TR-IT-0302

TIOP Protocol Connection Controller for ATR-MATRIX

グルーン ライナー
Rainer Gruhn

中村篤
Atsushi Nakamura

1999.3

We describe the implementation of the TIOP protocol to establish internet connections between speech recognition, translation and synthesis systems. The specification is based on a draft by Hiyan Alshawi [1] (Copyright AT&T Corp.) and the SIP 2.0 standard as set by the Internet Engineering Task Force [5]. This report is accessible for ITL members via /home/rgruhn/paper/tr99/.

Contents

1 **Introduction** 1

2 **The Customer Care Project** 2
- 2.1 CCS and Speech Translation Systems . . . 2
- 2.2 CCS Services Covered in this Project . . . 2
- 2.3 Examples . . . 2
  - (2.3.1) Asking for Country or Area Code . . . 2
  - (2.3.2) Asking for Charge Information . . . 3
  - (2.3.3) Asking for Time at Callee Site or Time Difference . . . 3

3 **The TIOP Protocol** 4
- 3.1 SIP-Messages . . . 4
  - (3.1.1) INVITE . . . 5
  - (3.1.2) OK . . . 6
  - (3.1.3) ACK . . . 7
  - (3.1.4) BYE . . . 7
- 3.2 The Current TIOP Version . . . 7

4 **Messages Inside ATR-MATRIX** 8
- 4.1 SipPackets from Main Controller to TIOP connection controller . . . 8
- 4.2 SipPackets to Main Controller from TIOP connection controller . . . 9
- 4.3 Timeline . . . 10
- 4.4 Problems . . . 10

5 **Summary and Future Directions** 11

References 13

A **Program TIOPConnectController.py** 15
- A.1 Installation . . . 15
- A.2 Behaviour . . . 15
- A.3 Configuration . . . 16
- A.4 Troubleshooting . . . 16

B **Technical Data** 19
- B.1 Host IP Addresses and Port Numbers at ATR . . . 19
- B.2 Host IP Addresses and Port Numbers at AT&T . . . 19
- B.3 Minimal SIP Messages . . . 19

1 Introduction

ATR's speech recognition and translation system ATR-MATRIX [7] is already successfully used in the CSTAR project. It provides an interface for a Japanese traveller to communicate with a foreign, e.g. an American travel agency without knowledge of any language but Japanese.

Another new application of ATR-MATRIX is the Customer Care Service (CCS) project in cooperation with AT&T which aims to an international telephone information retrieval system, offering for example country and area telephone numbers in a language of the callers choice.

This report describes the specification and implementation of a communication system working as an interface to ATR-MATRIX using the Translation Inter-Operating Protocol (TIOP) proposed by [1].

Chapter 2 provides an overview of the CCS project. Chapter 3 describes TIOP and analyzes the TIOP messages from the implementers point of view. In chapter 4 the connection to the ATR-MATRIX internal SipPacket[1] message format is explained. Finally, a conclusion summarizes and gives future prospects. The appendix provides installation, usage, configuration and troubleshooting information as well as connection experiment data.

[1] As two similar abbreviations will be used in this report, it is necessary to clarify their meaning and difference in the beginning: *SIP messages* are used in TIOP for the initial handshake. SIP stands for Session Initiation Protocol. *SipPacket* is the data and control message format used for communication between the various parts of ATR-MATRIX. Sip stands for System Integration Project, an old name for ATR-MATRIX.

2 The Customer Care Project

Purpose of the Customer Care Service (CCS) is to provide assistance to telephone users, helping them to conveniently and comfortably make their calls. CCS, in general sense, may include the following assistance services:

- Number directory
- Charge account change (e.g. for collect calls)
- Operator calls
- Various information requests about telephone calls (e.g. about charges)

2.1 CCS and Speech Translation Systems

When people stay in foreign countries and want to have assistance in making telephone calls, they might be in trouble because a local telephone carrier company may not provide CCS in their mother language. Speech translation systems can help in such a situation and make users feel as if they had assistance in their mother language. The development of a prototype system is being carried out in collaboration with AT&T Laboratories. ATR is in charge of developing a sub-system on the telephone users side which provides

- Japanese speech to English text conversion and
- Japanese text to speech conversion.

AT&T Laboratories is in charge of developing a sub-system on the operators side.

2.2 CCS Services Covered in this Project

This project will base on a limited set of CCS services:

- The customer is assumed to be a Japanese-speaking user in the USA
- Information is only available for overseas calls. Specifically, the user may only ask for
  - country and/or area codes,
  - charge informations and
  - local time at callee site or time difference.

CCS is likely to become a real world application of speech recognition, translation and synthesis systems soon. Therefore, the most important objective in this project is to do the first step towards an automated assistance service offering a wide variety of information which will be made available in many languages.

2.3 Examples

The following example dialogues illustrate the capabilities of the projected system.

