Thai Voice Application Gateway

Dararat Kaitrungrit*, Matthew N. Dailey*, and Chai Wutiwiwatchai†

*Computer Science and Information Management
Asian Institute of Technology
P.O. Box 4, Klong Luang, Pathumthani 12120 THAILAND
Email: u41dkr@hotmail.com, mdailey@ait.ac.th

†Human Language Technology Laboratory
National Electronics and Computer Technology Center (NECTEC)
Thailand Science Park, Klong Luang, Pathumthani 12120, Thailand
Email: chai.wutiwiwatchai@nectec.or.th

***Abstract*—Voice gateways enable the construction of interactive applications that combine the telephone system with speech recognition, speech synthesis, and information systems. A voice gateway is an enabler that has the potential to broaden access to the resources available on the Internet to include users that have no computer, IP network connection, or Web browser. However, commercial voice gateway technology is expensive, and applying speech recognition and text-to-speech technology to local languages is beyond the capabilities of most small and medium-sized enterprises that could benefit from voice-enabled applications. Towards solving this problem, we propose, implement, and evaluate a low-cost Thai-language voice gateway system based on open standards for speech technology and existing open source software projects. Our system supports the VoiceXML markup language for voice dialogs, the MRCP protocol for communication with a speech engine provider, and effectively recognizes and synthesizes Thai speech. The system uses a client/server architecture separated into 3 main modules: the VoiceXML interpreter, the speech engine interface, and the telephone integration system. The current prototype still needs improvement but is functional enough to provide a basis for future enhancement and localization to other languages. We have released the system as open source software for interested developers.**

I. Introduction

With modern speech recognition and text-to-speech technology, the dream of talking naturally to computers is gradually materializing. Voice recognition and synthesis technology is already prevalent in interactive call center systems; to provide a standard for voice-based applications that integrate with the Web, the W3C has released VoiceXML [1]. VoiceXML makes voice interfaces as easy to build, deploy, and use as the browser-based interfaces that currently dominate the Web. By combining telecommunications with the Web, developers can take advantage of Web technologies within telephone-based applications. This means that Web developers can invent applications that can be accessed via many platforms. However, existing VoiceXML development and deployment platforms are either commercial or do not support the Thai language. In this paper, we propose an architecture for an open-source Thai Voice Gateway system, describe a prototype implementation based on customization and integration of existing open source software packages, and evaluate the prototype on a simple example Thai-language enabled voice application. The resulting open source software package, available at http://webeng.cs.ait.ac.th/oslwiki/voicegateway, is usable and ready for further enhancement by developers.

II. System Architecture

Our Thai Voice Gateway integrates speech, telephony, and the Web to create conversation systems. Its client/server architecture allows users to communicate from the telephony system to a web application server and vice versa. It acts as a central hub for communication among components via VoiceXML (a markup language), which consists of XML tags to instruct each component what command to execute and in what order to execute them. Fig. 1 shows the broad architecture of the system. The main components in the gateway involve telephone management, the VoiceXML interpreter, and the speech recognition and synthesis engines. The voice gateway connects to back-end web applications using the HTTP protocol. This system allows an enterprise to develop business logic one time, but allows that logic to be used from a variety of devices such as web browsers, mobile phones, traditional phones, and e-mail clients.

Here we describe the voice gateway's implementation. It integrates three components: a VoiceXML interpreter, a telephony interface, and a speech engine interface. We began with existing open source software, added necessary glue code, and customized each module to support the Thai language.

A. VoiceXML Interpreter Interface

VoiceXML is a standard language released by the W3C in 2001. It is used for the development of voice-based user interfaces over telephones. It is an XML dialect for scripting voice interactions between humans and computers. It allows the developer to develop voice applications in the same way as writing a web application. The basic element of interaction is a spoken dialog. VoiceXML should be used with a speech interface. This interface allows the system to generate prompts using speech synthesis. It also processes spoken user responses

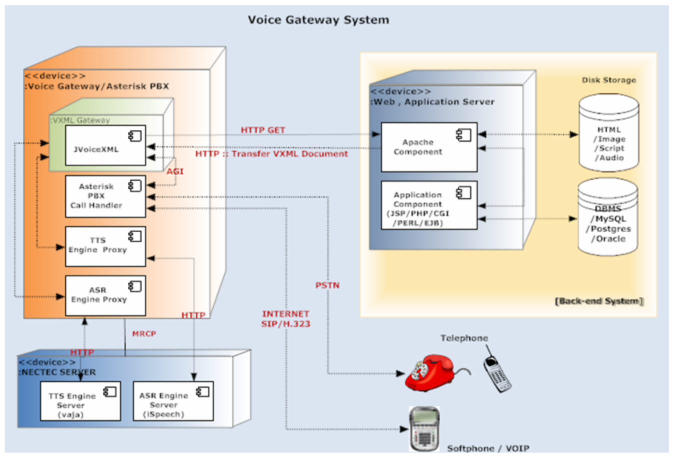

Fig. 1. Broad architectural view of our Thai Voice Gateway system.

