Voice2Web: Architecture for Managing Voice-Application Access to Web Resources

Jan Rudinsky,[1] Tomas Mikula,[1,2] Lukas Kencl,[1] Jakub Dolezal,[1] Xavier Garcia[1,3]

[1] R&D Centre for Mobile Applications (RDC), Czech Technical University in Prague
Technicka 2, 166 27 Prague 6, Czech Republic
{rudinsj,kencl,dolezj8}@fel.cvut.cz
[2] Faculty of Mathematics and Physics, Charles University in Prague
Ke Karlovu 3, 121 16 Prague 2, Czech Republic
tomas.mikula@gmail.com
[3] Universitat Politecnica de Catalunya (UPC)
Jordi Girona, 31, 08034 Barcelona, Spain
xavi.garci@gmail.com

Abstract. Advances in voice-recognition platforms have led to new possibilities in deploying automated voice-interactive engines for Web content. We present Voice2Web, an architecture allowing to manage access to the resources of the World Wide Web using voice interaction. It rests on the VoiceXML standard and enables rapid composition of dynamic services querying the Web resources. We demonstrate its use on practical examples, discuss architecture implications and invite further platform experimentation.

1 Introduction

The rapidly advancing technologies of the Internet and the World Wide Web (WWW) have become indispensable for functioning of the developed and, increasingly too, the developing world. Yet barriers to access still exist — technological, financial, cultural and physical — for large portions of the world population. The mobile voice services technology has grown to even higher market penetration, and currently outnumbers Internet penetration in an approximately 2:1 ratio worldwide and a 5:2 ratio in the developing countries [1]. Potential voice interfaces to the WWW thus far outnumber the visual ones.

More natural multi-modal interfaces to WWW have many advantages: information provided orally occupies only a part of the brain, leaving remaining capacity and senses free for other tasks (such as visual input or body movement: sports, driving, etc.); people with visual impairment or other handicap would benefit from voice-based Web access; literacy constraints to Web access in the developing countries can be overcome; and better customer interaction would enhance Internet commerce sales and inspire novel automated voice services.

Speech is a natural form of communication for humans. The technological challenge is to manage a better interface for voice access to complex systems such as WWW. In this work we build on decades of research in automated speech recognition (ASR) and text-to-speech synthesis (TTS) [2] and the Voice Extensible Markup Language (VoiceXML) [3, 4] and focus on the network service management architecture.

VoiceXML is a language for creating voice interfaces that use ASR and TTS. It is developing into a vital open standard, enabling rapid proliferation of new voice applications and services. Support by the VoiceXML Forum [5] and the key industrial players accelerates the adoption.

We focus on the problem of *designing and implementing an architecture for managing access of voice-interactive applications to the Web content via both the traditional and next-generation voice communication networks.* The expected architecture attributes are to be fast, manageable, modular, scalable and reliable and allow rapid prototyping of novel services. The logic of the user-interaction is driven by the natural logic of voice conversation, with the Web pages only providing the content (in contrast to the interaction being driven by the Web page structure).
The main contributions of this work are:

– a proposed novel modular *architecture* for building voice applications that use *voice-oriented logic* and *dynamically* access the *content of the World Wide Web*;
– working examples of such functionality;
– practical *experiments validating architecture feasibility*; and
– design of a novel *open VoiceXML Integrated Development Environment (IDE)*, allowing easy creation, sharing and replication of dynamic, WWW-interfacing voice applications. The IDE is Web-based, open to a world-wide developer community at [31] and provides instant setup of telephone and VoIP access to the voice applications.

The above proposed Voice2Web architecture thus represents a step towards a complete voice-services layer, functioning on top of WWW content.

The article is organized as follows: in Section 2 we discuss the related work, Section 3 describes the proposed architecture and Section 4 outlines voice-application dynamic Web access, including a real example. Section 5 describes the IDE, Section 6 presents practical results on experiments validating the architecture and Section 7 holds some concluding remarks and future outlook.

2 Related Work

Recent works develop the idea of the World Wide Telecom Web (WWTW) [10, 9, 12], a voice-driven ecosystem parallel to the existing WWW. It consists of interconnected *Voice Sites*, voice-driven applications created by users and hosted in the network [10], a *Voice Browser* providing access to the many voice sites [9] and the Hyperspeech Transfer Protocol (HSTP) [12] which allows for seamless interconnection of voice applications. Developing regions with large proliferation of phones but little Internet literacy are set to benefit. While WWTW supposedly exists in parallel to WWW, the authors envisage interconnection and interaction of the two systems, but do not (yet) offer architectural solutions of doing so.

