특집

CSTA와 TAPI의 기술 분석

김 현 규, 김 병 수, 강 환 종
LG 정보통신

I. 서 론

CTI(Computer Telephony Integration)는 PBX와 컴퓨터의 연동에 의해 전화기나 PBX의 기능 및 조작을 연동 수행함으로써 전화 및 정보 서비스를 보다 손쉽게 하기 위한 수단의 하나로 1990년대에 들어서면서 여러 표준화 작업이 진행됨에 따라 점차로 활성화되고 있다. 초기의 CTI 연구 및 제품 개발은 메인프레임이나 미니 컴퓨터를 비롯한 대용량 시스템들에서 비싼 가격으로 이루어졌으나 1990년대 중반으로 들어서면서 현재에 이르기까지 점차로 싼 가격에서 개발이 쉽게 이루어지고 있다.

위에서 언급한 CTI 구현상의 발전은 다음과 같은 두 단계의 과정을 거쳐 이루어졌다. 첫째로, 1990년대 초 ECMA(European Computer Manufacturers Association)로부터 CSTA(Computer Supported Telecommunications Applications) 표준을 수용한 것이며, 둘째로는 다양한 API(Application Programming Interface)가 개발되었고 지금까지도 계속적으로 개발이 이루어지고 있다는 점이다. 현재 널리 알려진 API로는 Microsoft와 Intel이 공동 개발한 TAPI (Telephony API), Lucent와 Novell이 공동 개발한 TSAPI(Telephony Services API), IBM의 CallPath 등이 있다. 이 중 TAPI는 국부 전화선의 데스트탑 제어를 지원하는 데스크탑 중심형으로 개발되었으며 TSAPI와 CallPath는 전체 전화 시스템의 서버 제어를 지원하는 서버 중심형의 API이다. 최근까지도 많은 교환기나 컴퓨터 제조업체들이 자체적인 인터페이스를 사용하여 범용성에 문제가 있으나 대부분 위에 언급한 표준 및 API를 지원하게 될 것이며 소프트웨어 제공업자들도 이를 따르게 될 것이다[1].

본 고에서는 CSTA와 TAPI 프로토콜의 전반적인 기술 사항을 언급한다. CSTA를 기반으로 하여 개발된 대표적인 프로그래밍 인터페이스는 Novell의 TSAPI가 있으며 최근에는 Versit에서 개발 환경에 독립적인 CTI 응용 프로그램이나 네트워크

제품 개발을 목적으로 CTI Encyclopedia라는 규격서를 작성하여 소개하고 있으며 CSTA 프로토콜을 기본으로 한 통합 규격서이다. Versit은 Apple, IBM, Lucent 및 Siemens에 의하여 구성되었고 통신과 컴퓨터 업계의 여러 업체간의 주도적 의안을 개발하고 있다. CSTA Phase III에서는 ECMA와 Versit이 공동 개발을 계획하고 있다. 본 고에서 CSTA 프로토콜은 ECMA-217, 218과 Versit CTI Encyclopedia를 기본으로 하고, TAPI 프로토콜은 Microsoft Windows 3.1 Telephony Service Provider Programmer's Guide와 TAPI 2.0을 기본으로 하여 언급한다. II장에서는 CSTA 프로토콜의 전반적인 사항을, III장에서는 TAPI 프로토콜의 전반적인 사항을 언급한다. IV장에서는 CSTA와 TAPI 프로토콜을 간단히 비교하고 V장에서 결론을 맺고자 한다.

II. CSTA 기술 분석

위에서 기술한 바와 같이 CSTA 프로토콜은 ECMA에 의해서 개발되고 있다. 1992년 ECMA-179, 180을 통하여 Phase I이 발표되었고 1994년에는 ECMA-217, 218로 이루어진 Phase II가 발표되었으며 지금까지도 계속 연구 및 개선되고 있다. 또한 현재 Phase III의 연구가 진행 중이다. CSTA 인터페이스는 OSI 계층 7인 응용 계층의 프로토콜로 개발되었기 때문에 다양한 사용자-망 인터페이스 및 망-망 인터페이스와는 무관하도록 구성되었다[2]. Phase I에서는 본 장에서 언급될 대부분의 서비스 및 프로토콜을 정의하고 있고 Phase II에서는 Phase I을 바탕으로 다음과 같은 영역을 확장했다[3].

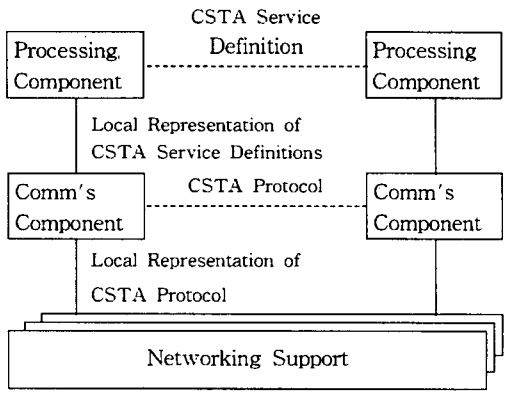
- 응용간의 협상 기법 추가
- I/O 서비스 추가
- 특별한 Resource Function 및 Voice Unit 서비스 추가
- 서비스와 Event Report 기능 개선

최근 ECMA의 CSTA 개발 팀은 Versit 팀과 연합하고 Versit CTI Encyclopedia를 사용함으로써 Phase III의 연구에 박차를 가하고 있다. Versit CTI Encyclopedia는 CSTA Phase II를 기반으로 기능 구조, 동작 모델, 서비스, 프로토콜 및 구현 가능한 환경에 이르기까지 통합된 규격을 정의하여 플랫폼간의 장애를 해결하고 있으며 본 고에서도 많은 부분을 이 규격서에서 참조하고 있다.

