# 4. COLECCIÓN DE ESTADÍSTICOS DE MULTIPLEXIÓN DE FUENTES VOIP

## 4.1. Introducción

Una vez presentada brevemente la tecnología de VoIP, sobre la cual se sustenta este proyecto; el conjunto de parámetros que definen la calidad de una comunicación y los posibles métodos para evaluarla, estamos en disposición de cumplir el objetivo que motiva la realización de este estudio.

A modo de recordatorio, el objetivo propuesto es la obtención de una expresión analítica capaz de otorgar el valor de la calidad, el retardo y las pérdidas de paquetes en la multiplexión de un conjunto de fuentes de tráfico de voz sobre IP en base a parámetros implicados en la codificación, multiplexión y emisión de las conversaciones.

Para ello se distinguen dos etapas claramente definidas. La primera de ella se corresponde con la obtención del conjunto de valores para las prestaciones. Esta fase es la contenida en la actual sección. Tras finalizar este capítulo se presenta la segunda de las fases. En ella se pretende conseguir la expresión matemática marcada como objetivo que ajuste los valores anteriores.

## 4.2. Escenario y metodología de simulación

El escenario de simulación se basa en una emulación del multiplexor de salida de una centralita de de conversaciones sobre VoIP. De tal manera que se dispone de un número variable de usuarios, $N$, que desean realizar una conversación vocal cuya duración, $T_c$, puede tomar tres valores: tres, cinco o treinta minutos. Además, se modela la situación para tres posibles idiomas de los interlocutores: inglés, español y mandarín.

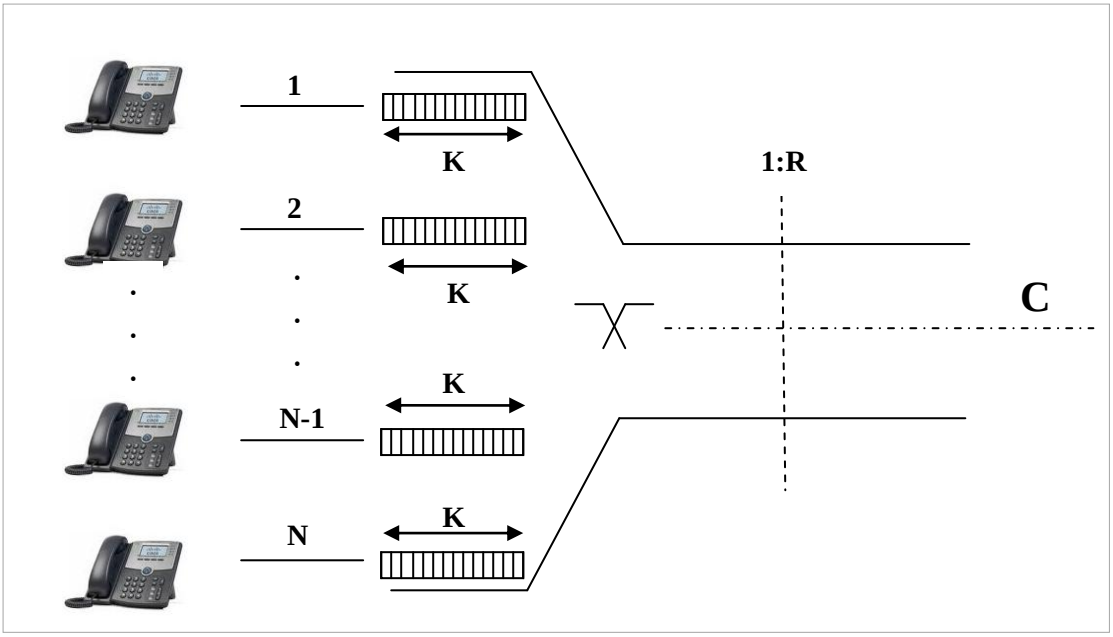



**Figura 12.** ***Escenario de simulación.***

El canal sobre el cual se realiza la transmisión de los paquetes que contienen la información vocal posee un ancho de banda o caudal $C$. No obstante, el multiplexor queda definido mediante los parámetros $K$ y $R$. El primero de ellos especifica la capacidad de almacenamiento del buffer por cada usuario, expresado en paquetes. El segundo expresa el factor de reducción del ancho de banda de salida con respecto al de entrada. Así, todos estos parámetros quedan reflejados en el esquema representado en la figura anterior.

Por otro lado, existen dos parámetros adicionales relacionados con la codificación y el empaquetado de las muestras de voz. Estos son el códec en sí mismo y el número de tramas por paquete, $N_{fpp}$.

