SIPIS - SIP Info Service

Peter Vicman, Simon Nedok, Boštjan Vlaovič, Zmago Brezočnik
Faculty of Electrical Engineering and Computer Science
University of Maribor
Smetanova ulica 17, SI-2000 Maribor, Slovenia
{peter.vicman, simon.nedok, bostjan.vlaovic, brezocnik}@uni-mb.si

SIPIS - SIP Info Service

In this paper SIP Info Service (SIPIS) is presented. The main purpose of this system is to deliver prerecorded audio data (traffic news, weather reports, etc.) over the Internet to the receiver using RTP (Real-time Transfer Protocol) with the UDP/IP (User Datagram Protocol/Internet Protocol) as its transport protocol. For signaling purposes SIP (Session Initiation Protocol) and SDP (Session Description Protocol) are used. The system can be called by any SIP user agent. The main stress in the development of this system was on SIP and SDP. SIPIS consists of UDP server, SIP and SDP parser and user agent for call control. UDP server and client are used for transmission of UDP packets carrying SIP and SDP messages. Parsers dismember corresponding messages and SIPIS user agent's job is to receive requests for audio information and then play it to the calling user.

1. Introduction

SIPIS is used to deliver prerecorded audio messages over the Internet to the user of the system. For call control signaling the SIP protocol is used. A user establishes session with his/her SIP user agent to the SIP address of the system. It is composed from suffix, which defines network address of the system, and prefix, which defines wanted audio information. If the media types are compatible at both sides, audio connection is established and desired audio information is played to the user. The session can be terminated by SIPIS at the end of the recording or by the user at any desired time.
During the implementation of the system, we have focused on the SIP and SDP [1][2]. Therefore, we have developed UDP server and client, SIP and SDP parser and processes with Call Control and Protocol Control. For media transmission slightly modified sfmike modul was used which is part of the Speak Freely for Unix [7] public domain program. This application allows talking over a network with other Unix or Windows workstations.
In the next section, we present short introduction of the SIP protocol and its characteristics. Third section describes basic characteristics of SDL (Specification and Description Language). Next, hierarchical structure of the SIPIS system is presented. Fifth section describes tools and methods used for parser development. In the sixth section, Agent Manager and Agent State Machine are presented. Seventh section shows some call examples with the system. We conclude with final remarks and observations.

2. Session Initiation Protocol

SIP was developed under the IETF (Internet Engineering Task Force). It is a signaling protocol used for creating, modifying, and terminating multimedia sessions. It can be used for point to point (unicast) or multicast sessions. SIP is an alternative to the part of H.323 bundle of protocols – call setup and signaling from H.225. H.323 was developed by ITU (International Telecommunications Union).
SIP defines procedures for negotiation of user capabilities (supported protocols, codecs, …), supports personal mobility, is independent of the lower layer transport mechanism and can be easily extended or updated with additional capabilities. With design similar to HTTP (HyperText Transfer Protocol), it enables usage of already known programming techniques. SIP is application-layer signaling control protocol. It can be used with UDP or TCP. With unreliable transport protocols its own protocol control mechanisms are used. SIP invitation carries SDP session description. It allows participants to see each other's capabilities and agree on a compatible subset of media types. The multimedia content is transported with the RTP. Multimedia sessions include conferences, distance learning applications, Internet telephony, messaging service and similar applications. SIP can invite persons, “robots”, and services to the unicast or multicast sessions.
For establishment and termination of multimedia session the following steps are required:

- User location: determination of the end system to be used for communication – IP address of the user's terminal equipment;
- User capabilities: determination of the media and media parameters to be used;
- User availability: determination of the willingness of the called party to engage in communications;

- Call setup: establishment of call parameters at both called and calling party;
- Call handling: including transfer and termination of calls.