(2.3.1) Asking for Country or Area Code

Client: Hello, International Telephone Customer Service.
Customer: Ah, hello, could you tell me the country code for Portugal, please?
Ah, and also, ah, what's the area code of Lisbon?
Client: If you want to make a direct call, first dial the 0815. Then, Portugal is three five one.
Customer: Three five one.
Client: Yes, and the Lisbon area code is one.
Customer: One? I see. Thank you very much.
Client: Thank you very much for calling.

(2.3.2) Asking for Charge Information

Client: Hello, International Telephone Customer Service.
Customer: Hello.
Client: This is "the telephone company". How may I help you?
Customer: Excuse me, but how much is a call to Seoul?
Client: For a direct call? It depends on the time, at current time it's twenty-eight Yen per six seconds.
Customer: Twenty-eight Yen per six seconds.
Client: Yes, but it's eighteen yen per six seconds after nine pm only.
Customer: So it's a hundred eighty Yen for one minute?
Client: Yes, that's correct.
Customer: Okay, I understand. Thanks.
Client: Thank you for calling the service line.
Customer: Bye.

(2.3.3) Asking for Time at Callee Site or Time Difference

Client: Hello, International Telephone Customer Service.
Customer: Hello, my name is Johnson, I have a question. Do you know what time it is in Jamaica?
Client: Jamaica. Yes, I will look it up. Just a moment, please. Sorry to keep you waiting. It's three fifty-five a.m.
Customer: Ah, how much time difference is that?
Client: They are fourteen hours behind.
Customer: Ah, I see. Thanks.
Client: It's now early morning of the third. Thank you for your call.

3 The TIOP Protocol

The implementation of a communication controller using TIOP follows the AT&T specification draft [1].

TIOP is designed for communication between servers, each doing speech recognition, translation and synthesis from one single language. One machine performs speech recognition and translation (to the partners language), sends the text and the partners machine will generate synthesized speech.

The TIOP standard divides communication into one main connection, called SIP connection, and typically two or more data connections. Messages sent over the SIP connection initiate the session by telling the other side about IP addresses and ports where data of specified types will be accepted. Data transfer is done on separate connections called TIOP connections. Figure 1 explains the sceme of a server-server system using TIOP protocol for communication.

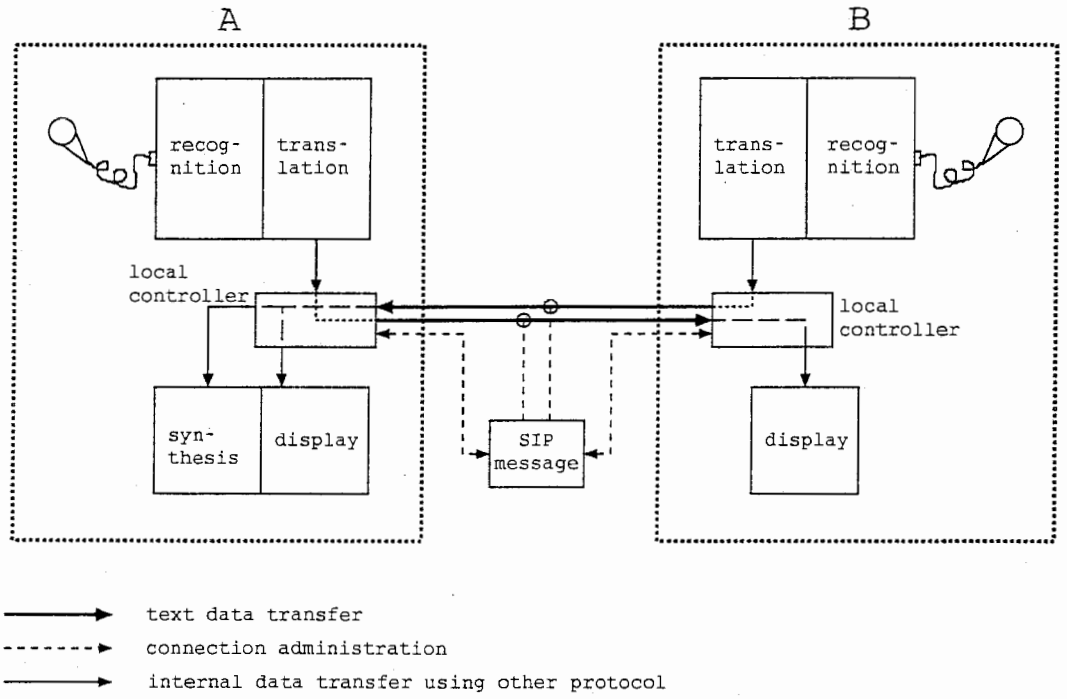

Figure 1: An overview of TIOP communication: First, SIP messages are exchanged. Using the information in these messages, independent data connections are established for exchange of translated text. Local controllers forward the data received to the display or speech synthesis devices

To establish data communication, TIOP uses four types of messages in SIP syntax:

- INVITE, the initial message sent from client to server, telling the server at which host and port the client wants to receive data and which kind of data. Which side will act as client and which as server has to be negotiated before the session begins. In Figure 1 either A or B could be client.
- OK, the answer to an INVITE message, both confirming the read information and telling the client, at which host and port the server will accept data.
- ACK, sent by the client, confirming the OK message, thus completing initiation.
- BYE, a message that can be sent by either side, ending the entire session.