Además, cada escenario se simula un número determinado de veces. De tal forma que es posible extraer la media y la desviación de los mismos. A continuación, se presenta los rangos y las unidades de cada uno de estos parámetros.

| Parámetro | Descripción | Unidad | Valores |
|---|---|---|---|
| N | Número de llamadas cursadas simultáneamente | Llamadas | 2 – 80 |
| Tc | Duración de cada conversación | Minutos | 3,5,30 |
| K | Tamaño de la cola de entrada del multiplexor por llamada cursada | Paquetes | 0 – 10 |
| R | Factor de reducción del caudal de salida con respecto al de entrada | Tanto por ciento | 40 – 100 |
| Nfpp | Número de tramas por paquete | Unidades | 1 – 6 |
| Idioma | Idioma de los interlocutores | - | Castellano, Inglés, Mandarín |
| Códec | Códec vocal usado para la codificación y compresión de la voz | - | G.711, G.729, AMR, iLBC |
| Iteraciones | Número de repeticiones para un mismo escenario | - | 50 |

**Tabla 11.** ***Resumen de los parámetros de simulación.***

Para simular el escenario planteado es necesario realizar un meticuloso tratamiento de los ficheros implicados. Esto se debe al elevado volumen de datos manejados. Es por ello que se comienza por la definición concisa de la estructura del árbol de directorio que contendrá todos los ficheros partícipes. Tras disponer de un marco de trabajo, se establece la metodología de simulación, tal cual recoge la figura 13.

A diferencia de otros trabajos que usan el modelo de voz artificial para generar el tráfico de voz, se persiguen resultados más fidedignos a la realidad. Por lo tanto, se utilizan bancos de conversaciones. En este caso ha sido extraído del Consorcio de Datos Lingüísticos (LDC) [35]. Este conjunto de instituciones ofrece un amplío catálogo de conversaciones reales grabadas que se prestan a la finalidad establecida.

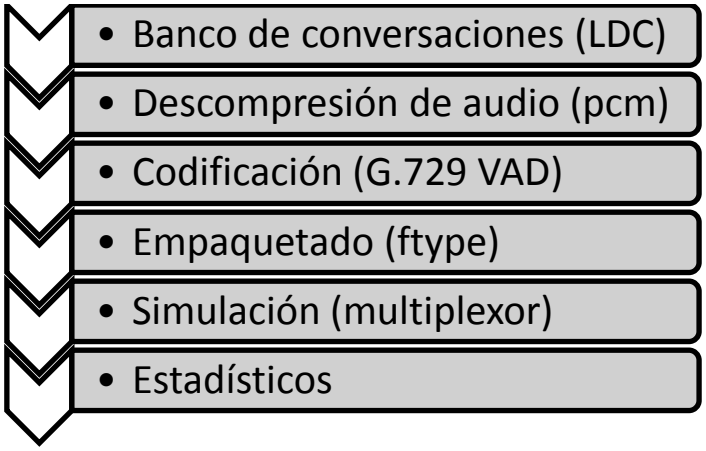



**Figura 13.** ***Metodología de simulación.***

Posteriormente, las conversaciones elegidas son sometidas a operaciones de descompresión, codificación y empaquetamiento como paso previo a su multiplexión y emisión a la red.

Así, estas conversaciones serán utilizadas para su multiplexión en un enlace de salida que simulará el punto más congestionado de las comunicaciones de VoIP como es, generalmente, el router de acceso a la Internet.

Por último, es medido el efecto que surte la multiplexión en las conversaciones (calidad, retardo y pérdida de paquetes) para diferentes condiciones de contorno: capacidad de salida, número de tramas por paquetes, número de conexiones…

### 4.2.1. Árbol de directorios

Antes de comenzar con la explicación de las fases de ejecución de este proyecto cabe mencionar la estructura seguida para la organización de los ficheros de datos y resultados obtenidos.

Se parte de un directorio base creado en el servidor *nerva.us.es*, en el cual se crea en su interior un nuevo directorio llamado *Project*, que servirá de origen para todos los ficheros pertenecientes a este trabajo.

Además, se necesita de la existencia de varios directorios adicionales que contienen ficheros esenciales para poder obtener los resultados perseguidos. Estas carpetas se detallan a continuación.

- **scripts**. Almacena todos los ficheros *.sh* que permiten ejecutar una serie de instrucciones en un terminal de manera automatizada. Además de varios ficheros *.m* pertenecientes a *Matlab*, que constituyen una primera aproximación al escenario de estudio y útiles para el ajuste de curvas.
- **speech**. Contiene los ficheros de conversación comprimidos en formato *sphere* , clasificados en carpetas según el tipo al que pertenezcan: entrenamiento, evaluación o desarrollo; dentro de cada uno de los idiomas seleccionados.
- **src**. Dentro de este directorio se hallan el código fuente, los makefiles y los ejecutables, obtenidos tras su compilación; de todo el software diseñado, ya sea en *C* o en *C++*, necesarios para el desarrollo del estudio planteado. Todo el contenido está organizado en carpetas según el programa al que corresponda.

A partir de estos tres directorios y haciendo uso de un script denominado ***dirTree.sh***, alojado en la carpeta *scripts*, se genera el resto de la jerarquía de directorios necesarias para ubicar todos los ficheros implicados. De tal forma que el conjunto de carpetas que se originan son las siguientes.

