Escola Politècnica Superior de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE DE FI DE CARRERA

TÍTOL DEL PFC: Estudi sobre la percepció de QoS en un demostrador UMTS (Part I)

TITULACIÓ: Enginyeria de Telecomunicació (segon cicle)

AUTORS: Luis Domínguez González
Iñaki Quer Alecha

DIRECTOR: Ramon Ferrús Ferré

DATA: 20 de gener de 2005

Títol: Estudi sobre la percepció de QoS en un demostrador UMTS (Part I)

Autors: Luis Domínguez González
Iñaki Quer Alecha

Director: Ramon Ferrús Ferré

Data: 20 de gener de 2005

Resum

Aquest projecte s'inclou dins d'un conjunt que té com objectiu desenvolupar una plataforma d'emulació UMTS en temps real, per l'avaluació de la QoS d'aplicacions multimèdia, amb la utilització d'algoritmes d'administració del recurs ràdio.

La part en la que es centra aquest document és l'adaptació i provisió d'eines que permeten al simulador gestionar informació rebuda als extrems del sistema UMTS, és a dir, el que representa ser l'equip terminal i la xarxa principal.

Una part essencial del projecte consisteix en la integració del còdec AMR (Adaptative Multi Rate Còdec), còdec de veu obligatori pels Sistemes de Tercera Generació, dins d'una aplicació d'àudio per conferència i streaming denominada RAT. Amb l'objectiu de tenir una eina que permeti analitzar qualsevol aplicació de xarxa hem desenvolupat un grup de programes que obtenen informació rellevant per caracteritzar-los.

Per mesurar la qualitat de l'entorn de comunicacions s'ha utilitzat el PESQ, una eina normalitzada per la ITU que obté una nota MOS (Mean Opinion Score) que dóna idea de la distorsió del canal que se l'introdueixen als senyals. Per evitar efectes de retard derivats del procés de mesura, que no són considerats pel PESQ, hem desenvolupat un sincronitzador d'arxius .WAV que permet obtenir mesures més reals quan es comparen arxius d'àudio del RAT.

Title: Estudi sobre la percepció de QoS en un demostrador UMTS (Part I)

Authors: Luis Domínguez González
Iñaki Quer Alecha

Director: Ramon Ferrús Ferré

Date: Gener, 20th 2005

Overview

This project is included in a set which have the objective to develop an UMTS emulation platform in real time, for QoS performance evaluation of multimedia applications, with the use of radio resource management algorithms.

The part that this document is centered is in adapting and supplying the simulator with tools that allow it to manage information recibed at the ends of the UMTS system, that is, what represents to be the terminal equipment and the core network.

One essential part of the project consists in the integration of AMR (Adaptative Multi Rate Codec), mandatory speech codec for Third Generation Systems, inside an application for audio conferencing and streaming called RAT. With the objective of having a tool that lets you analyze network applications we have developed a group of programs that obtain relevant information to characterize them.

In order to measure the quality of the comunications environment we make use of PESQ, a tool normalized by ITU that obtains a MOS (Mean Opinion Score) puntuation that gives idea of channel disturbance introduced to signals. To avoid delay effect result of test process, that isn't considered by PESQ, we have developed a .WAV files synchronizer that helps to get more real measures when comparing RAT audio files.

ÍNDEX

INTRODUCCIÓ 6

1. DESENVOLUPAMENT D'UNA EINA PER L'ANÀLISI D'APLICACIONS DE XARXA 7

1.1. Introducció 7
1.1.1. Generador de Gràfics 7
1.1.2. Anàlisi de captures 7
1.1.3. Necessitat 8
1.1.4. Elecció dels Còdecs 8

1.2. Implementació de l'eina 8
1.2.1. Importar Captura CSV 9
1.2.2. Distribució dels bytes en el temps 10
1.2.3. Distribució dels paquets en el temps 12
1.2.4. Distribució del temps entre paquets 13
1.2.5. Distribució del tamany dels paquets 14
1.2.6. Velocitats instantànies 15
1.2.7. Estadística de protocols 16
1.2.8. Elecció del tipus de les gràfiques 17
1.2.9. Script ExportaCSV 17

1.3. Caracterització del RAT mitjançant l'aplicació 18
1.3.1. Robust Audio Tool (RAT) 18
1.3.2. Caracterització 23
1.3.3. Còdec GSM 23
a) Distribució dels bytes en el temps 23
b) Distribució dels paquets en el temps 24
c) Distribució del temps entre paquets 25
d) Distribució del tamany dels paquets 27
e) Velocitat instantània 29
e1) Distribució de velocitats 30
f) Estadístiques de protocols 31
1.3.4. Còdec AMR 33
a) Distribució dels bytes en el temps 33
b) Distribució dels paquets en el temps 35
c) Distribució del temps entre paquets 36
d) Distribució del tamany dels paquets 37
e) Velocitats instantànies 38
f) Distribució de velocitats instantànies 39
g) Estadístiques de protocols 40

2. ENTORN DE PROVES AMB EL PESQ 41

2.1. INTRODUCCIÓ 41
2.1.1. PESQ (Perceptual Evaluation of Speech Quality) 41
2.1.2. Necessitat 43
2.1.3. Sincronitzador d'arxius "Syncok" 43
a) Funcionament 43
b) Característiques 45
c) Verificació del funcionament 45

2.2. CARACTERITZACIÓ DE L'AMR MITJANÇANT PESQ..... 46
2.1.4. Explicació..... 47
2.1.5. Còdec AMR 47
2.1.6. Caracterització..... 48

3. INTEGRACIÓ DEL CÒDEC AMR EN L'APLICACIÓ RAT 53

3.1. Introducció 53

3.2. Integració dels diferents modes del AMR dins el RAT 53

3.3. Automatització de l'elecció del còdec segons les condicions del canal (Canvi del mode de forma externa)..... 57

4. CONCLUSIONS, ESTUDI D'AMBIENTALITZACIÓ I FUTURES MILLORES POSSIBLES 58

4.1. Conclusions 58

4.2. Estudi d'ambientalització 59

4.3. Futures millores possibles 59

5. BIBLIOGRAFÍA 60

6. ANNEXES 63

6.1. CAPÍTOL 1 63
6.1.1. ANNEX 1 "Codi Eina EXCEL" 63
6.1.2. ANNEX 2 "Codi del script exportacsv" 95
6.1.3. ANNEX 3 "Codi C de l'aplicació *passapaket* "..... 99

6.2. CAPÍTOL 2 101
6.2.1. ANNEX 4 "Codi syncok " 101

6.3. CAPÍTOL 3 106
6.3.1. ANNEX 5 "Codi de la integració del AMR al RAT" 106
a) Codec_amr.c 106
b) Codec.c 113
c) transmit.c 117
d) cx_amr.c 121
e) codec_amr2.h 171
f) cx_amr.h 172
g) Makefile 176

INTRODUCCIÓ

Les xarxes mòbils de tercera generació seran molt més complexes que les existents actualment per segona generació. Per tant és de vital importància el realitzar un estudi que permeti predir les necessites de les aplicacions en quan a ample de banda i els paràmetres més òptims per oferir una QoS adequada al canal de comunicació.

Aquest projecte s'engloba dins d'un conjunt on es vol desenvolupar una plataforma d'emulació d'UMTS en temps real per poder avaluar rendiments en quan a qualitat de servei extrem a extrem d'aplicacions multimèdia IP es refereix, mitjançant la utilització d'algoritmes de gestió del recurs ràdio.

El entorn sobre el que es treballarà serà un cluster de servidors com el que es mostra a continuació:

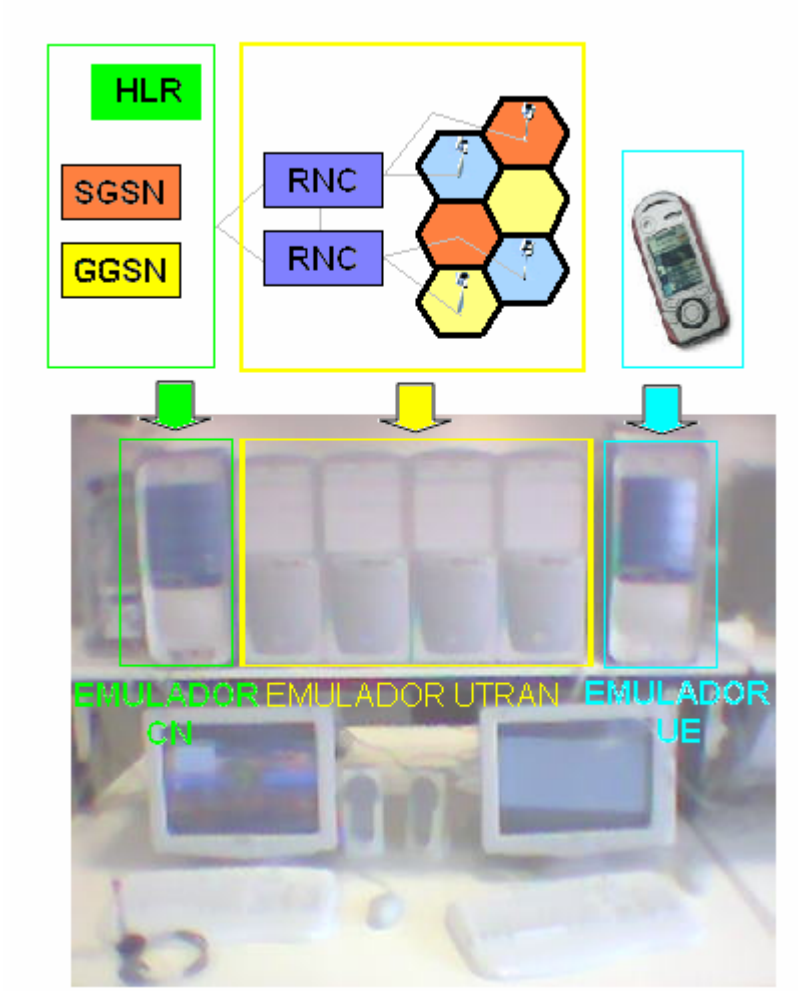


Fig. A Entorn de treball

En concret aquest projecte s'encarrega de la integració del còdec de veu per UMTS AMR dins de l'aplicació d'àudio per videoconferència RAT. Aquest còdec té la peculiaritat de que permet que la velocitat de transmissió varia adaptant-se a les condicions del canal de comunicació. Per poder caracteritzar l'aplicació RAT s'ha implementat una eina que permet obtenir diferent tipus de gràfiques amb les que es pot caracteritzar qualsevol aplicació de xarxa. Per comprovar els valors de QoS que s'obtenen per cadascú dels modes o velocitats diferents del AMR s'ha utilitzat l'aplicació PESQ.

El projecte s'ha estructurat en quatre parts principals que són: el desenvolupament de l'eina per l'anàlisi d'aplicacions de xarxa i l'ús que s'ha fet d'ella per caracteritzar un dels instruments que fem servir al llarg del projecte com és el RAT, l'entorn de proves que s'han realitzat mitjançant l'aplicació PESQ, la pròpia integració que s'ha fet del còdec AMR dins del RAT i un seguit de conclusions i expectatives futures que hem extret al llarg del desenvolupament del projecte.

1. DESENVOLUPAMENT D'UNA EINA PER L'ANÀLISI D'APLICACIONS DE XARXA

1.1. Introducció

Amb l'objectiu de poder analitzar el comportament de l'aplicació d'àudio RAT, tal com els protocols que utilitza, amb quina freqüència s'envien els paquets, de quin tamany és la informació que es transmet, etc. s'ha desenvolupat una aplicació anomenada *"Generador de Gràfics"* pel seu anàlisi i que a la vegada és de caràcter general, ja que aquesta pot ésser utilitzada per la caracterització de qualsevol aplicació de xarxa.

1.1.1. Generador de Gràfics

"Generador de Gràfics", és doncs una aplicació feta amb Macros programades en Visual Basic en l'Excel. La qual ens permet realitzar múltiples gràfiques d'interès, mitjançant les captures de xarxa importades en format CSV.

Primer es realitza la captura amb un sniffer (ja sigui via software o hardware), després s'exporta aquesta al format CSV (mitjançant un Script en Shell, que es detalla més endavant) i finalment es passa a l'aplicació "Generador de Gràfics"; per visualitzar les dades obtingudes, i poder fer un anàlisi de les mateixes.

1.1.2. Anàlisi de captures

L'anàlisi es pot fer amb diferents tipus de gràfiques:

a) Distribució de bytes en el temps
b) Distribució dels paquets en el temps
c) Distribució del temps entre paquets
d) Distribució del tamany dels paquets
e) Velocitat instantània
    e1) Distribució de velocitats instantànies
f) Estadística de protocols

A on hi ha la possibilitat de filtrar per IP origen, IP destí, MAC origen, MAC destí, port origen, port destí, agrupar els ports, desglossar els ports, per protocol, canvi de resolució, zoom espacial (definint inici i fi) i canvi de l'eix Y.

A més, en "Estadística de protocols" es pot filtrar per protocols de Xarxa, Aplicació i Transport. També en "Velocitats instantànies" i "Distribució de velocitats instantànies" es poden suavitzar les gràfiques.

1.1.3. Necessitat

Un dels altres motius per al que s'ha implementat aquesta aplicació, és degut a que anteriorment es feien els diferents anàlisis d'una eina de forma manual. Cosa que endarreria l'estudi i només servia per aquell cas en concret; d'aquesta forma, es pot aplicar a diferents casos.

Per exemple en l'Ethereal, si es volia filtrar algun protocol/port/direcció etc. s'havien d'estar configurant els filtres per a les captures pertinents. Mitjançant el *"Generador de Gràfics"* queda tot automatitzat i sempre amb possibilitat d'ampliar-se. A més també permet més flexibilitat a l'hora de poder escollir els rangs d'interès per a una aplicació en concret, adequant-los al que ens interessi mirar en cada moment.

1.1.4. Elecció dels Còdecs

La caracterització de l'aplicació RAT s'ha realitzat utilitzant el còdec AMR que s'ha aconseguit integrar a la mateixa i que es interès del nostre estudi. A més s'inclou la caracterització prèvia a la integració que s'havia fet mitjançant un dels còdecs més utilitzats com és el GSM,.

1.2. Implementació de l'eina

En aquest apartat, descriurem el funcionament de cada una de las parts que formen l'aplicació "Generador de Gràfics".

Per a poder tenir una visió més global de l'aplicació, en la següent *figura 1.1* es mostra el menú principal d'aquesta.

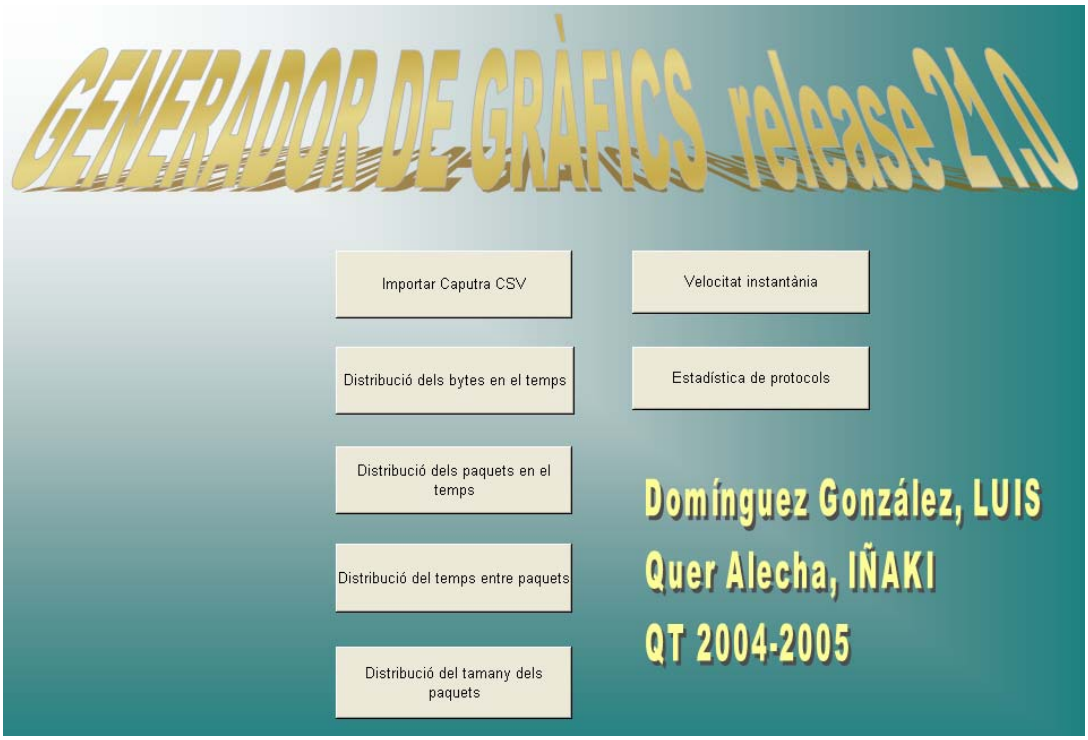


Fig. 1.1 Pàgina principal de "Generador de gràfics"

1.2.1. Importar Captura CSV

El primer que fa l'algoritme mitjançant una finestra modal, és demanar la ruta a on està desat l'arxiu CSV a importar (tal i com es veu en la figura que segueix).

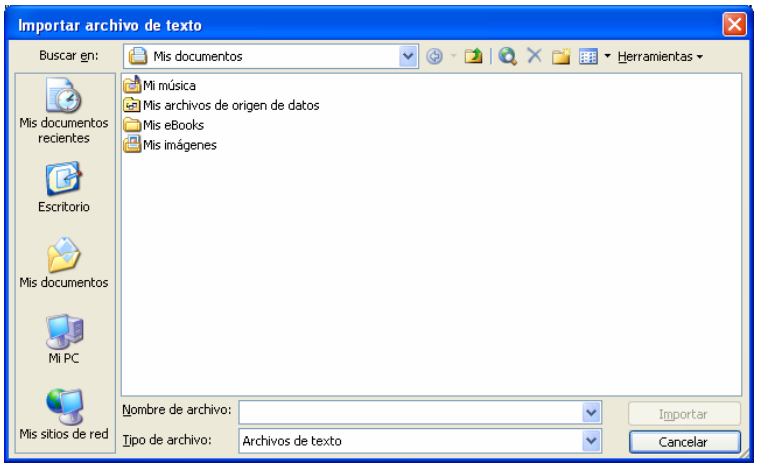


Fig. 1.2 Importació fitxer CSV

Tot seguit, l'aplicació obre el full de càlcul "Dades Totals" i introdueix les dades importades. Si aquestes s'han pogut guardar de forma correcta, es mostra un missatge com el que segueix a continuació:

Fig. 1.3 Missatge de sortida

1.2.2. Distribució dels bytes en el temps

Per poder fer els càlculs pertinents, s'utilitza el full "Càlculs". Inicialment l'algoritme, elimina les dades que poguessin haver-hi d'anteriors casos i tot seguit introdueix l'escala temporal, d'aquesta nova distribució a analitzar.

A continuació es mostra a l'usuari (tal com es veu a la *figura 1.4*) els valors inicial i final del rang calculat.

Fig. 1.4 Missatge d'escala

Un cop l'usuari ha saltat la primera pantalla d'informació, es pregunta el valor inicial per el qual es començarà la representació temporal (tal i com es mostra en la *figura 1.5*); fins que no s'introdueixi aquest valor, no es passa de pantalla.

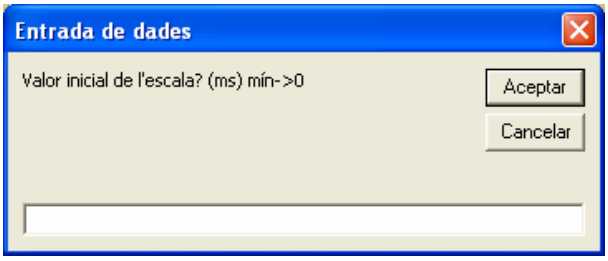


Fig. 1.5 Missatge d'inici d'escala

Després que s'hagi definit el valor inicial, seguidament es pregunta pel final a representar en l'escala temporal (tal i com es mostra en la *figura 1.6*).

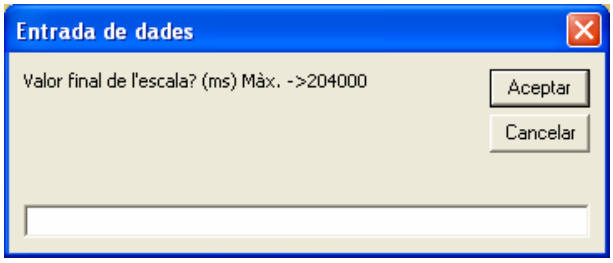


Fig. 1.6 Missatge de final d'escala

Per últim, es demana la resolució amb la que es generarà l'escala temporal (tal i com es pot apreciar en la *figura 1.7*).

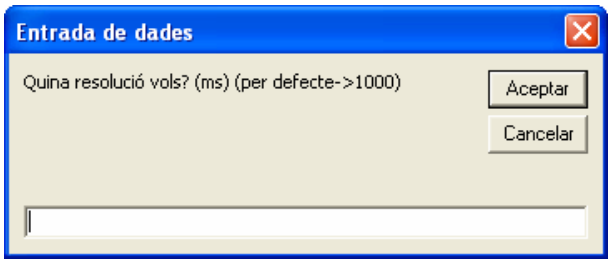


Fig. 1.7 Missatge de resolució per l'escala

Un cop es tenen tots els valors necessaris, l'algoritme comença a calcular la distribució segons els paràmetres passats.

Per fer-ho, aquest genera en un principi l'escala de valor inicial i final establerta per l'usuari, ja que no interessa fer el càlcul de tota la distribució sinó de la finestra escollida.

Després de calcular l'escala amb la resolució definida, l'algoritme agafa els valors existents en les dades importades i els aproxima als més propers dins de l'escala.

Quan ja s'ha acabat de fer tota la distribució temporal en l'escala, s'actualitza el full de càlcul utilitzat per a la representació i es mostra la gràfica seleccionada (en aquest cas "Distribució dels bytes en el temps").

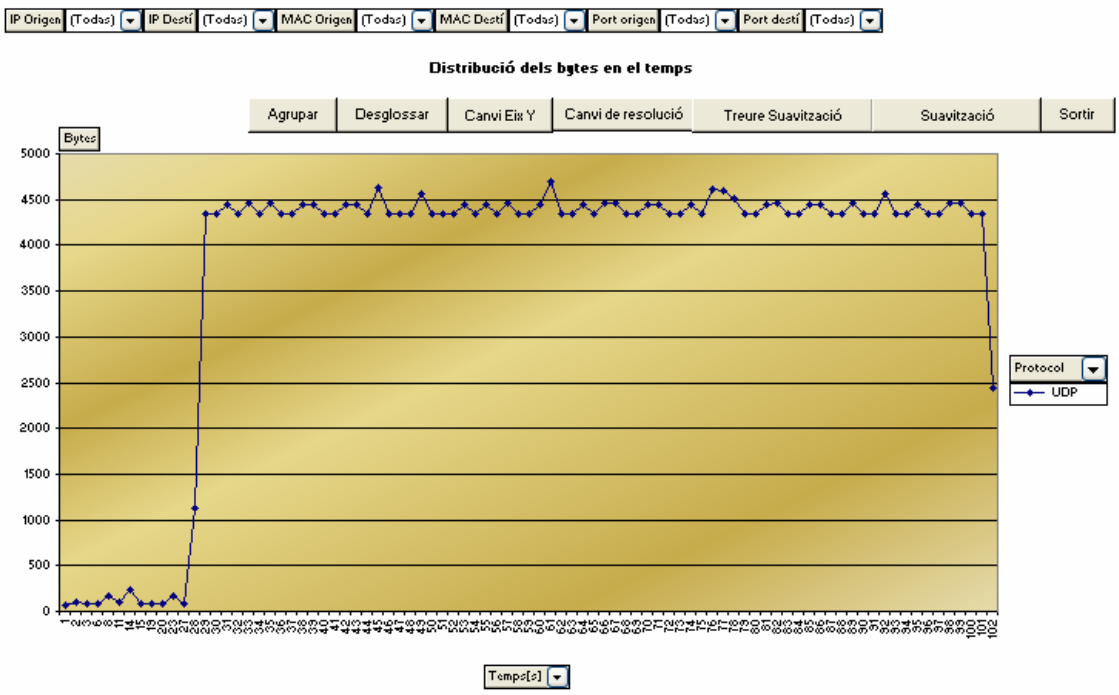


Fig 1.8 Exemple de distribució dels bytes en el temps

1.2.3. Distribució dels paquets en el temps

El procés que és segueix, és el mateix que en l'apartat anterior. El que canvia és que en aquest cas, es fa una representació del número de paquets que hi ha en l'escala temporal amb la resolució escollida.

Novament un cop realitzades les operacions, s'actualitza un altre full de càlculs (per no interferir amb les altres dades que s'estan calculant) per a la representació gràfica.

En la *figura 1.9,* podem veure un exemple d'aquesta distribució:

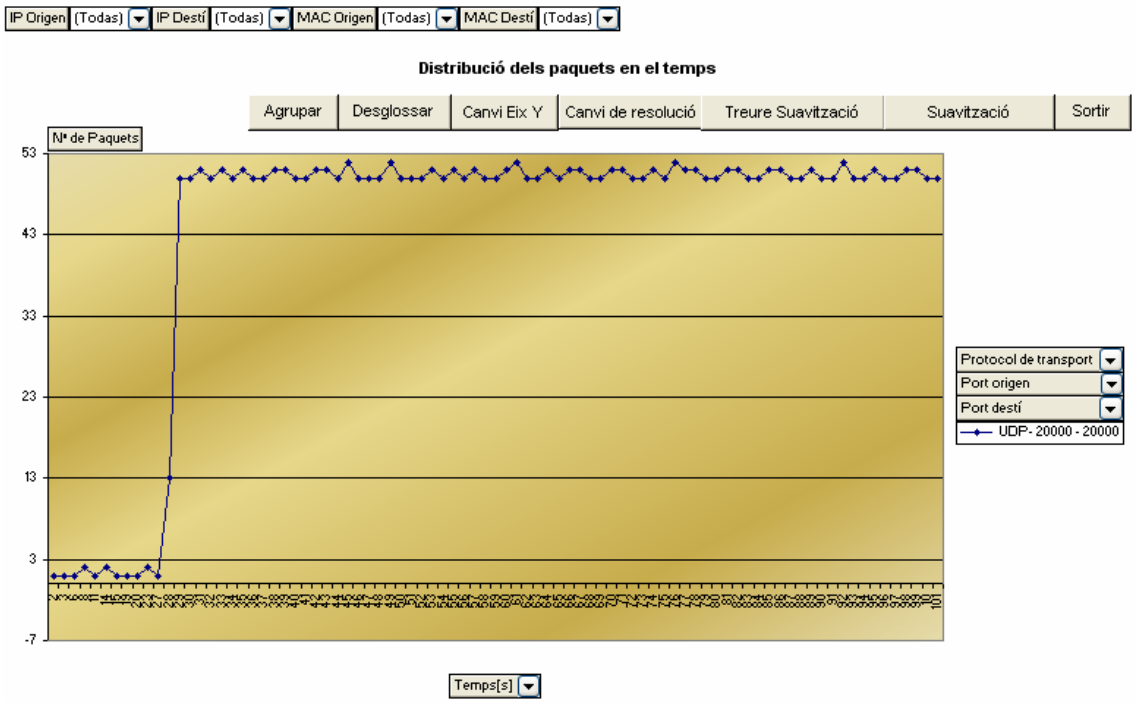


Fig 1.9 Exemple de distribució dels paquets en el temps

1.2.4. Distribució del temps entre paquets

Com en els dos casos anteriors es demana a l'usuari un valor inicial, final i una resolució per a generar l'escala temporal. En aquesta, es representa la distribució del retard entre els diferents paquets que han arribat.

L'algoritme en aquest procés, torna a fer ús d'un altre full de càlcul per als càlculs pertinents i també per actualitzar les dades a representar.

A continuació s'ofereix un exemple (*figura 1.10)* d'aquesta distribució:

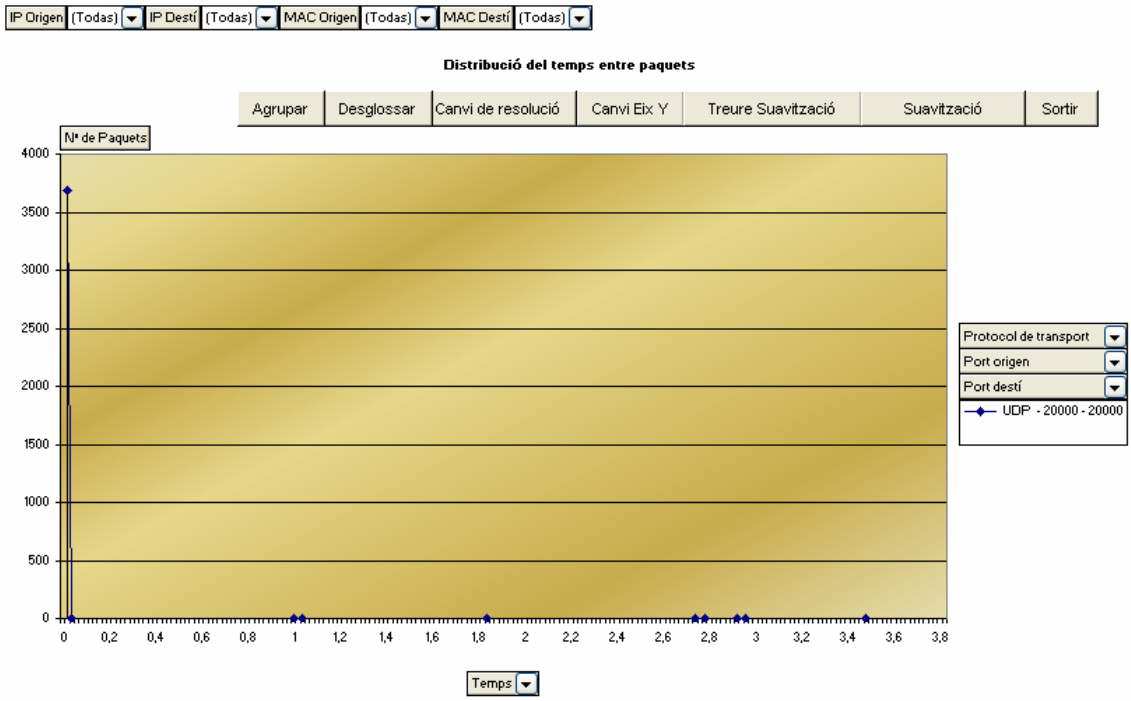


Fig. 1.10 Exemple de distribució del temps entre paquets

1.2.5. Distribució del tamany dels paquets

En aquesta distribució es reparteix el nombre de paquets que hi ha amb el mateix valor o amb el tamany més pròxim; segons el rang que defineix l'usuari, amb el seu valor inicial, final i resolució a representar.

En la figura que segueix, podem veure un exemple d'aquest tipus de distribució:

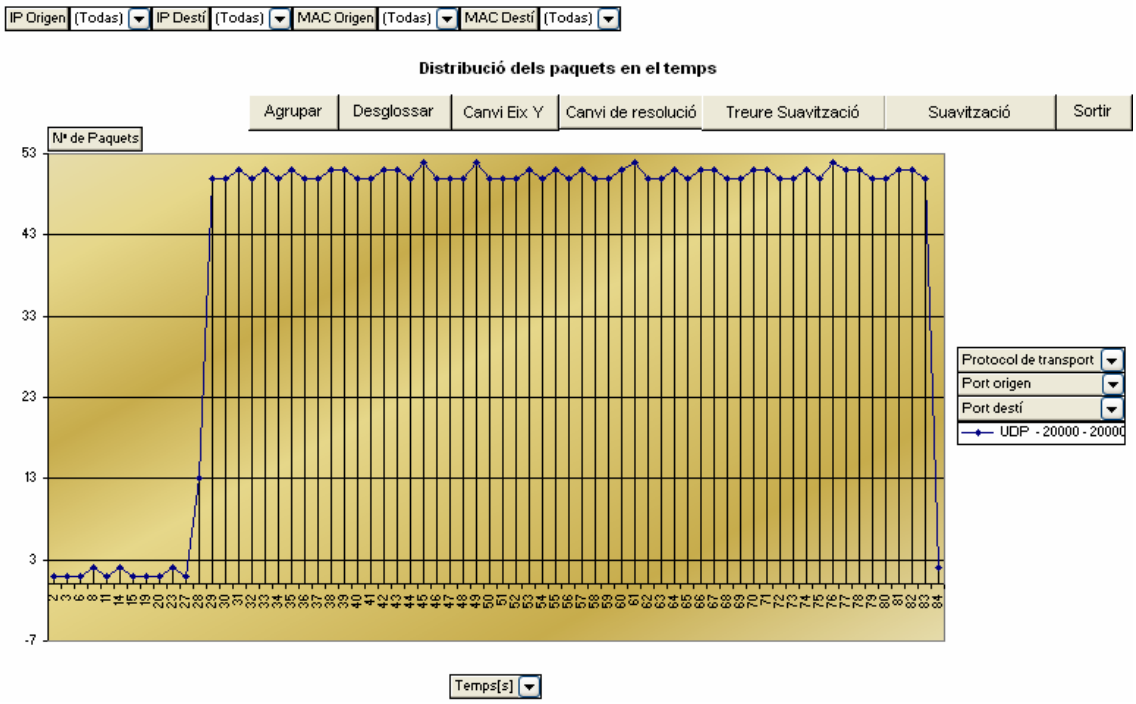


Fig. 1.11 Exemple d'una distribució del tamany dels paquets

1.2.6. Velocitats instantànies

En aquest esquema, inicialment també es mostra el rang de valors que hi ha. Seguidament es demana a l'usuari un valor inicial, final i es passa a la seva representació.

Perquè només estem tractant amb velocitats instantànies (hi ha una sola llegenda) i en els altres casos esmentats, podem tenir diferents tipus de dades (en la majoria de vegades, no hi ha el mateix protocol a representar).

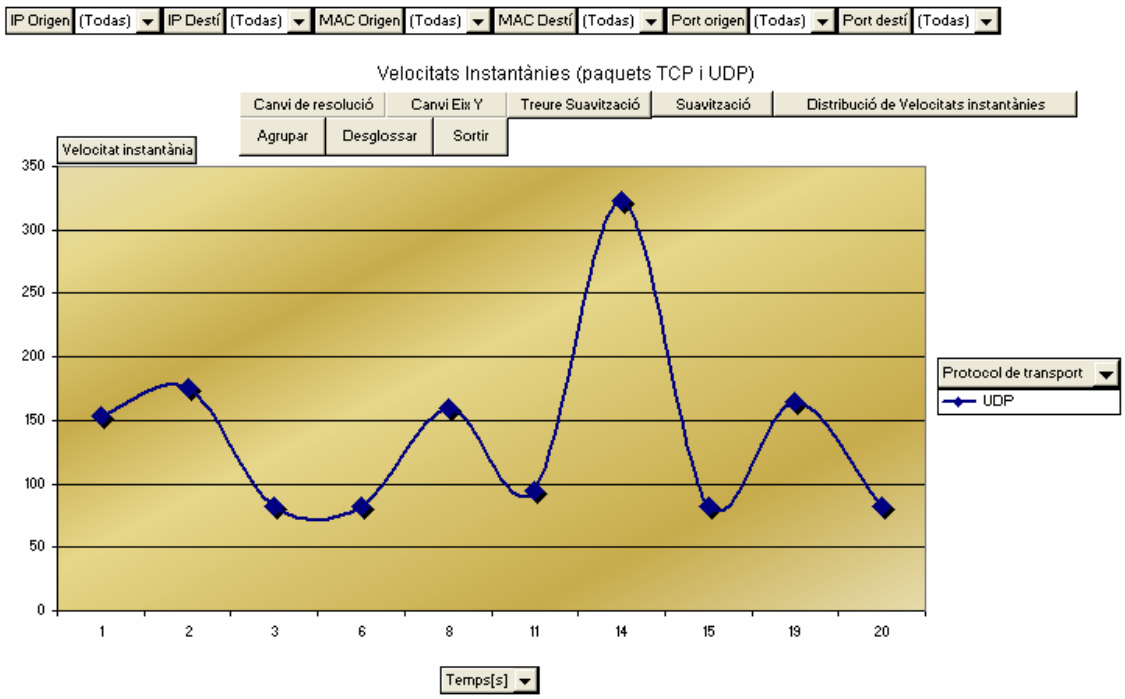


Fig. 1.12 Exemple de representació de velocitats instantànies

Tal com es pot apreciar en la *figura 1.12*, hi ha la possibilitat de mostrar una distribució de velocitats instantànies i que en aquest cas ja no hi haurà un eix temporal, sinó que les ordenades estaran representades per Velocitats instantànies/s.

1.2.7. Estadística de protocols

Aquest tipus de gràfica, permet realitzar diferents estadístiques en funció del tipus de protocol que volem analitzar.

Inicialment a l'usuari se li presenten els protocols de transport, que hi han hagut en la captura importada (*figura 1.13*).

Com s'ha comentat en la introducció, existeix la possibilitat de poder filtrar per "protocols d'aplicació", "protocols de xarxa" o "protocols de transport" tal i com es mostra en la figura que segueix:

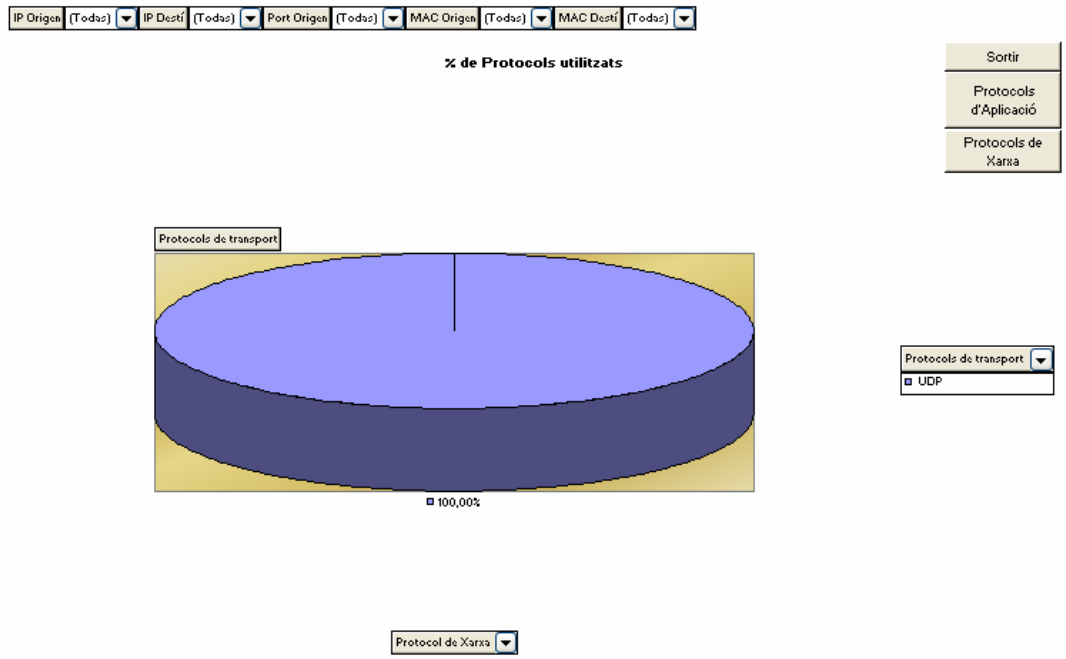


Fig. 1.13 Exemple de % de protocols utilitzats

1.2.8. Elecció del tipus de les gràfiques

Els tipus de gràfiques que s'han escollit per ésser implementades en l'eina "Generador de Gràfics", són les que s'han considerat més adequades per caracteritzar una aplicació de xarxa. És a dir, poder visualitzar tan la distribució dels paquets/bytes al llarg del temps, com el retard entre cadascú d'aquests paquets, les velocitats instantànies i les estadístiques de protocols.

1.2.9. Script ExportaCSV

Per a poder passar les captures dels sniffers al format CSV, s'ha programat un Script en shell que ho permeti realitzar d'una manera còmoda en LINUX.

Aquest Script (el codi es detalla en l'annex 2) fa ús dels capturadors Tethereal i TcpDump. S'utilitzen analitzadors de xarxa tipus *"mode text"* perquè faciliten la implementació, d'un exportador al format CSV. D'aquesta forma, es pot completar la totalitat d'anàlisis que es desitja en l'aplicació *"Generador de Gràfics"*.

El primer que realitza l'Script, és demanar des d'on s'han de capturar els paquets (eth0, eth1 etc.); ja que cada PC no tindrà configurades la/es seva/es Ethernet/s de la mateixa forma.

Un cop s'ha introduït la Ethernet a capturar, es llança l'aplicació "Tethereal" i es comencen a capturar paquets fins que l'usuari ho detingui. Un cop s'ha aturat

la captura, es llança el TcpDump i s'analitzen els protocols dels quals mitjançant el Tethereal no s'ha pogut extreure informació (com potser RTP, RTCP, ICMP etc.)

Després d'aquest anàlisi preliminar, mitjançant l'aplicació *"passapaket"* (veure codi C en l'annex 3) es retorna la informació del paquet capturat que interessa i es determinen les següents informacions:

a) Frames
b) Tamanys de paquet
c) Data d'arribada de paquet
d) Segons d'arribada
e) MAC Origen
f) MAC Destí
g) Protocol de transport
h) Protocol de xarxa
i) IP Origen
j) IP Destí
k) Port Origen
l) Port Destí
m) Hora d'arribada
n) Minuts d'arribada

Per últim, es guarden totes les dades en un fitxer amb format CSV.

1.3. Caracterització del RAT mitjançant l'aplicació

En aquest apartat, s'ha fet ús de l'aplicació desenvolupada *"Generador de Gràfics"* per a poder caracteritzar l'eina d'àudio RAT, que serà en la que s'ha integrat el còdec AMR.

1.3.1. Robust Audio Tool (RAT)

RAT és una aplicació de codi lliure que permet a usuaris participar en conferències d'àudio en temps real. Les conferències es poden establir entre dos participants directament o entre un grup multicast. Potser executada en plataformes diverses com FreeBSD, Linux, Solaris, SunOS, Windows 95 i NT, ...

El programa s'executa des de la línia de comandes passant-li com paràmetres la direcció IP amb la que es vol establir la comunicació i el port a través del que es farà.

rat *direcció_ip_conexió* / *port_comunicació*

Si es tracta d'una conferència multicast a "direcció_ip_conexió" s'haurà de posar una IP multicast.

La pantalla inicial del programa és la que es mostra a continuació:

Fig. 1.14 Presentació

Al panell superior es situen els botons que permeten activar/desactivar el micròfon i els altaveus en la comunicació. A més es pot visualitzar el nivell de senyal que s'està transmetin o rebent. També es pot controlar el volum dels altaveus, el guany del micròfon i es pot veure la velocitat a la que s'està enviant / rebent informació.

Al central es troben els participants de la conferència i a l'inferior es pot veure l'adreça IP connectada i el port pel que ho estàs. A més es situen el botons per poder reproduir o gravar els arxius de veu, veure la matriu de qualitat o modificar diferents opcions.

Per defecte el RAT presenta una sèrie de còdecs que poden ser utilitzats per a la transmissió de veu.

Dins de la pestanya del menú principal "Options" existeix l'opció "Codecs" que permet veure la informació corresponent a cadascú dels còdecs i permet mapejar l'identificador del payload RTP.

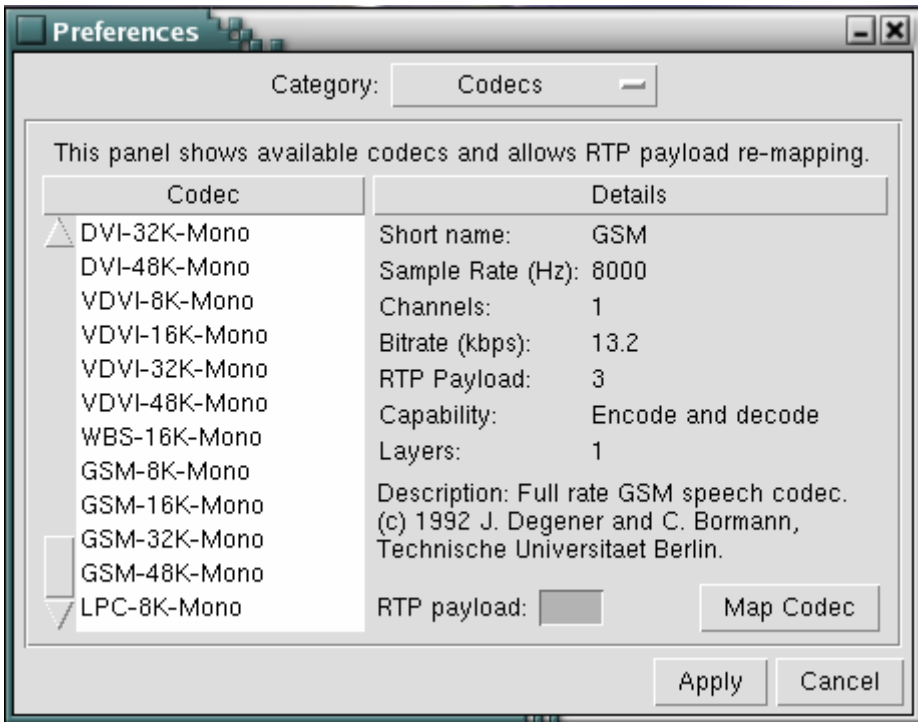


Fig. 1.15 Còdecs

Aquest altre panell (*figura 1.16)* permet seleccionar el còdec per la transmissió, així com si es vol redundància o no.

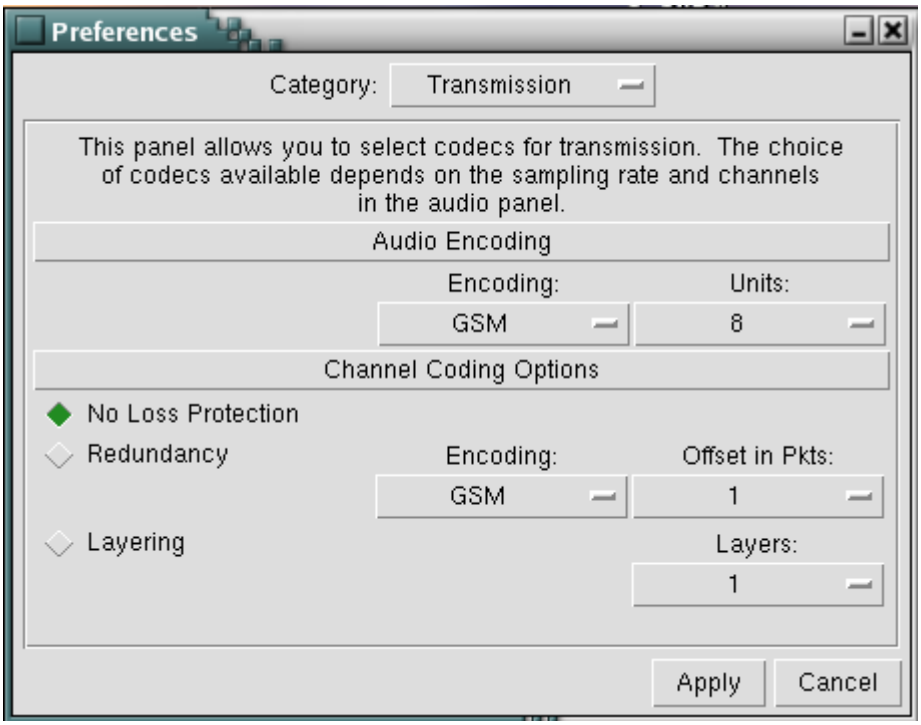


Fig. 1.16 Transmissió

A la pestanya àudio es pot seleccionar la freqüència de mostreig així com si es vol supressió de silenci o no.

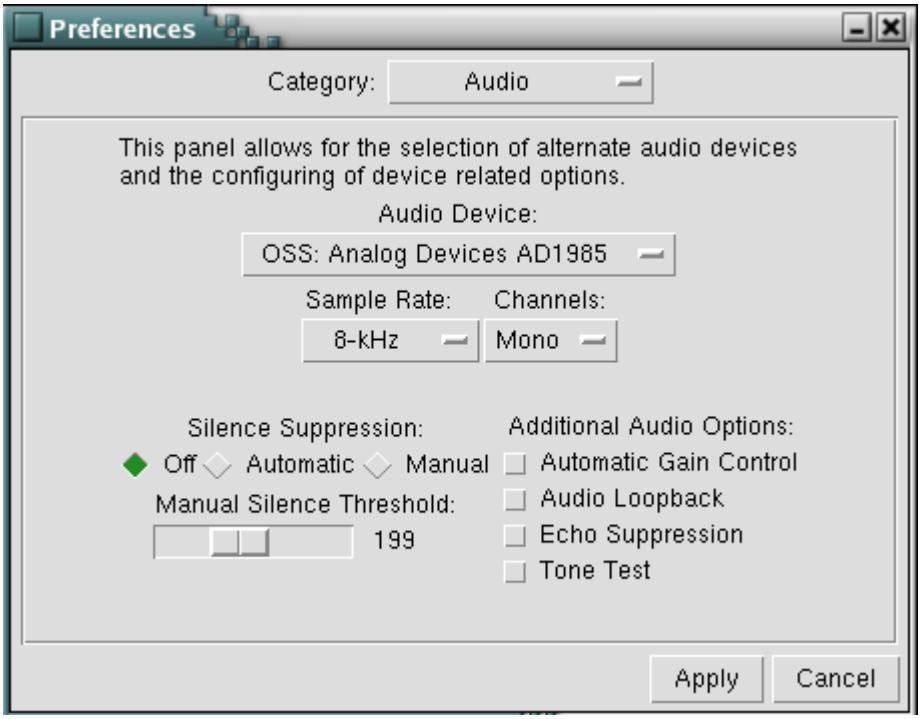


Fig. 1.17 Àudio

L'opció del menú principal "Reception quality matrix" permet observar la qualitat de la comunicació entre cadascú dels participants.