Similarly, SpeechWeb [20] is composed of a collection of hyperlinked applications, accessed remotely by speech browsers running on end-user devices. Spoken commands activate the links, using a combination of markup languages. The related MySpeechWeb environment [21] enables development and web deployment of speech applications including the question/answer type applications, created by web forms. The process is completely based on the web-browser, with the constraint to the Opera 9.27 browser with the voice feature installed. In comparison, Voice2Web environment is accessible to end-users by a plethora of voice devices.

The concept of a Voice Portal was suggested in [11], but offering few suggestions as to the architecture. The authors present a system for operation with existing services (email reading, phone calling) and Internet interaction is correctly identified as having tremendous potential. Voice support (i.e. a VoiceXML server) may also be integrated directly into the Web server [8]. This offers greater control over the voice application, but restricts to only one content provider (Web server). In contrast, Voice2Web allows to create a wide range of services, without any Web alteration and accessing an arbitrary number of servers.

A similar concept of voice access to the Web content is represented by the design and implementation of an audio-wiki application [22], accessible via the Public Switched Telephone Network (PSTN) and the Internet. Based on VoiceXML and other W3C standards the system provides voice interaction with wiki web applications. In contrast, Voice2Web is focused on any-web access and thus broadens the target area.

The idea of web-driven *Voice Browsing* is to convert original Web content into VoiceXML dialogues, using VoiceXML templates and extraction rules written in XSLT. The work presented in [7] identified typical HTML patterns and designed a way to browse them using voice. Although similar to Voice2Web, it is based on the opposite logic of building a voice application around a Web page design.

Mobile Web browsing has been shown to be less convenient than desktop browsing [13], in particular Web page navigation and content location. Augmenting the interaction with voice may improve it. Conversely, pure voice-response systems have been shown to benefit from augmenting with a visual interface [14]. This motivates adding more modalities into the user-Web interaction.

Other research has focused on Web browsing by voice and its applicability for the handicapped or elderly. The HearSay audio Web browser [17, 18] allows to automatically create voice applications from web documents using VoiceXML and domain specific ontologies and templates. Recently a prototype of a telephony service for web-browsing via phone, TeleWeb [19], has been designed to combine the phone interface with intelligent browsing features (context-directed browsing, template-detection, macro-replaying) of the HearSay web browser. To improve the Web accessibility for visually impaired without the need to alter the original Web content, the concept of external metadata repository has been developed and shared among research institutions [23].

An architecture of Voice Web Pages implemented by .NET [24] offers the possibility of browsing web sites and playing related streaming media simultaneously. The concept however does not include voice-controlled browsing, as speech recognition is not implemented.

Others investigate utilizing voice for controlling the conventional Web browsers [15], or presenting a typical Web page using voice [16], reporting poor results, with voice-control often being much less productive or convenient. Contrary to Voice2Web, these applications are not initially designed for voice control and thus the results are suboptimal.

3 Voice2Web Architecture

3.1 Architecture Alternatives Discussion

Various alternatives exist for the architecture of voice access to WWW content. The architecture of a *thin, streaming client and strong server* performs all voice processing and executes application logic at the server and maintains a voice connection open throughout the conversation. Server-based solution allows for greater manageability and reliability (pending a network connection) with fast voice-processing and response times. It allows to rapidly introduce novel services and use all standard voice communication protocols, putting less requirements on end-users. The architecture may be easily scaled by adding more hardware resources and load-balancing on the server side.

A *thick client* may perform all of the ASR and TTS on the client and only send text messages over the network, or possibly the entire application may operate locally. While inherently scalable in terms of number of users, such architecture is significantly less manageable (requiring pre-installations and client application updates). Speech recognition and synthesis are strongly memory- and compute-intensive processes, with inadequate resources available in current mobile devices, thus affecting both their performance and power consumption.

While both architectures have their advantages, manageability, performance and scalability seems well addressed by the *thin, streaming client* and a *dedicated strong ASR, TTS and application-logic server.*

3.2 Voice2Web Architecture and Components

As a result of the above discussion, our proposed modular server-based Voice2Web architecture consists of the following components (see Fig. 1):

– Client - a mobile or fixed terminal for end-user interaction
– Telephony-system frontend for communication-channel unification
– Speech-recognition and synthesis engines
– Call-processing server (Voice Browser)
– Voice-application repository and development environment
– Web-interaction Proxy