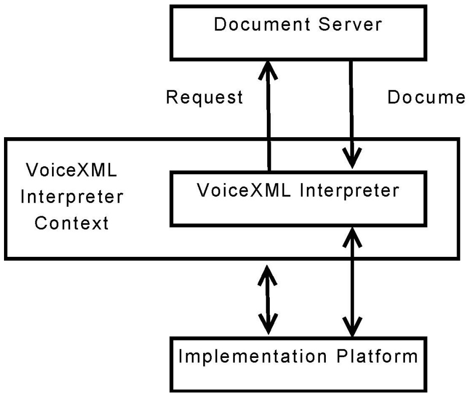

Fig. 2. Architecture of a VoiceXML interpreter server. Adapted from [1].

by using automatic speech recognition (ASR) and grammars defined in the VoiceXML program. Web application developers can output a VoiceXML script from the server side, and the script can then be rendered on a browser that supports the VoiceXML language, or by a voice application gateway. Fig. 2 shows that a VoiceXML application is responsible for processing requests from users by detecting incoming calls, answering those calls, and downloading the appropriate VoiceXML documents from the Internet. The document server replies to the VoiceXML interpreter, which processes the dialogs in the returned VoiceXML document. The VoiceXML interpreter is responsible for controlling the implementation platform to interact with the user through actions such as numeric or spoken input and system events such as recording timeouts and client disconnects.

Our VoiceXML interpreter component is implemented by the open source JVoiceXML library. JVoiceXML is written entirely in Java. The main function of JVoiceXML is to interpret VoiceXML tags and process instructions according to the specified dialog flow by receiving input and returning output through the telephony interface. Our version of the module enhances the original JVoiceXML package by customizing the `Menu` class to support auto-generated grammars, customizing

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar mode="voice" version="1.0" root="root" xml:lang="en">
  <rule id="root" scope="public">
      <one-of>
         <item>วานิลา</item>
         <item>สตอบอรี่</item>
         <item>คุกกี้แอนด์ครีม</item>
         <item>ชาเขียว</item>
      </one-of>
  </rule>
</grammar>
```

Fig. 3. An example XML grammar prompting the caller to speak one of four ice-cream flavors, "vanilla," "strawberry," "cookie and cream" or "green tea," in Thai. For more details on the XML grammar format, refer to [2].