- **bin**. En su interior se encuentran enlaces simbólicos a cada uno de los ejecutables generados en los subdirectorios de la carpeta *src*. Esto se realiza así para ofrecer un acceso rápido a cada uno de los ejecutables y agruparlos todos bajo un camino común para su posterior uso.
- **codedFiles**. Este directorio se divide a su vez en tres más para albergar los ficheros voz codificadas en función del idioma al que pertenezcan. Se dividen además en función a la duración de las conversaciones. Comentar que esto es sólo útil para la multiplexión en *Matlab*, pues para la solución final sólo se hace uso de las conversaciones originales, ya que son troceadas posteriormente.

- **decodedFiles**. Al igual que en el caso anterior, se añaden tres subdirectorios, uno para cada idioma. Pero en este caso se usan para almacenar los ficheros de voz que han sido multiplexados y, posteriormente, decodificados teniendo en cuenta la pérdida de paquetes y el códec vocal correspondiente. Este directorio sólo será usado en el primer modelado del problema ya que en el modelo final no es necesario decodificar los ficheros para obtener los parámetros de interés. A su vez se dividen según la duración de las conversaciones con el propósito anteriormente comentado.
- **muxFiles**. Contiene los ficheros de tramas perdidas para cada una de las conversaciones multiplexadas en el escenario de simulación creado con *Matlab*. Estos ficheros serán usados para la decodificación de las conversaciones multiplexadas teniendo en cuenta la pérdida de tramas.
- **pcmFiles**. Es el directorio destino de los ficheros de voz que han sido convertido del formato *sphere* a *pcm* de 16 bits sin cabecera, *raw*. Se organiza también por idiomas y por duración tal cual los anteriores.
- **results**. Este es el directorio donde se almacena el conjunto de resultados obtenidos tras las simulaciones realizadas. Existe un primer subdirectorio, *PESQ*, que contiene los valores de paquetes perdidos, retardo medio y calidad de la comunicación, obtenidos mediante la primera aproximación. En cuanto a los resultados obtenidos mediante la aplicación del modelo final se organizan en tres directorios: *data*, *errors* y *tmp*. El primero contendrá los estadísticos obtenidos tras la multiplexión. El segundo albergará los errores cometidos en el ajuste. El último se utiliza para almacenar ficheros temporales.
- **simulations**. A este directorio se direccionarán los archivos de salida de las distintas simulaciones tal cual son generados por los distintos programas, sin realizar ningún análisis ni tratamiento de los mismos. Se distinguen dos subdirectorios principales que son *PESQ* y *MOS*. El primero se dedica a almacenar los resultados obtenidos mediante la simulación en *Matlab*, mientras que el segundo hace lo propio para el multiplexor en *C++*.

- **typedFiles**. Contiene los denominados ficheros ftype. Éstos proporcionan la información correspondiente al tipo de trama que se ha codificado según la numeración adoptada: 0 para tramas de silencio; 1 para tramas activas y 2 para tramas SID. La jerarquía y distribución de este directorio es ligeramente más compleja. En el primer nivel se presentan los mencionados subdirectorios de idioma de la conversación. A continuación, en su interior se organizan los ficheros en distintos directorios en base al códec de voz usado. Por último, dentro de éstos existe una carpeta denominada *VAD*, en caso de que en la codificación de las muestras vocales se haya usado tramas de supresión de silencios.

Comentar que el script ***dirTree.sh*** también se encarga de compilar el código fuente de cada uno de los ejecutables y de generar los enlaces a los mismos para su posterior ejecución.

### 4.2.2. Banco de conversaciones

Primeramente, como parte de este experimento, es imprescindible disponer de un banco de conversaciones que permitan ser manejadas y utilizadas para su codificación, multiplexión y evaluación.

De tal manera que el conjunto de conversaciones se ha extraído del Consorcio de Datos Lingüísticos [35], más conocido por sus siglas anglosajonas LDC (*Linguistic Data Consortium*). Esta institución está conformada por universidades, empresas y laboratorios de investigación gubernamentales. Se encarga de la creación, recolección y distribución de bases de datos, léxicos y otros recursos de voz y texto con fines de investigación y desarrollo.

Dentro del amplio catálogo ofrecido se ha seleccionado el conjunto de conversaciones telefónicas entre hablantes nativos definido como *CALLHOME* en tres idiomas diferentes que son el inglés, el mandarín y el español. Cada una de estas recopilaciones de voz se compone de 120 conversaciones clasificadas en tres categorías: entrenamiento, *train*;

evaluación, *evl*, y desarrollo, *dev*. Para el propósito de este estudio se hará uso del grupo de conversaciones de entrenamiento puesto que ofrece un mayor número de conversaciones telefónicas que las otras dos opciones, 80 elementos frente a 20 por cada alternativa. En adición, cada conversación grabada dura como máximo treinta minutos.

Todos los archivos de audio obtenidos de LDC están disponibles en el formato *NIST SPHERE*. Este tipo de fichero consiste en una cabecera autodescriptora, que suele tener una longitud de 1024 octetos, seguida por las muestras de voz en binario. La cabecera proporciona información útil, en texto plano legible, sobre los datos contenidos en el cuerpo tales como el número de muestras, la tasa de muestreo, la cantidad de canales, etc.