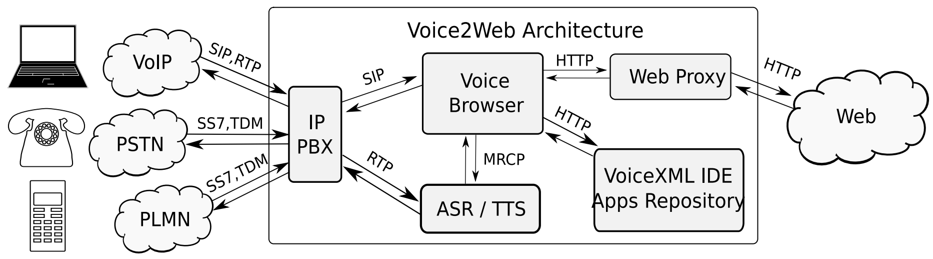


Fig. 1. Voice2Web architecture and function. A caller from the VoIP, PSTN or PLMN network is authenticated and a connection is unified by the IP PBX. Unified internal VoIP connection (SIP and RTP) is processed by Voice Browser according to VoiceXML retrieved from VoiceXML IDE and with help of ASR and TTS engines. To enable effective WWW access, the Web Proxy intermediates the connection to the Web.

The components are mutually independent as they are interlinked by standardized interfaces. Any component may appear multiple times ensuring system modularity, scalability and higher overall system reliability. Majority of the interface protocols are text based allowing easy monitoring and simplified system management.

The architecture should enable caller access by using any type of telephony network including traditional Public Switched Telephone Network (PSTN) and Public Land Mobile Network (PLMN) as well as Voice over IP (VoIP). A *multi-interface frontend telephony system* needs to be integrated, to unify inbound communication channels into a single internal channel. IP Private Branch eXchange (IP-PBX) is used for a small scale project while high-performance load-balanced servers should be used in larger scale networks. In our case Asterisk PBX [26] handles many analog and digital switched network signaling types and VoIP signaling and media protocols. All inbound protocols are converted into Session Initiation Protocol (SIP) and Real-time Transfer Protocol (RTP) internal protocol set. The PBX also performs user account and call management.

A server-side *call-processing* architecture is based on a voice application logic (dialogs). Voice Browser manages the dialogs with help of speech-recognition and speech-synthesis engines. The browser processes incoming calls by a set of predefined rules for call filtering, connection management and for linking traffic to a desired voice application. An increasingly popular W3C standard Call Control eXtensible Markup Language (CCXML) [30] is used to represent this ruleset. The dialogs, including user-machine communication and actions to be performed upon user's response, should be encoded in a human-readable form for easy development and should be platform-independent. This is met by using the W3C VoiceXML standard [4].

Voice Browser scalability may be achieved by DNS load-balancing [28], or a higher level of scalability may be achieved by implementing the Voice Browser as a service on top of multiple application servers. IBM WebSphere Voice Server [27] is an example of such distributed platfrom.

The *ASR and TTS engines* for speech recognition and synthesis should interoperate with the Voice Browser over a unified and open interface to ensure scalability. Media Resource Control Protocol (MRCP) [29] performs this function well. It has become the defacto standard for media resource management by a Voice Browser. MRCP conveys speech recognition results and synthesis progress to the Voice Browser, while input and output audio streams (setup by SIP) are exchanged directly between the end-client and the ASR/TTS engines via RTP.

The speech recognition system should recognize voice samples regardless of the speaker behavioral patterns, thus should not require any voice training. A presented solution is a speech recognition grammar based system which however brings a limitation in number of speech patterns included in the grammar.

A *repository of voice-application logic* can either be placed locally on Voice Browser or stored on a HTTP server. In our case the Web-based development environment VoiceXML IDE (see Section 5) serves as the source of voice applications, acting as an HTTP server.

Finally, voice-application access to the WWW is performed by a *Web Proxy*, which introduces benefits over direct Web access from VoiceXML (see Section 4).

4 Dynamic Voice-to-Web Access

4.1 Dynamic Data instead of Dynamic VoiceXML

Web data are dynamic—may change over time. Prior to VoiceXML version 2.1, there were no means of integrating Web data directly. There was, however, the `<submit>` element, which "*is used to submit information to the origin Web server and then transition to the document sent back in the response.*"[3] This document is commonly referred to as dynamic VoiceXML, due to its dynamic generation by the Web server. Dynamic VoiceXML has a few drawbacks: mixes application logic with data; unnaturally splits the application logic into two (or more) VoiceXML documents; must not be cached by the Voice Browser. This method, used e.g. in [7, 8], can now be considered legacy.

VoiceXML 2.1 introduces the `<data>` element, which "*allows a VoiceXML application to fetch arbitrary XML data from a document server without transitioning to a new VoiceXML document.*"[4] Our applications use this novel approach to Web access, which avoids the above drawbacks.

4.2 Web Proxy

Web Proxy (WP) is an extra layer between VoiceXML and the Web (see Fig. 2). It acts as an HTTP server for VoiceXML applications and as an HTTP client for the Web. Although VoiceXML application can query the Web source directly, it is advantageous to use WP in cases when the application can benefit from its features: *substitutability* of Web sources, *preprocessing* and *caching* of Web documents. A typical example are information-providing services.