1. 기능 구조(Functional Architecture)

CSTA 표준의 목표는 스위칭 기능(Switching Function)과 컴퓨팅 기능(Computing Function) 간의 응용 서비스 인터페이스를 제공하는 것이며 이들 두 기능들은 서비스 경계(Service Boundary)를 통하여 분산된다.

컴퓨팅 기능은 컴퓨터 망의 여러 컴퓨터들에 의해 구현되고 스위칭 기능은 전화 망의 여러 교환기들에서 구현되며 이들의 수행은 서로 포함되어 이루어질 수도 있다. CSTA 응용 프로세스들은 대부분의 경우 분산되어 있기 때문에 이들은 <그림 1>과 같은 세 가지 요소로 나뉘어 각각의 기능을 수행하게 된다[2].

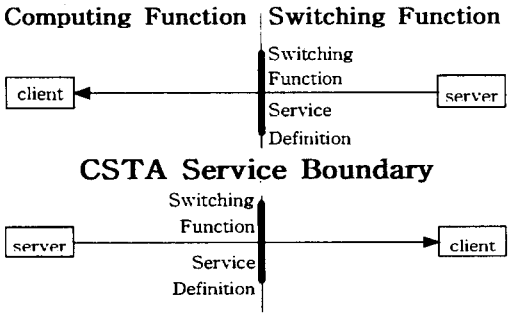


〈그림 1〉 CSTA 요소들간의 기능 관계

프로세싱 성분(Processing Component)들은 CSTA 표준에서 정의된 서비스 인터페이스로 상대 단과 상호 작용를 수행하고 통신 성분(Comm's Component)들은 메시지 형태, 파라미터 및 교환 기법을 포함하는 프로토콜을 사용하여 상대 단과

통신 기능을 수행한다. CSTA는 OSI 응용 계층의 인터페이스이므로 하위 계층의 전송 프로토콜을 필요로 한다. 이는 Networking Support에서 이루어지며 실제 구현에 적합한 프로토콜을 선택적으로 사용할 수 있다.

CSTA 응용 프로세스에서 요구되는 통신 기법은 client/server의 관계로 모델화된다. 위에서 언급된 프로세싱 성분이 서비스를 요구하면 통신 성분이 client의 역할을 수행하며 요구된 서비스를 상대 단에 전송하여 server를 호출한다. 이러한 모델이 <그림 2>에 나타나있다[2].

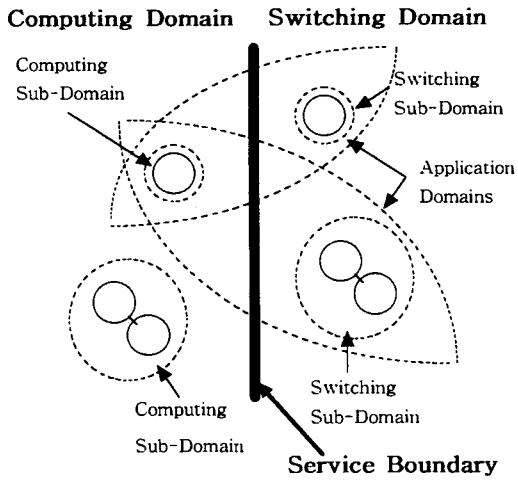


〈그림 2〉 양방향 client/server 서비스

컴퓨팅 기능이 client이고 스위칭 기능이 server인 서비스는 스위칭 기능 서비스로 정의되고 Make Call 서비스를 예로 들 수 있으며 그 역은 컴퓨팅 기능 서비스로 정의되고 Route Request 서비스 등이 있다.

2. 동작 모델(Operational Model)

CSTA 동작 모델의 기본 개념은 경계를 공유하는 모든 두 인접 요소간에는 항상 스위칭 영역(Switching Domain)과 컴퓨팅 영역(Computing Domain)으로 분류되고 모델화된다는 것이다. 각 영역은 기능들로 구성되고 각 기능을 통해서 CSTA 응용 프로세스는 영역에 대한 특성을 알게 된다. 같은 특성을 갖는 기능들은 동일한 부영역(Sub-Domain)을 형성하고 CSTA 응용 프로세스는 적어도 하나씩의 스위칭과 컴퓨팅 부영역을 포함하고 있다. 이들은 CSTA 프로토콜에서 가장 기본적인 구조를 형성하며 이들간의 관계를 <그림 3>에 나타내었다.



〈그림 3〉 동작 모델

스위칭 영역은 컴퓨팅 영역에 의해 엑세스되는 전화 자원을 말하며, 중심국(CO), 사설 교환기(PBX), 전화 단말, ISDN 단말과 카드 및 키 시스템 등이 포함된다. 컴퓨팅 영역은 스위칭 영역으로부터 CTI 서비스를 얻고 있는 컴퓨터 자원을 말하며 하드웨어와 소프트웨어를 모두 포함한다. 또한 이 영역은 적어도 하나의 소프트웨어 요소를 가진다. 서비스 경계는 두 영역간에 제어 및 상태 정보가 전달되는 논리적 경계를 말하고 소프트웨어에 대해서는 프로그래밍 인터페이스가 되고 그 외의 경우는 두 영역간에 수행되는 프로토콜이 해당될 수 있다. 이는 두 요소간의 상호 동작을 가능하게 해 준다.

단, 영역에 대한 해석은 상황에 따른 문맥에 따라 달라질 수 있다는 것이 중요하다. 예를 들면, 교환기와 컴퓨터를 연결하는 Telephony Server의 경우 교환기와의 관계에서는 컴퓨팅 영역에 해당되고 컴퓨터와의 관계에서는 스위칭 영역에 해당된다. 또한 하나의 스위칭-컴퓨팅 영역 쌍에서의 서비스 경계는 유일하며 각각의 관계에서는 서로 다른 서비스 경계가 존재하게 된다[4].

