

LE1-1930

Deliverable D 3.2 & D 3.3 - public version



This report is the public version of deliverable D 3.2 & D 3.3 (confidential)

Table of contents

1. Introduction.....	2
1.1. Purpose.....	2
1.2. Scope.....	2
1.3. Background	2
1.4. Glossary	3
1.5. Standards.....	3
2. System objectives.....	4
2.1. General Description	4
2.2. System Capabilities.....	4
3. Speech Server.....	4
3.1. Architecture overview.....	4
3.2. Programming model	6
3.3. Resource Manager API	7
3.3.1. Introduction	7
3.3.2. Resource Manager API overview.....	7
3.4. Speech Recognition API (SR-API).....	7
3.4.1. Introduction	7
3.4.2. Speech Recognition API (SR-API) overview	8
3.4.3. Data structures	8
3.5. Speaker Verification API (SV-API)	8
3.5.1. Introduction	8
3.5.2. Speaker Verification API (SV-API) API overview.....	8
3.6. Template database API	9
3.6.1. Template database API overview.....	9
4. Speech Server integration.....	9
4.1. Speech bus.....	9
4.1.1. Introduction	9
4.1.2. Integration requirements	10
4.2. Data bus	11
4.2.1. Introduction	11
4.2.2. Integration requirements	12
5. Implementations.....	13
5.1. Overview	13
6. SV Template Database	14
6.1. Introduction	14
6.2. Real-time analysis.....	15
6.2.1. Low/moderate usage centralized SV system	15
6.2.2. High usage centralized SV system.....	15

This work has been supported by the Telematics programme of the European Union and in Switzerland by the Office Federal de l'Education et de la Science (Bundesamt fuer Bildung und Wissenschaft)



1. Introduction

1.1. Purpose

This document provides the draft Technical Specification for the design of the CAVE Speaker Verification Demonstrator systems in each of the banking and telecom application domains.

The document is based on the Functional Design Specification (please refer D3.1) and corresponding detailed technical meetings and correspondence held with CAVE project partners. These efforts will be adopted as the design specification for WP6 Demonstrator Building.

1.2. Scope

The document addresses:

- CAVE speech server system objectives and architecture
- CAVE speech server software architecture (API)
- CAVE speech server hardware and integration issues
- CAVE speech server SV template database
- Existing banking and telecom system architectures
- CAVE Telecom SV demonstrator
- CAVE Banking SV demonstrator

The technical specification will be restricted to the system components required to implement the demonstrators described in the Functional Specifications. Some details of application dialogues will not be addressed in this document as they are not defined precisely in the Functional Specification but in any case should be implementable using the current technical design.

The technical specification is based on expected functional and technical requirements for SV algorithms and SV template storage, maintenance and access. Updates to the technical design may be required in the light of developments in both of these areas in WP4 and WP7 (field trials and validation), although WP4 work must take due account of the fundamental architecture and technical constraints set by the current technical design described in this document.

1.3. Background

The project will implement demonstrators in three stages, in order to:

- provide prototype systems for early feedback into design and service provision
- incorporate more advanced SV algorithms as they are developed during the project lifecycle
- incorporate system improvements (functional and technical, including human factors) as a result of experience gained with initial demonstrators
- support later planning and design of systems for in-service integration

The three demonstrator stages are nominally:

1. standalone demonstrator of SV functionality
 - basic SV enrolment and authentication functions
 - not necessarily including speech recognition
 - not necessarily implementing target application functionality
 - not integrated with existing service



2. standalone demonstrator of basic application incorporating SV
 - including speech recognition if required by application
 - implementing (mimicking) main target application functionality
 - not integrated with existing service
3. fully integrated demonstrator system
 - implementing full target application functionality
 - integrated with existing service

1.4. Glossary

SV	Speaker Verification
Speech Server	A distinct computer system equipped with SV and Speech Recognition (SR) resources. This is a black-box offering speech services (i.e. SV API and SR API) that can be called through RPCs. Speech is provided by a SCx bus connection and RPCs travel on any LAN supporting TCP/IP and UDP/IP, like Ethernet for instance.
Telephony Server	A computer system with telephony resources and software to control them. Usually, the application runs on the telephony server, although the latter can also be driven by an application executing on a separate system.
Voice Profile / Template / User Model	The speaker verification voice templates for specific passwords or security codes generated during enrolment, and used in authentication. They will comprise HMM model parameter sets and possibly additional parameters such as client/impostor thresholds

1.5. Standards

SCSA (ECTF) has been specified in the Technical Annex as the intended standard to which CAVE project developments should adhere if possible. At present the bus and hardware standards have been well-defined and are adopted by a number of CTI vendors and manufacturers, including Dialogic who provide a wide range of industry standard SCSA-compatible boards suitable for project use and approved for installation in many countries.