Así, en la propia página web del consorcio se puede conseguir el software que permite convertir los ficheros comprimidos en ficheros con el formato conveniente. Esta herramienta, denominada *"sph2pipe"*, permite convertir los ficheros de uno en uno con múltiples opciones.

De tal manera que la primera acción a realizar es descomprimir los archivos *sphere* y convertirlo en ficheros del tipo *pcm*, válidos para su posterior tratamiento. Esto es posible llevarlo a cabo mediante la ejecución de la anterior herramienta junto con la especificación de ciertas opciones en su llamada. Por defecto se considera que los ficheros de entrada contienen una cabecera tal cual se ha descrito con anterioridad, y que el fichero de audio de salida contendrá ambos canales, será de la misma duración y estará codificado al igual que el archivo de entrada, y, por último, el formato de salida es *wav*. Por lo tanto es imprescindible cambiar determinadas opciones para conseguir el archivo de salida deseado que no es otro que un fichero *pcm* lineal de 16 bits sin cabecera (*raw*) que contenga únicamente el primer canal de comunicación y de duración idéntica al de entrada. A modo de ejemplo se presenta una llamada genérica con las opciones pertinentes.

$$\boldsymbol{sph2pipe\ -c\ 1\ -p\ -f\ raw\ input.sph\ output.raw}$$

Para poder llevar a cabo estas operaciones de forma automatizada se ha desarrollado tres scripts, ***en_conversor.sh***, ***ma_conversor.sh*** y ***sp_conversor.sh***, que acceden a la carpeta *train* de cada idioma y convierte todos los ficheros que se hallan en su interior. En cuanto a la nomenclatura

de los ficheros queda como sigue: para los ficheros de entrada se usa el formato ***lang_num.sph*** y para los de salida, ***lang_num.sph.chunk.train.c1.16pcm.raw***, donde *lang* es una abreviatura del idioma de la conversación (*en*, *ma* o *sp*) y *num* es un número de cuatro dígitos que identifica la conversación. Además, *chunk* es un número que determina la porción de conversación y su posición dentro de la conversación completa.

Este último identificador es necesario porque para llevar a cabo la medida de la calidad mediante el código diseñado en *Matlab* se trocean las conversaciones originales en particiones de 3 minutos. Por lo tanto, para su posterior tratamiento es imprescindible identificar cada porción inequívocamente. La numeración seguida comienza en 0 y alcanza, como máximo, el valor 9. Esto es así ya que, si las conversaciones duran 30 minutos como máximo y se toman trozos de 3 minutos cada uno, se obtendrán como mucho 10 particiones.

### 4.2.3. Codificación de los archivos de audio

El siguiente aspecto a tratar es una de las piezas clave que motivan la realización de esta investigación: el códec de voz. Es por ello que se intenta escoger un amplio abanico de posibilidades entre aquellos más utilizados en la tecnología de VoIP. De tal manera que se estudiarán los códecs de banda estrecha presentados en la sección 2.4 de este documento.

Esta es una de las tareas que más tiempo requiere su ejecución. Esto se debe en su mayor parte a la cantidad de ficheros que se tratan y al amplio conjunto de códecs que se utilizan. Hay que tener en cuenta que el códec G.729 y todas las modalidades del códec AMR permiten el uso de VAD, por lo que se obtiene la cantidad de 21 códecs disponibles.

Con el objetivo de realizar todo este proceso de forma automatizada se han diseñado dos ficheros nombrados como ***ftype.sh*** y ***getftype.sh***. El primero de ello se encarga de codificar el archivo de entrada *pcm* con el códec de voz deseado y de situar el fichero *.ftype* generado en el directorio correcto. Una llamada genérica a este fichero sería como sigue:

***ftype.sh codec pcm_dir coded_dir ftype_dir input.raw***

El primer argumento se corresponde con el identificador del códec de voz a usar, la lista de códecs se presenta en la siguiente tabla. Posteriormente se especifican los directorios donde se ubican los ficheros de voz, los ficheros codificados y los ficheros de tramas, respectivamente. Por último, se indica el conjunto de ficheros de voz que deben ser codificados.

| **Códec** | **Identificador (con VAD)** | **Identificador (sin VAD)** |
|---|---|---|
| AMR 4,75 | 1 | 10 |
| AMR 5,15 | 2 | 11 |
| AMR 5,90 | 3 | 12 |
| AMR 6,70 | 4 | 13 |
| AMR 7,40 | 5 | 14 |
| AMR 7,95 | 6 | 15 |
| AMR 10,2 | 7 | 17 |
| AMR 12,2 | 8 | 18 |
| G.729 | 9 | 20 |
| iLBC 30 | 16 | - |
| iLBC 20 | 19 | - |
| G.711 | 21 | - |

**Tabla 12.** ***Lista de identificadores de los códecs de voz.***

El orden seguido para los identificadores de los códecs de voz reside en el régimen binario medio ofrecido por cada uno de ellos. Dicho régimen binario se calcula mediante la siguiente ecuación:

$$\overline{Rb} = \rho_{act} \frac{H+L_{act} \cdot N_{fpp}}{T_f \cdot N_{fpp}} + \rho_{sid} \frac{H+L_{sid}}{T_f} \quad (4.1)$$