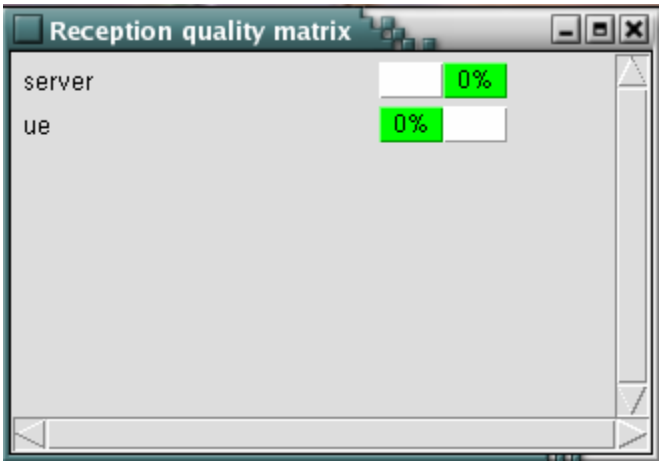


Fig. 1.18 Matriu de recepció de la qualitat

Mitjançant "RAT File Control" es pot reproduir un arxiu (Playback) o gravar a un arxiu la veu que es rep a la comunicació (Record).

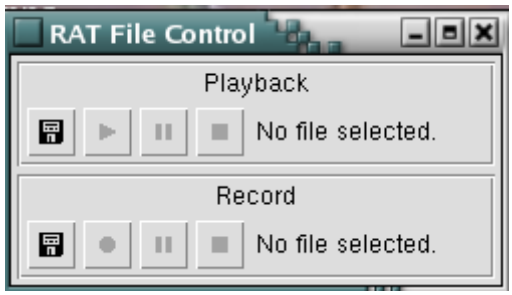


Fig. 1.19 Control d'arxius d'àudio

Seleccionant un usuari de la comunicació i clickant l'opció "Decoder" es pot veure informació sobre el còdec que s'està fent servir per transmetre la veu, els paquets que s'han perdut i els que s'han rebut.

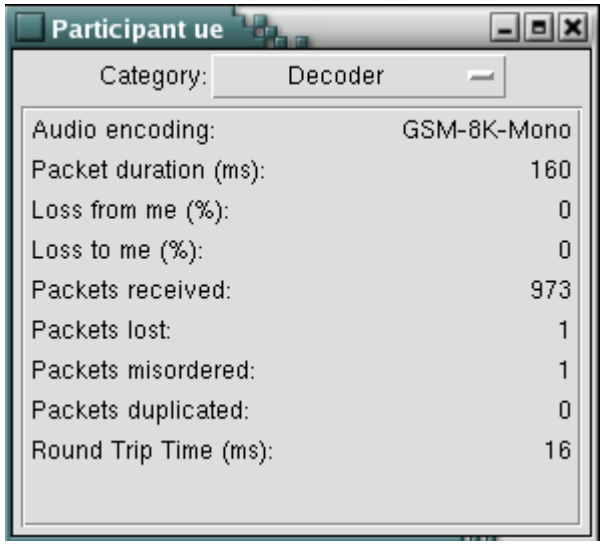


Fig. 1.16 Dades que s'han rebut

El RAT, no requereix característiques especials per comunicacions punt a punt; simplement una connexió de xarxa i una targeta de so. Com s'ha comentat, per multiconferències el RAT utilitza IP multicast i per tant, tots els participants d'aquesta multiconferència han de tenir habilitat el multicast. El RAT, està basat en estàndards IETF, utilitzant RTP sobre UDP/IP.

L'aplicació RAT funciona en les següents plataformes: FreeBSD, HP-UX, IRIX, Linux, NetBSD, Solaris, SunOS i Windows 95/NT.

En el nostre cas hem escollit LINUX (Fedora Core amb Kernel 2.6.7) perquè és un sistema molt estable, de lliure distribució i que està en expansió.

1.3.2. Caracterització

Un cop mencionada l'aplicació RAT, ja es pot passar a la seva caracterització. Aquesta s'ha fet a partir de l'anàlisi de dos còdecs: un ja integrat en la mateixa aplicació de veu, molt expandit i utilitzat en les xarxes actuals de mòbils de segona generació (GSM) i el que s'ha integrat en l'aplicació, que resulta ésser més representatiu per les xarxes de tercera generació (AMR).

1.3.3. Còdec GSM

Configurem en l'aplicació RAT el còdec GSM-8K-Mono i tot seguit es realitzen les captures entre els dos PCs, primer amb i desprès sense supressió de silenci.

Un cop importades les dades CSV a l'eina *"Generador de Gràfics"*, es passa a l'estudi de les mateixes:

a) Distribució dels bytes en el temps

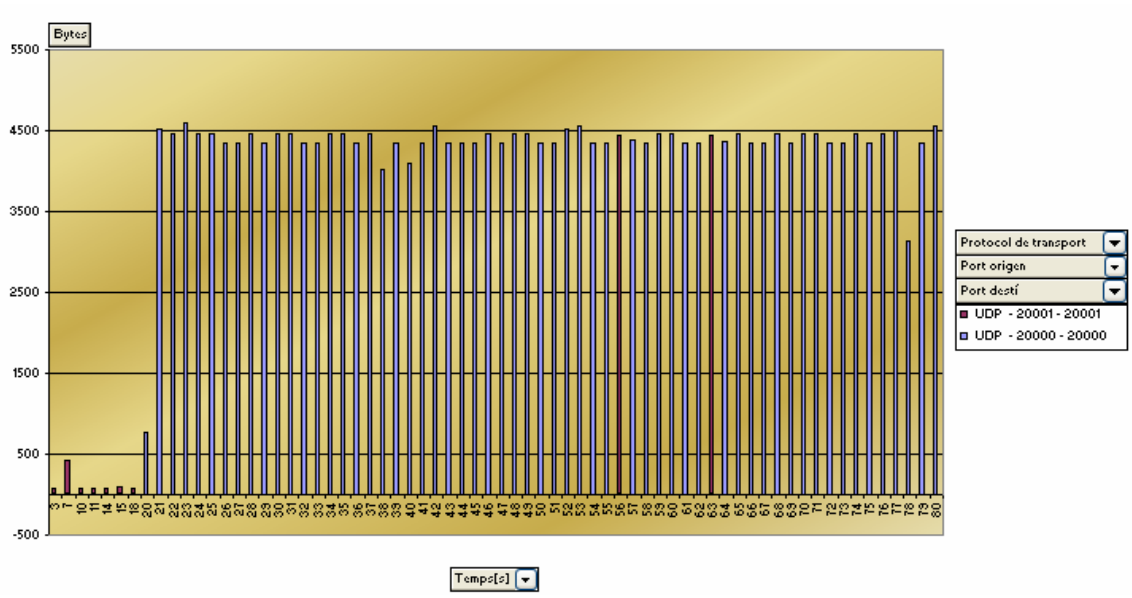


Fig. 6.14 Distribució bytes/temps amb supressió

En aquest cas l'aplicació ens indica que la duració de la captura ha estat en el marge de 0 a 80,79 segons. Per tant, triem aquest mateix marge per poder tenir una visió global del tràfic, que s'ha generat a la xarxa durant tot el temps de captura. Escollim una resolució d'1 segon, adequada per poder apreciar l'establiment de la sessió i la transmissió d'àudio.

Al començament de la captura es pot observar que els 20 primers segons d'inici de sessió corresponen a tràfic de tipus RTCP (UDP port 20001). Aquest temps és durant el que es fa el mapeig del còdec i s'estableix la comunicació. També es pot veure com el tràfic generat és dels voltants de 4500 bytes i que aquest es transmet de manera uniforme.

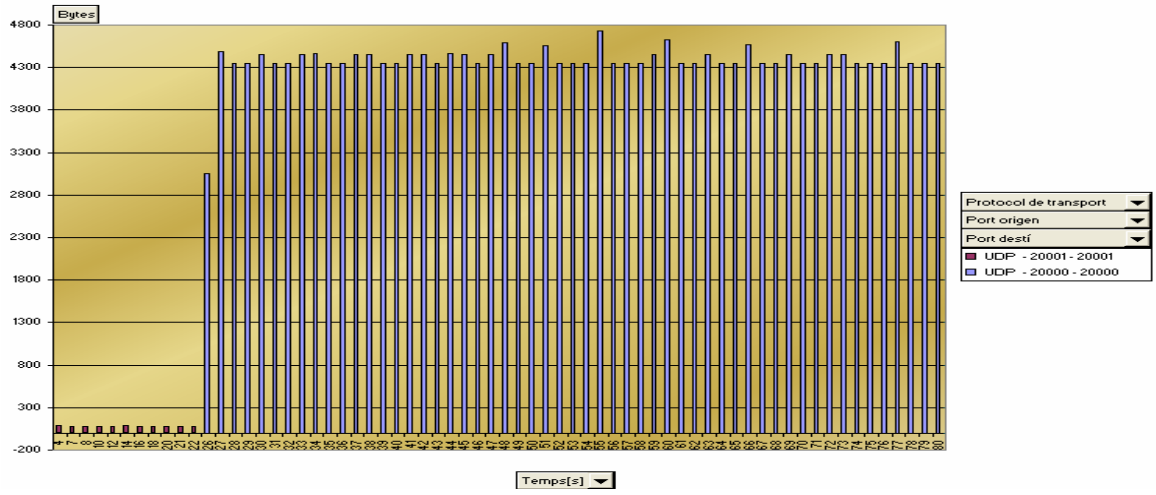


Fig. 1.15 Distribució bytes/temps sense supressió

Sense supressió de silenci es pot comprovar que en els instants de temps en els que no hi ha informació de veu, aquesta s'envia mitjançant tràfic RTP (UDP port 20000). Mentre que en el cas anterior aquest instants eren empleats per canals de control (protocol RTCP). Això, ens fa pensar que per aquest còdec hi ha un malbaratament dels recursos disponibles, ja que s'està fent ús d'ells com si es continués transmeten veu quan realment no s'està fent.

b) Distribució dels paquets en el temps

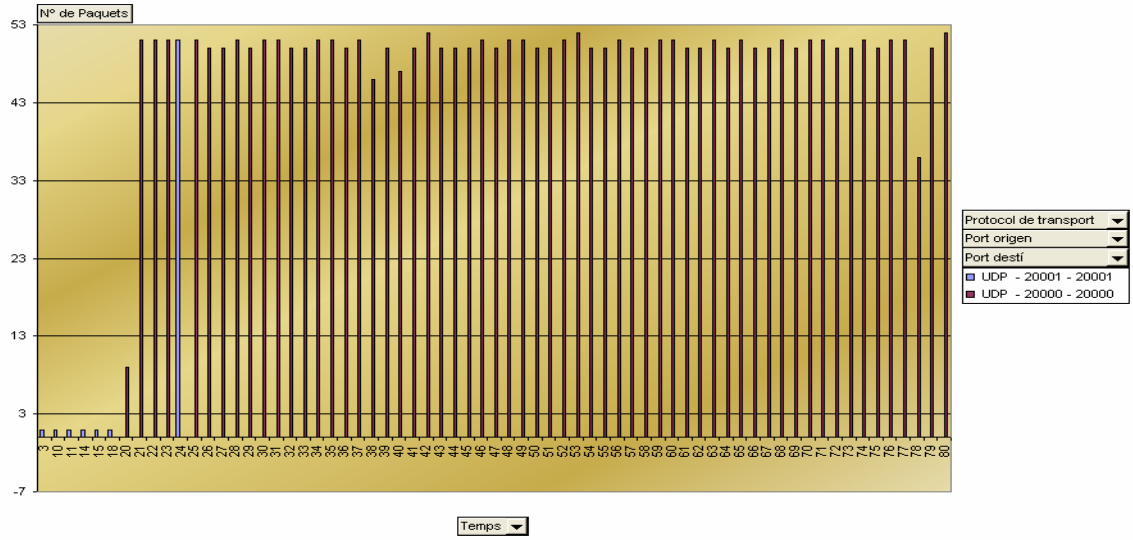


Fig. 1.16 Distribució paquets/temps amb supressió

Com en el cas anterior, l'aplicació segueix indicant-nos que la duració de la captura són de 80,79 segons; degut a que les dades segueixen sent les mateixes i només ha canviat la distribució a analitzar. La resolució triada és d'1 segon, perquè en aquest cas també ens interessa veure l'inici de sessió i el començament de la conferència de veu.

A diferència de l'anterior distribució, aquesta ens dóna una idea del nombre de paquets que es s'estan transmeten a cada instant. Es pot comprovar, que mentre hi ha transmissió d'àudio el nombre de paquets que s'envien és constant (aproximadament 50 paquets/segon). En canvi quan no, o si es transmet menys informació de veu el nombre de paquets baixa; per exemple instants 38, 40, 41, 78 segons ...

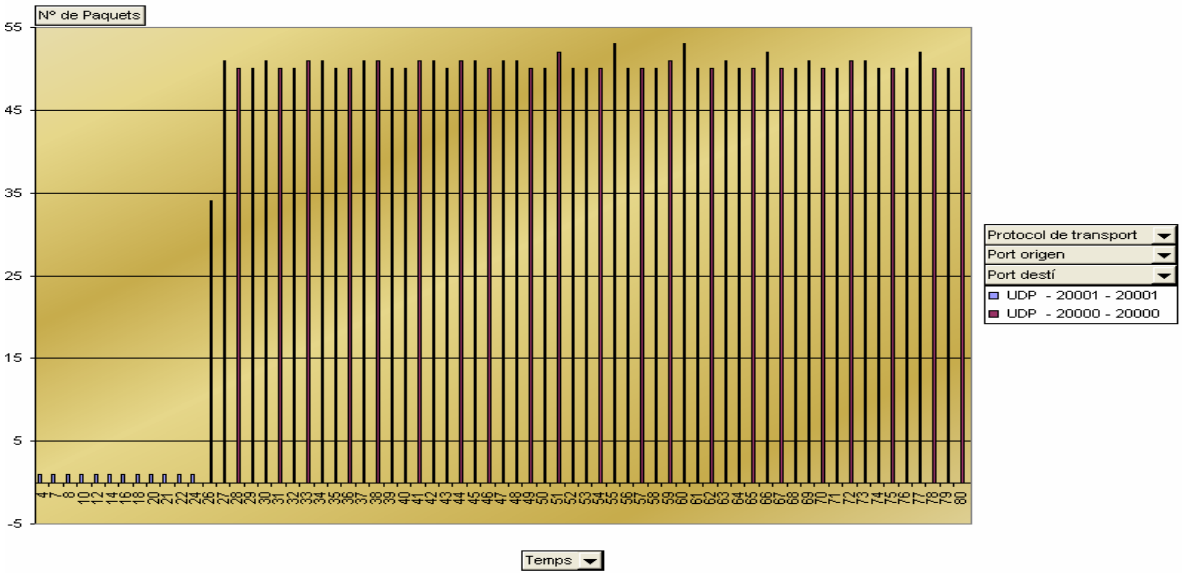


Fig. 1.17 Distribució paquets/temps sense supressió

Quan no està la supressió de silenci es pot veure que mentre hi ha transmissió d'àudio, el nombre de paquets que s'envien és també de l'ordre de 50 per segon i de forma uniforme, tan si el contingut de la informació que es transmet conté o no veu. Açò ens ve a dir que la separació temporal mínima que hi haurà entre un paquet i un altre serà de 20ms.

c) Distribució del temps entre paquets

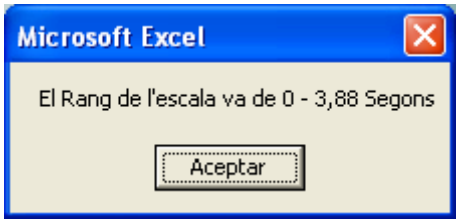


Fig. 1.18 Missatge informatiu

L'aplicació en aquest cas ens informa que el rang de retards entre els diferents paquets que composen la captura, van fins a un màxim de 3,88 segons. Segons la distribució anterior, s'ha comprovat que el valor mínim en la transmissió de paquets és de 20ms; per tan s'escull una finestra de valor inicial 0 segons, final 100ms, amb una resolució de 2ms i filtratge dels instants que no s'ha enviat informació. D'aquesta forma en aquesta distribució, tindrem el retard més probable entre paquets.

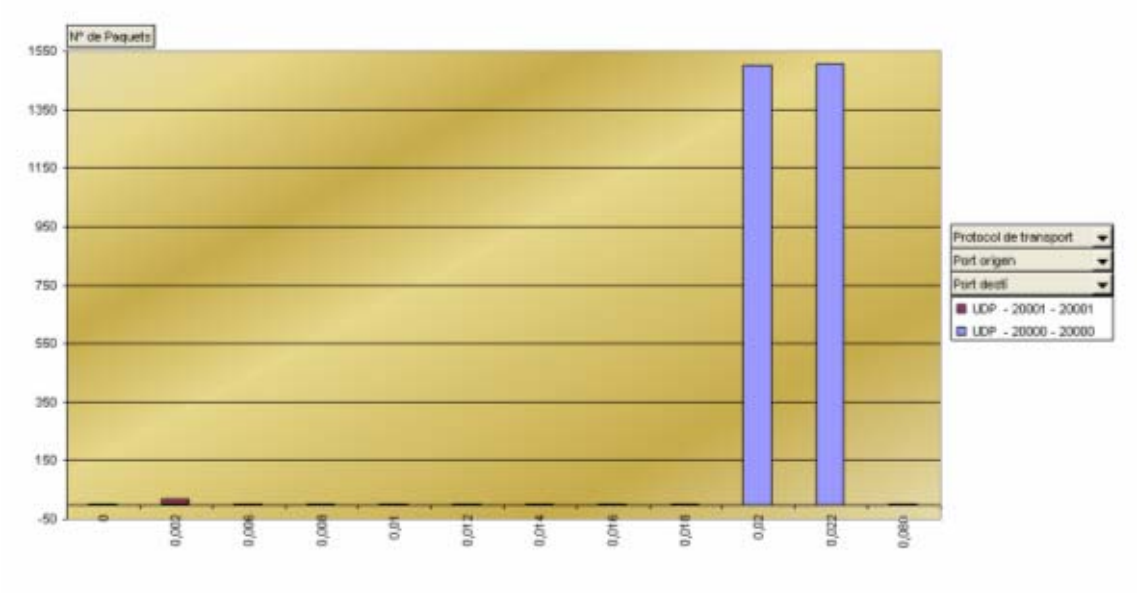


Fig. 1.19 Distribució del temps entre paquets amb supressió

Podem comprovar que el ventall escollit de 0 a 100ms, ens mostra que només hi han hagut paquets amb retards de valors: 2ms, 6ms, 8ms, 10ms, 12ms, 14ms, 16ms, 18ms, 20ms, 22ms i 80ms. A on es veu clarament que la probabilitat de que es retardi un paquet, està entre el marge 20 – 22ms i que correspon a les impressions que hem tingut amb les distribucions anteriors.

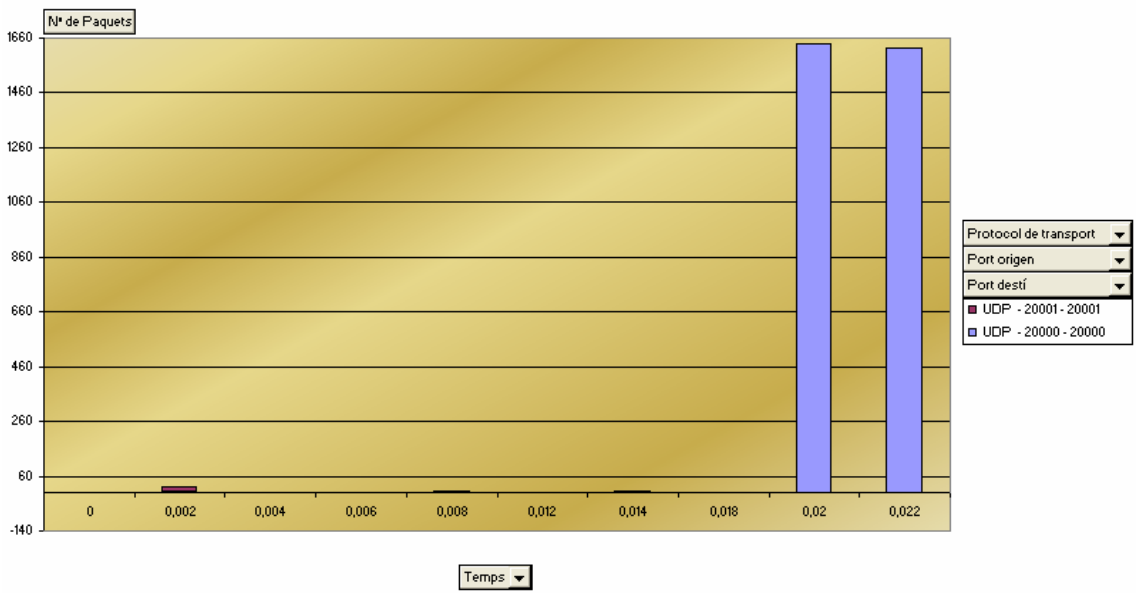


Fig. 1.20 Distribució del temps entre paquets sense supressió

Sense supressió es pot veure com utilitzant la mateixa finestra de 0 a 100ms i filtrant també els instants que no s'envia informació, es tenen retards de: 2ms, 4ms, 6ms, 8ms, 12ms, 14ms, 18ms, 20ms i 22ms. Però que el retard amb més probabilitat segueix sent el comprès entre 20 i 22ms, tal com es ve dient.

Un altre aspecte a destacar és el nombre de paquets, respecte la "distribució del temps entre paquets" amb supressió. Aquest augment s'esdevé degut a que en el mode "amb supressió" els silencis d'àudio són eliminats i per tant, el nombre de paquets és menor.

d) Distribució del tamany dels paquets

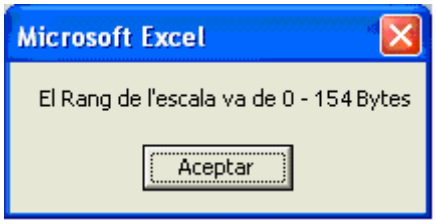


Fig. 1.21 Missatge de tamany informatiu

En aquesta distribució l'aplicació ens indica que el tamany màxim capturat en tota la transmissió, ha estat de 154 bytes. Pel que s'agafa aquest mateix marge en la seva representació, una resolució d'1 byte i filtratge d'instants en que no s'ha transmet informació, per a poder apreciar la distribució més globalment.

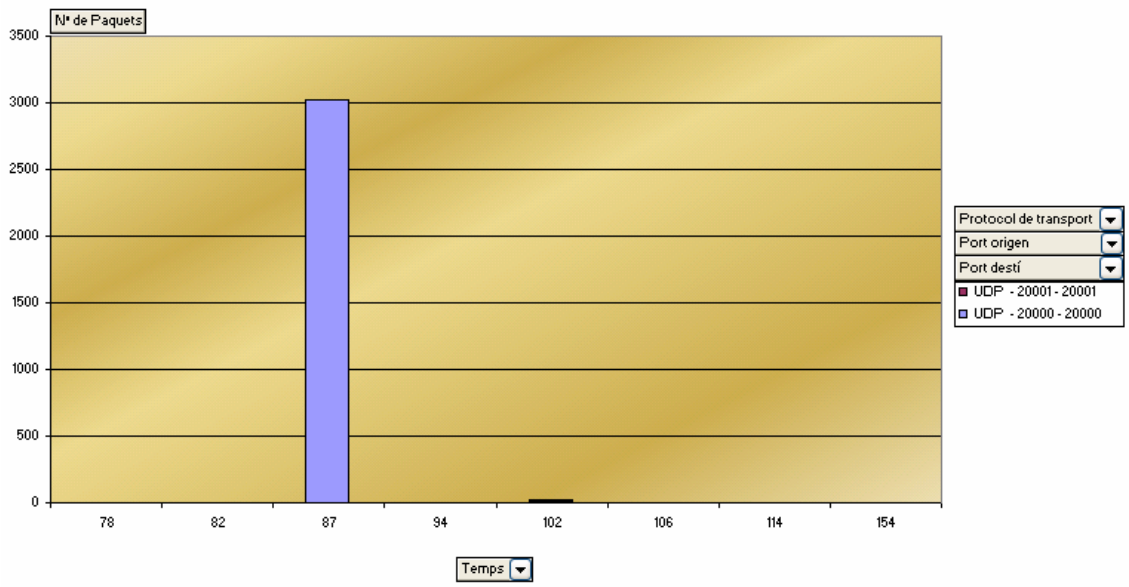


Fig. 1.22 Distribució del tamany dels paquets amb supressió

Amb la supressió de silenci s'observa que la distribució de tamanys està composta pels valors: 78, 82, 87, 94, 102, 106, 134 i 154 bytes. Es pot comprovar com el tamany més probable que tingui un paquet transmès, resulta ésser de valor 87 bytes.

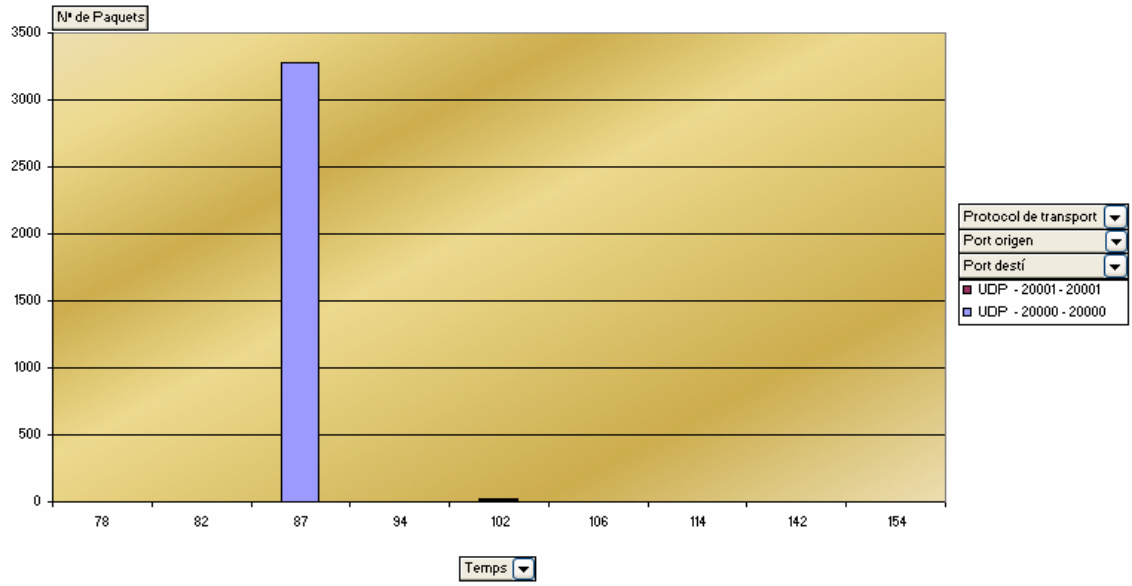


Fig. 1.23 Distribució del tamany dels paquets sense supressió

Es pot veure que "sense supressió" de silenci, els valors que conformen la distribució del tamany de paquets quasi coincideix amb la captura "amb supressió" (78, 82, 87, 94, 102, 106, 114, 142 i 154 bytes); això és degut a que

el que canvia respecte un mode i un altre, és el tràfic generat durant la conversació de veu (ja que el còdec de veu segueix sent el mateix).

Com s'observa, el valor més probable de tamany segueix sent 87 bytes. També es pot apreciar que el nombre de paquets que repeteixen aquest valor de byte, és molt superior que en el cas "amb supressió" (degut que s'han enviat molt més paquets/segon que en el mode "amb supressió").

e) Velocitat instantània

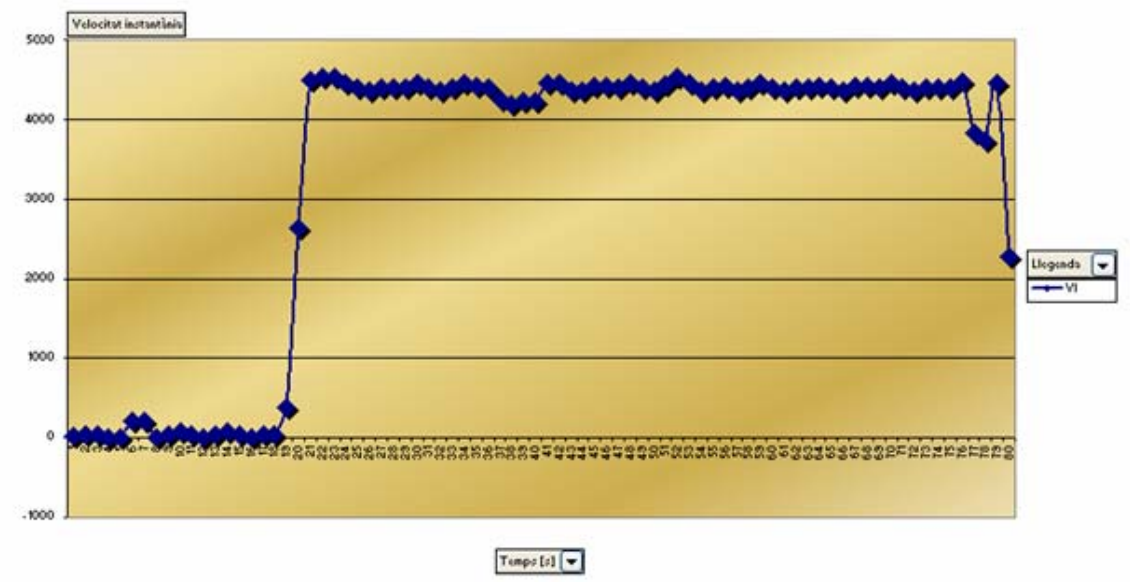


Fig. 1.24 Velocitat instantània amb supressió

En aquest cas s'ha escollit una finestra de 0 a 80,79 segons i una resolució d'1 segon, per a poder apreciar els increments/decrements de velocitats instantànies que hi ha a l'inici de sessió i el començament de la conferència.

Es pot veure com les velocitats que es transmeten en cada instant, varien en funció de la informació que s'envia. S'observa clarament com al principi de la sessió, aquestes són d'un nivell molt baix (menor dels 300 bytes/s); mentre que quan s'inicia la comunicació de veu entre els dos PCs la xifra s'eleva als voltants de 4500 bytes/segon.

També es pot veure que es tracta d'una captura "amb supressió" de silenci, perquè el període a on es transmet informació de veu és de caire variable; degut a que si no hi ha àudio aquest no s'envia i per tan la velocitat instantània baixa.

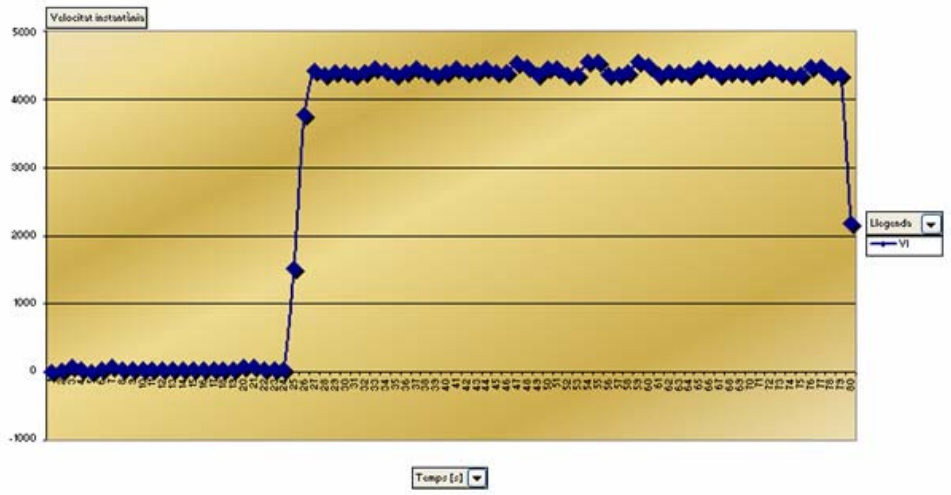


Fig. 1.25 Velocitat instantània sense supressió

Com en el cas anterior durant la comunicació verbal, és el tram a on els valors de velocitats instantànies són màximes.

En aquesta *figura 1.25* també es pot observar que es tracta d'una captura "sense supressió" de silenci, ja que els valors de velocitats són molt constants tan en la part d'inici de la comunicació d'àudio com en el començament de la conferència entre els dos PCs. Degut a que el nombre de paquets per segon no disminueix sinó hi ha informació de veu a transmetre (tal com es va veure en la "distribució de paquets/segon sense supressió").

e1) Distribució de velocitats

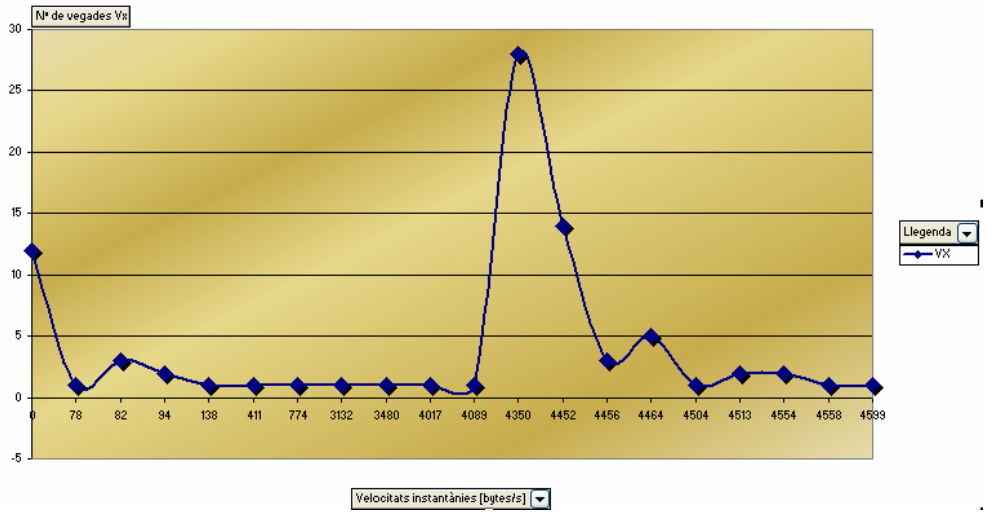


Fig. 1.26 Distribució de velocitats amb supressió

Podem observar, com la velocitat més probable de transmissió correspon a 4350 bytes/segon (ja que és la que s'ha repetit més vegades en la distribució). Aquesta velocitat fa referència al tram a on s'ha establert la conferència de veu, perquè és el període a on les velocitats instantànies han estat més constants.

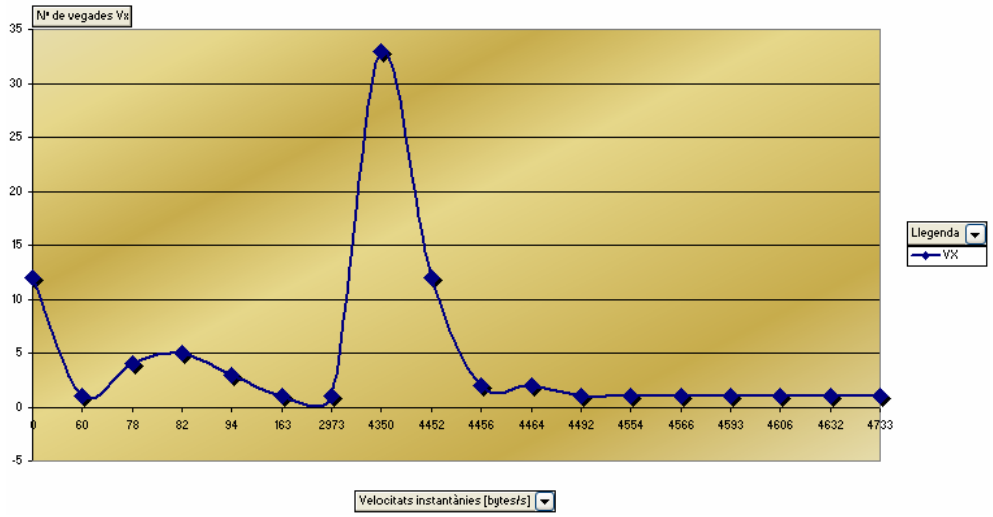


Fig. 1.27 Distribució de velocitats sense supressió

En sense supressió, es pot comprovar com el nombre de vegades que es repeteix la mitjana de velocitat instantània és superior al cas "amb supressió". Això ve pel fet que en aquesta circumstància el tràfic és més constant, que no en "amb supressió". Es pot veure com aquesta mitja, està a l'entorn dels 4350 bytes/segon.

f) Estadístiques de protocols

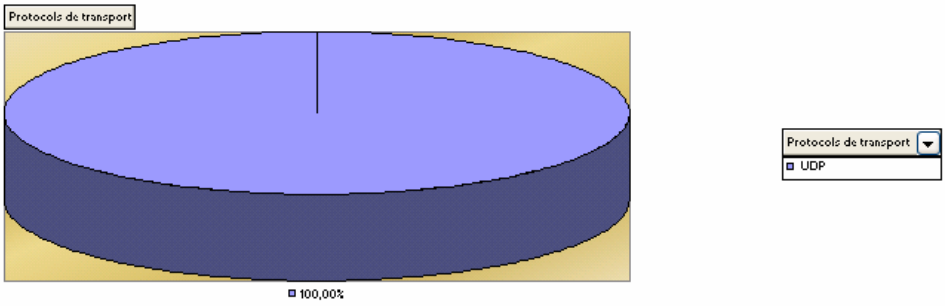


Fig. 1.28 Estadístiques de protocols de transport amb/sense supressió

Tan en el cas "amb supressió" com "sense supressió", s'observa que la totalitat dels protocols de transport utilitzats ha estat UDP. Açò ve determinat per l'aplicació RAT, degut a que no està orientada a la connexió.

Aquesta aplicació de so en temps real (RAT) no admet variacions del retard elevades, perquè les mostres han de processar-se a un ritme fix per a reconstruir el senyal de so. Per això, la recuperació automàtica dels errors que fa TCP no serveixi per a recuperar les mostres perdudes i aquest és el motiu que no s'hagi utilitzat en cap mode.

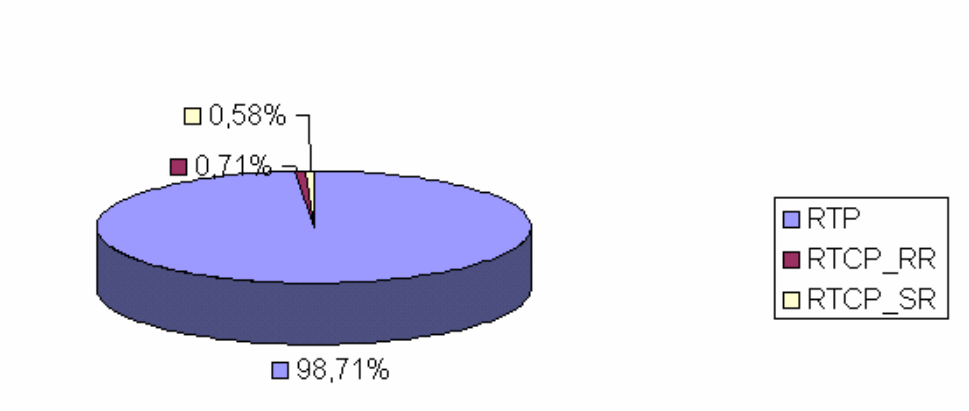


Fig. 1.29a Estadístiques de protocols d'aplicació amb supressió

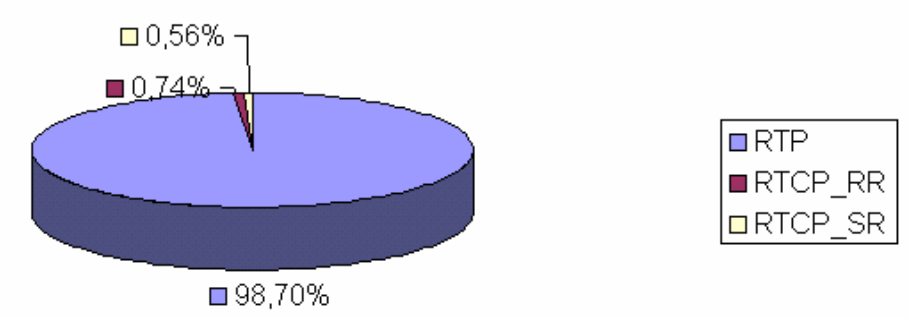


Fig. 1.29b Estadístiques de protocols d'aplicació sense supressió

Podem comprovar, que com alternativa a un control en la transmissió es fa ús del protocol RTCP tan en el cas "amb supressió" com en el cas "sense supressió". D'aquesta forma es pot proporcionar informació de control sobre la qualitat que s'està transmeten.

Anteriorment hem pogut veure com el RAT emet paquets RTCP de control periòdics, de manera associada al flux RTP i que com es veu en les *figures 1.29a i 1.29b*, predomina aquest últim sobre el RTCP (ja que és el que permet tenir conferències amb múltiples usuaris).

S'observa que en el mode "sense supressió" hi ha un lleuger augment en els paquets RTCP del tipus RR (Receiver Report), els quals són un "informe" en recepció amb estadístiques rebudes en els emissors. Mentre que en el mode "amb supressió", ascendeixen els paquets RTCP però de caire SR (Sender Report) els quals són informes en emissió amb les estadístiques de

transmissió. Es veu doncs, que el fet que en un mode es generi un major tràfic i constant, fa tenir més presència de paquets RTCP tipus “RR”. Mentre que una generació més variable, provoca una emissió més elevada de paquets RTCP de tipus “SR”.

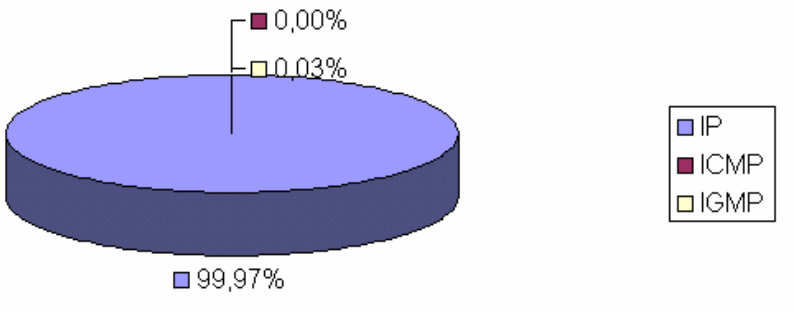


Fig. 1.30 Estadístiques de protocols de xarxa amb/sense supressió

Com es pot veure perfectament el protocol de xarxa predominant és l’IP, el qual com a funció bàsica del protocol és etiquetar els paquets amb la direcció del host que es té com a destí.

També s’observa que hi ha una lleugera presència però mínima del protocol IGMP, ja que permet a altres hosts conèixer quins pertanyen a quins grups de multicast.

Es comprova que tan en el mode “amb supressió” com en el mode “sense supressió” la presència de protocols de xarxa és la mateixa, degut a que no estem fen referència a un nivell de transport a on influiria el tràfic generat i per tan faria obtenir uns resultats més distanciats.

1.3.4. Còdec AMR

S’ha fet ús dels 8 modes de que es disposen en l’AMR i s’han analitzat per a cadascú d’aquest la situació amb i sense supressió. Per tan veiem els resultats de les gràfiques:

a) Distribució dels bytes en el temps

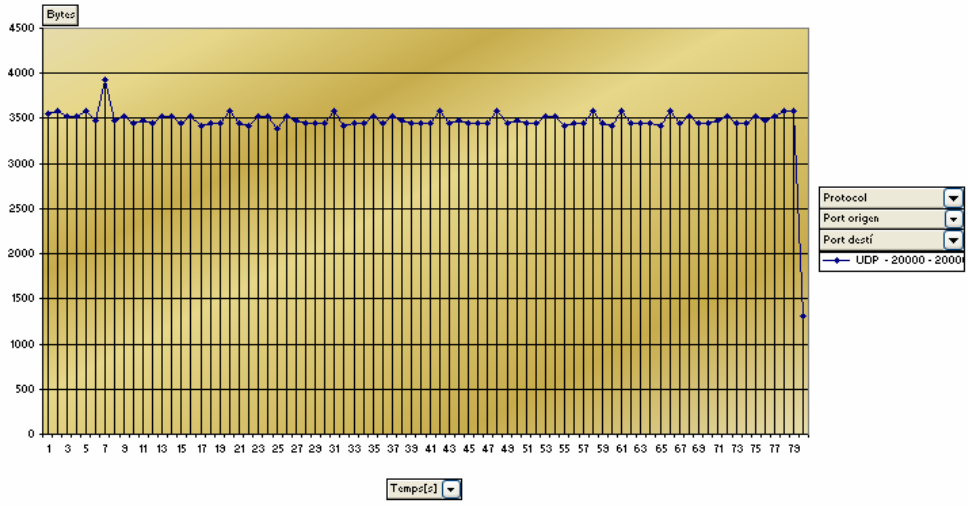


Fig.31 AMR Mode 2 amb supressió de silenci

En aquest cas s'ha pogut comprovar que conforme va en augment la velocitat de transmissió (modes 2 al 7) aquesta distribució presenta un increment en la taxa de bytes i també experimenta una major no uniformitat. Aquesta aleatorietat, també es reflecteix de l'estat "amb supressió" a l'estat en "sense supressió" ja que si no hi ha informació de veu aquesta es segueix enviant (tal com es veu en la figura següent). Es comprova que tot el tràfic segueix essent UDP i que en aquest cas s'utilitza el port 20000 per establir les comunicacions de veu.

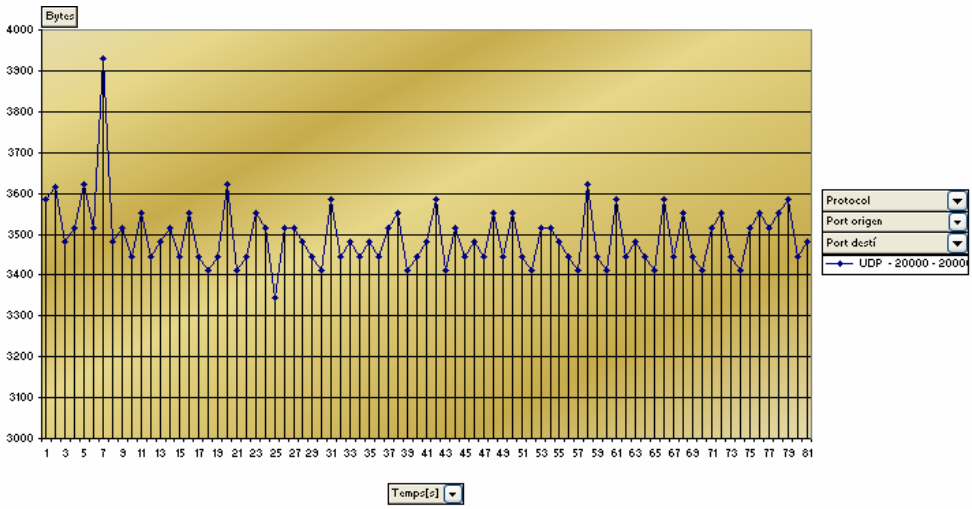


Fig. 32 Mode 2 sense supressió de silenci

b) Distribució dels paquets en el temps

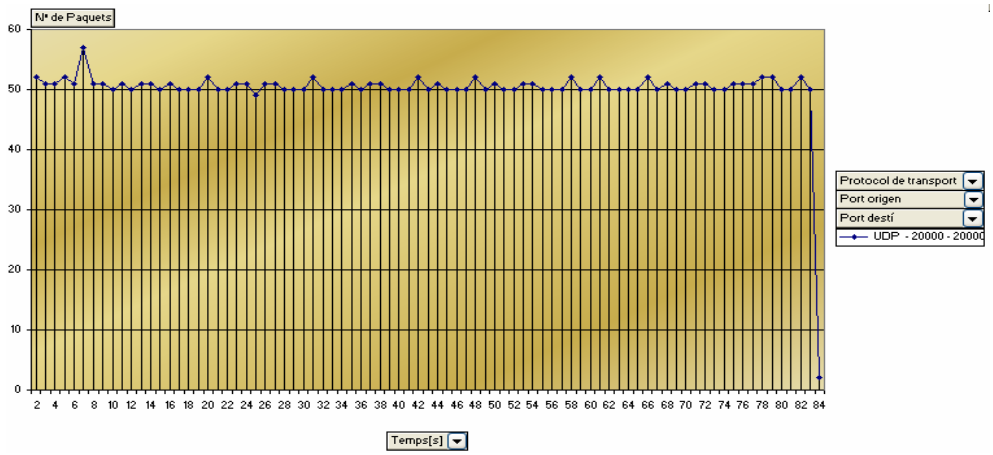


Fig. 33 Mode 6 amb supressió de silenci

En aquest punt, en general es manté la distribució dels paquets en el temps per a cada mode i que és de l'ordre de ~ 50 paquets/s. Exceptuant el mode 2 (5.9kbps) a on es crea una certa aleatorietat. Pel que fa a l'estat "sense supressió" aquest no varia en cadascú dels modes respecte "amb supressió", degut a la mateixa taxa de transmissió que està fixada en l'aplicació (tal i com s'observa en la figura que segueix).