The software standards are still in development, although working documents are now available. These will be used to guide the software architecture design in CAVE, but it should be noted that until a stable standards document has been ratified by the industry, project developments will only partially meet these standards. It is hoped that by the end of the project the final systems will fully comply with the emerging ECTF standards, but this will depend upon the timetable for their approval and the constraints upon project resources.

- SCSA (ECTF):
 - Enterprise Computer Telephony Forum
 - S.100 Revision 1.0 Media Services "C" Language
 - Application Programming Interfaces
- Dialogic Product and Services Guide, 1996



2. System objectives

2.1. General Description

The objectives of the demonstrator are:

- develop a real-time system to demonstrate the technology
- investigate system, hardware, software issues required to implement it
- support a field trial to evaluate speaker verification performance, user interface, customer reaction
- identify capabilities and limitations of speaker verification in real applications
- demonstrate capability of system to business units

2.2. System Capabilities

The Speech Server capabilities are outlined below:

- support for speech recognition
- support for text dependent SV
- support for text independent SV
- support for text prompted SV is not required by CAVE
-

3. Speech Server

3.1. Architecture overview

The Speech Server architecture enables efficient sharing of expensive speech recognition (SR) and speaker verification (SV) resources among multiple Voice Response Units (called Telephony Servers throughout this document). Based on the client/server model, the Speech Server architecture distributes the basic voice processing tasks among specialised systems to achieve better overall performance and resource utilization.

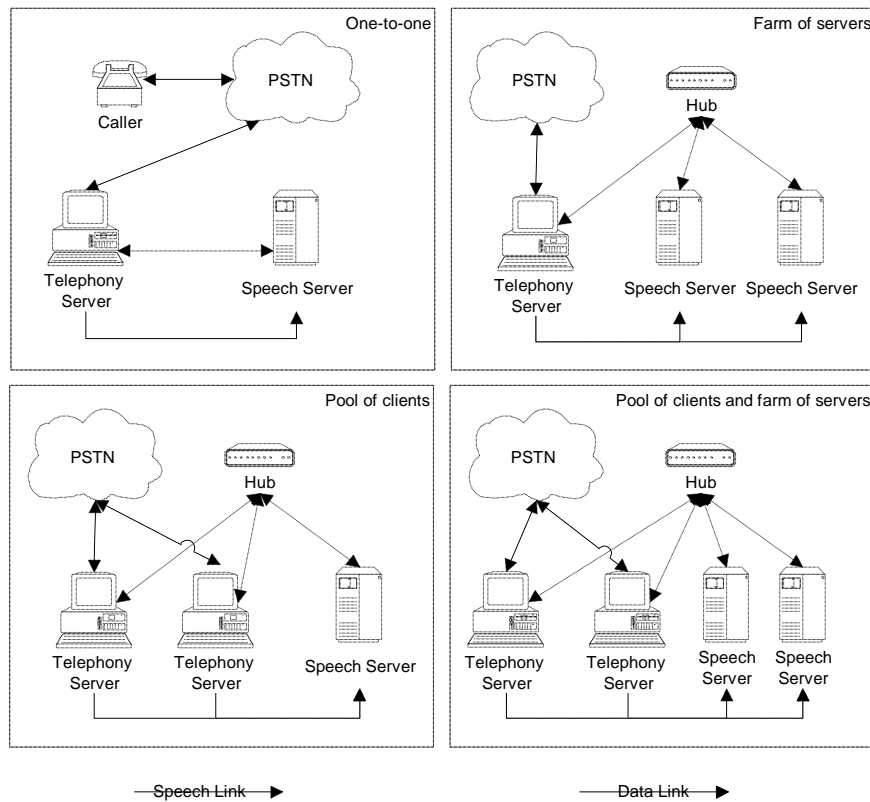
Speech Servers are also ideally suited to speech technologies prototyping and evaluation because they offer a well defined, distinct and confined, controlled environment to experiment innovative hardware and software.