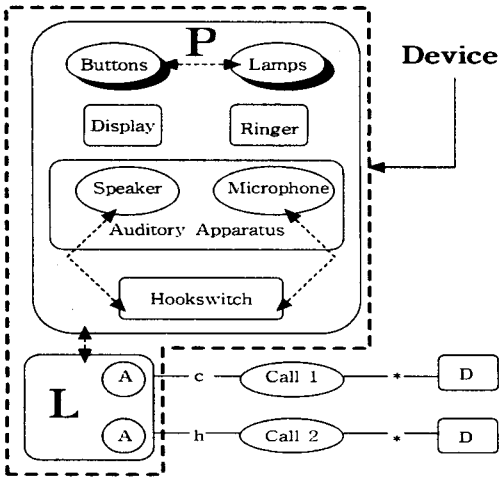
2.1 스위칭 영역 모델

이 모델은 응용 프로세스로 하여금 스위칭 기능의 동작을 개념화할 수 있게 하며 CSTA에서는 컴

퓨팅 기능에 의해 제어 및 관찰될 수 있는 여러 객체를 정의하고 있다. 이러한 객체들로는 Device, Call 및 Connection이 있다[2].

(1) Device

Device는 전화 기능을 수행하는 다양한 형태의 기능 종단(end-point)을 의미하고 전화 서비스에 액세스할 수 있는 스위칭 영역 내의 추상 개념이다. Device는 하나의 기능 종단에서 그룹을 형성하는 기능 종단 집합에 이르기까지 다양하게 해당되며 논리적 요소(Logical element)와 물리적 요소(Physical element)로 구성되고 속성(Attribute)/특징(Feature)/서비스로 표현된다. 논리적 요소는 Device에서 Call의 제어 및 관찰과 관련있는 일련의 속성/특징/서비스들을, 물리적 요소는 사용자 인터페이스를 구성하는 물리적 성분(Physical component)과 관련을 갖는 일련의 속성/특징/서비스들을 포함한다. <그림 4>에 일반적인 디바이스의 예를 나타내었다[4].



D Another Device
P Physical element
L Logical element
A Appearance of a logical element

〈그림 4〉 일반적인 Device

물리적 요소는 물리적 성분의 속성과 사용자 인터페이스를 구성하는 특징 및 서비스를 표현하는 추상적 개념이다. 예를 들면 전화 단말의 여러 구성 요소들, 즉 음성 수신 장치, 훅 스위치, 버튼, 램프, 링어(Ringer), 디스플레이 등이 물리적 요소에 해당한다. 그리고 이러한 물리적 요소들은 스위칭 영역에 의해 디바이스 식별자(Identifier)를 사용하여 관찰 및 제어된다.

논리적 요소는 Device에서 Call과 상호 작용하거나 Call을 관리하는데 사용되는 부분을 일컫는 추상적 개념이다. 예를 들면 매체 스트림 채널이나 신호 처리 기능 등이 이에 해당하고 물리적 요소와도 상호 작용하게 된다. 관련된 논리적 성분 및 속성들은 Appearance와 Group으로 크게 구분된다. Appearance는 최대 하나의 Call을 처리하는데 사용되는 성분으로 논리적 요소에는 여러 Appearance가 있을 수 있고 Call Appearance라 불리기도 하며 예로는 Connection Handler 등이 있다. <그림 4>에서 A로 표시된 부분이 Appearance이며 각각 하나씩의 Call과 연결되어 있는 것을 볼 수 있다. Appearance는 다음과 같은 Call 및 Call에 관련된 속성/서비스들을 관찰 및 제어하는 데 사용된다[4].

- Make Call, Answer Call 등과 같은 Call 제어 속성/서비스
- 관련 자료, DTMF Tone 등과 같은 Call 관련 속성/서비스
- Route Request, Route Select 및 Route End 등과 같은 Routing 서비스
- 매체 스트림 엑세스 서비스

Group은 하나의 논리적 요소만을 갖는 Device와 여러 다른 Device와의 약결합에 관련된 특정 형태를 정의하는 논리적 요소의 속성을 말하고 Call을 Group 내의 일련의 Device들간에 분산 처리할 수 있다. Group의 예로는 ACD Group, Pick Group, Hunt Group 등이 있다.

Device에 관련된 속성으로는 형태(type), 디바이스 식별자(Device Identifier), 및 상태(Status) 등이 있다. Device의 형태는 매우 다양한데 일반적으로 전화 단말, 망 인터페이스 장치(예를 들면, Trunk, CO Line등), ACD, ACD Group, Hunt Group, Park, Park Group 등이 있으며 그 외에도 여러 장치가 해당될 수 있다. 디바이스 식별자는 물리적/논리적 요소, Appearance 및 ACD와 관련

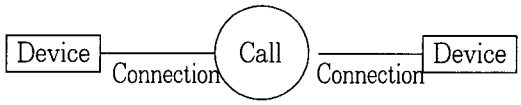
된 agentID 등을 참조하는데 사용될 수 있고 그 값은 Call에서 유일하며 주어진 스위칭 영역 내에서 고정적이다. 그리고 Device 자체는 상태를 갖지 않으나 Device와 관련된 성분, 요소 및 관련된 Call은 상태를 갖는다. 이러한 상태들에는 연결 상태(Connection Status), 물리적/논리적 성분의 상태 등이 있다[4].