Por un lado se distinguen parámetros intrínsecos del códec como son la longitud de las tramas de voz y silencio, $L_{act}$ y $L_{sid}$, respectivamente; el

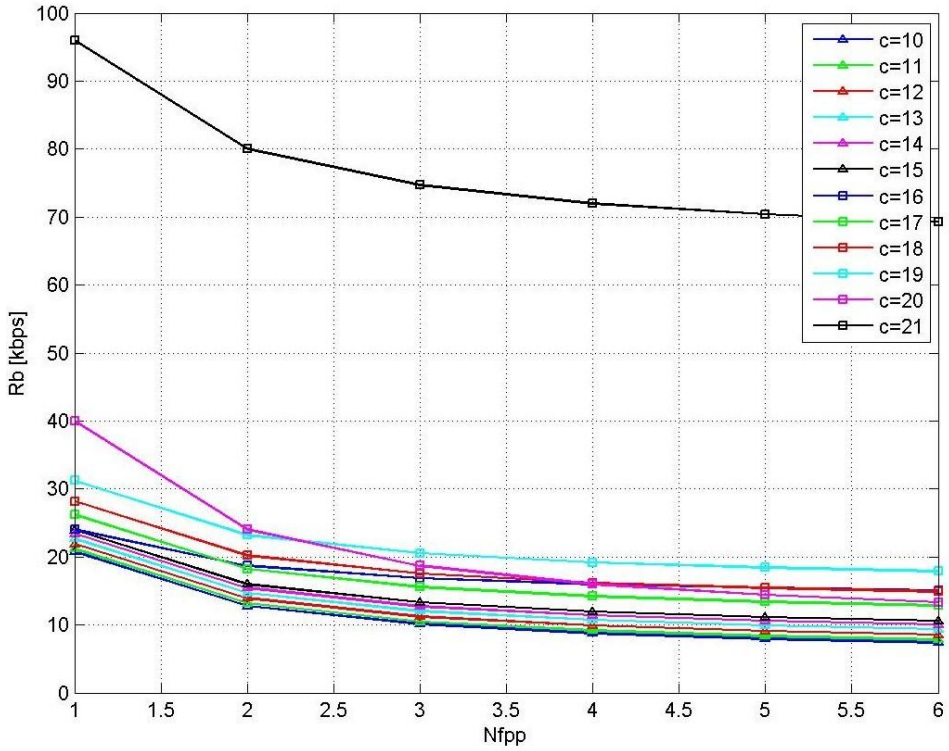
tiempo de codificación, $T_f$, y el número de tramas por paquete, $N_{fpp}$. Por otro lado, $H$ se corresponde con el tamaño de cabecera de cada paquete (40 octetos, pues sólo se considera hasta nivel de red). Por últimо, $\rho_{act}$ y $\rho_{sid}$ son los factores de actividad vocal y de silencio, respectivamente, calculados como el cociente entre el número de tramas activas o tramas SID, según corresponda, entre el número de tramas totales codificadas.

Así, para facilitar la obtención de estos factores de actividad, se ha desarrollado un sencillo script, ***bandWidth.sh***, encargado de leer los ficheros *.ftype* y calcular tanto dichos factores como el régimen binario. Comentar que los factores de actividad vocal y de silencio para códecs que no hagan uso de tramas de silencio son conocidos, pues toman los valores 1 y 0, respectivamente. Para el resto de casos se presenta en la siguiente tabla los distintos valores obtenidos.