Substitutability of Web sources means that a voice application can transparently use any of configured websites as the source of data, while sending just

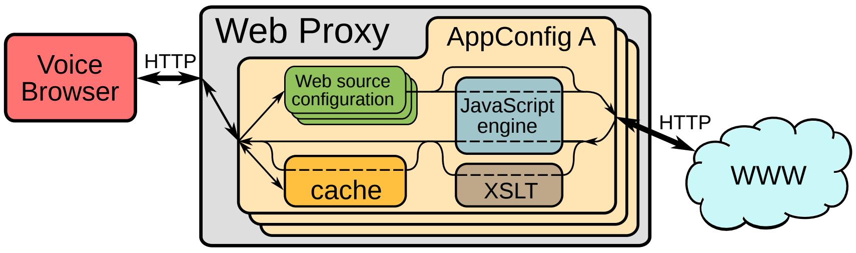


Fig. 2. Web Proxy schema. Upon the Voice Browser request, the Web Proxy decides which application configuration to use (here AppConfig A) based on the URL of the requested document. Then, it either serves the document from the cache, or requests it from a remote Web source. The request is formed based on the Web-source configuration, optionally using a JavaScript function. The response is processed by XSLT or a JavaScript function, optionally stored in cache, and returned to the Voice Browser.

one unified request to WP. The transformation of the request to WP into the request to the Web source is defined in the configuration of each Web source. Single-parametric requests can be transformed by a table that maps this parameter to a URI. More complex requests can be transformed by a JavaScript function. Currently, the Web sources are queried in the order defined in application configuration. If one fails, the next one is tried. An improvement could be to periodically reorder the sources based on their evaluated response time.

By preprocessing, uniform format of Web data is achieved, regardless of the Web source used. Furthermore, preprocessing typically results in a considerably smaller document, thus saving Voice Browser's processing time. The rules for preprocessing are specified for each Web source, either by XSLT or a JavaScript function. Using JavaScript for preprocessing also extends the domain of possible content source formats from just XML (required by both the `<data>` element and XSLT) to any reasonably structured text document.

The response of each request to WP can be cached for a specified *expiration time*, defined for each application. This saves requests to the Internet and processing time (cached documents are already preprocessed). Our implementation assumes that the cache can store documents for all possible requests to each Web Proxy application. This assumption is not unrealistic, as current voice applications can typically issue only a limited number of different requests. To support applications that can generate a large number of distinct requests, employment of a cache replacement algorithm (such as LRU) would be necessary.

Clearly, when the document is not cached, WP introduces some overhead. Disregarding other benefits of WP, let's calculate under what circumstances the average response time of WP is shorter than that of the Web. Expiration time (T_e) is application specific. The average response times of the Web (T_w) and WP for a cached (T_c) and uncached (T_u) document can be measured experimentally. We can safely assume $T_c < T_w < T_u$. We further assume exponential probabilistic distribution of intervals between requests. It can be shown that the expected response times of Web and WP are equal when the average interval between requests is $T_e(T_w - T_c)/(T_u - T_w)$. Thus, a shorter interval means that using the WP pays off.

4.3 Examples of VoiceXML applications

The *Weather* application provides information about the weather around the world. It prompts the caller to choose a location and replies with the current weather conditions. Such application was proposed in [3]. It used the `<submit>` element to transition to dynamic VoiceXML containing the weather information. Our implementation, however, benefits from using the `<data>` element (see 4.1) to obtain the information from the Web Proxy (see 4.2) and, ultimately, from an existing Web page. Web Proxy obtains the information from one of the configured sources (Yahoo! Weather [32], Weather Underground [33]) and converts it to a uniform XML format. The application fetches the information from XML and says it back to the caller.

The *VoiceQuote* application helps to stay up-to-date on stock quotes. A user calls VoiceQuote and says a company name. Stock quote of the company is then retrieved from an available Web financial data provider (e.g. Yahoo) by Web Proxy. Finally, the quote is read back to the user (see Fig. 1).

5 VoiceXML Integrated Development Environment

VoiceXML Integrated Development Environment (*VXML IDE*) is a Web-based tool for development and management of voice applications, with emphasis on usability. Applications are designed in text or graphic mode. In text mode the developer writes the VoiceXML code directly, with the option to check the validity of the document by a VoiceXML validator. Graphic mode enables to create a voice application without knowledge of VoiceXML (see below). VXML IDE offers machine translation of voice applications to other languages (using third-party Web translators). Saved applications become *instantly available* to callers.