(2) Call

Call은 여러 Device들의 통신 관계를 나타내는 추상적 개념이며 Call의 동작은 서비스 경계에서 관찰 및 제어될 수 있다. Call의 진행 단계에 따라 포함된 Device가 하나일 수도 있고(초기화 시) 진행 중 포함된 Device가 대체될 수도 있다. 또한 Call은 관련된 일련의 속성들로 표현되며 이러한 속성에는 호 식별자(Call Identifier), 호 상태(Call state) 및 Correlator data, 호 형태(Call type) 등이 있다. 호 식별자는 스위칭 영역에 의해 Call의 완전한 설정이 이루어지기 전에 할당되며 진행 중 두 영역에 의해 모두 식별된다. 스위칭 영역 내에서 모든 Call들간에 유일하며 같은 Call 내의 모든 Device에서도 같은 값을 가진다. 호 상태는 Call을 구성하는 Connection들에 대한 일련의 연결 상태들로 구성된다. Correlator data는 컴퓨팅 영역에 의해 관찰되고 제어되고 있는 Call에 추가되는 컴퓨팅 영역에 특정한 데이터이다. 예를 들면, 이와 같은 정보는 데이터베이스의 Key나 파일 이름 등이 될 수 있다. 이 정보가 일단 Call에 관련되면 전체 기간 동안 또는 컴퓨팅 영역이 새로운 정보로 바꿀 때까지 Call 내에서 일관되게 유지된다. 호 형태는 Call이 음성 또는 데이터와 같은 특성을 나타내는 정보이며 이러한 특성들로는 대역폭, 비트율, 지연 저항력 및 스위칭 영역의 호 제어 정보 등이 있다[4].

(3) Connection

Connection은 스위칭 영역 내에서 Device와 그 Device를 포함하는 Call 간의 관계에 대한 추상적 개념이며 이 관계는 컴퓨팅 영역에 의해 관찰되고 제어될 수 있다. <그림 5>에 이들간의 관계를 간단하게 나타내었다[4].



〈그림 5〉 Call, Device 및 Connection의 관계

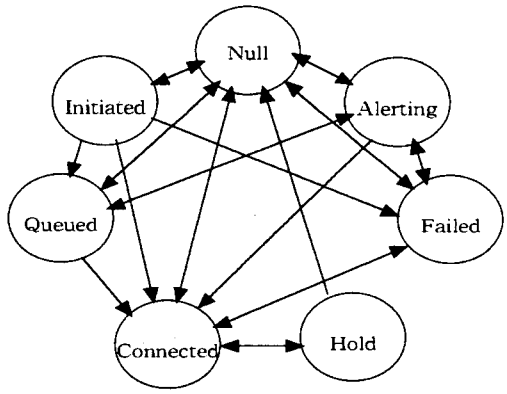
Connection에 관계된 속성들은 다음과 같다.

① 연결 식별자(Connection Identifier)

이 속성은 스위칭 영역에 의해 할당되며 Device ID와 Call ID의 쌍으로 이루어진다. 하나의 서비스 경계 상에서, 그리고 하나의 영역 내에서는 유일하며 새로운 Call이 발생하거나 존재하는 Call에 새로운 Device가 포함될 때 할당된다. 또한 동작(예 : 회의, 전송 등)에 따라 진행 중에 ID가 변할 수도 있는데 이때는 컴퓨팅 영역으로 알려져야 한다.

② 연결 상태(Connection state)

하나의 Call/Device의 연결 관계를 표현하며 다양한 단계에 따라 전환된다. 상태 변화는 컴퓨팅 영역에 의해 관찰 및 제어되고 이는 Event Reports, 또는 Call과 Device 상의 Snapshot에 의해 알려진다. 상태는 사용자의 명령 입력이나 서비스 경계를 통하여 발생하는 서비스에 의해 전환된다.



〈그림 6〉 Connection State Model

2) 제어 및 관리 서비스

(1) Capability Exchange

컴퓨팅 영역이 스위칭 영역을 관찰하고 제어하

기 위해서는 스위칭 영역의 정보를 전달받아야 할 필요가 있는데 이러한 기능을 Capability Exchange를 통하여 달성한다. 이 기능은 동작 모델과 특성의 측면에서 관련 정보를 컴퓨팅 영역에게 알리는 것으로 다음과 같은 정보를 교환한다[4].

- 일반적인 동작 모델 정보 : 스위칭 영역 이름, 특성
- 서비스와 Event 정보
- 스위칭 영역 내의 Device list
- 특정 Device 관련 정보

(2) Monitoring

스위칭 영역에서 발생하는 모든 변화 표시를 수신하고 Call 제어와 여러 동작들을 살피기 위하여 컴퓨팅 영역에서 사용되는 특징(Feature)이다. Monitor가 일단 설정되면 스위칭 영역은 event report 또는 event라는 메시지를 보냄으로서 관련 정보를 알리고 이 정보에 따라 컴퓨팅 영역은 주어진 Connection에 대해 어떠한 서비스가 가능한지를 결정하게 된다. Monitor의 설정과 해제는 Monitor Start Service와 Monitor Stop Service로 이루어진다. Monitor object는 Call-과 Device-object의 두 가지가 있고 감시 방식에도 Call이 특정 Device에서 해제되면 더 이상 감시하지 않는 Device-type과 Call이 스위칭 영역 내에 존재하는 한 감시를 계속하는 Call-type의 두 가지가 있다[4].

(3) Snapshot

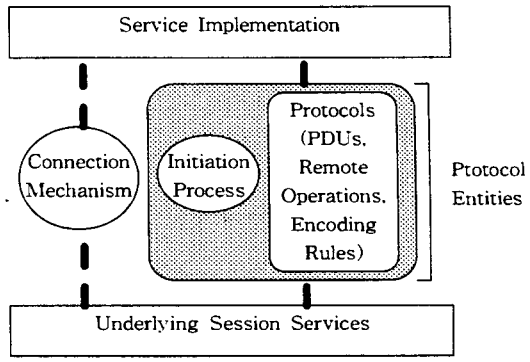
Call과 Device에 대한 정보를 결정하기 위해 컴퓨팅 영역에서 사용되며 어느 때에나 가능하고 Monitor와는 별도로 또는 조합하여 사용 가능하다[4].