There is no refusal or error handling message provided by TIOP version 1.0 (which is the current version).

3.1 SIP-Messages

TIOP messages are based on the Session Initiation Protocol (SIP) 2.0 standard [5], an internet protocol designed to enable the invitation of users to multimedia conferencing sessions. The standard was set by the Multiparty Multimedia Session Control (MMUSIC) working group of the Internet Engineering Task Force.

SIP messages consist of two parts: header and body.

- The header contains message type, information about caller and callee and the content length of the message body. The message style is very much based on HTTP. The header part is delimited by an empty line.
- The second part is the message body, containing information which kind of data is expected where, including host IP address, port number and data type. Only INVITE and OK messages have message bodies. The body uses Session Description Protocol (SDP) [4], as suggested in the SIP specification.

The best way to explain the SIP messages used in TIOP is by example. Please note that this explanation is implementation-oriented, a more formal, official definition is given in [1]. ATR is assumed to run an SIP server (on atra57 port 20000, which must be announced to the client somehow before the session).

(3.1.1) INVITE

AT&T (at host 135.207.20.125) sends an INVITE message to atra57:

```
INVITE sip:rgruhn@atra57.itl.atr.co.jp SIP/2.0\r\n
Via:  SIP/2.0/TCP 135.207.20.125\r\n
From:  Hiyan Alshawi\r\n
To:  Rainer Gruhn\r\n
Call-ID: 123456789@xxx.research.att.com\r\n
CSeq:  1 INVITE\r\n
Subject:  test\r\n
Content-Type: application/sdp\r\n
Content-Length: 199\r\n
\r\n
v=0\r\n\
o=rgruhn 012345678 012345678 IN IP4 135.207.20.125\r\n\
s=prototype\r\n\
m=text 25101 TCP/IP plain\r\n\
c=IN IP4 135.207.20.125\r\n\
a=lang:en\r\n\
a=fmtp:plain charset=ISO-8859-1\r\n\
\r\n
```

The SIP server reads and parses the message linewise. CRLF (\r\n) marks line ends. A detailed practical look at each line follows:

```
INVITE sip:rgruhn@atra57.itl.atr.co.jp SIP/2.0\r\n
```

The first line contains the message type, INVITE, the address of the caller and the SIP version. This line is mandatory and must be on top of the message.

```
Via:  SIP/2.0/TCP 135.207.20.125\r\n
From:  Hiyan Alshawi <sip:Hiyan@research.att.com>\r\n
To:  Rainer Gruhn <sip:rgruhn@itl.atr.co.jp>\r\n
```

These lines contain information how the SIP message was transmitted. They should be written but are not parsed by either side.

```
Call-ID: 123456789@xxx.research.att.com\r\n
```

The call ID is a number set in the INVITE message and kept fixed in all messages belonging to the same session. It should be a unique number; for this test prototype it is implemented as 8 digit random number which was considered to be sufficient.

```
Subject:  test\r\n
```

An optional field for subject information.

```
CSeq:  1 INVITE\r\n
Content-Type: application/sdp\r\n
```

Two lines which are not parsed but written for formal reasons.

```
Content-Length: 199\r\n
```

This line contains important information about the length of the message body in bytes. The ATR implementation requires it.

```
\r\n
```

This empty line marks the end of the message header. It is not included in the content-length byte number. All following lines are in the message body.

```
v=0\r\n\
o=rgruhn 012345678 012345678 IN IP4 135.207.20.125\r\n\
s=prototype\r\n\
```

These three for TIOP optional lines contain redundant information already given in the header, including username, session-ID, network and address type (IN IP4) and machine where the session was created. The subject is also given again. The lines are obligatory in the SDP specification.

```
m=text 25101 TCP/IP plain\r\n\
```

The data type and the port number where the caller accepts data: plain text on port 25101.

```
c=IN IP4 135.207.20.125 \r\n\
```

Host IP address where the caller accepts data.

```
a=lang:en\r\n\
a=fmtp:plain charset=ISO-8859-1\r\n\
```

More precise information about the data expected is given here: language and encoding format. The lang:en item uses language abbreviations as defined in RFC 1766 [2]. Supported values are currently `en` for English and `jp` for Japanese text. The charset is one of `ISO-8859-1` for English and `EUC-JP` or `Shift_JIS` for Japanese text. The charsets are defined by IANA [3].

```
\r\n
```

This newline is not part of the SIP specification but required by the AT&T implementation as end of message body marker and thus must be written.