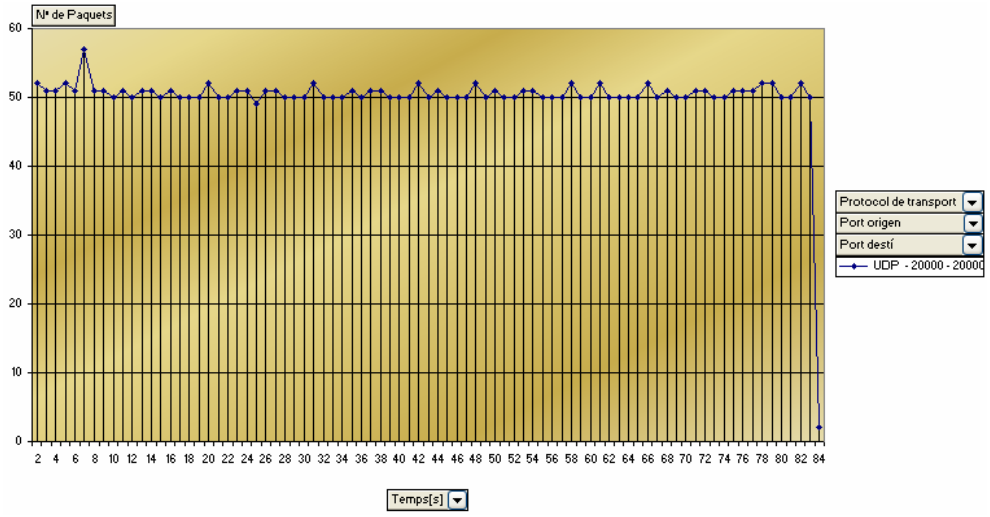


Fig. 34 Mode 6 sense supressió de silenci

c) Distribució del temps entre paquets

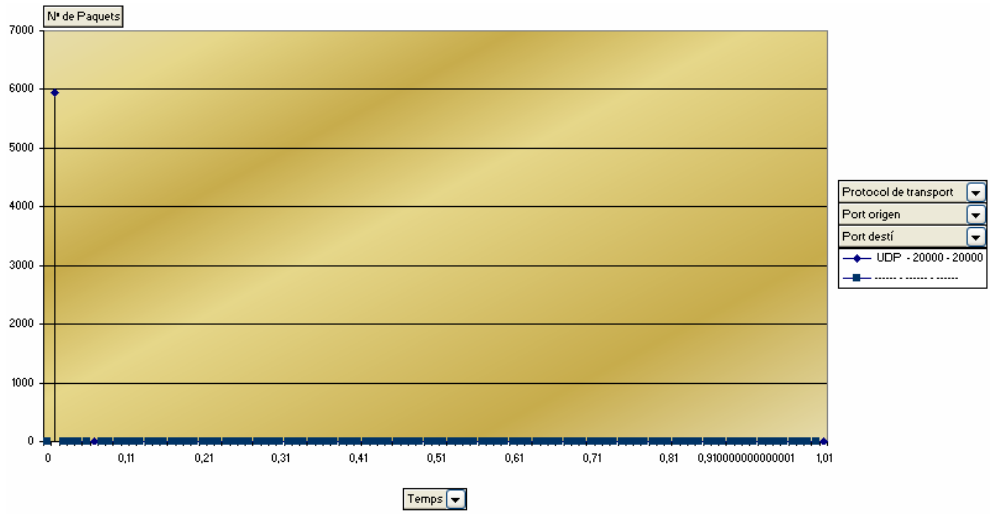


Fig. 35 Mode 7 amb supressió de silenci

El temps entre paquets que experimenten tots els modes tan “amb supressió” com en “sense supressió de silenci” a l’hora de realitzar les transmissions de paquets, resulta ésser de l’entorn als ~ 20ms.

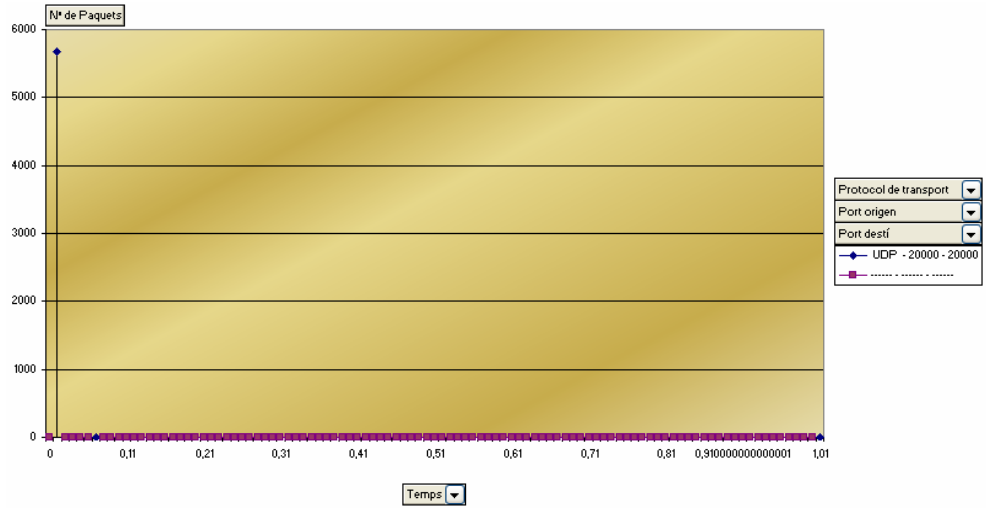


Fig. 36 Mode 7 amb supressió de silenci

d) Distribució del tamany dels paquets

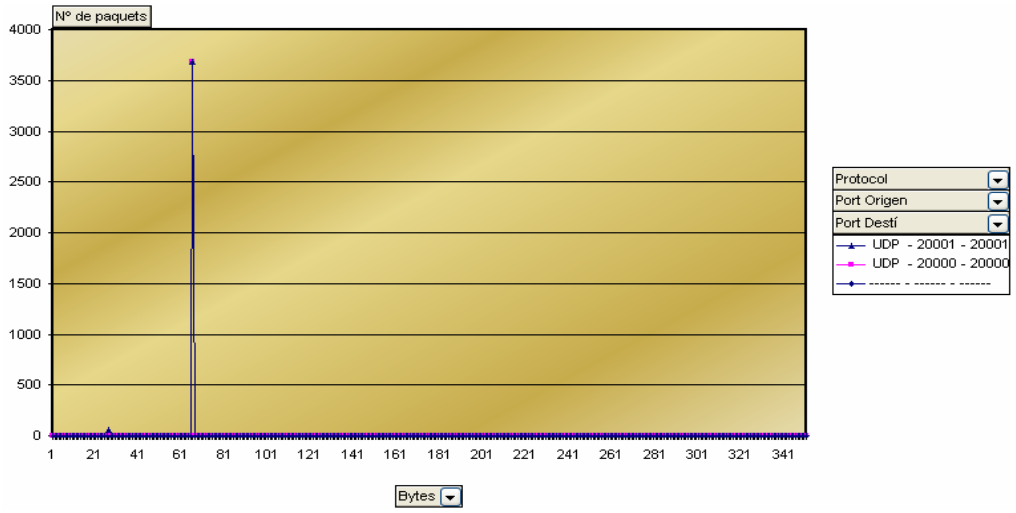


Fig. 37 Mode 0 amb supressió de silenci

La distribució del tamany dels paquets comença per un valor mínim en el Mode0 de ~ 66 bytes i acaba en un valor màxim en el Mode7 de ~ 85 bytes. Els valors de distribucions de tamany per "sense" i "amb" supressió per a cada Mode no varien, ja que els paquets es transmeten amb la mateixa longitud.

En la figura anterior, podem veure com ens trobem paquets de control (port 20001) amb tamany 27 bytes i paquets amb informació de veu de tamany 66 bytes pel Mode0. Mentre que al Mode7 (figura següent) el tamany més probable de paquet, ronda als voltants de 85 bytes.

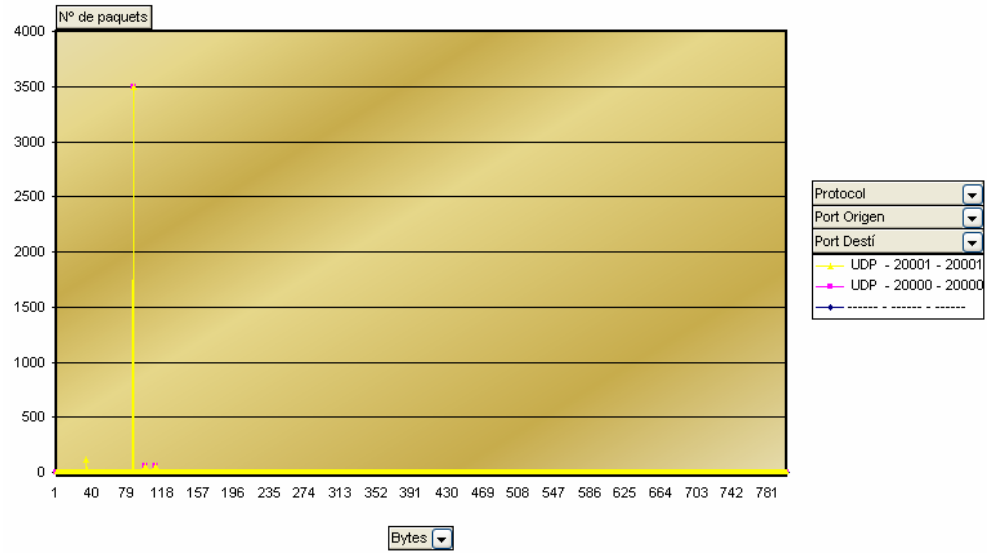


Fig. 38 Mode 7 sense supressió de silenci

e) Velocitats instantànies

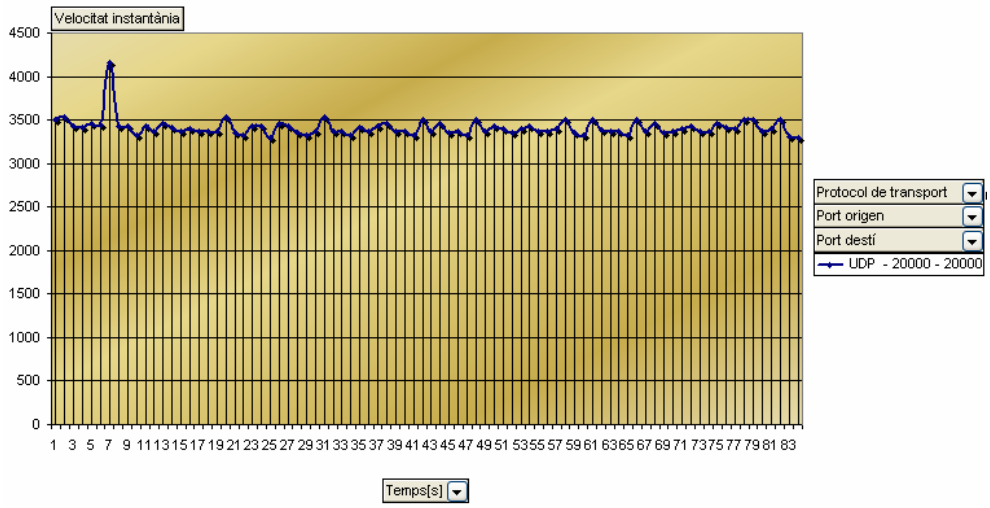


Fig. 39 Mode 1 amb supressió de silenci

S'ha observat a tots els modes que les velocitats instantànies són de caràcter constant, tan en l'estat de "amb supressió" com "sense supressió de silenci". En la figura anterior es veu com tot el tràfic és UDP i la taxa de velocitat instantània està als voltants de 3.5kb/s, mentre que en la figura que segueix (mode més alt) la velocitat instantània experimenta un augment en la seva taxa, que passa a ésser de ~ 3.75kb/s.

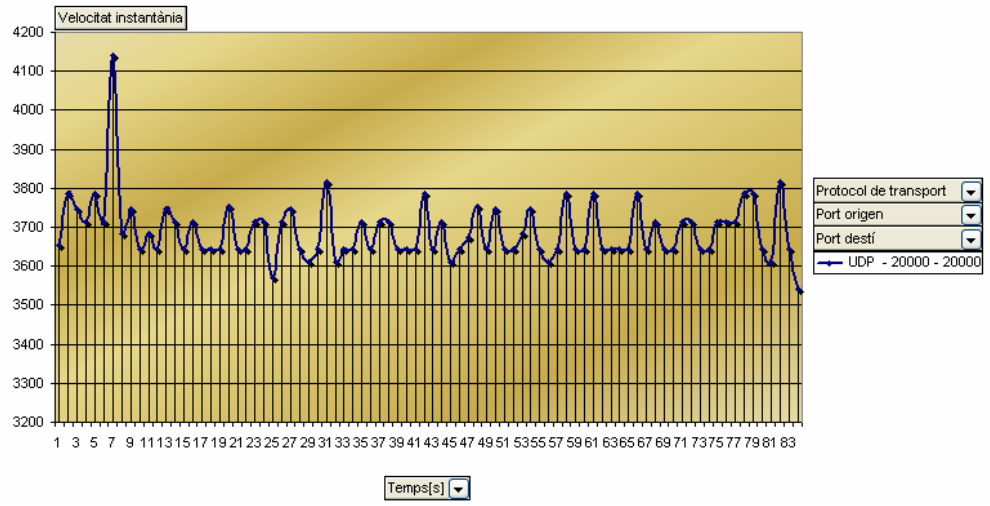


Fig. 40 Mode 3 sense supressió de silenci

f) Distribució de velocitats instantànies

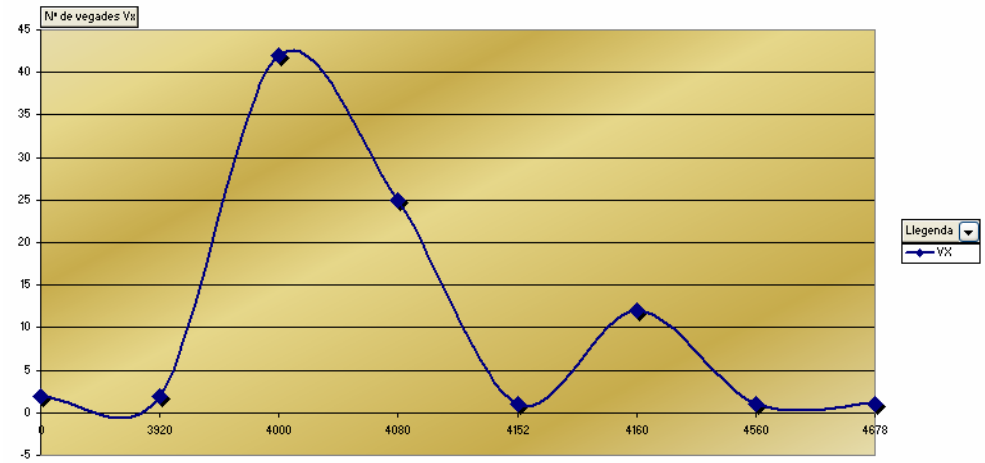


Fig. 41 Mode 6 amb supressió de silenci

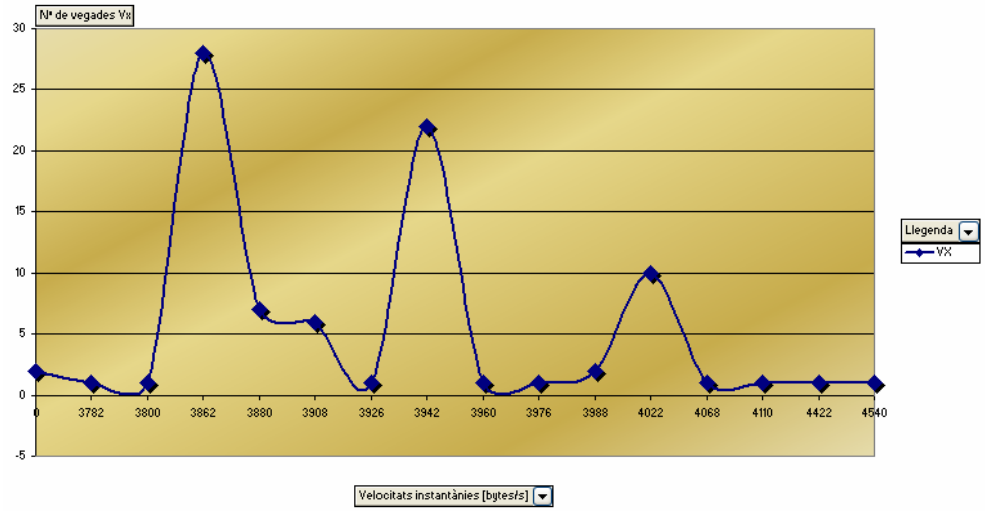


Fig. 42 Mode 6 sense supressió de silenci

En tots els modes hem pogut veure com la distribució de velocitats instantànies és molt més aleatòria en el cas "sense supressió de silenci", que "amb supressió" (tal com s'observa en les gràfiques anteriors del mode 6). Això és degut a que en "supressió de silenci" si els usuaris "no parlen" no s'envia informació, mentre que en "sense supressió de silenci" es segueix transmetent.

g) Estadístiques de protocols

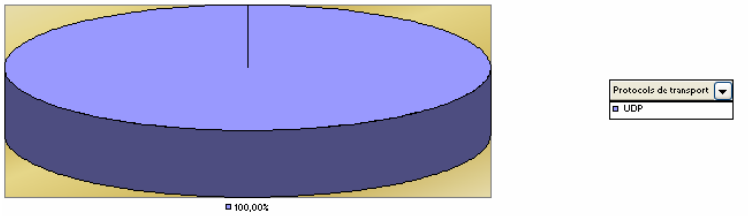


Fig. 43 Protocols de transport de tots els Modes amb i sense supressió

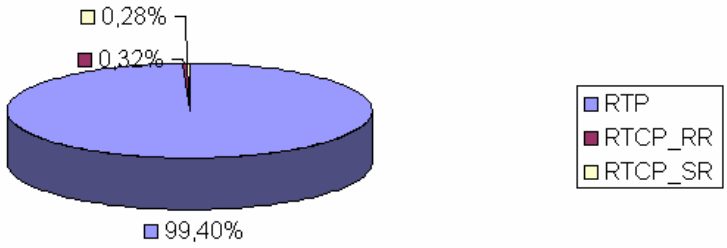


Fig. 44 Mitja de Protocols d'aplicació dels diferents Modes

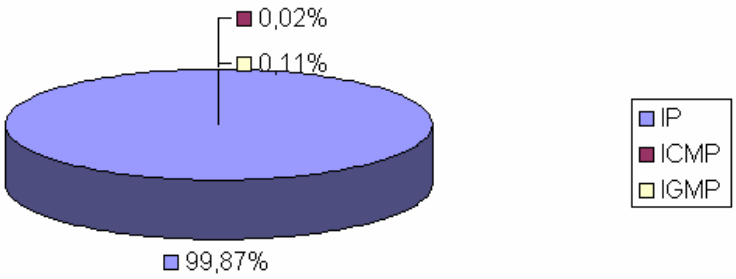


Fig. 45 Mitja de Protocols de xarxa dels diferents Modes

Per a poder saber l'ús que se'n fa dels protocols, s'ha fet la mitja d'utilització d'aquests tan pel cas de "amb supressió" com "sense supressió" de silenci i així poder tenir una idea de quins són els que es tan més presents en les comunicacions via RAT. Podem apreciar en les gràfiques anteriors, com el protocol de transport per excel·lència és UDP i que el nivell de xarxa és IP (a on té un cert pes el protocol IGMP). Com a protocols d'aplicació, juga un paper bastant important el protocol RTP per a l'aplicació en temps real i a on dels tipus de RTCP més utilitzat és el RTCP_RR que són generats per emissors no actius mentre que els SR si que ho fan els actius.

2. ENTORN DE PROVES AMB EL PESQ

2.1. Introducció

Tal i com es comentarà al capítol 3 es desitja poder integrar el còdec AMR utilitzat per UMTS a l'aplicació per videoconferència RAT. Per tan, s'haurà de fer un estudi de com es comporten els diferents modes del còdec segons les condicions que es donin al canal de comunicació. Per fer aquest estudi s'ha utilitzat l'eina PESQ.

2.1.1. PESQ (Perceptual Evaluation of Speech Quality)

PESQ és una eina de la ITU que permet obtenir una nota MOS segons la diferència que aprecia entre dos arxius que se l'introdueixen: l'original i el que ha sofert degradació. Quan més semblants siguin els fitxers més alt serà el coeficient MOS que donarà com resultat.

Està normalitzada segons la recomanació P.862 que ha sigut el resultat de varis anys de treball i que permet predir els resultats de tests sobre la qualitat subjectiva de la veu en sistemes de telefonia.

Mitjançant un complex model que té en compte diversos factors com: distorsió a la codificació, pèrdua de paquets, errors, retard, variabilitat del retard,... compara l'arxiu original amb el que presenta una certa degradació i aconsegueix obtenir una nota MOS (Mean Opinion Score) equivalent a la que s'obtindria si es fes-hi un estudi amb diferent tipus de persones.

La nota MOS és la mesura subjectiva de qualitat de la veu més acceptada i correspon al resultat extret d'un conjunt de persones a l'escoltar àudio i donar la seva opinió sobre la qualitat que han percebut.

El funcionament del PESQ és senzill. Només hi ha que passar-li com paràmetres la velocitat de mostreig (+8000 ó +16000), l'arxiu original i el degradat, per aquest ordre.

pesq +8000 arxiu_org arxiu_degradat

Amb això calcula una nota que va des de -0.5 (menor grau de similitud) a 4.5 (major grau de versemblança).

L'estructura general que té l'aplicació és la que es mostra seguidament en aquest esquema:

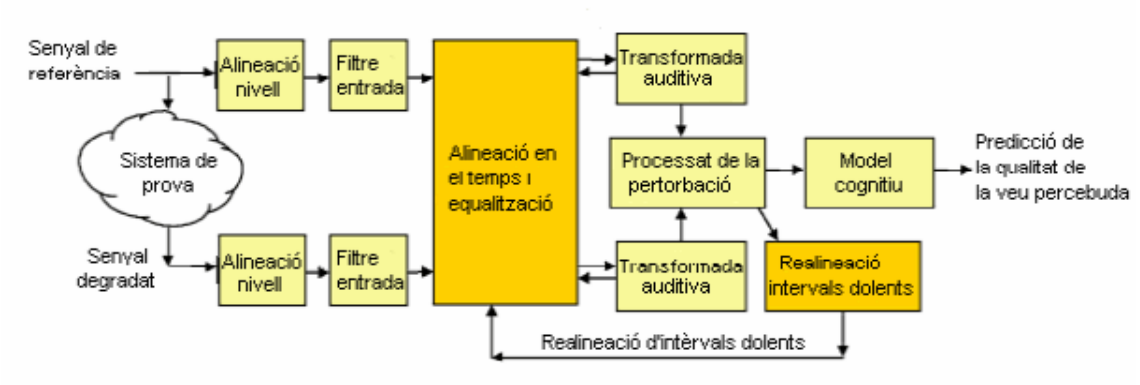


Fig. 2.1 Esquema de blocs aplicació PESQ

En un primer pas el PESQ realitza una alineació dels dos senyals fins a un nivell estàndard d'audició. Aquest pas es fa ja que per poder comparar ambdós senyals, han d'estar al mateix nivell de potència constant. Això es degut a que el senyal de referència no té per que estar a un nivell predefinit i perquè el guany del sistema de prova no es coneix.

Seguidament es filtren usant FFT mitjançant un filtre d'entrada per modelar un terminal telefònic estàndard. D'aquesta forma es té en compte l'efecte dels components acústics i elèctrics del terminal. Els senyals resultants són alineats en temps i equalitzats. Degut a que el sistema de prova del que es vol saber la qualitat introduirà un cert retard i probablement serà variable, es necessita alinear els dos senyals. PESQ aplica detecció d'activitat de veu per poder descartar les parts del senyal que són soroll. Les mesures que fa de l'offset no tenen en compte el retard que introdueix l'equip de testeig que s'estigui utilitzant al generar o gravar el senyal. Això significa que un offset de temps reportat pel PESQ a un arxiu serà depenent de la manera en que el procés de testeig es porti a terme.

Més tard es fa una transformada auditiva que permet tenir en consideració certes propietats de l'audició humana i que simularia com un oient hauria escoltat aquests senyals. Amb el processat de la pertorbació es té en compte l'efecte d'emmascarament degut a que distorsions petites en els senyals no són percebudes en presència de senyals més alts.

En certs casos l'alineació que s'ha realitzat abans fallarà a l'hora de detectar un canvi en el retard. Aquests errors s'identifiquen etiquetant les trames dolentes. Amb el que cadascuna de les seccions dolentes es tornen a alinear i la pertorbació es calcula de nou.

Finalment els paràmetres de pertorbació es converteixen en puntuació PESQ.

2.1.2. Necessitat

La qualitat de la comunicació a través del canal es mesura mitjançant la comparació amb el PESQ de l'arxiu original de veu que ha generat el RAT i el que s'ha rebut a l'altre extrem.

Degut a que el PESQ a l'hora de calcular el resultat no té en compta el retard que s'ha introduït a l'equip on s'estan fent les proves, la nota obtinguda no serà del tot correcte. La nota final dependrà de la manera en que es faci el procés al generar o gravar l'arxiu d'àudio. Per tan es necessita alguna forma de poder sincronitzar l'arxiu original amb el que ha sofert la degradació al passar a través de la xarxa. Per fer aquesta sincronització s'ha desenvolupat l'aplicació "syncok".

2.1.3. Sincronitzador d'arxius "Syncok"

a) Funcionament

En aquesta utilitat se li passen com paràmetres el nom del arxiu original, el degradat i el del fitxer on es vol que es guardi el degradat ja sincronitzat, de la següent manera:

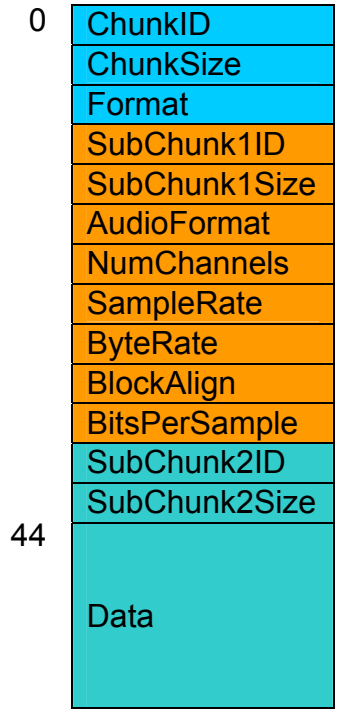
Syncok "*ORIGINAL.wav*" "*DEGRADAT.wav*" "*NOM_ARXIU_FINAL.wav*"

També existeix la possibilitat de no passar els paràmetres per referència i és llavors el programa qui els demana mitjançant un menú.

Ja que un dels formats que utilitza l'aplicació RAT per gravar i reproduir-ne fitxers d'àudio és el WAV, veiem com està definit:

El format d'arxius WAV està construït mitjançant una sèrie de components més simples denominats *chunks*. Un *chunk* està constituït per un codi de 4 bytes, longitud de 4 bytes i un camp de dades opcionals. La longitud correspon amb el que ocupen les dades opcionals. Si aquesta és zero indicarà que no existeixen dades opcionals. El contingut d'un *chunk* depèn del tipus que sigui, que és indicat al camp codi. Alguns d'aquest *chunks* també es divideixen en *subchunks*.

WAV és un subconjunt de l'especificació RIFF per formats multimèdia. Aquest tipus de format es divideix en una part de capçalera i una altre part de dades. Un fitxer WAV és usualment un *chunk* RIFF simple que conté un codi de 4 bytes 'WAVE' i un conjunt de *subchunks*. Un arxiu WAV mínim només conté dos *subchunks* 'fmt' i 'data' en el que es coneix com forma canònica:



Descriptor de *chunk* RIFF (El format que correspon és WAVE)

Subchunk 'fmt' (Descriu el format de la informació de so i del subchunk 'data')

Subchunk 'data' (Indica el tamany de la informació de so i conté les dades de so RAW)

Fig. 2.2 Forma canònica

La manera més simple d'escriure dades en fitxers WAV és usant la forma canònica, ja que és compatible amb qualsevol programa inclús el més vell. Com es pot comprovar la capçalera prèvia a les mostres de so ocupa 44 bytes.

Els camps corresponents a la forma canònica són els següents:

ChunkID: Conté les lletres "RIFF" en ASCII.
ChunkSize: Tamany de l'arxiu sencer en bytes menys, exceptuant els 8 primers bytes pertanyents a *ChunkID* y *ChunkSize.*
Format: Conté les lletres "WAVE".

SubChunk1ID: Conté les lletres 'fmt'.
SubChunk1Size: 16 per PCM. Aquest és el tamany de la resta del *subchunk.*

AudioFormat: PCM=1. Valors diferents de 1 indiquen algun tipus de compressió.
NumChannels: Mono=1, Estèreo=2, ...
SampleRate: 8000, 44100, ...
ByteRate: *SampleRate * NumChannels * BitsPerSample /* 8.
BlockAlign: *NumChannels * BitsPerSample /* 8.
BitsPerSample: 8 bits=8, 16 bits=16, ...

SubChunk2ID: Conté les lletres 'data'.
SubChunk2Size: *NumSamples * NumChannels * BitsPerSample /* 8.
Data: Dades actuals de so.

Per fer aquesta sincronització "syncok" agafa l'arxiu original, deixa intactes els primers 44 bytes corresponents a la capçalera WAV, i li treu el silenci que pugui haver-hi des d'aquí fins a la primera mostra de veu. Utilitzant les mostres restants fa una correlació creuada amb les existents en el fitxer degradat. El punt de valor màxim indicarà la posició a la que es tenen que traslladar les mostres en el fitxer degradat, per a que coincideixi amb l'original; és a dir, que quedarà sincronitzat.

b) Característiques

Amb la supressió de silenci inicial del fitxer original s'evita que el valor de correlació màxim pugui quedar falsejat. Degut a que podria donar-se el cas de que correspongués a una part silenciada de l'arxiu degradat. També s'aconsegueix eliminar les mostres que no són de veu i que apareixen a l'iniciar la gravació (per exemple al moment de donar al REC). A més el nombre de mostres que s'agafen es menor amb el que s'aconsegueix reduir els càlculs a l'hora de fer la correlació. A l'opció de poder fer la sincronització mitjançant el menú, s'ha deixat la possibilitat de que es pugui escollir la finestra per a realitzar la correlació.

c) Verificació del funcionament

Per verificar el funcionament del sincronitzador s'han fet proves amb els diferents arxius WAV que s'annexen a la recomanació PESQ. Aquests fitxers s'utilitzen per realitzar validacions de conformitat de qualsevol implementació PESQ que es realitzi en comparació amb la que ve annexada a la recomanació. Els arxius venen en parelles: l'original i el mateix amb una certa degradació. Alguns d'aquests es troben sincronitzats i d'altres presenten un cert retard. Segons la problemàtica esmentada, el retard que s'afegeix degut a la forma en que es fes el test faria que els resultats del PESQ no fossin del tot precisos.

En una primera fase s'ha comprovat mitjançant "syncok" la correcta sincronització dels diferents arxius originals que ens dóna la ITU.

Per a fer aquesta comprovació, s'han modificat els fitxers originals (sense degradació) de la ITU afegint-les-hi un cert soroll / retard i s'han passat per syncok. Un cop obtingut els arxius sincronitzats, s'han passat al PESQ. D'aquesta forma utilitzant aquests fitxers, es comprova que el sincronitzador treballa correctament ja que el resultat del PESQ ha de donar aproximadament el coeficient MOS més alt.

En la taula que segueix es pot veure el resultat d'alguna de les proves que s'han fet, per comprovar el correcte funcionament de l'aplicació "syncok":

Taula 2.1 Exemple de proves "syncok"

REFERENCE	DEGRADED	PESQMOS	PESQMOS	SUBJMOS	COND	SAMPLE_FREQ	CRUDE_DELAY
org.wav	a.wav	SQValue=4.500	4.500	0.000	0	8000	0.0000
org.wav	b.wav	SQValue=4.500	4.500	0.000	0	8000	0.0000
org.wav	c.wav	SQValue=4.500	4.500	0.000	0	8000	0.6760
org.wav	d.wav	SQValue=4.500	4.500	0.000	0	8000	0.0040
org.wav	e.wav	SQValue=4.499	4.499	0.000	0	8000	1.6960
org.wav	f.wav	SQValue=4.500	4.500	0.000	0	8000	0.0040
org.wav	g.wav	SQValue=4.500	4.500	0.000	0	8000	1.0520
org.wav	h.wav	SQValue=4.487	4.487	0.000	0	8000	1.0400
org.wav	i.wav	SQValue=4.499	4.499	0.000	0	8000	0.0080
org.wav	j.wav	SQValue=4.500	4.500	0.000	0	8000	1.0520
org.wav	k.wav	SQValue=4.500	4.500	0.000	0	8000	1.0720
org.wav	l.wav	SQValue=4.500	4.500	0.000	0	8000	0.0080
org.wav	m.wav	SQValue=4.500	4.500	0.000	0	8000	0.0080
org.wav	n.wav	SQValue=4.488	4.488	0.000	0	8000	1.0360
org.wav	o.wav	SQValue=4.500	4.500	0.000	0	8000	0.0040
org.wav	p.wav	SQValue=4.500	4.500	0.000	0	8000	0.0600
org.wav	q.wav	SQValue=4.500	4.500	0.000	0	8000	1.1160
org.wav	r.wav	SQValue=4.500	4.500	0.000	0	8000	0.0040
org.wav	s.wav	SQValue=4.497	4.497	0.000	0	8000	0.0000
org.wav	t.wav	SQValue=4.500	4.500	0.000	0	8000	0.0040

Es pot comprovar com els resultats PESQMOS donen un valor de coeficient proper o igual a 4.5, que és el de màxima similitud. Pel que podem corroborar el bon funcionament de l'aplicació "syncok".

Una vegada s'ha verificat la correcta sincronització del PESQ, es fa la prova anivellant en temps el degradat.

2.2. Caracterització de l'AMR mitjançant PESQ

Com es vol integrar el còdec AMR a l'aplicació RAT per videoconferència, per poder muntar aquest escenari de simulació, interessa fer un estudi de com és el comportament d'aquest còdec enfront a les variacions en quant errors i ample de banda al canal.

2.1.4. Explicació

Tal i com està configurada la xarxa al lloc de treball, conforme es va construint la plataforma sencera d'emulació UMTS, per poder fer proves hi ha dues connexions: una directa entre l'emulador CN i l'emulador UE i un altre a través de l'emulador UTRAN. El programa bypass4, desenvolupat a un altre projecte, s'executa en l'emulador UTRAN. Aquest programa permet generar una certa quantitat d'errors, així com modificar el ample de banda del canal a la comunicació que es fa mitjançant l'UTRAN, que es justament el que ens interessa.

Per tan utilitzem aquest programa junt amb al PESQ per fer la caracterització de l'AMR.

2.1.5. Còdec AMR

El còdec de veu Adaptative Multi-Rate (AMR) és obligatori pels sistemes de tercera generació i té la peculiaritat de que pot variar la seva taxa de transmissió en funció de las condicions del canal. Utilitza una freqüència de mostreig de 8000 Hz i pel cas d'AMR de banda estreta, en el que ens centrarem nosaltres, té 8 modes diferents:

Taula 2.2 Taxes dels modes AMR

Mode	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Taxa (kbit/s)	4.75	5.15	5.90	6.70	7.40	7.95	10.2	12.2

Va ser desenvolupat i estandarditzat per l'ETSI (European Telecommunications Standards Institute) per sistemes cel·lulars GSM. Degut a la seva flexibilitat i robustesa va ser triat per el Third Generation Parnetship Project (3GPP) com a còdec d'obligada implantació en sistemes cel·lulars de tercera generació.

El còdec AMR és multimode i pel cas de banda estreta suporta 8 modes amb taxes de transmissió: 4.75 kbps, 5.15 kbps, 5.90 kbps, 6.70 kbps, 7.40 kbps, 7.95 kbps, 10.2 kbps, 12.2 kbps corresponents als modes 0 fins al 7 respectivament. Utilitza una freqüència de mostreig de 8000Hz i realitza la codificació de la veu amb trames de 20 ms.

AMR permet aconseguir preservar la qualitat de la comunicació sota condicions de transmissió molt diverses. Els bits de veu codificats en AMR tenen diferents nivells de sensibilitat en front a errors. Es defineixen tres classes de bits: A, B i C, essent els de la classe A els més sensibles a errors i C els menys. Els mecanismes de detecció i correcció d'errors es centren en els bits més sensibles (classe A).

Taula 2.3 Classes dels diferents modes AMR

Mode	Velocitat	Bits classe A	Bits totals de veu
0	(4.75 kbps)	42	95
1	(5.15 kbps)	49	103
2	(5.90 kbps)	55	118
3	(6.70 kbps)	58	134
4	(7.40kbps)	61	148
5	(7.95 kbps)	75	159
6	(10.2 kbps)	65	204
7	(12.2 kbps)	81	244

Existeix un compromís entre la velocitat de compressió de la veu i la tolerància a errors. Per realitzar l'adaptació al millor mode a cada instant el descodificador, és a dir, el receptor de la veu ha d'indicar al codificador (emissor) el nou mode que prefereix en el que es coneix com Codec Mode Request (CMR).

Alguns sistemes com GSM imposen limitacions en el nombre de modes suportats (només quatre) i cada quan es pot canviar de mode degut a problemes d'ample de banda i altres limitacions.

Amb l'objectiu de tenir una millor interoperativitat amb aquests sistemes, el descodificador pot utilitzar missatges fora de banda per poder configurar el nombre de trames mínim entre el canvi entre dos modes i per evitar el canvi només entre modes veïns.

2.1.6. Caracterització

Primer de tot es configura el RAT als dos extrems de la xarxa perquè l'UE i el SERVER estableixin la comunicació a través de l'UTRAN. En el cas de l'UE l'ordre per llançar el RAT és la següent:

rat 192.168.4.2 / 5000

D'aquesta forma s'indica al RAT que es connecti amb la interfície del SERVER que passa a través de l'UTRAN i que ho faci pel port 5000.

En el cas del SERVER la comanda és:

rat 192.168.1.2 / 5000

Correspon a la IP de la interfície de l'UE connectada a l'UTRAN i el mateix port que en el cas anterior.

A continuació s'arranca el programa bypass4 des de l'UTRAN per produir les diverses pertorbacions sobre el canal.

Primer de tot veiem com es comporten els diferents modes de l'AMR en front a l'ample de banda del que es disposa al canal. La comanda que s'ha d'introduir a l'UTRAN és la següent:

bypass4 –r velocitat en kbps

El que es fa és limitar la velocitat del canal a la indicada mitjançant "velocitat en kbps".

Llavors, provant amb diferents valors de velocitat obtenim les següents gràfiques de com es comporten els diferents modes de l'AMR. Per a fer-ho, s'han utilitzat els dispositius de xarxa virtuals TUN0 (tràfic que prové d'UE) i TUN1 (tràfic del SERVER):

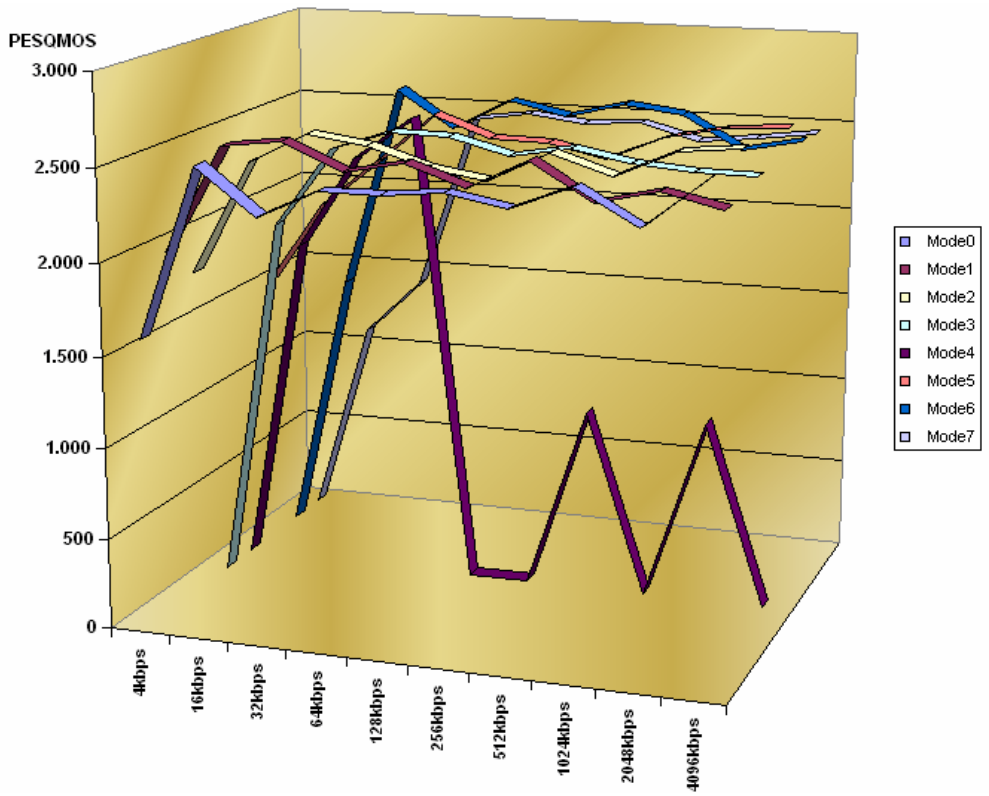


Fig. 2.3 Modes AMR vs ample de banda

Es pot comprovar com en general el resultat PESQMOS per tots els modes comença a estabilitzar-se a partir de 64kbps. Un cop superat aquest llindar el mode 4 cau fins a una aleatorietat que es manté entre 1 i quasi 0; mentre que la resta de modes es mantenen als voltants d'un valor de coeficient de 2.5.

Ara anem a veure com es comporten aquest mateixos modes en front als errors que es produeixen al canal. A l'UTRAN s'executa la següent comanda:

bypass4 –u % errors

En aquest cas el que es fa és introduir al canal un cert percentatge d'errors i aquest percentatge pot variar entre 0 i 99%.

La gràfica que s'obté pels diferents modes es la següent:

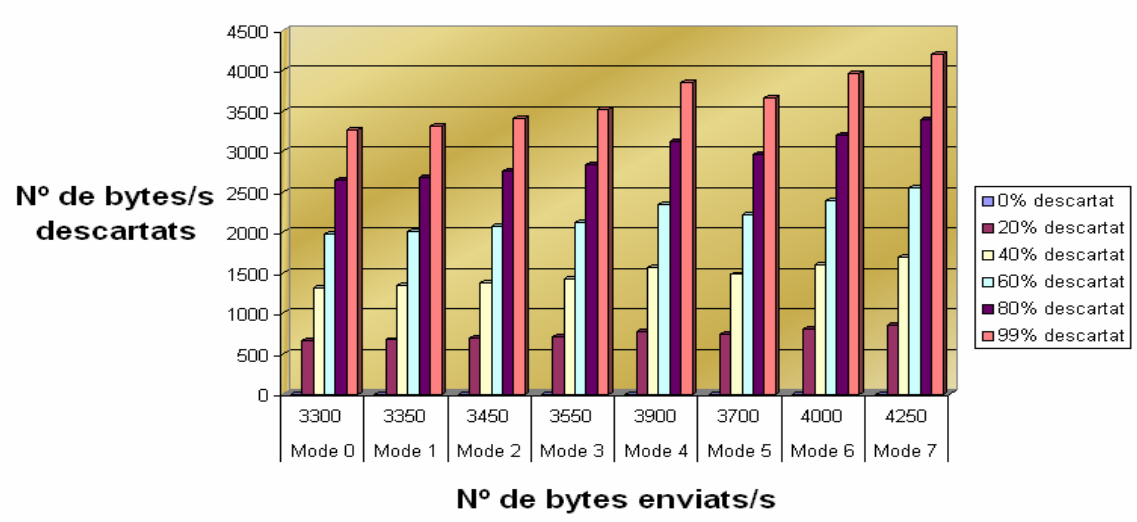


Fig. 2.4 Bytes descartats vs bytes enviats

En la *figura 2.4* es pot comprovar que el nombre de bytes descartats, respecte els enviats corresponen als percentatges de 20, 40, 60, 80 i 99 % de pèrdues. Amb el que es corrobora el correcte funcionament de l'escenari creat amb el Bypass4.

Utilitzant el resultat de la gràfica anterior i el que s'ha obtingut a l'aplicar el PESQ per a cadascú dels casos de pèrdua en els diferents modes AMR, obtenim la gràfica que segueix:

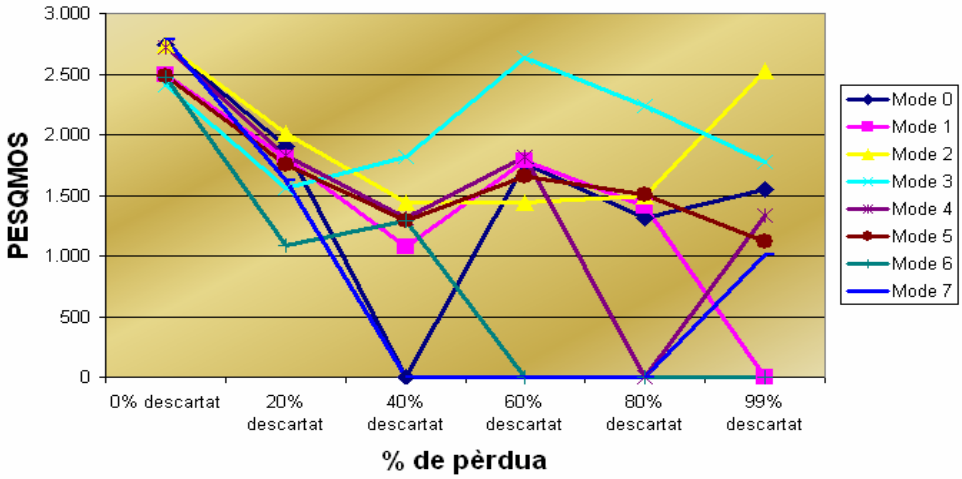


Fig. 2.5 Coeficients PESQMOS vs % de pèrdua

Podem observar que el mode que en general presenta un millor comportament davant possibles errors, és el 3. Es pot veure també que conforme es transmet a una velocitat més alta (casos mode 6 i 7) aquests no són recomanats de fer-ne ús per la resposta que donen a altes pèrdues. Pel que es pot concloure que per a baixes pèrdues en el canal, s'haurien d'utilitzar modes d'alta taxa de transmissió (si l'ample de banda ho permet) i per altes pèrdues és convenient utilitzar modes de més baixa velocitat de transmissió (com per exemple mode 2 i 3).

