

Requirements, Design points and Recommendations for Conversational Distributed Protocols and Conversational Engine Remote Control

Stéphane H. Maes, IBM
smaes@us.ibm.com

1. Introduction

In this document, we present preliminary requirements for conversational protocols to support:

- Conversational Distributed Protocols
- Remote Control of Conversational Engines.

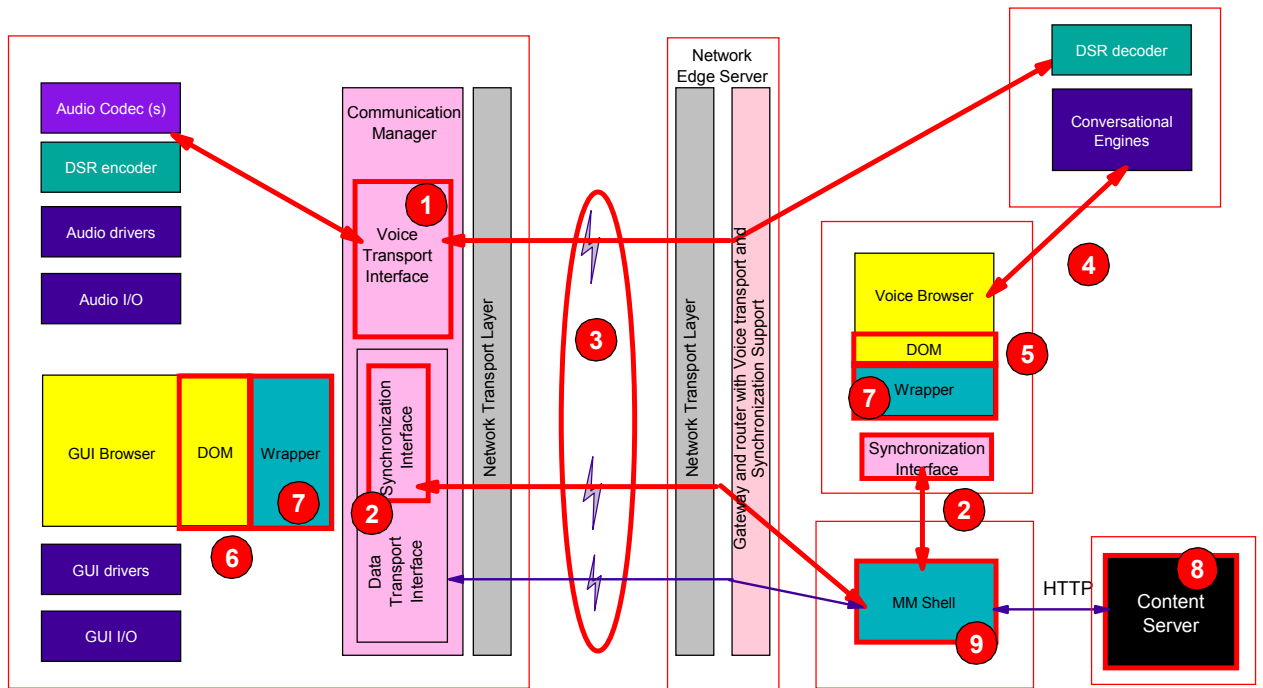


Figure 1 – Standards, protocols and interfaces to specify to support the multi-modal browser reference architecture

This document defines the tasks of specifying items 1 and 4 in figure 1 that illustrated the current multi-modal browser reference architecture.

In this document, we use the term “**conversational**” engines or technologies to denote engine and middleware used to support spoken dialogs (speech only or speech and other modalities), with no particular assumption in terms of the level of “conversational capability”. It includes input

technology (e.g. speech recognition that it be grammar-based, LM-based LVCSR, NLU; speaker recognition etc...) output technology (TTS, prompt concatenation/splicing, NLG) and possibly dialog management technology.

By DSR (Distributed Speech Recognition) scheme, we mean an encoding scheme of raw acoustic speech waveform into acoustic features designed to minimize the adverse effects of channel distortion on recognition performance.

2. Conversational protocols

We distinguish between:

- **Conversational transport protocols** that enable coding and transport (streamed or not) of the speech I/O in manner compatible with the different conversational engines. When voice is a DSR encoding scheme is used up stream – towards the server, we speak of **Conversational Distributed Protocols**.
- **Conversational control protocols** that enable:
 - Synchronization of the different renderers
 - Remote control of the conversational engines.

We identified SOAP [1] as a candidate protocol to synchronize different renderers.