(3.1.2) OK

After the INVITE was read and parsed by ATR, a OK message is sent to AT&T:

```
SIP/2.0 200 OK\r\n
Via:  SIP/2.0/TCP 133.186.34.57\r\n
From:  Rainer Gruhn\r\n
To:  Hiyan Alshawi\r\n
Call-ID: 123456789@xxx.research.att.com\r\n
CSeq:  1 INVITE\r\n
Content-Length: 192\r\n
\r\n
v=0\r\n\
o=hiyan 012345678 012345678 IN IP4 135.207.20.125\r\n\
s=prototype\r\n\
m=text 20001 TCP/IP plain\r\n\
c=IN IP4 133.186.34.57\r\n\
a=lang:jp\r\n\
a=fmtp:plain charset=EUC-JP\r\n\
\r\n
```

Besides acknowledging the INVITE message, an OK provides the same information about data type, host and port as the INVITE message does. The message type (OK) is identified by the first line. The 200 is derived from the SIP status coding system: 200 is the code for a successfully received, understood and accepted message. Other codes values are not specified in TIOP 1.0.

(3.1.3) ACK

After reading and parsing the OK message, AT&T acknowledges it by sending an ACK message, completing the handshake:

```
ACK sip:rgruhn@atra57.itl.atr.co.jp SIP/2.0\r\n
Via:  SIP/2.0/TCP 135.207.20.125\r\n
From:  Hiyan Alshawi\r\n
To:  Rainer Gruhn\r\n
Call-ID: 123456789@xxx.research.att.com\r\n
CSeq:  1 ACK\r\n
Content-Length: 0\r\n
\r\n
```

The ACK message is read but not really parsed, the only important information is the word `ACK`. As there is no message body, the content-length is 0. After this message is received, data sockets are opened to exchange data.

(3.1.4) BYE

If one side wants to end the session, it sends a BYE message to the other side. In this example, AT&T decides to end the session and sends the BYE:

```
BYE sip:rgruhn@atra57.itl.atr.co.jp SIP/2.0\r\n
Via:  SIP/2.0/TCP 135.207.20.125\r\n
From:  Hiyan Alshawi\r\n
To:  Rainer Gruhn\r\n
Call-ID: 123456789@xxx.research.att.com\r\n
CSeq:  2 BYE\r\n
Content-Length: 0\r\n
\r\n
```

The TIOP communication controller in this prototype will accept any message containing the string `BYE` in the first line as BYE message, the rest is not parsed.

3.2 The Current TIOP Version

As it is a first baseline system, TIOP 1.0 has several simplifying limits:

- only two parties communicate
- only text is transmitted, no audio or video signals
- each server accepts only one type of data, e.g. ATR MATRIX accepts only plain Japanese text in Shift_JIS format.

Some technical details are not or not clearly written in the specification draft for TIOP 1.0, problems during implementation and testing made their clarification or addition necessary. This includes the following items:

- Both the SIP message and the text data are read linewise. For SIP, the line delimiter is defined as CRLF, which is \r\n.
- One empty line must be between message header and message body. The CRLF characters in this empty line do not count for the content length.
- The empty line at the end of the message body is not mentioned in the SIP specification, but the message reading routine in the AT&T implementation uses this instead of the content-length field as the end of message body marker.
- Field names in the header have to be parsed case-insensitive, but the body items are case sensitive.
- Except for the first line in the header, the order of the lines within the header and the order of the lines within the body is not fixed.

4 Messages Inside ATR-MATRIX

One task of the communication controller is to translate between the TIOP protocal and the inter-controller protocol used within the ATR-MATRIX Speech Translation System. The ATR-MATRIX main controller sends utterance data and control information together on one channel in separate packets, while TIOP demands separation of plain text and SIP messages. Figure 2 shows an overview of the used data formats. The format of the packets used in ATR-MATRIX have the name of SipPacket. In this document, SIP in

