Virata 평가보드 장착용 VOVIDA SIP UA 코드 분석 및 제안*

최숙영°, 박석영, 문승진
수원대학교 정보공학대학 컴퓨터학과
{wind°,psywin,sjmoon}@mail.suwon.ac.kr

Analysis and New Proposal of User Agent codes of VOVIDA SIP for Virata Evaluation Board

Sook-young Choi° Suk-young Park Seung-Jin Moon
Dept. of Computer Science, The University of Suwon

요 약

VoIP기술은 기존의 음성서비스를 인터넷을 통하여 제공하고자 하는 기술로 기본적인 인터넷 전화 서비스에서부터 화상회의, 원격교육, 인터넷 콜 센터 등 최근 다양한 응용분야로 그 기술이 확대되어 가고 있다. 본 논문에서는 이러한 VoIP 기술의 적용을 위해 현재 MGCP를 사용하고 있는 Virata 평가보드를 SIP UA(User Agent)로 사용하여 간단한 인터넷 전화를 구현하는데 필요한 SIP 스택분석 및 효과적인 활용방안에 관해 논하였다. 이를 위해 VOVIDA사의 VOCAL(Vovida Open Communication Application Library) 오픈 프로젝트 중 SIP UA 부분을 분석하였으며, 이를 기반으로 분석된 소스를 평가보드에 효과적으로 탑재하고 활용하는 방안을 제안하였다.

1. 서 론

VoIP(Voice over Internet Protocol)기술은 기존의 음성서비스를 인터넷을 통하여 제공하고자 하는 기술이며 이 기술을 이용하면 기존 인터넷망을 그대로 이용하여 음성전화 서비스를 통합, 구현함으로써 전화나 팩스 송신에 소요되던 회선 비용을 크게 절감 할 수 있다[1,2]. VoIP관련 기술은 IETF와 ITU-T에서 표준화가 진행중이며 주요 표준으로는 ITU-T의 H.323과 IETF의 SIP, MGCP 그리고 두 단체가 함께 참여한 H.248/Megaco가 있다[3]. 본 논문에서는 MGCP가 적재되어 있는 평가보드에 SIP를 적용하여 간단한 인터넷 전화를 구현하기 위해 VOVIDA SIP의 UA부분을 분석하였다.

본 논문의 구성은 다음과 같다. 2장의 관련 연구에서 Virata 평가보드, MGCP, SIP 그리고 VOVIDA SIP 스택에 대해 알아본 후 3장에서 SIP UA를 분석하고 평가보드에 적용시킬 방안을 제안하고 4장의 결론 및 향후 연구과제에서 앞으로의 계획과 해결해야 할 문제들을 기술하였다.

2. 관련 연구

2.1 Virata 평가보드

Virata 평가보드는 광범위한 네트워킹 및 통신 장비를 개발하기 위해 GlobespanVirata사에서 만든 제품을 재구성한 보드이다. Helum210 칩을 사용하고 있으며 네트웍에 접속하기 위한 RJ45잭과 전화를 연결하기 위한 4개의 RJ11잭이 있다. 그리고 제어를 위해 호스트 컴퓨터와 시리얼 포트로 연결할 수 있다.

2.2 MGCP

MGCP(Media Gateway Control Protocol)는 미디어 게이트웨이 컨트롤러 또는 콜 에이전트라 불리는 외부 콜 제어 요소로부터 음성신호와 데이터 패킷 사이의 변환을 수행하는 텔레포니 게이트웨이를 제어하기 위한 프로토콜이다.

MGCP는 1999년 10월 IETF에서 RFC2705로 표준화되었다. <그림1>은 MGCP 콜 흐름을 나타낸다. RTP는

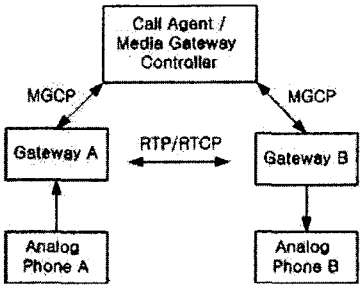

그림 1 MGCP Call Flow

오디오와 비디오 같은 실시간 데이터를 전송하기 위한 인터넷 프로토콜이고 RTCP는 RTP의 QoS를 유지하기 위해 함께 쓰이는 프로토콜이다. 전화A가 수화기를 들면 게이트웨이A는 콜 에이전트로 신호를 보낸다. 게이트웨이 A는 통화음을 발생시키고 눌러진 전화번호를 콜 에이전트로 보낸다. 콜 에이전트는 게이트웨이 B로 명령을 보내어 게이트웨이 B는 전화B에게 벨을 울리게 하고 전화B가 수화기를 들면 콜 에이전트는 RTP/RTCP세션을 생성하기 위해 양쪽의 게이트웨이로 명령을 보낸다.[4,9]