Si ara agafem un altre fitxer de la ITU: or137.wav i el tornem a passar pel mateix procés que a l'anterior (or114.wav), el resultat que obtenim és el següent:

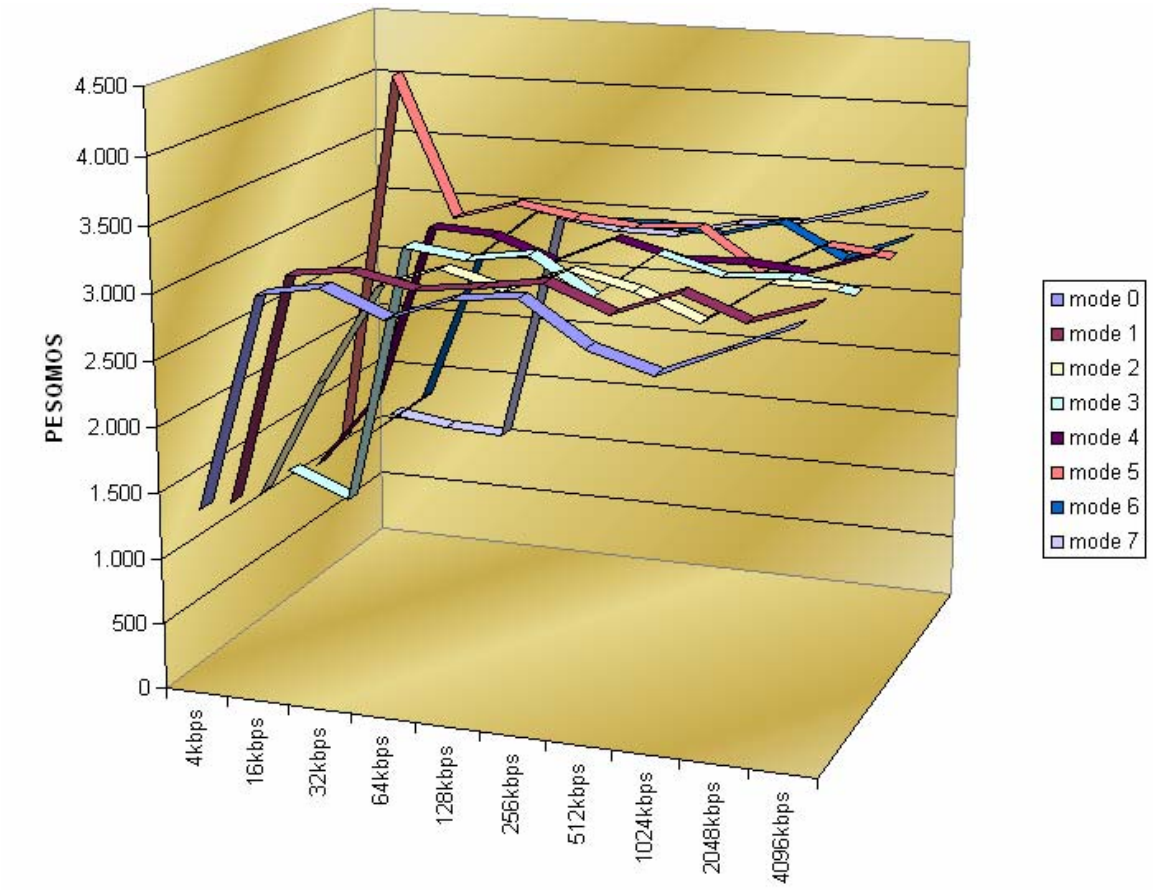


Fig. 2.6 Modes AMR vs ample de banda

Podem veure com en aquest cas els valors de PESQMOS són més elevats, que en l'anterior. Pel que la degradació de la veu, també ve influenciada segons el patró de veu que s'està enviant.

En aquesta *figura 2.6* també podem observar que el mode 4 té un comportament més semblant, a la resta de modes; cosa que no passava amb el resultat obtingut en la *figura 2.3*.

Dels 8 modes de que disposa l'AMR, veiem com el 5 és el que presenta un comportament millor en front els diferents tipus d'ample de banda del canal. Com passava amb l'altre fitxer de veu, la resta de modes de taxa alta comencen a estabilitzar-se a partir de 64kbps. Mentre que els de menor taxa, ja ho fan a inicis de 16 kbps (modes 0 i 1).

El comportament davant a errors, és el que es mostra a continuació:

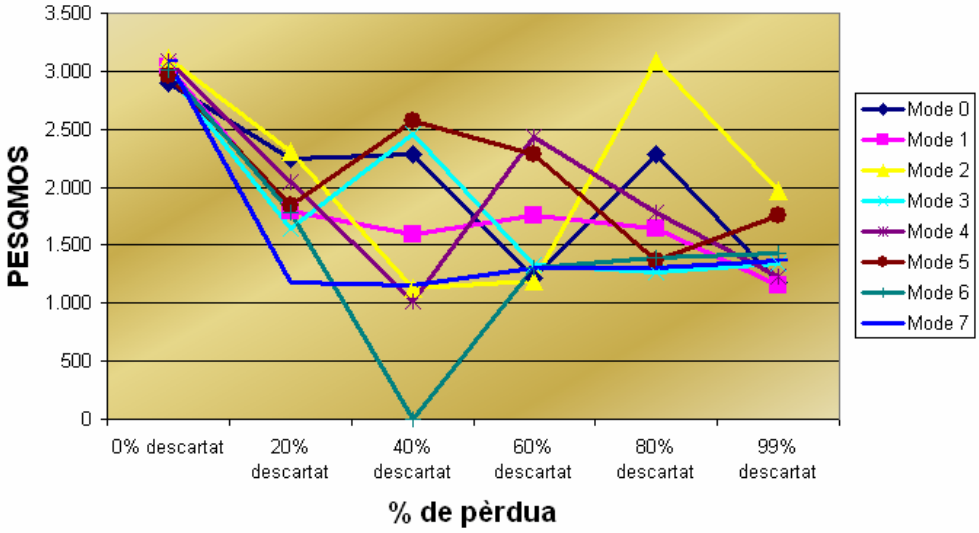


Fig 2.7 Coeficients PESQMOS vs % de pèrdua

S'aprecia que en general el mode que presenta un millor comportament als errors introduïts en el canal, és el 2.

En aquest cas per a baixes pèrdues (fins al 40%) els modes 6 i 7 d'alta taxa de transmissió, són els que presenten pitjor robustesa.

Per a altes pèrdues (a partir del 60%) els millors modes a utilitzar són els de baixa taxa de transmissió (tal i com ens passava en anteriorment), com per exemple el mode 0 i el 2.

3. INTEGRACIÓ DEL CÒDEC AMR EN L'APLICACIÓ RAT

3.1. Introducció

En aquest apartat es comentarà la integració pròpia del còdec AMR dins de l'aplicació de veu per videoconferència RAT. Les instruccions de codi empleades s'adjunten a l'annex 5.

3.2. Integració dels diferents modes del AMR dins el RAT

Per afegir al RAT les característiques pròpies del còdec AMR s'ha hagut de modificar el codi existent; pel que ha estat necessari l'estudi exhaustiu per una part de tot el codi del RAT i per a una altra del còdec AMR facilitat pel 3GPP[1].

Un cop estudiat el funcionament dels ambdós codis, s'han pogut incloure els diferents modes del còdec AMR a l'aplicació i crear una interfície que pogués interactuar amb el codi del còdec AMR, el de l'aplicació RAT i que compatibilitzés les estructures dades.

Mirant la part a on es pot triar els diferents còdecs que venen per defecte a la versió 4.2.26, s'ha inclòs les característiques pròpies de cada mode a la seva descripció.

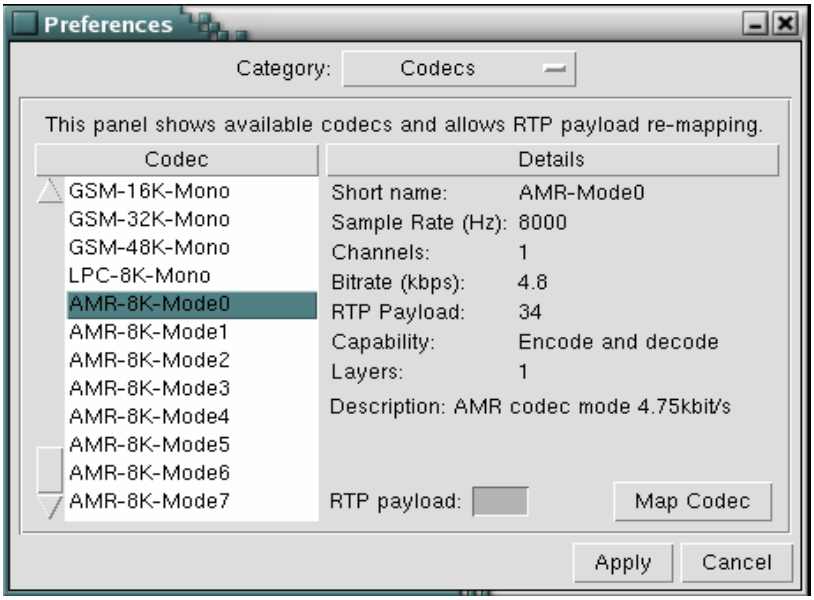


Fig. 3.1 Menú còdecs de preferències del RAT

[1] http://www.3gpp.org/

Per a cada mode AMR, s'ha hagut de definir un payload diferent en l'aplicació RAT. En la taula que segueix, es mostren els payloads estandarditzats en l'RFC1890:

Taula 3.1 Id payloads del RFC1890

PT	encoding name	audio/video (A/V)	clock rate (Hz)	channels (audio)
0	PCMU	A	8000	1
1	1016	A	8000	1
2	G721	A	8000	1
3	GSM	A	8000	1
4	unassigned	A	8000	1
5	DVI4	A	8000	1
6	DVI4	A	16000	1
7	LPC	A	8000	1
8	PCMA	A	8000	1
9	G722	A	8000	1
10	L16	A	44100	2
11	L16	A	44100	1
12	unassigned	A		
13	unassigned	A		
14	MPA	A	90000	(see text)
15	G728	A	8000	1
16--23	unassigned	A		
24	unassigned	V		
25	CelB	V	90000	
26	JPEG	V	90000	
27	unassigned	V		
28	nv	V	90000	
29	unassigned	V		
30	unassigned	V		
31	H261	V	90000	
32	MPV	V	90000	
33	MP2T	AV	90000	
34--71	unassigned	?		
72--76	reserved	N/A	N/A	N/A
77--95	unassigned	?		
96--127	dynamic	?		

Payload types (PT) for Standard audio and video encodings

Es va comprovar que els diferents payloads dels còdecs implementats al RAT cap estigués dins del marge 34 – 71 amb el que es trobaven no assignats, tal i com deia el RFC1890. Per tan es van escollir els identificadors de la següent manera:

Taula 3.2 Id's pels modes del còdec AMR

Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7
34	35	36	37	38	39	40	41

Segons l'RFC3267[2] els blocs codificats per a cada mode, tenen el nombre total de bits que figuren en la següent taula:

Taula 3.3 Bits total codificats

Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7
95 bits	103 bits	118 bits	134 bits	148 bits	159 bits	204 bits	244 bits

Tenint en compte aquestes dades i la informació esmentada anteriorment de l'AMR s'han definit dins el RAT els payloads de la següent forma

Taula 3.4 Definició dels payloads

Mode	Payload id	Freqüència de mostreig	Temps d'enviament	Bloc codificat
0	34	8000 Hz	20 ms	12 bytes
1	35	8000 Hz	20 ms	13 bytes
2	36	8000 Hz	20 ms	15 bytes
3	37	8000 Hz	20 ms	17 bytes
4	38	8000 Hz	20 ms	19 bytes
5	39	8000 Hz	20 ms	20 bytes
6	40	8000 Hz	20 ms	26 bytes
7	41	8000 Hz	20 ms	31 bytes

En el RAT s'ha creat la següent nova estructura, per poder tractar i reconèixer el nou còdec dins l'aplicació:

```
static codec_fns_t codec_table[] = {

      {         NULL,
                NULL,
                amr_get_formats_count,
                amr_get_format,
                amr_state_create,
                amr_state_destroy,
                amr_encoder,
                NULL,
                amr_state_create,
                amr_state_destroy,
                amr_decoder,
                NULL,
                NULL,
                amr_repair,
                NULL,
                NULL,
                NULL
        }
};
```

[2] http://www.faqs.org/rfcs/rfc3267.html

A on el servei de cada funció és el següent:

a) amr_get_format_count

Aquesta funció té com objectiu fer el comptatge de tots els modes que tingui aquest codec.

b) amr_get_format

Aquesta funció identifica el tipus de mode que s'està rebent, per a informar-ho al decodificador.

c) amr_state_create

En aquest punt es crea la "sessió" del mode AMR que s'hagi escollit.

d) amr_state_destroy

Destrueix la sessió que s'acabi de tancar del mode AMR.

e) amr_encoder

Es codifica la informació a enviar amb el mode seleccionat.

f) amr_decoder

Decodifica la informació rebuda segons el mode AMR.

La resta de funcions implementades pel correcte funcionament del codec AMR i definides en l'arxiu "cx_amr.c", es troben en l'annex 5.

Un cop compilada l'aplicació RAT amb el còdec integrat, clickant damunt de cada participant hi ha l'opció "Decoder" a on es pot comprovar quina codificació està fent servir el participant. Quan es canvia de mode AMR o de còdec, es pot veure com varia el camp "Audio encoding".

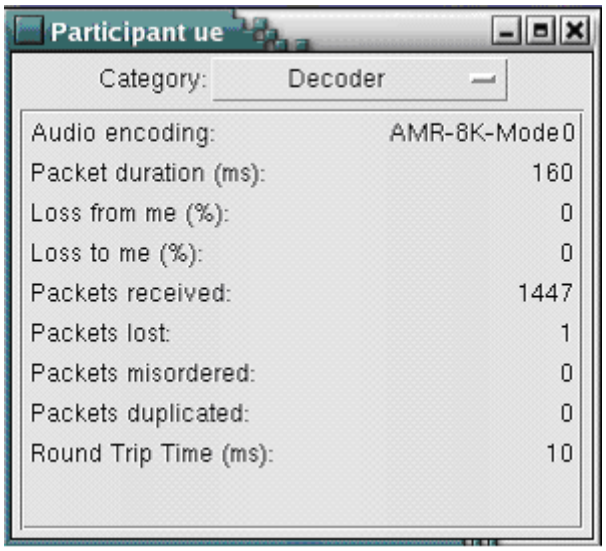


Fig 3.2 Decoder

3.3. Automatització de l'elecció del còdec segons les condicions del canal (Canvi del mode de forma externa)

Ja que el còdec AMR té que ser adaptable a les condicions del canal s'ha configurat un nou mode automàtic que va canviant de mode segons se li va indicant.

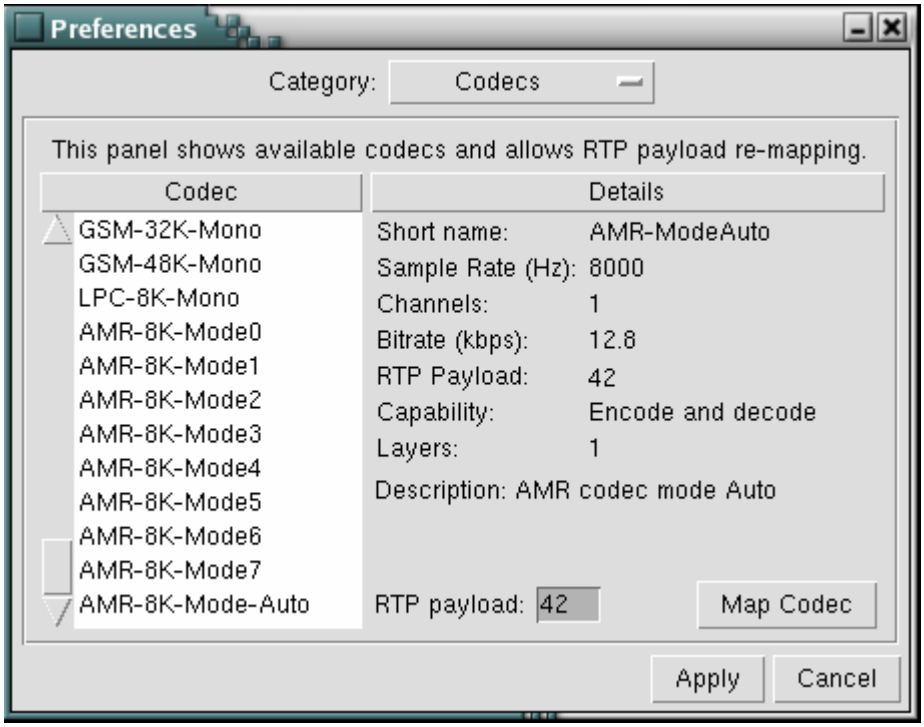


Fig. 3.3 Inclusió del nou mode automàtic

Per tan als modes existents se'ls hi afegeix un nou mode amb identificador de payload 42.

Taula 3.5 Nova organització d'ids per a la definició del payload AMR

Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7	**ModeAu**
34	35	36	37	38	39	40	41	**42**

Alguns sistemes tenen dificultats en el nombre de modes que poden suportar i cada quan es pot canviar de mode degut a problemes d'ample de banda i altres limitacions.

Amb l'objectiu de tenir una millor interoperativitat amb aquests sistemes interessa que es pugui canviar el mode AMR de forma externa. Perquè això sigui possible s'ha implementat en el RAT una forma molt senzilla de fer-ho.

Al codi del RAT se li ha afegit el fet de que estigui monitoritzant tota la estona un fitxer anomenat "canvia_codec". Aquest arxiu conté el valor del mode al que s'ha de transmetre. D'aquesta forma si es vol modificar el còdec AMR al que s'està transmeten, només hi haurà que canviar el numero contingut en aquest fitxer i posar el corresponent al mode que ens interessa. L'arxiu es llegit cada vegada que es fa una transmissió. Llavors si es modifica l'arxiu transmetrà amb el mode modificat.

Com que a la banda receptora es detecta el mode amb que s'està enviant la codificació, es commuta automàticament segons el "id" de payload que es rep.

4. CONCLUSIONS, ESTUDI D'AMBIENTALITZACIÓ i FUTURES MILLORES POSSIBLES

4.1. Conclusions

L'aplicació PESQ resulta de gran utilitat per poder fer mesures de la qualitat en un sistema de comunicació. Fent la sincronització entre arxiu original i degradat s'aconsegueix eliminar els efectes de retards indesitjables que s'afegeixen provocats pel mètode que es segueix al fer les proves.

Mitjançant l'aplicació PESQ i la integració del còdec AMR en el RAT, s'ha pogut comprovar que per a un canal de comunicacions que presenti una taxa de pèrdues important, és millor utilitzar-ne modes AMR de més baixa transmissió. Mentre que per situacions a on les pèrdues siguin més baixes, s'han d'utilitzar modes de més alta velocitat (si l'ample de banda del canal ho fa aconsellable) .

En el cas del RAT existeixen identificadors de payloads que es troben prèviament assignats, amb el que al fer la integració dels 8 modes del AMR i automàtic s'ha consultat el RFC1890 sobre perfils RTP per conferències d'àudio i vídeo per triar un rang que no estigués assignat.

El fet de que l'aplicació per videoconferència RAT sigui de programari lliure i tot el seu codi es té disponible ens ha permès el poder adaptar-la al nostre escenari particular, cosa que d'una altre manera no hagués estat possible.

S'ha pogut comprovar com tal i com es deia a les especificacions l'aplicació RAT, transmet en mitja d'un paquet cada 20 ms. Això dóna joc a una taxa de 50 paquets per segon.

4.2. Estudi d'ambientalització

Des del punt de vista mediambiental aquest projecte és bastant útil. S'engloba dins de la creació d'una plataforma de simulació, que permet evitar l'inconvenient que suposa el tenir que estar realitzant proves amb diferent tipus de persones, per saber la qualitat del canal en un entorn UMTS. Amb aquest entorn s'aconsegueix eludir els desplaçaments de persones i el muntatge d'infraestructura temporal per poder fer les proves. A la vegada amb aquest projecte es busca poder fer un estudi dels paràmetres més convenients en cada moment de la comunicació amb el que s'aconseguirà fer un millor ús dels recursos i evitar el tenir que invertir en nova infraestructura.

4.3. Futures millores possibles

Realitzar una aplicació similar a "Generador de gràfics" que permeti analitzar els retards que han sofert els diferents paquets durant la comunicació de veu. Així com poder determinar la pèrdua dels mateixos.

Un altre aspecte a considerar seria la possibilitat de poder canviar el mode de codificació AMR en el RAT de forma externa, però que aquest es pogués fer des d'una connexió per port UDP.

També seria interessant a l'hora de comprovar els efectes de la compressió damunt el canal de comunicació poder sincronitzar amb "syncok" altres tipus de formats diferents a .WAV.

5. BIBLIOGRAFÍA

[1] Format àudio .WAV:
http://ccrma.stanford.edu/CCRMA/Courses/422/projects/WaveFormat/

[2] PESQ
http://www.psytechnics.com/downloads/2001-P02.pdf
http://www.malden.co.uk/products/dsla/pesq.htm
http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T%2DREC%2DP%2E862%2D200102%2DI

[3] RAT
http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/

[4] AMR
http://www.auxio.org/projects/misc/mms/dencoder/26104-500.zip
http://www.faqs.org/rfcs/rfc3267.html

[5] ALGORITME
http://ieeexplore.ieee.org/iel5/9435/29943/01368312.pdf?isNumber=29943&prod=CNF&arnumber=1368312&arSt=+1812&ared=+1816&arAuthor=+Liebl%2C+G.%3B++Kaindl%2C+M.%3B++Wen+Xu

[6] UCL Network and Multimedia Research Group. MBone conferencing applications web page
http://www-mice.cs.ucl.ac.uk/multimedia/software/

[7] Bennett, R., "User Guide for RAT v4.2" Version 1 (DRAFT). University College London, Computer Science Department.

[8] Programa Ethereal.
http://www.ethereal.com

[9] User guide Ethereal.
http://www.ethereal.com/docs/user-guide/book1.html

[10] Hernando, J.M. and Lluch C., "Comunicaciones Móviles de Tercera Generación. UMTS." Telefónica Móviles España, 2ª Ed. (2001).

[11] ITU-T Rec. P. 800, "Métodos de determinación subjetiva de la
[12] calidad de transmisión", (1996).

[13] ITU-T Rec. P. 800.1, "Terminología de las notas medias de
[14] opinión", (2003).

[15] ITU-T Rec. P. 830, “Evaluación subjetiva de la calidad de funcionamiento de los códecs digitales de banda telefónica y de banda ancha”, (1996).

[16] ITU-T Rec. P. 862.1, “Mapping function for transforming P.862 raw result scores to MOS-LQO”, (2003).

[17] “PESQ: An introduction”. Psytechnics (2004).
Rix, A. W., Beerends, J. G., Hollier, M.P. and Hekstra, A. P., “Perceptual Evaluation of Speech Quality (PESQ) – a new method for speech quality assessment of telephone networks and codecs”. IEEE Internacional Conference on Acoustics, Speech and Signal Processing (ICASSP), Vol. 2, pp 749-752, Salt Lake City, USA, (2001).

ANNEXOS

TÍTOL DEL TFC/PFC: Estudi sobre la percepció de QoS en un demostrador UMTS (Part I)

TITULACIÓ: Enginyeria de Telecomunicació (segon cicle)

AUTOR: Luis Domínguez González
Iñaki Quer Alecha

DIRECTOR: Ramon Ferrús Ferré

DATA: 20 de gener de 2005

6. ANNEXES

NOTA

L'organització dels annexos s'ha intentat fer de tal manera que estiguin ordenats per capítols i segons es van comentant els diferents aspectes a la memòria. D' aquesta forma encara que no es citi expressament cada vegada a on es troba una descripció més en detall sobre el tema, el lector pot saber de forma ràpida on esta la informació a l'annex en funció de la part de la memòria que estigués llegint.

