VoIP 표준 프로토콜을 위한 MGCP 및 SDP 스택 개발에 관한 연구

고 광 만†

요 약

최근 VoIP(Voice over IP) 관련 기술이 국내외 IP 네트워크 시장에서 급부상하고 있다. 현재까지 국내에서 MGCP, MEGACO, SIP, SDP 등 게이트웨이 제어 관련 프로토콜 스택의 개발을 위한 연구는 아직까지 보고된 바가 없으며 아직은 열악한 환경과 많은 시간 및 기술을 요하는 관계로 인해 이를 진행하기 위한 시도만 일부 이루어지고 있다.

따라서 본 연구에서는 IETF에서 제공되는 MGCP, SDP 프로토콜 문법을 기반으로 각 프로토콜 메시지를 위해 파서, 구문 분석기, 엔코더/디코더, 엔코더/디코더를 위한 헤더 파일 생성기로 구성된 프로토콜 스택을 개발하였다. 이를 위해 각 프로토콜 문법에 대해 문법-지시적(Syntax-directed) 변환 방법을 적용하여 엔코딩/디코딩을 위한 API, 구문 분석기, 헤더 파일 생성기를 구현한다.

A Study on the Development of MGCP and SDP Stack for VoIP Standard Protocols

Kwang-Man Ko†

ABSTRACT

Recently Technology regarding VoIP (Voice over IP) is emerging over the market of the IP network. So far nothing is unfortunately there any attempt to try any research with respect to the development of the protocol stack relating to such control of gateway as MGCP, MEGACO, SIP, SDP. The reasons come from the low level of infrastructue, the shortage of the time and technology required at the moment, and so on.

In this regards, this paper is focused on developing a protocol stack made with encoder/decoder, the generator of the header file etc. based on the protocol grammars of MGCP, SDP supported by IETF. For the sake of it, first develops the syntax analyzer, encoder/decoder, header file generator for encoding/decoding as applying the method of syntax-directed to each protocol grammar.

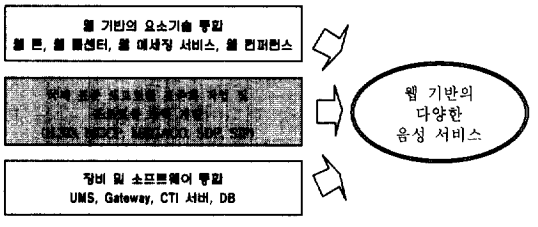
1. 서 론

최근 서비스 질(Quality of Service ; QoS)과 비용 면에서 장점을 갖는 데이터 네트워크를 통해 음성 신호의 전달하고자 하는 VoIP(Voice over IP)에 관련된 기술 개발 및 연구가 국내외적으로 활성화되고 있다. 초창기 IP 네트워크는 데이터 송수신을 목적으로 만들어진 네트워크로 아날로그 신호인 음성을 전달하도록 고안되지는 않았지만 IP 네트워크의 월등히 저렴한 비용 때문에 IP 네트워크를 통해 음성을 주고받을 수 있다면 값비싼 국제 전화료 또는 장거리 전화료를 파격적으로 줄일 수 있는 점에 착안해 꾸준히 IP 네트워크상에서 음성을 전달하려는 기술의 진보가 있어 왔다[9].

1995에 VoIP 포럼이 인터넷상에서 음성 신호를 전

† 종신회원 : 광주여자대학교 컴퓨터학부 교수
논문접수 : 2000년 10월 31일, 심사완료 : 2000년 12월 29일

송하는 전화 제품들과 사설 IP 기업 네트워크간의 상호 호환을 가능하게 하고 이러한 기술을 개선하고자 미국에서 시작되었다. 그 후 세계의 많은 기업들과 연구진의 참여로 IP 네트워크상에서 비디오, 오디오, 데이터 전송을 위한 표준 프로토콜인 H.323 등을 정의하였다. 국내에서는 한국통신을 비롯해 많은 통신 관련 업체들에 의해 VoIP를 이용한 인터넷폰이 점차 실용화 단계에 접어들고 있다. 특히, 인터넷폰에 필수적인 장비인 게이트웨이 장비를 출시하는 업체들이 속속 등장하고 있다. 이러한 VoIP 관련 시장 및 연구 동향은 국내외적으로 (그림 1)과 같은 분야에 집중되고 있다[8].