In a typical Speech Server configuration, Telephony Servers are client systems that hold telephony hardware and run the application itself. They are networked to Speech Servers, server systems equipped with speech recognition and/or SV hardware and software (globally called “recognisers” or “resources”).

There can be a pool of clients and only one server, or a farm of servers and one client, or even a pool of clients and a farm of servers. Such details are totally transparent to the application. Simple configurations with one Telephony Server linked to a single Speech Server are also supported.

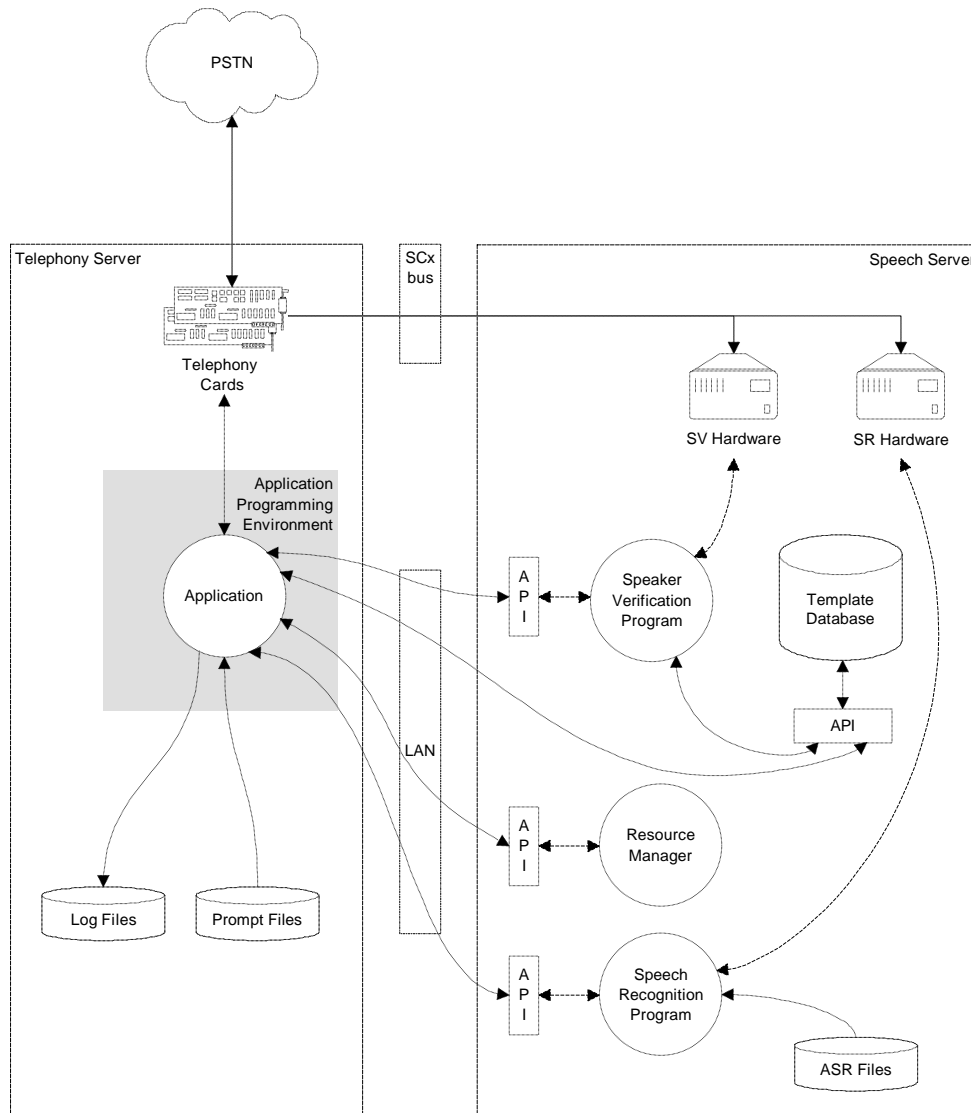
Caller Verification in Banking & Telecommunications

Project : LE1-1930
Deliverable : Initial Technical Requirement Specification
Reference : D 3.2 & D 3.3 – public versions



Clients don't necessarily have to be networked together but need to be linked to every single server using a TCP/IP-UDP/IP LAN or WAN. Although data and commands are transmitted through RPCs, the recogniser gets its speech data for recognition purposes from an external speech bus: the SCxbus. As a result, clients and servers are doubly linked: by a data-only TCP/IP-UDP/IP network and also by an external SCxbus speech bus.