VXML IDE has a *three-tier architecture* with data, logic and presentation tiers (see Fig. 3). We further divide the logic tier into management and computation components. The presentation tier provides user interface (UI). As the

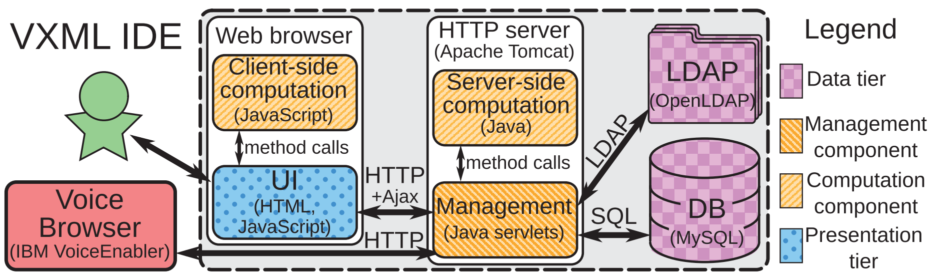
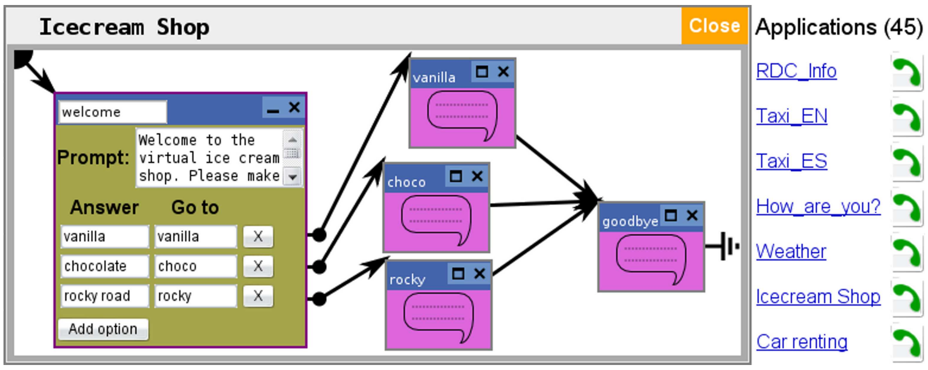


Fig. 3. VoiceXML IDE: the three-tier architecture



Welcome to the virtual ice cream shop. Please make

Answer	Go to	
vanilla	vanilla	X
chocolate	choco	X
rocky road	rocky	X

Fig. 4. VoiceXML IDE: graphic mode screenshot

VXML IDE is Web-based, it is coded in HTML and JavaScript and runs on the client. The data tier stores the user information and developed voice applications. Using LDAP directory for user information enables us to use VXML IDE user accounts in other applications, too (we use them for forum, bug-tracking system and Asterisk). For storing voice applications, relational database is a natural choice. The management component controls access to the resources of the data tier. It comprises several Java servlets within a HTTP server. The computation component performs tasks such as converting between textual and graphical representation of an application, translation of an application, or VoiceXML validation. To relieve burden from the server, an effort is made to put it mostly on the client. However, some computation is still needed on the server.

The graphical design of a voice application (see Fig. 4) consists of several graphical components, each representing a simple dialog. The call flow is illustrated by links between components. Internally, each component is a JavaScript object that implements a certain interface. Each component has methods to output its content as VoiceXML and to reconstruct (load) itself from a VoiceXML snippet. Turning the graphical design into VoiceXML then comes down to iterating over all the components and asking them for their VoiceXML output. To store the graphical design and to be able to return to it later, we extend VoiceXML by adding new elements and attributes that hold the graphical information.

The Web-based VXML IDE is openly available at [31] to the world-wide community, who are thus encouraged to indulge in voice-application experimentation!

6 Experimental Validation

In the experimental validation we focus on the promptness of the system interaction and the capacity issue. Section 3 discusses the attribute of scalability and reliability and Section 5 explains the rapid prototyping of novel services.

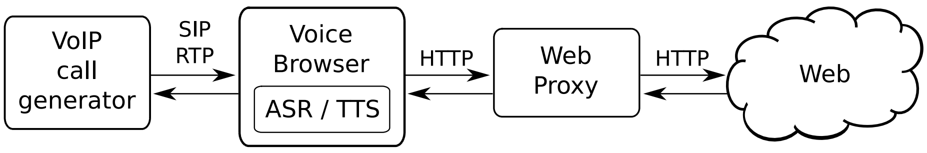


Fig. 5. VoIP call generator produces different amount of traffic load to stress Voice Browser and Web Proxy capabilities.