(그림 1) VoIP 시장 및 연구 동향

인터넷 표준 프로토콜 및 VoIP 표준 프로토콜은 IETF(The Internet Engineering Task Force) 커뮤니티 등을 중심으로 H.323 중심의 ITU-T 표준안 업그레이드, 게이트웨이 제어 관련 프로토콜 국제 표준화 및 프로토콜 스택 개발, RSVP/DiffServ 등 품질(QoS) 보장 기술 적용 방안 반영, H.235 등 보안 기능 강화, Mobile H.323 등 차세대 통신망과의 연동 방안 표준화 작업과 같은 부분을 꾸준히 추진되어오고 있다[8, 9].

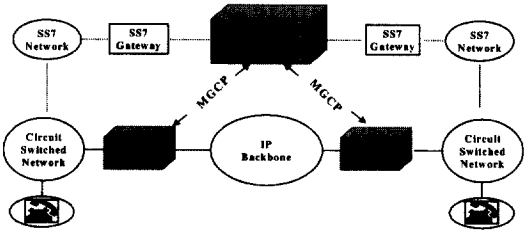
현재 국내외적으로 VoIP 관련 시장의 지속적인 성장으로 인해 VoIP 표준 프로토콜에 대한 표준화 작업과 병행하여 프로토콜 스택의 개발이 절실한 상황이다. 국외에서 게이트웨이를 제어하기 위한 VoIP 표준 프로토콜에 대한 프로토콜 스택의 개발이 진행중이며 일부 제품은 상용화되어 판매되고 있지만 아직은 초기 단계이다. 실제 상용화되어 판매되고 있는 제품은 엄청난 비용을 지불해야 하는 어려움을 가지고 있다. 현재까지 국내에서 MGCP, SDP, MEGACO, SIP 등 게이트웨이 제어 관련 프로토콜 스택의 개발을 위한 연구는 아직까지 열악한 환경과 많은 시간 및 기술을 요하는 관계로 인해 이를 진행하기 위한 시도만 일부 이루어지고 있다.

본 논문에서는 IETF RFC2705[1] 및 RFC2327[2]에서 제시한 MGCP 문법과 SDP 문법을 기반으로 해서 문법 지식적 변환 기법을 적용하여 미디어 게이트웨이가 전달하는 MGCP 및 SDP 요청 메시지를 구조체 형태로 변환하는 엔코더와 미디어 게이트웨이 제어기로부터 전달되는 구조체 형태의 응답 메시지를 텍스트 형태로 변환하는 디코더를 구현하였다. 이를 위해 RFC2705 및 RFC2327에서 제안한 Augmented-BNF 형식의 MGCP 및 SDP 문법을 파서 생성기인 bison의 입력 형태에 적합하게 BNF 형식으로 기술하였으며 어휘 분석기와 파서 생성기를 이용하여 구문 트리를 생성하였다. 또한 MGCP 및 SDP 문법으로부터 생성 가능한 모든 정보를 저장할 수 있는 구조체 형태의 헤더 파일을 생성하였다. 본 연구에서 구현한 엔코더는 텍스트 형태를 갖는 요청 메시지를 구조체 형태로 변환하며 디코더는 구조체에 저장되는 응답 메시지를 텍스트 형식으로 변환하는 동작을 수행한다. 이를 위해 텍스트 형태의 요청 메시지를 구조체 형태로 저장하기 위한 저장소와 응답 메시지를 저장하기 위한 저장소로서 문법으로 생성 가능한 모든 규칙에 적용되는 정보를 저장하기 위해 헤더 파일을 사용하였다.

2. MGCP 및 SDP 특징

2.1 Media Gateway Control Protocol

MGCP는 MGC(Media Gateway Control) 또는 호출 에이전트(call agents)라 불리는 외부 호출 제어로부터 텔레포니 게이트웨이를 제어하기 위해 제시된 표준 프로토콜이다. 이러한 MGCP 프로토콜은 IETF의 RFC에 의해 제안된 것으로 VoIP 솔루션의 핵심 기술로서 (그림 2)와 같이 미디어 게이트웨이(MG)와 미디어 게이트웨이 제어기(MGC) 사이에서 사용된다.