(4) Routing Service

스위칭 영역에서 Routing Service를 요구할 때 행해지며 컴퓨팅 영역이 Call-by-Call 단위로 목적지를 알려준다. 이를 위하여 컴퓨팅 영역은 내부 데이터베이스를 Call 정보와 함께 사용하여 목적지를 결정한다[4].

3. 프로토콜 구조

CSTA에서는 정의된 서비스의 기능 구현과 관련하여 PDU(Protocol Data Units)를 정의하고 하위 전송 프로토콜에서는 이 PDU들을 신뢰성있게 전송한다. 이 때 전송 프로토콜을 선택적으로 사용할 수 있다. <그림 7>에서 실제 구현 환경과 관련된 프로토콜 구조를 보여주고 있다[4].



〈그림 7〉 프로토콜 구조

<그림 7>에서 연결의 초기화, 관리 및 종료에 관한 기법은 연결 기법에 따라 다양하게 적용할 수 있다.

CSTA의 서비스는 ITU-T 권고 X.219의 원격 동작(Remote Operation)으로 모델화되었다. 하나의 개체가 특정 동작의 수행을 요구하게 되면, 다른 개체가 그 동작을 시도하고 응답을 보내게 되는데 결과적으로 CSTA 프로토콜의 동작은 OSI 응용 계층에서 기본적인 요구/응답(request/reply)의 상호 작용으로 구성되며 응용 프로세스간의 연관 관계에 따라 기능이 수행된다. CSTA에서는 서비스에 따라 서비스의 요구를 받은 개체가 응답을 생성하는 경우 <표 1>과 같은 세 가지 방식이 존재한다.

CSTA에서는 성공이나 실패를 나타내는 하나의 응답을 원래의 요구와 상호 연관시키는데 ROSE

〈표 1〉 요구에 대한 응답 방식

비동기, 성공 또는 실패 보고	ITU-T 권고 X.219의 Class 2 동작
비동기, 실패만 보고	ITU-T 권고 X.219의 Class 3 동작
비동기, 결과를 보고하지 않음	ITU-T 권고 X.219의 Class 5 동작

(Remote Operation Service Element)에서의 기법을 사용한다. 〈표 1〉에서 나타난 서비스 응답을 세부적으로 살펴보면 다음과 같이 분류될 수 있다[4].

- 특정 서비스의 경우, 올바른 요구에 대한 응답을 보내지 않는 비확인 모드를 가질 수 있다.
- 서버가 응답을 보내기 전에 서비스 요구의 정확성을 검사한다. 잘못된 요구에 대해서는 오류를 알리는 응답을 보낸다.
- 응답이 서비스가 완료되기 전에 보내지면 이후에 발생하는 서버의 동작을 살피기 위해 event reporting을 사용한다.

CSTA 프로토콜은 주로 third-party 호 제어를 위해 사용되도록 고안되었고 이의 구현은 OSI 응용 서비스 개체로서 정의되었다. OSI 응용 계층을 다루기 위한 기법은 전형적으로 OSI의 ACSE (Application Control Service Element)를 사용하고 서비스 요구, 확인 응답 및 event를 전송하기 위한 프로토콜로는 위에서 언급한 OSI의 ROSE를 사용한다.

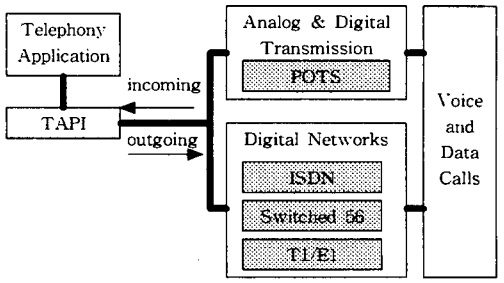
III. TAPI 기술 분석

Microsoft Windows TAPI(Telephony API)는 Microsoft에서 개발되었고 현재 Microsoft Windows NT Server version 4.0과 Windows NT Workstation version 4.0 운영 체제에 탑재되어 있다. TAPI 2.0은 하드웨어 계층을 추상화하여 망과 디바이스에 무관하게 개발자와 사용자에게 제공된다[6].

Windows Telephony SDK는 크게 TAPI와 TSPI(Telephony Service Provider Interface)의 두 부분으로 구성된다[5].

- TAPI : 윈도우즈 사용자들에게 개인용 전화 기능을 제공하는 응용 프로그램의 개발을 위한 인터페이스를 제공한다.
- TSPI : 응용 프로그램으로부터의 요구를 다룰 디바이스 중심적인 서비스의 개발을 위한 인터페이스를 제공한다.

윈도우즈 TAPI의 가장 큰 특징은 다양한 전화망을 투명하게 다룬다는 점이다. 응용 프로그램은 하위 망의 세부 사항에 대해서는 모르는 상태에서 'line'과 'phones'라는 추상 객체만을 사용하며 이에 따라 전화 응용 프로그래밍을 간소화한다. 하위 망의 세부 사항과 응용 프로그램과의 인터페이스는 모두 TAPI를 통해서 이루어지며 이러한 관계가 <그림 8>에 나타나 있다.