*본 연구는 (주)맥시스템과 공동수행한 제10차 산학연 컨소시엄 연구결과를 일부 포함하였음

2.3 SIP

SIP(Session Initiation Protocol)는 멀티미디어 세션 또는 콜을 생성, 수정, 중단하기 위한 어플리케이션 계층의 제어 프로토콜이다. 이 같은 세션에는 화상회의, 원격 교육, 인터넷 전화 그리고 이와 비슷한 어플리케이션들이 포함된다. SIP는 1999년 3월에 IETF에서 RFC2543으로 표준화되었다. <표1>은 SIP의 구성요소이다[6]. SIP는

표 1 SIP의 구성요소

UA	UAC(User Agent Client)	call을 시작, call중단
	UAS(User Agent Server)	call을 받음, call중단
Proxy Server		다른 클라이언트를 대신하여 요청
Location Server		callee의 가능한 위치에 대한 정보
Redirect Server		callee의 주소를 반환해 줌
Registrar		REGISTER 요청을 받아들임

client/server 구조이며 한쪽이 요청을 하면 다른 한쪽이 이에 대한 응답을 하는 요청/응답 구조이기도 하다[7]. <표2>는 SIP 메시지의 구조이다. SIP 메시지는 클라이언

표 2 SIP 메시지

```
generic-message = start-line
                  *message-header
                  CRLF
                  [ message-body ]
start-line = Request-Line | Status-Line
message-header = ( general-header
                 | request-header
                 | response-header
                 | entity-header )
```

트에서 서버로의 요청과 서버에서 클라이언트로의 응답으로 나뉘며 메시지-헤더와 메시지-바디로 이루어져 있고 CRLF로 구분된다. <표3>은 요청에 해당되는 메소드와 응답에 해당되는 3자리 숫자로 이루어진 여섯 종류의 상태 코드에 대한 설명이다[8]. SIP는 텍스트 기반 프로토콜이

표 3 Methods 와 Status Codes

Methods		Status Code	
INVITE	콜 요청	1xx	Informational
ACK	INVITE에 대한 최종 응답	2xx	Success
OPTIONS	Capability 정보요청	3xx	Redirection
BYE	콜 해지	4xx	Client Error
CANCEL	미결정의 콜 해지	5xx	Server Error
REGISTER	위치정보 등록, 삭제, 수정	6xx	Global Failure

기 때문에 구현과 디버깅이 쉽고 유연성과 확장성이 뛰어나다. 로케이션 서비스로 사용자의 이동성을 지원하며 하위 계층의 프로토콜에 중립적이다.[5]

2.4 VOVIDA SIP 스택

VOVIDA SIP 스택은 C++로 개발되었고, IETF의 SIP 정의를 충실히 따르고 있다. TCP와 UDP를 모두 지원하며, 사운드 카드나 Quicknet 카드를 가진 UA 사용 시 기본적인 보이스 콜을 지원한다. 이 스택은 많은 클래스들로 이루어져 있는데 중요한 클래스와 역할은 다음과 같다. SipTransceiver는 SIP 메시지들을 주고받는 메인 클래스이다. SipTransceiverFilter는 SipTransceiver로부터 상속된 클래스로서 받은 메시지를 거르는데 사용한다. SipMsg는 모든 SIP메시지를 위한 기본적인 클래스로서 SipMsg로부터 두 개의 하위 클래스가 파생된다. 하나는 모든 SIP 응답 코드를 위한 클래스인 StatusMsg이고 다른 하나는 모든 SIP method를 위한 SipCommand 클래스이다.

VOVIDA SIP는 여러 종류의 메시지를 테스트 해볼 수 있다. UA 실행 파일의 메인은 ua.cxx파일에 있으며 확장자가 .cfg인 파일이 설정파일이며 UA실행에 필요한 정보들을 설정한다. Quicknet 카드나 사운드 카드를 사용할 수 있고, 장치 없이도 콜 시그널링을 시험해 볼 수 있다.[9]

3. User Agent 분석 및 제안

3.1 소스 분석

VOVIDA SIP 소스는 1.4.0버전을 사용하였고 리눅스 기반에서 UA로 컴파일 하여 UA 대 UA로 테스트를 해보았다. <그림 2>는 UA-UA의 콜 흐름을 나타낸다.