(그림 2) Media Gateway, Media Gateway Controller, MGCP

게이트웨이를 제어하기 위한 MGCP 명령어는 연결

을 처리하는 부분과 엔드포인트를 처리하기 위한 10개의 명령어(EndPointConfiguration, EndpointId, Bearer Information, NotifyRequest, Notify, CreateConnection, ModifyConnection, DeleteConnection, AuditEndPoint, AuditConnection, RestartInProgress", "MoveConnection)로 구성되어 있다. 모든 명령어는 명령어 헤더 부분과 선택적으로 기술되는 세션 부분으로 구성되어 있으며 응답 메시지도 헤더 부분과 선택적으로 기술되는 세션 부분으로 구성되어 있다.[1] MGCP 명령어는 각 명령어의 해당되는 매개변수를 활용하여 다양한 요청(request) 동작 및 응답(response) 동작을 수행한다. 각 명령어가 갖는 매개변수는 명령어의 종류에 따라 서로 다른 매개변수를 갖을 수 있으며 디폴트(default), 선택(option), 금지(forbidden) 형태로 구분된다.

(그림 3)은 "CreateConnection" 명령어의 요청 메시지와 응답 메시지를 Augmented-BNF 형식을 기반으로 작성한 예이다.

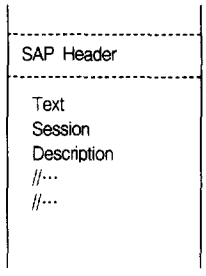
```
//요청 메시지
CRCX 1204 aaln/1@rgw-2567.whatever.net MGCP 1.0 NCS 1.0
C: A3C47F21456789F0
L: p:10, a:PCMU
M: recvonly

//응답 메시지
200 1204 OK
I: FDE234C8
```

(그림 3) "CreateConnection" Request and Reply Message

2.2 Session Description Protocol

SDP는 인터넷 커뮤니티상에서 멀티미디어 세션을 기술하기 위해 IETF2327에서 제안되었다. 세션 기술은 "멀티미디어 세션에 참가하거나 멀티미디어 세션을 발견하기 위해 충분한 정보를 전송하기 위해 잘 정의된(well-defined) 형식"이라고 정의할 수 있다. 따라서 SDP는 멀티미디어 세션을 위한 세션 기술 프로토콜이다. SDP가 사용되는 일반적인 형태는 클라이언트가 이미 알려진 멀티캐스트 주소나 포트에 SAP(Session Announcement Protol)을 사용하여 일정한 주기로 컨퍼런스 세션을 알려주며 (그림 4)와 같이 SDP는 SAP 혼용되어 사용되며 SAP 패킷이라 부른다[2].

SDP는 멀티미디어 세션에서 실질적인 정보를 전송하기 위해 사용되므로 세션 이름, 세션이 활성화되어



(그림 4) SAP Packet

있는 시간 등에 대한 정보를 포함하고 있어야 하며 세션에 참여할 수 있도록 충분한 정보를 전송하는 것은 물론 현재 세션에 참가하지 않고 있는 참가자가 필요로 하는 정보를 제공해 줄 수 있어야 한다. 따라서 SDP는 미디어 타입(예, 비디오, 오디오, …), 전송 프로토콜(RTP/UDP/IP, …), 미디어 포맷 등과 같은 미디어 정보를 가지고 있어야 한다. 또한 IP 멀티캐스트 세션을 위해서는 미디어에 대한 멀티캐스트 주소 및 미디어 대한 전송 포트 등이 전송되어져야 한다. IP 유니캐스트를 위해서는 미디어의 원격 주소 및 접속 주소를 위한 전송 포트 등이 전송되어져야 한다. 세션은 지정된 시간내에서 결합되거나 결합되지 않을 수 있지만 특정한 시간 동안에는 결합되어 있어야 한다. 따라서 SDP는 시작 시점과 정지 시점에서 결합되어 있는 세션에 대한 정보를 전송할 수 있으며 각각 결합된 세션에 대해 특정 시간에 반복되어 전송되도록 지정할 수 있다. 이러한 시간에 대한 정보는 국지적인 시간 정보에 의존하는 것이 아니라 전세계의 시간을 고려하여 지정한다.

