An Architecture for Seamless Access to Distributed Multimodal Services

David Pearce, Jonathan Engelsma, James Ferrans, John Johnson
Motorola Labs
david.pearce@motorola.com

Abstract
In this paper we present a standards-based architecture that enables ubiquitous access to distributed speech and multimodal services. Motorola's "Seamless Mobility" initiatives are focused on giving people continuity of services regardless of context, location, device, or type of network connectivity. Speech technology is a key aspect of seamless mobility: not only does it make it easier to enter information on small devices, but it allows devices to be used in contexts where eyes and hands cannot be used. But the visual modality comprised of keypad input and display output is also vital: speech input can't be used in very noisy environments or where privacy is an issue, and speech output is difficult to remember. Multimodal systems combining visual, speech, and other modalities are therefore crucial for seamless mobility. The architecture depends on standard VoIP protocols such as SIP and RTP. It also uses standard web languages for voice dialogs (VoiceXML) and visual dialogs (XHTML, J2ME). And to provide very high performance speech recognition we use the Distributed Speech Recognition (DSR) standards. The result is a responsive system that places minimal demands on the device and maximally leverages existing mobile content ecosystems.

1. Introduction
There are now more than 1.6 billion mobile phone subscribers worldwide, with two billion expected by the end of 2006. While ordinary voice calling remains the dominant "application", mobile devices are becoming increasingly sophisticated, with new features like instant messaging, multimedia messaging, cameras, web browsers, games, video, and music. Mobile networks are also improving, with 2.5G packet data networks widely available and 3G networks starting to appear. High speed data networks can deliver a wide range of audio and visual information to devices, and they enable access to content while on the move. User interface technology has not improved at the same rate, and small keypads and displays are still barriers to usability. But integrating a speech modality to the existing visual modality helps greatly: if at any point in a dialog the user is prompted with audio along with the visual prompt, and can respond with speech as well as the keypad, the resulting multimodal user interface greatly enhances the mobile user's experience and the effectiveness of the interaction.

In the following sections we describe the architectural goals and give an overview of the architecture, authoring approaches and protocols that flow from these goals. The result is a standards-based architecture that delivers seamless access to multimodal services.

2. Architectural Goals
Our end-to-end architecture had to meet these objectives:
- **Support mid-tier, mass-market devices, not just high-tier smart phones.** Therefore speech recognition had to be offloaded to network based servers, and the device resident software kept as small as possible.
- **Support existing 2.5G networks.** Therefore network bandwidth and latency had to be strictly managed, and an efficient synchronization protocol developed.
- **Have excellent speech recognition rates.** This meant that speech being sent to the voice server should be encoded in the ETSI Aurora DSR standards [1,2].
- **Be open standards based.** Voice dialogs should be defined in the W3C's VoiceXML 2.0 markup language [3], while visual dialogs should similarly be standards based, using XHTML or J2ME. Audio formats, the synchronization protocol, etc. should either use existing standards or have a clear path to future standardization.
- **Leverage existing content ecosystems.** It's desirable to provide an architecture that can draw on the extensive network of companies working to produce content in VoiceXML 2.0, XHTML, and Java to help drive application creation. So the approach builds on commercial VoiceXML servers with minor extensions.

3. Overall Architecture
The architecture is outlined in Figure 1. The right half of the diagram, comprising the voice server and the web server, is essentially the same setup that supports pure voice services today over the Public Switched Telephone Network (PSTN). When you call such a service, your handset's microphone is connected to the voice server's automatic speech recognizer (ASR), while the server's audio subsystem, which includes a text-to-speech synthesizer (TTS), connects to the speaker. The voice browser manages the ensuing dialogue, deciding what to say and what to listen for at each step. The browser is driven by VoiceXML markup documents fetched from the web server, and also fetches the resources needed by each markup document, e.g., audio prompts played to the caller. The web application is just a traditional web application that generates VoiceXML pages instead of HTML. How do we adapt a standard voice server to work in a multimodal environment, in which the handset has an XHTML browser, a Java J2ME environment, or some other means of rendering visual information?
The first step is simply setting up the audio channels over packet data. The Session Initiation Protocol (SIP) is used to set up the session between the mobile handset and a (multimodalized) voice server. The audio generated by the server is transported using the particular voice codec for the wireless network: Figure 1 illustrates the situation on a GSM network using the AMR format. In the uplink direction, the user’s speech is fed to the voice server. The ETSI Aurora DSR features are used here because they give significantly better speech recognition rates.