Speech Servers can be fitted with any Vocalis recogniser hardware. Recognition resources are controlled remotely, from a voice processing application running on a Telephony Server, by calling Speech Server API functions. Invoking a distributed function causes a Remote Procedure Call (RPC) to be issued from the Telephony Server to the appropriate Speech Server system where the function's actual code is executed. Function results are returned transparently across the data network. In effect, there is no difference for an application developer between calling a Speech Server API function and invoking a local function. The Speech Server API supports the latest Vocalis recogniser hardware/software.



Speech Servers can easily be integrated in infrastructures with existing non-Vocalis speech recognition resources and coexist peacefully with them. Applications can even be written that interleave recognitions on legacy recognisers with recognitions on the Speech Server (restrictions may apply).

The Speech Server architecture is based on open industry standards: UNIX operating system, TCP/IP-UDP/IP protocols, RPC-based API, resilient industrial PC platform, Ethernet data networking, SCx bus speech bus (Dialogic SCSA hardware), C language software.

3.2. Programming model

Applications request recognisers by executing a Remote Procedure Call (RPC) that gets transmitted to Resource Managers (see paragraph 3.1 for more information about the terms mentioned here). It is the responsibility of all the Resource Manager server processes to collaborate with a client to agree on a recogniser to be allocated. Then the application is given the IP address of the server where the allocated recogniser sits.



Afterwards, the application can invoke Speech Server API functions that will be directed to the Recogniser Libs on the server identified by its IP address. Every Recogniser Libs server process provide the full Speech Server API to applications, including SV functions. If a function isn't supported by the underlying recogniser hardware, it simply returns an appropriate failure code. Once the application has finished with the allocated recogniser, it calls the server's Resource Manager to deallocate it.

3.3. Resource Manager API

3.3.1. Introduction

The Resource Manager controls the allocation of recognisers to applications. It uses RPCs to allow any application, wherever it may actually be running on the network, to request recognisers and obtain them from any Speech Server. The Resource Manager is a regular RPC server executing as a separate process on Speech Servers.

When it is launched, an application is not bound to any particular Speech Server. As a matter of fact, it doesn't even know if any Speech Server exists: it has to discover it. In consequence, before invoking any function that interacts with a recogniser, an application has to find a suitable speech recognition resource, reserve it for its own use and get a handle that would permit later access to it. The Resource Manager helps the application carry these tasks and arbitrate conflicting requests.

Before a recognition can occur the Resource Manager must connect the incoming call's timeslot to a speech recognition resource timeslot. Once the recognition has been performed, timeslots must be disconnected. These tasks are handled by the Resource Manager with the help of additional Speech Server modules.

3.3.2. Resource Manager API overview

Function Name	Purpose
vrn_recogniser_allocate	allocate a recogniser to an application
vrn_recogniser_deallocate	return a recogniser to the pool of free recognisers
vrn_recogniser_set	set a recogniser parameter (e.g. status)
vrn_recogniser_get	get a recogniser parameter (e.g. status)
vrn_stop	stop the resource manager
vrn_application_cleanup	clear pending requests from an application

3.4. Speech Recognition API (SR-API)

3.4.1. Introduction

A recogniser consist of a specialized card, basic recognition software that must be downloaded to the board, and data files that must also be downloaded to perform specific recognition tasks. Not all of these tasks can take place in real-time. Actions can be divided into three categories: configuration time, initialisation time and run time actions.

Most non real-time actions occur at configuration and initialisation time. A set of tools is provided on the Speech Server to control these actions, like ASCII to binary conversion of grammars and recogniser boot up, including software and models download.



The Speech Recognition API (SR-API) described below only deals with run-time actions, like initiating a recognition.

All functions may, as a general rule, be run asynchronously or synchronously. Actual implementation may limit asynchronous call support to a few functions but synchronous calls will always be possible.

3.4.2. Speech Recognition API (SR-API) overview

Function Name	Purpose
vsr_recognition_start	start a recognition
vsr_recognition_retrieve	retrieve recognition results
vsr_parameters_get	get recogniser parameters
vsr_parameters_set	set recogniser parameters

3.4.3. Data structures

A Key Value Set (KVS) is an opaque type defined in the ECTF S.100 specifications. It gives API designers the opportunity to update the number and kind of results returned by an API function call without modifying the API itself since the function's formal parameter remains the same. KVS are used heavily in the SR-API both for passing argument to functions and getting results.