SDP 세션 기술은 필드 이름과 속성 이름 등으로 (그림 5)와 같이 3개의 부분으로 기술된다[1, 2].

```
※ Session description
        v =  (protocol version)
        o =  (owner/creator and session identifier).
        s =  (session name)
        I =* (session information)
        u =* (URI of description)
        e =* (email address)
        p =* (phone number)
        c =* (connection information - not required if
             included in all media)
        b =* (bandwidth information)
        One or more time descriptions (see below)
        z =* (time zone adjustments)
        k =* (encryption key)
```

```
        a=* (zero or more session attribute lines)
        Zero or more media descriptions (see below)

  ※ Time description
        t= (time the session is active)
        r=* (zero or more repeat times)

  ※ Media description
        m= (media name and transport address)
        i=* (media title)
        c=* (connection information - optional if
             included at session-level)
        b=* (bandwidth information)
        k=* (encryption key)
        a=* (zero or more media attribute lines)
```

(그림 5) SDP Field Name and Attribute

위와 같은 필드 이름 및 속성 이름 등을 이용하여 (그림 6)과 SDP 메시지 예를 볼 수 있다.

```
v = 0
o = mhandley 2890844526 2890842807 IN IP4 126.16.64.4
s = SDP Seminar
I = A Seminar on the session description protocol
u = http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e = mjh@isi.edu (Mark Handley)
c = IN IP4 224.2.17.12/127
t = 2873397496 2873404696
a = recvonly
m = audio 49170 RTP/AVP 0
m = video 51372 RTP/AVP 31
m = application 32416 udp wb
a = orient:portrait
```

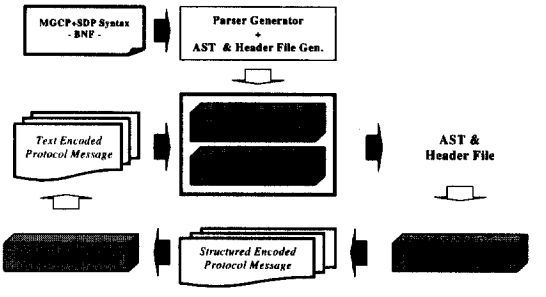
(그림 6) SDP Message

3. MGCP 및 SDP 스택 구현

3.1 프로토콜 스택 구현 개요

본 논문에서는 IETF RFC2705 및 RFC2327에서 제시한 MGCP와 SDP 문법을 기반으로 문법 지식적 변환 기법으로 MGCP와 SDP 요청 메시지를 구조화된 형태로 변환하는 엔코더와 구조화된 형태의 응답 메시지를 텍스트 형태로 변환하는 디코더를 구현한다. 이를 위해 BNF 형식의 문법을 기술하여 구문 분석 동작 후 구문 트리 및 헤더 파일을 정형화된 방법으로 생성할 수 있는 프로토콜 스택을 (그림 7)과 같이 시스템 구성 모델을 설정하고 관련 부분을 구현하였다.

MGCP와 SDP 문법은 bison 파서 생성기에 입력되는 형식인 BNF로 재구성되었으며 문법 기술시에 구문 트리 생성 및 헤더 파일 생성을 위한 의미 수행 코드(action code)를 기술한다. 따라서 파서 생성기는 MGCP와 SDP에 대한 BNF 문법을 입력으로 받아 어휘 분석기, 구문 분석기, 헤더 파일 생성기를 자동으로 생성한다. 또한 입력되는 텍스트 형식의 요청 메시지에 대해 파서는 구문 분석을 수행한 후 구문 트리를 생성하며 텍스트 메시지에 대한 정보를 저장하기 위해 헤더 파일을 생성한다. 헤더 파일 생성기에 의해 생성되는 헤더 파일은 문법에서 허용되는 모든 가능한 규칙에 대한 정보를 저장할 수 있도록 구현되었으므로 모든 가능한 MGCP 및 SDP 메시지 정보를 저장할 수 있다. 따라서 헤더 파일은 입력으로 허용되는 요청 및 응답 메시지에 대한 모든 정보를 저장하고 있는 저장소 역할을 한다. 메시지 엔코더(message encoder)는 입력으로 허용되는 올바른 요청 메시지에 대해 헤더 파일이 저장된 정보를 참조하여 구조체 형식의 구조화된 메시지를 생성한다. 또한 메시지 디코더(message decoder)는 헤더 파일에 저장된 메시지 정보를 추출하여 텍스트 형식의 응답 메시지를 생성한다.