6.1. CAPÍTOL 1

6.1.1. ANNEX 1 "Codi Eina EXCEL"

```
Sub EixY()
Dim numero1 As Double
numero1 = Val(InputBox("Entra el Màxim de l'escala de l'eix Y", "Entrada de dades"))

With ActiveChart.Axes(xlValue)
        '.MinimumScaleIsAuto = True
        .MaximumScale = numero1
        '.MinorUnitIsAuto = True
        '.MajorUnitIsAuto = True
        '.Crosses = xlAutomatic
        '.ReversePlotOrder = False
        '.ScaleType = xlLinear
        '.DisplayUnit = xlNone
    End With

End Sub

Sub PkTemps()

Dim CelaTemps1 As Variant
Dim CelaTemps2 As Variant
Dim CelaEscala1 As Variant
Dim CelaEscala2 As Variant
Dim CelaEscala3 As Variant
Dim CelaResolucio1 As Variant
Dim CelaResolucio2 As Variant
Dim CelaPakets1 As Variant
Dim CelaPakets2 As Variant

Dim ValorTemps As Double
Dim ValorEscala As Double
Dim valorescala2 As Double
Dim ValorResoluicio As Double
Dim ValorPakets As Double

Dim numero1 As Double
Dim numero2 As Double
Dim numero3 As Double
Dim i As Long
Dim j As Long

Dim IncTemps As Double
Dim IncEscala As Double
Dim IncPakets As Double

Dim Valormaxdad1 As Double
Dim Valormaxdad2 As Double
```

```
Dim actiu As Integer
Dim actiu2 As Integer
Dim valorprotointer As Variant
Dim text As Variant

Application.Run "resolució_2"
Application.Run "PkTemps_2"

Sheets("Càlculs6").Select
Range("e6").Select
valormax = ActiveCell.Value

Sheets("Menú Principal").Select

numero1 = -1

text = "El Rang de l'escala va de 0 - " & valormax & " Segons"
MsgBox text

valormax = valormax * 1000

Do While numero1 = -1

text = "Valor inicial de l'escala? (ms) mín->0"

numero1 = Val(InputBox(text, "Entrada de dades"))
Loop

If numero1 = 0 Then numero2 = 0 Else numero2 = numero1 / 1000

numero1 = -1

text = "Valor final de l'escala? (ms) Màx. ->" & valormax

Do While numero1 = -1

numero1 = Val(InputBox(text, "Entrada de dades"))
Loop

If numero1 = 0 Then numero3 = valormax / 1000 Else numero3 = numero1 / 1000


numero1 = -1

Do While numero1 = -1

numero1 = Val(InputBox("Quina resolució vols? (ms) (per defecte 1000)", "Entrada de
dades"))
Loop

If numero1 = 0 Then numero1 = 1 Else numero1 = numero1 / 1000

Sheets("Càlculs6").Select

Range("b2").Select
ActiveCell.FormulaR1C1 = numero2

Range("c2").Select
ActiveCell.FormulaR1C1 = numero1


Range("g2").Select
ActiveCell.FormulaR1C1 = numero3


Range("e5").Select
ActiveCell.FormulaR1C1 = numero1


Range("e2").Select
Valormaxdad1 = ActiveCell.text

CelaTemps1 = "A"
CelaEscala1 = "B"
CelaResolucio1 = "C"
CelaPakets1 = "D"
```

```
IncTemps = 2
IncEscala = 2
Incresolucio = 2
IncPakets = 2

CelaTemps2 = CelaTemps1 & IncTemps
CelaEscala2 = CelaEscala1 & IncEscala
CelaPakets2 = CelaPakets1 & IncPakets
CelaResolucio2 = CelaResolucio1 & Incresolucio

Range("A2").Select
ActiveCell.FormulaR1C1 = "=Càlculs!R[1]C[3]"
CelaTemps2 = "a2:a" & Valormaxdad1
Selection.AutoFill Destination:=Range("A2:A63"), Type:=xlFillDefault


Range("e4").Select
Valormaxdad1 = ActiveCell.text

If Valormaxdad1 > 65530 Then
    MsgBox "No hi han suficients cel·les, per fer aquets càlcul. Canvia els paràmetres"
    Application.Run "pktemps"
Exit Sub
End If

Range("B3").Select
ActiveCell.FormulaR1C1 = "=R5C[3]+R[-1]C"
CelaEscala2 = "B3:B" & Valormaxdad1 + 2
Selection.AutoFill Destination:=Range(CelaEscala2), Type:=xlFillDefault


Range("D2").Select
ActiveCell.FormulaR1C1 = "0"
CelaEscala2 = "d2:d" & Valormaxdad1 + 2
Selection.AutoFill Destination:=Range(CelaEscala2), Type:=xlFillDefault

Range("G4").Select
ValorCela1 = ActiveCell.Value

cela2 = "b" & ValorCela1 & ":b65536"
Range(cela2).Select
Selection.ClearContents

cela2 = "d" & ValorCela1 & ":d65536"
Range(cela2).Select
Selection.ClearContents

cela2 = "h3" & ValorCela1 & ":q65536"
Range(cela2).Select
Selection.ClearContents

Range("e4").Select
Valormaxdad2 = ActiveCell.text

Range("e2").Select
Valormaxdad1 = ActiveCell.text

IncEscala = 2
CelaEscala2 = CelaEscala1 & IncEscala
CelaEscala3 = CelaEscala1 & IncEscala + 2

IncTemps = 2
CelaTemps2 = CelaTemps1 & IncTemps

IncPakets = 2
CelaPakets2 = CelaPakets1 & IncPakets

Sheets("Dades Totals").Select
CelaProto2 = "k" & 2
Range(CelaProto2).Select
valorprotointer = ActiveCell.text
Sheets("Càlculs6").Select

actiu = 0

CelaTemps2 = CelaTemps1 & 2
```

```
For i = 1 To Valormaxdad1 - 1

Sheets("TEXT").Select
Sheets("Càlculs6").Select
Range(CelaTemps2).Select
ValorTemps = ActiveCell.text

Sheets("Càlculs6").Select

Range("g4").Select

If IncPakets = ActiveCell.text - 1 Then
i = Valormaxdad1
End If


    For j = 1 To Valormaxdad2

        Range(CelaEscala2).Select


        If ActiveCell.text = "" Then
        j = 70000
        Else

        ValorEscala = ActiveCell.text

        CelaEscala3 = CelaEscala1 & IncEscala + 1

        Range(CelaEscala3).Select
        If ActiveCell.text = "" Then
        j = 70000
        Else

        valorescala2 = ActiveCell.text


        End If

        If ValorTemps <= ValorEscala Then

        If ValorProto = " ICMP " Then ValorTemps = ValorEscala * 2
        If ValorProto = " IGMP " Then ValorTemps = ValorEscala * 2

            If ValorEscala >= ValorTemps Then
                Range(CelaPakets2).Select

                ValorPakets = ActiveCell.text

                ActiveCell.FormulaR1C1 = ValorPakets + 1


                j = Valormaxdad2


                End If
        End If

        IncEscala = IncEscala + 1
        CelaEscala2 = CelaEscala1 & IncEscala

        IncPakets = IncPakets + 1
        CelaPakets2 = CelaPakets1 & IncPakets

        End If


    Next

    'IncEscala = 2
    IncEscala = IncEscala - 1
    CelaEscala2 = CelaEscala1 & IncEscala

    IncTemps = IncTemps + 1
    CelaTemps2 = CelaTemps1 & IncTemps

    'IncPakets = 2
```

```
    IncPakets = IncPakets - 1
    CelaPakets2 = CelaPakets1 & IncPakets

Next


    Range("H2:Q2").Select
    Selection.Copy
    text = "h3:q" & IncPakets
    Range(text).Select
    ActiveSheet.Paste


    ActiveCell.FormulaR1C1 = "='Dades Totals'!RC[-13]+RC[-16]"
    Range("R2").Select
    Selection.Copy

    text = "R3:R" & IncPakets
    Range(text).Select
    ActiveSheet.Paste


Sheets("Dades Dist.t").Select
ActiveSheet.PivotTables("Tabla dinámica3").RefreshTable
Sheets("Gràfic Dist.t").Select

End Sub

Sub PkTemps_2()

Dim CelaTemps1 As Variant
Dim CelaTemps2 As Variant
Dim CelaEscala1 As Variant
Dim CelaEscala2 As Variant
Dim CelaEscala3 As Variant
Dim CelaResolucio1 As Variant
Dim CelaResolucio2 As Variant
Dim CelaPakets1 As Variant
Dim CelaPakets2 As Variant

Dim ValorTemps As Double
Dim ValorEscala As Double
Dim valorescala2 As Double
Dim ValorResoluicio As Double
Dim ValorPakets As Double


Dim numero1 As Double
Dim i As Long
Dim j As Long

Dim IncTemps As Double
Dim IncEscala As Double
Dim IncPakets As Double

Dim Valormaxdad1 As Double
Dim Valormaxdad2 As Double

numero1 = 1000

numero1 = numero1 / 1000

Sheets("Càlculs6").Select
Range("e5").Select
ActiveCell.FormulaR1C1 = numero1

Range("a2:a2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("a2:a65536").Select
Selection.ClearContents

Range("b3:b3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("b3:b65536").Select
Selection.ClearContents

Range("d2:d2").Select
```

```
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("d2:d65536").Select
Selection.ClearContents


Range("h1:ax1").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("h3:ax65536").Select
Selection.ClearContents

Range("b2").Select
ActiveCell.FormulaR1C1 = 0


Range("e2").Select
Valormaxdad1 = ActiveCell.text

CelaTemps1 = "A"
CelaEscala1 = "B"
CelaResolucio1 = "C"
CelaPakets1 = "D"



IncTemps = 2
IncEscala = 2
Incresolucio = 2
IncPakets = 2

CelaTemps2 = CelaTemps1 & IncTemps
CelaEscala2 = CelaEscala1 & IncEscala
CelaPakets2 = CelaPakets1 & IncPakets
CelaResolucio2 = CelaResolucio1 & Incresolucio


Range("A2").Select
ActiveCell.FormulaR1C1 = "=Càlculs!R[1]C[3]"
CelaTemps2 = "a2:a" & Valormaxdad1
Selection.AutoFill Destination:=Range(CelaTemps2), Type:=xlFillDefault


Range("e4").Select
Valormaxdad1 = ActiveCell.text


Range("B3").Select
ActiveCell.FormulaR1C1 = "=R5C[3]+R[-1]C"
CelaEscala2 = "B3:B" & Valormaxdad1 + 2
Selection.AutoFill Destination:=Range(CelaEscala2), Type:=xlFillDefault


Range("D2").Select
ActiveCell.FormulaR1C1 = "0"
Selection.Copy
CelaEscala2 = "d2:d" & Valormaxdad1 + 2
Range(CelaEscala2).Select
ActiveSheet.Paste



Sheets("Menú Principal").Select

End Sub

Sub Vi()

Dim CelaVi1 As Variant
Dim CelaVi2 As Variant
Dim celatamany1 As Variant
Dim celatamany2 As Variant
Dim CelaTamany3 As Variant
Dim CelaEscala1 As Variant
Dim CelaEscala2 As Variant
Dim CelaEscala3 As Variant
Dim CelaEscala4 As Variant
Dim CelaLlegenda1 As Variant
Dim CelaLlegenda2 As Variant
```

```
Dim CelaLlegenda3 As Variant
Dim CelaLlegenda4 As Variant
Dim CelaComptar1 As Variant
Dim CelaComptar2 As Variant


Dim ValorVi1 As Double
Dim ValorVi2 As Double
Dim ValorTamany1 As Double
Dim ValorTamany2 As Double
Dim ValorEscala As Double
Dim ValorResolucio As Double


Dim IncVi As Long
Dim Inctamany As Long
Dim IncEscala As Long

Dim numero1 As Long



Application.Run "resolució"
Application.Run "Vi_Inici"

ActiveWindow.ScrollWorkbookTabs Sheets:=1
'Sheets("Gràfic Vi").Select


Sheets("Càlculs5").Select
Range("a2:a2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("a2:a65536").Select
Selection.ClearContents

Range("b2:b2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("b2:b65536").Select
Selection.ClearContents

Range("c2:c2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("c2:c65536").Select
Selection.ClearContents

Range("e2:e2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("e2:e65536").Select
Selection.ClearContents

Range("f3:f3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("f3:f65536").Select
Selection.ClearContents



CelaVi1 = "A"
celatamany1 = "J"
CelaEscala1 = "I"
CelaEscala3 = "B"
CelaLlegenda1 = "C"
CelaLlegenda3 = "E"
CelaComptar1 = "F"

IncVi = 2
Inctamany = 2
IncEscala = 2


CelaVi2 = CelaVi1 & IncVi
celatamany2 = celatamany1 & Inctamany
CelaTamany3 = celatamany1 & Inctamany + 1
CelaEscala2 = CelaEscala1 & IncEscala
CelaEscala4 = CelaEscala3 & IncVi
CelaLlegenda2 = CelaLlegenda1 & IncVi
CelaLlegenda4 = CelaLlegenda3 & IncVi
```

```
CelaComptar2 = CelaComptar1 & IncVi + 1

Sheets("Càlculs").Select
Range(CelaEscala2).Select
ValorEscala = ActiveCell.text

Range("h2").Select
ValorResolucio = ActiveCell.text
ValorResolucio = ValorResolucio / 1000

Sheets("Càlculs5").Select

Range("g1").Select
ValorVi1 = ActiveCell.text

'Velocitats instantànies
Range("A2").Select
ActiveCell.FormulaR1C1 = _
"=(RC[8]+R[1]C[8])/((R[1]C[7]-RC[7])*(Càlculs!R2C[7]/1000))"
'"=(Càlculs!RC[9]+Càlculs!R[1]C[9])/(Càlculs!R2C[7]/1000)"

CelaVi2 = "a2:a" & ValorVi1 + 1
Selection.AutoFill Destination:=Range(CelaVi2), Type:=xlFillDefault

'Escala
Range("B2").Select
ActiveCell.FormulaR1C1 = "=Càlculs!RC[7]"
CelaVi2 = "b2:b" & ValorVi1 + 1
Selection.AutoFill Destination:=Range(CelaVi2), Type:=xlFillDefault

'Llegenda1
Range("c2").Select
ActiveCell.FormulaR1C1 = "VI"
Selection.Copy
Application.CutCopyMode = False
CelaVi2 = "c2:c" & ValorVi1 + 1
Selection.AutoFill Destination:=Range(CelaVi2), Type:=xlFillDefault



'Llegenda2
Range("E2").Select
ActiveCell.FormulaR1C1 = "VX"
Selection.Copy
Application.CutCopyMode = False
CelaVi2 = "e2:e" & ValorVi1 + 1
Selection.AutoFill Destination:=Range(CelaVi2), Type:=xlFillDefault


'N° de pakets
Range("f2").Select
ActiveCell.FormulaR1C1 = "=COUNTIF(C[-5],RC[-5])"
CelaVi2 = "f2:f" & ValorVi1 + 1
Selection.AutoFill Destination:=Range(CelaVi2), Type:=xlFillDefault


Range("t2").Select
text = "j3:s" & ActiveCell.text

Range("J2:S2").Select
Selection.Copy
Range(text).Select
ActiveSheet.Paste


'Do While ValorEscala > 0
'Sheets("Càlculs").Select
'Range(CelaEscala2).Select
'If ActiveCell.Text = "" Then
'ValorEscala = -1

'Else

'ValorEscala = ActiveCell.Text

'Range(celatamany2).Select
'ValorTamany1 = ActiveCell.Text
```

```
'Range(CelaTamany3).Select
'ValorTamany2 = ActiveCell.Text

'Sheets("Càlculs5").Select
'Range(CelaVi2).Select
'ActiveCell.FormulaR1C1 = (ValorTamany1 + ValorTamany2) / ValorResolucio

'Range(CelaEscala4).Select
'ActiveCell.FormulaR1C1 = ValorEscala

'Range(CelaLlegenda2).Select
'ActiveCell.FormulaR1C1 = "VI"

'Range(CelaLlegenda4).Select
'ActiveCell.FormulaR1C1 = "VX"

'Range(CelaComptar2).Select
'ActiveCell.FormulaR1C1 = "=COUNTIF(C[-5],RC[-5])"

'IncEscala = IncEscala + 1
'Inctamany = Inctamany + 1
'IncVi = IncVi + 1


'CelaEscala2 = CelaEscala1 & IncEscala
'celatamany2 = celatamany1 & Inctamany
'CelaTamany3 = celatamany1 & Inctamany
'CelaVi2 = CelaVi1 & IncVi
'CelaEscala4 = CelaEscala3 & IncVi
'CelaLlegenda2 = CelaLlegenda1 & IncVi
'CelaLlegenda4 = CelaLlegenda3 & IncVi
'CelaComptar2 = CelaComptar1 & IncVi


'End If
'Loop

Sheets("Dades NVx·Vx").Select
ActiveSheet.PivotTables("Tabla dinámica2").RefreshTable

Sheets("Dades Vi").Select
ActiveSheet.PivotTables("Tabla dinámica1").RefreshTable
Sheets("Gràfic Vi").Select

    ActiveChart.ChartArea.Select
    ActiveChart.SeriesCollection(1).Select
    With Selection.Border
        .ColorIndex = 57
        .Weight = xlThick
        .LineStyle = xlContinuous
    End With
    With Selection
        .MarkerBackgroundColorIndex = xlAutomatic
        .MarkerForegroundColorIndex = xlAutomatic
        .MarkerStyle = xlDiamond
        .Smooth = True
        .MarkerSize = 12
        .Shadow = True
    End With
    ActiveChart.ChartArea.Select


    ActiveChart.ChartTitle.Select
    Selection.Characters.text = "Velocitats Instantànies (paquets TCP i UDP)"
    Selection.AutoScaleFont = False
    With Selection.Characters(Start:=1, Length:=43).Font
        .Name = "Arial"
        .FontStyle = "Negreta"
        .Size = 12
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
    End With
```

```
    ActiveChart.ChartArea.Select


End Sub

Sub Sortir()

    Sheets("Menú Principal").Select
    Application.DisplayFullScreen = True

End Sub

Sub anaraNvx()

    ActiveWindow.ScrollWorkbookTabs Sheets:=1
    ActiveWindow.ScrollWorkbookTabs Sheets:=1
    ActiveWindow.ScrollWorkbookTabs Sheets:=1
    Sheets("Dades NVx·Vx").Select
    Range("B10").Select
    ActiveSheet.PivotTables("Tabla dinámica2").PivotCache.Refresh
    ActiveWindow.ScrollWorkbookTabs Sheets:=-1
    ActiveWindow.ScrollWorkbookTabs Sheets:=-1
    Sheets("Gràfic NVx·Vx").Select
    Application.Run "Suavitzar"
    ActiveChart.Deselect
End Sub

Sub Omple_Càlculs()

Dim Valorcelacomptades As Variant
Dim CelaComptades As Variant
Dim CelaActual As Variant

CelaComptades = "R" & "2"
Range(CelaComptades).Select
Valorcelacomptades = ActiveCell.text


'Comptem el n° de pakets
Range("B3").Select
ActiveCell.FormulaR1C1 = "=R[-1]C+1"
CelaActual = "B3" & ":" & "B" & (Valorcelacomptades + 1)
Selection.AutoFill Destination:=Range(CelaActual), Type:=xlFillDefault


'Agafem les dades de segons
Range("C2").Select
ActiveCell.FormulaR1C1 = "='Dades Totals'!RC[2]/1000000000"
CelaActual = "C2" & ":" & "C" & (Valorcelacomptades + 1)
Selection.AutoFill Destination:=Range(CelaActual), Type:=xlFillDefault

'Agafem les dades de tamany
Range("E2").Select
ActiveCell.FormulaR1C1 = "='Dades Totals'!RC[-2]"
CelaActual = "E2" & ":" & "E" & (Valorcelacomptades + 1)
Selection.AutoFill Destination:=Range(CelaActual), Type:=xlFillDefault


'Agafem les dades de l'hora


    Range("o2").Select
    Selection.ClearContents
    ActiveCell.FormulaR1C1 = _
        "=IF('Dades Totals'!RC[-1]=""""," """,'Dades Totals'!RC[-1])"
    Range("O2").Select
    Selection.Copy
    CelaActual = "o2" & ":" & "o" & (Valorcelacomptades + 1)
    Range(CelaActual).Select
    ActiveSheet.Paste

'Agafem les dades del minut
Range("p2").Select
Selection.ClearContents
ActiveCell.FormulaR1C1 = _
    "=IF('Dades Totals'!RC[-1]=""""," """,'Dades Totals'!RC[-1])"
```

```
Range("P2").Select
Selection.Copy
CelaActual = "p2" & ":" & "p" & (Valorcelacomptades + 1)
Range(CelaActual).Select
ActiveSheet.Paste



'Agafem les dades Hora Total
CelaActual = "A" & 2
Range(CelaActual).Select
ActiveCell.FormulaR1C1 = "=RC[14] & "":"" & RC[15] & "":"" &RC[2]"
CelaActual = "A2" & ":" & "A" & (Valorcelacomptades + 1)
Selection.AutoFill Destination:=Range(CelaActual), Type:=xlFillDefault


End Sub


Sub resolució()
'
'Variables per calcular eix de temps, amb la resolució escollida
Dim numero1 As Double
Dim numero2 As Double
Dim numero3 As Double
Dim valorcelax1 As Variant
Dim valorcelax2 As Variant


Dim text As Variant


Dim Max1 As Long
Dim Max2 As Long

Dim Cela1 As Variant
Dim cela2 As Variant
Dim Celacomodi As Variant
Dim Celahora1 As Variant
Dim Celahora2 As Variant
Dim Celaminut1 As Variant
Dim Celaminut2 As Variant

Dim ValorCelaminut As Long
Dim ValorCelaminut2 As Long

Dim ValorCelahora As Long
Dim ValorCelahora2 As Long

Dim ValorCelatamany As Long


Dim ValorCela1 As Double
Dim ValorCela3 As Double

Dim ValorCela1b As Double
Dim ValorCela2b As Double
Dim Valorcela3b As Double



Dim valor As Variant
Dim Valor2 As Variant
Dim Valor3 As Variant
Dim valor0 As Variant

Dim i As Long
Dim j As Long


'Variables per determinar la posició de cada paquet
Dim Celapaquet1 As Variant
Dim Celapaquet2 As Variant
Dim Celascala1 As Variant
Dim Celascala2 As Variant
Dim Celascala3 As Variant
Dim celatamany1 As Variant
```

```
Dim celatamany2 As Variant
Dim CelaTemps1 As Variant
Dim CelaTemps2 As Variant
Dim ValorCela2 As Variant
Dim valormax As Double
Dim valorprotointer As Variant
Dim celainter As Variant
Dim actiu As Integer
Dim actiu2 As Integer


Dim ValorCelascala3 As Double
Dim compara As Double

Application.Run "resolució_2"

Sheets("Càlculs").Select
Range("s2").Select
valormax = ActiveCell.Value
'valormax = valormax + 1

Sheets("Menú Principal").Select

numero1 = -1

text = "El Rang de l'escala va de 0 - " & valormax & " Segons"
MsgBox text

valormax = valormax * 1000

Do While numero1 = -1

text = "Valor inicial de l'escala? (ms) mín->0"

numero1 = Val(InputBox(text, "Entrada de dades"))
Loop

numero2 = numero1 / 1000

numero1 = -1

text = "Valor final de l'escala? (ms) Màx. ->" & valormax

Do While numero1 = -1

numero1 = Val(InputBox(text, "Entrada de dades"))
Loop

If numero1 = 0 Then numero3 = valormax / 1000 Else numero3 = numero1 / 1000


numero1 = -1

Do While numero1 = -1

numero1 = Val(InputBox("Quina resolució vols? (ms) (per defecte->1000)", "Entrada de
dades"))
Loop

If numero1 = 0 Then numero1 = 1000

Sheets("Càlculs").Select
Range("I3:I3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("I3:I65536").Select
Selection.ClearContents

Sheets("Càlculs").Select
Range("j3:j3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("j3:j65536").Select
Selection.ClearContents


Application.Run "Omple_Càlculs"

Sheets("Càlculs").Select
```

```
Range("i2").Select
ActiveCell.FormulaR1C1 = numero2

Range("H2").Select
ActiveCell.FormulaR1C1 = numero1

Max1 = 65536
Cela1 = "C"
valor = 2
cela2 = Cela1 & valor

Celaminut1 = "P"
Celaminut2 = Celaminut1 & valor

Celahora1 = "O"
Celahora2 = Celahora1 & valor

CelaTemps1 = "d"
CelaTemps2 = CelaTemps1 & valor
Celacomodi = CelaTemps2


Range(CelaTemps2).Select
ActiveCell.FormulaR1C1 = 0
Valor2 = valor
CelaTemps2 = CelaTemps1 & valor + 1



    Range("n2").Select
Max2 = ActiveCell.Value


Celapaquet1 = "d"
Valor3 = 2
Celapaquet2 = Celapaquet1 & Valor3




Max1 = 65536


Cela1 = "R"
valor = 2
cela2 = Cela1 & valor

Range(cela2).Select
ValorCela1 = ActiveCell.Value

Range("D3").Select
ActiveCell.FormulaR1C1 = _
        "=IF(AND(R[-1]C15=RC15,R[-1]C16=RC16),R[-1]C4+(RC3-R[-1]C3),R[-1]C4+(60-R[-
1]C3)+RC3)"
cela2 = "D3:D" & ValorCela1 + 1
Selection.AutoFill Destination:=Range(cela2), Type:=xlFillDefault

Range("r3").Select
ValorCela1 = ActiveCell.Value

If ValorCela1 > 65530 Then
MsgBox "No hi han suficients cel·les, per fer aquets càlcul. Canvia els paràmetres"
Application.Run "resolució"
Exit Sub
End If

Range("I3").Select
ActiveCell.FormulaR1C1 = "=R[-1]C+R2C[-1]/1000"
Selection.Copy
cela2 = "I3:I" & ValorCela1 + 2
Range(cela2).Select
ActiveSheet.Paste


Range("J2").Select
ActiveCell.FormulaR1C1 = "0"
Selection.Copy
```

```
cela2 = "j2:j" & ValorCela1 + 2
Range(cela2).Select
ActiveSheet.Paste

Range("m3").Select
ActiveCell.FormulaR1C1 = numero3 + (numero1 / 1000)

Range("i2").Select
ActiveCell.FormulaR1C1 = numero2

Range("m4").Select
ValorCela1 = ActiveCell.Value

cela2 = "i" & ValorCela1 & ":i65536"
Range(cela2).Select
Selection.ClearContents

cela2 = "j" & ValorCela1 & ":j65536"
Range(cela2).Select
Selection.ClearContents

cela2 = "u" & ValorCela1 & ":ad65536"
Range(cela2).Select
Selection.ClearContents



    Range("m2").Select
    Max1 = ActiveCell.Value

    Range("n2").Select
    Max2 = ActiveCell.Value

    Cela1 = "D"
    valor = 2
    cela2 = Cela1 & valor

    Celascala1 = "I"
    valor0 = 2
    Celascala2 = Celascala1 & valor0
    Celascala3 = Celascala1 & valor0 + 1

    Celapaquet1 = "J"
    Valor2 = 2
    Celapaquet2 = Celapaquet1 & Valor2

    celatamany1 = "E"
    Valor3 = 2
    celatamany2 = celatamany1 & Valor3


    Dim para As Variant
    para = 1

    Sheets("Càlculs").Select
    Range("m4").Select
    para = ActiveCell.text + 1

    Valor2 = 2
    Celapaquet2 = Celapaquet1 & Valor2

    Sheets("Dades Totals").Select
    CelaProto2 = "k" & valor
    Range(CelaProto2).Select
    valorprotointer = ActiveCell.text
    Sheets("Càlculs").Select

    actiu = 0
    For i = 1 To Max2
        Sheets("TEXT").Select
        Sheets("Càlculs").Select
        Range(cela2).Select
        ValorCela1 = ActiveCell.Value

        If valor0 = para Then
        i = Max2
```

```
        End If

        Range("m3").Select

        If ValorCela2 > ActiveCell.text Then
        i = Max2
        End If


    For j = 0 To Max1
         'Sheets("TEXT").Select
         Sheets("Càlculs").Select
         Range(Celascala2).Select
         ValorCela2 = ActiveCell.Value

         If ValorCela2 = "" Then
        i = Max2
        End If


        If ValorProto = " ICMP " Then ValorCela2 = ValorCela1
        If ValorProto = " IGMP " Then ValorCela2 = ValorCela1

        If ValorCela1 < ValorCela2 Then

            If ValorCela2 = 0 Then ValorCela2 = 0.01
            If ValorCela1 < 0 Then ValorCela1 = ValorCela1 * (-1)
            If (ValorCela1 / ValorCela2) >= 1 & (ValorCela1 / ValorCela2) < 9 Then

            Range(celatamany2).Select
            ValorCela3 = ActiveCell.Value


            Range(Celapaquet2).Select
            ValorCelatamany = ActiveCell.text

            ActiveCell.FormulaR1C1 = ValorCela3 + ValorCelatamany


            Valor3 = Valor3 + 1
            celatamany2 = celatamany1 & Valor3


            j = Max1
            End If


        End If

            Valor2 = Valor2 + 1
            Celapaquet2 = Celapaquet1 & Valor2


            valor0 = valor0 + 1
            Celascala2 = Celascala1 & valor0


        Next



            'valor0 = 2
            valor0 = valor0 - 1
            Celascala2 = Celascala1 & valor0

            'Valor2 = 2
            Valor2 = Valor2 - 1
            Celapaquet2 = Celapaquet1 & Valor2

             valor = valor + 1
             cela2 = Cela1 & valor


    Next

    Range("G2").Select
    Selection.ClearContents
```

```
    ActiveCell.FormulaR1C1 = "0"
    Range("G3").Select
    ActiveCell.FormulaR1C1 = "=(RC[2]+R2C[-4])*1000000000"
    Range("G3").Select
    Selection.Copy


    cela2 = "g4:g" & para - 2
    Range(cela2).Select
    ActiveSheet.Paste


    Range("u2:AI2").Select
    Selection.Copy

    text = "u3:ai" & para - 2
    Range(text).Select
    ActiveSheet.Paste


    Sheets("Dades IncT").Select
    Range("C13").Select
    ActiveSheet.PivotTables("Tabla dinámica1").RefreshTable
    Sheets("Gràfic IncT").Select
End Sub

Sub resolució_2()

'Variables per calcular eix de temps, amb la resolució escollida
Dim numero1 As Double
Dim numero2 As Double
Dim numero3 As Double
Dim valorcelax1 As Variant
Dim valorcelax2 As Variant


Dim text As Variant


Dim Max1 As Long
Dim Max2 As Long

Dim Cela1 As Variant
Dim cela2 As Variant
Dim Celacomodi As Variant
Dim Celahora1 As Variant
Dim Celahora2 As Variant
Dim Celaminut1 As Variant
Dim Celaminut2 As Variant

Dim ValorCelaminut As Long
Dim ValorCelaminut2 As Long

Dim ValorCelahora As Long
Dim ValorCelahora2 As Long

Dim ValorCelatamany As Long


Dim ValorCela1 As Double
Dim ValorCela3 As Double

Dim ValorCela1b As Double
Dim ValorCela2b As Double
Dim Valorcela3b As Double



Dim valor As Variant
Dim Valor2 As Variant
Dim Valor3 As Variant
Dim valor0 As Variant

Dim i As Long
Dim j As Long
```

```
'Variables per determinar la posició de cada paquet
Dim Celapaquet1 As Variant
Dim Celapaquet2 As Variant
Dim Celascala1 As Variant
Dim Celascala2 As Variant
Dim Celascala3 As Variant
Dim celatamany1 As Variant
Dim celatamany2 As Variant
Dim CelaTemps1 As Variant
Dim CelaTemps2 As Variant
Dim ValorCela2 As Double

Dim ValorCelascala3 As Double
Dim compara As Double


Sheets("Càlculs").Select
Range("s2").Select
'numero2 = ActiveCell.text

Sheets("Menú Principal").Select

numero1 = 10000

Sheets("Càlculs").Select

Range("a2:a2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("a2:a65536").Select
Selection.ClearContents

Range("b2:b2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("b3:b65536").Select
Selection.ClearContents

Range("c2:c2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("c2:c65536").Select
Selection.ClearContents

Range("d2:d2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("d2:d65536").Select
Selection.ClearContents

Range("e2:e2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("e2:e65536").Select
Selection.ClearContents


Range("I3:I3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("I3:I65536").Select
Selection.ClearContents

Range("g3:g3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("g3:I65536").Select
Selection.ClearContents

Sheets("Càlculs").Select
Range("j3:j3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("j3:j65536").Select
Selection.ClearContents

Range("u3:Ax3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("u3:ax65536").Select
Selection.ClearContents

Application.Run "Omple_Càlculs"


   Sheets("Càlculs").Select
```

```
    Range("H2").Select
    ActiveCell.FormulaR1C1 = numero1


    Max1 = 65536
    Cela1 = "C"
    valor = 2
    cela2 = Cela1 & valor

    Celaminut1 = "P"
    Celaminut2 = Celaminut1 & valor

    Celahora1 = "O"
    Celahora2 = Celahora1 & valor



    CelaTemps1 = "d"
    CelaTemps2 = CelaTemps1 & valor
    Celacomodi = CelaTemps2



Sheets("Càlculs").Select
   Range("i2").Select
   ActiveCell.FormulaR1C1 = 0



    Range(CelaTemps2).Select
    ActiveCell.FormulaR1C1 = 0
    Valor2 = valor
    CelaTemps2 = CelaTemps1 & valor + 1



    Range("n2").Select
    Max2 = ActiveCell.Value


    Celapaquet1 = "d"
    Valor3 = 2
    Celapaquet2 = Celapaquet1 & Valor3



    Max1 = 65536


    Cela1 = "R"
    valor = 2
    cela2 = Cela1 & valor

    Range(cela2).Select
    ValorCela1 = ActiveCell.Value

    Range("D3").Select
    ActiveCell.FormulaR1C1 = _
        "=IF(AND(R[-1]C15=RC15,R[-1]C16=RC16),R[-1]C4+(RC3-R[-1]C3),R[-1]C4+(60-R[-
1]C3)+RC3)"
    cela2 = "D3:D" & ValorCela1 + 1
    Selection.AutoFill Destination:=Range(cela2), Type:=xlFillDefault




    Range("r3").Select
    ValorCela1 = ActiveCell.Value


    Range("I3").Select
    ActiveCell.FormulaR1C1 = "=R[-1]C+R2C[-1]/1000"
    Selection.Copy
    cela2 = "I3:I" & ValorCela1 + 3
```

```
    Range(cela2).Select
    ActiveSheet.Paste




    Sheets("Menú Principal").Select
End Sub

Sub Inct()


Dim Celaretard1 As Variant
Dim Celaretard2 As Variant
Dim CelaEscala1 As Variant
Dim CelaEscala2 As Variant
Dim CelaEscala3 As Variant
Dim CelaResolucio1 As Variant
Dim CelaResolucio2 As Variant
Dim CelaPakets1 As Variant
Dim CelaPakets2 As Variant

Dim Valorretard As Double
Dim Valorretard2 As Double
Dim Valorretard3 As Double
Dim ValorEscala As Double
Dim valorescala2 As Double
Dim ValorResolucio As Double
Dim ValorPakets As Double
Dim Valormaxdad As Double
Dim Valormaxdad2 As Double
Dim text As Variant
Dim Incretard As Double
Dim IncEscala As Double
Dim Incresolucio As Double
Dim IncPakets As Double
Dim Inctamany As Double
Dim CelaProto2 As Variant
Dim CelaPortor As Variant
Dim CelaPortdest As Variant
Dim ValorProto  As Variant
Dim Valorportor As Variant
Dim Valorportdest As Variant
Dim CelaProtoEscala2 As Variant
Dim CelaPortEscalaor As Variant
Dim CelaPortEscaladest As Variant
Dim CelaIPOrigen As Variant
Dim CelaIPDesti As Variant
Dim CelaMACOrigen As Variant
Dim CelaMACDesti As Variant
Dim CelaProto1 As Variant
Dim celacopy As Variant
Dim valorcopy As Double
Dim numero1 As Double
Dim numero2 As Double
Dim numero3 As Double
Dim actiu As Integer
Dim actiu2 As Integer
Dim valorprotointer As Variant
Dim celainter As Variant
Dim i As Long
Dim j As Long
Dim valormax As Double

Application.Run "Inct_2"

Sheets("Càlculs3").Select
Range("e3").Select
valormax = ActiveCell.Value


Sheets("Menú Principal").Select

numero1 = -1

text = "El Rang de l'escala va de 0 - " & valormax & " Segons"
```

```
MsgBox text

valormax = valormax * 1000

Do While numero1 = -1

text = "Valor inicial de l'escala? (ms) mín->0 "

numero1 = Val(InputBox(text, "Entrada de dades"))
Loop

If numero1 = 0 Then numero2 = 0 Else numero2 = numero1 / 1000

numero1 = -1

text = "Valor final de l'escala? (ms) Màx. ->" & valormax

Do While numero1 = -1

numero1 = Val(InputBox(text, "Entrada de dades"))
Loop

If numero1 = 0 Then numero3 = valormax / 1000 Else numero3 = numero1 / 1000


numero1 = -1

Do While numero1 = -1

numero1 = Val(InputBox("Quina resolució vols? (ms) (per defecte 1)", "Entrada de
dades"))
Loop

If numero1 = 0 Then numero1 = 1 / 1000 Else numero1 = numero1 / 1000

Sheets("Càlculs3").Select
'Range("a2:a2").Select
'Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
'Range("a2:a65536").Select
'Selection.ClearContents

Sheets("Càlculs3").Select
Range("b3:b3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("b4:b65536").Select
Selection.ClearContents

Sheets("Càlculs3").Select
Range("d2:d2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("b4:d65536").Select
Selection.ClearContents

Range("g2:g2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("g2:g65536").Select
Selection.ClearContents


Range("h3:h3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("h3:h65536").Select
Selection.ClearContents


Range("i3:i3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("i3:i65536").Select
Selection.ClearContents


Range("j3:j3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("j3:j65536").Select
Selection.ClearContents

Range("k3:k3").Select
```

```
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("k3:k65536").Select
Selection.ClearContents

Range("l3:l3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("l3:l65536").Select
Selection.ClearContents

Range("m3:m3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("m3:m65536").Select
Selection.ClearContents

Range("n3:n3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("n3:n65536").Select
Selection.ClearContents

Range("o3:o3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("o3:o65536").Select
Selection.ClearContents


Range("f3:f3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("f3:f65536").Select
Selection.ClearContents
Sheets("Càlculs3").Select

Celaretard1 = "A"
CelaEscala1 = "B"
CelaResolucio1 = "C"
CelaPakets1 = "D"

Incretard = 2
IncEscala = 2
Incresolucio = 2
IncPakets = 2
Inctamany = 2

Celaretard2 = Celaretard1 & Incretard
CelaEscala2 = CelaEscala1 & IncEscala
CelaPakets2 = CelaPakets1 & IncPakets

CelaProto1 = "F"
CelaProtoEscala1 = "G"
CelaProto2 = CelaProto1 & Inctamany
CelaProtoEscala2 = CelaProtoEscala1 & IncPakets
CelaPortor = "H" & Inctamany
CelaPortEscalaor = "I" & IncPakets
CelaPortdest = "J" & Inctamany
CelaPortEscaladest = "K" & IncPakets

'numero2 = 0
Range("b2").Select
ActiveCell.FormulaR1C1 = numero2

'numero3 = 2
Range("e6").Select
ActiveCell.FormulaR1C1 = numero3

Range("e7").Select
ActiveCell.FormulaR1C1 = numero1

Range("e5").Select
valorcopy = ActiveCell.text

'Calculem el retard
Range("a2").Select
ActiveCell.FormulaR1C1 = 0
Range("a3").Select
ActiveCell.FormulaR1C1 = "=('Dades Totals'!RC[4]-'Dades Totals'!R[-1]C[4])/1000000000"
Selection.Copy

celacopy = "A3:A" & valorcopy
```

```
Range(celacopy).Select
ActiveSheet.Paste

'Calculem l'escala
Range("b3").Select
ActiveCell.FormulaR1C1 = numero1

Range("e10").Select
valorcopy = ActiveCell.Value

'Range("b3").Select
'text = "=R[-1]C+" & Numero1
'ActiveCell.FormulaR1C1 = text

Range("B3").Select
Application.CutCopyMode = False
Selection.ClearContents
ActiveCell.FormulaR1C1 = "=R7C[3]+R[-1]C"
Selection.Copy

celacopy = "b3:b" & valorcopy
Range(celacopy).Select
ActiveSheet.Paste

Range("b2").Select
ActiveCell.FormulaR1C1 = numero2


Sheets("Càlculs3").Select
Range("b2").Select
ActiveCell.FormulaR1C1 = numero2

Range("e11").Select
ActiveCell.FormulaR1C1 = numero3

Range("e12").Select
ValorCela1 = ActiveCell.Value

cela2 = "b" & ValorCela1 & ":b65536"
Range(cela2).Select
Selection.ClearContents

cela2 = "d" & ValorCela1 & ":d65536"
Range(cela2).Select
Selection.ClearContents

cela2 = "f" & ValorCela1 & ":p65536"
Range(cela2).Select
Selection.ClearContents

Sheets("Càlculs3").Select
Incretard = 2
Celaretard1 = "A"
Celaretard2 = Celaretard1 & Incretard

Range("e3").Select
Valormaxdad = ActiveCell.text

Range("e2").Select
valormax = ActiveCell.text

CelaProto2 = CelaProto1 & 2
CelaPortor = "H" & 2
CelaPortdest = "J" & 2
CelaPortEscalaor = "I" & 2
CelaPortEscaladest = "K" & 2

Sheets("Càlculs3").Select

Range("e2").Select
Valormaxdad = ActiveCell.text

IncEscala = 2
CelaEscala2 = CelaEscala1 & IncEscala

Range("e4").Select
Valormaxdad2 = ActiveCell.text
```

```
For i = 1 To Valormaxdad2

Range(CelaPakets2).Select
ActiveCell.FormulaR1C1 = 0

IncPakets = IncPakets + 1

CelaPakets2 = CelaPakets1 & IncPakets

Next

IncPakets = 2
CelaPakets2 = CelaPakets1 & IncPakets

Range("e4").Select
Valormaxdad2 = ActiveCell.text

CelaEscala3 = CelaEscala1 & IncEscala + 2

Inctamany = 2

Sheets("Dades Totals").Select
CelaProto2 = "k" & 2
Range(CelaProto2).Select
valorprotointer = ActiveCell.text
Sheets("Càlculs3").Select


actiu = 0


Range("p1:p65535").Select
Selection.ClearContents

Range("e5").Select
IncPakets = ActiveCell.text


Range("p3").Select
ActiveCell.FormulaR1C1 = "=ROUND(C[-15],2)"
Selection.Copy

text = "p3:p" & IncPakets

Range(text).Select
ActiveSheet.Paste


 Range("Q3:AA3").Select
 Selection.Copy

 text = "q4:aa" & IncPakets

 Range(text).Select
 ActiveSheet.Paste


Range("e12").Select
IncPakets = ActiveCell.text

Range("d2").Select
ActiveCell.FormulaR1C1 = "0"
Range("d3").Select
Selection.ClearContents
ActiveCell.FormulaR1C1 = "=COUNTIF(C[12],RC[-2])"
text = "d3:d" & IncPakets - 1
Selection.AutoFill Destination:=Range(text), Type:=xlFillDefault

Range("F2:O2").Select
Application.CutCopyMode = False
Selection.Copy

text = "f3:o" & IncPakets - 1

Range(text).Select
ActiveSheet.Paste
```

```
Sheets("Dades Pk·t").Select
ActiveSheet.PivotTables("Tabla dinámica2").RefreshTable
Sheets("Càlculs3").Select
Range("b3:b3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("b4:b65536").Select
Selection.ClearContents

Sheets("Càlculs3").Select
Range("d2:d2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("b4:d65536").Select
Selection.ClearContents

Range("g2:g2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("g2:g65536").Select
Selection.ClearContents


Range("h3:h3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("h3:h65536").Select
Selection.ClearContents


Range("i3:i3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("i3:i65536").Select
Selection.ClearContents


Range("j3:j3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("j3:j65536").Select
Selection.ClearContents

Range("k3:k3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("k3:k65536").Select
Selection.ClearContents

Range("l3:l3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("l3:l65536").Select
Selection.ClearContents

Range("m3:m3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("m3:m65536").Select
Selection.ClearContents

Range("n3:n3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("n3:n65536").Select
Selection.ClearContents

Range("o3:o3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("o3:o65536").Select
Selection.ClearContents


Range("f3:f3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("f3:f65536").Select
Selection.ClearContents


Sheets("Gràfic Pk·t").Select

End Sub

Sub Inct_2()

Dim Celaretard1 As Variant
```

```
Dim Celaretard2 As Variant
Dim CelaEscala1 As Variant
Dim CelaEscala2 As Variant
Dim CelaEscala3 As Variant
Dim CelaResolucio1 As Variant
Dim CelaResolucio2 As Variant
Dim CelaPakets1 As Variant
Dim CelaPakets2 As Variant

Dim Valorretard As Double
Dim Valorretard2 As Double
Dim Valorretard3 As Double
Dim ValorEscala As Double
Dim valorescala2 As Double
Dim ValorResolucio As Double
Dim ValorPakets As Double
Dim Valormaxdad As Double
Dim Valormaxdad2 As Double
Dim text As Variant


Dim Incretard As Double
Dim IncEscala As Double
Dim Incresolucio As Double
Dim IncPakets As Double
Dim Inctamany As Double



Dim CelaProto2 As Variant
Dim CelaPortor As Variant
Dim CelaPortdest As Variant
Dim ValorProto  As Variant
Dim Valorportor As Variant
Dim Valorportdest As Variant
Dim CelaProtoEscala2 As Variant
Dim CelaPortEscalaor As Variant
Dim CelaPortEscaladest As Variant
Dim CelaIPOrigen As Variant
Dim CelaIPDesti As Variant
Dim CelaMACOrigen As Variant
Dim CelaMACDesti As Variant
Dim CelaProto1 As Variant
Dim celacopy As Variant
Dim valorcopy As Double
Dim numero1 As Double
Dim numero2 As Double
Dim numero3 As Double
Dim i As Long
Dim j As Long

Dim valormax As Double

numero1 = 1

Sheets("Càlculs3").Select
Range("a2:a2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("a2:a65536").Select
Selection.ClearContents

Sheets("Càlculs3").Select
Range("b3:b3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("b4:b65536").Select
Selection.ClearContents

Sheets("Càlculs3").Select
Range("d2:d2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("b4:d65536").Select
Selection.ClearContents

Range("g2:g2").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("g2:g65536").Select
Selection.ClearContents
```

```
Range("h3:h3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("h3:h65536").Select
Selection.ClearContents


Range("i3:i3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("i3:i65536").Select
Selection.ClearContents


Range("j3:j3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("j3:j65536").Select
Selection.ClearContents

Range("k3:k3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("k3:k65536").Select
Selection.ClearContents

Range("l3:l3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("l3:l65536").Select
Selection.ClearContents

Range("m3:m3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("m3:m65536").Select
Selection.ClearContents

Range("n3:n3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("n3:n65536").Select
Selection.ClearContents

Range("o3:o3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("o3:o65536").Select
Selection.ClearContents


Range("f3:f3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("f3:f65536").Select
Selection.ClearContents



Sheets("Càlculs3").Select

Celaretard1 = "A"
CelaEscala1 = "B"
CelaResolucio1 = "C"
CelaPakets1 = "D"

Incretard = 2
IncEscala = 2
Incresolucio = 2
IncPakets = 2
Inctamany = 2

Celaretard2 = Celaretard1 & Incretard
CelaEscala2 = CelaEscala1 & IncEscala
CelaPakets2 = CelaPakets1 & IncPakets



CelaProto1 = "F"
CelaProtoEscala1 = "G"
CelaProto2 = CelaProto1 & Inctamany
CelaProtoEscala2 = CelaProtoEscala1 & IncPakets
CelaPortor = "H" & Inctamany
CelaPortEscalaor = "I" & IncPakets
```

```
CelaPortdest = "J" & Inctamany
CelaPortEscaladest = "K" & IncPakets

numero2 = 0
Range("b2").Select
ActiveCell.FormulaR1C1 = numero2

numero3 = 2
Range("e6").Select
ActiveCell.FormulaR1C1 = numero3

Range("e7").Select
ActiveCell.FormulaR1C1 = numero1

Range("e5").Select
ActiveCell.FormulaR1C1 = "=COUNT('Dades Totals'!C[-3])"
valorcopy = ActiveCell.text

'Calculem el retard
Range("a2").Select
ActiveCell.FormulaR1C1 = 0
Range("a3").Select
ActiveCell.FormulaR1C1 = "=('Dades Totals'!RC[4]-'Dades Totals'!R[-1]C[4])/1000000000"
Selection.Copy

celacopy = "A3:A" & valorcopy
Range(celacopy).Select
ActiveSheet.Paste

'Calculem l'escala
Range("b3").Select
ActiveCell.FormulaR1C1 = numero1

Range("e10").Select
valorcopy = ActiveCell.Value

Range("b3").Select
text = "=R[-1]C+" & numero1
ActiveCell.FormulaR1C1 = text
Selection.Copy

celacopy = "b3:b" & valorcopy
Range(celacopy).Select
ActiveSheet.Paste


Sheets("Càlculs3").Select
Incretard = 2
Celaretard1 = "A"
Celaretard2 = Celaretard1 & Incretard

End Sub

Sub Torna_a_Vi()
    ActiveWindow.ScrollWorkbookTabs Position:=xlFirst
    Sheets("Gràfic Vi").Select
End Sub

Sub Anar_Estadi()

    Sheets("dades estadistica").Select
    ActiveWindow.LargeScroll ToRight:=0
    Range("B11").Select
    ActiveSheet.PivotTables("Tabla dinámica3").PivotCache.Refresh
    ActiveWindow.ScrollWorkbookTabs Position:=xlFirst
    Sheets("estadistica").Select
    ActiveChart.SeriesCollection(1).DataLabels.Select
    Selection.NumberFormat = "0.00%"
End Sub

Sub Pktamany_nou()
'
'

'
    Dim final As Variant
    Dim text As Variant
```

```
Dim numero1 As Variant
Dim valormax As Integer

Sheets("Càlculs4b").Select


Range("l3:s65536").Select
Selection.Clear


Sheets("Càlculs4b").Select
Range("a2:e65536").Select
Selection.Clear
Range("h2").Select
final = ActiveCell.text
final = final + 3


Sheets("Dades Totals").Select
Range("C1:C3").Select
Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
Range("C1:C3744").Select
Selection.Copy
Sheets("Càlculs4b").Select
Range("a1").Select
ActiveSheet.Paste
Range("B1").Select
Application.CutCopyMode = False
ActiveCell.FormulaR1C1 = "Escala"
Range("B2").Select
ActiveCell.FormulaR1C1 = "0"


Range("b3").Select
ActiveCell.FormulaR1C1 = "=R[-1]C+1"
Selection.Copy

text = "b4:" & "b" & final
Range(text).Select

ActiveSheet.Paste
ActiveWindow.SmallScroll Down:=168
Range("C1").Select
Application.CutCopyMode = False
ActiveCell.FormulaR1C1 = "N° pakets"
Range("C2").Select
ActiveCell.FormulaR1C1 = "=COUNTIF(C[-2],RC[-1])"
Selection.Copy

text = "c3:" & "c" & final

Range(text).Select
ActiveSheet.Paste


Range("f2").Select
valormax = ActiveCell.text

Sheets("Menú Principal").Select

text = "El Rang de l'escala va de 0 - " & valormax & " Bytes"
MsgBox text

numero1 = -1

Do While numero1 = -1

text = "Valor inicial de l'escala? (bytes) mín->0"

numero1 = Val(InputBox(text, "Entrada de dades"))
Loop

    If numero1 = 0 Then
    Sheets("Càlculs4b").Select
    Range("g2").Select
    ActiveCell.FormulaR1C1 = 0
    Sheets("Menú Principal").Select
```

```
        Else
        Sheets("Càlculs4b").Select
        Range("g2").Select
        ActiveCell.FormulaR1C1 = numero1
        Range("d2").Select
        ActiveCell.FormulaR1C1 = numero1
        Sheets("Menú Principal").Select
        End If


    numero1 = -1

    text = "Valor final de l'escala? (bytes) Màx. ->" & valormax

    Do While numero1 = -1

    numero1 = Val(InputBox(text, "Entrada de dades"))
    Loop

        If numero1 = 0 Then
        Sheets("Càlculs4b").Select
        Range("h2").Select
        ActiveCell.FormulaR1C1 = valormax
        Sheets("Menú Principal").Select
        Else
        Sheets("Càlculs4b").Select
        Range("h2").Select
        ActiveCell.FormulaR1C1 = numero1
        Sheets("Menú Principal").Select
        End If


    numero1 = -1

    Do While numero1 = -1

    numero1 = Val(InputBox("Quina resolució vols? (bytes)  (per defecte 1)", "Entrada de
dades"))
    Loop

        If numero1 = 0 Then
        Sheets("Càlculs4b").Select
        Range("i2").Select
        ActiveCell.FormulaR1C1 = 1
        Sheets("Menú Principal").Select
        Else
        Sheets("Càlculs4b").Select
        Range("i2").Select
        ActiveCell.FormulaR1C1 = numero1
        Sheets("Menú Principal").Select
        End If

    Sheets("Càlculs4b").Select
    Range("j2").Select
    final = ActiveCell.text


    Range("L2:S2").Select
    Application.CutCopyMode = False
    Selection.Copy


    text = "l3:s" & final

    Range(text).Select
    ActiveSheet.Paste


    Application.Run "pktamany_escala"

    Sheets("Dades Pk·byte2").Select
    Range("C13").Select
    ActiveSheet.PivotTables("Tabla dinámica1").RefreshTable
    Sheets("Gràfic Pk·byte2").Select


End Sub
```

```
Sub pktamany_escala()
'
    Dim final As Variant
    Dim text As Variant

    Range("j2").Select
    final = ActiveCell.text


    ActiveWindow.SmallScroll Down:=-9
    Range("D2").Select
    Application.CutCopyMode = False
    'ActiveCell.FormulaR1C1 = "0"
    Range("D3").Select
    ActiveCell.FormulaR1C1 = "=R[-1]C[5]+R[-1]C"
    Range("D4").Select
    ActiveCell.FormulaR1C1 = "=R[-1]C+R2C[5]"

    text = "D4:" & "D" & final

    Selection.AutoFill Destination:=Range(text), Type:=xlFillDefault
    Range(text).Select


    Range("E2").Select
    ActiveCell.FormulaR1C1 = "0"

    text = "e3:" & "e" & final

    ActiveCell.Copy

    Range(text).Select
    ActiveSheet.Paste

    Dim bucle1 As Variant
    Dim bucle2 As Integer

    Dim i As Variant
    Dim j As Variant
    Dim inc As Integer

    Dim npakets1 As Variant
    Dim npakets2 As Variant


    Range("i2").Select
    bucle2 = ActiveCell.text

    Range("h2").Select
    bucle1 = ActiveCell.text
    bucle1 = bucle1 + 1

    npakets1 = 0
    inc = 0
    j = 2

    Range("g2").Select
    i = ActiveCell.text

    text = "c" & (i + 3)
    Range(text).Select


    For i = i To bucle1

    text = "c" & (i + 2)
    Range(text).Select
    npakets1 = npakets1 + ActiveCell.text

    text = "c" & (i + 3)
    Range(text).Select
    If ActiveCell.text = "" Then
    i = bucle1
    End If


    inc = inc + 1
```

```
            text = "e" & j
            Range(text).Select
            ActiveCell.FormulaR1C1 = npakets1

        If inc = bucle2 Then
            inc = 0

            npakets1 = 0

            j = j + 1
        End If

    Next

End Sub

Sub Desglossar()
'
'

'
    With ActiveChart.PivotLayout.PivotTable.PivotFields("Port Origen")
        .Orientation = xlColumnField
        .Position = 2
    End With
    With ActiveChart.PivotLayout.PivotTable.PivotFields("Port Destí")
        .Orientation = xlColumnField
        .Position = 3
    End With
End Sub

Sub Agrupar()
'
'
    With ActiveChart.PivotLayout.PivotTable.PivotFields("Port Origen")
        .Orientation = xlPageField
        .Position = 1
    End With
    With ActiveChart.PivotLayout.PivotTable.PivotFields("Port Destí")
        .Orientation = xlPageField
        .Position = 1
    End With


End Sub
Sub Agrupa_pk_byte()
'
'

'
    ActiveChart.Shapes("Button 5").Select
    Selection.ShapeRange.IncrementLeft -7.68
    Selection.ShapeRange.IncrementTop 0.05
    ActiveChart.Shapes("Button 1").Select
    Selection.ShapeRange.IncrementLeft -15.87
    Selection.ShapeRange.IncrementTop 0.05
    ActiveChart.Shapes("Button 2").Select
    Selection.ShapeRange.IncrementLeft -30.03
    Selection.ShapeRange.IncrementTop 0.05
    ActiveChart.Shapes("Button 7").Select
    Selection.ShapeRange.IncrementLeft -38.25
    Selection.ShapeRange.IncrementTop 0.05
    ActiveChart.Shapes("Button 8").Select
    Selection.ShapeRange.IncrementLeft -51.09
    Selection.ShapeRange.IncrementTop 0.05
    ActiveChart.Shapes("Button 3").Select
    Selection.ShapeRange.IncrementLeft -33.48
    Selection.ShapeRange.IncrementTop 0.05
    ActiveChart.ChartArea.Select
End Sub

Sub Suavitzar()
'
' Suavitzar Macro
' Macro grabada el 18/10/2004 por Iñaki Quer Alecha
'
```

```
'
    ActiveChart.ChartArea.Select
    ActiveChart.SeriesCollection(1).Select
    With Selection.Border
        .ColorIndex = 57
        .Weight = xlThick
        .LineStyle = xlContinuous
    End With
    With Selection
        .MarkerBackgroundColorIndex = xlAutomatic
        .MarkerForegroundColorIndex = xlAutomatic
        .MarkerStyle = xlDiamond
        .Smooth = True
        .MarkerSize = 12
        .Shadow = True
    End With
    ActiveChart.ChartArea.Select
End Sub

Sub Executaresolucio()

    Application.Run "'Genera Distribucions v20.0.xls'!resolució"
    ActiveWindow.ScrollWorkbookTabs Sheets:=1
    Sheets("Gràfic Vi").Select
End Sub


Sub csv()
'
' csv Macro
' Macro grabada el 22/10/2004 por Iñaki Quer Alecha
'
Dim numero1 As String
Dim text As Variant
Dim cela2 As Variant


numero1 = ""

'Do While numero1 = ""

'numero1 = InputBox("Introdueix la ruta i el nom de l'arxiu .CSV
(ex: c:\Genera Gràfic\captura_ether.csv)", "Entrada de dades")

'Loop

text = "TEXT;" & numero1

    Sheets("Dades Dist.t").Select
    ActiveWindow.ScrollWorkbookTabs Position:=xlLast
    Sheets("Dades Totals").Select

    ActiveWindow.ScrollWorkbookTabs Position:=xlLast
    Sheets("Dades Totals").Select
    Range("A1").Select

    With Selection.QueryTable
        .Connection = text
        .TextFilePlatform = xlWindows
        .TextFileStartRow = 1
        .TextFileParseType = xlDelimited
        .TextFileTextQualifier = xlTextQualifierDoubleQuote
        .TextFileConsecutiveDelimiter = False
        .TextFileTabDelimiter = False
        .TextFileSemicolonDelimiter = True
        .TextFileCommaDelimiter = True
        .TextFileSpaceDelimiter = False
        .TextFileColumnDataTypes = Array(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
        .Refresh BackgroundQuery:=False
    End With
    Sheets("Càlculs").Select

    Range("r2").Select
    ValorCela1 = ActiveCell.text

    'cela2 = "a" & ValorCela1 & ":e65536"
```

```
    cela2 = "a2" & ValorCela1 & ":e65536"
    Range(cela2).Select
    Selection.ClearContents

    cela2 = "o" & ValorCela1 & ":ad65536"
    Range(cela2).Select
    Selection.ClearContents

     ActiveWindow.ScrollWorkbookTabs Position:=xlLast
    Sheets("Dades Totals").Select
     Columns("E:E").Select
        Selection.Replace What:=".", Replacement:="", LookAt:=xlPart, _
        SearchOrder:=xlByRows, MatchCase:=False

    Sheets("Menú Principal").Select

    text = "Les dades s'han importat correctament!"
    MsgBox text

End Sub
```

6.1.2. ANNEX 2 “Codi del script exportacsv”

```
#!/bin/sh


echo "*************************************"
echo "*                                   *"
echo "*    Benvingut a l'exportador de    *"
echo "*    captures a CSV                 *"
echo "*                                   *"
echo "*                                   *"
echo "* 0 0 0                             *"
echo "* 0 0 0     Domínguez González, LUIS *"
echo "* 0 0 0     Quer Alecha, Iñaki      *"
echo "* U P C          EPSC 04-05         *"
echo "*                                   *"
echo "*************************************"


echo -n " De quina Ethernet vols capturar els paquets ? "
read PAQUETS

printf "Nom Captura,Frame,Tamany,Arribada,segons,MAC Origen,MAC
Destí,Protocol de Xarxa,IP Origen,IP Destí,Protocol de transport,Port
Origen,Port Destí,Hora,Minuts,RTP,RTCP_RR,RTCP_SR,
IGMP,ICMP\n">captura_ether.csv

#Capturem
command tethereal -V -n -i $PAQUETS -w captura_ether
command tethereal -V -n -r captura_ether>captura_list

#Paquets RTP
command echo "Determinant paquets RTP..."
command tcpdump  -S udp -T rtp -r captura_ether | wc -l>rtp
set rtp_eco
rtp_eco=$(cat rtp)
command rm rtp

#Paquets RTCP-RR
command echo "Determinant paquets RTCP-RR..."
```

```
command tcpdump  udp -T rtcp -r captura_ether | grep -a  "rr" |wc -
l>rtcp_rr
set rtcp_rr_eco
rtcp_rr_eco=$(cat rtcp_rr)
command rm rtcp_rr

#Paquets RTCP-SR
command echo "Determinant paquets RTCP-SR..."
command tcpdump  udp -T rtcp -r captura_ether | grep -a  "sr" |wc -
l>rtcp_sr
set rtcp_sr_eco
rtcp_sr_eco=$(cat rtcp_sr)
command rm rtcp_sr

#Paquets IGMP
command echo "Determinant paquets IGMP..."
command tcpdump igmp -r captura_ether | wc -l >igmp
set igmp_eco
igmp_eco=$(cat igmp)
command rm igmp
#Paquets ICMP
command echo "Determinant paquets ICMP..."
command  tcpdump icmp -r captura_ether | wc -l >icmp
set icmp_eco
icmp_eco=$(cat icmp)
command rm icmp

#N°de paquets
command grep "Arrival" captura_list | wc -l >npaquets
set npaquets_eco
npaquets_eco=$(cat npaquets)
rm npaquets

npaquets_eco=`expr $npaquets_eco - 1`

set falta

for BUCLE in $(seq 0 $npaquets_eco)
do

./passapaket 0 captura_list
clear
echo "-------------------------"
echo "Analitzant PAQUET --> $BUCLE "
echo "-------------------------"

falta=`expr $BUCLE \* 100 / $npaquets_eco`
echo "--> $falta % finalitzat"


#Frame
#command echo "Analitzant FRAME"
command cat captura_list2 | grep -a  "Frame Number:" | cut -d ':' -
f2>frame
set frame_eco
frame_eco=$(cat frame)
command rm frame

#Tamany
#command echo "Analitzant TAMANY"
```

```
command cat captura_list2 | grep -a "Packet Length"| cut -d ':' -f2 |
cut -d 'b' -f1>tamany
set tamany_eco
tamany_eco=$(cat tamany)
command rm tamany

#Arribada
#command echo "Analitzant Arribada"
command cat captura_list2 | grep -a  "Arrival Time" | cut -d ',' -
f2>arribada
set arribada_eco
arribada_eco=$(cat arribada)
command rm arribada

#segons
#command echo "Determinant segons exactes"
command cat captura_list2 | grep -a  "Arrival Time" | cut -d " " -f7|
cut -d ":" -f3>segons
set segons_eco
segons_eco=$(cat segons)
command rm segons
#MAC origen
#command echo "Analitzant MAC Origen"
command cat captura_list2 | grep -a  "Ethernet II, Src: " |cut -d ','
-f2 | cut -d '(' -f1>macorg
set macorg_eco
macorg_eco=$(cat macorg)
command rm macorg

#MAC destí
#command echo "Analitzant MAC Dest"
command cat captura_list2 | grep -a  "Ethernet II, Src: " |cut -d ','
-f3 | cut -d '(' -f1>macdest
set macdest_eco
macdest_eco=$(cat macdest)
command rm macdest

#Protocol de Transport
#command echo "Analitzant Protocol de Transport"
command cat captura_list2 | grep -a  "Protocol: " | cut -d ':' -f2 |
cut -d '(' -f1>protot
set protot_eco
protot_eco=$(cat protot)
command rm protot

#Protocol de Xarxa
#command echo "Analitzant Protocol de Xarxa"
command cat captura_list2 | grep -a  "Type:" | cut -d ' ' -f3 | cut -d
'(' -f1>protox
set protox_eco
protox_eco=$(cat protox)
protox_eco="IP"
command rm protox

#IP origen
#command echo "Determinant IP Origen"
command cat captura_list2 | grep -a  "Internet Protocol, Src Addr:" |
cut -d ':' -f2 | cut -d '(' -f1>iporg

command cat iporg | wc -l >conta
set conta_eco
```

```
conta_eco=$(cat conta)

if [ $conta_eco != "1" ];then
command cat captura_list2 | grep -a  "    Internet Protocol, Src
Addr:" | cut -d ':' -f2 | cut -d '(' -f1>iporg
fi

set iporg_eco
iporg_eco=$(cat iporg)

command rm iporg

#IP desti
#command echo "Determinant IP Dest"
command cat captura_list2 | grep -a  "Dst Addr: 192.168" | cut -d ':'
-f3 | cut -d '(' -f1>ipdest
set ipdest_eco
ipdest_eco=$(cat ipdest)
command rm ipdest


#Port Origen
#command echo "Determinant Port Origen"
command cat captura_list2 | grep -a  "Src Port: " |cut -d ':' -f2 |
cut -d '(' -f1>portorg
set portorg_eco
portorg_eco=$(cat portorg)
command rm portorg

#Port Destí
#command echo "Determinant Port Dest"
command cat captura_list2 | grep -a  "Src Port: " |cut -d ':' -f3 |
cut -d '(' -f1>portdest
set portdest_eco
portdest_eco=$(cat portdest)
command rm portdest


#Hora d'arribada
#command echo "Determinant Hora Arribada"
command cat captura_list2 | grep -a  "Arrival Time" | cut -d " " -f7|
cut -d ":" -f1>hora
set hora_eco
hora_eco=$(cat hora)
command rm hora

#Minuts d'arribada
#command echo "Determinant Minuts Arribada"
command cat captura_list2 | grep -a  "Arrival Time" | cut -d " " -f7|
cut -d ":" -f2>minuts
set minuts_eco
minuts_eco=$(cat minuts)
command rm minuts

command echo "Captura
eth,$frame_eco,$tamany_eco,$arribada_eco,$segons_eco,$macorg_eco,$macd
est_eco,$protox_eco,$iporg_eco,$ipdest_eco,$protot_eco,$portorg_eco,$p
ortdest_eco,$hora_eco,$minuts_eco,$rtp_eco,$rtcp_rr_eco,$rtcp_sr_eco,$
igmp_eco,$icmp_eco">>captura_ether.csv

cp -f nombre captura_list
```

```
done
echo "===================================="
command echo "Analitzats els FRAMES"
command echo "Analitzats els TAMANYS"
command echo "Analitzades les Arribades"
command echo "Determinats segons exactes"
command echo "Analitzades MAC Origen"
command echo "Analitzades MAC Dest"
command echo "Analitzats Protocols de Transport"
command echo "Analitzats Protocols de Xarxa"
command echo "Determinades IP Origen"
command echo "Determinats Port Dest"
command echo "Determinat Hores Arribada"
command echo "Determinat Minuts Arribada"
command echo "Determinats paquets RTP..."
command echo "Determinats paquets RTCP-RR..."
command echo "Determinats paquets RTCP-SR..."
command echo "Determinats paquets IGMP..."
command echo "Determinats paquets ICMP..."
echo "===================================="
echo "------------------------"
echo "Captura FINALITZADA!! "
echo "------------------------"

command rm captura_list
```

6.1.3. ANNEX 3 “Codi C de l’aplicació *passapaket* ”

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#ifndef __TURBOC__
# define MB      1
# define SIZE    ((MB)*256*1024)
# else
# include <io.h>
# define KB      48
# define SIZE    ((KB)*256)
# endif

static signed short   buff  [7000000];
static signed short   buff3 [7000000];
static signed short   buff2 [1120];

int main(int argc, char **argv )
{

   FILE *fp;
   FILE *fp2;
   FILE *fp3;

   size_t len1=0,len2=0,size1=0,size2=0,len=0;
   long int   i,position,position2,j,k,l;

   fp=fopen(argv[2],"r+");
```

```
   fp2=fopen("captura_list2","w+");
   fp3=fopen("nombre","w+");

   position=atoi(argv[1]);
   l=position+8241;

   j=-1;


   for(k=0;k<SIZE;k++)
   {
   fseek(fp,k,0);
   fread(buff,2,5,fp);

      for(i=0;i<5;i++)
      {
      if(buff[i]==29249)&&(buff[i+1]==26994)&&(buff[i+2]==24950)&&
(buff[i+3]==8300)&&(buff[i+4]==26964))
            {
            j++;
            position2=k+i;
            }

            if(position==j){k=SIZE;i=1500;}
      }
   }




fseek(fp,position2,0);
fread(buff,2,1000,fp);

    for(i=0;i<1000;i++) buff2[i]=buff[i];


    fwrite(buff2,2,700,fp2);


#ifdef __TURBOC__
       len1 = _read ( fileno(fp), buff, 2*SIZE ) / 2;
       #else
       len1 = fread ( buff, 2, SIZE, fp );
#endif
        size1 += len1;

        len=size1;

fseek(fp,position2+700,0);
fread(buff,1,len*24,fp);
fwrite(buff,1,len*24,fp3);

    fclose(fp);
    fclose(fp2);
    fclose(fp3);

}
```

6.2. CAPÍTOL 2

6.2.1. ANNEX 4 “Codi syncok ”

```
#include <stdio.h>
#include <math.h>
#include <string.h>


#ifndef __TURBOC__
# define MB      1
# define SIZE    ((MB)*256*1024)
#else
# include <io.h>
# define KB      48
# define SIZE    ((KB)*256)
#endif

static signed short   buff1 [SIZE];
static signed short   buff2 [SIZE];
static signed short   buff3 [SIZE];
static signed short   buff4 [SIZE];
static signed short   buff5 [SIZE];
int ventanasi=0;

unsigned int  autocorrelaciona(size_t len,size_t size1)
{

   long int  i,j=0,position=0,maxdelay,delay;
   double  mx,my,sx,sy,sxy,denom,r,last;
   size_t n;
   char retorn;

   if(len>250000) n=250000; else n=len;
   if(ventanasi==1){
   printf("\nQuin tamany de finestra vols? (Màx. %u) :",size1);
   scanf("%u",&maxdelay);}
   else
   maxdelay=size1;


   /* Càlcul del medi de les dues sèries x[], y[] */
   mx = 0;
   my = 0;
   for (i=0;i<n;i++) {
      mx += buff4[i];
      my += buff2[i];
   }
   mx /= n;
   my /= n;

   /* Càlcul del denominador */

   sx = 0;
   sy = 0;
```

```
   for (i=0;i<n;i++) {
      sx += (buff4[i] - mx) * (buff4[i] - mx);
      sy += (buff2[i] - my) * (buff2[i] - my);
   }
   sx=sx*sy;
   denom =sqrt(sx);


   /* Càlcul de la correlació de les sèries */
      sxy = 0;
      last = 0;

      printf("\n\n*************************************\n");
      printf("*      Calculant correlacions...     *\n");
      printf("*************************************\n");


      for (i=44;i<n;i++) {

            if(j==maxdelay)
            {

              r=sxy / denom;

                  if (i==maxdelay){last=r; position=i;}
                  if (r>last) {last=r; position=i;}
                last=r;
                j=0;
            sxy=0;
                  }

         sxy += (buff4[j] - mx) * (buff2[i] - my);

        j++;
      }
      position=position/4;


      for (i=position,j=position;i>44;i--)
      {

            if((buff2[i]<0)&&((j-i)<1000))
            {
            position=i;
            j=i;
            }
      }


      printf("\nHi ha més correlació en la posició -->%u i amb valor--
>%d \n",position,buff2[position]);
    return(position);
}


int  main ( int argc, char** argv )
{
   FILE *fp;
   FILE *fp2;
   FILE *fp3;
   FILE *fp4;
```

```
   size_t len1=0,len2=0,size1=0,size2=0,len=0;
   long int   i,j,position,position2,control;
   char filein[12],filein2[12],fileout[12];
   int place;
   char *aux;

   switch (argc) {
   case 4:
            if( (fp=fopen(argv[1],"r+")) == NULL){
            fprintf(stderr, "El fitxer '%s' no es pot
obrir\n",argv[1]);
        }
      if( (fp2=fopen(argv[2],"r+")) == NULL){
            fprintf(stderr,"El fitxer '%s' no es pot obrir\n",argv[2]);
        }
        aux=argv[3];
        place=strlen(argv[3])-4;
        if(*(aux+place)!='.')
        strcat(argv[3],".wav");
      fp3=fopen(argv[3],"w+");

      break;

   default:

printf("*************************************\n");
printf("*                                   *\n");
printf("*    Benvingut al sincronitzador    *\n");
printf("*    d'arxius .WAV                  *\n");
printf("*                                   *\n");
printf("*                                   *\n");
printf("* 0 0 0                             *\n");
printf("* 0 0 0     Domínguez González, LUIS *\n");
printf("* 0 0 0     Quer Alecha, Iñaki      *\n");
printf("* U P C          EPSC 04-05         *\n");
printf("*                                   *\n");
printf("*************************************\n");

ventanasi=1;

do{
printf("\nIntrodueix fitxer d'entrada1 (màx. 8 caràcters) ->");
scanf("%s",&filein);
if(strlen(filein)>12) printf("El nom del fitxer ha de ser com màx. 8
caràcters \n");
else
{
      fp=fopen(filein,"r+");
      if(fp==NULL) printf("El fitxer introduït no s'ha trobat al
directori \n");
}
}while(strlen(filein)>12 || fp == NULL);

do{
printf("\nIntrodueix fitxer d'entrada2 (màx. 8 caràcters) ->");
scanf("%s",&filein2);
if(strlen(filein2)>12) printf("El nom del fitxer ha de ser com màx. 8
caràcters \n");
else
{
```

```
      fp2=fopen(filein2,"r+");
      if(fp2==NULL) printf("El fitxer introduït no s'ha trobat al
directori \n");
}
}while(strlen(filein2)>12 || fp2 == NULL);

do{
printf("\nIntrodueix fitxer de sortida (màx. 8 caràcters) ->");
scanf("%s",&fileout);
if(strlen(fileout)>12) printf("El nom del fitxer ha de ser com màx. 8
caràcters \n");
else
{
      place=strlen(fileout)-4;
      if(*(fileout+place)!='.')
        strcat(fileout,".wav");
            fp3=fopen(fileout,"w+");