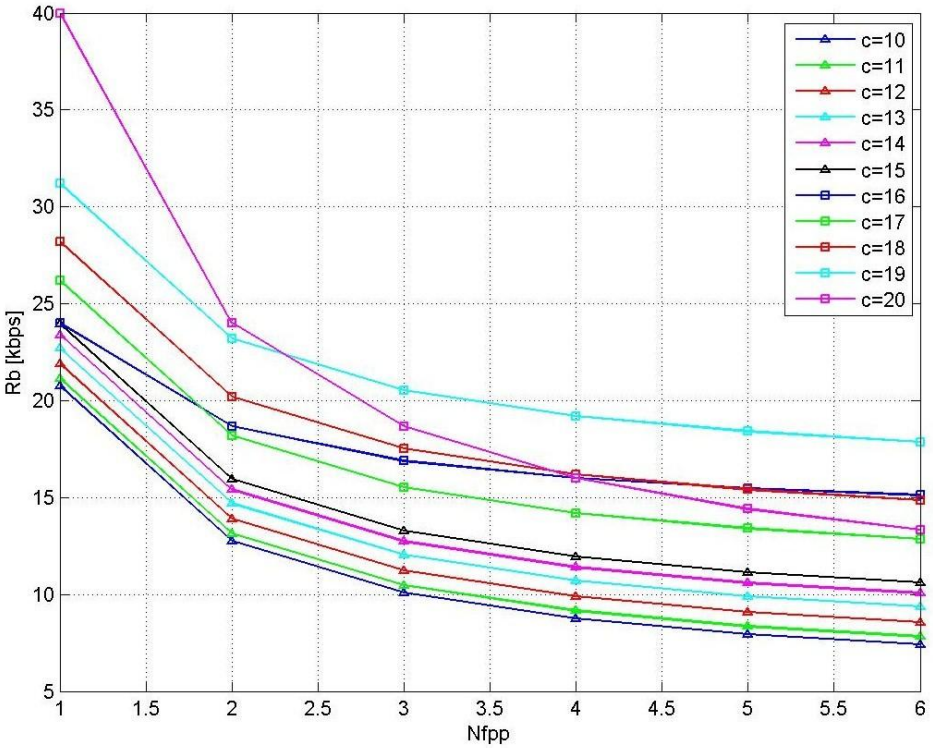
| **Códecs** | $\boldsymbol{\rho_{act}}$ | $\boldsymbol{\rho_{sid}}$ |
|---|---|---|
| AMR 4,75 VAD | 0,5725 | 0,0642 |
| AMR 5,15 VAD | 0,5671 | 0,0705 |
| AMR 5,90 VAD | 0,5721 | 0,0643 |
| AMR 6,70 VAD | 0,5721 | 0,0643 |
| AMR 7,40 VAD | 0,5721 | 0,0643 |
| AMR 7,95 VAD | 0,5721 | 0,0643 |
| AMR 10,2 VAD | 0,5727 | 0,0642 |
| AMR 12,2 VAD | 0,5730 | 0,0642 |
| G.729 VAD | 0,5794 | 0,0844 |

**Tabla 13.** ***Factores de actividad vocal y de silencio.***

A partir de todos estos datos se han generado las gráficas siguientes donde se muestra el valor del régimen binario alcanzado para cada uno de los códecs estudiados en base al número de tramas que se empaquetan en un mismo paquete.




***a)***




***b)***

**Figura 14.** ***a)Régimen binario medio para códecs sin VAD. b)Detalle.***

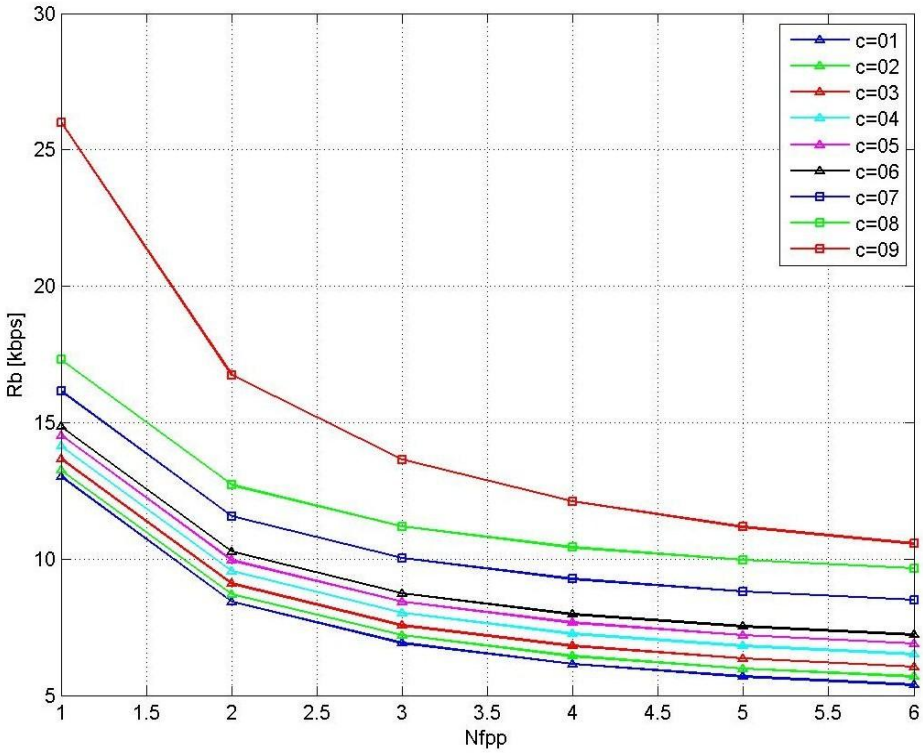



**Figura 15.** ***Régimen binario medio para códecs con VAD.***

El segundo de los ficheros permite realizar la codificación de los ficheros de voz por lotes. Es decir, permite mediante un conjunto de opciones definir el códec, el idioma de la conversación y el tipo de conversación, pero también otorga la posibilidad de realizar todo el conjunto en una única acción.

Los archivos codificados sólo tienen sentido para obtener la calidad del audio mediante el método de evaluación PESQ. Esto se debe a que, como se ha explicado, dicho modelo necesita además de la señal de referencia la señal decodificada sobre la cual medir la calidad. La nomenclatura que se utiliza para el nombrado de los ficheros es idéntica a la explicada anteriormente a excepción de la extensión del mismo, en este caso es ***.cod.códec***, donde *códec* es el nombre del codificador de audio.