〈그림 8〉 call과 line에 대한 제어

TAPI는 1993년 11월 첫 버전인 TAPI 1.3으로 발표되었다. 이 버전에서는 호 제어 관련 응용 분야만 제공되었고 16비트 응용 프로그램만 가능했으며 다이얼 기능은 제공하지 못했다. Windows 95의 일부로 발표된 TAPI 1.4에서는 상당한 개선을 이루었는데, 32 비트 응용 프로그램이 가능해졌으며, Plug-and-Play 기능과 일반적인 다이얼 기능도 포함하게 되었으며 그 외의 여러 가지 기능이 추가되었다. TAPI 1.3과 1.4에서는 first-party 호 제어 기능만 제공되고 client/server 모델은 지원하지 못했으나 TAPI 2.0에서는 이 문제를 해결하였다. TAPI 2.0에서의 주요 개선 사항은 다음과 같다[6].

- 전체적으로 32비트 구조를 가지게 되었기 때문에 32비트와 16비트 응용 프로그램을 모두 지원할 수 있게 되었다.
- 윈도우즈 상에서 콜 센터의 동작을 더욱 완전하게 지원할 수 있도록 확장된 특성을 지니게 되었다.
- 응용 프로그램이 QoS 성능 파라미터를 요구

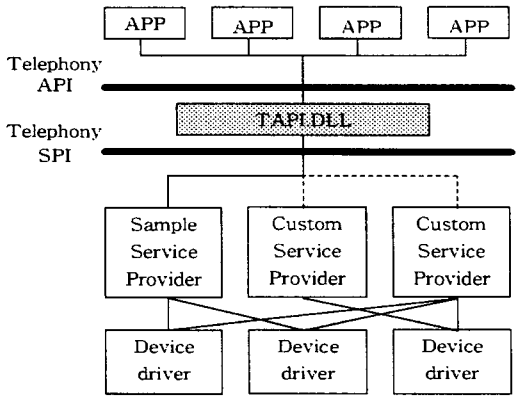
하고 협상할 수 있게 되었다.

– 디바이스 공유 기능이 확장되었다.

TAPI는 WOSA(Windows Open Service Architecture)의 일부이며 이기종간의 컴퓨팅 자원에 대한 추상화 계층의 API를 제공한다. 따라서 서비스간 통신을 수행할 때 WOSA 인터페이스에 따른 API만을 통하면 되고 이것을 가능하게 하는 것이 TAPI.DLL이다. 다음 절에서 TAPI의 구성과 프로그래밍 모델을 살펴보겠다.

1. 구성

Windows Telephony는 WOSA 모델을 따르도록 설계되었기 때문에 응용 프로그램과 서비스 제공자간의 인터페이스를 위해 TAPI.DLL을 제공한다. 이는 응용 프로그램에서 호출되는 Telephony API와 Service Provider(이하 SP)에 의해 구현되는 TSPI 간에 위치하고 있다. 즉, 응용 프로그램은 TAPI.DLL에 의해 관리되는 TAPI 함수를 호출하고 TAPI.DLL은 실행을 위해 적절한 SP에 경로 배정을 수행한다. 이러한 관계가 <그림 9>에 나타나있다.



〈그림 9〉 WOSA 모델에서의 TAPI 구성

위 그림에서 'Sample'로 되어있는 SP는 Microsoft가 TAPI 내에 탑재해 놓은 것으로 기본적인 전화 기능만을 수행할 수 있는 함수들로 구성되어 있다. 이와는 달리 'Custom'으로 되어있는 SP는 기본적인 전화 서비스에 더하여 추가적인 기능을 제공하기 위해 third-party에 의해 개발된다. 위 그림에서 보듯이 여러 응용 프로그램이 동시에 동작할 수 있고 하나의 응용 프로그램이 여러 SP를 사용할 수도 있으며 여러 하드웨어 디바이스도 액세스될 수 있다. 이러한 관계를 가능하게 하기 위하여 TAPI.DLL은 'broker'로 동작하고 어떠한 SP에 대한 함수 호출도 이를 통하여 전달되게 되는데 이러한 경로 배정은 함수 호출시의 파라미터를 분석하여 어떠한 SP와 관련되었는가를 파악하여 수행된다.

이 외에도 Media-stream을 제어하기 위하여 다른 API들을 필요로 한다. Windows Telephony는 그 자체로는 line device와 phone device에 대한 제어만을 제공하고 호의 진행 중 교환되는 정보에 대한 액세스는 제공하지 않기 때문에 실제 media stream을 다루기 위해 개발자가 응용 프로그램과 SP 내에 다른 API로부터의 함수를 결합하여 사용해야 한다.

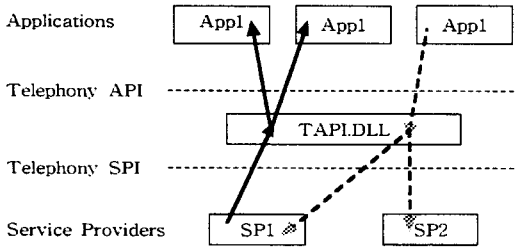
TAPI에서 제공되는 기능은 실제 전화 망과 컴퓨터와 전화망간의 연결 체계와는 무관하기 때문에 TAPI를 이용하여 다양한 물리적 환경을 구성할 수 있다. 가능한 환경은 다음과 같다.

– Computer-centric
– Phone-centric
– BRI-ISDN connections
– LAN based connections
– Wireless connection
– Shared Telephony connections

2. 프로그래밍 모델

1) TAPI.DLL과 SP의 역할

앞에서 언급한 것과 같이 TAPI와 TSPI의 함수들은 서로 추상화된 개체로 인식한다. 이들의 계층관계를 예를 들어 아래 <그림 10>에 나타내었다. TAPI는 그림에서 보듯이 계층간에 각각의 API를 둠으로써 응용 프로그램과 SP간에 가상의 환경을 제공하여 응용 프로그램에게는 SP를, SP에게는 응용 프로그램을 각각 추상화시킨다. 그리고 이들간의 중재를 TAPI.DLL이 다루게 된다.



