전화기-교환기 혼합 호 제어 방식에 입각한 CTI 콜 서버 시스템에 관한 연구

정회원 정 윤 찬*, 함 미 옥**

A Study on CTI Call Server System based on a Phone-Switch Hybrid Call Control Scheme

Youn-chan Jung*, Mi-ok Ham** *Regular Members*

요 약

본 논문은 컴퓨터 전화 통합 이용 CTI 시스템을 전화기 기반 호 제어 구조와 교환기 기반 호 제어 구조로 분류하여 이 두 가지 구성을 대비시켜 구조적으로 분석하였다. 그리고 교환기 기반 호 제어 구조를 가지면서도 전화기 기반 호 제어 방법의 장점을 혼합한 전화기-교환기 혼합 호 제어 방식에 입각한 콜 서버 시스템을 제안한다. 또 제안된 콜 서버 시스템은 일반 PC에서 CTI 웹 응용을 이용하는 사용자가 쉽게 CTI 플랫폼에 접근할 수 있도록 해주는 웹 기반의 시스템이다. 제시한 콜 서버의 분석 모델을 개발하고 콜 서버 모델을 자바 언어로 구현한다. 이렇게 구현된 콜 서버를 사용하여 전화기-교환기 하이브리드 호 제어 구조의 수행 특성을 조사하기 위하여 임의 사이트의 클라이언트 PC수와 호 접속 요구 발생률 등 클라이언트 환경 변수와 모뎀수, 대기큐의 크기, 모뎀 라이프 사이클 시간 등의 서버측 변수들이 호 접속 실패율과 호 접속 처리 완료 시간 등에 미치는 영향을 분석한다.

ABSTRACT

In this paper, we present a structural analysis of two CTI configurations : the phone-oriented call control scheme and the switch-oriented call control scheme. And we propose a Call Server system, which is based on a phone-switch hybrid call control scheme in which the switch-oriented call control architecture of the Call Server is combined advantageously with a phone-oriented call control method. Also this web-based Call Server easily offers the best possible CTI platform for CTI web application users in PC. We develop the Call Server analytical model and implement the model with Java to verify performance of the phone-switch hybrid call control scheme. By using the implemented Call Server model, we investigate the effect of client environment parameters (the number of client PCs, call request generation rate) and server side parameters (the number of modems, waiting queue size, modem life cycle time) on performances of call connection fail rate and call connection time.

Ⅰ. 서 론

전화망에서 더욱 우수한 서비스를 제공해주기 위한 기술로는 지능망 IN(Intelligent Network) 기술과 컴퓨터 전화 통합 CTI(Computer Telephony Integration) 기술이 있다[1]. IN 기술은 전화망의 구성 요소들 간에 네트워크 지능을 서로 주고받으면서 공유하는 개념이며 인터넷인 IP(Internet Protocol) 네트워크과 구별되는 핵심 특징이다. 즉 IP 네트워크는 네트워크 지능이 거의 없고 반면에 단말인 호스트 컴퓨터가 지능을 대부분 갖는 시스템인데 비해 전화망은 전화 네트워크 구성 장비간에 IN 기술의

* 가톨릭대학교 컴퓨터 · 전자공학부
논문번호 : 00419-1030, 접수일자 : 2000년 10월 30일

강력한 지능이 동작하는 반면에 단말인 전화기는 지능이 거의 없는 상반된 개념의 네트워크이다. 두 번째의 CTI 기술은 컴퓨터와 인터넷 기술을 이용하여 전화망의 능력과 서비스 수준을 향상시키고자 하는 기술이다.

1970년대에 형성된 초기 개념의 CTI 기술의 목적은 디지털 PBX(Private Branch eXchange)를 원격에서 제어할 수 있는 메인 프레임 컴퓨터를 개발하기 위하여 태동되었다. 이후 1990년대에 들어와서는 PC 기술이 CTI 기술과 통합되면서 여러 가지 CTI 표준이 만들어졌다[2]. 최근의 CTI는 발달된 인터넷에 연결된 컴퓨터의 지능을 이용하여 전화 통화를 제어하는 목적으로 개발되고 있다. CTI 기술 개념의 큰 흐름은 초기에는 전화망의 핵심인 교환기를 원격 제어하기 위한 교환기 운영자를 위한 기능 중심이었으나 현재는 PC가 개인에게 보급되고 또 이들이 인터넷으로 모두 결합되어 운영되는 상태에서 고객 개인이 직접 자기 PC에서 전화망이 제공하는 지능을 제어하고 활용할 수 있게 해주는 형태로 변화되었다[3].

CTI 기능은 호 제어, 미디어 처리 및 고객 데이터 관리 영역의 3 가지로 나누어진다[3]. 첫 번째로 호 제어 기능이 하는 일은

- 다이얼링 기능을 수행하여 호를 설정하거나 해제하는 서비스
- 자동으로 호를 전환해주는 라우팅 관련 서비스
- 톤을 인식하고 또 발생시키는 일과 같은 인 밴드 시그널링 인식 및 처리 기능

두 번째로 미디어 처리 기능이 하는 일은

- 음성이나 팩스의 기록, 응답 및 전송 기능
- 문서의 음성 변환, 음성 인식 등
- 호의 모니터링 및 과금

마지막으로 개인 정보 관리 기능이 하는 일은

- 통화 요청자와 피요청자의 확인 정보에 대응하는 개인 정보 데이터베이스의 처리 기능을 활용하여 전화 고객을 관리

하는 일이다.

CTI의 운영 형태는 First-Party 호 제어라 불리는 전화 기반 호 제어 구조와 Third-Party 호 제어라고 부르는 교환기 기반 호 제어 구조로 나눌 수가 있다. 그림 1.1에 표시된 전화 기반 호 제어 방식은 교환기와 전화기 사이의 라인 시그널링 정보를 PC에서 탐지하거나 발생시켜 컴퓨터로 호 제어를 수행하는 형태이다. 그러므로 전화 기반 호 제어 방식으로 수행할 수 있는 기능은 교환기와 전화기에 부여된 라인 시그널링 기능까지만 호 제어를 할 수밖에 없는 한계를 가지게 된다. 이 방식을 활용한 대표적 산업화 예는 인텔과 마이크로소프트가 개발한 TAPI 규격이다[4].