```
<root>          =   "ฉันต้องการดื่ม"  <kindofdrink>
<kindofdrink> =   <drink> "ขนาด" <size>
<size>          =   "เล็ก"   | "กลาง"  |"ใหญ่"
<drink>         =   "น้ำเปล่า" | "แป๊บซี่" | "ชาเขียว"
```

Fig. 4. An example grammar written in ABNF notation, for a drink ordering application. The speech engine accepts user utterances such as "I would like a small Pepsi" in Thai. For more details, refer to [2].

the `Audio` class to stream audio files back to the telephone, and implementing a grammar plugin to support both of the standard grammar specification formats, XML and BNF. With our modifications, JVoiceXML XML and BNF grammar files now support the Thai character set, as shown in Fig. 3 and Fig. 4.

B. Telephony Interface

This module uses the open source Asterisk PBX to handle telephone system interaction. Asterisk not only supports traditional phone equipment, but it also enhances them with additional capabilities. Users use Asterisk by making a call. Asterisk provides internal management capabilities for information such as personal data and business data. One advantage of building on Asterisk is that it supports multimodal interaction. SIP (the Session Initiation Protocol), defined by an IETF standard, comes from the VOIP community. It defines the establishment of multimedia sessions. These sessions are used for real-time data communication sessions such as audio, video, or instant messaging. SIP signaling uses a text-based protocol similar to the HTTP protocol. It is one of the most important protocols built into Asterisk because it supports mobility. The media stream is associated with the user, and not with the specific devices currently being used. A SIP user can register his location from a variety of devices.

To connect Asterisk and JVoiceXML, we use the Asterisk-Java library to communicate with JVoiceXML over TCP. The Asterisk-Java library allows developers to write Java programs to manage an Asterisk PBX through an API including commands to answer, hang up, record audio, or play audio [3].

C. Speech Engine Interface

The last module in our system is the speech processing engine, which provides JVoiceXML with synthesis and recognition services according to a VoiceXML dialog. To support the Thai speech engines provided by NECTEC, we must implement or customize two components: the Thai speech engine interface and the MRCP protocol interface. In the voice gateway, JVoiceXML acts as a client and makes requests to the speech engine server through the MRCP protocol [4]. The MRCP specification is defined by the IETF as a standard protocol for speech interfacing. It uses text to send and receive messages between client and server resources, with additional mechanisms to transmit embedded binary data. It is used to control and communicate with a speech engine by defining the requests, responses, and events to process the stream.

The MRCP open standardized protocol allows VoiceXML platforms to communicate with speech resource engines. It is used to manage media processing resources that provide speech services such as automatic speech recognition (ASR), speech synthesis (TTS), speaker verification, and speaker identification. It enables the VoiceXML platform to be deployed separately from the speech resources and allows a VoiceXML platform to integrate components from multiple speech engine providers. For example, a VoiceXML platform can use ASR and TTS engines from one provider for one set of languages, and use another provider for another set of languages. Speech technology providers can add value to their products without changing the user interface.

We built a MRCP server wrapping NECTEC's Thai speech recognition and synthesis engines using Cairo [5], an open source MRCP server. Here we describe the design and implementation of our MRCP-compliant Thai speech recognition and synthesis resource server.

1) Thai speech recognition resource: To create the Thai speech recognition resource, we implemented the following classes:

- `iSpeechReceiverResource`: This is the main class for the speech recognition server. After `iSpeechReceiverResource` is started, it listens on a port defined in the configuration files and registers its service with the `ResourceServerImpl` class. `ResourceServerImpl` acts as the media resource pool; it waits for requests from the client and sends each request to a responsible resource. `iSpeehReceiverResource` invokes the Thai speech engine class to interpret the input speech, encapsulate the result text in a MRCP message, and send the MRCP response message back to the client. Developers have to implement their own version of this class anytime they want to deploy additional resources on the MRCP server.
