FlexCode

Project no: FP6-2002-IST-C 020023-2

Project title: FlexCode

Instrument: STREP

Thematic Priority: Information Society Technologies

D3.4 Coding requirements for the real-time demonstrator and methodology for real-channel testing

Due date of deliverable: 2008-03-01
Actual submission date: 2008-03-19
Actual submission date Rev 2.0: 2008-05-21

Start date of project: 2006-07-01 Duration: 36 Months

Organisation name of lead contractor for this deliverable: Ericsson AB

Revision: 2.0

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

1 Introduction

This document describes the coding requirements for the FlexCode real-time demonstrator and the methodology for real-channel testing. The requirements are too a large extend a consequence of the two scenarios chosen as the main focus scenarios in D3.1 “Ordered List of Real World Service Scenarios” and the setup of the real-time demonstrator hardware. The two chosen scenarios are the mobile conversation scenario and the multimedia on-demand streaming scenario. Deliverable 3.1 gives ranges for parameters like rate, delay and networks that are relevant to the described scenarios. This document narrows these ranges based on decisions within the project. Some of the suggested parameters are derived from operating points of reference codecs or parameters imposed by the particular networks tested.

In the chapter methodology for real-channel testing we describe conditions to run the FlexCode system in a way that correspond to the two selected scenarios. These include the type of signals used and the setup of the hardware platform in WP4. For the formal listening tests of WP5 the framework and rules for subjective quality evaluations, e.g., ITU-T P.800 have to be taken into account in addition to the methodology described in the current deliverable.

2 Coding Requirements for the Real-Time Demonstrator

The FlexCode demonstrator is designed to test and demonstrate the system performance in the selected scenarios. Since the target of the FlexCode system is to operate in heterogeneous networks in various operation conditions the demonstrator needs to provide corresponding environment to stress the system components and the adaptation functionality. To be able to evaluate and assess all aspects of the system, a real-time demonstration environment is needed.

This chapter will first describe the demonstrator design and secondly list the requirements for the source and channel coding for the real-time environment.

2.1 Real-time demonstrator setup

The intention of the demonstrator is to simulate the FlexCode system in various channel conditions in heterogeneous networks. However, the scope of the work is not to build complete network interface and infrastructure to operate the system in real-life conditions. Instead, the system will utilise simulated network conditions with specific channel error patterns or traces inserted into transmitted FlexCode bit stream. The focus is to enable FlexCode functionality.

The real-time demonstrator architecture for both conversational and streaming use case scenarios is basically the same:

- The demonstrator will consists of two PCs connected to each other with network cable or wireless connection between the devices.
- The network simulation consisting of up and down link radio access and core network simulations is comprised of an error insertion device reading a specific channel error pattern or trace.
- Up- and downlink radio access and core network error files (channel traces) are combined to a single file. Hence, only one error insertion device is needed for each end-to-end transmission leg.

Figure 1 and 2 provide the real-time demonstration architecture. FlexCode clients and servers are running on separate PCs while the channel simulation functionality is embedded in the client.

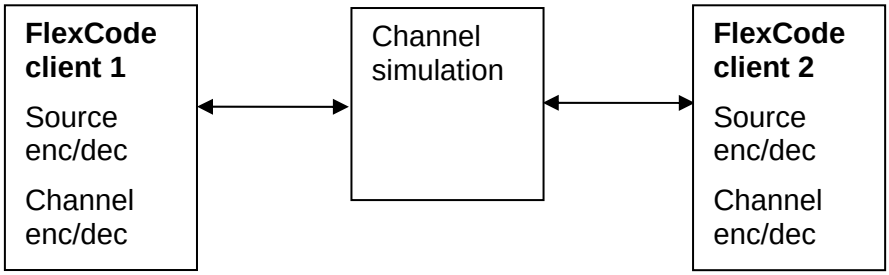

Figure 1 Conversational scenario demonstration consisting of two FlexCode clients.

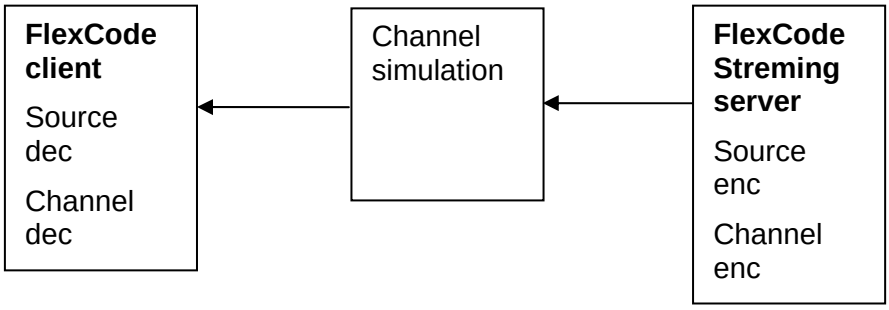

Figure 2 Streaming scenario demonstration consisting of FlexCode client and server.