We also identified DOM as the recommended interface to exchange events and update the views. Issues associated to the synchronization interface and protocols are beyond the scope of this document. Clearly it is possible to use the conversational protocols addressed here in conjunction with other synchronization mechanisms (e.g. DOM sub-sets, other proprietary events-based methods, etc...).

3. Conversational transport protocols

Whenever a voice channel is available simultaneously to a data channel (voice and data – e.g. GPRS or W-CDMA), and enough bandwidth is available, this voice channel can be used to transport voice to the conversational engines.

However, this approach has some challenges:

- Decent voice and data channels are not yet widely available over wireless networks (or even over modem connections...) and it will take time before this capability will have worldwide coverage.
- Conventional voice channel may degrade significantly the voice signal transmitted to distributed conversational engines. This may result into unacceptable accuracy degradations.

This emphasizes the value of distributed speech recognition approaches (**DSR**) [2,3]. It is not the object of this activity to discuss non-DSR based speech transport protocols: conventional voice over IP specification and voice compression schemes can be directly used, possibly in conjunction with conversational engine remote control protocols.

Conventional DSR [3] relies on the a priori specification of an acoustic front-end and an associated encoding scheme. Speech input are processed through this front end and compressed before transmission to a remote engine. The remote engine decodes the acoustic features and performs its conversational function.

3.1. Value Proposition of DSR

The fundamental value proposition of DSR is that it relies on a compression scheme that has been optimized to minimize the distortion of the acoustic features that result in degradation of recognition performance. This is at the difference of other compression schemes designed with other conventional cost function in order to fundamentally minimize for a given bit rate, some perceptual impact of distortions of the waveform or the spectrum; with no regard for the impact of the compression on some acoustic features.

The following drivers support the use of DSR:

- Limited client resources with respect to the conversational engine requirements
- Too low bandwidth to send data files from the server to a local conversational engine
- Delay to send data files from the server to a local conversational engine
- Proprietary aspect of such data files (grammars, acoustic models etc...)
- Security (client side authentication is a weak security solution)
- Network & system load management
- Specialized conversational engines using specialized algorithms and function not provided by generic engines and typically not supported by client engines.

However, there are significant challenges to conventional DSR.

3.2 Challenges of conventional DSR

Different coding schemes have been proposed to guarantee that speech compression and transport does not introduce any degradation of the conversational engine performances. For example, the ETSI Aurora Front-end working group has established different work items to specify such front-ends. The latest work item aims at specify robust front-ends.

The standardization of a front-end for speech recognition is extremely challenging. For example [2] uses an alternative scheme with a different acoustic front-end and a different compression scheme. Also, every single speech vendor has a different acoustic front-end optimized for its recognition algorithm. Most of the time these front-end change as a function of the task. This is especially true with new front-ends as for example described in [4,5,6].

Also numbers of vendors use different acoustic front-ends for other conversational engines like for speaker recognizers, while others use the same front-ends with possibly different transformations [5,6].

As a result, even after many years of research in conversational technologies, it seems premature to impose a frozen acoustic front-end specification.

Besides the acoustic front-end issues, it is also very difficult to test acoustic feature compression scheme. Even for a same front-end, the distortions introduced by a scheme may be acceptable on numbers of test tasks (low perplexity and complexity grammar based recognitions), but unable to address more complex tasks (e.g. LVCSR).

Until now, these considerations have severely limited the endorsement of DSR by speech vendors despite its undeniable advantages.

In addition, the bit rate associated with existing DSR schemes like Aurora WI-7, WI-8 or [2] may be to high compared to other existing codecs like GSM AMR 4.75. Therefore, for specific network connections or applications, it may be important to provide compromises between bit rate and minimization of the acoustic feature distortions.

3.3 Conversational Distributed Protocols

As explained above, we speak of Conversational Distributed Protocols when the conversational transport protocols use a DSR encoding scheme to transmit voice uplink towards the server. We propose to define a protocol stack that enables negotiation of the DSR encoding scheme rather than a priori selection of a particular encoding scheme.

This constitutes an attractive solution that provides the same advantages as conventional DSR without restricting application developers and speech vendors into sub-optimal front-ends for a particular task.

Speech dialogs as well as multi-modal interactions impose similar real time constraints as human-to-human conversations. Therefore conversational distributed protocols must be real-time and designed with criteria similar to Voice over IP. We will designate them as RT-CDP.