At this stage we have nearly all we need to support a pure voice modality that uses DSR. The only additional piece needed is a small amount of control logic on the device (perhaps written as a Java MIDlet) to oversee the session establishment and manage the audio channels. A typical function of this logic is to turn on DSR transmission when the push-to-talk (PTT) key is depressed, and to turn off transmission when the key is released. PTT is an excellent way to limit interference from background noise.

The next step is to integrate the visual modality with the voice modality. This is a matter of getting the two “user agents” – the voice browser on the server and the visual browser (or other visual interface software) on the handset – to coordinate their activities. Our approach is to integrate the coordination logic in with the visual modality, so that the voice modality is essentially a slave of the visual modality.

In a traditional voice-only call to a voice server, the VoiceXML voice browser is in control of the dialogue. At any point in time the browser is evaluating a VoiceXML form using the Form Interpretation Algorithm, or FIA. The FIA is essentially a loop in which the next field to listen for is determined, the relevant prompts are played to the user, the speech input is collected from the user and analysed by the ASR, and the results of the speech recognition are used to decide what to do next.

To enable the device to control the voice browser session, the FIA needs to be modified so that on each loop it first checks for any control messages from the device and after each recognition it tells the device about any new field values and field focus changes. In order to do this, a Controller object is introduced to manage inbound messages and outbound responses from the FIA. In the default (voice-only) case the Controller does nothing, but in the multimodal, or remote-controlled case, the Controller will have messages the FIA must process and respond to.

Control messages are in our Voice Browser Control Protocol (VBCP). This protocol is at a high conceptual level that minimizes message traffic by letting the voice browser operate semi-autonomously. There are VBCP messages to initiate the session, tell the voice browser to load a particular VoiceXML document, and start up the FIA for a particular VoiceXML form in the currently loaded document. Once the voice dialogue for a form is running, the voice browser tells the VBCP Controller object about any speech recognition results (e.g., “city is London”, or “there was a no match error”) or focus changes (“the current field is date”), and the VBCP Controller sends this information back to the device. On the device, the “+V Framework” (discussed below) feeds these recognition results and focus change events into the visual interface. Similarly, actions made using the visual interface cause field values and focus change events to be sent via the “+V Framework” to the VBCP Controller, which tells the FIA to update its field values and focus. When the visual side decides to advance the overall dialogue state, it tells the voice browser to run another VoiceXML form in the same (or perhaps different) document, or it simply tells the voice browser to do nothing until further notice.

4. Authoring – the “Plus V” Approach

Our authoring model is the “+V” approach, so called because it adds a voice modality to any visual modality. If the visual modality is via an XHTML browser, the authoring model is “X+V”. If the visual modality is done in the J2ME Java environment, the authoring model is “J+V”, and so on.

The voice modality is implemented by a VoiceXML voice browser session, which has a loaded VoiceXML document comprising a number of forms (dialogs) that can be interpreted. This document can be viewed as a remote object and its forms as the methods implemented by that object. The device has a framework – the “+V” Framework – that abstracts away the details of this remote method invocation. This framework also handles session setup and the overall interaction and synchronization management.

4.1. VoiceXML

VoiceXML is to the voice web what HTML is to the visual web. It was first published by the VoiceXML Forum in 1999, revised in early 2000, and then turned over to the World-Wide Web Consortium (W3C). For the last five years, the W3C has carefully refined VoiceXML, and VoiceXML 2.0 reached Recommendation status in early 2005 [3]. The W3C
is finishing up VoiceXML 2.1 and beginning VoiceXML 3.0. Commercially, VoiceXML has seen very substantial uptake, displacing a large segment of the proprietary interactive voice response languages and increasing the overall voice services market. Even very large systems, such as the North American directory assistance service, now use VoiceXML. This industry trend towards VoiceXML motivated its use at the center of this multimodal architecture.