SIP follows client/server architecture. The main logical entity is user agent, which is an end point for communication. It acts as user agent client and user agent server for the duration of the call. SIP is request-response protocol. User agent clients send requests and user agent servers respond to that requests with responses. Requests include method, which defines the nature of the request and an address to which the request should be send. The response includes status code, which defines the type of the response.
A successful SIP invitation consists of two requests, "INVITE" followed by "ACK". The "INVITE" request invites the callee to join a session. After the callee has agreed to participate in the session with a "200 OK" response, the caller confirms reception by sending an "ACK" request.
The "INVITE" request typically contains a session description usually written in SDP format. It provides the called party with enough information to join the session. The callee receives information about media types and formats that the caller is willing to use and where it wishes the media data to be sent.
If the callee wishes to accept the call, he/she responds to the invitation by returning a similar description listing the media types it wishes to use.

3. SDL programming language

SDL is a programming language for the specification and description of systems at conceptual level as well as for detailed description of parts or a whole system. It is mostly used for unambiguous formal specification and description of the telecommunication systems. In general, specifications may cover various aspects of a system, such as hardware design, physical dimensions, power consumption and so on. The SDL concentrates on the behaviour of the system. Behaviour description can be on abstract or implementation detailed level. Specifications are concerned with the black box view of the system. Little or no consideration is given to the implementation issues. Descriptions, on the other hand, reflect the structure of a planned or implemented system. SDL does not differentiate between specification and description. Therefore, SDL specification may or may not reflect the structure of a design based on the specification [3].
SDL was first defined in 1980 by CCITT (Comité Consultatif International de Téléphonique et Télégraphique) as recommendation Z.100. In the years 1984, 1988, 1992, 1996, and 1999 new versions of SDL arose. In SDL-2000 better support for object modeling is provided.
The strength of SDL lies in its recognition as an international standard. It has the commitment and support of ITU and ISO. This provides security for investment and training, as it is ensured that SDL will be maintained and supported in the future.

4. Hierarchical structure of SIPIS

Hierarchical structure of SIPIS is shown in Figure 1. UDP server/client and parser were written in C programming language. UDP server receives incoming messages and UDP client sends outgoing messages. Parser’s job is to recognize received message, to dismember it to logical parts, check the correctness of the message and then send collection of basic elements to the agent manager (AM). AM and agent state machine (ASM) were written in SDL [3][6]. AM is a static process – it exists from the beginning to the end of the system run. It mediates received information (protocol data) to the ASM. It is a a dynamic process – each call has its own ASM process. Received protocol data is forwarded from AM to appropriate ASM. If received message initiates new call, AM creates a new ASM. After ASM receives protocol data and executes appropriate actions it prepares necessary data for the response. Prepared SIP message is sent to the AM and then forwarded to UDP client for delivery to the calling user.

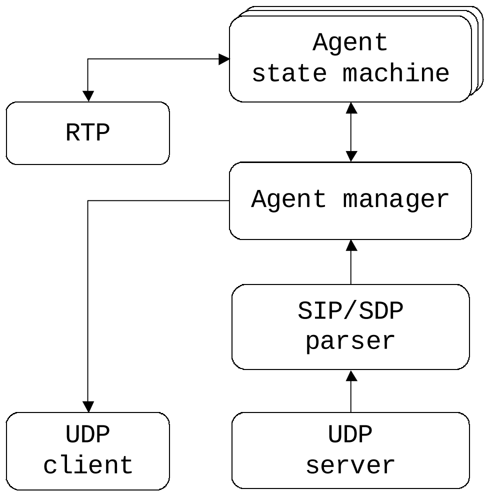

Figure 1: Hierarchical structure of the SIPIS modules.

5. Parser

First operation after receiving the SIP message is to recognize (parse) it. The received message is first split in two parts, with SIP and SDP message, respectively. Both parts are sent to the corresponding parser. Parser code is produced with tools for parser generation [5]. Complete parser is composed from two parts: parser and lexical scanner. Standard tools in Unix and Linux environments are Flex and Bison [4]. Therefore, we have chosen them for this purpose.