- `iSpeechRecEngine`: This class recognizes the input wave form by invoking the iSpeech implementation class. It handles the entire recognition process including getting the result from the Thai speech recognizer. A schematic of the speech recognition process is shown in Figure 5.
- `MrcpRecognitionClient`: This class acts as the client making recognition requests to the MRCP server. One instantiation of this class supports one recognizer session. It has to send a grammar file and write the speech wave form into the voice channel.
- `NativeMediaClient`: This utility class is used for sending or receiving media streams in each session. We implemented this class using the Java Media Framework (JMF). It can read a media file, play a received audio stream, and sink a stream to a sound file.

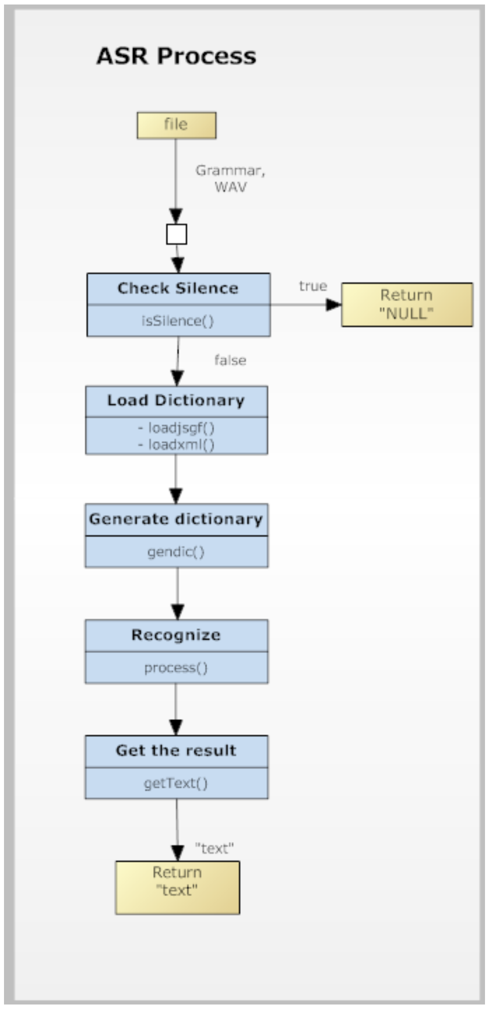

Fig. 5. Process diagram for speech recognition. On the server we need two input parameters: wav file and a grammar file. The server process validates the input files, loads the dictionary, and generates the result text.

2) Thai speech synthesis resource: To create the Thai speech synthesis resource, we implemented the following classes:

- `TransmitterResource`: This is the main class for the speech synthesis server. We created the `TransmitterResource` class in the same manner as `iSpeechReceiverResource`. It invokes the `MrcpSpeechSynthChannel` class to synthesize the

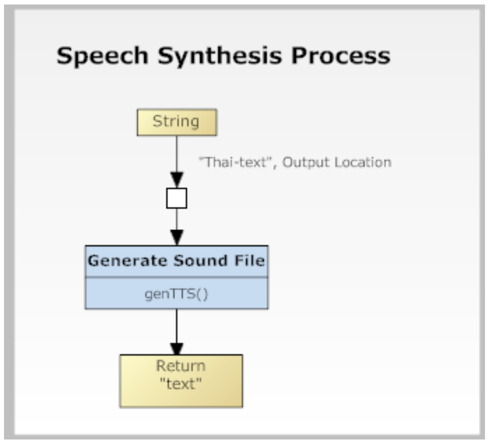

Fig. 6. Process diagram for speech synthesis or text-to-speech (TTS). The TTS engine converts normal input text into a speech file at a specific location.

speech when requested by a client.

- `MrcpSpeechSynthChannel`: This class synthesizes speech and sends an audio stream back to a session media channel. The speech synthesis process is shown in Figure 6.
- `MrcpSpeechSynthClient`: This class acts as the client sending speech synthesis requests to the MRCP server. The role of MrcpSpeechSynthClient is to send input text, handle the MRCP response event, and convert the synthesized speech to an audio file.

III. Evaluation

To evaluate the proposed architecture, design, and prototype implementation, we developed a small voice application. The application runs under the voice gateway with the integrated Thai speech engine provided by NECTEC. As an experiment, we provided 60 short sentences consisting of 7 or 8 continuous words, and stored them in the dictionary. Approximately 70% were correctly recognized over 20 trials. The performance of the system is limited in terms of speed, due to time consumed in the recognition process. Currently, the time for recording is fixed at 10 seconds for every recording. After speech is recorded as a WAV file, there is another delay for transferring the WAV file from the speech engine server, to the JVoiceXML interpreter server, then to the Asterisk server. In our experiments, approximately 20 seconds on average is spent for this transfer. Therefore, in practice, approximately 30 seconds are required to complete one round of recognition. The main reason for this latency is that the speech engine tools do not support stream input/output. Thus, a large amount of time is required for storing then transferring WAV files to Asterisk. We plan to improve on the end-to-end media streaming capabilities of the system in future work.

IV. Conclusion

We propose a standard architecture for an open source voice gateway and provide technical details on how to integrate Thai speech tools from NECTEC with open source modules available on the Internet. Our system fully supports VoiceXML, speech standards, and telephony. We have developed a simple voice application using the platform to show that the voice gateway provides a practical means to create voice-accessible Thai-language services with today's technology. The Thai speech engines are usable and intelligible. Although the system still needs improvement, it is functional enough to provide a basis for future enhancement. The system is available as open source software at http://webeng.cs.ait.ac.th/~oslwiki/voicegateway.

Acknowledgments

We thank Sumanta Guha for helpful comments on this work.

References

[1] D. Burke *et al.*, "Voice extensible markup language (VoiceXML) 2.1," W3C Recommendation, 2007. [Online]. Available: http://www.w3.org/TR/2007/REC-voicexml21-20070619/

[2] M. Brown *et al.*, "Speech recognition grammar specification version 1.0," W3C Recommendation, 2004. [Online]. Available: http://www.w3.org/TR/speech-grammar

[3] Digium, Inc., "Asterisk: The open source technology platform," 2008. [Online]. Available: http://www.asterisk.org

[4] S. Shanmugham, P. Monaco, and B. Eberman, "A media resource control protocol (MRCP) developed by Cisco, Nuance, and Speechworks," IETF Request for Comments RFC 4463, April 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4463.txt

[5] SpeechForge, "Cairo MRCP-compliant speech server," 2007. [Online]. Available: http://www.speechforge.org