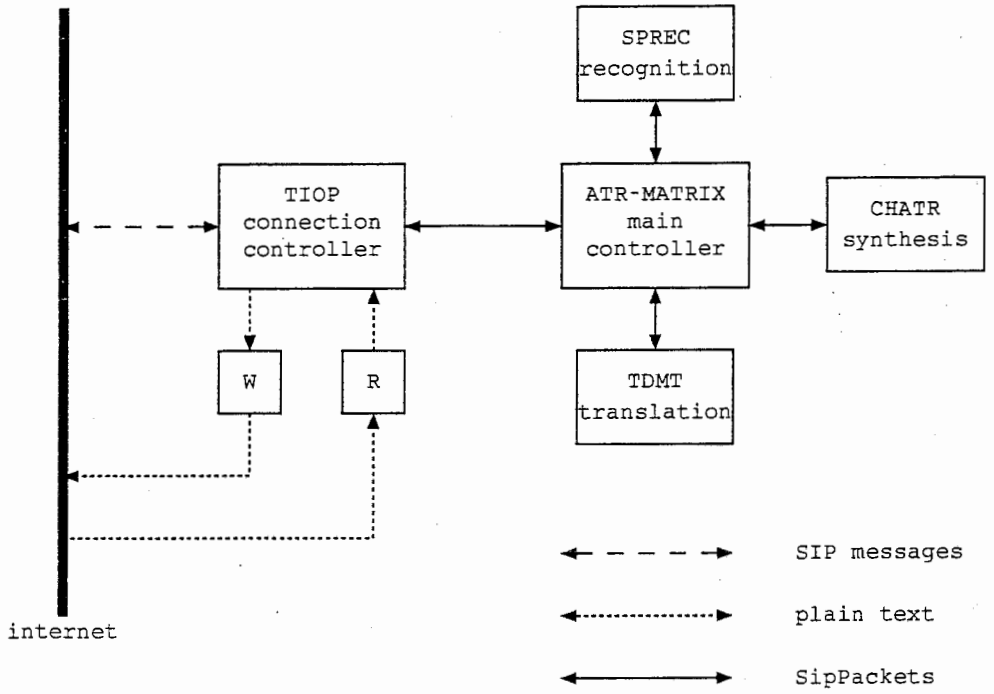

Figure 2: The communication controller understands and produces three different data classes: SIP messages as specified by TIOP, plain data on the data connections and SipPackets for communication with ATR-MATRIX

capitals refers to Sisseion Initiation Protocol (used by TIOP) and "Sip" refers to the System Integration Project, the old name of ATR-MATRIX. A comprehensive explanation about Sip and SipPackets is given in [6].

A SipPacket consists of five fields: receiver, sender, type, data-length and data. Depending on packet type the data field may be empty. An example for a SipPacket is this hello type SipPacket sent from connection controller to main controller:

```
SipMainController.............
SipConnectController..........
language......................
8*********
JAPANESE
```

The fields of a SipPacket are not separated by a newline, this was done in this example for clarity reasons.

This chapter will focus on the communication between the main controller and the TIOP connection controller. The first section will describe the packets send to the connection controller by the main controller and how they are parsed. The second section will list the packets send to the main controller by the connection controller and how they are generated.

The original ATR-MATRIX connection controller had been concepted to be used in the CStar Project, which allows providing additional information about translated utterances. Some of the information available can not be used in TIOP architecture communication, so SipPackets of several types are simply ignored if read or generated based on assumptions from configuration settings. These types are as follows:

4.1 SipPackets from Main Controller to TIOP connection controller

initialize This packet is sent at system startup. At the time the connection controller is ready to read messages from the main controller it is already initialized, so this packet is ignored.

connect This packet tells the communication controller to open up socket connections. In ATR-MATRIX version 2.3.2 or higher this packet contains host and port information in the first two positions in the data field, which is used by the communication controller. Other data items are ignored. If the data field is empty, connection is set up using host and port information from configuration file or default setting. Connect packets in ATR-MATRIX version 2.3.1 or lower do not contain host and port information. A connect packet is confirmed by an accept or error packet.

language This packet is the first of typically four packets that are sent for one recognized or translated utterance: language, speaker, recog/trans, end. Its data field contains the language the text in the following recog or trans packet will be. It is ignored.

speaker This packet contains gender information for the speaker who produced the utterance transcribed in the following recog or trans packet. It is ignored.

recog A recog packet contains the recognized sentence. In the CCS scenario only the translation result is of interest, so this packet is ignored.

trans The translated sentence string can be found after the string `TARGET_SENT=` in the data field of this packet. The translation is filtered out and written to the write socket. All other information in the data field is ignored. If there is no translated sentence in the data field of a trans type packet, a warning is written.

end An end type packet marks the end of one group of packets sent for one utterance. It has no additional information, so it is ignored.

ignore This packet is used in the CStarII scenario to tell the other side to ignore the last sent utterance. As no way to transfer this information is specified in the current TIOP version, this packet is ignored.

disconnect This packet tells the connection controller to close the data connection. A disconnect type packet from the main controller is confirmed by a disconnect packet sent back to the main controller. A BYE type SIP message is written by the connection controller to the other party, all sockets are closed, the child processes terminated, then the controller waits for a new connect packet. All other packets are ignored in this state.

exit Tells the connection controller to terminate. An exit packet is confirmed by a byebye type packet. A BYE type SIP message is written to the connection partner, all sockets are closed, the child processes terminated, then the controller exits.

All other SipPacket types are ignored if received. In verbose mode, a warning is printed to `stderr` for every ignored packet.

4.2 SipPackets to Main Controller from TIOP connection controller

hello This packet tells the main controller that the communication controller has started successfully and is waiting for a connect packet.

accept Accept informs the main controller about a successful connection. As the status of the forked data sockets can not be known by the TIOPConnectController, an accept packet means only the successful exchange of SIP messages.

error An error packet is sent in case of an unsuccessful SIP handshake attempt.

language This packet informs about the language of the utterance the text in the following trans type packet will be in. The data field depends on the configuration item "mylanguage", it is either English or Japanese. Appendix A.3 explaines details about configuration.

speaker The speaker is always male. This is a hardcoded assumption. TIOP does not provide this information, but ATR-MATRIX requires it, so male is set as default.

trans The trans packets data field containes the string `TARGET_SENT=` followed by the received text from the read connection. An utterance number is also provided, the controller simply counts all received lines coming from the connection.

end An end packet always follows a trans packet.

byebye After an exit packet was received, the byebye information tells the main controller about the impending shutdown of the connection controller.

deadcat This packet is a prototype that should be sent when the other side closes the connection. This case does not appear in the CStarII scenario and is thus not implemented in ATR-MATRIX. Still, sending this packet causes the main controller to crash without cleaning up, i.e. without terminating Chatr, ATRSPREC or TDMT. Because of that, the current version of TIOPConnectController sends an error packet instead.