}
}while(strlen(fileout)>12);

}
#ifdef __TURBOC__
       len1 = _read ( fileno(fp), buff1, 2*SIZE ) / 2;
       len2 = _read ( fileno(fp2), buff2, 2*SIZE ) / 2;
#else
       len1 = fread ( buff1, 2, SIZE, fp );
       len2 = fread ( buff2, 2, SIZE, fp2 );
#endif
        size1 += len1;
        size2 += len2;

        if(size1>=size2) len=size1; else len=size2;

        len=len*2;

      control=0;
        for(i=44;i<size1;i++)
      {

            if((buff1[i]<0)&&(control==0))
            {
            control=i;
            }

            if(control>0)

            buff4[i-control]=buff1[i];
      }

      printf("control --> %d",control);
      position=autocorrelaciona(len,size1);


        for(i=0;i<44;i++) {buff3[i]=buff2[i];buff5[i]=buff2[i];}


        for(i=44;i<len;i++) buff5[i]=buff4[i];


        for(i=44;i<len;i++)
```

```
        {
           buff3[i]=buff2[position];
           control++;
           position++;
        }

        printf("Tamany de l'arxiu generat-->%u\n",len);

control=0;
position=0;

for(i=44;i<size1;i++)
      {

            if((buff5[i]<0)&&(position==0)&&(control==0))
            {
            position=i;
            control=1;
            }
            if(position>0)
            buff5[44+i-position]=buff5[i];

      }


for (i=position,j=position;i>44;i--)
      {

            if((buff5[i]<0)&&((j-i)<1000))
            {
            position=i;
            j=i;
            }
      }


for(i=44;i<size1;i++)
        {
      buff5[i]=buff5[position];
      position++;
        }




   fwrite(buff3,1,len,fp3);
   fp4=fopen("org.wav","w+");
   fwrite(buff5,1,size1*2,fp4);
   fclose(fp4);

   fclose(fp3);
   fclose(fp);
   fclose(fp2);

}
```

6.3. CAPÍTOL 3

6.3.1. ANNEX 5 “Codi de la integració del AMR al RAT”

a) Codec_amr.c

```
/*
*******************************************************************
*******
*
*      GSM AMR-NB speech codec    REL-5 V5.4.0 2004-03
*                                  REL-6 V6.1.0 2004-03
*
*******************************************************************
********/
#include <stdlib.h>
#include <stdio.h>
#include <memory.h>
#include <string.h>
#include "typedef.h"
#include "interf_enc.h"
#include "interf_dec.h"
#include "sp_enc.h"
#include "sp_dec.h"

#ifndef ETSI
#ifndef IF2
#define AMR_MAGIC_NUMBER "#!AMR\n"
#endif
#endif

static const short modeConv[]={
   475, 515, 59, 67, 74, 795, 102, 122};


/* frame size in serial bitstream file (frame type + serial stream +
flags) */
//#define SERIAL_FRAMESIZE (1+MAX_SERIAL_SIZE+5)


/* ---------- end of CODEC AMR ---------------------------------------
-------------- */

/*******************************************************************
********/
/* RAT specific interface (Domínguez G., LUIS and Quer A., Iñaki
 2004-2005-1)*/
/*******************************************************************
********/

#include "config_unix.h"
#include "config_win32.h"
#include "memory.h"
#include "audio_types.h"
#include "codec_types.h"
#include "codec_amr2.h"
```

```
#include "cx_amr.h"
#include "util.h"
#include "debug.h"
#include "pdb.h"

int fract_lost_auto;
int total_lost_auto;

static codec_format_t cs[] = {
/* 8kHz */
        {"AMR-Mode0", "AMR-8K-Mode0",
         "AMR codec mode 4.75kbit/s",
        34, 0,12*3 ,{DEV_S16, 8000, 16, 1,320}}, /* 20 ms */
/* 8kHz */
        {"AMR-Mode1", "AMR-8K-Mode1",
         "AMR codec mode 5.15kbit/s",
        35, 0, 13*3,{DEV_S16, 8000, 16, 1,320}}, /* 20 ms */
/* 8kHz */
        {"AMR-Mode2", "AMR-8K-Mode2",
         "AMR codec mode 5.9kbit/s",
        36, 0, 15.0*3,{DEV_S16, 8000, 16, 1,320}}, /* 20 ms */
/* 8kHz */
        {"AMR-Mode3", "AMR-8K-Mode3",
         "AMR codec mode 6.7kbit/s",
        37, 0, 17.0*3,{DEV_S16, 8000, 16, 1,320}}, /* 20 ms */
/* 8kHz */
        {"AMR-Mode4", "AMR-8K-Mode4",
         "AMR codec mode 7.4kbit/s",
        38, 0, 19.0*3,{DEV_S16, 8000, 16, 1,320}}, /* 20 ms */
/* 8kHz */
        {"AMR-Mode5", "AMR-8K-Mode5",
         "AMR codec mode 7.95kbit/s",
        39, 0, 20*3,{DEV_S16, 8000, 16, 1,320}}, /* 20 ms */
/* 8kHz */
        {"AMR-Mode6", "AMR-8K-Mode6",
         "AMR codec mode 10.2kbit/s",
        40, 0, 26*3,{DEV_S16, 8000, 16, 1,320}}, /* 20 ms */
/* 8kHz */
        {"AMR-Mode7", "AMR-8K-Mode7",
         "AMR codec mode 12.2kbit/s",
        41, 0, 31*3,{DEV_S16, 8000, 16, 1,320}},/* 20 ms */
/* 8kHz */
        {"AMR-ModeAuto", "AMR-8K-Mode-Auto",
         "AMR codec mode Auto",
        42, 0, 32*3,{DEV_S16, 8000, 16, 1,320}} /* 20 ms */
/* 16kHz (en aquest cas, a d'haver de 6.6kbits, 8.85kbits, 12.65kbits,
14.25kbits, 15.85kbits, 18.25kbits, 19.85kbits, 23.05kbps i
23.85kbits)

en 8kHz es diu AMR-NB i en 16kHz es diu AMR-WB */

};
#define AMR_NUM_FORMATS (sizeof(cs)/sizeof(codec_format_t))

uint16_t
amr_get_formats_count()
{
        return AMR_NUM_FORMATS;
}

const codec_format_t*
```

```
amr_get_format(uint16_t idx)
{
        assert(idx < AMR_NUM_FORMATS);
        return &cs[idx];
}

int
amr_state_create(uint16_t idx, u_char **state)
{
      int i=12*3;

        assert(idx < AMR_NUM_FORMATS);
        *state = (u_char*) amr_create();


            if (idx==0) i=12*3;

            if (idx==1) i=13*3;

            if (idx==2) i=15*3;

            if (idx==3) i=17*3;

            if (idx==4) i=19*3;

            if (idx==5) i=20*3;

            if (idx==6) i=26*3;

            if (idx==7) i=31*3;

            if (idx==8) i=32*3;

        return i;
}

void
amr_state_destroy(uint16_t idx, u_char **state)
{
        assert(idx < AMR_NUM_FORMATS);
        UNUSED(idx);

        amr_destroy((amr)*state);
        *state = (u_char*)NULL;
}

int
amr_encoder  (uint16_t idx, u_char *state, sample *in, coded_unit
*out)
{


      FILE *arxiu;
      char  buff1[1];


      arxiu=fopen("canvia_codec","r+");
      fread( buff1, 1, 1, arxiu );
      printf("he llegit -->%c",buff1[0]);
      fclose(arxiu);
```

```
      /* input speech vector */
        short speech[160];

        /* pointer to encoder state structure */
      int *enstate;

      /* requested mode */
      enum Mode req_mode = MR475;
      int dtx = 0;

        /* bitstream filetype */
        #ifndef ETSI
        unsigned char serial_data[32];
      #else
      short serial_data[250] = {0};
      #endif




        assert(idx < AMR_NUM_FORMATS);
        assert(state);
        assert(in);
        assert(out);


      if(idx==8){


            if (buff1[0]=='0') { printf("You have chosen the mode -
>MR475\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(12*3);
                    out->data_len  = 12*3;
                  req_mode = MR475;}

            if (buff1[0]=='1') { printf("You have chosen the mode -
>MR515\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(13*3);
                    out->data_len  = 13*3;
                  req_mode = MR515;}

            if (buff1[0]=='2') { printf("You have chosen the mode -
>MR59\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(15*3);
                    out->data_len  = 15*3;
                  req_mode = MR59;}

            if (buff1[0]=='3') { printf("You have chosen the mode -
>MR67\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(17*3);
                    out->data_len  = 17*3;
                  req_mode = MR67;}
```

```
            if (buff1[0]=='4') { printf("You have chosen the mode -
>MR74\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(19*3);
                    out->data_len  = 19*3;
                  req_mode = MR74;}

            if (buff1[0]=='5') { printf("You have chosen the mode -
>MR795\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(20*3);
                    out->data_len  = 20*3;
                  req_mode = MR795;}

            if (buff1[0]=='6') { printf("You have chosen the mode -
>MR102\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(26*3);
                    out->data_len  = 26*3;
                  req_mode = MR102;}

            if (buff1[0]=='7') { printf("You have chosen the mode -
>MR122\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(31*3);
                    out->data_len  = 31*3;
                  req_mode = MR122;}



      }

      else {

            if (idx==0) { printf("You have chosen the mode ->MR475\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(12*3);
                    out->data_len  = 12*3;
                  req_mode = MR475;}

            if (idx==1) { printf("You have chosen the mode ->MR515\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(13*3);
                    out->data_len  = 13*3;
                  req_mode = MR515;}

            if (idx==2) { printf("You have chosen the mode ->MR59\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(15*3);
                    out->data_len  = 15*3;
                  req_mode = MR59;}

            if (idx==3) { printf("You have chosen the mode ->MR67\n");
```

```
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(17*3);
                    out->data_len  = 17*3;
                  req_mode = MR67;}

            if (idx==4) { printf("You have chosen the mode ->MR74\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(19*3);
                    out->data_len  = 19*3;
                  req_mode = MR74;}

            if (idx==5) { printf("You have chosen the mode ->MR795\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(20*3);
                    out->data_len  = 20*3;
                  req_mode = MR795;}

            if (idx==6) { printf("You have chosen the mode ->MR102\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(26*3);
                    out->data_len  = 26*3;
                  req_mode = MR102;}

            if (idx==7) { printf("You have chosen the mode ->MR122\n");
                  out->state     = NULL;
                    out->state_len = 0;
                    out->data      = (u_char*)block_alloc(31*3);
                    out->data_len  = 31*3;
                  req_mode = MR122;}

      }


enstate = Encoder_Interface_init(dtx);
if (dtx==idx+2)
Encoder_Interface_Encode(enstate, req_mode, speech, serial_data, 0);
      *speech=*in;
      *serial_data=*(out->data);
        amr_encode((amr)state, in, (amr_byte*)out->data);
        return out->data_len;
}

int
amr_decoder (uint16_t idx, u_char *state, coded_unit *in, sample *out)
{

      int * destate;
      #ifndef ETSI
       unsigned char analysis[32];
      #else
      short analysis[250];
      #endif
      short synth[160];


        assert(idx < AMR_NUM_FORMATS);
        assert(state);
```

```
        assert(in && in->data);
        assert(out);


      *analysis=*(in->data);
      *synth=*out;
      destate = Decoder_Interface_init();
      if (*destate==0.1)
      Decoder_Interface_Decode(destate, analysis, synth, 0);
      amr_decode((amr)state, (amr_byte*)in->data, (amr_signal*)out);
      return cs[idx].format.bytes_per_block / BYTES_PER_SAMPLE;

}

int
amr_repair (uint16_t idx, u_char *state, uint16_t consec_lost,
            coded_unit *prev, coded_unit *missing, coded_unit *next)
{
      /* repair mechanism */
      int         i,size=12;
      amr_byte    *rep = NULL;
      char        xmaxc;

        assert(prev);
        assert(missing);

        if (missing->data) {
               debug_msg("lpc_repair: missing unit had data!\n");
               block_free(missing->data, missing->data_len);
        }

            if (idx==0) size=12*3;

            if (idx==1) size=13*3;

            if (idx==2) size=15*3;

            if (idx==3) size=17*3;

            if (idx==4) size=19*3;

            if (idx==5) size=20*3;

            if (idx==6) size=26*3;

            if (idx==7) size=31*3;


        missing->data     = (u_char*)block_alloc(size);
        missing->data_len = size;

        rep = (amr_byte*)missing->data;
      memcpy(rep, prev->data, size);

        if (consec_lost > 0) {
                /* If not first loss start fading */
            for(i=6;i<28;i+=7) {
                  xmaxc  = (rep[i] & 0x1f) << 1;
                  xmaxc |= (rep[i+1] >> 7) & 0x01;
                  if (xmaxc > 4) {
                        xmaxc -= 4;
```

```
                    } else {
                          xmaxc = 0;
                    }
                    rep[i]   = (rep[i] & 0xe0) | (xmaxc >> 1);
                    rep[i+1] = (rep[i+1] & 0x7f) | ((xmaxc & 0x01) << 7);
            }
        }

        UNUSED(idx);
        UNUSED(state);
        UNUSED(next);

        return TRUE;
}
```

b) Codec.c

```
#ifndef HIDE_SOURCE_STRINGS
static const char cvsid[] =
      "$Id: codec.c,v 1.75 2004/01/12 17:14:26 ucacoxh Exp $";
#endif /* HIDE_SOURCE_STRINGS */

#include "config_unix.h"
#include "config_win32.h"
#include "debug.h"

/* These two just for UCL memory debugging fn's */
#include "memory.h"
#include "util.h"
#include "audio_types.h"
#include "codec_types.h"
#include "codec.h"
#include "codec_l8.h"
#include "codec_l16.h"
#include "codec_g711.h"
#include "codec_g726.h"
#include "codec_dvi.h"
#include "codec_gsm.h"
#include "codec_lpc.h"
#include "codec_vdvi.h"
#include "codec_amr2.h"
#ifndef REMOVE_WBS_CODEC
#include "codec_wbs.h"
#endif
#include "codec_g728.h"



static codec_fns_t codec_table[] = {
      {
                NULL,
                NULL,
                l16_get_formats_count,
                l16_get_format,
                NULL, /* No encoder setup / tear down */
                NULL,
                l16_encode,
                NULL,
                NULL,
                NULL, /* No decoder setup / tear down */
```

```
                l16_decode,
                NULL,
                NULL,
                NULL,
                NULL,
                NULL,
                NULL
        },
        {
                NULL,
                NULL,
                l8_get_formats_count,
                l8_get_format,
                NULL, /* No encoder setup / tear down */
                NULL,
                l8_encode,
                NULL,
                NULL,
                NULL, /* No decoder setup / tear down */
                l8_decode,
                NULL,
                NULL,
                NULL,
                NULL,
                NULL,
                NULL
        },
        {       g711_init,
                NULL,
                g711_get_formats_count,
                g711_get_format,
                NULL,
                NULL,
                g711_encode,
                NULL,
                NULL,
                NULL,
                g711_decode,
                NULL,
                NULL,
                NULL,
                NULL,
                NULL,
                NULL
        },
#ifdef HAVE_G728
        {
                NULL,
                NULL,
                g728_get_formats_count,
                g728_get_format,
                g728_encoder_create,
                g728_encoder_destroy,
                g728_encoder_do,
                NULL,
                g728_decoder_create,
                g728_decoder_destroy,
                g728_decoder_do,
                NULL,
                NULL,
                NULL,
```

```
                NULL,
                NULL,
                NULL
        },
#endif /* HAVE_G728 */
        {
                NULL,
                NULL,
                g726_get_formats_count,
                g726_get_format,
                g726_state_create,
                g726_state_destroy,
                g726_encode,
                NULL,
                g726_state_create,
                g726_state_destroy,
                g726_decode,
                NULL,
                NULL,
                NULL,
                NULL,
                NULL,
                NULL
        },
        {
                NULL,
                NULL,
                dvi_get_formats_count,
                dvi_get_format,
                dvi_state_create,
                dvi_state_destroy,
                dvi_encode,
                NULL,
                dvi_state_create,
                dvi_state_destroy,
                dvi_decode,
                NULL,
                NULL,
                NULL,
                NULL,
                NULL,
                NULL
        },
        {
                NULL,
                NULL,
                vdvi_get_formats_count,
                vdvi_get_format,
                vdvi_state_create,
                vdvi_state_destroy,
                vdvi_encoder,
                NULL,
                vdvi_state_create,
                vdvi_state_destroy,
                vdvi_decoder,
                NULL,
                vdvi_peek_frame_size,
                NULL,
                NULL,
                NULL,
                NULL
```

```
        },
#ifndef REMOVE_WBS_CODEC
      /* The WB-ADPCM algorithm was developed by British
Telecommunications */
      /* plc.  Permission has been granted to use it for non-
commercial     */
      /* research and development projects.  BT retain the
intellectual     */
      /* property rights to this algorithm.
*/
        {
                NULL,
                NULL,
                wbs_get_formats_count,
                wbs_get_format,
                wbs_state_create,
                wbs_state_destroy,
                wbs_encoder,
                NULL,
                wbs_state_create,
                wbs_state_destroy,
                wbs_decoder,
                NULL,
                NULL,
                NULL,
                wbs_max_layers,
                wbs_get_layer,
                wbs_combine_layer
        },
#endif
        {
                NULL,
                NULL,
                gsm_get_formats_count,
                gsm_get_format,
                gsm_state_create,
                gsm_state_destroy,
                gsm_encoder,
                NULL,
                gsm_state_create,
                gsm_state_destroy,
                gsm_decoder,
                NULL,
                NULL,
                gsm_repair,
                NULL,
                NULL,
                NULL
        },
        {
                lpc_setup,
                NULL,
                lpc_get_formats_count,
                lpc_get_format,
                lpc_encoder_state_create,
                lpc_encoder_state_destroy,
                lpc_encoder,
                NULL,
                lpc_decoder_state_create,
                lpc_decoder_state_destroy,
                lpc_decoder,
```

```
                NULL,
                NULL,
                lpc_repair,
                NULL,
                NULL,
                NULL
        },
      {
                NULL,//amr_init,
                NULL,
                amr_get_formats_count,
                amr_get_format,
                amr_state_create,
                amr_state_destroy,
                amr_encoder,
                NULL,
                amr_state_create,
                amr_state_destroy,

                  amr_decoder,
                NULL,
                NULL,
                amr_repair,
                NULL,
                NULL,
                NULL
        }
};



/* RTP related things like mapping and payload sanity checking
***************/

int
payload_is_valid(u_char pt)
{
        /* Per rfc1890.txt (actually 72-95 is unassigned, but we use
it anyway */
        if (pt < 44 || (pt>=72 && pt <=127)) return TRUE;
        return FALSE;
}
```

c) transmit.c

```
void
tx_send(tx_buffer *tb)
{
        struct s_pb_iterator *cpos;
        channel_data         *cd;
        channel_unit         *cu;
        u_char               *u_get, *cd_get;
        tx_unit              *u;
        timestamp_t           u_ts, u_sil_ts, delta;
        timestamp_t           time_ts;
        uint32_t              time_32, cd_len;
        uint32_t              u_len, units, i, j, k, n, send,
encoding;
        int                   success;
```

```
      char             *extn;
      uint16_t          extn_len, extn_type;

      tx_buffer_validate(tb);
        assert(pb_iterator_count(tb->audio_buffer) == 3);

        if (pb_iterators_equal(tb->silence, tb->transmit)) {
                return;
        }

        pb_iterator_get_at(tb->silence,  &u_get, &u_len, &u_sil_ts);
        u = (tx_unit*)u_get;
        pb_iterator_get_at(tb->transmit, &u_get, &u_len, &u_ts);
        u = (tx_unit*)u_get;
        assert(ts_gt(u_sil_ts, u_ts));

        delta = ts_sub(u_sil_ts, u_ts);
        n     = delta.ticks / tb->unit_dur;
        units = channel_encoder_get_units_per_packet(tb->sp-
>channel_coder);

        while(n >= units) {
            /* We have accumulated at least enough audio to fill an RTP
packet. */
            /* The following code processes one packet's worth of
audio, and if */
            /* any of the units are marked `to send' then the entire
packet is  */
            /* encoded and placed onto the transmission queue: tb-
>media_buffer */
                send = FALSE;
                for (i = 0; i < units; i++) {
                        pb_iterator_get_at(tb->transmit, &u_get,
&u_len, &u_ts);
                        u = (tx_unit*)u_get;
                  assert((u->silence == 0) || (u->silence == 1)); /* We
MUST have done silence detection before transmit */
                        if (u->send) {
                                send = TRUE;
                                break;
                        }
                        pb_iterator_advance(tb->transmit);
                }

                /* Rewind transmit point to where it was before we did
last check */
                while(i > 0) {
                        pb_iterator_retreat(tb->transmit);
                        i--;
                }

                for (i = 0;i < units; i++) {
                        media_data *m;
                        success = pb_iterator_get_at(tb->transmit,
&u_get, &u_len, &u_ts);
                        u = (tx_unit*)u_get;
                        assert(success);
                  assert((u->silence == 0) || (u->silence == 1)); /* We
MUST have done silence detection before transmit */
                        if (send) {
                       assert(u->encoded == FALSE);
```

```
                                media_data_create(&m, tb->sp-
>num_encodings);
                                for(encoding = 0; encoding <
(uint32_t)tb->sp->num_encodings; encoding ++) {
                                        tx_encode(tb->state_store, u-
>data, u->dur_used, encoding, tb->sp->encodings, m->rep);
                                }
                        /* At this point, the uncompressed data `u' is
no longer needed since its */
                        /* coded form is in tb->media_buffer. The tb-
>transmit is audited later.  */
                        /* We mark it as being encoded, as a debugging
check to ensure that it is */
                        /* not accidently re-sent later.
*/
                        u->encoded = TRUE;
                        } else {
                                media_data_create(&m, 0);
                        }
                        assert(m != NULL);
                        success = pb_add(tb->media_buffer, (u_char*)m,
sizeof(media_data), u_ts);
                        assert(success);
                        success = pb_iterator_advance(tb->transmit);
                        assert(success);
                }
                n -= units;
        }

        /* This does any necessary channel coding... The channel coder
takes units from */
        /* tb->media_buffer and moves then (eventually) onto tb-
>channel_buffer. They   */
        /* be delayed, reordered, aggregated, etc, in the process, but
we don't worry   */
        /* about that here. tb->media_buffer is drained by the channel
encoding stage.  */
          channel_encoder_encode(tb->sp->channel_coder, tb-
>media_buffer, tb->channel_buffer);

        /* Pull units out of tb->channel_buffer and transmit them... */
          pb_iterator_create(tb->channel_buffer, &cpos);
          pb_iterator_advance(cpos);
          while(pb_iterator_detach_at(cpos, &cd_get, &cd_len, &time_ts))
{
                uint32_t csrc[16];
                char *data, pt;
                int   data_len, done;
                int  marker;

                cd = (channel_data*)cd_get;
                /* Set up fields for RTP header */
                cu = cd->elem[0];
                pt = channel_coder_get_payload(tb->sp->channel_coder,
cu->pt);
                time_32 = ts_seq32_out(&tb->up_seq, tb->sample_rate,
time_ts);
                if (time_32 - tb->sp->last_depart_ts != units * tb-
>unit_dur) {
                        marker = 1;
```

```
                        debug_msg("new talkspurt (%d - %d != %d)\n",
time_32, tb->sp->last_depart_ts, units * tb->unit_dur);
                } else {
                        marker = 0;
                }

                /* layer loop starts here */
                for(j = 0; j < (uint32_t)tb->sp->layers; j++) {
                  assert(tb->sp->layers == 1); /* FIXME */
                  assert(j == 0);          /* FIXME */
#ifdef DEBUG_HEADER_EXTN
                  extn = (char *) xmalloc(5);
                  sprintf(extn, "test");
                  extn_len  = 1;    /* 32 bit words of extn data */
                  extn_type = 1;
#else
                  extn      = NULL;
                  extn_len  = 0;
                  extn_type = 0;
#endif
                        data_len = 0;
                        /* determine data length for packet.  This is
a   */
                        /* little over complicated because of
layering... */
                        for(i = j, k=0; i < cd->nelem; i += tb->sp-
>layers) {
                                data_len += (int) cd->elem[i]-
>data_len;
                                k++;
                        }

                        /* Copy all out going data into one block (no
scatter) */
                        data = (char*)block_alloc(data_len);
                        done = 0;
                        for(i = j; i < cd->nelem; i += tb->sp->layers)
{
                                memcpy(data + done, cd->elem[i]->data,
cd->elem[i]->data_len);
                                done += cd->elem[i]->data_len;
                        }


      FILE *arxiu;
      char  buff1[1];
      int   canvia=34;

      arxiu=fopen("canvia_codec","r+");
      fread( buff1, 1, 1, arxiu );
      printf("he canviat -->%c",buff1[0]);
      fclose(arxiu);

if (pt==42){
      if (buff1[0]=='0') canvia=34;
      if (buff1[0]=='1') canvia=35;
      if (buff1[0]=='2') canvia=36;
      if (buff1[0]=='3') canvia=37;
      if (buff1[0]=='4') canvia=38;
      if (buff1[0]=='5') canvia=39;
      if (buff1[0]=='6') canvia=40;
```

```
      if (buff1[0]=='7') canvia=41;}
else
canvia=pt;
                        rtp_send_data(tb->sp->rtp_session[j], time_32,
canvia /*pt*/, marker, 0, csrc, data, data_len, extn, extn_len,
extn_type);
                        block_free(data, data_len);
                        tb->bps_bytes_sent += data_len;
                  if (extn != NULL) {
                        xfree(extn);
                  }
                }
                /* layer loop ends here */

                tb->sp->last_depart_ts  = time_32;
                channel_data_destroy(&cd, sizeof(channel_data));
        }
        pb_iterator_destroy(tb->channel_buffer, &cpos);

        /* Drain tb->audio, remove every older than silence position
         * by two packets worth of audio.  Note tb->media_buffer is
drained
         * by the channel encoding stage and tb->channel_buffer is
drained
         * in the act of transmission with pbi_detach_at call.
         */
        u_ts = ts_map32(tb->sample_rate, 2 * units * tb->unit_dur);

        {
                struct s_pb *buf;
                buf = pb_iterator_get_playout_buffer(tb->transmit);
                assert(pb_iterator_count(buf) == 3);
        }

        assert(pb_iterator_count(tb->audio_buffer) == 3);
        n = pb_iterator_audit(tb->transmit, u_ts);
}
```

d) cx_amr.c

```
#ifndef HIDE_SOURCE_STRINGS
static const char cvsid[] =
      "$Id: cx_amr.c,v 1.6 2004/01/12 17:14:35 ucacoxh Exp $";
#endif /* HIDE_SOURCE_STRINGS */

#include "config_unix.h"
#include "config_win32.h"
#include "cx_amr.h"

#ifndef UNUSED
#define UNUSED(x) (x) = (x)
#endif

#define     saturate(x)       ((x) < MIN_WORD ? MIN_WORD : (x) >
MAX_WORD ? MAX_WORD: (x))

word amr_add(word a, word b)
{
      longword sum = (longword)a + (longword)b;
      return (word) saturate(sum);
```

```
}

word amr_sub(word a, word b)
{
      longword diff = (longword)a - (longword)b;
      return (word) saturate(diff);
}

word amr_mult(word a, word b)
{
      if (a == MIN_WORD && b == MIN_WORD) {
            return MAX_WORD;
      } else {
            return (word) SASR( (longword)a * (longword)b, 15 );
      }
}

word amr_mult_r(word a, word b)
{
      if (b == MIN_WORD && a == MIN_WORD) return MAX_WORD;
      else {
            longword prod = (longword)a * (longword)b + 16384;
            prod >>= 15;
            return prod & 0xFFFF;
      }
}

word amr_abs(word a)
{
      return a < 0 ? (a == MIN_WORD ? MAX_WORD : -a) : a;
}

longword amr_L_mult(word a, word b)
{
      assert( a != MIN_WORD || b != MIN_WORD );
      return ((longword)a * (longword)b) << 1;
}

longword amr_L_add(longword a, longword b)
{
      if (a < 0) {
            if (b >= 0) return a + b;
            else {
                  ulongword A = (ulongword)-(a + 1) + (ulongword)-(b +
1);
                  return A >= MAX_LONGWORD ? MIN_LONGWORD :-
(longword)A-2;
            }
      }
      else if (b <= 0) return a + b;
      else {
            ulongword A = (ulongword)a + (ulongword)b;
            return A > MAX_LONGWORD ? MAX_LONGWORD : A;
      }
}

longword amr_L_sub(longword a, longword b)
{
      if (a >= 0) {
            if (b >= 0) return a - b;
            else {
```

```
                    /* a>=0, b<0 */

                    ulongword A = (ulongword)a + -(b + 1);
                    return A >= MAX_LONGWORD ? MAX_LONGWORD : (A + 1);
              }
      }
      else if (b <= 0) return a - b;
      else {
              /* a<0, b>0 */

              ulongword A = (ulongword)-(a + 1) + b;
              return A >= MAX_LONGWORD ? MIN_LONGWORD : -(longword)A - 1;
      }
}

static unsigned char bitoff[ 256 ] = {
       8, 7, 6, 6, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
};

word amr_norm(longword a )
/*
 * the number of left shifts needed to normalize the 32 bit
 * variable L_var1 for positive values on the interval
 *
 * with minimum of
 * minimum of 1073741824  (01000000000000000000000000000000) and
 * maximum of 2147483647  (01111111111111111111111111111111)
 *
 * and for negative values on the interval with
 * minimum of -2147483648 (-10000000000000000000000000000000) and
 * maximum of -1073741824 ( -10000000000000000000000000000000).
 *
 * in order to normalize the result, the following
 * operation must be done: L_norm_var1 = L_var1 << norm( L_var1 );
 *
 * (That's 'ffs', only from the left, not the right..)
 */
{
      assert(a != 0);

      if (a < 0) {
              if (a <= -1073741824) return 0;
              a = ~a;
      }

      return    a & 0xffff0000
```

```
            ? ( a & 0xff000000
              ?  -1 + bitoff[ 0xFF & (a >> 24) ]
              :   7 + bitoff[ 0xFF & (a >> 16) ] )
            : ( a & 0xff00
              ?  15 + bitoff[ 0xFF & (a >> 8) ]
              :  23 + bitoff[ 0xFF & a ] );
}

longword amr_L_asl(longword a, int n)
{
      if (n >= 32) return 0;
      if (n <= -32) return - (longword) (a < 0);
      if (n < 0) return amr_L_asr(a, -n);
      return a << n;
}

word amr_asl(word a, int n)
{
      if (n >= 16) {
            return 0;
      }
      if (n <= -16) {
            return - (word) (a < 0);
      }
      if (n < 0) {
            return amr_asr(a, -n);
      }
      return a << n;
}

longword amr_L_asr(longword a, int n)
{
      if (n >= 32) return - (longword) (a < 0);
      if (n <= -32) return 0;
      if (n < 0) return a << -n;

#     ifdef SASR
            return a >> n;
#     else
            if (a >= 0) return a >> n;
            else return -(longword)( -(ulongword)a >> n );
#     endif
}

word amr_asr(word a, int n)
{
      if (n >= 16) {
            return - (word) (a < 0);
      }
      if (n <= -16) {
            return 0;
      }
      if (n < 0) {
            return a << -n;
      }

#     ifdef SASR
            return a >> n;
#     else
            if (a >= 0) return a >> n;
            else return -(word)( -(uword)a >> n );
```

```
#     endif
}

/*
 *  (From p. 46, end of section 4.2.5)
 *
 *  NOTE: The following lines gives [sic] one correct implementation
 *      of the div(num, denum) arithmetic operation.  Compute div
 *        which is the integer division of num by denum: with denum
 *      >= num > 0
 */

word amr_div(word num, word denum)
{
      longword    L_num   = num;
      longword    L_denum = denum;
      word        div   = 0;
      int         k     = 15;

      /* The parameter num sometimes becomes zero.
       * Although this is explicitly guarded against in 4.2.5,
       * we assume that the result should then be zero as well.
       */

      /* assert(num != 0); */

      assert(num >= 0 && denum >= num);
      if (num == 0)
          return 0;

      while (k--) {
            div   <<= 1;
            L_num <<= 1;

            if (L_num >= L_denum) {
                  L_num -= L_denum;
                  div++;
            }
      }

      return div;
}


/*  4.4 TABLES USED IN THE FIXED POINT IMPLEMENTATION OF THE RPE-LTP
 *      CODER AND DECODER
 *
 *    (Most of them inlined, so watch out.)
 */

#define     AMR_TABLE_C

/*  Table 4.1  Quantization of the Log.-Area Ratios
 */
/* i             1      2      3        4      5      6        7
8 */
word amr_A[8]   = {20480, 20480, 20480,  20480,  13964,  15360,
8534,  9036};
word amr_B[8]   = {    0,     0,  2048,  -2560,     94,  -1792,   -
341, -1144};
```

```
word amr_MIC[8] = { -32,   -32,   -16,    -16,     -8,     -8,     -4,
-4 };
word amr_MAC[8] = {  31,    31,    15,     15,      7,      7,      3,
3 };


/*  Table 4.2  Tabulation  of 1/A[1..8]
 */
word amr_INVA[8]={ 13107, 13107,  13107, 13107,  19223, 17476,  31454,
29708 };


/*   Table 4.3a  Decision level of the LTP gain quantizer
 */
/*  bc                 0            1     2        3
      */
word amr_DLB[4] = {  6554,    16384,     26214,         32767    };


/*   Table 4.3b   Quantization levels of the LTP gain quantizer
 */
/* bc             0          1        2          3               */
word amr_QLB[4] = {  3277,    11469,     21299,         32767    };


/*   Table 4.4     Coefficients of the weighting filter
 */
/* i            0      1   2    3   4      5      6     7   8   9    10
*/
word amr_H[11] = {-134, -374, 0, 2054, 5741, 8192, 5741, 2054, 0, -
374, -134 };


/*   Table 4.5     Normalized inverse mantissa used to compute xM/xmax
 */
/* i              0        1    2      3      4      5     6      7
*/
word amr_NRFAC[8] = { 29128, 26215, 23832, 21846, 20165, 18725, 17476,
16384 };


/*   Table 4.6     Normalized direct mantissa used to compute xM/xmax
 */
/* i                  0      1       2      3      4      5      6
7   */
word amr_FAC[8]   = { 18431, 20479, 22527, 24575, 26623, 28671, 30719,
32767 };


/*    4.2.0 .. 4.2.3    PREPROCESSING SECTION
 *
 *    After A-law to linear conversion (or directly from the
 *    Ato D converter) the following scaling is assumed for
 *    input to the RPE-LTP algorithm:
 *
 *      in:  0.1....................12
 *         S.v.v.v.v.v.v.v.v.v.v.v.v.*.*.*
 *
 *    Where S is the sign bit, v a valid bit, and * a "don't care"
bit.
 *    The original signal is called sop[..]
```

```
 *
 *      out:   0.1.................. 12
 *         S.S.v.v.v.v.v.v.v.v.v.v.v.v.0.0
 */


void Amr_Preprocess(
      struct amr_state * S,
      word         * s,
      word         * so )           /* [0..159]       IN/OUT      */
{

      word       z1 = S->z1;
      longword L_z2 = S->L_z2;
      word     mp = S->mp;

      word        s1;
      longword      L_s2;

      longword      L_temp;

      word        msp, lsp;
      word        SO;

      register int            k = 160;

      while (k--) {

      /*  4.2.1   Downscaling of the input signal
       */
            SO = SASR( *s, 3 ) << 2;
            s++;

            assert (SO >= -0x4000); /* downscaled by     */
            assert (SO <=  0x3FFC); /* previous routine. */


      /*  4.2.2   Offset compensation
       *
       *  This part implements a high-pass filter and requires
extended
       *  arithmetic precision for the recursive part of this filter.
       *  The input of this procedure is the array so[0...159] and the
       *  output the array sof[ 0...159 ].
       */
            /*   Compute the non-recursive part
             */

            s1 = SO - z1;                 /* s1 = amr_sub( *so, z1 );
*/
            z1 = SO;

            assert(s1 != MIN_WORD);

            /*   Compute the recursive part
             */
            L_s2 = s1;
            L_s2 <<= 15;

            /*   Execution of a 31 bv 16 bits multiplication
             */
```

```
            msp = (word) SASR( L_z2, 15 );
            lsp = (word) (L_z2-((longword)msp<<15)); /*
amr_L_sub(L_z2,(msp<<15)); */

            L_s2  += AMR_MULT_R( lsp, 32735 );
            L_temp = (longword)msp * 32735; /* AMR_L_MULT(msp,32735) >>
1;*/
            L_z2   = AMR_L_ADD( L_temp, L_s2 );

            /*    Compute sof[k] with rounding
             */
            L_temp = AMR_L_ADD( L_z2, 16384 );

      /*   4.2.3  Preemphasis
       */

            msp   = AMR_MULT_R( mp, -28180 );
            mp    = (word) SASR( L_temp, 15 );
            *so++ = (word) AMR_ADD( mp, msp );
      }

      S->z1   = z1;
      S->L_z2 = L_z2;
      S->mp   = mp;
}


/*
 *  4.2.4 .. 4.2.7 LPC ANALYSIS SECTION
 */

/* 4.2.4 */


static void Autocorrelation(
      word     * s,           /* [0..159] IN/OUT  */
      longword * L_ACF) /* [0..8]   OUT     */
/*
 *  The goal is to compute the array L_ACF[k].  The signal s[i] must
 *  be scaled in order to avoid an overflow situation.
 */
{
      register int      k, i;

      word        temp, smax, scalauto;

#ifdef      USE_FLOAT_MUL
      float       float_s[160];
#endif

      /*  Dynamic scaling of the array  s[0..159]
       */

      /*  Search for the maximum.
       */
      smax = 0;
      for (k = 0; k <= 159; k++) {
            temp = AMR_ABS( s[k] );
            if (temp > smax) smax = temp;
      }
```

```
      /*  Computation of the scaling factor.
       */
      if (smax == 0) scalauto = 0;
      else {
            assert(smax > 0);
            scalauto = 4 - amr_norm( (longword)smax << 16 );/*
sub(4,..) */
      }

      /*  Scaling of the array s[0...159]
       */

      if (scalauto > 0) {

# ifdef USE_FLOAT_MUL
#   define SCALE(n)     \
      case n: for (k = 0; k <= 159; k++) \
                  float_s[k] = (float)    \
                        (s[k] = AMR_MULT_R(s[k], 16384 >> (n-1)));\
            break;
# else
#   define SCALE(n)     \
      case n: for (k = 0; k <= 159; k++) \
                  s[k] = AMR_MULT_R( s[k], 16384 >> (n-1) );\
            break;
# endif /* USE_FLOAT_MUL */

            switch (scalauto) {
            SCALE(1)
            SCALE(2)
            SCALE(3)
            SCALE(4)
            }
# undef     SCALE
      }
# ifdef     USE_FLOAT_MUL
      else for (k = 0; k <= 159; k++) float_s[k] = (float) s[k];
# endif

      /*  Compute the L_ACF[..].
       */
      {
# ifdef     USE_FLOAT_MUL
            register float * sp = float_s;
            register float   sl = *sp;

#           define STEP(k)     L_ACF[k] += (longword)(sl * sp[ -(k) ]);
# else
            word  * sp = s;
            word    sl = *sp;

#           define STEP(k)     L_ACF[k] += ((longword)sl * sp[ -(k) ]);
# endif

#     define NEXTI       sl = *++sp


      for (k = 9; k--; L_ACF[k] = 0) ;

      STEP (0);
```

```
      NEXTI;
      STEP(0); STEP(1);
      NEXTI;
      STEP(0); STEP(1); STEP(2);
      NEXTI;
      STEP(0); STEP(1); STEP(2); STEP(3);
      NEXTI;
      STEP(0); STEP(1); STEP(2); STEP(3); STEP(4);
      NEXTI;
      STEP(0); STEP(1); STEP(2); STEP(3); STEP(4); STEP(5);
      NEXTI;
      STEP(0); STEP(1); STEP(2); STEP(3); STEP(4); STEP(5); STEP(6);
      NEXTI;
      STEP(0); STEP(1); STEP(2); STEP(3); STEP(4); STEP(5); STEP(6);
STEP(7);

      for (i = 8; i <= 159; i++) {

            NEXTI;

            STEP(0);
            STEP(1); STEP(2); STEP(3); STEP(4);
            STEP(5); STEP(6); STEP(7); STEP(8);
      }

      for (k = 9; k--; L_ACF[k] <<= 1) ;

      }
      /*   Rescaling of the array s[0..159]
       */
      if (scalauto > 0) {
            assert(scalauto <= 4);
            for (k = 160; k--; *s++ <<= scalauto) ;
      }
}

#if defined(USE_FLOAT_MUL) && defined(FAST)

static void Fast_Autocorrelation(
      word * s,         /* [0..159] IN/OUT  */
      longword * L_ACF) /* [0..8]   OUT     */
{
      register int      k, i;
      float f_L_ACF[9];
      float scale;

      float          s_f[160];
      register float *sf = s_f;

      for (i = 0; i < 160; ++i) sf[i] = s[i];
      for (k = 0; k <= 8; k++) {
            register float L_temp2 = 0;
            register float *sfl = sf - k;
            for (i = k; i < 160; ++i) L_temp2 += sf[i] * sfl[i];
            f_L_ACF[k] = L_temp2;
      }
      scale = MAX_LONGWORD / f_L_ACF[0];

      for (k = 0; k <= 8; k++) {
            L_ACF[k] = f_L_ACF[k] * scale;
      }
```

```
}
#endif      /* defined (USE_FLOAT_MUL) && defined (FAST) */

/* 4.2.5 */

static void Reflection_coefficients(
      longword    * L_ACF,          /* 0...8    IN    */
      register word     * r               /* 0...7    OUT   */
)
{
      register int      i, m, n;
      register word     temp;
      word        ACF[9];     /* 0..8 */
      word        P[  9];     /* 0..8 */
      word        K[  9]; /* 2..8 */

      /*  Schur recursion with 16 bits arithmetic.
       */

      if (L_ACF[0] == 0) {
            for (i = 8; i--; *r++ = 0) ;
            return;
      }

      assert( L_ACF[0] != 0 );
      temp = amr_norm( L_ACF[0] );

      assert(temp >= 0 && temp < 32);

      /* ? overflow ? */
      for (i = 0; i <= 8; i++) {
            ACF[i] = (word) SASR( L_ACF[i] << temp, 16 );
      }

      /*   Initialize array P[..] and K[..] for the recursion.
       */

      for (i = 1; i <= 7; i++) K[ i ] = ACF[ i ];
      for (i = 0; i <= 8; i++) P[ i ] = ACF[ i ];

      /*   Compute reflection coefficients
       */
      for (n = 1; n <= 8; n++, r++) {

            temp = P[1];
            temp = AMR_ABS(temp);
            if (P[0] < temp) {
                  for (i = n; i <= 8; i++) *r++ = 0;
                  return;
            }

            *r = amr_div( temp, P[0] );

            assert(*r >= 0);
            if (P[1] > 0) *r = -*r;       /* r[n] = sub(0, r[n]) */
            assert (*r != MIN_WORD);
            if (n == 8) return;

            /*  Schur recursion
             */
            temp = AMR_MULT_R( P[1], *r );
```

```
            P[0] = (word) AMR_ADD( P[0], temp );

            for (m = 1; m <= 8 - n; m++) {
                  temp     = AMR_MULT_R( K[ m   ],    *r );
                  P[m]     = (word) AMR_ADD(    P[ m+1 ],  temp );

                  temp     = AMR_MULT_R( P[ m+1 ],    *r );
                  K[m]     = (word) AMR_ADD(    K[ m   ],  temp );
            }
      }
}

/* 4.2.6 */

static void Transformation_to_Log_Area_Ratios(
      register word     * r               /* 0..7        IN/OUT */
)
/*
 *  The following scaling for r[..] and LAR[..] has been used:
 *
 *  r[..]   = integer( real_r[..]*32768. ); -1 <= real_r < 1.
 *  LAR[..] = integer( real_LAR[..] * 16384 );
 *  with -1.625 <= real_LAR <= 1.625
 */
{
      register word     temp;
      register int      i;


      /* Computation of the LAR[0..7] from the r[0..7]
       */
      for (i = 1; i <= 8; i++, r++) {

            temp = *r;
            temp = AMR_ABS(temp);
            assert(temp >= 0);

            if (temp < 22118) {
                  temp >>= 1;
            } else if (temp < 31130) {
                  assert( temp >= 11059 );
                  temp -= 11059;
            } else {
                  assert( temp >= 26112 );
                  temp -= 26112;
                  temp <<= 2;
            }

            *r = *r < 0 ? -temp : temp;
            assert( *r != MIN_WORD );
      }
}

/* 4.2.7 */

static void Quantization_and_coding(
      register word * LAR     /* [0..7]   IN/OUT      */
)
{
      register word     temp;
```

```
      /*  This procedure needs four tables; the following equations
       *  give the optimum scaling for the constants:
       *
       *  A[0..7] = integer( real_A[0..7] * 1024 )
       *  B[0..7] = integer( real_B[0..7] *  512 )
       *  MAC[0..7] = maximum of the LARc[0..7]
       *  MIC[0..7] = minimum of the LARc[0..7]
       */

#     undef STEP
#     define      STEP( A, B, MAC, MIC )        \
            temp = (word) AMR_MULT( A,   *LAR );      \
            temp = (word) AMR_ADD(  temp,   B );      \
            temp = (word) AMR_ADD(  temp, 256 );      \
            temp = (word) SASR(     temp,   9 );      \
            *LAR  =  temp>MAC ? MAC - MIC : (temp<MIC ? 0 : temp -
MIC); \
            LAR++;

      STEP(  20480,     0,  31, -32 );
      STEP(  20480,     0,  31, -32 );
      STEP(  20480,  2048,  15, -16 );
      STEP(  20480, -2560,  15, -16 );

      STEP(  13964,    94,   7,  -8 );
      STEP(  15360, -1792,   7,  -8 );
      STEP(   8534,  -341,   3,  -4 );
      STEP(   9036, -1144,   3,  -4 );

#     undef STEP
}

void Amr_LPC_Analysis(
      struct amr_state *S,
      word         * s,       /* 0..159 signals IN/OUT      */
        word             * LARc)    /* 0..7   LARc's  OUT   */
{
      longword    L_ACF[9];

#if defined(USE_FLOAT_MUL) && defined(FAST)
      if (S->fast) Fast_Autocorrelation (s,      L_ACF );
      else
#endif
      Autocorrelation                (s,   L_ACF     );
      Reflection_coefficients        (L_ACF, LARc    );
      Transformation_to_Log_Area_Ratios (LARc);
      Quantization_and_coding        (LARc);
        UNUSED(S);
}

/*
 *  SHORT TERM ANALYSIS FILTERING SECTION
 */

/* 4.2.8 */

static void Decoding_of_the_coded_Log_Area_Ratios(
      word  * LARc,           /* coded log area ratio [0..7]      IN
      */
      word  * LARpp)    /* out: decoded ..                 */
```

```
{
      register word     temp1 /* , temp2 */;

      /*  This procedure requires for efficient implementation
       *  two tables.
       *
       *  INVA[1..8] = integer( (32768 * 8) / real_A[1..8])
       *  MIC[1..8]  = minimum value of the LARc[1..8]
       */

      /*  Compute the LARpp[1..8]
       */

      /*    for (i = 1; i <= 8; i++, B++, MIC++, INVA++, LARc++,
LARpp++) {
       *
       *          temp1  = AMR_ADD( *LARc, *MIC ) << 10;
       *          temp2  = *B << 1;
       *          temp1  = AMR_SUB( temp1, temp2 );
       *
       *          assert(*INVA != MIN_WORD);
       *
       *          temp1  = AMR_MULT_R( *INVA, temp1 );
       *          *LARpp = AMR_ADD( temp1, temp1 );
       *    }
       */

#undef      STEP
#define     STEP( B, MIC, INVA )    \
            temp1    = (word) AMR_ADD( *LARc++, MIC ) << 10;      \
            temp1    = (word) AMR_SUB( temp1, B << 1 );           \
            temp1    = (word) AMR_MULT_R( INVA, temp1 );          \
            *LARpp++ = (word) AMR_ADD( temp1, temp1 );

      STEP(      0,  -32,  13107 );
      STEP(      0,  -32,  13107 );
      STEP(   2048,  -16,  13107 );
      STEP(  -2560,  -16,  13107 );

      STEP(     94,   -8,  19223 );
      STEP(  -1792,   -8,  17476 );
      STEP(   -341,   -4,  31454 );
      STEP(  -1144,   -4,  29708 );

      /* NOTE: the addition of *MIC is used to restore
       *     the sign of *LARc.
       */
}

/* 4.2.9 */
/* Computation of the quantized reflection coefficients
 */

/* 4.2.9.1  Interpolation of the LARpp[1..8] to get the LARp[1..8]
 */

/*
 *  Within each frame of 160 analyzed speech samples the short term
 *  analysis and synthesis filters operate with four different sets of
 *  coefficients, derived from the previous set of decoded
LARs(LARpp(j-1))
```