2.2 Real-time demonstrator logical components

Real-time demonstration of the FlexCode system requires some specific issues of the source/channel codec implementations. Encoding and decoding functions need to operate on frame basis independently so that all inter-frame codec internal data, such as filter states, is stored and conveyed from frame to frame. For the same reason, global variables are not allowed since the client may run several encoder/decoder functions simultaneously.

Figure 3 describes the architecture for the FlexCode client and server. The client consists of the following components:

- Audio source: Input from the sound card (e.g. ALSA source), sampling rate (default 16 kHz) selected by the application/user, output audio stream
- Source encoder: Input from the audio source, audio input framing and encoding frame-by-frame basis, codec adaptation by the application/user, output bit stream or index stream.
- Channel encoder: Input from the source encoder, encoding frame-by-frame basis, adaptation by the application/user, output coded bit stream
- Network interface (TX): Input from the channel encoder, adaptation by the application, output UDP (optionally RTP) packets
- Network interface (RX): Input UDP (optionally RTP) packets, adaptation by the application, output coded bit stream
- Error insertion device: Removal or alteration of packets, insertion of bit errors according to chosen channel characteristics.
- Channel decoder: Input coded bit stream, decoding frame-by-frame basis, adaptation by the application/user, output bit stream or index stream.
- Source decoder: Input bit stream or index stream, decoding frame-by-frame basis, adaptation by the application/user, output audio stream

– Audio sink: Input audio stream, sampling rate (default 16 kHz) selected by the application/user, output to sound card (e.g. ALSA sink)

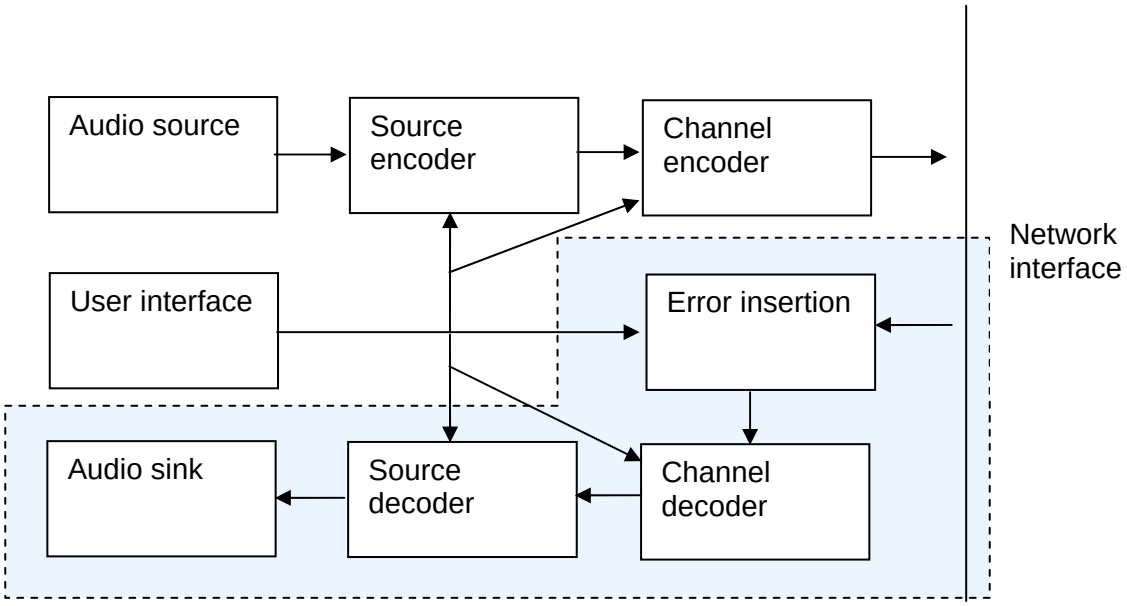

Figure 3 FlexCode client architecture, FlexCode server consists of all blocks outside the blue box.

The server consists of only the first 4 components in the above list.