4.3 Timeline

Table 1 shows an example, in which order SipPackets are assumed to be produced.

Table 1: A typical message timeline.

destination	source	type
main	connect	hello
connect	main	initialize
connect	main	connect
main	connect	accept
main	connect	language
main	connect	speaker
main	connect	trans
main	connect	end
connect	main	language
connect	main	speaker
connect	main	recog
connect	main	end
connect	main	language
connect	main	speaker
connect	main	trans
connect	main	end
connect	main	disconnect
connect	main	exit
main	connect	byebye

4.4 Problems

There is a severe structural problem with the connection handling. The main controller sends a connect packet to the communication controller and blocks until either an error or an accept packet is returned.

The architecture was constructed for the CStarII scenario, in which a communication server is running anytime, and all communication partners are clients. Thus, it is reasonable to set a time out limit of e.g. 15 seconds, if connection to the server was not possible in this time, an error is printed.

This is not reasonable for the TIOP type communication. If ATR is client, the same time limit can be applied as in the CStarII communication system. But if the ATR system is acting as server, setting a time limit for a connection is not appropriate. It is hard to coordinate two partners in separate places to set up connection within few seconds. This leads to the question which action is adequate in the server case.

ATR-MATRIX is blocked as long as the communication server does not announce either success or error. But successfully opening a server and waiting for a client to connect is neither error nor success, so none of the available messages should be sent. Also, sending an error message would cancel the connection attempt, leaving only a small time window for connection attempts. Sending an accept packet would confuse the user who would believe to have a successful connection and start communicating.

Sending no message blocks the system. No method of cancelling a connection attempt is provided. Still, this is the setting chosen in the implementation. This is a problem that still has to be solved.

Proposed sollutions are either to change ATR-MATRIX not to block until accept is received or to introduce a new status display telling the user that the system is in the process of connecting.

5 Summary and Future Directions

The TIOP protocol is a promising way to connect two speech recognition, translation and synthesis servers. Successful tests between AT&T in the USA and ATR in Japan have been conducted; communication was fast and stable.

TIOP 1.0 is a minimal basic concept for two-party server-server communication. A possible extension is allowing not only plain text, but also data types such as audio to be transferred. For audio, direct data connections to CHATR or ATRSPREC without an interpreting local controller in between are possible.

An additional data stream for control messages would also be desirable to provide speaker information and error handling. Current assumptions such as "speaker is always male" could be replaced by factual information.

Allowing more parties to take part is not planned as it does not fit to the customer care scenario: There will always be a dialog between one customer and one service provider. Third parties would not make sense.

Acknowledgements

We would like to thank Dr. Seiichi Yamamoto, President of ATR-ITL, and Dr. Yoshinori Sagisaka, Department Head, for giving us a chance to do this research at ITL. We also would like to thank Harald Singer and all members of TSG (technical support group), especially Benjamin Reaves, for valuable discussions.

References

[1] H. Alshawi. Translation inter-operating protocol. specification by AT&T Labs, March 1999. Copyright AT&T Corp.

[2] H. Alvestrand. Tags for the identification of languages. Standards Track RFC 1766, Network Working Group, March 1995. http://www.imc.org/rfc1766.

[3] K. Simonsen et al. Character sets. IANA standard, February 1999. http://www.isi.edu/in-notes/iana/assignments/cl sets.

[4] M. Handley and V. Jacobson. Sdp:session description protocol. Internet-draft RFC 2327, Internet Engineering Task Force, April 1998. ftp://ftp.isi.edu/in-notes/rfc2327.txt.

[5] M. Handley, H. Schulzrinne, and E. Schooler. Sip:session initiation protocol. Internet-draft RFC 2543, Internet Engineering Task Force, March 1997. ftp://ftp.isi.edu/confctrl/docs/draft-ietf-mmusic-sip-02.ps.

[6] T. Matsuda. *ATR-MATRIX version 2.3.0.* ATR ITL TSG, November 1998.

[7] B. Reaves, A. Nishino, and T. Takezawa. ATR-MATRIX: Implementation of a speech translation system. In *Proc. Acoust. Soc. Jap.*, Spring 1998.

Appendix A Program TIOPConnectController.py

`TIOPConnectController.py` is the communication interface between ATR-MATRIX and other servers or clients using the TIOP protocol. It is written in Python, a object-oriented rapid prototyping language.