Por otro lado, los ficheros más importantes son los ficheros de tramas denominados *.ftype*. En ellos se almacena la información de cada una de la tramas concerniente al tipo al que pertenecen. El nombre por el que quedan

designados posee el identificador de conversación de cuatro dígitos, *num*, y de porción de un solo dígito, *chunk*, derivado de los archivos *pcm* originales: ***numchunk.ftype***. No es necesario indicar información adicional puesto que están organizados en las distintas carpetas para tal efecto dentro del directorio *typedFiles*.

Mencionar adicionalmente que el hecho de denotar una porción de fichero mediante el indicador *chunk* sólo será de utilidad siempre que los ficheros *.ftype* sean particionados antes de la simulación, de tal manera que sea posible reconocer a qué parte de la conversación se corresponde dicha partición. Sin embargo, para la segunda aproximación no es necesaria tal información pues la partición se lleva a cabo dentro de la propia simulación, haciendo que carezca de sentido el mencionado identificador y es por ello que siempre se encuentra a cero.

### 4.2.4. Multiplexación de conversaciones

Esta tarea puede considerarse el núcleo de las simulaciones sobre la cuales se apoya el conjunto de resultados obtenidos. Es por ello que ha sido necesario en su desarrollo plantear distintas soluciones y analizarlas para determinar cuál de entre las posibilidades permite obtener unos resultados competentes en un tiempo permisible.

Así, en primer lugar se opta por la implementación de una aproximación diseñada en *Matlab*. Sin embargo, por motivos que se detallan posteriormente y atendiendo a las premisas anteriores, tuvo que desarrollarse un nuevo modelo de multiplexión para obtener el grueso de resultados buscado.

#### 4.2.4.1. Modelo de multiplexión en *Matlab*

En un primer momento, para llevar a cabo la multiplexion de los archivos de audio se piensa en utilizar el lenguaje de alto nivel para programación numérica *Matlab*.

Este lenguaje posee una clara ventaja con respecto a otras opciones, y es la capacidad para el manejo y tratamiento de datos vectorialmente. Esto reporta un considerable beneficio en la carga, operación y análisis de un gran volumen de datos como es el caso que nos atañe.

El modelo llevado a cabo se divide en dos componentes recogidas en sendos ficheros *.m*. El primero de ellos se denomina ***eventlist.m*** debido a que se encarga de realizar una lista ordenada de eventos que contiene los paquetes que posteriormente deben ser multiplexados. Esta segunda tarea se implementa en el fichero ***mux.m*** que constituye el verdadero multiplexor.

Así, el funcionamiento básico de ***eventlist.m*** se basa en cuatro subtareas fundamentales: carga de los ficheros de tramas, tratamiento de los ficheros de tramas, creación de la lista de eventos y ordenación de los eventos.

La carga de ficheros consiste en volcar en memoria el contenido de los ficheros especificados mediante argumentos de entrada a la función. Los ficheros son identificados por el directorio al que pertenecen y se vuelcan en memoria para que su acceso sea más rápido. El tratamiento de los mismos consiste en identificar cada una de las tramas que han sido cargadas con una numeración que permita relacionarla con la conversación a la que pertenece y su posición original. Además, se eligen aleatoriamente las conversaciones en función del número de las mismas deseado.

En la creación de la lista de eventos interviene el análisis de cada trama para asignarle el tiempo de llegada al multiplexor, el tamaño que ocupa y el número de ellas que viajan en el mismo paquete. Por último, se lleva a cabo la ordenación temporal de la lista de eventos creada.

Una vez terminado todo este proceso, se invoca al fichero ***mux.m*** para llevar a cabo la simulación del multiplexor. Dicho elemento posee un funcionamiento simple, ilustrado en la figura 16, según el cual si la llegada de un paquete ocurre cuando el multiplexor está ocioso o existe capacidad de almacenamiento en el buffer de entrada, dicho paquete se cursa y se actualiza el estado del multiplexor; en caso contrario, el paquete se considera perdido. En caso de que el paquete se almacene temporalmente en el buffer, se guarda la información sobre el retraso que sufre.