Since KVS function arguments must be serialised when employed as RPC parameters, their implementation is done with regards to XDR marshalling limitations and efficiency constraints (complex serialization takes time).

3.5. *Speaker Verification API (SV-API)*

3.5.1. Introduction

(NB: this is early draft material)

Non-blocking RPCs offer a way to simulate asynchronous SV. Asynchronous RPCs, also called "No Wait RPCs", cannot be automatically generated by rpcgen under SCO Openserver 5. They require the server process implementing its own custom svc_run() loop to enable non-blocking processing of incoming RPC requests and immediate reply. On the client side, at least one callback function must be defined to provide the server with a way to return RPC results. Such a callback function has to be included in a RPC server process running on the client machine (a simple svc_run() loop is good enough). In consequence a portmapper process is required on both the client and the server systems.

**]

3.5.2. Speaker Verification API (SV-API) API overview

Function Name	Purpose
vsv_verification_start	start a SV operation
vsv_verification_retrieve	retrieve SV results
vsv_parameters_get	get recogniser parameters
vsv_parameters_set	set recogniser parameters



3.6. Template database API

3.6.1. Template database API overview

(NB: this is early draft material)

Function name	Description
SV_Template_Set	store a voice profile (e.g. after enrolment)
SV_Template_Get	fetch a voice profile (e.g. before SV)

The index key to this database is a field containing the user ID and the template ID concatenated.

4. Speech Server integration

4.1. Speech bus

4.1.1. Introduction

The Speech Server architecture rely on the ability of speech recognition cards to get access to speech at the same time as it is spoken by the caller. Therefore, a realtime chassis interconnect speech bus is required to link telephony servers to speech servers. The following table summarises the available technologies and why they're suited or not:

Technology	Verdict
Audio conferencing	Degradates speech signal and may induce noise
Analog audio playout	Delay proportional to recording length; speech signal degradation
Digital audio playout	Delay proportional to recording length; expensive and tricky too
Ethernet (IEEE 802.3)	Not realtime: delay due to speech download to the recognition card
Isochronous Ethernet	Quicknet offers SCbus compatible cards, but SCO isn't supported
MVIP (interchassis flavour)	Not natively supported by Vocalis recognition card
SCxbus - Amtelco	SCO not supported
SCxbus - Dialogic	Realtime, digital, chassis interconnect bus, with SCO support

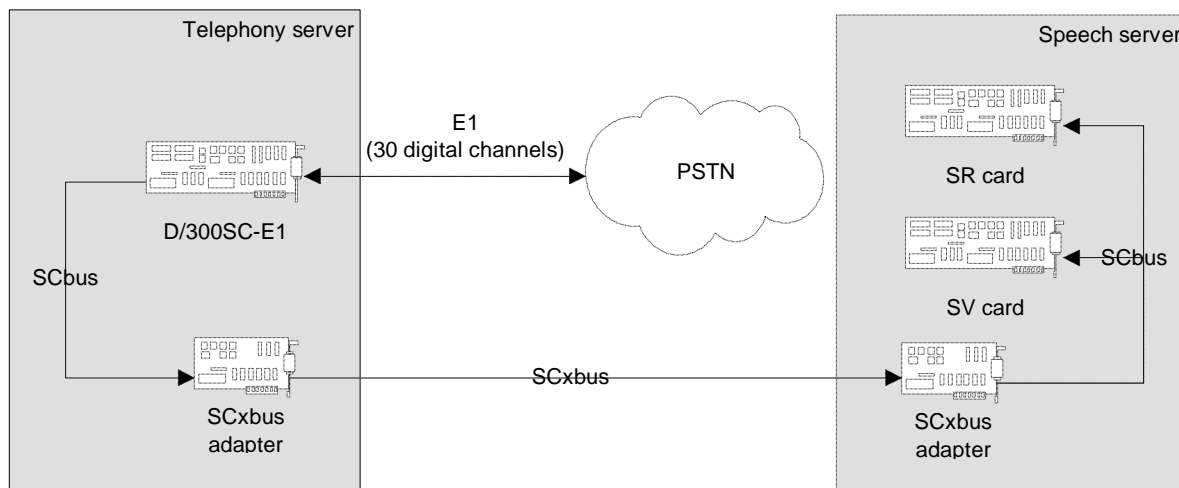
Dialogic alone offers a realtime chassis interconnect digital speech bus card (called the SCxbus adapter) with support for SCO UNIX, Interactive UNIX, UNIXWare, DOS and soon Windows NT.