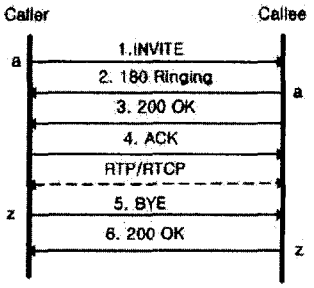

그림 2 UA-UA Call Flow

Ready가 출력된 후 caller쪽에서 a를 입력하면 수화기를 든 역할을 한다. 그러면 callee쪽에서는 caller쪽에서 요청을 했다는 메시지가 뜨고 a를 입력하면 caller와 callee간에는 세션이 성립되고 둘 중 하나가 z를 눌러 수화기를 내려놓으면 연결은 끊어진다[9].

다음은 UA의 main()이 들어있는 ua.cxx파일의 중요 함수들에 대한 설명이다. 먼저 UaCommandLine::instance()에서 명령어 라인의 옵션을 처리해 준 후 설정파일의 이름을 받아서 설정파일을 분석한다. 설정파일은 줄마다 <tag><type><value>로 구성되어 있으며 각각 빈칸이나 탭으로 구분된다. parse3tuple()함수를 이용하여 라인별로 분석한다. UaBuilder의 생성자 부분에서는 설정파일의 LoadGen_On부분과 caller인지 callee인지 설정해주는 RunMode 부분의 값을 검사한 후 push_back()함수를 이용하여 operator, state, feature들을 각각의 리스트에 넣는다. <표4>에 각각의 정의가 있다. 이들은 각각 State,

표 4 operator, state, feature의 정의

typedef list < Sptr < Operator > > OperatorContainer	OperatorContainer myOperators
typedef list < Sptr < State > > StateContainer	StateContainer myStates
typedef list < Sptr < Feature > > FeatureContainer	FeatureContainer myFeatures

Feature, Builder 클래스에 정의 되어있다. <그림3>은 operator, state, feature들이 어떤 경로로 리스트에 삽입되는지 보여준다. UaBuilder 생성자에서 UaStateMachine을 할당 할 때 실행되는 UaStateMachine 생성자에서는 addState함수를 이용하여 State로 시작하는 state들을 리스트에 넣고 addState의 매개변수가 되는 State로 시작하는 각 클래스의 생성자에서는 addOperator함수를 이용하여 Op로 시작하는 operator들을 리스트에 넣고 마지막으

로 addFeature함수를 이용하여 feature를 리스트에 넣는

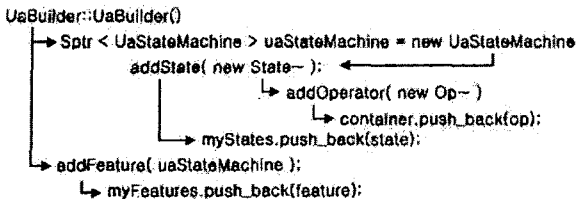

그림 3 UaBuilder 생성자

다. UserAgent 생성자에서는 Quicknet 카드, 사운드 카드, Voicemail 중 사용할 장치를 정하고 해당되는 게 없으면 장치를 Null로 설정한 후 LoadGen_On, Subscribe_On, MonitorMsgOn의 값과 Null장치의 가부를 검사하여 RtpThread, DeviceThread, SubscribeManager, FeatureThrea, LoadGenThread의 할당 여부를 결정한다. 그 다음 Register_On이 활성상태라면 등록을 하고 마지막으로 화면에 Ready를 출력하여 사용자의 입력을 기다린다. ua.run()은 스레드를 생성하기 위해 spawn()함수를 사용한다. <그림4>는 ua.run()의 구조를 나타낸다. 먼저 Worker와 Sip 스레드를 생성한 뒤

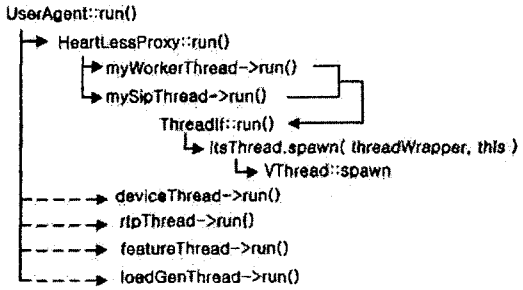

그림 4 ua.run()

deviceThread, rtpThread, featureThread, loadGenThread는 UserAgent 생성자의 결과에 따라 스레드를 생성한다. <그림4>의 모든 클래스들은 ThreadIf 클래스로부터 상속된 클래스들이다. ua.join()은 pthread_join()함수를 사용하여 스레드가 끝나기를 기다렸다가 인자로 받은 value_ptr값이 Null이 아니라면 스레드의 종료 값을 돌려준다. 0을 반환한 경우는 성공적으로 끝난 경우이다. 또한 이 함수는 성공적으로 완료된 스레드를 제거하고 제거된 스레드는 다시 조인되거나 취소될 수 없다. <그림5>는 ua.join()의 구조를 나타낸다. <그림6>은