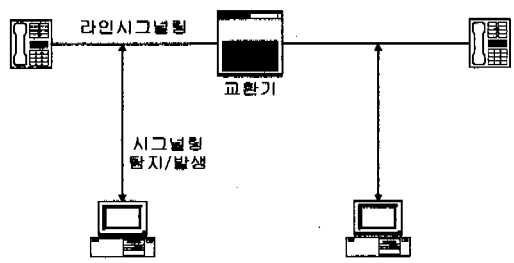

그림 1.1 전화 기반 호 제어 구조

그림 1.2에 나타낸 교환기 기반 호 제어 구조는 통화 요청자와 피요청자를 접속시키는 제 3의 장치(Third Party)가 필요하며 이 제 3의 장치와 교환기 간은 CTI 링크 프로토콜로 연결된다. 이 방식이 산업화된 예는 Novell사와 Lucent사를 중심으로 개발한 TSAPI 규격이 있다[5]. TSAPI로 개발된 CTI 응용 프로그램은 통화 요청자와 피요청자들을 접속시키는 제 3의 장치 역할을 한다. 그런데 그림 1.2에서처럼 이 제 3의 장치는 보통 LAN을 통하여 PC 클라이언트들에게 전화 기능을 확장해주기 위하여 클라이언트/서버 모델로 동작하게 되는데 이 이유로 이 제 3의 장치를 CTI 서버라고 부른다. PC 클라이언트가 아우트고잉(Out Going) 호를 접속하거나 인커밍(In Coming) 호를 인지할 수 있도록 제어 정보를 교환기에 보내거나 받기 위해서 CTI 서버를 제 3의 에이전트로 이용하는 구조이다.

이렇게 CTI 서버를 활용하는 교환기 기반 호 제어 방식은 각각의 데스크탑마다 CTI 하드웨어나 소프트웨어를 구비할 필요가 없기 때문에 주로 대형 사이트에 적합한 구조이다. 특히 이 구조는 CTI 서버와 PBX를 연결하는 CTI 링크가 CCS(Common Channel Signaling)로 이루어지기 때문에 CTI 서버가 전화망 PSTN의 지능을 더욱 효과적으로 이용할 수 있고, 이는 곧 클라이언트 컴퓨터에서 전화망의 지능을 더욱 적극적으로 활용할 수 있다는 사실을 의미한다. 그런데 이 CTI 링크를 구현하기 위해서

는 전화망의 핵심 기술인 CCS 시그널링 프로토콜을 이용할 수 있는 CCS 드라이버와 같은 역할을 CTI 서버가 해주어야 하는데 이는 원천적으로 전화망 입장의 고급 기술이기 때문에 컴퓨터나 인터넷 기술과 접목시키는 데 가장 어려움이 있는 인터페이스 기술이다. 현재도 컴퓨터 입장에서는 CTI 응용 프로그램을 개발하기 위하여 JAVA 언어를 이용한 JTAPI와 같은 API를 제공하지만 문제는 이 CTI 링크의 CCS와 JTAPI를 바인딩(Binding)시킬 수 있는 구현 기술이 아직 초기 단계에 있기 때문에 CTI 응용 분야의 발전이 인터넷 응용 분야 발전만큼 진척을 보지 못하고 있는 이유이다.

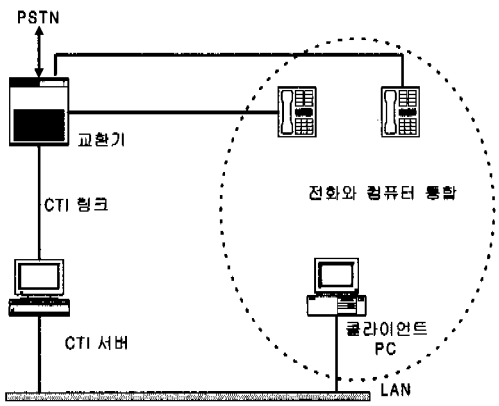

그림 1.2 교환기 기반 호 제어 구조

이 논문에서 전화-스위치 하이브리드 방법을 이용한 웹 기반 호 제어 방식을 제안하는 배경은 다음과 같다. 이 논문이 추구하는 바는 CTI 링크 처럼 교환망의 지능을 충분히 활용할 수가 있어 성능높은 CTI 기능을 제공하는 방식을 제안하는 것이 아니다. 그렇지만 PC에서 웹을 통하여 요청자와 피요청자 사이의 전화연결을 요청할 때, 이 동시다발 요청을 수신처리하는 콜서버는 여기서 제안하는 하이브리드 방식의 인터페이스를 통하여 인밴드 라인 처리 기능을 공통적으로 갖는 모든 종류의 교환기를 Third Party 자격으로 제어하여 요청자와 피요청자 전화가 접속될수 있게 한다.

예를 들어 어떤 요청자가 웹을 이용하여 제안하는 방식의 시스템을 구축하고 있는 사이트의 홈페이지로 들어와서 담당자를 클릭했다고 가정하자. 이때 이 사이트에서 일어나는 일은 요청자의 전화번호와 담당자의 전화번호를 분석한 후에 인밴드라인 인터페이스를 통하여 이 두가지 전화번호를 교환기로 넘겨준다. 그러면 교환기에서는 요청자와 담당자 전화를 연결시키게 된다. 이러한 시스템의 필요성은 회원제로 운영하는 사이트에서 고객이 사이트의 담당자를 웹에서 클릭하기만 하면 전화가 연결되는 응용이라든지 반대로 담당자가 자기고객에게 전화를 걸때에도 웹DB에 기록된 고객전화번호를 클릭하기만 하면 고객과 담당자간의 전화가 연결되는 등의 응용에 적용될 수 있게 하는 기술이다. 특히 이 논문이 제안하는 전화-스위치 하이브리드 방식은 인밴드라인 인터페이스를 사용하기 때문에 사이트의 구축교환기가 키폰과 같은 CTI링크가 없는 교환기인 경우에도 이런 시스템이 적용될 수 있다. 그래서 소형사이트에서도 기존의 시설(전화시설, 웹서버, LAN 등)을 가지고 최소한의 인밴드라인 인터페이스만으로 컴퓨터, 인터넷, 전화가 통합 이용될 수 있는 경제적환경을 제공하기 때문에 CTI 링크를 구비하지 않은 사이트를 목표로 전화-스위치 하이브리드 방식 시스템 기술을 제공하는 데 이 논문의 의의가 있다.