Up to 16 nodes (i.e. computers), each with hardware resources to handle a maximum of 32 full-duplex audio channels, can be connected together. The SCxbus (name of the speech bus) architecture enable developers to build a 512-port multinode system. Nodes must be located within a 15 meter distance. They are "hot-pluggable".



4.1.2. Integration requirements

An example of how a Telephony Server might be connected to a Speech Server using SCxbus is shown below:



The Telephony Server must be equipped with SCbus compliant telephony cards, manufactured by Dialogic (as in the above example) or any other brand with 100% SCbus compatibility. This is necessary to connect all of these cards to a single SCbus ribbon ending at a Dialogic SCxbus adapter.

SCxbus (or more accurately SCSA MNA) specifications can be implemented in a number of different ways. Therefore, only SCxbus cards from the same manufacturer are guaranteed to be compatible. Since Speech Servers use a Dialogic SCxbus adapter board (the reason being driver support for SCO), Telephony Servers are constrained to use the very same card to ensure proper connection to the common external SCxbus. SCxbus adapters are 16-bit ISA cards with one internal SCbus port and one external SCxbus port. According to Dialogic USA, the SCxbus adapter will go into production in early July with simultaneous availability of SCO UNIX drivers. Lead time will be around three weeks.

Interface requirements summary:

- SCbus compatible telephony boards must be used internally by Telephony Servers
- a Dialogic SCxbus adapter card must be used to link Telephony Servers to Speech Servers
- an operating system supported by Dialogic's SCxbus adapter driver
- SCbus version of the generic Dialogic drivers



4.2. Data bus

4.2.1. Introduction

Telephony Servers and Speech Servers have to exchange quantities of requests and responses during the length of a typical telephone call. SV and speech recognition resources are allocated and used on-demand for maximum flexibility. Fundamentally, the Speech Server API is distributed over one or more Speech Servers, enabling transparent scalability to installations with hundreds of ports.

The distributed Speech Server API exhibits the following properties: client/server (Telephony Servers are clients, Speech Servers are servers), concurrency (simultaneous handling of multiple requests), connection oriented behaviour (reliable data exchange), stateful server (Speech Servers keep track of resource usage). In addition, cross-platform client support is a key feature to integrate with Telephony Servers built with the widest possible range of operating systems (limited by Dialogic SCxbus adapter support requirement).

Available technologies to implement the Speech Server API are shown below:

Technology	Verdict
RS-232 - proprietary protocol	Limited throughput and portability; only point-to-point connections
Ethernet - TCP/IP	Protocol design for a large API is complex and hard to maintain
Ethernet - NFS shared file	Unnecessary overhead to process API calls
Ethernet - RPC	Fast, portable, flexible, and relatively simple to code and maintain

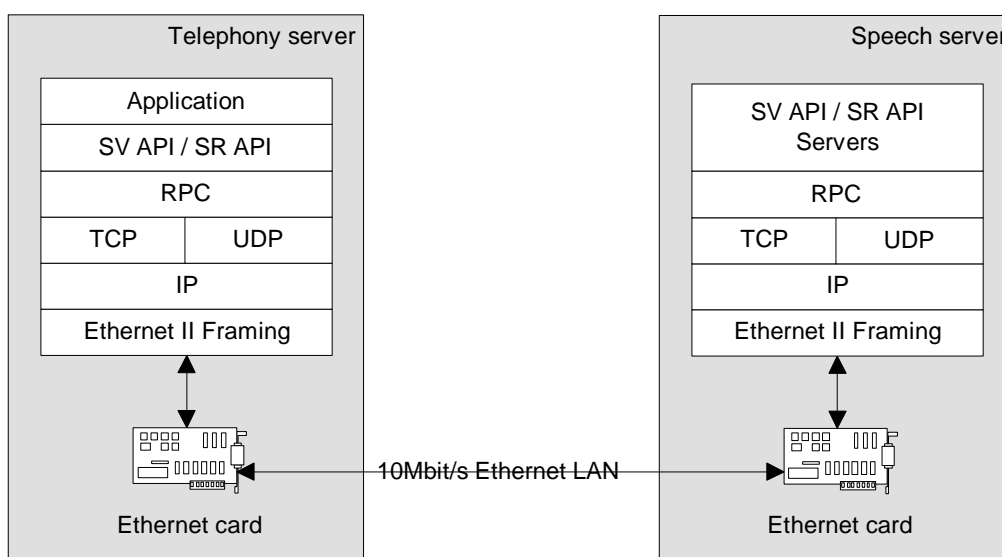
RPCs (Remote Procedure Calls), or more precisely ONC RPCs (based on the original SUN Microsystems Request For Comments RFC 1057 specifications), sit on top of TCP/IP and UDP/IP. Both protocols are available on every major platform as well as most LAN and WAN hardware, including serial/telephone lines thanks to SLIP and PPP. ONC RPCs offer transparent function calls across a network and are supported under all flavours of UNIX, as well as Windows 3.1/95/NT, OS/2, DOS, VMS, etc.