4.2. XHTML + Voice Profile
IBM, Motorola, and Opera have proposed the XHTML + Voice Profile (X+V) language for multimodal authoring, and have submitted it to the W3C for consideration [4]. X+V leverages the W3C’s existing “standards stack” by using XHTML 1.0, VoiceXML 2.0, and XML Events. The XHTML document is viewed as a container in which other modalities’ markup can reside. Because X+V closely relies on W3C mechanisms, we believe it is a good approximation of the eventual W3C multimodal markup language.

For a time, one critique of VoiceXML was that it could not be used in multimodal contexts, but X+V demonstrates that in fact it is highly effective for this purpose.

4.3. Java + Voice
XHTML is not the only authoring approach used by mobile application developers. Others include Microsoft’s .NET Compact Framework, Qualcomm’s BREW, and of course Java 2 Micro Edition (J2ME). To demonstrate that the Compact Framework, Qualcomm’s BREW, and of course application developers. Others include Microsoft’s .NET Compact Framework, Qualcomm’s BREW, and of course Java 2 Micro Edition (J2ME). To demonstrate that the architecture can be used in these other environments, we have developed a variant that supports Java + Voice, or J+V. This involves extending J2ME’s LCDUI user interface primitives so that the “+V” Framework gets notifications of focus and value changes, which it forwards to the voice modality. The MIDlet also looks after any visual rendering.

5. Protocols and Other Standards
Any content ecosystem for multimodality has to be based on open standards or it will not be accepted by carriers, application developers, tool vendors, device manufacturers, and other participants. +V uses open authoring standards, including VoiceXML 2.0, XHTML 1.0, XML Events, and Java. Session setup uses SIP and SDP. For audio standard ETSI Aurora DSR packets are streamed to the server, and standard audio packets sent back to the device, using RTP.

Remaining work involves tracking the emerging architectural standards from the W3C and the Open Mobile Alliance (OMA), and markup standards from the W3C. The synchronization protocol will need to be standardised in the IETF, and the “+V” Framework’s Java APIs via the Java Community Process.

6. Speech Input and Output
6.1. DSR
The desire for improved user interfaces for distributed speech and multimodal services on mobile devices has motivated the need for reliable recognition performance over mobile channels. Performance needs to be robust both to background noise and to any errors introduced by the mobile transmission channel. Much work has taken place in the telecommunications standards bodies of ETSI and 3GPP to develop standards to achieve this and enable interoperable services of high performance.

The performance advantages of DSR are well documented: for example see [5,6]. The use of DSR avoids the degradations introduced by the speech codec and channel transmission errors over mobile voice channels:

- By using a packet data channel (for example GPRS for GSM) to transport the DSR features, instead of the circuit switched voice channel normally used for voice calls, the effects of channel transmission errors are greatly reduced and consistent performance is obtained over the coverage area.
- By performing the front-end processing in the device directly on the speech waveform, rather than after transcoding with a voice codec, degradations introduced by the codec are avoided.
- In addition, the DSR advanced front-end is very noise robust and halves the error rate in background noise compared to the mel-cepstrum front-end, giving robust performance for mobile users who are often calling from environments where there is background noise.

3GPP (3rd Generation Partnership Project) is the body that sets the standards for GSM and UMTS mobile communications. In June 2004, 3GPP approved the DSR Extended Advanced Front-end [7] as the recommended codec for “Speech Enabled Services”. A fixed-point standard for this DSR front-end has also be specified [8]. This selection was based on extensive evaluations made by two of the leading ASR vendors (IBM and Scansoft) that confirmed the performance advantages of DSR compared to the normal voice codec [6]. In these tests DSR gave 36% reduction in word error rate compared to AMR 4.75. The significance of the selection by 3GPP is that DSR will find widespread deployment in future GSM and 3G mobile handsets.

While 3GPP uses AMR as its voice codec for person-to-person communication other networks have standardized upon different voice codecs. As a result there are many different voice codecs, each tuned to give balance of tradeoffs for the each particular system at the time the selection was made (e.g. latency, frame size, channel capacity etc). Each has different consequences for recognition performance e.g., the EVRC codec used in CDMA is substantially worse than AMR. In addition, to get the best performance the recognizer’s acoustic models should be trained using data transmitted over the same codec that it is used for. For a single application server to support access from many networks many different sets of models are required. By comparison, DSR provides a better alternative for each of these networks and can be supported with single common set of DSR acoustic models at the voice server. The benefit to the user is better and consistent performance; the benefit to the service provider is ease of implementation and maintenance.