현재 개발된 대부분 CTI 서버들은 대형이고 고가인 콜 센터의 구현에 초점이 맞추어져 있다. 또 현재까지는 CTI 응용 프로그램이 콜 센터의 기능인 텔레마케팅, 부채 처리 및 전화 안내 등에 제한적 담당자만이 풍부한 기능을 이용할 수 있도록 운영되고 있다[6]. 그러나 CTI 기술의 다양한 사용자 접근성 측면에서 보았을 때 어떤 PC 사용자가 인터넷 상에서 필요한 정보를 HTTP 프로토콜에 의해 바로 웹형태로 볼려올수 있듯이, 웹상에서 상대방의 전화와 사용자 전화를 연결시킬수만 있다면 컴퓨터와 인터넷 및 전화가 쉽게 함께 이용될 수 있어 편리할 것이다. 이를 위해서는 CTI 응용 프로그램이 웹을 통하여 고객들이 언제든지 이용할 수 있게 하지 않으면 안된다. PC 고객은 허가된 CTI 서버에 웹으로 들어와서 CTI 서비스를 받는 웹 기반 호 제어 방식의 응용 프로그램이어야 한다. 웹기반은 인터넷에 연결된 모두에게 CTI서버의 접근을 허용하지만 CTI 서버 자체가 교환기와의 인터페이스 문제나 가격문제로 인하여 제한적으로 구축될수 밖에 없다면 클라이언트들이 보편적으로 CTI시스템을 이용할 수 있게 해주어야한다는 목표에서 벗어나게 된다. 그래서 CTI서버 구축사이트의 교환기 종류에 제한 받지않고 보편적으로 웹상에서 사용자와 상대방간의 전화끼리 호접속을 가능케하는 방법으로 전화-스위치 하이브리드 방식에 의한 콜서버를 이 논문에서 제안한다.

CTI 서버 호 제어 방식이 CCS 시그널링 신호를 모두 제어할 수 있기 때문에 강력한 CTI 응용을 개발해 낼 수는 있다. 뿐만 아니라 전화기 기반 호 제어 방식은 호 제어의 범주가 인 밴드 시그널링으로 제어할 수 있는 기능까지로 국한된다. 그러므로 CTI 성능면에서는 교환기 기반 호 제어 구조가 적합하다. 그런데 CTI 활용의 보편성과 접근성 측면에서 볼 때 교환기 기반 호 제어 구조는 교환기에 CTI 링크 프로토콜을 만족하는 기능을 구비해야 하고 또 CTI 서버도 이 프로토콜에 맞게 개발되어야 한다. 이것이 CTI를 제한적으로 이용할 수밖에 없게 만드는 요인이다. 통신 장비간에는 인터페이스가 명확하여 상호 호환성이 만족되어야 하나 교환기와 CTI 서버가 완전히 상호 매칭되기 위해서는 CTI 링크 기능이 있는 교환기 개발자가 거기에 의존하여 CTI 서버를 개발해야만 상호 동작이 원만하다는 사실이다. 또 CTI 서버의 응용 프로그램을 바인딩 시켜 줄 수 있는 API로 구현하기 위해서는 결국 벤더별 특정 교환기에 맞는 전화 제공자(provider)가 별도로 소프트웨어로 구현되어야만 한다[7]. 이는 인터넷 응용 프로그램이나 데이터베이스 이용 응용 프로그램에서 지향하고 있는 높은 이식 가능성(portability)이 CTI 응용 프로그램에서는 문제가 된다. 이와 같은 이유로 현재까지도 CTI 서비스가 개인 PC 영역까지 보편적으로 활용되지 못하는 주된 원인이다. 그래서 이 논문에서 제안하는 방식을 Third Party 호제어 콜서버가 수용했을 때 비록 고도의 CTI성능을 만족시키지는 못하지만 쉽게 호접속 중심의 제한된 웹기반 CTI 응용 프로그램을 개발할 수 있을뿐아니라 사용자도 웹을 클릭하기만하면 전화기간의 호접속이 쉽게 성사되는 시스템에 접근할 수가 있게 된다.

본 논문에서는 호제어 중심의 제한된 CTI 기능이 수행되지만 사용자접근을 용이하게 해주고 경제적인 전화-스위치 하이브리드 웹 기반 호 제어 방식에 입각한 CTI 콜서버 시스템을 제안하고 이 방식의 성능을 분석해본다.

Ⅱ. 전화-스위치 하이브리드 웹 기반 호 제어 방식에 입각한 시스템 제안

2.1 전화-스위치 하이브리드 호 제어 방식

그림 2.1은 이 논문에서 제안하고 있는 전화-스위치 하이브리드 호 제어 방식 구성도이다. 앞장에서 설명한 기존의 전화 기반 호 제어 구조와 교환기 기반 호 제어 구조를 하이브리드해 놓은 형태이다. 먼저 여러 클라이언트 PC로부터의 호 접속 요구를 제 3자가 대행하는 장치로 콜 서버를 사용하고 있다는 점은 교환기 기반 호 제어 구조와 같은 방식으로 동작한다. 그러나 교환기와 콜 서버 사이의 인터페이스 방식을 보면 제안하는 구조에서는 인 밴드 시그널링 라인을 사용하고 있어 오히려 전화 기반 호 제어 구조와 같다고 볼 수 있다.

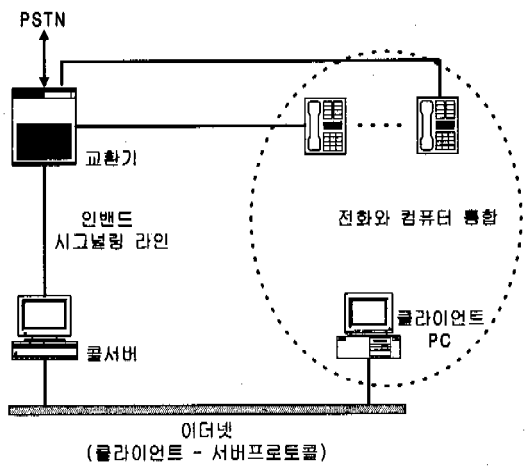

그림 2.1 전화-스위치 하이브리드 호 제어 방식

제안하고 있는 이 하이브리드 방식은 콜 서버와 클라이언트 PC 사이는 서로 교환기 기반 호 제어 구조와 동일한 기능을 발휘하게 되어 있으나 문제가 되는 것은 교환기와 콜 서버간의 인터페이스이다. 교환기-콜 서버 인터페이스는 일반 전화기 라인과 같은 인 밴드 시그널링 라인을 사용하기 때문에 콜 서버와 교환기 사이에 호 제어를 위하여 사용할 수 있는 기능이 인 밴드 시그널링 기능 정도로 제한될 수밖에 없다.