We define conversational distributed protocols as the protocol stack that enable real-time streaming of DSR data (upstream) and downstream of perceptually (most probably – DSR can be used but there are no obvious reasons to do so) coded data. It includes the handshake mechanism for selection of the upstream and downstream codecs / DSR encoders, at the beginning of the exchange or dynamically during the interaction.

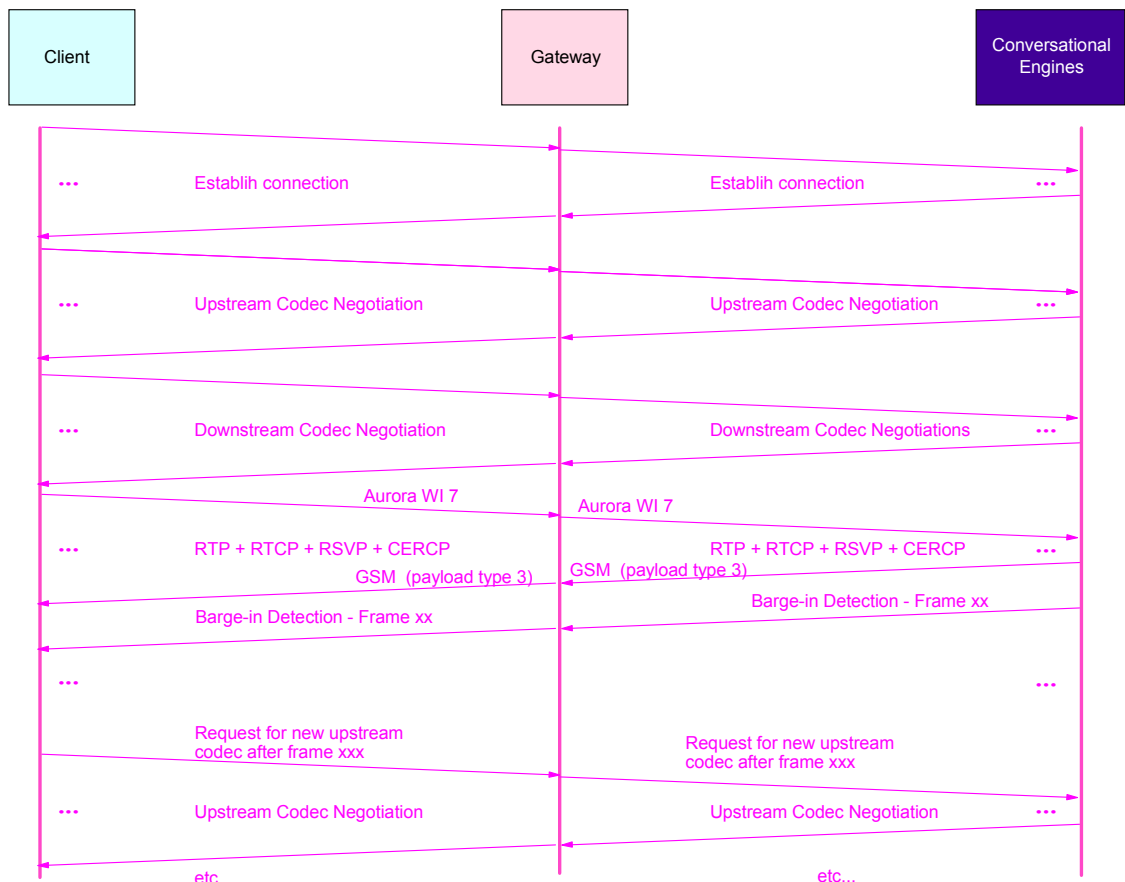


Figure 2: Abstract representation of RT-CDP exchanges

3.4 Requirements and Recommendations for RT-CDP

An abstract representation of RT-CDP is presented in figure 2.

We propose the following requirements and recommendations for conversational distributed protocol stack:

- The conversational distributed protocol stack should fit Voice over IP protocol stacks with minimum modifications/ extensions (if possible with no required modifications).
 - Should fit H.323 as well as the SIP protocol stacks [7].
 - Should be compatible with H.323 or SIP gateways and terminals.
 - This may require appropriate registration of the headers and payload with appropriate standard bodies (IETF; 3GPP).
 - Because RT-CDP is expected to be important for wireless networks, where packet losses can be a significant problem, it may be appropriate to target the protocol design mostly for SIP over UDP instead of SIP over TCP or H.323. Indeed, Voice over IP over TCP seems to suffer more from packet losses than RTP over UDP.
 - However, we recommend throughout this section:
 - To remain as compatible as possible with the H.323 stack
 - Consider using some key principles of H.323 (especially H.245) to implement specific behaviors of RT-CDP.
- A unique nomenclature must be defined to support some key default DSR schemes:
 - Acoustic front-end (e.g. Aurora front-end(s))
 - Compression scheme (e.g. Aurora compression)
- This nomenclature should be compatible with H.245 tables of H.323 or SIP codecs (conventional/perceptual coders)
- We should consider GSM (FR, HR, EFR or AMR xx) and/or G.723.1 as possible (downlink) default codecs. These are just possible examples. GSM FR is available on mobile phones anyway and might be a natural choice for them. But mobile phones are not the only devices in the scope of RT-CDP.
- For mobile devices other than GSM-based mobile phones, we should consider other possible downlink codecs, such as the CDMA QCELP (IS-96), The CDMA EVRC (IS-127) for CDMA-based phones, True Speech G.723.1 MobileVoice, GSM 6.10, etc. used for personal digital assistants (PDAs) with operating systems such as WindowsCE, Palm, or Symbian OS.
- We should select a default DSR scheme and a default codec to be supported by default (cf H.323 approach). This aims at minimizing incompatible connections and also address the concerns of handset and other embedded clients that can only support one DSR compression scheme.
 - A mechanism should be provided to immediately identify if an end-point supports the default codec and scheme or not.
- The default DSR scheme (and in general most DSR schemes) should support reconstruction or no reconstruction configurations
- A mechanism should be provided to describe arbitrary (i.e. non default and non-parametrized) DSR schemes (e.g. a XPath namespace conventions [8])
- DSR data should be wrapped/transmitted in RTP streams along with its RTCP control layer.
 - The data streamed in the RTP stream (i.e. DSR stream) should really use Error Correction Code mechanisms to reduce the effect of packet losses. A compression scheme in the nomenclature includes a particular ECC mechanism.
- Call settings and control messaging should be limited to fit H.323 stack or SIP messages:
 - SIP or H.323 could be arbitrarily selected as design point as it is (or rather, it will be) possible to convert (gateway) between SIP and H.323. However as stated before, the initial design point should probably be SIP to provide maximum robustness to packet losses by relying mostly on UDP.
 - We recommend specification of the messages for both stacks
- Codec negotiation must be supported by a socket connection active throughout the communication (TCP or UDP with a confirmation mechanism).

- Because H.245 provide exchange of tables of supported codecs and DSR schemes, we recommend using a H.245 scheme. This enables terminal to select on the basis of the terminal capabilities. We can also use or specify a corresponding SIP codec negotiation.
- Upstream and downstream coders must be separately negotiated.
- The codec tables (or SIP negotiation) should enable pointing to an object code (e.g. applet to download to implement a particular DSR scheme, etc...)
 - Should include a mechanism to negotiate the type of object code (i.e. applet, vs OS specific binaries etc...)
 - Security issues associated to this feature should be carefully considered. We recommend that the protocol supports the capability to specific codec object code. At the same time, a terminal or server may decide not to support this capability and not accept it during codec negotiation or to accept it only after appropriate authentication of the provider of the object.
- The H.245 connection (H.323) or the Session Initiation connection (SIP) should be used to dynamically change the codec / DSR schemes as needed throughout the evolution of the application (e.g. to ship different acoustic features for a particular utterance): the protocols should permit acoustic front-end parameters to change during a session.
 - We need to evaluate if this can achieved through a different RTP stream (different ports) or by simply switching the codec starting after a given packet number.
 - The latter option goes beyond what H.245 or SIP provides.
- Similarly new RTP connections can be opened when extra DSR streams must be provided (e.g. to provide simultaneous speech and speaker recognition (using different acoustic features)).

We recommend also considering two additional mechanisms not currently provided by conventional Voice over IP stacks (but not necessary incompatible):

- Capability for the source to specify that an utterance should be transmitted with guaranteed delivery (e.g. TCP or UDP + confirmation instead of RTP).
- Capability for the recipient to request repetition of a particular utterance segment specified by its end points (within a given time frame after initial transmission)
- Capability for the sender to request confirmation that the recipient has received a particular utterance segment.
- Guaranteed delivery (utterance segments and other protocols) should account for possible losses of connections:
 - Threads that wait for confirmations that will never arrive because of a loss of connection should appropriate unblock or terminate with events that the client application (or server application) can handle.
 - Guaranteed delivery mechanisms should fail and return an error after a parametrized time (or amount of re-transmission).