2.3 Channel coder requirements

The channel encoder has knowledge about the overall available bit rate and the channel conditions. Combining this information it decides on the protection level of the bit stream and communicates the residual bit rate to the source encoder that configures itself accordingly.

The algorithms to decide on the protection levels and the required rates for these are developed in WP2. The channel conditions in terms of packet loss rates (possibly including late losses) or soft-information alterations are to be derived from the channel traces given in deliverable D3.3 “Database of transmission channel traces”. This work is ongoing in WP2.

The networks considered differ slightly for the different scenarios. For the Mobile Conversation Scenario (MCvS) the project agreed to consider 3G HSPA networks in the up- and down-link and a reliable core-network as transport network. For the Multimedia On Demand Streaming Scenario (MODSS) the project agreed to consider a 3G HSPA network for the downlink and the general Internet as a transport network. Another option that can be considered for the downlink is a digital subscriber line (DSL) link. The decision will be based on the outcome of the study of the relevant channel traces provided in D3.3.

2.4 Source coder requirements

The source encoder receives, besides the incoming audio signal, setup parameters including maximum allowed delay and target bit-rate. It configures itself accordingly and produces a bit-stream. Each block of the audio source is represented by a set of bits. One block can be divided into sub-blocks internally in the source encoder.

The requirements in terms of delay and rate are defined in deliverable D3.1 “Ordered List of Real World Service Scenarios” for the two scenarios.

For the MCvS the source coder rate is limited between 10 kbps and 32 kbps. The performance of other transform based coders, e.g., 3GPP enhanced AAC+ [3GPP TS26.401] as characterized in [3GPP TS26.936] suggests rates of at least 14 kbps for mono speech signals to be used. As stated in D3.1, the mouth-to-ear delay for conversational services should not exceed 300 ms. First analysis of the channel traces in D3.2 performed by WP2 suggest that

reasonable HSPA channel delays are in the order of 200 ms (see Figure 4). Allowing some additional delay for the A/D and D/A components of the simulation PCs the codec should not exhibit more than 25 ms to 40 ms delay.

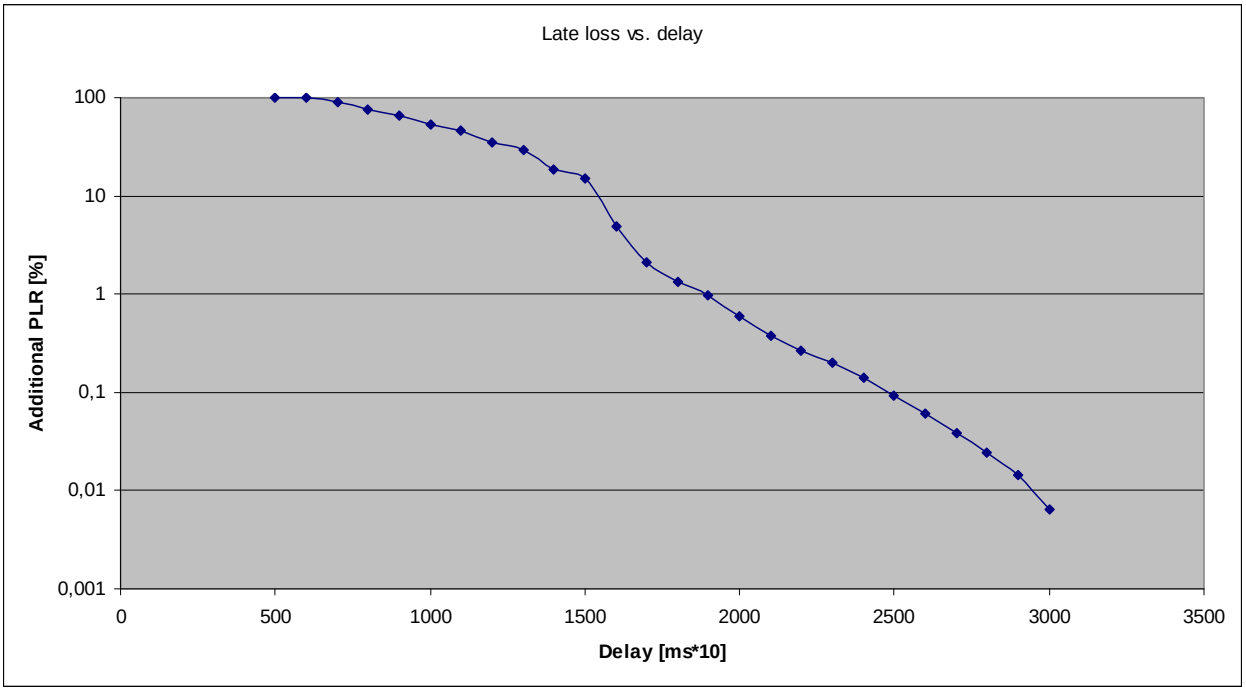

Figure 4 Additional packet loss rate caused by late loss as a function of maximum allowed delay.