〈그림 10〉 추상화 계층

위 그림에서 살펴보면, SP1에 의해 보여지는 서비스는 실선으로 나타난다. 실제 두 응용 프로그램으로부터 서비스 요구를 받고 있지만 SP1은 TAPI .DLL에 의해 하나의 큰 응용 프로그램으로부터 요구를 받는 것으로 인식한다. 이러한 관계는 점선으로 나타나는 App1에서도 마찬가지로 하나의 SP에게 보낸 서비스 요구가 TAPI. DLL을 통하며 여러 SP에게로 주어진다. 또한 TAPI.DLL이 SP1과 SP2로부터의 다른 서비스를 조합하여 App1에게로 전송하는데 이러한 방식으로 SP와 응용 프로그램들은 시스템에서 자신만이 유일한 것으로 생각할 수 있도록 추상화가 제공된다.

TAPI.DLL의 역할은 응용 프로그램과 SP 간의 트래픽 관리를 담당하는 것이다. 하나의 SP를 사용하는 여러 응용 프로그램과 한 응용 프로그램에 서비스를 제공하는 여러 SP들을 관리하는데 요구와 서비스의 결합 및 분산을 관리하고 이를 위한 자료 구조에 대한 중재나 동기화의 세부 사항은 숨긴다. 이 외에도 파라미터 적합성 검사와 권한 레벨의 구현 등을 포함한 서비스를 제공한다. 이렇게 TAPI.DLL이 특정 전화 장치의 하위 세부 사항과는 무관한 것과는 달리 SP는 하위의 장치 의존적인 특성을 보인다. SP의 역할은 호 상태, 다이얼링, 신호 발생과 검사 및 switchboard 제어 등의 기능을 수행하며 이러한 기능들은 장치별로 다르게 구현된다.

2) Device Class와 Service

TAPI 프로그래밍 모델에서는 전화 장치, 모뎀, 전화 선 등과 같은 실제 개체들을 API 레벨에서 유사한 클래스로 추상화시켜서 개발을 용이하게 한다. TSPI에서는 이러한 장치들을 그들의 특성에 따라 크게 Line Device와 Phone Device 클래스로 구분하고 SPI는 그들에 대한 함수와 메시지들을 정의한다. Line Device는 SPI에서 정의된 'line behavior'를 구현하는 장치로 전화 선과 그들을 제어하는 fax board, 모뎀, ISDN card 등과 같은 하드웨어로 구성된다. Phone Device는 SPI에서 정의된 'phone behavior'를 구현하는 장치로 훅스위치, Volume Control/Mute, Ringer, Display, Button 및 Lamp 등으로 구성된다. 이 두 클래스의 구현과 실제 전화에 매핑되는 방법은 전적으로 SP에 달려 있다. Line Device는 여러 채널들로 구성되고 한 Line에 속한 채널들은 모두 동일하다. 이 외에도 Call 개체가 있는데 이는 여러 remote parties에 대한 호 연결의 종단을 의미한다. 정적으로 정의된 Line과 Phone Device와는 달리 Call은 동적으로 만들어지고 외부 이름도 가지지 않는데 outbound call 시에는 요청이 있을 때, inbound call 시에는 자동적으로 발생한다. 이 객체는 호 상에 동작하는 함수들의 파라미터로 사용된다.

TAPI에서 제공하는 서비스는 모두 세 가지 레벨로 구분할 수 있는데 이들은 각각 기본(Basic) 서비스와 추가(Supplementary) 서비스, 그리고 확장(Extended) 서비스가 있다. 기본 서비스는 POTS에 대응되는 기능들의 최소 집합으로서 TAPI SP는 모든 기본 서비스를 반드시 제공해야 한다. 추가 서비스는 hold, transfer 등과 같은 발전된 교환기 특성을 나타내는 기능들이 정의되고 이 서비스의 구현은 SP에 따라 선택적이며, 동작 정의는 TAPI에서, 구현은 SP에서 이루어진다. 확장 서비스는 장치 의존적인 서비스로서 TAPI에 의해 정의되지 않은 SP-specific 기능을 응용 프로그램 개발자가 사용할 수 있도록 하는 API 확장 기법을 제공하는 서비스이다. 이러한 서비스들은 함수들로 구현되는데 TAPI 함수 집합은 TSPI 함수와 일대일로 대응되는 것은 아니라는 점을 고려해야 한다. 예를 들면, 권한, 전화 번호 변환, 응용 프로그램간 통신 등의 기능에 관련된 함수들은 TSPI에는 대응하는 것이 없고, SP 구성, 초기화 함수 등과 같은 함수들은 TAPI에서 대응하는 것이 없다.

위에서 설명한 개념들을 바탕으로 TAPI는 client-only, client/server, server-based 등의 여러 구성 환경을 가질 수 있고 다음과 같은 많은 응용 분야에 적용할 수 있다[6].

- 가상 호 제어
- 음성의 데이터 전환
- 전화 기능을 수행하는 응용 제품 개발
- Auto attendent
- 탁상 회의
- 콜 센터 적용

IV. CSTA와 TAPI의 비교

CTI 시스템에서 컴퓨터와 전화 시스템을 연결하는 물리적 링크는 데스크 탑에 위치하는 First-pary 구조와 교환기와 서버 사이에 위치하는 Third-party 구조로 구분된다.

First-party 구조에서는 CTI 게이트웨이가 컴퓨터와 전화선 사이에 위치하고 선상의 신호의 번역과 시뮬레이션이 데스트 탑 컴퓨터에서 수행되기 때문에 컴퓨터에서 실행하는 명령이 전화에서 가능한 기능만으로 제한된다. 그러므로 컴퓨터는 하나의 전화선이나 그에 대응하는 것을 제어해야 한다. 전화가 PBX에 연결되어 있다면 전화와 PBX 간의 신호 방식은 PBX 제조업체 전용의 것일 것이며 기본적인 특징 이상의 서비스를 원할 때에는 PBX 제조업체에서 개발된 추가적인 장비가 필요할 것이다. 위에서 언급한 First-party 구조를 목적으로 개발된 API가 Windows TAPI이며 대부분의 First-party 링크는 이 TAPI를 기본으로 한다.