6.1 Test Setup

The test setup is shown in Fig. 5. VoIP call generator (SIPp [25]) stresses the architecture with different traffic patterns. It produces signaling messages to manage variable amount of SIP calls and simultaneously generate streams carrying payload. It is also the point where the total response delay is measured. Generated traffic is received by the Voice Browser (IBM Voice Enabler), which makes use of ASR and TTS engines (IBM WebSphere Voice Server [27]). Voice Enabler governs the call connection and VoiceXML dialog processing. Web Proxy is used to retrieve Web information requested by voice applications. Part of the delay added by Web requests is measured separately.

The total response delay was derived by analysis of RTP streams as the interval between the caller query end instant (e.g. "London") and the initial time of response arrival (e.g. "Weather in London is.."). The Web delay was measured by analysis of HTTP packets between Voice Browser and Web Proxy, see Fig. 6.

The Voice Browser and the Web Proxy each run on a 2 GHz Intel Pentium 4 server with 2 GB RAM. The ASR/TTS engines utilize a 2 GHz Intel Xeon server with 2 GB RAM. All servers run OS Linux.

6.2 Delay Measurement

Ordinary traffic test simulates the conditions of a typical load. The Weather application (see 4.3) with Yahoo! Weather as the Web source received 500 queries for weather conditions in a city randomly selected from thirty european capitals. The intervals between call arrival times were exponentially distributed with a mean of 1s. Test call scenario was approximately 10 seconds long.

Results of the total response delay per call are in Fig. 7. Two cases were studied: (1) calls where the weather information is retrieved via Web-Proxy

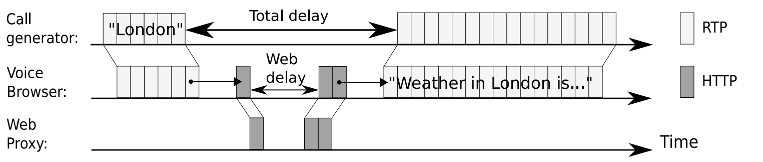


Fig. 6. Total response delay is measured as the time between the end of caller query in RTP and the start of Voice Browser response in RTP.

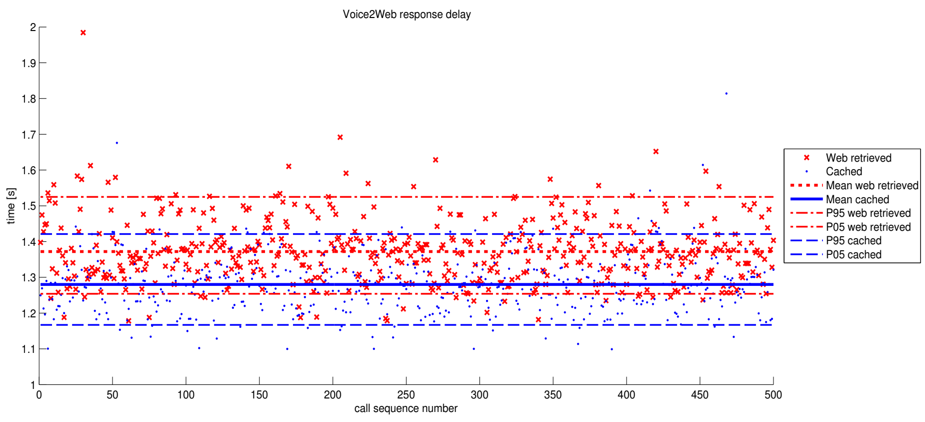


Fig. 7. Total response delay in case of Web-retrieved and cached information (500 values, mean and 5th and 95th percentile).

from the Web. The average delay is 1.37 seconds; (2) utilizing the Web-Proxy caching, where the average system response delay is 1.28 seconds. The cache can save hundred milliseconds per call for applications where information refresh interval is in the range of minutes (such as Weather). Table 1 discusses the Web-query part of the delay.

Call delay variation is caused by the non-deterministic speech recognition process (difference ∼200 ms per word), Voice Browser grouping requests to ASR/TTS engines (up to 500 ms) and variable initial Web-response delay.

System capacity measurement determines the throughput of our platform in the testbed environment. We maintained a constant number of parallel calls, initiating a new call every 1-2s (uniformly randomly) after any of the calls terminated.

measured average response time of			expiration (T_e)	threshold interval
WP, cached (T_c)	WP, uncached (T_u)	Web (T_w)		
2.55ms	104.01ms	88.53ms	10min	55.54min

Table 1. Web delay of the Weather application with Yahoo! Weather as the Web source. We measured average response time of Web Proxy when the requested document was cached (T_c) and uncached (T_u). We also measured the response time of direct queries to Yahoo! Weather (T_w) to determine when the expected response time of Web Proxy is shorter than that of the Web. If we set the expiration period of cache (T_e) to 10 minutes, we get (applying the formula from the end of Section 4.2) that the *threshold interval* between requests for one location is 55.54min. More frequent requests mean shorter expected response time of Web Proxy.