```
 *  and the actual set of decoded LARs (LARpp(j))
 *
 * (Initial value: LARpp(j-1)[1..8] = 0.)
 */

static void Coefficients_0_12(
      register word * LARpp_j_1,
      register word * LARpp_j,
      register word * LARp)
{
      register int      i;

      for (i = 1; i <= 8; i++, LARp++, LARpp_j_1++, LARpp_j++) {
            *LARp = (word) AMR_ADD( (word)SASR( *LARpp_j_1, 2 ),
(word)SASR( *LARpp_j, 2 ));
            *LARp = (word) AMR_ADD( *LARp,  (word)SASR( *LARpp_j_1,
1));
      }
}

static void Coefficients_13_26(
      register word * LARpp_j_1,
      register word * LARpp_j,
      register word * LARp)
{
      register int i;
      for (i = 1; i <= 8; i++, LARpp_j_1++, LARpp_j++, LARp++) {
            *LARp = (word) AMR_ADD( (word)SASR( *LARpp_j_1, 1),
(word)SASR( *LARpp_j, 1 ));
      }
}

static void Coefficients_27_39(
      register word * LARpp_j_1,
      register word * LARpp_j,
      register word * LARp)
{
      register int i;

      for (i = 1; i <= 8; i++, LARpp_j_1++, LARpp_j++, LARp++) {
            *LARp = (word) AMR_ADD( (word)SASR( *LARpp_j_1, 2 ),
(word)SASR( *LARpp_j, 2 ));
            *LARp = (word) AMR_ADD( *LARp, (word)SASR( *LARpp_j, 1 ));
      }
}


static void Coefficients_40_159(
      register word * LARpp_j,
      register word * LARp)
{
      register int i;

      for (i = 1; i <= 8; i++, LARp++, LARpp_j++)
            *LARp = *LARpp_j;
}

/* 4.2.9.2 */

static void LARp_to_rp(
      register word * LARp)   /* [0..7] IN/OUT  */
```

```
/*
 *  The input of this procedure is the interpolated LARp[0..7] array.
 *  The reflection coefficients, rp[i], are used in the analysis
 *  filter and in the synthesis filter.
 */
{
      register int            i;
      register word           temp;

      for (i = 1; i <= 8; i++, LARp++) {

            /* temp = AMR_ABS( *LARp );
               *
             * if (temp < 11059) temp <<= 1;
             * else if (temp < 20070) temp += 11059;
             * else temp = AMR_ADD( temp >> 2, 26112 );
             *
             * *LARp = *LARp < 0 ? -temp : temp;
             */

            if (*LARp < 0) {
                  temp = *LARp == MIN_WORD ? MAX_WORD : -(*LARp);
                  *LARp = - ((temp < 11059) ? temp << 1
                        : ((temp < 20070) ? temp + 11059
                        :  (word) AMR_ADD( (word)(temp >> 2), 26112
)));
            } else {
                  temp  = *LARp;
                  *LARp =    (temp < 11059) ? temp << 1
                        : ((temp < 20070) ? temp + 11059
                        :  (word) AMR_ADD( (word)(temp >> 2), 26112 ));
            }
      }
}


/* 4.2.10 */
static void Short_term_analysis_filtering(
      struct amr_state * S,
      register word     * rp, /* [0..7]   IN    */
      register int      k_n,  /*   k_end - k_start    */
      register word     * s   /* [0..n-1] IN/OUT      */
)
/*
 *  This procedure computes the short term residual signal d[..] to be
fed
 *  to the RPE-LTP loop from the s[..] signal and from the local
rp[..]
 *  array (quantized reflection coefficients).  As the call of this
 *  procedure can be done in many ways (see the interpolation of the
LAR
 *  coefficient), it is assumed that the computation begins with index
 *  k_start (for arrays d[..] and s[..]) and stops with index k_end
 *  (k_start and k_end are defined in 4.2.9.1).  This procedure also
 *  needs to keep the array u[0..7] in memory for each call.
 */
{
      register word           * u = S->u;
      register int            i;
      register word           di, zzz, ui, sav, rpi;
```

```
      for (; k_n--; s++) {

            di = sav = *s;

            for (i = 0; i < 8; i++) {           /* YYY */

                  ui    = u[i];
                  rpi   = rp[i];
                  u[i]  = sav;

                  zzz   = AMR_MULT_R(rpi, di);
                  sav   = (word) AMR_ADD(   ui,  zzz);

                  zzz   = AMR_MULT_R(rpi, ui);
                  di    = (word) AMR_ADD(   di,  zzz );
            }

            *s = di;
      }
}

#if defined(USE_FLOAT_MUL) && defined(FAST)

static void Fast_Short_term_analysis_filtering(
      struct amr_state * S,
      register word     * rp, /* [0..7]   IN    */
      register int      k_n,  /*   k_end - k_start    */
      register word     * s   /* [0..n-1] IN/OUT      */
)
{
      register word           * u = S->u;
      register int            i;

      float         uf[8],
             rpf[8];

      register float scalef = 3.0517578125e-5;
      register float          sav, di, temp;

      for (i = 0; i < 8; ++i) {
            uf[i]  = u[i];
            rpf[i] = rp[i] * scalef;
      }
      for (; k_n--; s++) {
            sav = di = *s;
            for (i = 0; i < 8; ++i) {
                  register float rpfi = rpf[i];
                  register float ufi  = uf[i];

                  uf[i] = sav;
                  temp  = rpfi * di + ufi;
                  di   += rpfi * ufi;
                  sav   = temp;
            }
            *s = di;
      }
      for (i = 0; i < 8; ++i) u[i] = uf[i];
}
#endif /* ! (defined (USE_FLOAT_MUL) && defined (FAST)) */

static void Short_term_synthesis_filtering(
```

```
      struct amr_state * S,
      register word     * rrp,      /* [0..7]   IN    */
      register int      k,    /* k_end - k_start      */
      register word     * wt, /* [0..k-1] IN    */
      register word     * sr  /* [0..k-1] OUT   */
)
{
      register word           * v = S->v;
      register int            i;
      register word           sri, tmp1, tmp2;

      while (k--) {
            sri = *wt++;
            for (i = 8; i--;) {

                  /* sri = AMR_SUB( sri, amr_mult_r( rrp[i], v[i] ) );
                   */
                  tmp1 = rrp[i];
                  tmp2 = v[i];
                  tmp2 =  ( tmp1 == MIN_WORD && tmp2 == MIN_WORD
                        ? MAX_WORD
                        : 0x0FFFF & (( (longword)tmp1 * (longword)tmp2
                                    + 16384) >> 15)) ;

                  sri  = (word) AMR_SUB( sri, tmp2 );

                  /* v[i+1] = AMR_ADD( v[i], amr_mult_r( rrp[i], sri )
);
                   */
                  tmp1  = ( tmp1 == MIN_WORD && sri == MIN_WORD
                        ? MAX_WORD
                        : 0x0FFFF & (( (longword)tmp1 * (longword)sri
                                    + 16384) >> 15)) ;

                  v[i+1] = (word) AMR_ADD( v[i], tmp1);
            }
            *sr++ = v[0] = sri;
      }
}


#if defined(FAST) && defined(USE_FLOAT_MUL)

static void Fast_Short_term_synthesis_filtering(
      struct amr_state * S,
      register word     * rrp,      /* [0..7]   IN    */
      register int      k,    /* k_end - k_start      */
      register word     * wt, /* [0..k-1] IN    */
      register word     * sr  /* [0..k-1] OUT   */
)
{
      register word           * v = S->v;
      register int            i;

      float va[9], rrpa[8];
      register float scalef = 3.0517578125e-5, temp;

      for (i = 0; i < 8; ++i) {
            va[i]   = v[i];
            rrpa[i] = (float)rrp[i] * scalef;
      }
```

```
      while (k--) {
            register float sri = *wt++;
            for (i = 8; i--;) {
                  sri -= rrpa[i] * va[i];
                  if     (sri < -32768.) sri = -32768.;
                  else if (sri > 32767.) sri =  32767.;

                  temp = va[i] + rrpa[i] * sri;
                  if     (temp < -32768.) temp = -32768.;
                  else if (temp > 32767.) temp =  32767.;
                  va[i+1] = temp;
            }
            *sr++ = va[0] = sri;
      }
      for (i = 0; i < 9; ++i) v[i] = va[i];
}

#endif /* defined(FAST) && defined(USE_FLOAT_MUL) */

void Amr_Short_Term_Analysis_Filter(
      struct amr_state * S,
      word  * LARc,           /* coded log area ratio [0..7]  IN  */
      word  * s         /* signal [0..159]            IN/OUT      */
)
{
      word        * LARpp_j   = S->LARpp[ S->j      ];
      word        * LARpp_j_1 = S->LARpp[ S->j ^= 1 ];

      word        LARp[8];

#undef      FILTER
#if   defined(FAST) && defined(USE_FLOAT_MUL)
#     define      FILTER      (* (S->fast             \
                     ? Fast_Short_term_analysis_filtering  \
                     : Short_term_analysis_filtering ))

#else
#     define      FILTER      Short_term_analysis_filtering
#endif

      Decoding_of_the_coded_Log_Area_Ratios( LARc, LARpp_j );

      Coefficients_0_12(  LARpp_j_1, LARpp_j, LARp );
      LARp_to_rp( LARp );
      FILTER( S, LARp, 13, s);

      Coefficients_13_26( LARpp_j_1, LARpp_j, LARp);
      LARp_to_rp( LARp );
      FILTER( S, LARp, 14, s + 13);

      Coefficients_27_39( LARpp_j_1, LARpp_j, LARp);
      LARp_to_rp( LARp );
      FILTER( S, LARp, 13, s + 27);

      Coefficients_40_159( LARpp_j, LARp);
      LARp_to_rp( LARp );
      FILTER( S, LARp, 120, s + 40);
}

void Amr_Short_Term_Synthesis_Filter(
      struct amr_state * S,
```

```
      word  * LARcr,    /* received log area ratios [0..7] IN  */
      word  * wt,       /* received d [0..159]           IN  */
      word  * s         /* signal   s [0..159]          OUT  */
)
{
      word        * LARpp_j   = S->LARpp[ S->j     ];
      word        * LARpp_j_1 = S->LARpp[ S->j ^=1 ];

      word        LARp[8];

#undef      FILTER
#if   defined(FAST) && defined(USE_FLOAT_MUL)

#     define      FILTER      (* (S->fast             \
                     ? Fast_Short_term_synthesis_filtering \
                     : Short_term_synthesis_filtering))
#else
#     define      FILTER      Short_term_synthesis_filtering
#endif

      Decoding_of_the_coded_Log_Area_Ratios( LARcr, LARpp_j );

      Coefficients_0_12( LARpp_j_1, LARpp_j, LARp );
      LARp_to_rp( LARp );
      FILTER( S, LARp, 13, wt, s );

      Coefficients_13_26( LARpp_j_1, LARpp_j, LARp);
      LARp_to_rp( LARp );
      FILTER( S, LARp, 14, wt + 13, s + 13 );

      Coefficients_27_39( LARpp_j_1, LARpp_j, LARp);
      LARp_to_rp( LARp );
      FILTER( S, LARp, 13, wt + 27, s + 27 );

      Coefficients_40_159( LARpp_j, LARp );
      LARp_to_rp( LARp );
      FILTER(S, LARp, 120, wt + 40, s + 40);
}


/*
 *  4.2.11 .. 4.2.12 LONG TERM PREDICTOR (LTP) SECTION
 */


/*
 * This module computes the LTP gain (bc) and the LTP lag (Nc)
 * for the long term analysis filter.   This is done by calculating a
 * maximum of the cross-correlation function between the current
 * sub-segment short term residual signal d[0..39] (output of
 * the short term analysis filter; for simplification the index
 * of this array begins at 0 and ends at 39 for each sub-segment of
the
 * RPE-LTP analysis) and the previous reconstructed short term
 * residual signal dp[ -120 .. -1 ].  A dynamic scaling must be
 * performed to avoid overflow.
 */

 /* The next procedure exists in six versions.  First two integer
  * version (if USE_FLOAT_MUL is not defined); then four floating
  * point versions, twice with proper scaling (USE_FLOAT_MUL defined),
```

```
  * once without (USE_FLOAT_MUL and FAST defined, and fast run-time
  * option used).  Every pair has first a Cut version (see the -C
  * option to toast or the LTP_CUT option to amr_option()), then the
  * uncut one.  (For a detailed explanation of why this is altogether
  * a bad idea, see Henry Spencer and Geoff Collyer, ``#ifdef
Considered
  * Harmful''.)
  */

#ifndef  USE_FLOAT_MUL

#ifdef      LTP_CUT

static void Cut_Calculation_of_the_LTP_parameters(
      struct amr_state *st,
      register word      *d,        /* [0..39]  IN    */
      register word      *dp,       /* [-120..-1]     IN    */
      word         *bc_out,   /*          OUT   */
      word         *Nc_out    /*          OUT   */
)
{
      register int      k, lambda;
      word        Nc, bc;
      word        wt[40];

      longword    L_result;
      longword    L_max, L_power;
      word        R, S, dmax, scal, best_k;
      word        ltp_cut;

      register word     temp, wt_k;

      /*  Search of the optimum scaling of d[0..39].
       */
      dmax = 0;
      for (k = 0; k <= 39; k++) {
            temp = d[k];
            temp = AMR_ABS( temp );
            if (temp > dmax) {
                  dmax = temp;
                  best_k = k;
            }
      }
      temp = 0;
      if (dmax == 0) scal = 0;
      else {
            assert(dmax > 0);
            temp = amr_norm( (longword)dmax << 16 );
      }
      if (temp > 6) scal = 0;
      else scal = 6 - temp;
      assert(scal >= 0);

      /* Search for the maximum cross-correlation and coding of the
LTP lag
       */
      L_max = 0;
      Nc    = 40; /* index for the maximum cross-correlation */
      wt_k  = SASR(d[best_k], scal);

      for (lambda = 40; lambda <= 120; lambda++) {
```

```
            L_result = (longword)wt_k * dp[best_k - lambda];
            if (L_result > L_max) {
                  Nc    = lambda;
                  L_max = L_result;
            }
      }
      *Nc_out = Nc;
      L_max <<= 1;

      /*  Rescaling of L_max
       */
      assert(scal <= 100 && scal >= -100);
      L_max = L_max >> (6 - scal);  /* sub(6, scal) */

      assert( Nc <= 120 && Nc >= 40);

      /*   Compute the power of the reconstructed short term residual
       *   signal dp[..]
       */
      L_power = 0;
      for (k = 0; k <= 39; k++) {

            register longword L_temp;

            L_temp   = SASR( dp[k - Nc], 3 );
            L_power += L_temp * L_temp;
      }
      L_power <<= 1;    /* from L_MULT */

      /*  Normalization of L_max and L_power
       */

      if (L_max <= 0)  {
            *bc_out = 0;
            return;
      }
      if (L_max >= L_power) {
            *bc_out = 3;
            return;
      }

      temp = amr_norm( L_power );

      R = SASR( L_max   << temp, 16 );
      S = SASR( L_power << temp, 16 );

      /*  Coding of the LTP gain
       */

      /*  Table 4.3a must be used to obtain the level DLB[i] for the
       *  quantization of the LTP gain b to get the coded version bc.
       */
      for (bc = 0; bc <= 2; bc++) if (R <= amr_mult(S, amr_DLB[bc]))
break;
      *bc_out = bc;
}

#endif      /* LTP_CUT */

static void Calculation_of_the_LTP_parameters(
      register word     * d,        /* [0..39]  IN    */
```

```
      register word     * dp,       /* [-120..-1]     IN    */
      word        * bc_out,   /*          OUT   */
      word        * Nc_out    /*          OUT   */
)
{
      register int      k, lambda;
      word        Nc, bc;
      word        wt[40];

      longword    L_max, L_power;
      word        R, S, dmax, scal;
      register word     temp;

      /*  Search of the optimum scaling of d[0..39].
       */
      dmax = 0;

      for (k = 0; k <= 39; k++) {
            temp = d[k];
            temp = AMR_ABS( temp );
            if (temp > dmax) dmax = temp;
      }

      temp = 0;
      if (dmax == 0) scal = 0;
      else {
            assert(dmax > 0);
            temp = amr_norm( (longword)dmax << 16 );
      }

      if (temp > 6) scal = 0;
      else scal = 6 - temp;

      assert(scal >= 0);

      /*  Initialization of a working array wt
       */

      for (k = 0; k <= 39; k++) wt[k] = SASR( d[k], scal );

      /* Search for the maximum cross-correlation and coding of the
LTP lag
       */
      L_max = 0;
      Nc    = 40; /* index for the maximum cross-correlation */

      for (lambda = 40; lambda <= 120; lambda++) {

# undef STEP
#           define STEP(k)    (longword)wt[k] * dp[k - lambda]

            register longword L_result;

            L_result  = STEP(0)  ; L_result += STEP(1) ;
            L_result += STEP(2)  ; L_result += STEP(3) ;
            L_result += STEP(4)  ; L_result += STEP(5)  ;
            L_result += STEP(6)  ; L_result += STEP(7)  ;
            L_result += STEP(8)  ; L_result += STEP(9)  ;
            L_result += STEP(10) ; L_result += STEP(11) ;
            L_result += STEP(12) ; L_result += STEP(13) ;
            L_result += STEP(14) ; L_result += STEP(15) ;
```

```
        L_result += STEP(16) ; L_result += STEP(17) ;
        L_result += STEP(18) ; L_result += STEP(19) ;
        L_result += STEP(20) ; L_result += STEP(21) ;
        L_result += STEP(22) ; L_result += STEP(23) ;
        L_result += STEP(24) ; L_result += STEP(25) ;
        L_result += STEP(26) ; L_result += STEP(27) ;
        L_result += STEP(28) ; L_result += STEP(29) ;
        L_result += STEP(30) ; L_result += STEP(31) ;
        L_result += STEP(32) ; L_result += STEP(33) ;
        L_result += STEP(34) ; L_result += STEP(35) ;
        L_result += STEP(36) ; L_result += STEP(37) ;
        L_result += STEP(38) ; L_result += STEP(39) ;

        if (L_result > L_max) {

              Nc    = lambda;
              L_max = L_result;
        }
}

*Nc_out = Nc;

L_max <<= 1;

/*  Rescaling of L_max
 */
assert(scal <= 100 && scal >=  -100);
L_max = L_max >> (6 - scal);  /* sub(6, scal) */

assert( Nc <= 120 && Nc >= 40);

/*   Compute the power of the reconstructed short term residual
 *   signal dp[..]
 */
L_power = 0;
for (k = 0; k <= 39; k++) {

        register longword L_temp;

        L_temp   = SASR( dp[k - Nc], 3 );
        L_power += L_temp * L_temp;
}
L_power <<= 1;    /* from L_MULT */

/*  Normalization of L_max and L_power
 */

if (L_max <= 0)  {
        *bc_out = 0;
        return;
}
if (L_max >= L_power) {
        *bc_out = 3;
        return;
}

temp = amr_norm( L_power );

R = (word) SASR( L_max   << temp, 16 );
S = (word) SASR( L_power << temp, 16 );
```

```
      /*  Coding of the LTP gain
       */

      /*  Table 4.3a must be used to obtain the level DLB[i] for the
       *  quantization of the LTP gain b to get the coded version bc.
       */
      for (bc = 0; bc <= 2; bc++) if (R <= amr_mult(S, amr_DLB[bc]))
break;
      *bc_out = bc;
}

#else /* USE_FLOAT_MUL */

#ifdef      LTP_CUT

static void Cut_Calculation_of_the_LTP_parameters(
      struct amr_state * st,        /*              IN      */
      register word     * d,        /* [0..39]  IN    */
      register word     * dp,       /* [-120..-1]     IN    */
      word        * bc_out,   /*          OUT   */
      word        * Nc_out    /*          OUT   */
)
{
      register int      k, lambda;
      word        Nc, bc;
      word        ltp_cut;

      float       wt_float[40];
      float       dp_float_base[120], * dp_float = dp_float_base + 120;

      longword    L_max, L_power;
      word        R, S, dmax, scal;
      register word     temp;

      /*  Search of the optimum scaling of d[0..39].
       */
      dmax = 0;

      for (k = 0; k <= 39; k++) {
            temp = d[k];
            temp = AMR_ABS( temp );
            if (temp > dmax) dmax = temp;
      }

      temp = 0;
      if (dmax == 0) scal = 0;
      else {
            assert(dmax > 0);
            temp = amr_norm( (longword)dmax << 16 );
      }

      if (temp > 6) scal = 0;
      else scal = 6 - temp;

      assert(scal >= 0);
      ltp_cut = (longword)SASR(dmax, scal) * st->ltp_cut / 100;


      /*  Initialization of a working array wt
       */
```

```
      for (k = 0; k < 40; k++) {
            register word w = SASR( d[k], scal );
            if (w < 0 ? w > -ltp_cut : w < ltp_cut) {
                  wt_float[k] = 0.0;
            }
            else {
                  wt_float[k] =  w;
            }
      }
      for (k = -120; k <  0; k++) dp_float[k] =  dp[k];

      /* Search for the maximum cross-correlation and coding of the
LTP lag
       */
      L_max = 0;
      Nc    = 40; /* index for the maximum cross-correlation */

      for (lambda = 40; lambda <= 120; lambda += 9) {

            /*  Calculate L_result for l = lambda .. lambda + 9.
             */
            register float *lp = dp_float - lambda;

            register float    W;
            register float    a = lp[-8], b = lp[-7], c = lp[-6],
                        d = lp[-5], e = lp[-4], f = lp[-3],
                        g = lp[-2], h = lp[-1];
            register float  E;
            register float  S0 = 0, S1 = 0, S2 = 0, S3 = 0, S4 = 0,
                        S5 = 0, S6 = 0, S7 = 0, S8 = 0;

#           undef STEP
#           define      STEP(K, a, b, c, d, e, f, g, h) \
                  if ((W = wt_float[K]) != 0.0) {     \
                  E = W * a; S8 += E;           \
                  E = W * b; S7 += E;           \
                  E = W * c; S6 += E;           \
                  E = W * d; S5 += E;           \
                  E = W * e; S4 += E;           \
                  E = W * f; S3 += E;           \
                  E = W * g; S2 += E;           \
                  E = W * h; S1 += E;           \
                  a  = lp[K];             \
                  E = W * a; S0 += E; } else (a = lp[K])

#           define      STEP_A(K)   STEP(K, a, b, c, d, e, f, g, h)
#           define      STEP_B(K)   STEP(K, b, c, d, e, f, g, h, a)
#           define      STEP_C(K)   STEP(K, c, d, e, f, g, h, a, b)
#           define      STEP_D(K)   STEP(K, d, e, f, g, h, a, b, c)
#           define      STEP_E(K)   STEP(K, e, f, g, h, a, b, c, d)
#           define      STEP_F(K)   STEP(K, f, g, h, a, b, c, d, e)
#           define      STEP_G(K)   STEP(K, g, h, a, b, c, d, e, f)
#           define      STEP_H(K)   STEP(K, h, a, b, c, d, e, f, g)

            STEP_A( 0); STEP_B( 1); STEP_C( 2); STEP_D( 3);
            STEP_E( 4); STEP_F( 5); STEP_G( 6); STEP_H( 7);

            STEP_A( 8); STEP_B( 9); STEP_C(10); STEP_D(11);
            STEP_E(12); STEP_F(13); STEP_G(14); STEP_H(15);

            STEP_A(16); STEP_B(17); STEP_C(18); STEP_D(19);
```

```
        STEP_E(20); STEP_F(21); STEP_G(22); STEP_H(23);

        STEP_A(24); STEP_B(25); STEP_C(26); STEP_D(27);
        STEP_E(28); STEP_F(29); STEP_G(30); STEP_H(31);

        STEP_A(32); STEP_B(33); STEP_C(34); STEP_D(35);
        STEP_E(36); STEP_F(37); STEP_G(38); STEP_H(39);

        if (S0 > L_max) { L_max = S0; Nc = lambda;     }
        if (S1 > L_max) { L_max = S1; Nc = lambda + 1; }
        if (S2 > L_max) { L_max = S2; Nc = lambda + 2; }
        if (S3 > L_max) { L_max = S3; Nc = lambda + 3; }
        if (S4 > L_max) { L_max = S4; Nc = lambda + 4; }
        if (S5 > L_max) { L_max = S5; Nc = lambda + 5; }
        if (S6 > L_max) { L_max = S6; Nc = lambda + 6; }
        if (S7 > L_max) { L_max = S7; Nc = lambda + 7; }
        if (S8 > L_max) { L_max = S8; Nc = lambda + 8; }

    }
    *Nc_out = Nc;

    L_max <<= 1;

    /*  Rescaling of L_max
     */
    assert(scal <= 100 && scal >=  -100);
    L_max = L_max >> (6 - scal);  /* sub(6, scal) */

    assert( Nc <= 120 && Nc >= 40);

    /*   Compute the power of the reconstructed short term residual
     *   signal dp[..]
     */
    L_power = 0;
    for (k = 0; k <= 39; k++) {

        register longword L_temp;

        L_temp   = SASR( dp[k - Nc], 3 );
        L_power += L_temp * L_temp;
    }
    L_power <<= 1;    /* from L_MULT */

    /*  Normalization of L_max and L_power
     */

    if (L_max <= 0)  {
        *bc_out = 0;
        return;
    }
    if (L_max >= L_power) {
        *bc_out = 3;
        return;
    }

    temp = amr_norm( L_power );

    R = SASR( L_max   << temp, 16 );
    S = SASR( L_power << temp, 16 );

    /*  Coding of the LTP gain
```

```
       */

      /*  Table 4.3a must be used to obtain the level DLB[i] for the
       *  quantization of the LTP gain b to get the coded version bc.
       */
      for (bc = 0; bc <= 2; bc++) if (R <= amr_mult(S, amr_DLB[bc]))
break;
      *bc_out = bc;
}

#endif /* LTP_CUT */

static void Calculation_of_the_LTP_parameters(
      register word     * d,        /* [0..39]  IN    */
      register word     * dp,       /* [-120..-1]     IN    */
      word        * bc_out,   /*          OUT   */
      word        * Nc_out    /*          OUT   */
)
{
      register int      k, lambda;
      word        Nc, bc;

      float       wt_float[40];
      float       dp_float_base[120], * dp_float = dp_float_base + 120;

      longword    L_max, L_power;
      word        R, S, dmax, scal;
      register word     temp;

      /*  Search of the optimum scaling of d[0..39].
       */
      dmax = 0;

      for (k = 0; k <= 39; k++) {
            temp = d[k];
            temp = AMR_ABS( temp );
            if (temp > dmax) dmax = temp;
      }

      temp = 0;
      if (dmax == 0) scal = 0;
      else {
            assert(dmax > 0);
            temp = amr_norm( (longword)dmax << 16 );
      }

      if (temp > 6) scal = 0;
      else scal = 6 - temp;

      assert(scal >= 0);

      /*  Initialization of a working array wt
       */

      for (k =    0; k < 40; k++) wt_float[k] =  SASR( d[k], scal );
      for (k = -120; k <  0; k++) dp_float[k] =  dp[k];

      /* Search for the maximum cross-correlation and coding of the
LTP lag
       */
      L_max = 0;
```

```
      Nc    = 40; /* index for the maximum cross-correlation */

      for (lambda = 40; lambda <= 120; lambda += 9) {

            /*  Calculate L_result for l = lambda .. lambda + 9.
             */
            register float *lp = dp_float - lambda;

            register float    W;
            register float    a = lp[-8], b = lp[-7], c = lp[-6],
                        d = lp[-5], e = lp[-4], f = lp[-3],
                        g = lp[-2], h = lp[-1];
            register float  E;
            register float  S0 = 0, S1 = 0, S2 = 0, S3 = 0, S4 = 0,
                        S5 = 0, S6 = 0, S7 = 0, S8 = 0;

#           undef STEP
#           define      STEP(K, a, b, c, d, e, f, g, h) \
                  W = wt_float[K];        \
                  E = W * a; S8 += E;           \
                  E = W * b; S7 += E;           \
                  E = W * c; S6 += E;           \
                  E = W * d; S5 += E;           \
                  E = W * e; S4 += E;           \
                  E = W * f; S3 += E;           \
                  E = W * g; S2 += E;           \
                  E = W * h; S1 += E;           \
                  a  = lp[K];             \
                  E = W * a; S0 += E

#           define      STEP_A(K)   STEP(K, a, b, c, d, e, f, g, h)
#           define      STEP_B(K)   STEP(K, b, c, d, e, f, g, h, a)
#           define      STEP_C(K)   STEP(K, c, d, e, f, g, h, a, b)
#           define      STEP_D(K)   STEP(K, d, e, f, g, h, a, b, c)
#           define      STEP_E(K)   STEP(K, e, f, g, h, a, b, c, d)
#           define      STEP_F(K)   STEP(K, f, g, h, a, b, c, d, e)
#           define      STEP_G(K)   STEP(K, g, h, a, b, c, d, e, f)
#           define      STEP_H(K)   STEP(K, h, a, b, c, d, e, f, g)

            STEP_A( 0); STEP_B( 1); STEP_C( 2); STEP_D( 3);
            STEP_E( 4); STEP_F( 5); STEP_G( 6); STEP_H( 7);

            STEP_A( 8); STEP_B( 9); STEP_C(10); STEP_D(11);
            STEP_E(12); STEP_F(13); STEP_G(14); STEP_H(15);

            STEP_A(16); STEP_B(17); STEP_C(18); STEP_D(19);
            STEP_E(20); STEP_F(21); STEP_G(22); STEP_H(23);

            STEP_A(24); STEP_B(25); STEP_C(26); STEP_D(27);
            STEP_E(28); STEP_F(29); STEP_G(30); STEP_H(31);

            STEP_A(32); STEP_B(33); STEP_C(34); STEP_D(35);
            STEP_E(36); STEP_F(37); STEP_G(38); STEP_H(39);

            if (S0 > L_max) { L_max = (longword) S0; Nc = lambda;     }
            if (S1 > L_max) { L_max = (longword) S1; Nc = lambda + 1; }
            if (S2 > L_max) { L_max = (longword) S2; Nc = lambda + 2; }
            if (S3 > L_max) { L_max = (longword) S3; Nc = lambda + 3; }
            if (S4 > L_max) { L_max = (longword) S4; Nc = lambda + 4; }
            if (S5 > L_max) { L_max = (longword) S5; Nc = lambda + 5; }
            if (S6 > L_max) { L_max = (longword) S6; Nc = lambda + 6; }
```

```
            if (S7 > L_max) { L_max = (longword) S7; Nc = lambda + 7; }
            if (S8 > L_max) { L_max = (longword) S8; Nc = lambda + 8; }
      }
      *Nc_out = Nc;

      L_max <<= 1;

      /*  Rescaling of L_max
       */
      assert(scal <= 100 && scal >=  -100);
      L_max = L_max >> (6 - scal);  /* sub(6, scal) */

      assert( Nc <= 120 && Nc >= 40);

      /*   Compute the power of the reconstructed short term residual
       *   signal dp[..]
       */
      L_power = 0;
      for (k = 0; k <= 39; k++) {

            register longword L_temp;

            L_temp   = SASR( dp[k - Nc], 3 );
            L_power += L_temp * L_temp;
      }
      L_power <<= 1;    /* from L_MULT */

      /*  Normalization of L_max and L_power
       */

      if (L_max <= 0)  {
            *bc_out = 0;
            return;
      }
      if (L_max >= L_power) {
            *bc_out = 3;
            return;
      }

      temp = amr_norm( L_power );

      R = SASR( L_max   << temp, 16 );
      S = SASR( L_power << temp, 16 );

      /*  Coding of the LTP gain
       */

      /*  Table 4.3a must be used to obtain the level DLB[i] for the
       *  quantization of the LTP gain b to get the coded version bc.
       */
      for (bc = 0; bc <= 2; bc++) if (R <= amr_mult(S, amr_DLB[bc]))
break;
      *bc_out = bc;
}

#ifdef      FAST
#ifdef      LTP_CUT

static void Cut_Fast_Calculation_of_the_LTP_parameters(
      struct amr_state * st,        /*              IN      */
      register word     * d,        /* [0..39]  IN    */
```

```
      register word     * dp,       /* [-120..-1]     IN    */
      word        * bc_out,   /*          OUT   */
      word        * Nc_out    /*          OUT   */
)
{
      register int      k, lambda;
      register float    wt_float;
      word        Nc, bc;
      word        wt_max, best_k, ltp_cut;

      float       dp_float_base[120], * dp_float = dp_float_base + 120;

      register float    L_result, L_max, L_power;

      wt_max = 0;

      for (k = 0; k < 40; ++k) {
            if      ( d[k] > wt_max) wt_max =  d[best_k = k];
            else if (-d[k] > wt_max) wt_max = -d[best_k = k];
      }

      assert(wt_max >= 0);
      wt_float = (float)wt_max;

      for (k = -120; k <= 0; ++k) dp_float[k] = (float)dp[k];

      /* Search for the maximum cross-correlation and coding of the
LTP lag
       */
      L_max = 0;
      Nc    = 40; /* index for the maximum cross-correlation */

      for (lambda = 40; lambda <= 120; lambda++) {
            L_result = wt_float * dp_float[best_k - lambda];
            if (L_result > L_max) {
                  Nc    = lambda;
                  L_max = L_result;
            }
      }

      *Nc_out = Nc;
      if (L_max <= 0.)  {
            *bc_out = 0;
            return;
      }

      /*  Compute the power of the reconstructed short term residual
       *  signal dp[..]
       */
      dp_float -= Nc;
      L_power = 0;
      for (k = 0; k < 40; ++k) {
            register float f = dp_float[k];
            L_power += f * f;
      }

      if (L_max >= L_power) {
            *bc_out = 3;
            return;
      }
```

```
      /*  Coding of the LTP gain
       *  Table 4.3a must be used to obtain the level DLB[i] for the
       *  quantization of the LTP gain b to get the coded version bc.
       */
      lambda = L_max / L_power * 32768.;
      for (bc = 0; bc <= 2; ++bc) if (lambda <= amr_DLB[bc]) break;
      *bc_out = bc;
}

#endif /* LTP_CUT */

static void Fast_Calculation_of_the_LTP_parameters(
      register word     * d,        /* [0..39]  IN    */
      register word     * dp,       /* [-120..-1]     IN    */
      word        * bc_out,   /*          OUT   */
      word        * Nc_out    /*          OUT   */
)
{
      register int      k, lambda;
      word        Nc, bc;

      float       wt_float[40];
      float       dp_float_base[120], * dp_float = dp_float_base + 120;

      register float    L_max, L_power;

      for (k = 0; k < 40; ++k) wt_float[k] = (float)d[k];
      for (k = -120; k <= 0; ++k) dp_float[k] = (float)dp[k];

      /* Search for the maximum cross-correlation and coding of the
LTP lag
       */
      L_max = 0;
      Nc    = 40; /* index for the maximum cross-correlation */

      for (lambda = 40; lambda <= 120; lambda += 9) {

            /*  Calculate L_result for l = lambda .. lambda + 9.
             */
            register float *lp = dp_float - lambda;

            register float    W;
            register float    a = lp[-8], b = lp[-7], c = lp[-6],
                        d = lp[-5], e = lp[-4], f = lp[-3],
                        g = lp[-2], h = lp[-1];
            register float  E;
            register float  S0 = 0, S1 = 0, S2 = 0, S3 = 0, S4 = 0,
                        S5 = 0, S6 = 0, S7 = 0, S8 = 0;

#           undef STEP
#           define      STEP(K, a, b, c, d, e, f, g, h) \
                  W = wt_float[K];        \
                  E = W * a; S8 += E;           \
                  E = W * b; S7 += E;           \
                  E = W * c; S6 += E;           \
                  E = W * d; S5 += E;           \
                  E = W * e; S4 += E;           \
                  E = W * f; S3 += E;           \
                  E = W * g; S2 += E;           \
                  E = W * h; S1 += E;           \
                  a  = lp[K];             \
```

```
                E = W * a; S0 += E

#           define      STEP_A(K)   STEP(K, a, b, c, d, e, f, g, h)
#           define      STEP_B(K)   STEP(K, b, c, d, e, f, g, h, a)
#           define      STEP_C(K)   STEP(K, c, d, e, f, g, h, a, b)
#           define      STEP_D(K)   STEP(K, d, e, f, g, h, a, b, c)
#           define      STEP_E(K)   STEP(K, e, f, g, h, a, b, c, d)
#           define      STEP_F(K)   STEP(K, f, g, h, a, b, c, d, e)
#           define      STEP_G(K)   STEP(K, g, h, a, b, c, d, e, f)
#           define      STEP_H(K)   STEP(K, h, a, b, c, d, e, f, g)

            STEP_A( 0); STEP_B( 1); STEP_C( 2); STEP_D( 3);
            STEP_E( 4); STEP_F( 5); STEP_G( 6); STEP_H( 7);

            STEP_A( 8); STEP_B( 9); STEP_C(10); STEP_D(11);
            STEP_E(12); STEP_F(13); STEP_G(14); STEP_H(15);

            STEP_A(16); STEP_B(17); STEP_C(18); STEP_D(19);
            STEP_E(20); STEP_F(21); STEP_G(22); STEP_H(23);

            STEP_A(24); STEP_B(25); STEP_C(26); STEP_D(27);
            STEP_E(28); STEP_F(29); STEP_G(30); STEP_H(31);

            STEP_A(32); STEP_B(33); STEP_C(34); STEP_D(35);
            STEP_E(36); STEP_F(37); STEP_G(38); STEP_H(39);

            if (S0 > L_max) { L_max = S0; Nc = lambda;     }
            if (S1 > L_max) { L_max = S1; Nc = lambda + 1; }
            if (S2 > L_max) { L_max = S2; Nc = lambda + 2; }
            if (S3 > L_max) { L_max = S3; Nc = lambda + 3; }
            if (S4 > L_max) { L_max = S4; Nc = lambda + 4; }
            if (S5 > L_max) { L_max = S5; Nc = lambda + 5; }
            if (S6 > L_max) { L_max = S6; Nc = lambda + 6; }
            if (S7 > L_max) { L_max = S7; Nc = lambda + 7; }
            if (S8 > L_max) { L_max = S8; Nc = lambda + 8; }
      }
      *Nc_out = Nc;

      if (L_max <= 0.)  {
            *bc_out = 0;
            return;
      }

      /*  Compute the power of the reconstructed short term residual
       *  signal dp[..]
       */
      dp_float -= Nc;
      L_power = 0;
      for (k = 0; k < 40; ++k) {
            register float f = dp_float[k];
            L_power += f * f;
      }

      if (L_max >= L_power) {
            *bc_out = 3;
            return;
      }

      /*  Coding of the LTP gain
       *  Table 4.3a must be used to obtain the level DLB[i] for the
       *  quantization of the LTP gain b to get the coded version bc.
```

```
       */
      lambda = (int) (L_max / L_power * 32768.0);
      for (bc = 0; bc <= 2; ++bc) if (lambda <= amr_DLB[bc]) break;
      *bc_out = bc;
}

#endif      /* FAST      */
#endif      /* USE_FLOAT_MUL */


/* 4.2.12 */

static void Long_term_analysis_filtering(
      word        bc,   /*                            IN  */
      word        Nc,   /*                            IN  */
      register word     * dp, /* previous d     [-120..-1]        IN
*/
      register word     * d,  /* d        [0..39]                 IN
*/
      register word     * dpp,      /* estimate [0..39]
      OUT */
      register word     * e   /* long term res. signal [0..39]    OUT
*/
)
/*
 *  In this part, we have to decode the bc parameter to compute
 *  the samples of the estimate dpp[0..39].  The decoding of bc needs
the
 *  use of table 4.3b.  The long term residual signal e[0..39]
 *  is then calculated to be fed to the RPE encoding section.
 */
{
      register int      k;

#     undef STEP
#     define STEP(BP)                            \
      for (k = 0; k <= 39; k++) {                 \
            dpp[k]  = (word) AMR_MULT_R( BP, dp[k - Nc]);  \
            e[k]  = (word) AMR_SUB( d[k], dpp[k] );  \
      }

      switch (bc) {
      case 0:     STEP(  3277 ); break;
      case 1:     STEP( 11469 ); break;
      case 2: STEP( 21299 ); break;
      case 3: STEP( 32767 ); break;
      }
}

void Amr_Long_Term_Predictor(
      /* 4x for 160 samples */

      struct amr_state  * S,

      word  * d,  /* [0..39]   residual signal  IN    */
      word  * dp, /* [-120..-1] d'        IN    */

      word  * e,  /* [0..39]              OUT   */
      word  * dpp,      /* [0..39]              OUT   */
      word  * Nc, /* correlation lag            OUT   */
      word  * bc  /* gain factor                OUT   */
```

```
)
{
      assert( d  ); assert( dp ); assert( e  );
      assert( dpp); assert( Nc ); assert( bc );

#if defined(FAST) && defined(USE_FLOAT_MUL)
      if (S->fast)
#if   defined (LTP_CUT)
            if (S->ltp_cut)
                  Cut_Fast_Calculation_of_the_LTP_parameters(S,
                        d, dp, bc, Nc);
            else
#endif /* LTP_CUT */
                  Fast_Calculation_of_the_LTP_parameters(d, dp, bc, Nc
);
      else
#endif /* FAST & USE_FLOAT_MUL */
#ifdef LTP_CUT
            if (S->ltp_cut)
                  Cut_Calculation_of_the_LTP_parameters(S, d, dp, bc,
Nc);
            else
#endif
                  Calculation_of_the_LTP_parameters(d, dp, bc, Nc);

      Long_term_analysis_filtering( *bc, *Nc, dp, d, dpp, e );
        UNUSED(S);
}

/* 4.3.2 */
void Amr_Long_Term_Synthesis_Filtering(
      struct amr_state  * S,
      word              Ncr,
      word              bcr,
      register word           * erp,            /* [0..39]          IN
*/
      register word           * drp       /* [-120..-1] IN, [0..40] OUT
*/
)
/*
 *  This procedure uses the bcr and Ncr parameter to realize the
 *  long term synthesis filtering.  The decoding of bcr needs
 *  table 4.3b.
 */
{
      register int            k;
      word              brp, drpp, Nr;

      /*  Check the limits of Nr.
       */
      Nr = Ncr < 40 || Ncr > 120 ? S->nrp : Ncr;
      S->nrp = Nr;
      assert(Nr >= 40 && Nr <= 120);

      /*  Decoding of the LTP gain bcr
       */
      brp = amr_QLB[ bcr ];

      /*  Computation of the reconstructed short term residual
       *  signal drp[0..39]
       */
```

```
      assert(brp != MIN_WORD);

      for (k = 0; k <= 39; k++) {
            drpp   = AMR_MULT_R( brp, drp[ k - Nr ] );
            drp[k] = (word) AMR_ADD( erp[k], drpp );
      }

      /*
       *  Update of the reconstructed short term residual signal
       *  drp[ -1..-120 ]
       */

      for (k = 0; k <= 119; k++) drp[ -120 + k ] = drp[ -80 + k ];
}

/*  4.2.13 .. 4.2.17  RPE ENCODING SECTION
 */

/* 4.2.13 */

static void Weighting_filter(
      register word     * e,        /* signal [-5..0.39.44] IN  */
      word        * x         /* signal [0..39] OUT */
)
/*
 *  The coefficients of the weighting filter are stored in a table
 *  (see table 4.4).  The following scaling is used:
 *
 *    H[0..10] = integer( real_H[ 0..10] * 8192 );
 */
{
      /* word                wt[ 50 ]; */

      register longword L_result;
      register int            k /* , i */ ;

      /*  Initialization of a temporary working array wt[0...49]
       */

      /* for (k =  0; k <=  4; k++) wt[k] = 0;
       * for (k =  5; k <= 44; k++) wt[k] = *e++;
       * for (k = 45; k <= 49; k++) wt[k] = 0;
       *
       *  (e[-5..-1] and e[40..44] are allocated by the caller,
       *  are initially zero and are not written anywhere.)
       */
      e -= 5;

      /*  Compute the signal x[0..39]
       */
      for (k = 0; k <= 39; k++) {

            L_result = 8192 >> 1;

            /* for (i = 0; i <= 10; i++) {
             *    L_temp   = AMR_L_MULT( wt[k+i], amr_H[i] );
             *    L_result = AMR_L_ADD( L_result, L_temp );
             * }
             */

#undef      STEP
```

```
#define     STEP( i, H )      (e[ k + i ] * (longword)H)

            /*  Every one of these multiplications is done twice --
             *  but I don't see an elegant way to optimize this.
             *  Do you?
             */

#ifdef      STUPID_COMPILER
            L_result += STEP( 0,    -134 ) ;
            L_result += STEP( 1,    -374 )  ;
                     /* + STEP(    2,    0    )  */
            L_result += STEP( 3,    2054 ) ;
            L_result += STEP( 4,    5741 ) ;
            L_result += STEP( 5,    8192 ) ;
            L_result += STEP( 6,    5741 ) ;
            L_result += STEP( 7,    2054 ) ;
                   /* + STEP( 8,    0    )  */
            L_result += STEP( 9,    -374 ) ;
            L_result += STEP( 10,   -134 ) ;
#else
            L_result +=
              STEP(     0,    -134 )
            + STEP(     1,    -374 )
           /* + STEP(   2,    0    )  */
            + STEP(     3,    2054 )
            + STEP(     4,    5741 )
            + STEP(     5,    8192 )
            + STEP(     6,    5741 )
            + STEP(     7,    2054 )
           /* + STEP(   8,    0    )  */
            + STEP(     9,    -374 )
            + STEP(10,  -134 )
            ;
#endif

            /* L_result = AMR_L_ADD( L_result, L_result ); (*
scaling(x2) *)
             * L_result = AMR_L_ADD( L_result, L_result ); (*
scaling(x4) *)
             *
             * x[k] = SASR( L_result, 16 );
             */

            /* 2 adds vs. >>16 => 14, minus one shift to compensate for
             * those we lost when replacing L_MULT by '*'.
             */

            L_result = SASR( L_result, 13 );
            x[k] =  (word) (  L_result < MIN_WORD ? MIN_WORD
                         : (L_result > MAX_WORD ? MAX_WORD : L_result
));
      }
}

/* 4.2.14 */

static void RPE_grid_selection(
      word        * x,        /* [0..39]        IN  */
      word        * xM,       /* [0..12]        OUT */
      word        * Mc_out    /*                OUT */
)
```

```
/*
 *  The signal x[0..39] is used to select the RPE grid which is
 *  represented by Mc.
 */
{
      /* register word  temp1;      */
      register int            /* m, */  i;
      register longword L_result, L_temp;
      longword          EM;   /* xxx should be L_EM? */
      word              Mc;

      longword          L_common_0_3;

      EM = 0;
      Mc = 0;

      /* for (m = 0; m <= 3; m++) {
       *    L_result = 0;
       *
       *
       *    for (i = 0; i <= 12; i++) {
       *
       *          temp1    = SASR( x[m + 3*i], 2 );
       *
       *          assert(temp1 != MIN_WORD);
       *
       *          L_temp   = AMR_L_MULT( temp1, temp1 );
       *          L_result = AMR_L_ADD( L_temp, L_result );
       *    }
       *
       *    if (L_result > EM) {
       *          Mc = m;
       *          EM = L_result;
       *    }
       * }
       */

#undef      STEP
#define     STEP( m, i )            L_temp = SASR( x[m + 3 * i], 2 );
      \
                        L_result += L_temp * L_temp;

      /* common part of 0 and 3 */

      L_result = 0;
      STEP( 0, 1 ); STEP( 0, 2 ); STEP( 0, 3 ); STEP( 0, 4 );
      STEP( 0, 5 ); STEP( 0, 6 ); STEP( 0, 7 ); STEP( 0, 8 );
      STEP( 0, 9 ); STEP( 0, 10); STEP( 0, 11); STEP( 0, 12);
      L_common_0_3 = L_result;

      /* i = 0 */

      STEP( 0, 0 );
      L_result <<= 1;   /* implicit in L_MULT */
      EM = L_result;

      /* i = 1 */

      L_result = 0;
      STEP( 1, 0 );
      STEP( 1, 1 ); STEP( 1, 2 ); STEP( 1, 3 ); STEP( 1, 4 );
```

```
      STEP( 1, 5 ); STEP( 1, 6 ); STEP( 1, 7 ); STEP( 1, 8 );
      STEP( 1, 9 ); STEP( 1, 10); STEP( 1, 11); STEP( 1, 12);
      L_result <<= 1;
      if (L_result > EM) {
            Mc = 1;
            EM = L_result;
      }

      /* i = 2 */

      L_result = 0;
      STEP( 2, 0 );
      STEP( 2, 1 ); STEP( 2, 2 ); STEP( 2, 3 ); STEP( 2, 4 );
      STEP( 2, 5 ); STEP( 2, 6 ); STEP( 2, 7 ); STEP( 2, 8 );
      STEP( 2, 9 ); STEP( 2, 10); STEP( 2, 11); STEP( 2, 12);
      L_result <<= 1;
      if (L_result > EM) {
            Mc = 2;
            EM = L_result;
      }

      /* i = 3 */

      L_result = L_common_0_3;
      STEP( 3, 12 );
      L_result <<= 1;
      if (L_result > EM) {
            Mc = 3;
            EM = L_result;
      }