We should investigate the impact of DSR on QoS messages and criteria.

- QoS typically needed for DSR
- Capability to dynamically change the required QoS during a session.
- Should have minimum impact on existing QoS protocols (RSVP)

Eventually, RT-CDP introduce new requirements:

- The need to appropriately handle barge-in, an issue usually automatically handled by the interlocutor in a human-to-human communication, but not by conversational engines.
- When barge-in can be detected by the local DSR encoder or via Voice Activity Detection, it could block the downstream audio play.
 - It would make sense to notify the remote engines that the output play has been interrupted and indicate at what packet number.

- To reduce the delay associated with end-point detection [6,7], it is useful to perform barge-in and voice activity detection on the client and to transmit this information in parallel to the RTP DSR stream to the remote engine. End-point information should be transmitted in a separate (control) channel. It is a protocol requirement rather than an encoder feature.
- When barge-in is not detected by the local DSR encoder or when it is better processed on a remote engine, it should be possible to send a control signal that will interrupt output playback on the client.
 - Clearly network latencies make this approach challenging.
- We can identify three mechanisms to do so:
 - Reuse an existing available connection:
 - RTCP layer with extension to send elementary control messages
 - H.245 or codec negotiation connection to send elementary control messages
 - Open an additional dedicated connection for this kind of control messages.
- These alternatives should be examined with the objective of being supported now (or with minimum changes in the immediate future) by Voice over IP and wireless gateways.

We may want to consider compiling a list of classes of RT-CDP connections:

- Function of the client capabilities
- Function of the network characteristics and available bandwidth.
- In terms of default codecs, we may want to propose a scheme per classes of RT-CDP connections.
 - In particular clients would be expected to support at least one RT-CDP default encoding schemes and the server may have to support several possible default encoding schemes.

Open Issues:

- Should we target close compatibility with SIP and H.323 or select one?
- What are the default uplink and downlink codecs / coding schemes that should be considered?
- Etc...? (Please send us your comments and additional issues)
- Given the limited client resources and low bandwidth of the wireless Internet today and in the near future, it seems challenging to squeeze in (1) application code, (2) DSR front end, and (3) transmission protocol. We need to measure how much resource is available on the client minus the application code and assess the computational complexity of the protocols and code suggested. If it is going to be a major challenge to fit all these into the limited memory and CPU resources, programmers and designers will likely bypass this solution and go for proprietary implementation of transmission protocol and DSR front-end. Yet we don't want to sacrifice flexibility and compatibility.
- Regarding security, e.g. user using a HTTPS page, how do we make sure MM-Shell propagates this security mode to the voice browser and then to the speech engine? If the speech engine is independent from the application content, how do we prevent sensitive information (maybe thru possible reconstruction) from being captured at the speech engine side? This concerns both RT-CDP and CERC.
- We'll also need to create a list of functions that Speech Engine **must** support when in relation to CERC. The paper proposed using SOAP. We'll need to find out how 'thin' SOAP can be on a client. And how much thicker adding RTSP will be.
- Here are some of the requirements and issues we need to address:
 - What is the minimal set of functionalities that the conversational speech engine must offer? (e.g. complex grammar, NLP, speaker/noise adaptation at the client AND server)
 - For "fat" clients that contain a local speech engine, how does this protocol work in conjunction with a remote speech engine?

- How does H.323 handle load balancing of conversational engines, if we have a distributed computing architecture for speech engines (i.e. multiple engines per DSR server)? What about the scalability of the proposed RT-CDP and CERC?

4. Conversational Engine Remote Control Protocols

4.1. Motivation

The working group decided to add Conversational Engine Remote Control Protocols (CERCP) to its work item in order to specify item 4 in figure 1.

