A network architecture for building applications that use speech recognition and/or synthesis

Dominique Vaufreydaz, José Rouillard, Mohamad Akbar

A network architecture for building applications that use speech recognition and/or synthesis

Dominique Vaufreydaz, José Rouillard, Mohammad Akbar

Laboratoire CLIPS-IMAG, équipe GEOD
Université Joseph Fourier
Campus Scientifique, BP 53 38041 Grenoble Cedex 9 - France
Tél.: 33 + (0)4.76.63.45.26 or 33+ (0)4.76.63.56.51 - Fax: 33+ (0)4.76.63.55.52
E-mail: {Dominique.Vaufreydaz, Jose.Rouillard, Mohammad.Akbar}@imag.fr

ABSTRACT

This paper proposes a simple versatile architecture for sharing speech recognition and synthesis resources in a heterogeneous networked environment. Using a unified application programming interface and the concept of proxies allows many clients to gain access to services that would normally be reserved to specific computers. This helps reducing the total cost of ownership while providing sophisticated service through simple approaches. At the same time, multi-speaker data collecting will be simplified. We show that using this architecture for speech activated software reduces the requirements of client computers while adding a negligible latency compared to a powerful workstation.

INTRODUCTION

Nowadays, a lot of people access the Internet by the way of a proxy. The idea is to provide a speed increase for retrieving documents on the Web by using a cached version of the document on a nearby server. In fact, every data processed by the proxy server (i.e. every Web document cached locally) for one user can be useful to others. Reusing data or documents can be adapted to a new generation of servers: voice-activated servers. These services can share the common data among many clients. At the same time they may be used to gather useful data from clients to increase and adapt the system to the different environmental conditions.

In synthesis task, the proxy can make a ranking to keep signals that will possibly be requested later. It uses a cache in order to answer more efficiently to similar requests. This caching method is interesting in case of applications that often produce the same outputs, we will see an example of such a system later on this paper in section III.2.

The second way to use a voice-activated proxy is the most interesting one. The aim is to improve the performance of a speech recognition engine using the signals collected from clients during recognition sessions. To do that, the server keeps signals from clients and in a later manual or automatic transcription and/or labelling phase, it recalculates acoustic and language models to increase robustness to adverse conditions made by multiple microphones, environmental noise and multiple speakers.

Another advantage of this architecture is the ability to use lightweight clients without scarifying total system performance and at the same time proposing a shared service to the users.

Many commercial voice-activated products, like dictation or navigation on the Web, propose a single user package that takes a lot of resources (disk space, memory and CPU time) on the user's computer. On the other hand they do not offer a real time service most of the time. So, it is easily seen that lightweight computers can not make use of such systems because of their lack of sufficient resources.

Moreover, many laboratories or companies are nowadays equipped with a local and/or an Internet network. In the same time new technologies, like ATM or Ethernet Gigabytes, are being developed in order to allow an important increase in network performances. Finally, the new IP protocol (version 6) will include new Quality of Services (QoS) levels for multimedia data such as speech or audio data [1]. All these points lead us to assume that network data exchange is and will be increasingly fast so that transferring audio data will not be a major problem in a networked environment.

In addition to all positive aspects of a client server architecture, we consider that a reduction in the TCO (Total Cost of Ownership) may be done by transferring heavy calculations from clients to one or more dedicated strong computers. In such an environment, the users' powerful workstations, that would be needed to propose recognition and/or synthesis service, can be replaced by lightweight network computers with just the needed performance.

I. SYSTEM ARCHITECTURE

The client server architecture of the system is designed like a proxy. There may be one or more servers and several clients at the same time. Servers are dedicated powerful computers capable of answering many requests at a time. Based on their configuration they provide specialised services (i.e. task based recognition). In the next section, we will explain the current implementation of this architecture and we

discuss different choices we had made. The Fig. 1 hereafter shows the global network architecture.