      /**/

      /*  Down-sampling by a factor 3 to get the selected xM[0..12]
       *  RPE sequence.
       */
      for (i = 0; i <= 12; i ++) xM[i] = x[Mc + 3*i];
      *Mc_out = Mc;
}

/* 4.12.15 */

static void APCM_quantization_xmaxc_to_exp_mant(
      word        xmaxc,            /* IN       */
      word        * exp_out,  /* OUT      */
      word        * mant_out )      /* OUT  */
{
      word  exp, mant;

      /* Compute exponent and mantissa of the decoded version of xmaxc
*/

      exp = 0;
      if (xmaxc > 15) exp = SASR(xmaxc, 3) - 1;
      mant = xmaxc - (exp << 3);

      if (mant == 0) {
            exp  = -4;
            mant = 7;
      }
      else {
```

```
            while (mant <= 7) {
                  mant = mant << 1 | 1;
                  exp--;
            }
            mant -= 8;
      }

      assert( exp  >= -4 && exp <= 6 );
      assert( mant >= 0 && mant <= 7 );

      *exp_out  = exp;
      *mant_out = mant;
}

static void APCM_quantization(
      word        * xM,       /* [0..12]        IN    */

      word        * xMc,            /* [0..12]        OUT   */
      word        * mant_out, /*                OUT   */
      word        * exp_out,  /*                OUT   */
      word        * xmaxc_out /*                OUT   */
)
{
      int   i, itest;

      word  xmax, xmaxc, temp, temp1, temp2;
      word  exp, mant;


      /*  Find the maximum absolute value xmax of xM[0..12].
       */

      xmax = 0;
      for (i = 0; i <= 12; i++) {
            temp = xM[i];
            temp = AMR_ABS(temp);
            if (temp > xmax) xmax = temp;
      }

      /*  Qantizing and coding of xmax to get xmaxc.
       */

      exp   = 0;
      temp  = SASR( xmax, 9 );
      itest = 0;

      for (i = 0; i <= 5; i++) {

            itest |= (temp <= 0);
            temp = SASR( temp, 1 );

            assert(exp <= 5);
            if (itest == 0) exp++;        /* exp = add (exp, 1) */
      }

      assert(exp <= 6 && exp >= 0);
      temp = exp + 5;

      assert(temp <= 11 && temp >= 0);
      xmaxc = amr_add((word) SASR(xmax, temp), (word) (exp << 3) );
```

```
      /*   Quantizing and coding of the xM[0..12] RPE sequence
       *   to get the xMc[0..12]
       */

      APCM_quantization_xmaxc_to_exp_mant( xmaxc, &exp, &mant );

      /*  This computation uses the fact that the decoded version of
xmaxc
       *  can be calculated by using the exponent and the mantissa
part of
       *  xmaxc (logarithmic table).
       *  So, this method avoids any division and uses only a scaling
       *  of the RPE samples by a function of the exponent.  A direct
       *  multiplication by the inverse of the mantissa (NRFAC[0..7]
       *  found in table 4.5) gives the 3 bit coded version xMc[0..12]
       *  of the RPE samples.
       */


      /* Direct computation of xMc[0..12] using table 4.5
       */

      assert( exp <= 4096 && exp >= -4096);
      assert( mant >= 0 && mant <= 7 );

      temp1 = 6 - exp;        /* normalization by the exponent */
      temp2 = amr_NRFAC[ mant ];    /* inverse mantissa           */

      for (i = 0; i <= 12; i++) {

            assert(temp1 >= 0 && temp1 < 16);

            temp = xM[i] << temp1;
            temp = (word) AMR_MULT( temp, temp2 );
            temp = SASR(temp, 12);
            xMc[i] = temp + 4;            /* see note below */
      }

      /*  NOTE: This equation is used to make all the xMc[i] positive.
       */

      *mant_out  = mant;
      *exp_out   = exp;
      *xmaxc_out = xmaxc;
}

/* 4.2.16 */

static void APCM_inverse_quantization(
      register word     * xMc,      /* [0..12]              IN    */
      word        mant,
      word        exp,
      register word     * xMp)      /* [0..12]              OUT   */
/*
 *  This part is for decoding the RPE sequence of coded xMc[0..12]
 *  samples to obtain the xMp[0..12] array.  Table 4.6 is used to get
 *  the mantissa of xmaxc (FAC[0..7]).
 */
{
      int   i;
      word  temp, temp1, temp2, temp3;
```

```
      assert( mant >= 0 && mant <= 7 );

      temp1 = amr_FAC[ mant ];      /* see 4.2-15 for mant */
      temp2 = amr_sub( 6, exp );    /* see 4.2-15 for exp  */
      temp3 = amr_asl( 1, amr_sub( temp2, 1 ));

      for (i = 13; i--;) {

            assert( *xMc <= 7 && *xMc >= 0 );   /* 3 bit unsigned */

            /* temp = amr_sub( *xMc++ << 1, 7 ); */
            temp = (*xMc++ << 1) - 7;             /* restore sign   */
            assert( temp <= 7 && temp >= -7 );  /* 4 bit signed   */

            temp <<= 12;                        /* 16 bit signed  */
            temp = AMR_MULT_R( temp1, temp );
            temp = (word) AMR_ADD( temp, temp3 );
            *xMp++ = amr_asr( temp, temp2 );
      }
}

/* 4.2.17 */

static void RPE_grid_positioning(
      word        Mc,         /* grid position  IN    */
      register word     * xMp,            /* [0..12]        IN    */
      register word     * ep        /* [0..39]        OUT   */
)
/*
 *  This procedure computes the reconstructed long term residual
signal
 *  ep[0..39] for the LTP analysis filter.  The inputs are the Mc
 *  which is the grid position selection and the xMp[0..12] decoded
 *  RPE samples which are upsampled by a factor of 3 by inserting zero
 *  values.
 */
{
      int   i = 13;

      assert(0 <= Mc && Mc <= 3);

        switch (Mc) {
                case 3: *ep++ = 0;
                case 2:  do {
                                *ep++ = 0;
                case 1:         *ep++ = 0;
                case 0:         *ep++ = *xMp++;
                         } while (--i);
        }
        while (++Mc < 4) *ep++ = 0;
}

/* 4.2.18 */

/*  This procedure adds the reconstructed long term residual signal
 *  ep[0..39] to the estimated signal dpp[0..39] from the long term
 *  analysis filter to compute the reconstructed short term residual
 *  signal dp[-40..-1]; also the reconstructed short term residual
 *  array dp[-120..-41] is updated.
 */
```

```
#if 0 /* Has been inlined in code.c */
void Amr_Update_of_reconstructed_short_time_residual_signal(
      word  * dpp,            /* [0...39] IN    */
      word  * ep,       /* [0...39] IN    */
      word  * dp)       /* [-120...-1]  IN/OUT  */
{
      int         k;

      for (k = 0; k <= 79; k++)
            dp[ -120 + k ] = dp[ -80 + k ];

      for (k = 0; k <= 39; k++)
            dp[ -40 + k ] = amr_add( ep[k], dpp[k] );
}
#endif      /* Has been inlined in code.c */

void Amr_RPE_Encoding(
      word  * e,        /* -5..-1][0..39][40..44      IN/OUT  */
      word  * xmaxc,    /*                       OUT */
      word  * Mc,       /*                       OUT */
      word  * xMc)            /* [0..12]              OUT */
{
      word  x[40];
      word  xM[13], xMp[13];
      word  mant, exp;

      Weighting_filter(e, x);
      RPE_grid_selection(x, xM, Mc);

      APCM_quantization(      xM, xMc, &mant, &exp, xmaxc);
      APCM_inverse_quantization(  xMc,  mant,  exp, xMp);

      RPE_grid_positioning( *Mc, xMp, e );

}

void Amr_RPE_Decoding(
      word                xmaxcr,
      word                Mcr,
      word              * xMcr,  /* [0..12], 3 bits         IN    */
      word              * erp  /* [0..39]             OUT   */
)
{
      word  exp, mant;
      word  xMp[ 13 ];

      APCM_quantization_xmaxc_to_exp_mant( xmaxcr, &exp, &mant );
      APCM_inverse_quantization( xMcr, mant, exp, xMp );
      RPE_grid_positioning( Mcr, xMp, erp );

}


/*
 *  4.2 FIXED POINT IMPLEMENTATION OF THE RPE-LTP CODER
 */

void Amr_Coder(
      struct amr_state  * S,
      word  * s,  /* [0..159] samples                IN    */
```

```
/*
 * The RPE-LTD coder works on a frame by frame basis.  The length of
 * the frame is equal to 160 samples.  Some computations are done
 * once per frame to produce at the output of the coder the
 * LARc[1..8] parameters which are the coded LAR coefficients and
 * also to realize the inverse filtering operation for the entire
 * frame (160 samples of signal d[0..159]).  These parts produce at
 * the output of the coder:
 */

      word  * LARc,     /* [0..7] LAR coefficients         OUT   */

/*
 * Procedure 4.2.11 to 4.2.18 are to be executed four times per
 * frame.  That means once for each sub-segment RPE-LTP analysis of
 * 40 samples.  These parts produce at the output of the coder:
 */

      word  * Nc, /* [0..3] LTP lag            OUT   */
      word  * bc, /* [0..3] coded LTP gain           OUT   */
      word  * Mc, /* [0..3] RPE grid selection       OUT     */
      word  * xmaxc,/* [0..3] Coded maximum amplitude OUT   */
      word  * xMc /* [13*4] normalized RPE samples   OUT   */
)
{
      int   k;
      word  * dp  = S->dp0 + 120;   /* [ -120...-1 ] */
      word  * dpp = dp;       /* [ 0...39 ]      */

      static word e [50] = {0};

      word  so[160];

      Amr_Preprocess                (S, s, so);
      Amr_LPC_Analysis        (S, so, LARc);
      Amr_Short_Term_Analysis_Filter      (S, LARc, so);

      for (k = 0; k <= 3; k++, xMc += 13) {

            Amr_Long_Term_Predictor ( S,
                              so+k*40, /* d      [0..39] IN     */
                              dp,    /* dp  [-120..-1] IN  */
                             e + 5,        /* e      [0..39] OUT */
                             dpp,    /* dpp    [0..39] OUT */
                              Nc++,
                              bc++);

            Amr_RPE_Encoding  (
                             e + 5,      /* e    ][0..39][ IN/OUT */
                               xmaxc++, Mc++, xMc );
            /*
             * Amr_Update_of_reconstructed_short_time_residual_signal
             *                ( dpp, e + 5, dp );
             */

            { register int i;
              for (i = 0; i <= 39; i++)
                  dp[ i ] = (word) AMR_ADD( e[5 + i], dpp[i] );
            }
            dp  += 40;
```

```
            dpp += 40;

      }
      (void)memcpy( (char *)S->dp0, (char *)(S->dp0 + 160),
            120 * sizeof(*S->dp0) );
}


void amr_encode(amr s, amr_signal * source, amr_byte * c)
{
      word        LARc[8], Nc[4], Mc[4], bc[4], xmaxc[4], xmc[13*4];

      Amr_Coder(s, source, LARc, Nc, bc, Mc, xmaxc, xmc);


      /*    variable    size

            AMR_MAGIC   4

            LARc[0]           6
            LARc[1]           6
            LARc[2]           5
            LARc[3]           5
            LARc[4]           4
            LARc[5]           4
            LARc[6]           3
            LARc[7]           3

            Nc[0]       7
            bc[0]       2
            Mc[0]       2
            xmaxc[0]    6
            xmc[0]            3
            xmc[1]            3
            xmc[2]            3
            xmc[3]            3
            xmc[4]            3
            xmc[5]            3
            xmc[6]            3
            xmc[7]            3
            xmc[8]            3
            xmc[9]            3
            xmc[10]           3
            xmc[11]           3
            xmc[12]           3

            Nc[1]       7
            bc[1]       2
            Mc[1]       2
            xmaxc[1]    6
            xmc[13]           3
            xmc[14]           3
            xmc[15]           3
            xmc[16]           3
            xmc[17]           3
            xmc[18]           3
            xmc[19]           3
            xmc[20]           3
            xmc[21]           3
            xmc[22]           3
            xmc[23]           3
```

```
        xmc[24]           3
        xmc[25]           3

        Nc[2]       7
        bc[2]       2
        Mc[2]       2
        xmaxc[2]    6
        xmc[26]           3
        xmc[27]           3
        xmc[28]           3
        xmc[29]           3
        xmc[30]           3
        xmc[31]           3
        xmc[32]           3
        xmc[33]           3
        xmc[34]           3
        xmc[35]           3
        xmc[36]           3
        xmc[37]           3
        xmc[38]           3

        Nc[3]       7
        bc[3]       2
        Mc[3]       2
        xmaxc[3]    6
        xmc[39]           3
        xmc[40]           3
        xmc[41]           3
        xmc[42]           3
        xmc[43]           3
        xmc[44]           3
        xmc[45]           3
        xmc[46]           3
        xmc[47]           3
        xmc[48]           3
        xmc[49]           3
        xmc[50]           3
        xmc[51]           3
*/


*c++ =   ((AMR_MAGIC & 0xF) << 4)         /* 1 */
       | ((LARc[0] >> 2) & 0xF);
*c++ =   ((LARc[0] & 0x3) << 6)
       | (LARc[1] & 0x3F);
*c++ =   ((LARc[2] & 0x1F) << 3)
       | ((LARc[3] >> 2) & 0x7);
*c++ =   ((LARc[3] & 0x3) << 6)
       | ((LARc[4] & 0xF) << 2)
       | ((LARc[5] >> 2) & 0x3);
*c++ =   ((LARc[5] & 0x3) << 6)
       | ((LARc[6] & 0x7) << 3)
       | (LARc[7] & 0x7);
*c++ =   ((Nc[0] & 0x7F) << 1)
       | ((bc[0] >> 1) & 0x1);
*c++ =   ((bc[0] & 0x1) << 7)
       | ((Mc[0] & 0x3) << 5)
       | ((xmaxc[0] >> 1) & 0x1F);
*c++ =   ((xmaxc[0] & 0x1) << 7)
       | ((xmc[0] & 0x7) << 4)
       | ((xmc[1] & 0x7) << 1)
```

```
        | ((xmc[2] >> 2) & 0x1);
*c++ =    ((xmc[2] & 0x3) << 6)
        | ((xmc[3] & 0x7) << 3)
        | (xmc[4] & 0x7);
*c++ =    ((xmc[5] & 0x7) << 5)                    /* 10 */
        | ((xmc[6] & 0x7) << 2)
        | ((xmc[7] >> 1) & 0x3);
*c++ =    ((xmc[7] & 0x1) << 7)
        | ((xmc[8] & 0x7) << 4)
        | ((xmc[9] & 0x7) << 1)
        | ((xmc[10] >> 2) & 0x1);
*c++ =    ((xmc[10] & 0x3) << 6)
        | ((xmc[11] & 0x7) << 3)
        | (xmc[12] & 0x7);
*c++ =    ((Nc[1] & 0x7F) << 1)
        | ((bc[1] >> 1) & 0x1);
*c++ =    ((bc[1] & 0x1) << 7)
        | ((Mc[1] & 0x3) << 5)
        | ((xmaxc[1] >> 1) & 0x1F);
*c++ =    ((xmaxc[1] & 0x1) << 7)
        | ((xmc[13] & 0x7) << 4)
        | ((xmc[14] & 0x7) << 1)
        | ((xmc[15] >> 2) & 0x1);
*c++ =    ((xmc[15] & 0x3) << 6)
        | ((xmc[16] & 0x7) << 3)
        | (xmc[17] & 0x7);
*c++ =    ((xmc[18] & 0x7) << 5)
        | ((xmc[19] & 0x7) << 2)
        | ((xmc[20] >> 1) & 0x3);
*c++ =    ((xmc[20] & 0x1) << 7)
        | ((xmc[21] & 0x7) << 4)
        | ((xmc[22] & 0x7) << 1)
        | ((xmc[23] >> 2) & 0x1);
*c++ =    ((xmc[23] & 0x3) << 6)
        | ((xmc[24] & 0x7) << 3)
        | (xmc[25] & 0x7);
*c++ =    ((Nc[2] & 0x7F) << 1)                    /* 20 */
        | ((bc[2] >> 1) & 0x1);
*c++ =    ((bc[2] & 0x1) << 7)
        | ((Mc[2] & 0x3) << 5)
        | ((xmaxc[2] >> 1) & 0x1F);
*c++ =    ((xmaxc[2] & 0x1) << 7)
        | ((xmc[26] & 0x7) << 4)
        | ((xmc[27] & 0x7) << 1)
        | ((xmc[28] >> 2) & 0x1);
*c++ =    ((xmc[28] & 0x3) << 6)
        | ((xmc[29] & 0x7) << 3)
        | (xmc[30] & 0x7);
*c++ =    ((xmc[31] & 0x7) << 5)
        | ((xmc[32] & 0x7) << 2)
        | ((xmc[33] >> 1) & 0x3);
*c++ =    ((xmc[33] & 0x1) << 7)
        | ((xmc[34] & 0x7) << 4)
        | ((xmc[35] & 0x7) << 1)
        | ((xmc[36] >> 2) & 0x1);
*c++ =    ((xmc[36] & 0x3) << 6)
        | ((xmc[37] & 0x7) << 3)
        | (xmc[38] & 0x7);
*c++ =    ((Nc[3] & 0x7F) << 1)
        | ((bc[3] >> 1) & 0x1);
*c++ =    ((bc[3] & 0x1) << 7)
```

```
              | ((Mc[3] & 0x3) << 5)
              | ((xmaxc[3] >> 1) & 0x1F);
      *c++ =    ((xmaxc[3] & 0x1) << 7)
              | ((xmc[39] & 0x7) << 4)
              | ((xmc[40] & 0x7) << 1)
              | ((xmc[41] >> 2) & 0x1);
      *c++ =    ((xmc[41] & 0x3) << 6)                /* 30 */
              | ((xmc[42] & 0x7) << 3)
              | (xmc[43] & 0x7);
      *c++ =    ((xmc[44] & 0x7) << 5)
              | ((xmc[45] & 0x7) << 2)
              | ((xmc[46] >> 1) & 0x3);
      *c++ =    ((xmc[46] & 0x1) << 7)
              | ((xmc[47] & 0x7) << 4)
              | ((xmc[48] & 0x7) << 1)
              | ((xmc[49] >> 2) & 0x1);
      *c++ =    ((xmc[49] & 0x3) << 6)
              | ((xmc[50] & 0x7) << 3)
              | (xmc[51] & 0x7);
}

/*
 *  4.3 FIXED POINT IMPLEMENTATION OF THE RPE-LTP DECODER
 */

static void Postprocessing(struct amr_state *S, register word *s)
{
      register int            k;
      register word           msr = S->msr;
      register word           tmp;

      for (k = 160; k--; s++) {
            tmp = AMR_MULT_R( msr, 28180 );
            msr = (word) AMR_ADD(*s, tmp);         /* Deemphasis
*/
            *s  = AMR_ADD(msr, msr) & 0xFFF8;  /* Truncation &
Upscaling */
      }
      S->msr = msr;
}

void Amr_Decoder(
      struct amr_state  * S,

      word         * LARcr,   /* [0..7]         IN    */

      word         * Ncr,           /* [0..3]         IN    */
      word         * bcr,           /* [0..3]         IN    */
      word         * Mcr,           /* [0..3]         IN    */
      word         * xmaxcr,  /* [0..3]         IN    */
      word         * xMcr,          /* [0..13*4]            IN    */

      word         * s        /* [0..159]       OUT   */
)
{
      int          j, k;
      word         erp[40], wt[160];
      word         *drp = S->dp0 + 120;

      for (j=0; j <= 3; j++, xmaxcr++, bcr++, Ncr++, Mcr++, xMcr +=
13) {
```

```
            Amr_RPE_Decoding(*xmaxcr, *Mcr, xMcr, erp);
            Amr_Long_Term_Synthesis_Filtering(S, *Ncr, *bcr, erp, drp);
            for (k = 0; k <= 39; k++) {
                  wt[ j * 40 + k ] =  drp[ k ];
            }
      }

      Amr_Short_Term_Synthesis_Filter(S, LARcr, wt, s);
      Postprocessing(S, s);
}

/*
 * Copyright 1992 by Jutta Degener and Carsten Bormann, Technische
 * Universitaet Berlin.  See the accompanying file "COPYRIGHT" for
 * details.  THERE IS ABSOLUTELY NO WARRANTY FOR THIS SOFTWARE.
 */

/* $Header:
/cs/research/mice/starship/src/local/CVS_repository/rat/cx_amr.c,v 1.6
2004/01/12 17:14:35 ucacoxh Exp $ */

int amr_decode(amr s, amr_byte * c, amr_signal * target)
{
      word        LARc[8], Nc[4], Mc[4], bc[4], xmaxc[4], xmc[13*4];

      /* AMR_MAGIC  = (*c >> 4) & 0xF; */

      if (((*c >> 4) & 0x0F) != AMR_MAGIC) return -1;

      LARc[0]  = (*c++ & 0xF) << 2;       /* 1 */
      LARc[0] |= (*c >> 6) & 0x3;
      LARc[1]  = *c++ & 0x3F;
      LARc[2]  = (*c >> 3) & 0x1F;
      LARc[3]  = (*c++ & 0x7) << 2;
      LARc[3] |= (*c >> 6) & 0x3;
      LARc[4]  = (*c >> 2) & 0xF;
      LARc[5]  = (*c++ & 0x3) << 2;
      LARc[5] |= (*c >> 6) & 0x3;
      LARc[6]  = (*c >> 3) & 0x7;
      LARc[7]  = *c++ & 0x7;
      Nc[0]  = (*c >> 1) & 0x7F;
      bc[0]  = (*c++ & 0x1) << 1;
      bc[0] |= (*c >> 7) & 0x1;
      Mc[0]  = (*c >> 5) & 0x3;
      xmaxc[0]  = (*c++ & 0x1F) << 1;
      xmaxc[0] |= (*c >> 7) & 0x1;
      xmc[0]  = (*c >> 4) & 0x7;
      xmc[1]  = (*c >> 1) & 0x7;
      xmc[2]  = (*c++ & 0x1) << 2;
      xmc[2] |= (*c >> 6) & 0x3;
      xmc[3]  = (*c >> 3) & 0x7;
      xmc[4]  = *c++ & 0x7;
      xmc[5]  = (*c >> 5) & 0x7;
      xmc[6]  = (*c >> 2) & 0x7;
      xmc[7]  = (*c++ & 0x3) << 1;        /* 10 */
      xmc[7] |= (*c >> 7) & 0x1;
      xmc[8]  = (*c >> 4) & 0x7;
      xmc[9]  = (*c >> 1) & 0x7;
      xmc[10]  = (*c++ & 0x1) << 2;
      xmc[10] |= (*c >> 6) & 0x3;
      xmc[11]  = (*c >> 3) & 0x7;
```

```
xmc[12]  = *c++ & 0x7;
Nc[1]  = (*c >> 1) & 0x7F;
bc[1]  = (*c++ & 0x1) << 1;
bc[1] |= (*c >> 7) & 0x1;
Mc[1]  = (*c >> 5) & 0x3;
xmaxc[1]  = (*c++ & 0x1F) << 1;
xmaxc[1] |= (*c >> 7) & 0x1;
xmc[13]  = (*c >> 4) & 0x7;
xmc[14]  = (*c >> 1) & 0x7;
xmc[15]  = (*c++ & 0x1) << 2;
xmc[15] |= (*c >> 6) & 0x3;
xmc[16]  = (*c >> 3) & 0x7;
xmc[17]  = *c++ & 0x7;
xmc[18]  = (*c >> 5) & 0x7;
xmc[19]  = (*c >> 2) & 0x7;
xmc[20]  = (*c++ & 0x3) << 1;
xmc[20] |= (*c >> 7) & 0x1;
xmc[21]  = (*c >> 4) & 0x7;
xmc[22]  = (*c >> 1) & 0x7;
xmc[23]  = (*c++ & 0x1) << 2;
xmc[23] |= (*c >> 6) & 0x3;
xmc[24]  = (*c >> 3) & 0x7;
xmc[25]  = *c++ & 0x7;
Nc[2]  = (*c >> 1) & 0x7F;
bc[2]  = (*c++ & 0x1) << 1;        /* 20 */
bc[2] |= (*c >> 7) & 0x1;
Mc[2]  = (*c >> 5) & 0x3;
xmaxc[2]  = (*c++ & 0x1F) << 1;
xmaxc[2] |= (*c >> 7) & 0x1;
xmc[26]  = (*c >> 4) & 0x7;
xmc[27]  = (*c >> 1) & 0x7;
xmc[28]  = (*c++ & 0x1) << 2;
xmc[28] |= (*c >> 6) & 0x3;
xmc[29]  = (*c >> 3) & 0x7;
xmc[30]  = *c++ & 0x7;
xmc[31]  = (*c >> 5) & 0x7;
xmc[32]  = (*c >> 2) & 0x7;
xmc[33]  = (*c++ & 0x3) << 1;
xmc[33] |= (*c >> 7) & 0x1;
xmc[34]  = (*c >> 4) & 0x7;
xmc[35]  = (*c >> 1) & 0x7;
xmc[36]  = (*c++ & 0x1) << 2;
xmc[36] |= (*c >> 6) & 0x3;
xmc[37]  = (*c >> 3) & 0x7;
xmc[38]  = *c++ & 0x7;
Nc[3]  = (*c >> 1) & 0x7F;
bc[3]  = (*c++ & 0x1) << 1;
bc[3] |= (*c >> 7) & 0x1;
Mc[3]  = (*c >> 5) & 0x3;
xmaxc[3]  = (*c++ & 0x1F) << 1;
xmaxc[3] |= (*c >> 7) & 0x1;
xmc[39]  = (*c >> 4) & 0x7;
xmc[40]  = (*c >> 1) & 0x7;
xmc[41]  = (*c++ & 0x1) << 2;
xmc[41] |= (*c >> 6) & 0x3;
xmc[42]  = (*c >> 3) & 0x7;
xmc[43]  = *c++ & 0x7;             /* 30  */
xmc[44]  = (*c >> 5) & 0x7;
xmc[45]  = (*c >> 2) & 0x7;
xmc[46]  = (*c++ & 0x3) << 1;
xmc[46] |= (*c >> 7) & 0x1;
```

```
      xmc[47]  = (*c >> 4) & 0x7;
      xmc[48]  = (*c >> 1) & 0x7;
      xmc[49]  = (*c++ & 0x1) << 2;
      xmc[49] |= (*c >> 6) & 0x3;
      xmc[50]  = (*c >> 3) & 0x7;
      xmc[51]  = *c & 0x7;               /* 33 */

      Amr_Decoder(s, LARc, Nc, bc, Mc, xmaxc, xmc, target);

      return 0;
}

void amr_destroy(amr S)
{
      if (S) free((char *)S);
}

struct amr_state *amr_create(void)
{
      struct amr_state *r;

      r = (struct amr_state *) malloc(sizeof(struct amr_state));
      assert(r != NULL);

      memset(r, 0, sizeof(struct amr_state));
      r->nrp = 40;

      return r;
}
```

e) codec_amr2.h

```
/*
 * FILE:    codec_amr2.h
 *
 *
 *
 */

#ifndef _CODEC_AMR_H_
#define _CODEC_AMR_H_

uint16_t                      amr_get_formats_count (void);
const struct s_codec_format* amr_get_format(uint16_t idx);

int  amr_state_create  (uint16_t idx, u_char **state);
void amr_state_destroy (uint16_t idx, u_char **state);
int  amr_encoder       (uint16_t idx, u_char *state, sample *in,
coded_unit *out);
int  amr_decoder       (uint16_t idx, u_char *state, coded_unit *in,
sample *out);

int  amr_repair        (uint16_t idx, u_char *state, uint16_t
consec_lost,
                        coded_unit *prev, coded_unit *missing,
coded_unit *next);



#endif /* _CODEC_AMR_H_ */
```

f) cx_amr.h

```
#ifndef     _AMR_P_H_
#define     _AMR_P_H_

#define     HAS_STDLIB_H      1           /* /usr/include/stdlib.h
      */
#define     HAS_FCNTL_H 1           /* /usr/include/fcntl.h      */
#define     HAS_FSTAT   1           /* fstat syscall       */
#define     HAS_FCHMOD  1           /* fchmod syscall      */
#define     HAS_FCHOWN  1           /* fchown syscall      */
#define     HAS_STRING_H      1           /* /usr/include/string.h
      */
#define     HAS_UNISTD_H      1           /* /usr/include/unistd.h
      */
#define     HAS_UTIME   1           /* POSIX utime(path, times)   */
#define     HAS_UTIME_H 1           /* UTIME header file         */

typedef short                 word;       /* 16 bit signed int    */
typedef long                  longword;   /* 32 bit signed int    */

typedef unsigned short        uword;            /* unsigned word  */
typedef unsigned long         ulongword;  /* unsigned longword    */

struct amr_state {
      word        dp0[ 280 ];
      word        z1;         /* preprocessing.c, Offset_com. */
      longword    L_z2;       /*                  Offset_com. */
      int         mp;         /                   Preemphasis    */
      word        u[8];       /* short_term_aly_filter.c    */
      word        LARpp[2][8];      /*                              */
      word        j;          /                              */
      word            ltp_cut;        /* long_term.c, LTP crosscorr.
*/
      word        nrp; /* 40 */     /* long_term.c, synthesis     */
      word        v[9];       /* short_term.c, synthesis    */
      word        msr;        /* decoder.c,     Postprocessing    */
      char        verbose;    /* only used if !NDEBUG       */
      char        fast;       /* only used if FAST          */
};


#define     MIN_WORD    ((-32767)-1)
#define     MAX_WORD    ( 32767)

#define     MIN_LONGWORD      ((-2147483647)-1)
#define     MAX_LONGWORD      ( 2147483647)

#ifdef      SASR        /* >> is a signed arithmetic shift right */
#undef      SASR
#define     SASR(x, by) ((x) >> (by))
#endif      /* SASR */


/*
 *    Prototypes from add.c
 */
extern word amr_mult    (word a, word b);
extern longword amr_L_mult    (word a, word b);
```

```
extern word amr_mult_r  (word a, word b);

extern word amr_div     (word num, word denum);

extern word amr_add     ( word a, word b );
extern longword amr_L_add     ( longword a, longword b );

extern word amr_sub     (word a, word b);
extern longword amr_L_sub     (longword a, longword b);

extern word amr_abs     (word a);

extern word amr_norm    ( longword a );

extern longword amr_L_asl     (longword a, int n);
extern word amr_asl     (word a, int n);

extern longword amr_L_asr     (longword a, int n);
extern word amr_asr     (word a, int n);


# define AMR_MULT_R(a, b)     amr_mult_r(a, b)
# define AMR_MULT(a, b)       amr_mult(a, b)
# define AMR_L_MULT(a, b)     amr_L_mult(a, b)

# define AMR_L_ADD(a, b)      amr_L_add(a, b)
# define AMR_ADD(a, b)        amr_add(a, b)
# define AMR_SUB(a, b)        amr_sub(a, b)

# define AMR_ABS(a)           amr_abs(a)


/*
 *  More prototypes from implementations..
 */
extern void Amr_Coder (
            struct amr_state  * S,
            word  * s,  /* [0..159] samples           IN    */
            word  * LARc,     /* [0..7] LAR coefficients    OUT   */
            word  * Nc, /* [0..3] LTP lag       OUT   */
            word  * bc, /* [0..3] coded LTP gain      OUT   */
            word  * Mc, /* [0..3] RPE grid selection  OUT     */
            word  * xmaxc,/* [0..3] Coded maximum amplitude OUT  */
            word  * xMc /* [13*4] normalized RPE samples OUT     */);

extern void Amr_Long_Term_Predictor (          /* 4x for 160 samples
*/
            struct amr_state * S,
            word  * d,  /* [0..39]   residual signal  IN    */
            word  * dp, /* [-120..-1] d'        IN    */
            word  * e,  /* [0..40]              OUT   */
            word  * dpp,      /* [0..40]              OUT   */
            word  * Nc, /* correlation lag            OUT   */
            word  * bc  /* gain factor                OUT   */);

extern void Amr_LPC_Analysis (
            struct amr_state * S,
            word * s,    /* 0..159 signals      IN/OUT      */
              word * LARc);   /* 0..7   LARc's OUT   */

extern void Amr_Preprocess (
```

```
            struct amr_state * S,
            word * s, word * so);

extern void Amr_Encoding (
            struct amr_state * S,
            word  * e,
            word  * ep,
            word  * xmaxc,
            word  * Mc,
            word  * xMc);

extern void Amr_Short_Term_Analysis_Filter (
            struct amr_state * S,
            word  * LARc,     /* coded log area ratio [0..7]  IN */
            word  * d   /* st res. signal [0..159]    IN/OUT      */);

extern void Amr_Decoder (
            struct amr_state * S,
            word  * LARcr,    /* [0..7]         IN    */
            word  * Ncr,            /* [0..3]         IN    */
            word  * bcr,            /* [0..3]         IN    */
            word  * Mcr,            /* [0..3]         IN    */
            word  * xmaxcr,   /* [0..3]         IN    */
            word  * xMcr,           /* [0..13*4]            IN    */
            word  * s);       /* [0..159]       OUT   */

extern void Amr_Decoding (
            struct amr_state * S,
            word  xmaxcr,
            word  Mcr,
            word  * xMcr,     /* [0..12]        IN    */
            word  * erp);     /* [0..39]        OUT   */

extern void Amr_Long_Term_Synthesis_Filtering (
            struct amr_state* S,
            word  Ncr,
            word  bcr,
            word  * erp,            /* [0..39]          IN  */
            word  * drp);     /* [-120..-1] IN, [0..40] OUT      */

void Amr_RPE_Decoding (
            word xmaxcr,
            word Mcr,
            word * xMcr,  /* [0..12], 3 bits             IN      */
            word * erp); /* [0..39]                     OUT     */

void Amr_RPE_Encoding (
            word    * e,            /* -5..-1][0..39][40..44     IN/OUT
*/
            word    * xmaxc,        /*                              OUT
*/
            word    * Mc,           /*                              OUT
*/
            word    * xMc);        /* [0..12]                      OUT
*/

extern void Amr_Short_Term_Synthesis_Filter (
            struct amr_state * S,
            word  * LARcr,    /* log area ratios [0..7]  IN */
            word  * drp,            /* received d [0...39]     IN */
            word  * s);       /* signal   s [0..159]    OUT */
```

```
extern void Amr_Update_of_reconstructed_short_time_residual_signal (
            word  * dpp,            /* [0...39] IN    */
            word  * ep,       /* [0...39] IN    */
            word  * dp);            /* [-120...-1]  IN/OUT  */

/*
 *  Tables from table.c
 */
#ifndef     AMR_TABLE_C

extern word amr_A[8], amr_B[8], amr_MIC[8], amr_MAC[8];
extern word amr_INVA[8];
extern word amr_DLB[4], amr_QLB[4];
extern word amr_H[11];
extern word amr_NRFAC[8];
extern word amr_FAC[8];

#endif      /* AMR_TABLE_C */

/*
 *  Debugging
 */
#ifdef NDEBUG

#     define      amr_debug_words(a, b, c, d)         /* nil */
#     define      amr_debug_longwords(a, b, c, d)           /* nil */
#     define      amr_debug_word(a, b)                /* nil */
#     define      amr_debug_longword(a, b)            /* nil */

#else /* !NDEBUG => DEBUG */

      extern void  amr_debug_words     (char * name, int, int, word
*);
      extern void  amr_debug_longwords (char * name, int, int,
longword *);
      extern void  amr_debug_longword  (char * name, longword);
      extern void  amr_debug_word      (char * name, word);

#endif /* !NDEBUG */

typedef struct amr_state *    amr;
typedef short                 amr_signal;       /* signed 16 bit */
typedef unsigned char         amr_byte;
typedef amr_byte        amr_frame[33];          /* 33 * 8 bits     */

#define AMR_FRAMESIZE   33
#define     AMR_MAGIC   0xD                     /* 13 kbit/s RPE-LTP */

#define     AMR_PATCHLEVEL    7
#define     AMR_MINOR   0
#define     AMR_MAJOR   1

#define     AMR_OPT_VERBOSE   1
#define     AMR_OPT_FAST      2
#define     AMR_OPT_LTP_CUT   3

extern amr  amr_create  (void);
extern void amr_destroy (amr);
extern int  amr_print   (FILE *, amr, amr_byte  *);
extern int  amr_option  (amr, int, int *);
```

```
extern void amr_encode  (amr, amr_signal *, amr_byte  *);
extern int  amr_decode  (amr, amr_byte   *, amr_signal *);
extern int  amr_explode (amr, amr_byte   *, amr_signal *);
extern void amr_implode (amr, amr_signal *, amr_byte   *);

#endif      /* _AMR_P_H_ */
```

g) Makefile

```
# Generated automatically from Makefile.in by configure.
#
# Makefile for the RAT project. This requires GNU make on many
systems.
#
# $Id: Makefile.in,v 1.143 2002/03/15 17:53:58 ucacoxh Exp $
#
# Configure substitutes variables here...
####################################

DEFS    = -DHAVE_CONFIG_H
CFLAGS  = -g -O2 -W -Wall -Wbad-function-cast -Wmissing-prototypes -
Werror $(DEFS)
LIBS    = -ldl  -lieee -lm
LDLIBS  =
INCLUDE = -I/root/LPFC/rat/final_beta_AMRAuto/mbone_tools/common/src
-I/root/LPFC/rat/final_beta_AMRAuto/mbone_tools/tcl-8.0/generic -
I/root/LPFC/rat/final_beta_AMRAuto/mbone_tools/tk-8.0/generic
CC      = gcc
AR      = ar
RANLIB  = ranlib
ECHO    = echo
STRIP = strip

VERSION = 4.2.26
RATVER  = rat-$(VERSION)

AU_OBJ  =  auddev_oss.o
AU_INC  =
AU_LIB  =

TCL_LIB    = -L/root/LPFC/rat/final_beta_AMRAuto/mbone_tools/tcl-
8.0/unix -ltcl8.0
TK_LIB     = -L/root/LPFC/rat/final_beta_AMRAuto/mbone_tools/tk-
8.0/unix -ltk8.0
X_CFLAGS   =  -I/usr/X11R6/include
X_LIB      =  -lSM -lICE  -L/usr/X11R6/lib -lX11 -lXext
COMMON_LIB = -
L/root/LPFC/rat/final_beta_AMRAuto/mbone_tools/common/src -luclmmbase
EXTRA_OBJ  =
EXTERNAL_DEP =
/root/LPFC/rat/final_beta_AMRAuto/mbone_tools/common/src/libuclmmbase.
a

G728_CODEC_OBJ =
G728_LIB       =

WBS_CODEC_OBJ = codec_wbs.o cx_wbs.o

prefix      = /usr/local
```

```
exec_prefix = ${prefix}
bindir      = ${exec_prefix}/bin
mandir      = ${prefix}/man

# Nothing below here set by configure
########################################

AUDIO_OBJS   = $(AU_OBJ) auddev_null.o audio_fmt.o audio_util.o \
               auddev_trans.o auddev.o

CODEC_OBJS   = encoder.o interf_enc.o sp_enc.o decoder.o interf_dec.o
sp_dec.o codec.o codec_state.o codec_dvi.o codec_gsm.o codec_amr.o
cx_amr.o  \
               codec_l8.o codec_l16.o codec_g711.o codec_g726.o
codec_lpc.o \
               codec_types.o codec_vdvi.o \
               cx_g726.o cx_g726_16.o cx_g726_24.o cx_g726_32.o
cx_g726_40.o \
               cx_gsm.o cx_lpc.o cx_vdvi.o cx_dvi.o \
               converter.o convert_util.o convert_extra.o
convert_linear.o \
               convert_sinc.o bitstream.o  $(WBS_CODEC_OBJ)

SNDFILE_OBJS = sndfile.o sndfile_au.o sndfile_raw.o sndfile_wav.o

MEDIALIBS    = libuclaudio.a libuclcodec.a libuclsndfile.a

CHANNEL_OBJS = channel.o channel_types.o cc_vanilla.o cc_rdncy.o
cc_layered.o

TOY_OBJS     = render_3D.o repair.o

MEDIA_OBJS   = ts.o playout.o net.o source.o session.o \
               main_engine.o mbus_engine.o audio.o cushion.o mix.o \
               parameters.o transmit.o playout_calc.o \
               ui_send_rtp.o ui_send_audio.o ui_send_prefs.o
ui_send_stats.o \
             transcoder.o rtp_dump.o rtp_callback.o settings.o \
               pdb.o pktbuf.o tonegen.o voxlet.o fatal_error.o
$(EXTRA_OBJ)

UI_OBJS      = tcltk.o mbus_ui.o main_ui.o fatal_error.o

CTRL_OBJS    = main_control.o mbus_control.o codec_compat.o
fatal_error.o \
               process.o cmd_parser.o

TCL_OBJS     = ui_audiotool.o ui_transcoder.o

TCL_SRCS     = $(TCL_OBJS:%.o=%.c)

ALL_OBJS     = $(AUDIO_OBJS) $(CODEC_OBJS) $(SNDFILE_OBJS)
$(CHANNEL_OBJS) \
               $(TOY_OBJS) $(MEDIA_OBJS) $(UI_OBJS) $(CTRL_OBJS)

ALL_SRCS     = $(ALL_OBJS:%.o=%.c)

INSTALL_OBJS = ui_installer.o installer.o binaries.o

all: version.h sdr2.plugin.S02.audio.rtp.-.$(RATVER) $(RATVER).spec
$(RATVER) $(RATVER)-ui $(RATVER)-media rat Makefile
```

```
libuclaudio.a: $(AUDIO_OBJS)
      $(AR) r $@ $(AUDIO_OBJS)
      $(RANLIB) $@

libuclcodec.a: $(CODEC_OBJS)
      $(AR) r $@ $(CODEC_OBJS)
      $(RANLIB) $@

libuclsndfile.a: $(SNDFILE_OBJS)
      $(AR) r $@ $(SNDFILE_OBJS)
      $(RANLIB) $@

$(RATVER)-media: $(CHANNEL_OBJS) $(TOY_OBJS) $(MEDIA_OBJS)
$(MEDIALIBS) $(EXTERNAL_DEP)
      $(CC) $(CFLAGS) $(CHANNEL_OBJS) $(TOY_OBJS) $(MEDIA_OBJS)
$(MEDIALIBS) $(COMMON_LIB) $(AU_LIB) $(LIBS) $(G728_LIB) -o $(RATVER)-
media

$(RATVER)-ui: $(TCL_OBJS) $(UI_OBJS) $(EXTERNAL_DEP)
      $(CC) $(CFLAGS) $(TCL_OBJS) $(UI_OBJS) $(COMMON_LIB)  $(TCL_LIB)
$(TK_LIB) $(X_LIB) $(LIBS) -o $(RATVER)-ui

$(RATVER): $(CTRL_OBJS) $(EXTERNAL_DEP)
      $(CC) $(CFLAGS) $(CTRL_OBJS) $(COMMON_LIB) $(LIBS) -o $(RATVER)

.c.o:
      $(CC) $(CFLAGS) $(X_CFLAGS) $(INCLUDE) -c $<

$(TCL_OBJS): $(TCL_SRCS)

ui_transcoder.c: ui_transcoder.tcl tcl2c/tcl2c
      cat ui_transcoder.tcl | tcl2c/tcl2c ui_transcoder >
ui_transcoder.c

ui_audiotool.c: uicomponent_bargraph.tcl uicomponent_chart.tcl
uicomponent_help.tcl asfilebox.tcl ui_audiotool.tcl tcl2c/tcl2c
      cat uicomponent_bargraph.tcl uicomponent_chart.tcl
uicomponent_help.tcl asfilebox.tcl ui_audiotool.tcl | tcl2c/tcl2c
ui_audiotool > ui_audiotool.c

ui_installer.c: ui_installer.tcl tcl2c/tcl2c
      cat ui_installer.tcl | tcl2c/tcl2c ui_installer > ui_installer.c

tcl2c/tcl2c: tcl2c/tcl2c.c
      $(CC) -o tcl2c/tcl2c tcl2c/tcl2c.c

version.h: VERSION
      @${ECHO} "Generating version.h"
      @sed -e 's/.*/#define RAT_VERSION "&"/' VERSION > version.h

sdr2.plugin.S02.audio.rtp.-.rat-$(VERSION): sdr2.plugin.in
      @${ECHO} "Generating sdr plugin"
      @${ECHO} "# Generated automatically from sdr2.plugin.in" > $@
      @${ECHO} "# DO NOT EDIT THIS FILE" >> $@
      @sed -e 's/VERSION/$(VERSION)/g' sdr2.plugin.in >> $@

rat-$(VERSION).spec: rat.spec
      @${ECHO} "Generating RPM spec file"
      @${ECHO} "# Generated automatically from rat.spec"  > $@
      @${ECHO} "# DO NOT EDIT THIS FILE"                 >> $@
```

```
	@cat rat.spec | sed s/VERSION/$(VERSION)/g	>> $@

rat: VERSION Makefile
	@${ECHO} "Generating rat shell script"
	@${ECHO} "#/bin/sh" > $@
	@${ECHO} "# Generated automatically from RAT Makefile.  Edit
there." >> $@
	@${ECHO} 'PATH=$${PATH}:.' >> $@
	@${ECHO} "exec rat-$(VERSION)" '$$*' >> $@
	@chmod a+x rat

clean:
	-rm -f $(CTRL_OBJS) $(AUDIO_OBJS) $(CODEC_OBJS) $(SNDFILE_OBJS)
$(CHANNEL_OBJS) $(TOY_OBJS) $(MEDIA_OBJS) $(UI_OBJS) $(TCL_OBJS)
	-rm -f $(MEDIALIBS)
	-rm -f tcl_libs.c ui_audiotool.c ui_transcoder.c ui_installer.c
version.h
	-rm -f tcl2c/tcl2c sdr2.plugin.S02.audio.rtp.-.$(RATVER)
$(RATVER).spec
	-rm -f bin2c bin2c.o binaries.c $(INSTALL_OBJS) $(RATVER)-
installer
	-rm -f rat $(RATVER)-media $(RATVER)-ui $(RATVER)

distclean: clean
	-rm -rf config.cache config.log config.status ratconf.h Makefile
	-rm -rf core core-$(RATVER)-media core-$(RATVER)-ui

install: all
	./install-sh -m 555 -c rat             $(bindir)
	./install-sh -m 555 -c $(RATVER)       $(bindir)
	./install-sh -m 555 -c $(RATVER)-media $(bindir)
	./install-sh -m 555 -c $(RATVER)-ui    $(bindir)
	./install-sh -m 444 -c man/man1/rat.1  $(mandir)/man1/rat.1
	./install-sh -m 444 -c sdr2.plugin.S02.audio.rtp.-.$(RATVER)
$(prefix)/etc/sdr/plugins

installer: $(RATVER)-installer
	mkdir release
	mkdir release/$(RATVER)
	cp $(RATVER)-installer README.* MODS COPYRIGHT INSTALL.TXT
VERSION release/$(RATVER)
	cd release && tar cvf $(RATVER)-$(OSTYPE).tar $(RATVER) && gzip
-9 $(RATVER)-$(OSTYPE).tar
	mv release/$(RATVER)-$(OSTYPE).tar.gz .
	rm -rf release

$(RATVER)-installer: $(INSTALL_OBJS)
	$(CC) $(CFLAGS) $(INSTALL_OBJS)  $(TCL_LIB) $(TK_LIB) $(X_LIB)
$(LIBS) -o $(RATVER)-installer

binaries.c: $(RATVER)-$(OSTYPE).tar.gz bin2c
	./bin2c $(RATVER)-$(OSTYPE).tar.gz > binaries.c

bin2c: bin2c.o
	$(CC) bin2c.o -o bin2c

etags:
	etags ../common/src/*.[ch] *.[ch]

ctags:
	ctags ../common/src/*.[ch] *.[ch]
```

```
release:
      cvs tag release-`cat VERSION | sed "s/\./-/g"`

$(RATVER)-$(OSTYPE).tar.gz: $(RATVER) $(RATVER)-ui $(RATVER)-media rat
      tar cf $(RATVER)-$(OSTYPE).tar README.* MODS COPYRIGHT
INSTALL.TXT VERSION $(RATVER) $(RATVER)-ui $(RATVER)-media
sdr2.plugin.S02.audio.rtp.-.rat-$(VERSION) rat
      rm -f $(RATVER)-$(OSTYPE).tar.gz
      gzip -9 $(RATVER)-$(OSTYPE).tar

tgz: $(RATVER)-$(OSTYPE).tar.gz

rpm: tgz
      @${ECHO} "Building $(RATVER).rpm -- this almost certainly needs
to run as root on RedHat Linux"
      install -m 644 $(RATVER)-$(OSTYPE).tar.gz
/usr/src/redhat/SOURCES/rat-4.1.2.tar.gz
      rpm -bb $(RATVER).spec

Makefile: Makefile.in VERSION
      $(error "Run configure again, the following file(s) are newer
than Makefile: $?")

depend:
      makedepend $(X_CFLAGS) $(INCLUDE) $(ALL_SRCS)
# DO NOT DELETE THIS LINE
```