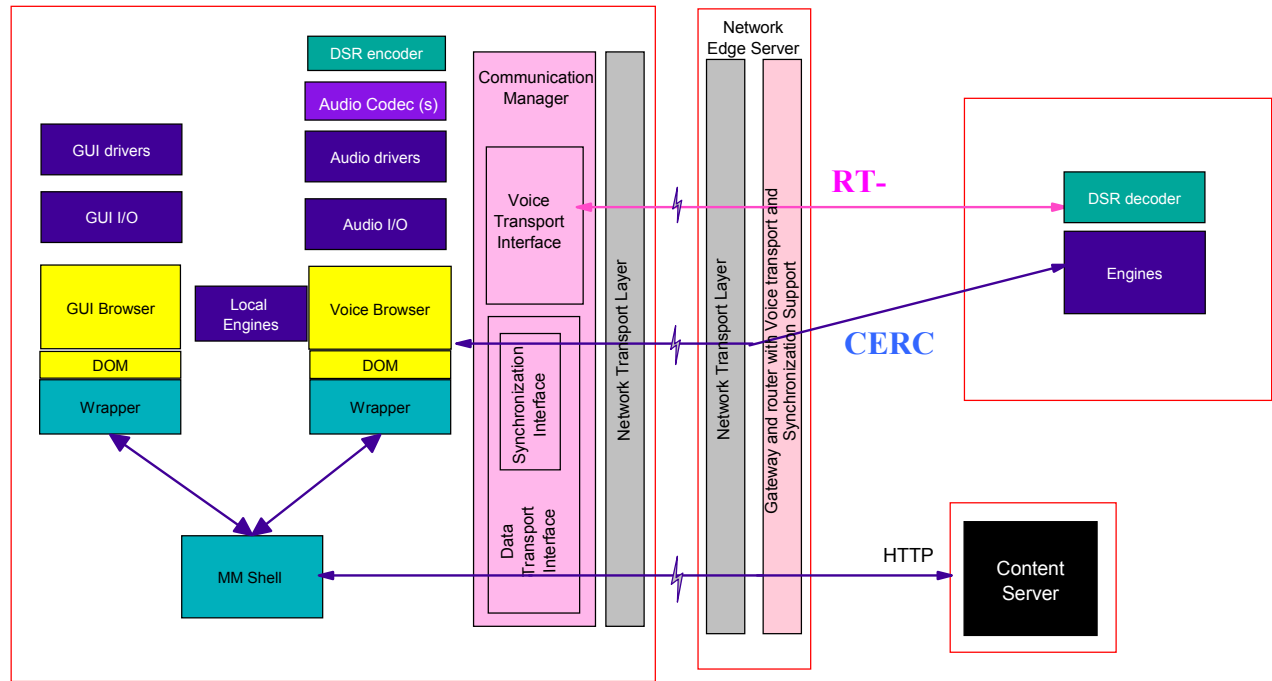


Figure 3 - Variation on the recommended multi-modal browser architecture. This configuration requires CERCP.

In figure 1 CERCP enables to distribute the conversational engines elsewhere. This enables network and system load management.

Other multi-modal scenarios require CERCP. Consider for example a fat client. As illustrated in figure 3, in this configuration, the voice browser is located on the client while the speech recognition is located on the server. Clearly, an appropriate mechanism must be put in place to remote control these engines.

However, CERCP are fundamentally needed by numbers of other DSR scenarios. In general, CERCP are needed whenever the engines are controlled:

- By the source of the audio (i.e. the client). A typical scenario is voice enabled cell phones applications using server side speech recognition.
- By a third party controller (i.e. application). A typical scenario is a server side application that relies on speech recognition performed elsewhere in the network.

As such CERCP are needed to satisfy most of the DSR drivers identified in section 3.1:

- *Limited client resources with respect to the conversational engine requirements:*

- The application can reside on the client side and drive conversational engines as if they were local
- *Too low bandwidth to send data files from the server to a local conversational engine:*
 - Remote engines can be driven remotely. Data files do not need to be sent to the client. Actually data files may remain on different remote engines without having to be sent to a particular server side engine. Server-side bandwidth requirements are also reduced
- *Delay to send data files from the server to a local conversational engine:*
 - Same comment as above. Different remote engines can be used instead of using a particular engine: we can use an engine close or that has already loaded the datafile for a particular processing.
- *Proprietary aspect of such data files (grammars, acoustic models etc...)*
 - Same comment as above: the owner of the data file can offer a recognition engine with data files as a web service.
- *Security (client side authentication is a weak security solution):*
 - Authentication can now be performed with the secure intranet
- *Network & system load management:*
 - Any available engine can now be used
- *Specialized conversational engines using specialized algorithms and function not*