On an Ethernet LAN under average load, TCP/IP-based RPCs takes roughly 6 ms (depending on the amount of data transferred inside the functions parameters). It should be emphasized that RPCs can use TCP as their underlying transport protocol, and thus provide a reliable way of exchanging data between computers.



4.2.2. Integration requirements

The Telephony Server operating system must support ONC RPC. Modern multi-tasking OSes like UNIX, Windows NT and OS/2 provide either native support for ONC RPC or can accommodate third-party ONC RPC runtime packages. The required software components are: a TCP/IP-UDP/IP protocol stack, ONC RPC runtime with a portmapper (PORTMAP) daemon. On unsupported platforms, ONC RPC development libraries with the full set of RPC and XDR functions may be required.



Speech Servers are equipped by default with regular 10 Mbit/s Ethernet cards offering 10BASE-T (UTP) and 10BASE-2 (BNC) connectors. On-site cabling must be ready on installation and compatible. Compatibility with other networking hardware is usually possible but the Telephony Server designer should contact Vocalis well in advance to discuss any non-standard requirement. It is also the responsibility of the Telephony Server designer to make sure the selected Telephony Server networking hardware is supported by the TCP/IP-UDP/IP protocol stack and to configure it to use Ethernet II framing. A correct TCP/IP-UDP/IP configuration is critical for proper operation of the Speech Server (i.e. broadcast addresses, etc. must be setup correctly). One IP address has to be allocated per Speech Server.

Given the time-critical nature of most Speech Server API calls, Telephony Servers and Speech Servers are better installed on a separate Ethernet LAN segment to avoid packet collisions. Cabling and routing should be performed to maximize performance and reliability.

Even though data transmitted across the Ethernet LAN isn't of sensitive nature (i.e. RPC arguments and return values contain no customer details, like PIN code or account number), security conscious organisations may take the following measures to protect their data from intruders:

- isolate the Ethernet LAN segment used (physical protection)
- encrypt communication (software protection)



Depending on which operating system the Telephony Server is running, Vocalis may or may not be able to deliver pre-compiled Speech Server API libraries. Furthermore, Vocalis is committed to support only C language developers. As a result, only the availability of SCO OpenServer 5 Speech Server API static libraries (COFF or ELF) and C header files is guaranteed.

However, client side C source code files may be provided to Telephony Server developers in order to facilitate the compilation of suitable static libraries or dynamic link libraries. Please note that cross-platform portability can be an issue on non-UNIX platforms (i.e. Windows NT), especially for the most advanced features of the Speech Server API (i.e. asynchronous RPCs). Please contact Vocalis for more information.

When Speech Server libraries are installed on a Telephony Server, API functions can be invoked from the application generator, provided it is extensible and supports C language “hooks” or dynamic link libraries (DLL) function calls. For DLL calls, any programming language may be employed as long as function parameters can be read and set.

Interface requirements summary:

- ONC RPC runtime, including portmapper daemon, available on the Telephony Server
- TCP/IP-UDP/IP protocol stacks
- 10Mbit/s Ethernet card
- a C language development environment may be required
- ONC RPC development libraries may be required

5. Implementations

5.1. Overview

The calling card lab-demonstrator will be installed at KPN Telecom in Holland. KPN is aiming to build a Telephony Server on a Windows NT environment since that will be the architecture for the VIP development platform at KPN Research. Field testing is planned to be executed in PTT Telecom..