그렇다면 교환기-콜 서버 인터페이스의 인 밴드 시그널링 한계를 극복하고 콜 서버가 Third-Party 호 제어 업무를 제 3의 에이전트 자격으로 수행할 수 있는 방법이 추구되어야 한다. 아우트고잉 콜에서 Third-Party 호 제어 업무 중에서 가장 기본이 되는 기능은 통화 요청자 전화 번호와 피요청자 전화 번호를 포함하는 접속 요구 메세지가 통화 요청자 클라이언트 PC로부터 콜 서버에 클라이언트/서버 형태로 도착했을 때 콜 서버는 도착 메세지를 분석하여 통화 요청자 전화 번호와 피요청자 전화

번호를 확인한 후 교환기-콜 서버 인터페이스를 통하여 인 밴드 시그널링의 형태로 호 제어 업무를 수행해야만 Third-Party 호의 설정이나 해제 업무를 수행할 수 있게 된다. 그러면 인 밴드 시그널링의 기능으로 Third-Party 호의 설정을 할 수 있는 방법이 필요한 데 이 논문에서는 거의 대부분의 PBX가 특징적으로 제공하고 있는 내선 서비스인 발신 신호 전송 서비스 기능을 Third-Party 호 설정 기법으로 이용하는 것을 제안한다. 이 인 밴드 시그널링 Third-Party 호 설정 기법은 원래는 PBX 제조 회사들이 전화기에서 통화를 수동으로 전환시켜 줄 수 있도록 제공하는 기능이다. 그림 2.2는 콜 서버가 클라이언트로부터 요청 받은 통화 요청자 전화 번호와 피요청자 전화 번호를 가지고 교환기 입장에서 호전환 서비스를 수행할 수 있게 인밴드시그널링방식으로 요청자와 피요청자 전화번호를 교환기에 넘겨주는 절차를 설명하고 있다. 물론 여기서 수화기를 든다거나 후래싱(flashing)을 한다고 표현한 부분들은 콜 서버 프로그램이 제어한다. 콜 서버가 Third-Party 호 제어 대행 업무를 종료하게 되면 콜 서버 안에는 방금 설정한 호의 상태를 가지고 있지 않는 State-less 호 설정 방식이다.

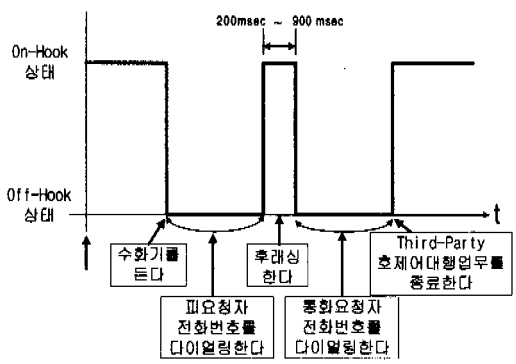

그림 2.2 인 밴드 시그널링 Third-Party 호 제어 기법

2.2 콜 서버/클라이언트 PC의 웹 기반 호 제어

그림 2.1에서와 같이 콜 서버와 클라이언트 PC들과는 물리적으로는 이더넷으로 연결되어 있고 클라이언트-서버 프로토콜에 의해 동작하게 되어 있다. 그림 2.3은 콜 서버가 인 밴드 방식으로 교환기와 연결되고 또 사용자 입장에서 콜 서버 이용의 접근을 편리하게 해주기 위하여 클라이언트-서버간의 프로토콜을 JAVA 애플릿과 서블릿을 이용하게 되어 사용자들은 웹으로 콜 서버에 쉽게 접근하여 호 접속을 콜 서버에 일임할 수가 있게 해주는 콜 서버 구성을 보여준다. 클라이언트들의 호제어 요구는 애플릿 프로그램으로 확인되고 이를 JAVA 서브릿으로 작성된 콜서버 응용프로그램이 해석하여 전화-스위치 간의 인터페이스를 담당하는 모뎀제어 프로그램과 소켓통신을 열고 업무를 넘겨준다. 이 업무를 넘겨받은 모뎀제어 프로그램은 모뎀카드로 업무에 해당하는 AT 명령어 제어신호를 보내어 모뎀카드가 관련 시그널링 정보를 교환기로 송출하도록 한다. 이와 같이 인밴드 라인을 통하여 신호를 받은 교환기는 결국 웹기반으로 PC 사용자가 지시한 업무를 수행할 수 있게 된다.

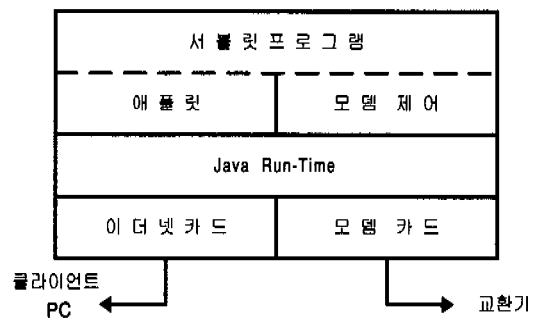

그림 2.3 인 밴드 시그널링 웹 기반 콜 서버 구성

2.3 CTI 서버와 콜 서버의 수행 특성 비교

이 논문에서 제안하고 있는 콜 서버가 다루는 호 제어 기법은 기존의 어떤 교환기도 이용 가능하고 또 클라이언트 PC로부터 사용자 입장에서는 웹으로 콜 서버를 이용하기 때문에 사용 편의성면에서 콜 서버가 편리한 장점이 있다. 그러나 호가 하나 만들어지고 접속이 해제될 때까지의 상황(state)을 다루는 방법에서 근본적인 차이가 있다.

보통 CTI 서버에서는 CTI 서버가 Third-Party 호를 다루면서 이 호에 대한 호 상태(Call State) 정보를 유지하고 있게 된다. 그림 2.4에서 보여주고 있는 것과 같이 하나의 호 접속이 만들어지려면 먼저 새로운 호 접속 객체를 생성한다. 이 초기 상태가 IDLE 상태이다. 다음으로 피요청자 전화로 시그널링이 송출 중인 상태가 시그널링 수행 중 상태가 되며, 피요청자 전화기에 신호음이 울리는 상태가 경계 상태가 된다. 그리고 피통화자가 전화를 받으면 접속 상태가 된다. 이 상태에서 통화가 이루어지며 한 쪽에서 전화를 끊으면 접속 해제 상태가 된다. 만약 통화 접속이 실패되는 호 접속에 대해서는 모두 접속 실패 상태로 처리된다.

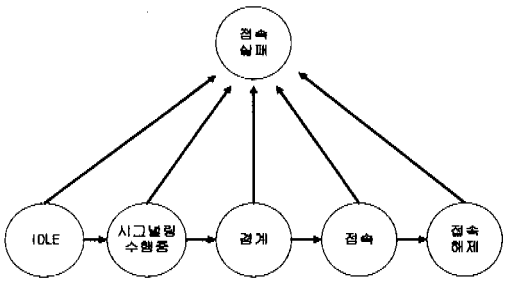

그림 2.4 CTI 서버에서의 호 상태 천이

이와 같은 호 접속 상태 정보는 교환기로부터 CTI 링크를 통하여 교환기의 상황에 맞게 실시간으로 CTI 서버에 전달되며, CTI 서버에 있는 호 접속 객체는 이 교환기의 호 접속 상태 정보를 항상 유지하고 있다. 이러한 상태 정보를 클라이언트 PC로 전달하여 사용자가 접속된 호의 전체 라이프 사이클동안 개인 PC로부터 상황 정보를 얻을 수가 있다. 결과적으로 호 제어를 수행하는 어플리케이션 프로그램의 기능 영역이 그만큼 넓어질 수 있다는 사실이 된다.