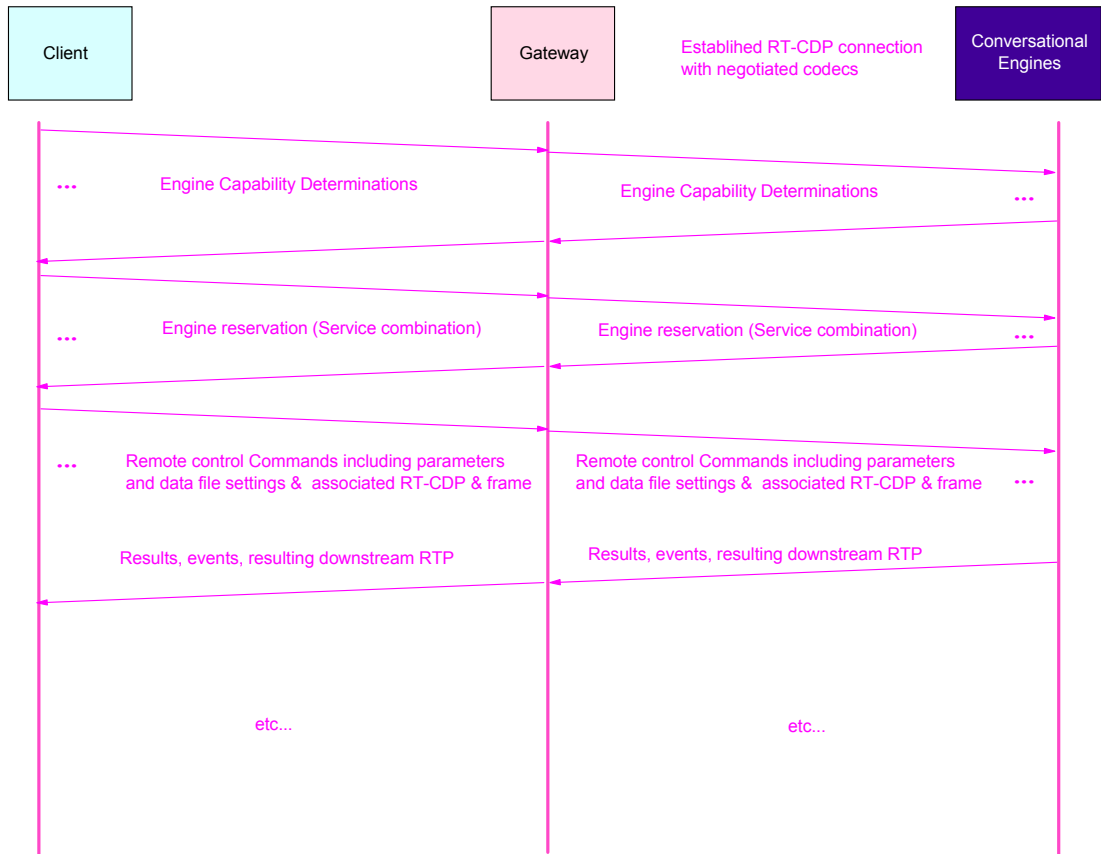


Figure 4 – Abstract representation of CERCAP exchanges

provided by generic engines and typically not client engines:

- The appropriate engine can be used, independently of where it is located.

4.2 Challenges

There are some challenges associated to CERCP:

- Past speech recognition APIs (and other conversational APIs) have received marginal engine vendor support.
 - Typically because of a too poor level of functionality
 - Difficulty to manipulate results and intermediate results (usually with proprietary formats).
- On the other hand, complex APIs for numerous engines and functions is a very complex and possibly intractable task.

4.3. Requirements and Recommendations

An abstract representation of CERCP is presented in figure 4.

Let us start with some high level principles:

- CERCP should be limited to Conversational Engine Remote Control.
- Call control functions should be left to the application or to a system/load manager.
 - In other words, CERCP does not provide re-direction decision mechanisms or mechanisms to re-direct a call. CERCP only specify how to remote control an engine (with the engine and the RTP stream (RTCDP) well identified).
- CERCP should not aim at specifying the format, commands or interface that an engine can or should support.

We recommend specifying a framework with:

- A set of widely supported commands
- Formalism to:
 - Pass parameters
 - Specify data files
 - Communicate/treat Events
 - Return results
 - Result may include the RT-CDP downlink stream (e.g. RTP stream of a TTS engine)
- Offer a mechanism to advertise the supported commands (cfr OPTIONS in RTSP) [9].
- Offer a mechanism to advertise the interface associated to a particular command and the function that it performs.

As a starting point, we recommend:

- Use as starting point RTSP (Real Time Streaming Protocol), which is already designed as a remote control protocol [9].
- Consider WSDL as a mechanism to describe the commands / interface supported by a given engine [10].
- As first widely basic command set, we should target to fully support VoiceXML 1.0 [11] (and extensions [12]) functionality:
 - Following the use of the Multi-modal browser architecture as guidance, we should use as criterion for the first version of the basic set, to be able to implement a full VoiceXML interpreter using the CERCP
 - This requires some extensions to VoiceXML to specify:
 - Remote engine to use
 - Data files to use.
 - We recommend putting a simple proposal together as needed to support CERCP and submitting to W3C.
 - Parameters and results should fit W3C Voice specifications [12] when appropriate. This should include support for arbitrary parameters and input (possibly based on Xschema [13]).
- As implementations are provided and command exposed, we can extend the basic set.