(그림 7) MGCP and SDP Protocol Stack Implementation Model

3.2 어휘 분석 및 구문 분석

어휘 분석기 및 구문 분석기는 MGCP 및 SDP 문법을 기반으로 기술한 형태를 입력으로 받아 요청 메시지에 대한 어휘 분석 및 구문 분석을 수행한다. 이를 위해 본 연구에서는 컴파일러 전단부 구성의 자동화 도구인 flex[4]를 이용하여 어휘 분석기를 자동 생성하였으며 bison[5]을 이용하여 구문 분석기를 자동 생성하였다.

텍스트 형태의 요청 메시지에 대해 구문 분석을 수행한 후 구문 트리 및 헤더 파일을 생성하기 위해 RFC2705 및 RFC2327의 Augmented-BNF 문법을 기반으로 하여 파서 생성기인 bison을 입력을 (그림 8)과 같이 BNF 형식으로 기술하였다.

```
MGCPMessage
        : MGCPCommand    { $$ = maketree(NULL, $1, NIL) ; }
        | MGCPResponse   { $$ = maketree(NULL, $1, NIL) ; }
        | SDP_Info_list  { $$ = maketree(NULL, $1, NIL) ; }
        ;
MGCPCommand
        : MGCPCommandLine TEOL MGCP_para_list TEOL
         { $$ = maketree(NULL, $1, $2, $3, $4, NIL) ; }
        | MGCPCommandLine TEOL MGCP_para_list
         { $$ = maketree(NULL, $1, $2, $3, NIL) ; }
        ;
MGCPCommandLine
        : MGCPVerb WSP_list Tran_ID WSP_list endpointName
          WSP_list MGCPversion
          { nt1 = MakeNode( nonterminal, Buffer ) ;
            $$ = maketree(NULL, $1, $2, nt1, $3, $4, $5, $6, $7, NIL) ;
          }
        ;
MGCPVerb
        : TEPCF          { $$ = maketree(NULL, $1, NIL) ; }
        | TCRCX          { $$ = maketree(NULL, $1, NIL) ; }
        | TMDCX          { $$ = maketree(NULL, $1, NIL) ; }
        | . . .
        | extensionVerb  { $$ = maketree(NULL, $1, NIL) ; }
        ;
        // 생략

SDP_Info_list
        : announcement
         { $$ = maketree( NULL,$1,NIL) ; }
        |
        ;
announcement
        : protoVersion originField session_nameField informationField
                    uriField phoneFields
         { $$ = maketree( NULL,$1,$2,$3,$4,$5,$6,NIL) ; }
        ;
protoVersion
        : T_v TOKEN11 NtDigit TEOL
          { $$ = maketree( NULL,$1,$2,$3,$4,NIL) ; }
        ;
originField
        : T_o TOKEN11 SuitableCharacter NtDigit WSP_list NtDigit
              WSP_list netType WSP_list addrType WSP_list addr
              TEOL
          {$$=maketree(NULL,$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$13,NIL) ; }
        ;
phone
        : TOKEN10 NtPDigit space_DIGIT_list
          { $$ = maketree( NULL,$1,$2,$3,NIL) ; }
        ;
space_DIGIT_list
        : space_DIGIT_list space_DIGIT
          { $$ = maketree( NULL,$1,$2,NIL) ; }
        | space_DIGIT
          { $$ = maketree( NULL,$1,NIL) ; }
        ;
// 생략
```

(그림 8) bison Input Grammar

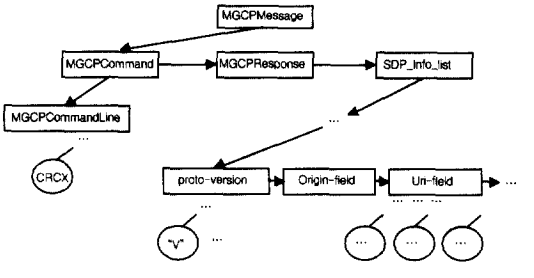
3.3 구문 트리 및 헤더 파일

올바른 요청 메시지 입력에 대해 어휘 분석 및 구문 분석을 수행한 후 구조체 형식을 가진 엔코딩된 메시지를 생성하기 위해 입력 메시지에 대한 (그림 9)와 같은 노드 구조를 갖는 구문 트리를 생성한다.

```
typedef enum kind { terminal, nonterminal } n_kind ;
typedef struct nodetype {
    n_kind node_kind ;  //노드 종류
    char  *node_name ; //노드 이름
    struct nodetype *brother ;
    struct nodetype *son ;
} NODE ;
```