A.1 Installation

The TIOP communication controller consists of the following files:

- TIOPConnectController.py
- TIOPreadconnection.py
- TIOPwriteconnection.py
- TIOPconfig

Required libraries are (in addition to standard Python libraries):

- sipPacket.py
- sipPacket2.py
- atrspawn.py
- atrconfig.py
- kconv.so

Matrix expects the file "`SipConnectController`" to be in the directory `resources/matrix/bin.$OS`. The easiest way to use this communication controller is to replace the original CStarII `SipConnectController` in this directory by `TIOPConnectController.py` (which has to be renamed or linked to `SipConnectController`). Alternatively, the `$PATH` environment variable can be set in a way that the (renamed) TIOP connection controller is used.

Also copy or link the files `TIOPreadconnection.py` and `TIOPwriteconnection.py` into the same directory. An option to select between various communication control programes is not provided.

The libraries `sipPacket.py` and `sipPacket2.py` have to be available, they contain classes and methods for handling of packets from the ATR-MATRIX main controller. `kconv.so` is a C library used to convert Shift_JIS encoded Japanese text to EUC coding, which is the text format in ATR-MATRIX. `atrspawn.py` is a socket library from ATRSPREC. `atrconfig.py` is a command line and configuration file parsing utility.

`TIOPconfig` should be installed to the matrix run directory, alternatively the default options in the script can be directly modified.

For ATR-MATRIX version 2.3.1 and lower, `TIOPConnectController.py` revision 1.7 has to be used. Only ATR-MATRIX 2.3.2 and (perhaps) higher can use newer versions of `TIOPConnectController.py`. This is due to changes in the SipPacket format. The older connect type message does not contain host and port information, thus not supporting reconnection after one session ended.

A.2 Behaviour

The program first waits for a connect packet from matrix, then establishes connections by sending SIP messages using the setting as specified in the configuration file and the host and port number as given in the connect packet. Then, the programs `TIOPreadconnection.py` and `TIOPwriteconnection.py` are started using `popen2`. After data sockets have been established, the controller waits for events on four connections:

- a BYE message from the SIP connection. It will end the session, the controller closes all sockets and prepares to reconnect. A disconnect packet is written to `stdout`.
- sip-packets from `stdin`.
  - `exit` type packets kill the controller (and the data connections).
  - `disconnect` packets make the controller close all connections and prepare to reconnect (which means waiting for a connect packet from `stdin`).
  - `trans` packets are searched for translated text. The text is written to the `TIOPwriteconnection.py` child process.

  - All other packets are ignored.
  - Reading an empty string from `stdin` causes a shutdown of the communication controller (and the data connections).

- the `stdout` of the child `TIOPreadconnection.py`. This is data received from the other server, plain text as specified in the configuration, typically Japanese text in EUC format. If an empty string is read, all socket connections are closed, a disconnect packet written to `stdout` and the program prepares to reconnect.
- the `stdout` of the child `TIOPwriteconnection.py`. If an empty string is read, all socket connections are closed, a disconnect packet written to `stdout` and the program prepares to reconnect.

A.3 Configuration

The configuration can be done using a config file or command line parameters.

ATR-MATRIX starts the connection controller with the following command line:

```
SipConnectController -config ./SipConfigFile
```

This is different from the ATRSPREC style command line handling, requiring an "=" between parameter and value. This problem is dealt with by always assuming `$MATRIX_PREFIX/run/TIOPconfig` to be the configuration file. If the file does not exist, a warning is printed and default settings are used.

Table 2 lists the command line options of `TIOPConnectController.py`.

A.4 Troubleshooting

This chapter is a list of possibly occuring problems and how to solve them.

- *The files* `TIOPreadconnection.py` *or* `TIOPwriteconnection.py` *cannot be executed.*
  Probably something went wrong during installation, check the files or links in the directory `$MATRIX_PREFIX/resources/matrix/bin.$OS`, they must exist and be executable. Also check the `$PATH` setting.
- *The controller does not behave like I configured it.*
  Check if the configuration file `$MATRIX_PREFIX/run/TIOPconfig` exists and is readable. If it does, set the configuration option `verbose_mode=on` and look at the additional warnings produced now.
- *The GUI connect window asks for host name and port number as it should, but also for room and site ID. What is this ?*
  These are parameters needed for the Cstar project. The TIOP communication controller will only use the first two. The room and site ID are of no importance.
- *After one session I want to immediately reconect but I can not.*
  This is a socket problem in python, after closing a socket it will linger for 60 seconds. You can either wait 60 seconds or exit the whole ATR-MATRIX system and restart it instead of disconnecting and reconnecting, because in this case there is no time limit. Startup of ATR-MATRIX may take some time, though.
- *I can open clients (like* `TIOPwriteconnection.py`*) and they work, but servers on this side (like* `TIOPreadconnection.py`*) do not.*
  This is a firewall problem, look in Appendix B for legal settings. If this does not help, contact the system administrators (e.g. Yamaguchi-san (`t_yamagu@itl`)) to reconfigure the router with your IP address and port numbers.
- *The system seemed to work normally, but in the middle of the session anything except the GUI hangs up.*
  A possible cause is that you typed something into the window where matrix was started. If something is typed there, or one of the arrow keys pressed, some controller gets confused and hangs up. This is not necessarily a communication controller problem.
- *After telling the connect controller to connect, with this side as server, nothing seems to happen.*
  The probable reason is that the connect controller successfully opened a socket server, but no client is connecting to it (or cannot connect, see firewall problem above). This problem is hard to handle, see section 4.4 for details.