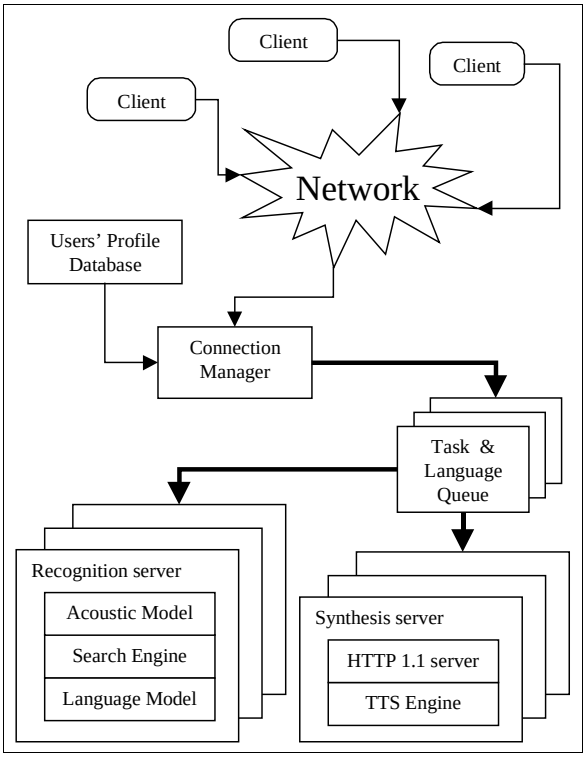

Fig. 1 : Global system architecture.

Each client may perform requests to the system in order to make text to speech synthesis or speech recognition on a client side recorded utterance. Clients may have their personal profile and customisations. To make a request the client connects to the connection manager which decides, based on client settings and servers' load, to redirect the request to an appropriate server. The settings can be the language in which the user interacts with the system, a set of special words he/she uses in every day work (special task oriented vocabulary), the gender in which the synthesis will be done and finally a personal recognition profile that may help increasing recognition performance. To reduce the network traffic, especially in case where connection manager is not running on the same computer as the server which responds the request, and based on the request type made by every client, the connection manger places the request into a queue managed by the selected recognition or synthesis server. Once the server is ready to respond, there will be a direct connection between client and server that will help also to save the data transit time that would be otherwise needed.

II. IMPLEMENTATION

According to our available tools and facilities, we had to do some choices for our implementation. The recognition server is based on our current French recognition module and the synthesis server is based on a commercial Text To Speech engine. However, thanks to the system architecture and the defined connection transmission protocols, any of these parts can be replaced by other available products. To support each new engine one only need to implement a simple broker that implements our protocol in term of that specific engine application programming interface (API).

II.1. Recognition Server

It is already shown [4] that the size of corpus is a primary factor in robustness of a speech recognition system. Recognition Proxy helps gathering either acoustic or linguistic data more easily and at the same time taking into account environment variability. The newly input data may be used to extend language model and phonetic variants coverage and also to retrain the acoustic models. This leads gradually to a better recognition performance. On the other hand using a powerful server, the recognition latency is reduced to an acceptable delay for the users who are working on lightweight computers.

Our recognition proxy is including in global architecture as shown in the Fig. 1. The recognition engine is based on RAPHAEL module [2] and currently supports French language. The connection manager is used to allow multiple connections to the same server or multiple servers at a time. As it is already explained it manages the queuing and routing of the recognition requests to the appropriate server and assures the recognition hypothesis routing back to the client. To speed up the recognition process clients can send the audio signal gradually to the server. This allows server to create partial hypothesis while it receives the signal. If it's needed, the clients can ask for reception of partial hypothesis and/or complete hypothesis based on different search methods. Actually the search parameters are controlled uniquely on the server but it is eventually possible to customise the recognition engine based on a user profile. This means that clients can eventually choose their language, their vocabulary and their search parameters amongst existing models and parameters.

The implementation of RAPHAEL uses Janus-III search engine [5]. The acoustic corpus contains 12 hours of continuous speech of 72 speakers extracted from BREF-80 corpus [6]. The vocabulary contains nearly 5500 phonetic variants of 2900 distinct words. The language model is trained over a textual corpus of 115 millions words. Using a class based language model helps to easily add new proper names to the vocabulary without retraining the entire language model.

II.2. Synthesis Server

This service is closer to the Internet proxy idea than the recognition one. The server makes a ranking to keep signals that possibly will be requested. It uses a cache in order to answer more efficiently to others requests. Today, it's not too costly for the server to keep many pre-synthesise signals. In the HALPIN system (we will explain it in III.2.), most of synthesis files don't exceed 100 Ko. A 9 Go disk drive can store more than 80000 of these files. This idea is interesting in case of applications

that periodically produce the same outputs. But the main goal of this server is to confer synthesis abilities to clients, even when they are working on a lightweight network computer. Another interesting point is that the technology behind the engine can be safeguarded and only the final results are transmitted. This is especially interesting for those who are providing public voice-activated web services.