이에 비하여 콜 서버는 아웃고잉 호 설정에 대한 Third-Party 임무만을 수행해준다. 하나의 아웃고잉 호 접속에 대한 라이프 사이클을 살펴보면 다음과 같다. 그림 2.5에서 표현된 것과 같이 애플릿으로 동작하는 클라이언트 PC에서 입력되거나 DB에서 찾아서 생성된 요청자와 피요청자 전화번호로 구성된 호접속 위임메세지가 콜 서버에 도달하면 콜 서버에서는 하나의 호 접속 객체가 초기 만들어진다. 이 상태가 IDLE 상태이며 모뎀을 통하여 그림 2.2에서 보인 바와 같이 피요청자 전화번호 다이얼링 과정을 거치면서 바로 시그널링 수행 중 상태로 들어가며, 여기서 피요청자 전화기로부터 정상적 신호음이 들리면 바로 후래싱 처리를 동반한 통화 요청자에게 시그널링이 수행된다. 이 과정이 정상적으로 수행되었으면 곧바로 이 호 접속 객체를 소멸시키게 된다.

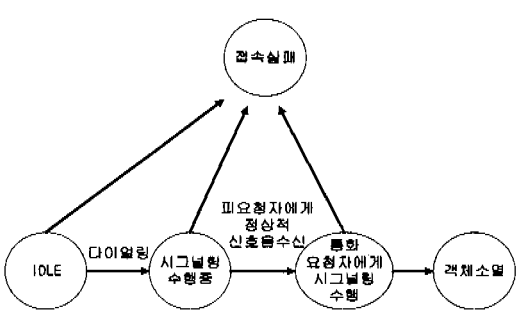

그림 2.5 콜 서버에서의 호 상태 라이프 사이클

이상과 같이 콜 서버에서 하나의 접속 객체를 다루는 방법은 통화 중에 호 접속 객체가 유지되느냐 여부로 판단할 때 State-less 호 제어 방식이라 볼 수 있다. 즉 콜 서버에서의 호 접속 상태의 라이프 사이클은 짧은 순간에 끝나게 되는데 대략적으로 피요청자에게 인 밴드 다이얼 신호가 송출되는 시간, 후래싱 소요 시간, 그리고 통화 요청자에게 인 밴드 다이얼 신호가 송출되는 시간의 합으로 표현될 수 있다. 이 시간은 보통 4~5초 정도에 해당하는 시간이기 때문에 콜 서버에서는 클라이언트로부터 호 접속 위임이 오면 호 접속 객체가 하나 만들어지고 4~5초 후에 바로 객체가 소멸되는 형태로 동작하게 된다[8].

CTI 서버와 콜 서버 안에서의 호 접속 객체의 라이프 사이클을 비교해 보면 콜 서버가 Third-Party 호 제어를 서비스하지만 호 접속 상태의 유지 없이 State-less 개념으로 동작하기 때문에 콜 서버 작동 신뢰성이 높아짐을 알 수 있다. 또 하나 하나의 호 접속 상태의 라이프 사이클이 짧기 때문에 여러 클라이언트로부터 동시다발적으로 위임하는 호 접속 요구 메세지가 콜 서버에 도착하더라도 요구 메세지를 큐잉 처리하여 호 접속 서비스 이용률을 높일 수가 있다. 또 모뎀을 통한 호 접속 상태가 라이프 사이클 동안만 짧게 유지되기 때문에 결과적으로 작은 수의 모뎀으로도 충분히 콜 서버를 동작시킬 수 있다는 사실을 직관적으로 알 수 있다.

이와 같이 본 논문에서 제안하는 전화-스위치 하이브리드 호 제어 방식으로 동작하는 콜 서버의 장점을 정량적으로 분석해보기 위하여 클라이언트들의 통화 요청 빈도, 대기큐의 길이 및 모뎀 수 등을 파라미터로 하여 분석 모델을 만들고 이를 콜 서버로 구현하는 JAVA 프로그램을 개발하였다. 아울러 콜 서버에 접근하는 클라이언트들의 기능을 에뮬레이션할 수 있는 클라이언트 JAVA 프로그램을 개발하였다.

Ⅲ. 콜 서버 모델 구현 및 성능 분석

3.1 콜 서버 분석 모델의 설계

콜 서버의 동작을 다수 서비스 처리를 하는 하나의 큐잉 모델로 간주한다. 그림 3.1에서와 같이 여러 클라이언트 PC로부터 출발하여 콜 서버에 도달하는 통화 접속 요구 위임 메세지의 도착 프로세스는 포아송 도착 프로세스를 따르는 것으로 가정하

고 시간당 통화 접속 요구 메세지의 도착 비율을 λ로 나타내었다. 그리고 콜 서버가 동시에 처리할 수 있는 호 접속 처리의 수, 즉 동시에 존재하는 호 접속 처리 객체의 최대수가 되며 또 모뎀의 총 갯수이기도 한 수를 K_{num}로 정의하였다. 또 통화 접속 요구 메세지가 콜 서버에 임의로 도달하므로 대기 큐가 필요하게 된다. 호 접속 처리 과정이 객체화되어 모뎀을 배당 받아 호 접속 처리되기까지는 대기 큐에서 순서를 기다리고 있어야 하며 제한된 수의 대기 메세지만을 허용한다. 제한된 대기큐의 길이를 Q_{size}로 나타내었다.

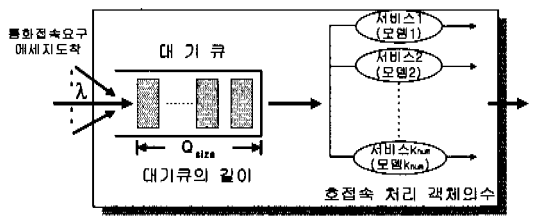

그림 3.1 콜 서버의 큐잉 모델화

3.2 클라이언트 모델링 및 구현

그림 3.1과 같이 설정된 콜 서버 큐잉 모델을 구현하기 위하여 네트워크에 연결된 2대의 PC에 각각 Call_Server 프로그램과 Call_Client 프로그램을 구현하였다. 그림 3.2에서 나타낸 것과 같이 여러 클라이언트들이 발생시키는 통화 접속 요구 메세지들 사건을 모델화해서 하나의 클라이언트 PC가 Call_Client.java 프로그램을 실행하면서 사건 발생이 일어나도록 하였다. 여기서 클라이언트 JAVA 프로그램이 동작하는 환경은 실재로 인터넷을 그대로 이용하였다. 즉 LAN의 MAC계층 프로토콜과 상위의 IP프로토콜 및 TCP프로토콜은 실재 인터넷 환경에서 동작시켰다. 단지 한 대의 클라이언트 PC가 여러대의 PC들 상황을 발생시키고 있기 때문에 엄밀히는 동시에 통화접속요구 메시지가 발생하지 않는다. 그러나 메시지 발생모델이 포아송 도착 프로세스를 따르도록 했기 때문에 통화접속요구 메시지의 발생사건만 고려한다면 MAC계층이 할 일은 일어나지 않는 모델이다.