(그림 9) AST Node Structure

구문 트리 생성에 필요한 (그림 9)의 노드 구조를 이용하여 본 연구에서 기술한 문법 생성 규칙에 따라 구문 트리를 생성하기 위해 해당 문법이 축약(reduce) 또는 이동(shift) 되었을 때 구문 트리를 구성하는 동작 코드인 makeNode() 및 makeTree() 함수를 구현하였다. 실질적으로 요청 메시지에 대한 구문 트리 생성 결과는 (그림 10)과 같은 구조를 갖는다.



(그림 10) AST Structure

3.4 엔코더 및 디코더

헤더 파일은 문법으로부터 생성 가능한 모든 규칙에 대한 정보를 저장하기 위해 사용되는 (그림 11)과 같은 구조체로서 구문 분석시에 헤더 파일 생성 루틴에 의해 생성된다. 이를 위해 본 연구에서는 문법 기술시에 각 문법 규칙으로부터 생성될 수 있는 모든 가능한 규칙에 대한 정보를 저장할 수 있도록 구문 트리 생성 정보를 참조하여 의미 수행 코드를 작성하였다. 따라서 헤더 파일은 요청 메시지를 구조체 형식으로 변환하여 저장하기 위한 저장소로서 사용되며 또한 미디어 게이트웨이 제어기로부터 헤더 파일의 구조체에 저

장되는 응답 메시지를 텍스트 메시지로 변환하여 전달하기 위한 정보를 저장하는 곳으로 사용된다.

```
typedef struct MGCPCommand {
 unsigned short choice;
 #define      rqEPCF_chosen 1
 // . . .
 #define      rqMVCX_chosen 10
 union {
        EndPointConfigurationCommand    rqEPCF;
        // . . .
        MoveConnectionCommand           rqMVCX;
    } u;
} MGCPCommand;

typedef struct MGCPRequest {
 unsigned cha            bit_mask;
 MGCPCommandHead    mgcpCommandHead;
 MGCPCommand        mgcpCommand;
} MGCPRequest;

// . . .

typedef struct SessionInformationSDP {
    unsigned short  length;
    unsigned char   value[16];
} SessionInformationSDP;

typedef struct UriISDP {
    unsigned short  length;
    unsigned char   value[32];
} UriISDP;

typedef struct ConnectionDataSDP {
    NetworkType     networkType;
    AddressType     addressType;
    struct {
        unsigned short  length;
        unsigned char   value[16];
    } cnxAddress;
} ConnectionDataSDP;
```

(그림 11) Header File Structure

실질적으로 MGCP 및 SDP 입력 메시지에 대한 검증을 위해 다음과 같은 요청 메시지에 대한 구문 트리의 구성은 최종적으로 (그림 12)와 같은 터미널로 구성

```
// MGCP 메시지
RQNT 1201 aaln/1@rgw-2567.whatever.net MGCP 1.0 NCS 1.0

// SDP 메시지
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
. . .

//생성된 AST
                    [T] RQNT
                    [T] 1201
                    [T] aaln
                    [T] /
                    [T] 1
                    [T] @
                    [T] rgw-2567.whatever.net
                    [T] MGCP
                    [T] 1.0
                    [T] NCS
                    [T] 1.0

                    [T] v=0
                    [T] o=mhandley
                    [T] 2890844526
                    [T] 2890842807
                    [T] IN
                    [T] IP4
                    [T] 126.16.64.4
                    [T] s=SDP
                    . . .
```

(그림 12) Example of the AST Generation

된다.

구문 분석 과정을 통해 생성된 요청 메시지의 터미널은 문법에서 생성할 수 가능성의 일부에 해당하므로 이러한 터미널을 헤더 파일 생성기에 의해 생성된 구조체에 저장하였다. 따라서 엔코더의 기능은 생성된 터미널을 (그림 11)과 같은 구조체에 직접 저장한다. 구조체에 저장된 정보는 미디어 게이트웨이 제어기에 의해 사용되며 응답 메시지는 다시 구조체 저장되어 디코더에 의해 다시 텍스트 형식으로 변환된다. 입력에 대한 응답 메시지의 최종 출력 결과는 헤더 파일에 저장된 메시지의 내용을 역 추적하여 다시 텍스트 메시지로 형태로 변환하는 과정을 통해 확인하였다. 이를 좀더 실질적으로 검사하기 위해 입력되는 요청 메시지를 엔코더에 의해 구조체로 저장한 후 다시 구조체에 저장된 정보를 읽어 처음과 일치하는 메시지를 생성하는 결과를 통해 본 시스템의 타당성을 검증하였다.