The bank demonstrators will be built at the premises of UBILAB, UBS' information technology research lab, which is the particular department of the bank involved in the project. As outlined in the WP3 functional specification document, the aim throughout shall be to host the project's technology in an environment as similar as possible to that currently deployed by the bank for its 24-hour automatic banking service..

CAVE is not allowed to disseminate operational details.



6. SV Template Database

6.1. Introduction

A realtime implementation of SV technology for telephony applications puts enormous pressure on the database subsystem used to store and retrieve callers voice profiles. The four major constraints are outlined below:

1. Template size and nature: each voice profile is a large, complex, set of information that depicts a word spoken by a user, i.e. a digit or a password. A realistic estimate puts the size of a single template around 1-2 Kb, although advanced techniques such as subwords may reduce this by some factor. If a set of password or digit templates are required this increases the storage/transmission requirements by an order of magnitude. Of course, voice profiles are speaker dependent and should be loaded afresh for each different caller.
2. Number of users: some SV systems, for instance in banking or telecommunications, are expected to have many thousands of registered users, each with its own voice profile.
3. Realtime operation: a delay of more than one second on the phone is hardly tolerable for most callers. Unless dialogue tricks (i.e. asynchronous playout of a waiting message) are employed, a given user's template has to be loaded in the time between the end of a prompt asking the caller to speak a password and him starting to say it (see paragraph 7.2 for a detailed example).
4. Distributed operation: technical or operational constraints might prevent the installation of a centralized SV system, especially in the telecommunications world. In consequence, template information must be accessible from all sites where SV resources are installed.

A crucial difference between SV and speech recognition technology is that SV templates cannot remain in memory (be it the host RAM or a dedicated SV card RAM) longer than the duration of the verification process. On the other hand, for speech recognition, speaker independent models can generally be kept in memory and accessed repeatedly without noticeable delay.

Also of interest is the fact that password-based text dependent SV systems typically require only one template to be loaded for verification, whereas text independent systems may need multiple voice profiles, thus demanding an even greater amount of information to be transferred in the same delay.

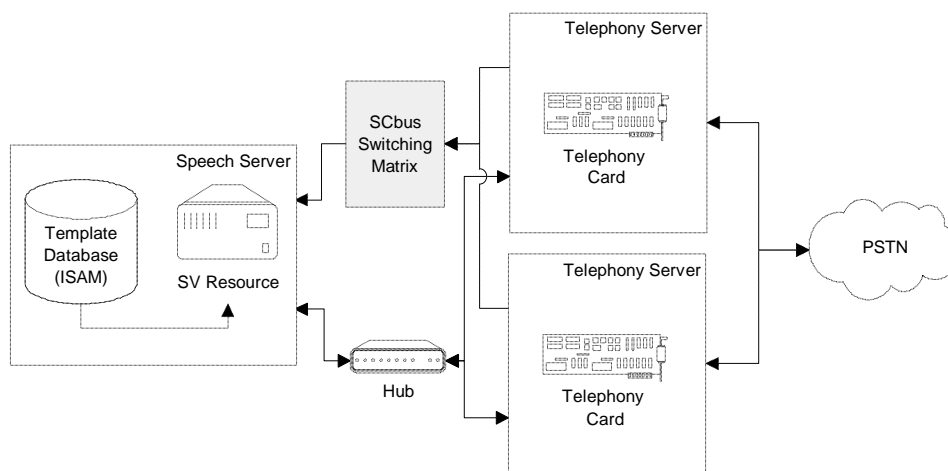
Happily enough, effective solutions to the template database problem exist.



6.2. Real-time analysis

6.2.1. Low/moderate usage centralized SV system

Solution: a single central Speech Server with the template database stored on the machine's local hard disk. The database technology used is low-level ISAM.



The above architecture will be adopted for the purpose of demonstrating SV technology for both banking and telecommunication applications as part of CAVE.

6.2.2. High usage centralized SV system

Solution 1: a series of Speech Servers with a partial template database stored on each machine's local hard disk. The database technology used is low-level ISAM.

Solution 2: a series of Speech Servers with a comprehensive template database stored on each machine's local hard disk. The database technology used is low-level ISAM with additional replication routines.

Solution 3: a series of Speech Servers with no template database stored locally. Instead, a high capacity client/server central database with fast networking equipment is used. The database technology is SQL or entity relation.