Call_Client.java로 구현된 통화 접속 요구 메세지 발생 사건의 모델은 다음과 같다. 클라이언트 PC에서 호 접속 요청 패킷을 발생시킨다. 하나의 요청 패킷이 발생되면 바로 서버 PC로 소켓을 통한 TCP 콘넥션이 만들어지고 바로 요청 패킷이 서버 PC로 전달된다. 그리고 서버에서 호 접속 처리가 완료되면 TCP 콘넥션을 해제한다. 콜 서버 모델을 직접 구현하기 위하여 클라이언트 PC는 아래와 같이 호 접속 요청 패킷을 발생시킴으로써 콜 서버에 도착하는 호 접속 요청 패킷들이 포아송 도착 프로세스에 근접하게 한다[9]. 호 접속 요청 패킷의 발생은 시간을 타임슬롯이 연속으로 이어지는 discrete time space로 가정한다. 그래서 실재 상황에서 특정 회사에 N_{phone}개의 총 전화대수(=총 클라이언트 PC대수)가 있다고 가정하고 각각의 클라이언트 PC가 임의의 타임슬롯 시간 안에 호 접속 요청을 서버로 보낼 확률을 $Call_{prob}$라고 정의한다. 그러면 전체 N_{phone}개의 클라이언트로부터 특정 타임슬롯 안에 k개의

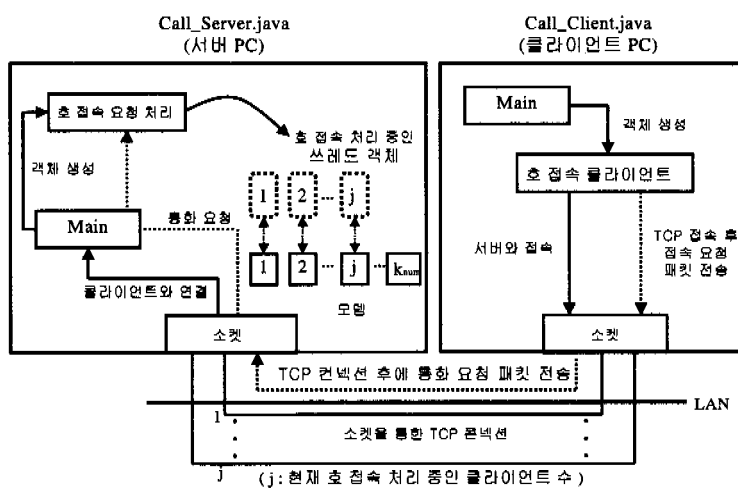

그림 3.2 콜 서버 모델의 구현

접속 요청이 이루어질 갯수의 분포 $G_{call}(k)$는 이항 분포를 따른다고 가정하면,

$$G_{call}(k) = \binom{N_{phone}}{k} Call_{prob}^{\ k} (1 - Call_{prob})^{N_{phone}-k},$$
$$k=0, 1, 2, \cdots, N_{phone} \quad (1)$$

이다. 이 $G_{call}(k)$ 분포에 입각하여 Call_Client 프로그램에서 특정 타임슬롯에 k개의 호 접속 요청 패킷을 확률적으로 발생시켰다. 그리고 이 특정 타임슬롯 안에 k개의 TCP 콘넥션이 서버 PC로 설정되게 프로그램 하였다.

3.3 콜 서버 구현

그림 3.2의 왼쪽은 서버 PC에 클라이언트 PC로부터의 호 접속 요청 패킷을 TCP 콘넥션으로 받아 처리하는 Call_Server.java 프로그램을 구현하였다. 클라이언트로부터 TCP 접속 요구가 오면 서버 소켓이 작동하여 하나의 콘넥션 객체를 생성하고 바로 이 객체가 호 접속 요청 패킷을 받아 처리하게 된다. 호 접속 요청 처리부에는 Q_{size}개만큼의 요청 패킷이 서비스를 기다리며 대기할 수가 있다. 대기중인 호 접속 요청 패킷들은 FIFO(First In First Out) 개념으로 하나씩 그림 2.5의 호 접속 처리를 다루는 라이프 사이클로 들어가게 된다. 여기서 콜 서버에는 K_{num}만큼의 모뎀이 있으므로 동시에 최대 K_{num}개의 호 접속 처리를 할 수가 있다. 그리고 모뎀이 호 접속 처리를 하는데 걸리는 시간, 즉 그림 2.5에서 표현한 라이프 사이클 시간은 M_{time}으로 정의하였으며 본 서버 구현에서는 M_{time}값을 악조건 한계(pessimistic bound)를 고려하여 충분히 큰 값인 10초로 고정하였다. 참고로 서버모델은 클라이언트로로부터 오는 접속요구를 처리하는 인밴드 라인수 즉 모뎀 수를 어느정도 까지만 콜서버에 설치하면 될것인가에 초점을 두고 실험결과를 분석하기 위한 모델이다. 그래서 호접속 실패율을 알아보는데 있어 서버측 사이트의 전화기가 현재 통화중임에 따른 호접속실패는 제3의 장치로서의 기능을 수행하는 서버의 성능분석에서는 무시되었다.