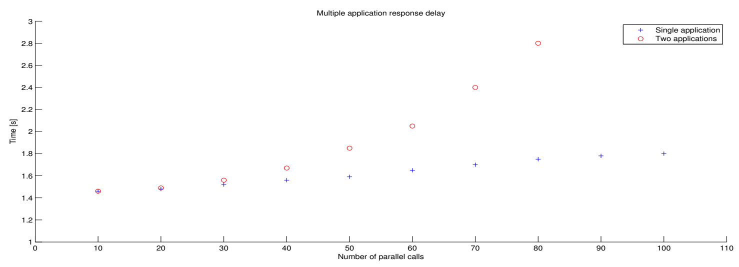


Fig. 8. Total response delay in dependence on the number of parallel sessions.

First, multiples of tens of parallel calls using Weather application with identical query were generated. The total response delay increased almost constantly with a step of around 40ms per 10 increased sessions, as shown in Fig. 8. However, for values approaching 100 parallel calls, the fraction of rejected calls became significant, causing a lower increase in delay as more system resources remained available for the accepted calls. Second, two voice applications (Weather and VoiceQuote) with equal load share were tested. The effect of additional application was insignificant for lower number of concurrent calls. When reached the maximum of 80 parallel calls the total response delay almost doubled. The maximum number of parallel calls was determined by system capabilities and limited by hardware resources.

Summary We evaluated the feasibility of an architecture for voice-application-managed access to the Web. From the human-computer interaction point of view the total response delay indicates a natural form of communication, where the interaction with the Web introduces only a small fraction of the delay that can be further reduced by caching. In the testbed environment we have reached the amount of 100 concurrent calls to single voice application with a per-call total response delay increase of about 15%. Measurements have indicated a dependence between the number of concurrent voice applications and the system response delay. The effect of multiple applications is insignificant for lower number of concurrent calls, however it can increase the delay by almost 100% in case of higher number of parallel calls in the testbed environment. This is likely caused by the increased delay in the Voice Browser and ASR/TTS engines due to multiple applications and could be avoided by application load-balancing or other scaling techniques.

7 Conclusion

Architecture discussion and experimental validation of the Voice2Web platform verifies its manageability and scalability. The platform enables rapid prototyping, replication, creation and immediate deployment of voice applications interfacing to the WWW. Performance scales well with the number of calls and improves with at least periodic cache use. Nevertheless, servicing multiple parallel voice applications may require increase in hardware capacity.

Practical realization of the Voice2Web management platform opens many possibilities, be it for specialized services for the communities of the handicapped or illiterate, or for commercial applications. Multi-lingual applications may be built and the Web-based IDE, openly available at [31], encourages experimentation and allows code-sharing among developer communities. An *open issue* remains designing an architecture for automating and managing the *reverse process of creating WWW content using voice.*

Validation of the architecture principles opens space for creation of a *multi-modal interaction management frontend to the resources of the World Wide Web*, allowing to build applications that respect or combine different modalities (voice, visual, haptic, etc.). Investigations of adding further modalities, such as visual avatars, 3D representations or visual pattern recognition, further contextual aspects, such as user location and behavior, as well as security and robustness considerations, are all part of the future activities of the project.

Acknowledgment

We wish to thank IBM Research for generous support of the project and Jan Kleindienst, Borivoj Tydlitat and Jan Sedivy for many thoughtful suggestions. We also thank CESNET for the generous lending of project equipment and Vodafone Foundation Czech Republic for the kind student scholarship support.

References

1. *Mobile cellular and Internet user penetration worldwide*, ITU 1997-2007 ICT Market Information and Statistics, http://www.itu.int/ITU-D/ict/statistics/maps.html
2. D. B. Roe and J. G. Wilpon; Editors, *Voice Communication Between Humans and Machines*, The National Academies Press, Washington D.C., USA, 1994.
3. *Voice Extensible Markup Language (VoiceXML) Version 2.0*, W3C Recommendation 16 March 2004, http://www.w3.org/TR/voicexml20/
4. *Voice Extensible Markup Language (VoiceXML) 2.1*, W3C Recommendation 19 June 2007, http://www.w3.org/TR/voicexml21/
5. VoiceXML Forum, http://www.voicexml.org/
6. World Wide Web Consortium (W3C), http://www.w3.org/
7. C. Gonzles-Ferreras and V. Cardeoso-Payo, *Building Voice Applications from Web Content*, TSD 2004, LNAI 3206, pp.587594, 2004, Springer-Verlag Berlin 2004.
8. Rahul Ram Vankayala and Hao Shi, *Dynamic Voice User Interface Using VoiceXML and Active Server Pages*, APWeb 2006, LNCS 3841, pp. 1181 1184, 2006.
9. S. Agarwal, A. Kumar, A. A. Nanavati and N. Rajput, *The World Wide Telecom Web Browser*, Poster at WWW 2008, April 21-25, 2008, Beijing, China.
10. A. Kumar, N. Rajput, D. Chakraborty, S. K. Agarwal, A. A. Nanavati, *WWTW: The World Wide Telecom Web*, NSDR, August 27, 2007, Kyoto, Japan.
11. E. L. Goldman, E. Panttaja, A. Wojcikowski and R. Braudes, *Voice Portals - Where Theory Meets Practice*, Int. Journal Of Speech Technology 4, 227-240, 2001.
12. S. K. Agarwal, D. Chakraborty, A. Kumar, A. A. Nanavati, N. Rajput, *HSTP: Hyperspeech Transfer Protocol*, ACM Hypertext, Sept. 10-12, 2007, Manchester, UK.