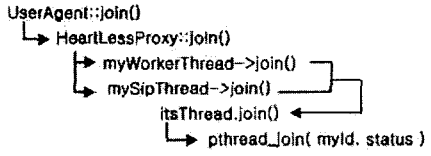

그림 5 ua.join()

SIP 콜 흐름도이다. MGCP에서의 Call Agent 부분이 Proxy Server의 역할을 하고 Gateway부분이 UA의 역할을 하게된다.

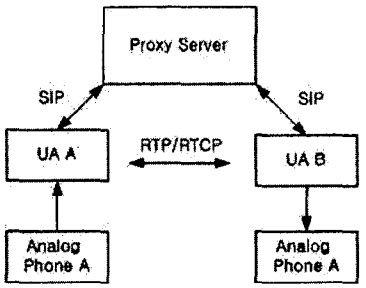

그림 6 SIP Call Flow

3.2 SIP 적재를 위한 제안

VOVIDA SIP UA를 보드에 적용시키기 위해서는 보드 환경에 맞추어 소스를 수정해야 한다. 설정파일을 이용한 설정대신 직접 명령어 라인 상에서 설정을 해야하고 장치 사용 에서는 Quicknet카드 부분을 보드의 하드웨어 환경에 맞게 수정해야 한다. 그리고 보드 상에 적재되는 이미지 컴파일 시 설정되는 VoIP 관련 값과 SIP 소스내의 해당 변수를 맞춰줘야 한다. 또한 저장 용량이 작은 보드의 저장 공간에 비해 VOVIDA SIP 소스는 용량이 크기 때문에 필요 없는 부분이나 여러 경우에 대비하여 검사를 하는 부분을 없애고 여러 단계로 중첩된 부분을 최적화 시켜 소스의 크기를 대폭 줄여야 한다.

4. 결론 및 향후 연구과제

본 논문에서는 MGCP가 적재되어 있는 평가보드에 SIP를 올리기 위한 선행작업으로 MGCP와 SIP에 대해 알아본 후 VOVIDA 사의 SIP UA를 분석하였고 SIP UA를 보드에 적용시킬 방안에 대해 제안해 보았다.

향후 과제로는 평가보드에 SIP를 탑재하기 위하여 SIP 소스를 보드에 맞게 수정및 축소할 예정이다. 또한 C로 작성된 MGCP와 C++로 작성된 VOVIDA SIP사이에서 발생할 수 있는 컴파일링 문제점등을 해소할 예정이며, 아울러 현재 1대1의 구조에서 발전된 다자간 통신이 가능한 멀티미디어 회의, 음성 통합 메신저 등등 다양한 응용 서비스를 개발하고자 한다.

5. 참고문헌

[1] 김영한, 고석갑, "VoIP 기술 개요 및 표준화 동향", 정보처리학회지 특집/VoIP기술, 제8권, 제2호, p.10, March. 2001
[2] 이일진, 강신각, "국내 VoIP 표준화 활동", 정보처리학회지 특집/VoIP기술, 제8권, 제2호, p.112, March. 2001
[3] 이상훈, "VoIP 시장 전개방향과 기술전망", 정보처리학회지 특집/VoIP기술, 제8권, 제2호, p.6-7, March. 2001
[4] ITU-T, "Media Gateway Control Protocol (MGCP) Version 1.0", October 1999
[5] IETF, "SIP: Session Initiation Protocol", March 1999
[6] 홍용기, 문성남, "SIP기반의 VoIP시스템", 정보처리학회지 특집/VoIP기술, 제8권, 제2호, p.92-93, March. 2001
[7] 윤형운, 김재은, 강현국, "SIP기반 UA 구현 구조 분석", 한국정보과학회 봄 학술발표논문집, p.299, 2001
[8] 이종화, 안상현, "SIP 기반 차세대 응용 기술", 정보처리학회지 특집/VoIP 기술, 제8권, 제2호, p.28, March. 2001
[9] http://www.vovida.org/