Ⅳ. 성능 분석 결과

Call_Server와 Call_Client 프로그램으로 구현한 콜 서버 모델을 수행시켜서 클라이언트들의 환경조건인 전체 클라이언트 PC대수 N_{phone}, 클라이언트 PC 각각의 타임슬롯 안에서의 호 접속 요청 발생 확률 $Call_{prob}$ 값을 변화시키면서, 또 서버에서의 동시 처리 모뎀수 K_{num}, 대기큐의 크기 Q_{size} 및 모뎀의 접속 처리 라이프 사이클 M_{time} 값을 파라메터로 하여 콜 서버에서의 호 접속 처리 실패율 및 평균 대기 시간을 측정하였다. 여기서 호 접속 처리 실패율은 제한된 Q_{size}에 따른 서버 용량 초과로 인한 실패율이며 실험에서는 [실패한 호 접속 처리수/총 호 접속 요구 발생수]로 결과를 얻었다. 또 평균 대기 시간은 모뎀에서의 처리 시간을 제외한 대기큐에서의 대기 시간을 모든 접속 처리된 호에 대하여 측정한 후 평균값을 취하였다. 참고적으로 클라이언트가 호 접속 요구를 발생한 후에 서버에서 호 접속 처리가 완료되는 전체 시간은 [TCP 콘넥션 설정 시간(약 0.4초) + 대기큐에서의 평균 대기 시간 + 모뎀의 접속 처리 라이프 사이클 시간(M_{time}=약 5초)]로 표현된다. 즉 아래에서 그림으로 보여주는 평균 대기 시간에 약 5.4초 정도를 합한 값이 총 호 접속 처리 소요 시간이 된다. 특히 이 분석 실험에서는 하나의 타임슬롯 주기를 20분으로 가정하였다. 이에 따라 $Call_{prob}$가 0.4라고 하는 것은 임의의 클라이언트 PC가 콜 서버로 20분마다 평균 0.4회의 빈도로 호 접속 요구 패킷을 보낸다는 의미이다. 또 이 분석 실험을 수행시켜 얻은 하나의 데이터는 실제 상황에서 6개월 동안 관측해야 얻을 수 있는 것을 시간 개념을 1/10000로 축소하여 약 0.5시간 실험으로 결과를 얻었다. 즉 실제 상황에서의 타임 슬롯이 20분 주기인데 비해 이 실험에서의 타임슬롯 주기는 120msec로 처리하였다. 그렇다고 해도 다음 결과로서 제시하는 실험 데이터는 그림마다 각각 0.5시간×16개 데이터값 = 8시간씩의 많은 컴퓨팅 처리 시간이 요구되었다.

그림 4.1은 클라이언트 각각이 20분마다 0.2회의 평균 호 접속 요구를 하는 경우 콜 서버에서의 대기큐 크기가 6인 경우에 호 접속 실패율을 보여주고 있다. 여기서 찾아진 중요한 사항은 서버에서 동시 호 접속 처리를 위한 모뎀수가 2개만 되면 전체 클라이언트 PC수가 100개 정도 이하인 규모의 사이트에서는 거의 100% 호 접속 성공률을 보인다는 점이다. 즉 CTI을 이용하는 클라이언트 PC가 100개 정도인 사이트에서 본 논문이 제안하는 방식의 콜 서버를 구축할 때 교환기와 콜 서버를 연결하는 인 밴드 시그널링 라인의 수가 2개이면 만족된다는 사실을 입증한 결과이다.

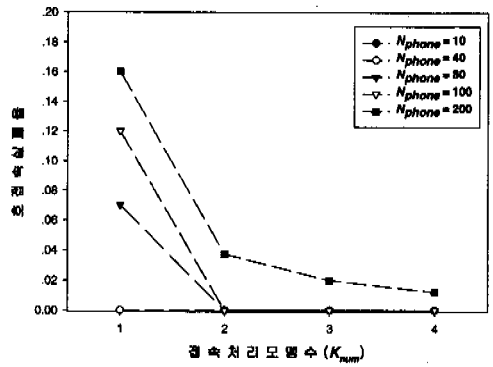

그림 4.1 호 접속 실패율(클라이언트 $Call_{prob}$=0.2, 콜 서버 Q_{size}=6)

그림 4.2는 그림 4.1과 동일한 클라이언트, 콜 서버 환경 조건에서 임의의 호 접속 요구 패킷에 대한 서버 대기큐에서의 평균 대기 시간을 표현하였다. CTI 이용 클라이언트 PC의 수가 100개 정도인 사이트에서 모뎀수가 2개인 콜 서버를 운영하면 서버에서의 평균 대기 시간이 12msec 정도임을 알 수 있으며, 여기에 M_{time}=5sec, TCP 콘넥션 설정 시간=0.4sec를 고려하면 클라이언트가 호 접속 처리를 요구하여 서버가 호 제어 임무 대행을 완료하는 데까지 걸리는 시간이 약 5.5초 정도에 완료됨을 알 수 있다. 이는 클라이언트 PC에서 웹을 통하여 호 접속 요구 버튼을 클릭한 후 약 5.5초 동안 블럭킹 되어 있다가 접속 처리 성공 메세지가 클라이언트 애플릿으로 돌아온다는 것을 의미한다.

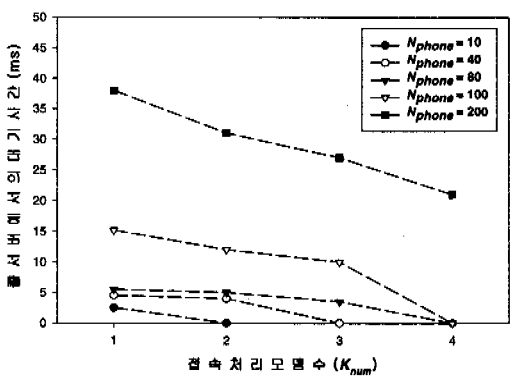

그림 4.2 콜 서버 대기큐에서의 평균 대기 시간(클라이언트 $Call_{prob}$=0.2, 콜 서버 Q_{size}=6)

그림 4.3은 $Call_{prob}$ 값을 0.4로 설정하고 다른 환경 조건은 그림 4.1, 그림 4.2와 같은 조건으로 실험하였다. 임의의 클라이언트 PC에서 하나의 타임 슬롯동안 호 접속 요구 발생 빈도가 0.4회라는 조건이므로, 예를 들어 그림 4.1에서 N_{phone}=100인 경우보다 그림 4.3에서 N_{phone}=100인 경우가 2배 더 빈번하게 호 접속 요구가 클라이언트들로부터 발생한다는 것을 나타낸다. 이 그림에서 알 수 있는 것은 20분마다 0.4회 정도의 CTI 호 접속을 이용하는 클라이언트 PC가 40대 정도 있는 사이트에서는 그림 2.1의 인 밴드 시그널링 라인수가 두 개 정도로 충분하지만, 만약 CTI 이용 클라이언트 PC가 80대가 넘어서면 인 밴드 시그널링 모뎀수가 두 개로는 부족하다는 점이다. 그러나 모뎀수를 4개로 하여 그림 2.1 시스템을 구축하면 CTI를 이용하는 클라이언트 PC수가 200개 정도의 사이트는 충분히 지원할 수 있다는 사실을 보여주고 있다.