13. Sujan Shrestha, *Mobile Web Browsing: Usability Study*, Proceedings of ACM Mobility, September 10-12, 2007, Singapore.
14. Min Yin and Shumin Zhai, *The Benefits of Augmenting Telephone Voice Menu Navigation with Visual Browsing and Search*, Proceedings of ACM CHI:Managing Voice Input, April 22-27, 2006, Montreal, Quebec, Canada.
15. V. L. Hanson, J. T. Richards, and C. C. Lee, *Web Access for Older Adults: Voice Browsing?*, Universal Access in HCI, Part I, HCII 2007, LNCS 4554, 904-913, 2007.
16. K. Christian, B. Kules, B. Shneiderman, A. Youssef, *A Comparison of Voice Controlled and Mouse Controlled Web Browsing*, ASSETS, 2000, Arlington, VA, USA.
17. I.V. Ramakrishnan, Amanda Stent and Guizhen Yang, *HearSay: Enabling Audio Browsing on Hypertext Content*, WWW 2004, May 17-22, 2004, New York, NY, USA.
18. Zan Sun, Amanda Stent and I.V. Ramakrishnan, *Dialog Generation for Voice Browsing*, W4A Workshop at WWW 2006, May 23-26, 2006, Edinburgh, UK.
19. Yevgen Borodin, Glenn Dausch, I.V. Ramakrishnan, *TeleWeb: Accessible Service for Web Browsing via Phone*, W4A2009 collocated with WWW 2009, April 20-21, 2009, Madrid, Spain.
20. Frost, Richard A., Ma, Xiaoli and Shi, Y., *A browser for a public-domain Speech-Web*, In Proceedings of the ACM WWW 2007, Banff, Alberta, Canada.
21. Frost, Richard A., et al., *MySpeechWeb: Software to Facilitate the Construction and Deployment of Speech Applications on the Web*, Proceedings of ACM SIGACCESS ASSETS'08, October 2008, Halifax, Canada.
22. Kolias. C., Kolias, V., Anagnostopoulos, I., Kambourakis, G., Kayafas, E., *A pervasive Wiki application based on VoiceXML*, Proceedings of PETRA '08, ACM, July 15-19, 2008, Athens, Greece.
23. Kawanaka, S., Masatomo, K., Takagi, H., Asakawa, C., *Accessibility Commons: A Metadata Repository for Web Accessibility*, SIGWEB Newsletter, Issue Summer, June 2009, ACM.
24. Di Guoqiang, Liu Yaoyao, Han Lingchao and Wu Jianping, *Design and Implementation of Voice Web Pages for Online Shopping Based on .NET and Streaming Media*, Management of e-Commerce and e-Government, 2008, ICMECG '08, 17-19 Oct. 2008, Nanchang, China, Page(s):226 - 229.
25. *SIPp test tool and traffic generator*, http://sipp.sourceforge
26. *Asterisk Private Branch eXchange*, http://www.asterisk.org/
27. *IBM WebSphere Voice Server*, http://www-01.ibm.com/software/voice/
28. A. Gulbrandsen, P. Vixie, L. Esibov, *A DNS RR for specifying the location of services (DNS SRV)*, IETF RFC 2782, Feb 2000, http://tools.ietf.org/html/rfc2782
29. S. Shanmugham, P. Monaco, B. Eberman, *A Media Resource Control Protocol (MRCP)*, IETF RFC 4463, April 2006, http://tools.ietf.org/html/rfc4463
30. *Voice Browser Call Control: CCXML Version 1.0*, W3C Working Draft, 19 January 2007, http://www.w3.org/TR/ccxml/
31. *Voice2Web VoiceXML IDE*, http://bolek.feld.cvut.cz:8080/vxmlide/
32. *Yahoo! Weather*, http://weather.yahoo.com
33. *Weather Underground*, http://www.wunderground.com