4. 결론 및 향후 연구과제

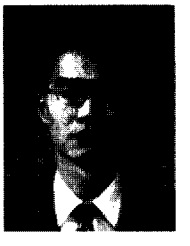
본 논문에서는 IETF RFC2705 및 RFC2327에서 제

시한 MGCP 문법과 SDP 문법을 기반으로 해서 문법 지식적 변환 기법을 적용하여 미디어 게이트웨이가 전달하는 MGCP 및 SDP 요청 메시지를 구조체 형태로 변환하는 엔코더와 미디어 게이트웨이 제어기로부터 전달되는 구조체 형태의 응답 메시지를 텍스트 형태로 변환하는 디코더를 구현하였다. 이를 위해 RFC2705 및 RFC2327에서 제안한 Augmented-BNF 형식의 MGCP 및 SDP 문법을 파서 생성기인 bison의 입력 형태에 적합하게 BNF 형식으로 기술하였으며 어휘 분석기와 파서 생성기를 이용하여 구문 트리를 생성하였다. 또한 MGCP 및 SDP 문법으로부터 생성 가능한 모든 정보를 저장할 수 있는 구조체 형태의 헤더 파일을 생성하였다. 본 연구에서 구현한 엔코더는 텍스트 형태를 갖는 요청 메시지를 구조체 형태로 변환하며 디코더는 구조체에 저장되는 응답 메시지를 텍스트 형식으로 변환하는 동작을 수행한다. 이를 위해 텍스트 형태의 요청 메시지를 구조체 형태로 저장하기 위한 저장소와 응답 메시지를 저장하기 위한 저장소로서 문법으로 생성 가능한 모든 규칙에 적용되는 정보를 저장하기 위해 헤더 파일을 사용하였다.

본 논문에서 시도한 문법 지시적 변환 방법을 이용하여 VoIP 표준 프로토콜 문법에 대한 프로토콜 스택에 관한 연구는 현재까지 MGCP와 SDP에 국한되어 시도되었지만 앞으로 MEGACO, SIP와 같은 프로토콜에서도 본 연구에서 제안한 모델을 기반으로 구현할 예정이다. 또한 본 연구 결과에 대한 성능 평가 방법을 고려하지 않고 단지 모델에 대한 구현으로 국한되어 있지만 현재 다른 연구 결과와 비교 분석을 진행하여 성능 개선을 진행중이다.

참 고 문 헌

[1] IETF, rfc2705, Media Gateway Control Protocol (MGCP) Version 1.0.
[2] IETF, RFC2327, SDP : Session Description Protocol.
[3] ITU-T, APC-1855, Proposal For an Advanced Audio Server Package for H.248.
[4] FLEX : http://www.combo.org/lex_yacc_page/.
[5] BISON : http://www.combo.org/lex_yacc_page/.
[6] Steven S. Muchnick, "Advanced Compiler Design and Implementation," Morgan Kaufmann, 1997.
[7] Mauricio Arango, Christian Huitema, Simple Gateway Control Protocol 1.1, draft-huitema-mgcp-v1-02.txt, July 30, 1998.
[8] http://www.netmanias.com.
[9] http://www.hsswrold.com.
[10] http://www.prptocols.com.
[11] http://www.catapult.com.
[12] http://www.ietf.com.
[13] http://www.iab.org/iab/.
[14] http://www.iana.org/.
[15] http://www.irtf.org/.



고 광 만

e-mail : kkman@www.kwu.ac.kr

1991년 원광대학교 컴퓨터공학과 졸업

1993년 동국대학교 컴퓨터공학과 석사학위 취득

1998년 동국대학교 컴퓨터공학과 박사학위 취득

1998년~현재 광주여자대학교 컴퓨터학부 전임강사

관심분야 : 프로그래밍 언어론, 컴파일러 구성론 임