6.2. IETF RTP Payload Formats for DSR
In addition to the standards for the front-end features themselves, protocols for the transport of these features from the device to the server are also needed. The IETF Real Time Protocol (RTP) is a well established mechanism for the transport of many different media types including video,
VoIP, and music. Associated with RTP are also the SIP protocols for session initiation and codec negotiation. By defining a RTP format for the DSR features, services benefit from all of the added functionality of this set of protocols, as well as the support of other media types for multimodal applications. Formats for the RTP payloads for all the DSR standards have reached the RFC stage in the IETF [9].

Within these payloads any number of frame pairs may be sent within a packet. For the front-ends on their own this takes 12 bytes per frame pair and with the extension it takes 14 bytes per frame pair. The choice of the number of frame pairs to send in each payload depends on the latency and bandwidth of the channel available.

6.3. Speech Output

For speech output from the system the voice server generates audio prompts in the same way as it would for a voice-only call using text-to-speech synthesis and pre-recorded audio files. To transport this to the device over packet data the audio is compressed using the appropriate codec for the target device and transported using RTP from the server to the client. The particular codec chosen depends on what is best for the receiving device. For example, GSM or 3GPP devices would use AMR, CDMA devices would use EVRC, and landline VoIP is likely to use G729. The choice of codec is made during the initial session startup negotiation using the Session Description Protocol (SDP). Alternatively, the speech server can generate the speech as a waveform and use a separate transcoding system block to encode to the desired target codec. The particular data rate of the codec to use depends on the bandwidth of the channel, for example for a GPRS channels its appropriate to use the lower data rates like AMR 4.75 or 5.15 in an identical way to what has been standardized for push-to-talk over cellular (PoC). For higher data rate channels like 802.11, however, higher rate options like AMR 12.2 can be selected to get better quality.

6.4. Transmission Latencies

The total overhead for the protocol headers in the stack is quite high as shown in the table below.

<table>
<thead>
<tr>
<th>Data</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTP</td>
<td>12</td>
</tr>
<tr>
<td>UDP</td>
<td>8</td>
</tr>
<tr>
<td>IP</td>
<td>20</td>
</tr>
<tr>
<td>Total:</td>
<td>40</td>
</tr>
</tbody>
</table>

*Table 1: Protocol header sizes for packet data transport.*

For low data rate channels such as GPRS it is appropriate to use multiple frames (DSR uplink or coded speech on downlink) per RTP payload to reduce the total bandwidth and therefore latency (e.g., four to ten). For higher data rate channels perhaps with residual packet loss such as UMTS a smaller number of frames per packet can be used (e.g., one or two). In testing a prototype implementation it has been found that even on GPRS the latencies are quite acceptable (less than two seconds) and for higher speed channels much less.

In future mobile networks it is expected that header compression (RoHC) will be available reducing the 40 bytes for the RTP, UDP, IP layers down to about four bytes.

7. Conclusions

We have described the main features of a standards-based distributed multimodal architecture called “+V”. Client/server communications within this architecture use well known IP protocols that enable a high degree of independence from the packet based data transport over which they run. For example, the services can operate in 2.5G GSM networks using GPRS as the data bearer. The same services are even better supported by higher speed networks like EDGE (EGPRS) or 3G data in UMTS. Similarly, mobile users of CDMA networks can use CDMA 1x data to access these services. As the deployment of WiFi increases, users can get the benefits of higher data rates and in building signal strength in the home, in enterprises or from hotspots. There are similar parallels in the wireline world both for voice, as VoIP services expand, and broadband and cable access to the home. Thus it can be seen how this architecture supports ubiquitous access to multimodal services – something Motorola calls “Seamless Mobility”.

8. Acknowledgements

Many people have contributed: Victor Adut, Rohit Chaudhri, Francis Jatico, Michael Pearce, Mark Randolph, Praveen Sanigepalli, Saritha Sivapuran, Yibing Song, Jerome Vogedes, Qiaoing Xie.

9. References