- To implement the remote control commands, we recommend considering SOAP over RTSP [9].

5. References

- [1] Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000, <http://www.w3.org/TR/SOAP/>
- [2] S. H. Maes, D. Chazan, G. Cohen, R. Hoory, Conversational Networking: Conversational protocols, for transport, coding and control, ICSLP 2000, Beijing, October 2000.
- [3] ETSI STQ Aurora DSR Working Group, http://webapp.etsi.org/tbhomepage/TBDetails.asp?TB_ID=275
- [4] G. Saon et al. Maximum Likelihood Discriminant Feature Spaces, ICASSP'2000, Istanbul
- [5] U. Chaudhari and S. H. Maes, [Pattern-Specific Maximum Likelihood Transformations and Speaker Recognition With Sparse Training Data](#), Submitted to IEEE Trans ASAP.
- [6] U. Chaudhari et al. Transformation Enhanced Multi-Grained Modeling for Text-Independent Speaker Recognition, ICSLP 2000, Beijing
- [7] O. Hersent et al., IP Telephony, Packet-base multimedia communication systems, Addison Wesley, 2000.
- [8] XML Path Language (XPath), Version 1.0, W3C Recommendation 16 November 1999, <http://www.w3.org/TR/xpath>
- [9] H. Schulzrinne et al., Real Time Streaming Protocol (RTSP), Network Working Group, RFC 2326, April 1998, <ftp://ftp.isi.edu/in-notes/rfc2326.txt>
- [10] Erik Christensen, et al., Web Services Description Language (WSDL) 1.0, September 2000, <http://www6.software.ibm.com/software/developer/library/w-wsdl.pdf>
- [11] VoiceXML 1.0, Submitted to W3C, April 2000, <http://www.w3.org/Submission/2000/04/>

6. Related documents

- S. H. Maes, Y. Muthusamy and W. Wajda, Multi-modal Browser Architecture Recommendation, "Clayman proposal" to the ETSI DSR Application and Protocol Working Group, ETSI, January 16, 2000
- S. H. Maes et al. Multi-Modal Browser Architecture Proposal - A Quest For Bronzemannship or better ..., Clayman Draft - ETSI DSR Application and Protocol Working Group, Amsterdam, February 16, 2001
- S. H. Maes and R. Reng, "Requirements and Recommendations for Conversational Distributed Protocols and Conversational Engine Remote Control; Version 0.5", ETSI AU/310/01, May 3, 2001.
- Y. Kim, Revised version of "Requirements and Recommendations for Conversational Distributed Protocols and Conversational Engine Remote Control; Version 0.3", ETSI AU/310/01, April 24, 2001.
- S. Balasuriya, S. H. Maes, D. Pearce and R. Reng, Call Control & Media Transport requirements for multimodal systems, ETSI STQ Aurora DSR Applications and Protocols sub-group, Joint input from Motorola, IBM and Temic, AU/328/01, version 0.1, May 3, 2001.
- W. Wajda, Alcatel comments on AU/328/01 and AU/310/01, May 21, 2001
- ETSI/Aurora, Input on feature vector size and latency from ASR vendors, ETSI STQ DSR e-mail poll sent by Hari Garudadri, May 9, 2001

- S. H. Maes, IBM Input on feature vector size and latency from ASR vendors, ETSI/Aurora, 5/21/01
- P. Walther, DSR Latency Figure, in SpeechWorks Input on feature vector size and latency from ASR vendors, ETSI/Aurora, 5/18/01
- Call Control Requirements in a Voice Browser Framework, W3C Voice activity, W3C Working Draft 13 April 2001 <http://www.w3.org/TR/call-control-reqs/>
- S. H. Maes et al., IBM-Nokia joint proposal for Multi-modal Browser Architecture, Overview, Version 1.0, Submitted to ETSI DSR Applications and Protocols Working Group, December 1, 2000.
- S. H. Maes et al., Multi-modal Browser Architecture Recommendation for Mobile e-Business, IBM - Nokia joint proposal to the ETSI DSR Application and Protocol Working Group, December 5, 2000