For the MODSS D3.1 gives bit-rate ranges as a function of the signal bandwidth. So far the project has focused on wideband (8 kHz bandwidth) signals. Future versions of the codec should support bandwidths of up to 16 kHz. Thus, the bit rate range relevant for the MODSS is between 12 kbps and 56 kbps. Delay is not a major issue for the MODSS scenario. A jitter buffer that compensates for most of the late loss caused by the HSPA channel is of the order of 300 ms. A jitter buffer of this size can be considered in the MODSS scenario. However, additional late loss has to be considered for the general Internet transport network. The analysis of the corresponding traces is currently ongoing in WP2. For the MODSS we consider a maximum jitter buffer of about 1 second realistic. Limitations of the jitter buffer stem from the fact that their complexity and memory contribution should be allowable on mobile devices.

3 Methodology for Real-Channel Testing

The scope of the project is to develop a flexible coding scheme for heterogeneous networks. Therefore, it is important to evaluate the system performance in real channel conditions. The overall performance and adaptation functionality need to be tested in different conditions.

Typically speech and audio codecs are evaluated in various channel conditions using off-line simulations. For that purpose, the channel conditions need to be simulated as well. In practice, the application level source and channel codec sees the channel conditions as erroneous IP/UDP/RTP packets, residual bit errors or erroneous coded symbols. Therefore, error traces are typically sufficient to evaluate the source and channel codecs, and there is no need to run channel simulations or to utilise real world transmission channels.

3.1 User Interface and setup for the scenarios

In this section we describe some parameters and functionality of the user interface of the real-time demonstrator that allow running the FlexCode system according to the chosen scenarios.

MCvS

For the MCvS the two PCs accommodating the two clients have to be separated such that reasonable acoustic isolation is accomplished. The sound cards have to run in full duplex and the input to the source encoder comes from a microphone while the output is played via

headphones. In addition, the source encoder, channel encoder, source decoder, and channel decoder have to run in parallel on each client to allow a two way communication.

The user interface has to contain buttons to start and stop the communication session, choose the cross bit-rate and some pre-defined channel conditions. Optionally control elements to control individual channel parameters can be placed in the user interface.

MODSS

For the MODSS the communication is unidirectional. Thus, only the source encoder and channel encoder need to run on the server and only the source decoder and channel decoder need to run on the client side. In addition, the acoustic separation between server and client is not necessary.

The user interface has to contain controls to select different material and quality of service settings. In addition, pre-defined channel conditions should be selectable. Optionally control elements to control individual channel parameters and the individual conditions for the transport and downlink channels can be placed in the user interface.

The material should reflect the different categories defined in deliverable D3.2 “Database of input signals”, i.e., music, speech, and music mixed with speech. Corresponding material is available in the database of D3.2. For the quality of service possible choices can comprise, e.g., pre-view, normal service, premium service. The exact configurations for these quality-of-service levels in terms of rate have to be found once the project has a better understanding of the exact performance of the FlexCode system.

3.2 Setup for formal subjective quality evaluation

The exact procedure for formal quality evaluation is to be defined within WP5 in milestone 5.1. Corresponding documents are available within the project. From these it is clear that the formal quality evaluation will follow procedures defined in, e.g., ITU-T P.800. Thus, for the formal evaluation tests the demonstrator can not be run in the way described in Section 3.1. Instead, for each scenario a defined subset of demonstrator outputs has to be recorded and used throughout the formal tests.

For the MODSS the material selectable in the real-time demonstrator does coincide with material in the database of D3.2. Thus, recorded output from the above described setup of the demonstrator can be used straight in the subjective tests.

For the MCvS it is beneficial to use some material from the database of D3.2 instead of using a recorded life communication session. Thus, to gather material for the formal subjective evaluation for the MCvS one additional change to the demonstrator setup has to be made: The microphone input to the source encoder has to be replaced by input from the database accompanying D3.2. In D3.2 the material meaningful for the MCvS is identified.