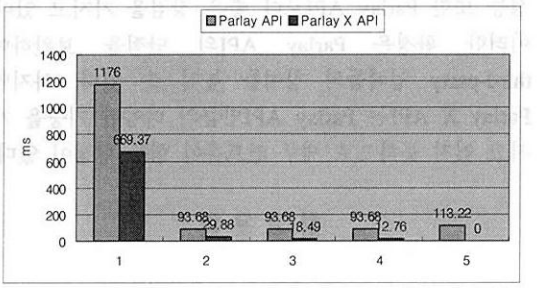


그림 5. 메소드 수행시간 비교

문에 미들웨어를 통과하는 만큼의 시간을 절약할 수 있게 된 결과이다.

VI. 결 론

Parlay API는 애플리케이션 계층과 서비스 컴포넌트 레이서 사이의 인터페이스로 사용하기 적합하다. Parlay API는 개방형 네트워크 구조에서 어떠한 방법으로 서비스를 제공할 것인가에 대하여 커다란 변화를 가져올 것이다. 현재 스위칭 시스템의 경우에 call control의 기능은 call setup이나 call release같은 서비스 로직이 스위치 노드들에 내장되어 있기 때문에 스위칭 노드들 간의 시그널 링을 통하여 실행되었다. 이러한 스위칭 노드들은 서비스 로직이 바뀌거나 새로운 서비스를 신속하게 제공하는 것이 어렵다. 하지만 Parlay API를 사용하면 적시에 다양한 서비스 로직의 제공자들에 의해서 새로운 네트워크 서비스들의 공급이 가능하게 된다.

본 논문에서는 구현한 API와 애플리케이션의 성능을 각각 측정하였고, 각각의 성능에 대하여 비교 하였다. 성능측정 후 결과를 비교하면 Parlay X API가 Parlay API의 수행시간 보다 짧음을 알 수 있다. Parlay X API가 추상적인 API를 제공함으로써 메시지 수를 감소하고 또한 미들웨어를 통과하는 메시지수를 줄인 결과이다. 실제적인 미들웨어 통과 시간은 시스템이 복잡한 웹 서비스가 오래 걸린다. 하지만 사용자 측에서는 Parlay 애플리케이션의 경우 GUI나 서비스 구동시간이 포함되지만 Parlay X 애플리케이션의 경우 웹 서비스를 이용하여 브라우저를 통해서 지원되기 때문에 GUI동작 시간이 없으므로 더욱 빠르게 느낄 수 있다.

Parlay X API는 최신의 기술을 도입하여 Parlay API보다 서비스 개발이 쉽고 사용자도 사용하기 쉬운 서비스를 제공하도록 하였고 또한 사용자가 느끼는

성능 또한 Parlay API보다 좋은 장점을 가지고 있다. 이러한 환경은 Parlay API의 단점을 보완하며 third-party 업체들의 참여를 높일 수 있다. 하지만 Parlay X API는 Parlay API만큼의 다양한 기능을 가지고 있지 못하므로 세부 컨트롤이 힘든 단점이 있다.

참 고 문 헌

[1] Parlay Group, *Parlay API Business Benefits White Paper*, June 1999.

[2] Parlay Group, Parlay API Spec. 3, Dec. 2001.

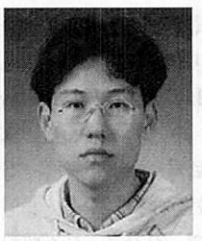
[3] A. John, R. Peter, *"Electric Communication Development," Communications of the ACM*, 40, pp. 71-79, May 1997.

[4] Parlay Group, *Parlay Web Services - White Papers*, Version 1.0, October 2002.

[5] IETF, draft-ietf-sip-rfc2543bis-03.txt, *"SIP: Session Initiation Protocol"*, May 2001.

[6] IETF, draft-ietf-sipping-3pcc-02.txt, *"Best Current Practices for Third Party Call Control in the Session Initiation Protocol"*, June 5, 2002.

김 형 민(Hyoung-min Kim) 준회원



2002년 2월 : 광운대학교 전자공학부 졸업

2004년 2월 : 광운대학교 전자통신공학과 석사

2004년 3월~현재 : 광운대학교 전자통신공학과 박사과정

<관심분야> NGN, 웹 서비스, 무선 이동통신

김 화 성(Hwa-sung Kim) 정회원

1981년 2월 : 고려대학교 전자공학과 졸업

1983년 2월 : 고려대학교 전자공학과 석사

1996년 : Lehigh Univ. 전산학박사

1984년 3월~2000년 2월 : ETRI 책임 연구원

2000년 3월~현재 : 광운대학교 전자공학부 교수

<관심분야> NGN 미들웨어 환경, QoS-aware 미들웨어, 그리드컴퓨팅

최 영 일 (Young-il Choi) 정회원

1983년 2월 : 서울대학교 전자공학과 졸업

1998년 2월 : 충남대학교 컴퓨터과학과 석사

2002년 : 충남대학교 컴퓨터과학과 박사

1996년 정보통신기술사

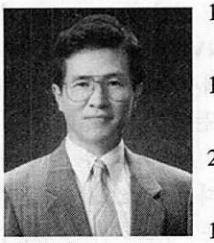
1985년~1986년 : Bell 연구소 객원연구원

1983년~현재 : 한국전자통신연구원 팀장

2002년~현재 : 표준화 전문위원 (MSF, Parlay)

<관심분야> 차세대 네트워크, Softswitch , 개방형 서비스 기술

이 병 선 (Byung-sun Lee) 정회원



1980년 : 성균관대학교 수학과졸업

1982년 : 동국대학교 전산학과 석사

2003년 : 한국과학기술원 전산학과 박사

1982년~현재 : 한국전자통신연구원 그룹장

2000년~현재 : 표준화전문가,TTA 통신망기술위원

<관심분야> NGN, 개방형 서비스 기술, Model-driven architecture