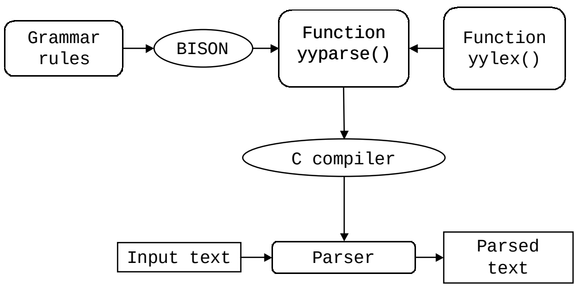

Figure 2: Using the Bison tool.

Bison is a general-purpose parser generator. It receives grammar file with SIP or SDP message description as its input. Grammar is described with BNF (Bacus-Naur Form). The output of the program is a C source file that parses the text described by the grammar. The job of the parser is to check the grammar of the message – it checks if received message has correct sequence of basic symbols. Figure 2 shows how the tool is used.

Grammar description in the input file is divided in three sections. Sections are separated by a line with two % signs:

….. declarations …..
%%
….. grammar rules …..
%%
….. user subroutines …..

Declarations and user subroutines sections are optional. Grammar rules section is the most important one. It includes grammar and actions in C programming language which are to be taken. Grammar in Bison is composed from a number of rules. Every rule is started with nonterminal symbol, followed by nonterminal symbols, terminal symbols, and actions. Actions are executed when parser recognizes a match for that grammar rule. In our case the actions include code for writing recognized symbol to the programming structure containing parsed message (SIP and SDP protocol data). Tokens are terminal symbols, returned from scanner as shown in Figure 3. When parser needs the next token it calls yylex scanner function.
Flex is a tool for generating scanners: programs which recognize lexical patterns in text. Flex reads given input file with a description of a scanner to generate. The description is in the form of pairs of regular expressions and C code, called rules. Scanner is examining the input text, recognizing symbols and then mediate them to the parser on its demand. The basic task of scanner is recognizing patterns in text. When it finds new pattern it executes appropriate action which is written in C code. Usually, the action sends this pattern to the parser.

Usage of Flex tool and structure of its input file is similar to Bison's.

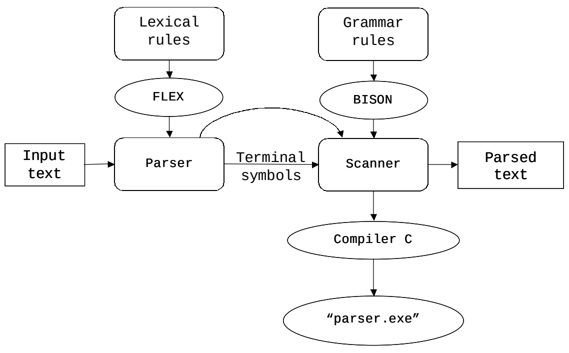

Figure 3: Connection bettwen Bison and Flex.

6. Agent Manager and Agent State Machine

Agent manager manages agent state machines. As we have already mentioned it is always present during the system run. Beside managing ASMs, its job is to analyse received messages and forward them to appropriate ASM – it acts as a message multiplex.
Each call has its own agent state machine. ASM includes behavioural logic of the SIPIS. It is written in SDL language with Verilog SDL ObjectGeode version 1.0. ASM is responsible for Call Control, Protocol Control and control of RTP module. When particular ASM is not needed anymore, it kills itself. Figure 4 shows small part of the SDL language code.

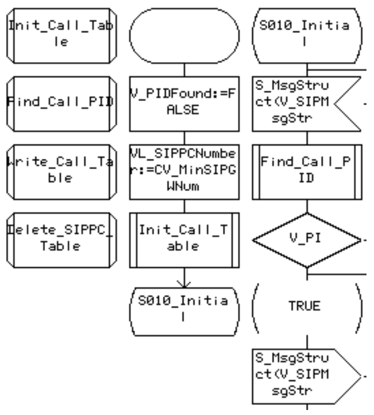

Figure 4: Example of SDL code.

Next, we will describe actions taken during user's call to the system.