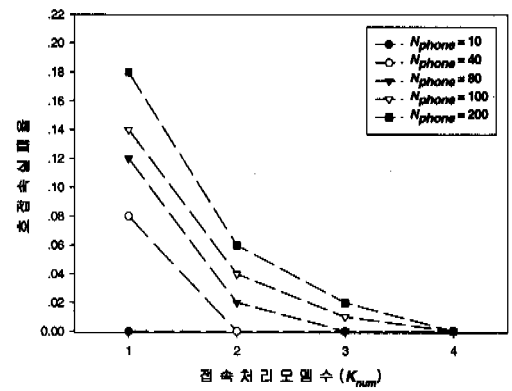

그림 4.3 호 접속 실패율(클라이언트 $Call_{prob}$=0.4, 콜 서버 Q_{size}=6)

그림 4.4는 그림 4.3과 동일한 클라이언트, 콜 서버 환경 조건에서 콜 서버 대기큐에서의 평균 대기 시간을 표현하였다. 클라이언트 PC수가 100개이고 모뎀수가 2개인 콜 서버를 운영하면 서버에서의 평균 대기 시간이 약 55msec정도가 됨을 알 수 있다.

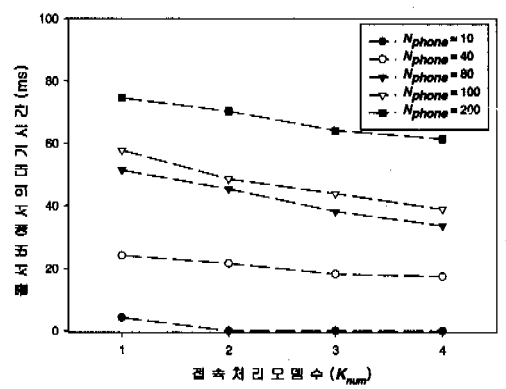

그림 4.4 콜 서버 대기큐에서의 평균 대기 시간(클라이언트 $Call_{prob}$=0.4, 콜 서버 Q_{size}=6)

이것은 모뎀에서의 호 처리 라이프 사이클 시간인 5sec에 비해 상대적으로 작은 값이기 때문에 전체적으로 클라이언트가 느끼기에는 하나의 CTI 호 처리 완료를 위하여 그림 4.2와 비슷한 총 5.5초 정도의 시간이 소요됨을 알 수 있다.

V. 결 론

전화망을 더욱 편리하게 이용할 수 있게 해주는 CTI 기술을 전화 기반 호 제어 구조와 교환기 기반 호 제어 구조로 분류하여 구조적으로 분석하였다. 그리고 쉽게 구축할 수 있는 전화 기반 호 제어 구조의 편리성과 Third-Party 호 제어 대행 업무를 효율적으로 처리할 수 있는 교환기 기반 호 제어 구조의 장점을 복합적으로 추구하는 전화-교환기 하이브리드 호 제어 방식에 입각한 콜 서버 시스템을 제안하였다. 특히 이 시스템은 클라이언트들이 CTI 기능을 대중적으로 쉽게 접근할 수 있도록 웹 기반 호 제어 방식으로 제안하였다. 그리고 이 전화-스위치 하이브리드 호 제어 방식의 콜 서버 수행 특성 및 장점을 분석해보기 위하여 콜 서버 분석 모델을 설정하고 콜 서버 모델을 JAVA로 구현했으며, 이 구현된 시스템을 이용하여 특정 사이트의 클라이언트 PC 대수와 호 접속 요구 발생 확률을 클라이언트 환경 변수로 하고 콜 서버의 동시 처리 모뎀수, 대기큐의 크기, 모뎀의 접속 처리 라이프 사이클을 서버 환경 변수로 하여 실재 상황에 준하는 분석 실험을 수행하였다.

분석 결과 이 논문에서 제안하고 있는 콜 서버와 교환기를 연결하는 모뎀을 이용하는 인 밴들 시그널링 라인수를 2개만 배당하면 CTI 기능을 이용하는 클라이언트 PC수가 100개 정도인 사이트는 지원할 수 있는 콜 서버 시스템이 될 수 있다는 사실을 확인하였다. 이처럼 콜 서버와 교환기 사이를 모뎀으로 연결하여 Third-Party 호 제어를 수행시키면서도 중형 규모의 사이트까지 2개의 모뎀으로 충분한 이유는 이 논문에서 제안하는 하이브리드 호 제어 방식 콜 서버가 근본적으로 State-less 호 제어 방식이기 때문에 가능한 일이다.

또 이 연구 결과에 입각하여 성능이 입증된 콜 서버 시스템을 구현하는 개발 프로젝트를 현재 수행 중이다.

참 고 문 헌

[1] T.A.Anschutz, "A Historial Perspective of CSTA," *IEEE Commun. Mag.*, Apr. 1996, pp. 30-35.

[2] K.Asatani, "Standardization on Multimedia Communications: Computer-Telephony-Integration-Related Issues," *IEEE Commun. Mag.*, vol. 36, no. 7, 1998, pp. 105-109.

[3] Sheng-Lin Chou, Yi-Bing Lin, "Computer Telephony Integration and Its Applications," http://www.comsoc.org/pubs/surveys/lgooissue/lin.html

[4] T.Nixon, "Design Considerations for Computer-Telephony Application Programming Interfaces and Related Components," *IEEE Commun. Mag.*, Apr. 1996, pp. 43-47.

[5] P.Cronin, "An Introduction to TSAPI and Network Telephony," *IEEE Commun. Mag.*, Apr. 1996, pp. 48-54.

[6] Java Telephony API, "An Overview," http://chatsubo.javasoft.com/products/jtapi/jtapi-1.2/Overview.html

[7] JTAPI, "An Introduction to JTAPI," http://chatsubo.javasoft.com/products/jtapi/jtapi-1.2/JTAPIWhitePaper_0_7.html

[8] Acacia Net, "Hayes Command Set," http://www.acacia-net.com/Clarinet/Protocol/v25_0tt0.htm, 1998.

[9] 정윤찬, 홍영진, "시간 지연을 최소화한 CAM형 트래픽 폴리싱 장치 설계", 한국통신학회논문지 '00-4, Vol. 25, No. 4B, pp. 604-612, 2000년 4월.

정 윤 찬(Youn Chan Jung) 정회원

1980년 2월 : 경북대학교 전자공학과(공학사)
1991년 2월 : KAIST 전기 및 전자공학과(공학석사)
1994년 8월 : KAIST 전기 및 전자공학과(공학박사)
1980년 3월~1990년 2월 : 국방과학연구소 선임연구원
1996년 3월~현재 : 가톨릭대학교 컴퓨터 · 전자공학부 조교수, 전자계산소장

<주관심 분야> ATM 교환구조, 기가라우터 구조, CTI, VoIP QoS, JAVA 네트워크 프로그래밍

함 미 옥(Mi Ok Ham) 정회원

1991년 2월 : 가톨릭대학교 수학과 졸업(이학사)

2000년 8월 : 가톨릭대학교 수학과 전산학전공 (이학석사)

2000년 9월~현재 : 가톨릭대학교 컴퓨터 · 전자공학부 강사

<주관심 분야> 네트워크 프로그래밍, 영상처리