이와는 달리 Third-party 구조는 호스트 컴퓨터와 교환기 간의 프로세서 대 프로세서 연결을 제공한다. 이 구조에서 CTI 게이트웨이를 위한 소프트웨어는 일반적으로 하나의 서버에서, 또는 여러 분산된 서버에서 실행될 것이며 CTI 게이트웨이 전용의 목적으로 사용된다. 그리고 이러한 구조를 용이하게 하기 위하여 주로 client/server 프로토콜을 지원하는 API가 필요하게 된다. 이러한 목적으로 이 구조에서는 주로 CSTA 표준이 사용되며 이를 사용하는 여러 API 중의 하나가 선택되게 된다.

그러나 물리적 구조가 반드시 응용 프로그램에서 보여지는 관점에 대응될 필요는 없다. 어떤 추가적인 미들웨어를 사용한다면 Third- party 구조도 First-party 구조를 위하여 구현된 응용 소프트웨어를 가지고 사용될 수 있다. 또한 First-party 구조를 시작으로 개발된 TAPI도 현재는 Third-party 구조를 지원할 수 있으며 이에 따라 실제 CTI 시스템 구현시 구현 환경과 여러 가지 제약 조건에 따라 적절한 표준 및 API를 선택해서 사용해야 할 것이다.

V. 결 론

컴퓨터와 전화 시스템은 대부분의 개발 업체들의 기술적 기본이 되는 두 가지의 큰 항목이다. 이 두 기술을 결합하여 상호 장점을 살리려는 노력이 활성화되어 왔고 현재에 이르러서는 각 개인의 데스크 탑에서까지 CTI의 장점을 얻을 수 있게 되었다. 이는 이 두 기술을 상호 연결할 수 있는 여러 표준들이 개발되어 CTI 시스템의 개발이 쉽고 싼 가격에서 이루어질 수 있었다는 점과 CTI 소프트웨어 산업이 활발하게 진행되어 여러 응용 소프트웨어와 개발 도구가 개발되었다는 점을 바탕으로 하여 가능할 수 있었다[1]. 현재 여러 표준 및 API들이 개발되어 발표되고 있는데 본 고에서는 특히 client/server 모델을 지원하는 CSTA 표준과 데스크 탑 중심의 구현 환경을 제공하는 TAPI에 대해 간략한 기술 사항을 살펴보았다. 이 둘의 가장 큰 차이는 기본 구조에 있음을 전기했으며 실제 환경에 적용함에 있어서 그 목적에 따라 적절한 API를 사용하여야 할 것이다.

오늘날 대부분의 CTI 사용은 콜 센터에서 이루어지고 있으나 가까운 미래에는 CTI가 기업 뿐 아니라 데스크 탑에서도 주로 사용될 것이다. 특히

2002년 경에는 콜 센터 외에서의 CTI 사용이 현재의 약 4배 이상이 될 것으로 예상된다[1]. 따라서 CTI 시장은 앞으로도 계속 커질 수 밖에 없을 것으로 보이며 향상된 CTI 제품 개발을 위하여 여러 표준 및 API에 대한 연구가 필수적이며 지속적이어야 할 것이다.

참 고 문 헌

[1] David Bradshaw, Madan Sheina, Simon Glassman, Mary Ann O'Loughlin, "Computer Telephony Integration," OVUM 1997.

[2] "Services for Computer Supported Telecommunications Applications(CSTA) Phase I," Standard ECMA-179, June 1992.

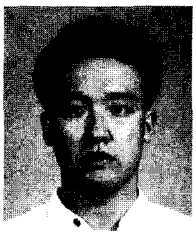
[3] "Services for Computer Supported Telecommunications Applications(CSTA) Phase II," Standard ECMA-217, December 1994.

[4] "Computer Telephony Integration(CTI) Encyclopedia," VersitTM, 1996.

[5] "Telephony Service Provider Programmer's Guide," Microsoft WindowsTM Version 3.1, Microsoft, 1993.

[6] "The Microsoft Windows Telephony Platform : Using TAPI 2.0 and Windows to Create the Next Generation of Computer-Telephony Integration," White Paper, Microsoft, 1996.

저 자 소 개



金 顯 奎

1969年 3月 11日生
1995年 2月 부산대학교 전자계산학과 졸업(학사)
1997年 2月 부산대학교 대학원 전자계산학과 졸업(석사)

1997年 1月~현재 LG정보통신 중앙연구소 연구원

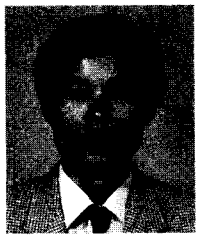
주관심 분야 : CTI, 신호망 프로토콜, 분산 처리

金 柄 秀

1965年 10月 24日生
1987年 2月 한양대학교 전자공학과 졸업(학사)
1989年 2月 한양대학교 대학원 전자공학과 졸업(석사)

1989年 1月～현재 LG정보통신 중앙연구소 선임연구원

주관심 분야 : CTI, ACD



姜 煥 鍾

1954年 4月 7日生
1981年 2月 인하대학교 전자공학과 졸업(학사)
1995年 8月 고려대학교 산업대학원 전자통신과 졸업(석사)

1981年 1月～1987年 6月 금성통신(주) 연구소
1987年 7月～현재 LG정보통신 중앙연구소 책임연구원

주관심 분야 : CTI, 멀티미디어 통신 시스템, 소프트웨어 공학