7. Call example

When a new INVITE request is received, parser recognizes and verifies received headers of the message and forwards it to the AM. It checks headers that identify a call and determines if message belongs to existing call. If ASM for this call does not exist, AM creates one, otherwise protocol data is forwarded to the

appropriate ASM. Each creation is recorded in the table of existing ASM processes.
ASM's task is to receive SIP and SDP protocol data, execute specific actions and send response back to the user of the system. From SDP description ASM determines user's audio capabilities. If the requested media is available, return confirmation is sent to the calling user with "200 OK" response. After the agent receives ACK request from the calling user it instructs RTP module to create appropriate connection. In our implementation only audio is used. RTP module starts sending requested audio information after a slight delay. The call flow between user and the SIPIS is shown in Figures 5, 6, and 7.

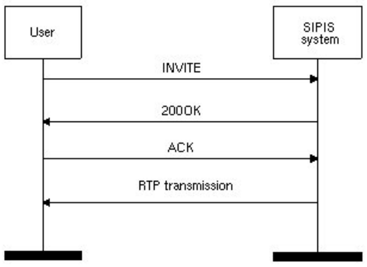

Figure 5: Call flow between user and SIPIS.

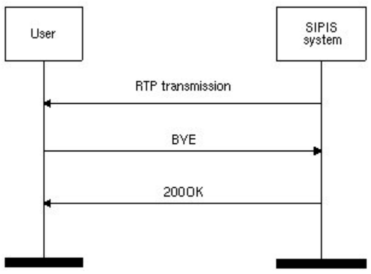

Figure 6: User call termination.

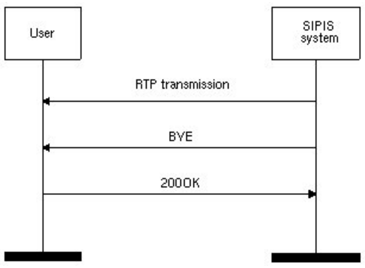

Figure 7: SIPIS call termination.

The user can interrupt transmission with the "BYE" request at any time of the call. SIPIS will terminate transmission at the end of the audio message.
Termination of connection is confirmed with the "200 OK" response. When SIPIS gets "BYE" request, ASM responds with "200 OK", waits for additional 40 seconds (due to possible retransmissions), notifies AM and kills itself. ASM updates the table of existing ASM processes.

Protocol Control is necessary due to unreliable transport protocol (UDP). It is realized with retransmissions as defined in SIP protocol.

8. Conclussion

In this paper a simple system for audio info services is presented. This application enables users to listen prerecorded audio information on his/her demand. It can be used for various automatic voice message systems like traffic, cinema and weather information. It can be easily extended to provide additional functionality. Our main objective was to test the implementation of SIP and SDP parser and Protocol Control logic. SIP protocol can be found in M-BONE, Internet telephony, softswitch architecture of next generation telecommunication networks, messaging service in Windows XP and other applications. SIPIS is still in its development phase. At the time of writing it is not jet fully completed, but first tests have been successful. Developed parser successfully recognizes most test messages and tortured test messages used on SIP interoperability test events.
We have tested SIPIS with applications (user agents) from different companies and universities. We have noticed that there are some interoperability problems. In the future we are planning to support as many user agents as possible.
SIPIS can be extended to support video distribution, Short Message Service, personal answering machine, and many other interesting applications. We are planning to use it as our test bed for SIP and SDP parser, Call Control and Protocol Control logic.

Literature

[1] M. Handley, H. Schulzrinne, E. Schooler and J. Rosenberg: "SIP: Session Initiation Protocol", IETF RFC2543, May 29, 2001.
[2] M. Handley, V. Jacobson and C. Perkins: "SDP: Session Description Protocol", IETF RFC 2327, 30 April 2001.
[3] F. Belina, D. Hogrefe and A. Sarma: SDL with APPLICATIONS from PROTOCOL specification, Prentice Hall, 1991.
[4] Brest Janez: Tool for automatic program analysis, Diploma work, University of Maribor 1995. In Slovene.
[5] Brian W. Kernighan, Dennis M. Ritchie: The "C" programming language, Prentice Hall, 1988.
[6] ObjectGEODE Method Guidelines, Verilog, april 1996.
[7] Speak Freely for Unix 7.2, http://www.fourmilab.ch/speakfree/unix/sfunix.html