The synthesis server uses the Elan TTS engine [3]. This engine enables several languages (French, English, Spanish…) and two voice types (a male and a female). In the current version, our server is monolingual (French male voice). Due to the restrictions inherent to the Elan TTS any change to these parameters needs human intervention. However, the server offers configuration facilities to some parameters by diacritics incorporated into requests: the pitch, the speed and pauses. Currently, 3 data types are supported: .wav (Microsoft audio format), .au (Sun audio file) and .raw (raw binary sample stream).

For its network interface, the synthesis server uses the HTTP 1.1 protocol. A client connects the server, sends a request and waits for the response. Because connecting to a server is expensive, the HTTP 1.1 protocol allows keeping the connection alive between each request. Any application that often uses this service can gain time enhancement by using this feature. Finally, like the recognition server, several synthesis servers can be used at a time and the connection manager can dispatch requests from clients to the best server according to their configuration, load, and network proximity.

III. CASES STUDY

In order to validate our idea we perform two kinds of test. First of all, we check whether we would be able to perform normal recognition and synthesis tasks on lightweight clients without adding a lot of network latency (the tasks which would take a lot more time on standalone lightweight client). Secondly, we performed and tested the validity of the proposed architecture in a real situation using HALPIN system.

III.1. Performance Tests

This set of tests is uses to measure the usability of our architecture. They don't reflect a real application but permit to validate our assumptions. The two servers don't use any kind of optimisation. The recognition server doesn't allow the connection keep-alive mode and the synthesis one doesn't cache any file. In these first sets of tests, we wanted to verify whether the following transformation is usable:

$$SingleComputerTime \Downarrow LocalComputerTime+NetworkTime+ServerTime$$

The aim is to verify whether the latency added by using our architecture, is reasonable and acceptable compared to the quality of the provided services. Indeed, this kind of server can be deployed over the network only if they can resolve problems without creating new ones, especially for powerless computers.

The test platform consists in three computers: a Pentium 90 MHz with 24 Mo RAM (lightweight client), a Pentium 166 MHz with 128 Mo RAM (synthesis server) and a Pentium II 300 MHz with 384 Mo RAM (recognition server). They are residing on a 10 Mb/s LAN of about 20 other computers.

III.1.1. *Recognition*

In this test, the task is to recognise a set of pre-recorded utterances to assure the same amount of computing power will be used to recognise the utterances. The test corpus contains 20 sentences from 3 to 14 words, all in the vocabulary. The utterance durations are between 0.928 s and 3.856 s.

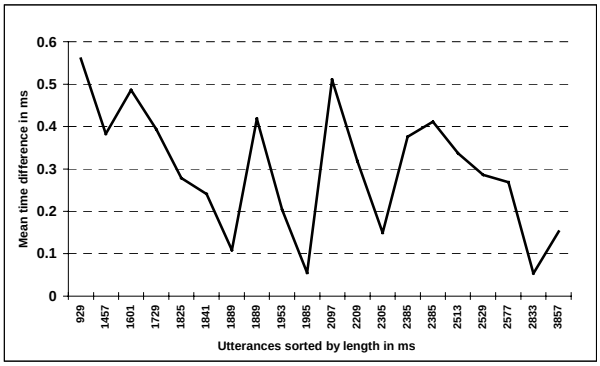

Fig. 2 : Mean time difference (in ms) between local and network modes in recognition task on 20 phrases.

Our clients are not at all able to do the recognition task so we calculate the time difference between a direct recognition on the server and the same recognition requested by client. Every sentence has been presented 20 times to the recognition engine, first locally and then with the lightweight client. Fig. 2 represents the mean time difference that is almost independent of the utterance length and depends only on the network latency. We can see that using this architecture enables an even lightweight client to provide a sophisticated recognition service with a reasonable delay.

III.1.2. *Synthesis*

In this test, the computer generates 15 utterances, which enumerate numbers. The first one is composed of 6 words, the second of 7, and so on. Due to Elan TTS inherent restrictions, the synthesis time is nearly equal to the pronunciation time. So, we can only verify the impact of network latency on this task. We repeated the task 100 times locally on the server and 100 times using a lightweight client. Then, we computed an average synthesis time for each utterance in each mode. We calculated the percentage of latency added in network mode compared to the local time. Fig. 3 shows the results of this experimentation.