Table 2: Configuration of `TIOPConectController.py`

option	default	description
`-help`	(n.a.)	Display help message.
`-config`	(none)	(not supported, see above).
`-SIP_host`	localhost	Name of the SIP server host. If the setting is localhost or the IP address of the used computer, the communication controller will work as server, otherwise as client. The setting will be overwritten by the contents of a connect type matrix packet.
`-myTIOPport`	20000	Port where the SIP server will be started if the ATR side is the SIP host.
`-myname`	`$USER`	name to be used in the SIP "from" fields, e.g. rgruhn.
`-myhost`	`$HOSTNAME+ '.itl.atr.co.jp'`	IP name of used computer, e.g. atra57.itl.atr.co.jp
`-myfullname`	Rainer Gruhn <sip:rgruhn@itl.atr.co.jp>	Text expected in the SIP header: Name and email-like address of sender.
`-mymedia`	text 20002 TCP/IP plain	Description of data type received at port. Please note that on the 2nd position in this line (which is 20002 here) the port number as set in 'mydataport' will be inserted, the setting in this string will be overwritten.
`-myipaddress`	133.186.35.196	IP address of machine where our connections (both SIP and data) will be made from and data shall be sent to.
`-mydataport`	20002	port where this side will open a data server to read from.
`-myattribute`	fmtp:plain charset=EUC-JP	Type of data expected on this port, in this example plain Japanese text of type EUC-JP. If EUC-JP is replaced by Shift-JIS, incoming text will be converted using the kconv.so library.
`-mylanguage`	jp	Language we want to receive, jp=Japanese, en=English
`-theiripaddress`	133.186.35.196	Where to connect if this side is SIP client. Setting will be overwritten by setting in connect packet. This IP address is also used for data connections if the information was not given in the SIP messages.
`-theirport`	25060	Port where the SIP server is waiting. Setting will be overwritten by setting in connect packet.
`-sipversion`	SIP/2.0	The SIP version.
`-sip_via`	SIP/2.0/TCP	Protocol used for communication.
`-subject`	prototype	Subject for SIP header.
`-networktype`	IN IP4	Network and address type.
`-packetdestination`	SipMainController	Name of main controller in matrix packets.
`-packetsource`	SipConnectController	Name of connection controller in matrix packets.
`-verbose_mode`	on	Print additional messages to `stderr`.

Appendix B Technical Data

B.1 Host IP Addresses and Port Numbers at ATR

For security reasons only a limited set of ports on selected hosts can be used for TIOP communication with partners outside of ATR.

The hosts are:

- DEC Alpha `atra57.itl.atr.co.jp`
- Linux PC `pcx196.itl.atr.co.jp`
- Linux note PC `itlnpc70.itl.atr.co.jp`

Available port numbers on these hosts are:

- 20000 (which is usually used for SIP messages)
- 20001
- 20002
- 20003
- 20004
- 20005
- 20006

B.2 Host IP Addresses and Port Numbers at AT&T

There is only one machine currently available for connection experiments. It is `135.207.20.125`.

Planned for future use are also `135.207.17.46` and `135.207.25.131`

Available port numbers are:

- 5060
- 25060 (which is usually used for SIP messages)
- 25101
- 25102
- 25103
- 25104
- 25105

B.3 Minimal SIP Messages

This excerpt from a log file generated during connection experiments shows that SIP messages can be handled freely, leaving out optional lines does not cause problems. The following INVITE and OK messages were generated by the AT&T communication controller during two connection experiments with the note PC itlnpc72 connected inside the AT&T network. The lines are excerpts from a packet logfile, the actual SIP message is only the text between the quotes.

```
TIOP: 'INVITE sip://twin40:20000/ SIP/2.0'
TIOP: 'to: sip://twin40:20000/'
TIOP: 'from: tribeca'
TIOP: 'content-type: application/sdp'
TIOP: 'cseq: 1 INVITE'
TIOP: 'Content-Length: 85'
TIOP: ''
TIOP: 'm=text 25101 TCP/IP plain'
TIOP: 'c=IN IP4 135.207.20.125'
```

```
TIOP: 'a=fmtp:plain charset=ISO-8859-1'
TIOP: ''
TIOP: ''


TIOP: 'SIP/2.0 200 OK'
TIOP: 'from: localhost'
TIOP: 'content-type: application/sdp'
TIOP: 'cseq: 1 INVITE'
TIOP: 'Content-Length: 85'
TIOP: ''
TIOP: 'm=text 25101 TCP/IP plain'
TIOP: 'c=IN IP4 135.207.20.125'
TIOP: 'a=fmtp:plain charset=ISO-8859-1'
TIOP: ''
TIOP: ''
```