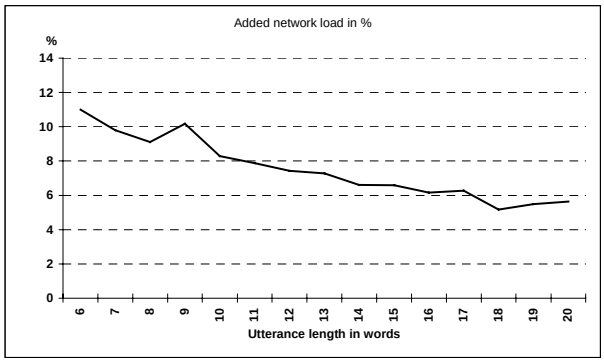

Fig. 3 : Added network load for synthesis task.

As we can see in this figure, the network impact on the synthesis task becomes less and less important with the increase of utterance length. Indeed the time spent in synthesis of one second of speech is bigger than the time needed to transfer it over the network. This is because our synthesiser cannot produce speech data sufficiently rapidly. The overload added by all network transitions is rarely more than 10%. It is evident that a 10% increase in response time of the server is negligible compared to the time needed for a lightweight computer to do the same task locally.

Based on these observations we can deduce that on a cache enabled proxy system we will observe a gain of performance despite network overload.

III.2. The HALPIN system

In this section we explain a real implementation of our architecture in a voice-activated application. The application is called HALPIN [7] and is a part of the ORION project [8]. The idea is to integrate existing speech technologies to a Web based multimodal system of navigation and information research. HALPIN was developed to implement a multimodal conversational model for information retrieval. The dialogue-oriented interface used in the system allows the access to a digital library (83297 documents available), on the Internet, in a natural language way, and gives its oral and hypertext responses via usual browsers (supporting Java Applets). This kind of multimodal natural language interaction, is a valid answer to the problems like confusion, cognitive overload, and evaluation of the answer's relevancy. It has been shown that is it possible to send a real time calculated speech synthesis through the Internet.

This system uses the speech synthesis server, previously described in this paper. The Java Applet makes an HTTP request in order to retrieve a sound document. This call ensures the creation of the synthesised speech, in case it is not already cached, and the network transfer from server to the user's browser, which will produce audio output of the given sentence. Many clients may use the synthesis server synchronously.

Concerning the speech recognition, the first method considered in the HALPIN system was to use the IBM ViaVoice engine on each client. That's the current version proposed on the Web that imposes installation of speech recognition engine on each client. The second solution is integrating our proxy architecture that shares a voice recognition server amongst many users. We are currently testing and evaluating the second approach to help users without a local recognition system to use the application on a computer with only a sound card and microphone installed.

CONCLUSION

In this paper we proposed a simple but very powerful scheme to provide networked speech oriented services on the Internet and Intranet. Using this architecture many simultaneous requests can be done by simple clients not having sufficient computing power to provide local speech services.

We have shown that the network overhead of the architecture is negligible compared to the needed time to provide the real service. The proposed architecture helps sharing of available resources and also provides a mean to collect speech data in order to retrain speech recognition engines.

The defined networked API provides needed speech services to every application that wishes to integrate and propose speech oriented functions. As a concrete example we mention our current contribution to the CSTAR-II [9] project in which we are integrating a multi-platform solution for French speech recognition and synthesis.

REFERENCES

[1] HUITEMA C., *IPv6: The New Internet Protocol*, Prentice Hall, Englewood Cliffs, New Jersey -1996.

[2] AKBAR M., CAELEN J., *Parole et traduction automatique: le module de reconnaissance RAPHAEL*, COLLING-ACL'98, pp. 36-40, Montreal (Quebec), August 1998.

[3] See ELAN Web site at http://www.elan.fr/

[4] JUNKA J.C., HATON J.P., *Robustness in automatic speech recognition*, Kluwer Academic Publisher, 1995.

[5] SCHULTZ T., WESTPHAL M., WAIBEL A., *The GlobalPhone Project: Multilingual LVCSR with JANUS-3*, Multilingual Information Retrieval Dialogs: 2nd SQEL Workshop, pp 20--27, Plzen, Czech Republic, April 1997.

[6] LAMEL L.F., GAUVAIN J.L., ESKENAZI M., *BREF, a Large Vocabulary Spoken Corpus For French*, EuroSpeech'91.

[7] ROUILLARD, J., CAELEN, J., *HALPIN: a multimodal and conversational system for information seeking on the World Wide Web*, ESCA ETRW workshop: Accessing information in spoken audio, Cambridge (UK), April 99.

[8] For more details see the ORION description at http://www.gate.cnrs.fr/~zeiliger/Orion99.doc

[9] BOITET Ch., *GETA's MT methodology and a blueprint for its adaptation to speech translation within C-STARII*, ATR International Workshop on Speech Translation, Kyoto, Japan.