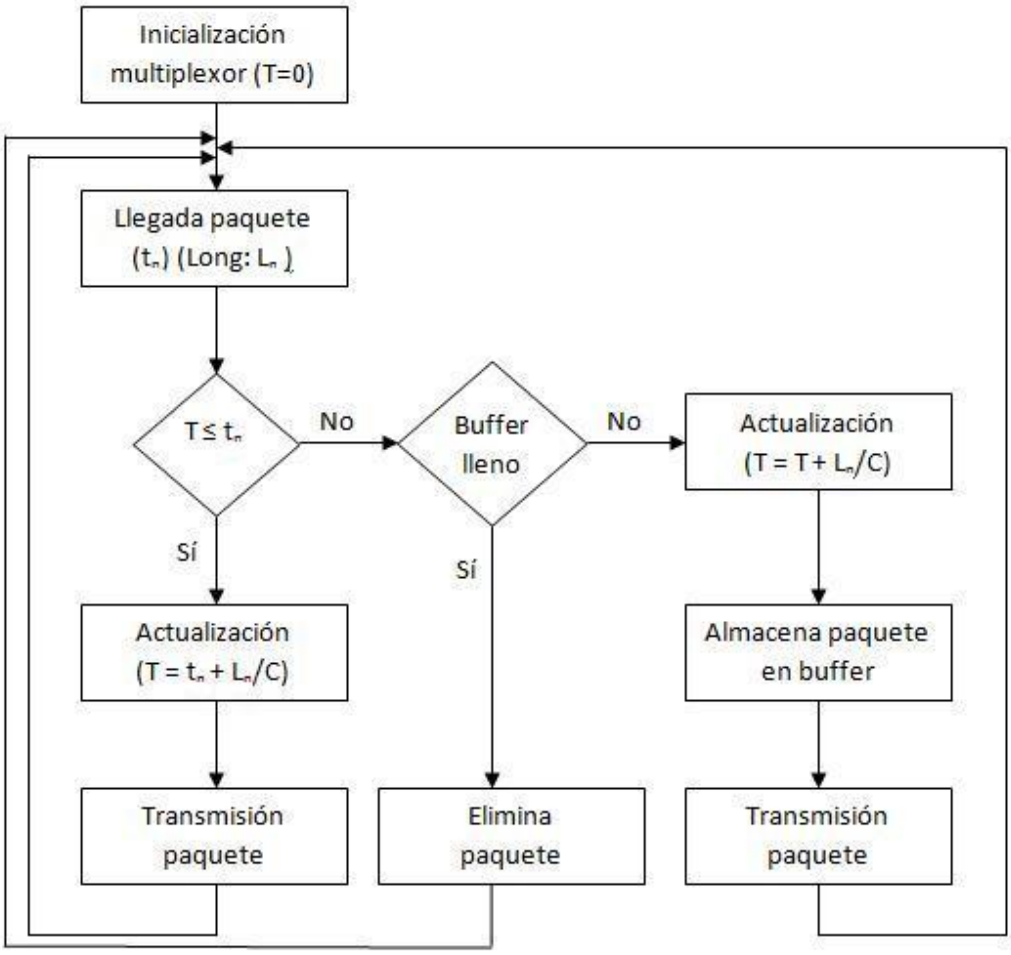



**Figura 16.** ***Funcionamiento multiplexor.***

No obstante, aunque el funcionamiento descrito es muy básico, presenta un grave problema. Este problema reside en el hecho de que es necesario recoger la información sobre todas las tramas perdidas y presentarla de forma coherente y útil para su posterior tratamiento automatizado. A pesar de manejar los datos vectorialmente, es tal el volumen que el tiempo de simulación empleado rebasa los límites permisibles.

A modo de justificar este hecho, se presenta una tabla con los tiempos empleados en generar diez simulaciones donde se varía el parámetro *R* para obtener distintas pérdidas de paquetes.

| **Parámetros fijos** | | | | | |
|---|---|---|---|---|---|
| *K* = 10 | *N* = 10 | *Tc* = 3 | *Nfpp* = 2 | *Lang* = en | *Códec* = G.729 VAD |
| | | | | | |
| ***R*** | **Probabilidad de pérdida de paquetes** | | **Retraso medio (s)** | **Tiempo de simulación (s)** | |
| 0,2 | 80 % | | 0,0984 | 404,4696 | |
| 0,3 | 70 % | | 0,0653 | 345,2692 | |
| 0,4 | 60 % | | 0,0486 | 311,5550 | |
| 0,5 | 50 % | | 0,0385 | 271,1109 | |
| 0,6 | 40 % | | 0,0317 | 229,3913 | |
| 0,7 | 30 % | | 0,0272 | 189,7081 | |
| 0,8 | 20 % | | 0,0232 | 154,7645 | |
| 0,9 | 10 % | | 0,0196 | 117,7002 | |

**Tabla 14.** ***Tiempo de simulación en Matlab.***

Es interesante observar cómo disminuye el tiempo de simulación conforme la pérdida de paquetes también lo hace. Esto refuerza la idea comentada anteriormente de que el hecho de almacenar la información de las tramas perdidas hace que el tiempo de simulación aumente considerablemente.

Finalmente, existe una última tarea que lleva a cabo el primero de los scripts. Dicha tarea consiste en recoger los valores devueltos por el multiplexor simulado y presentarlos en dos tipos de ficheros. Ambos ficheros comparten la misma nomenclatura a excepción del identificador *type*, el cual puede tomar el valor 0, contiene el retardo medio y el ratio de pérdidas de los paquetes; ó 1, contiene las tramas perdidas por cada conversación multiplexada. Así, en el propio nombre del fichero quedan recogidos todos los parámetros de simulación de la siguiente forma, ***KRNTcNfpp.lang.códec.type.txt***.

#### 4.2.4.2. Modelo de multiplexión en *C++*

Con el objetivo de reducir drásticamente los tiempos anteriores, se opta por una solución programada en lenguaje compilado *C++*. De manera que esta opción se compone de diversos ficheros fuente con sus respectivos ficheros de cabeceras que logran dedicarse cada uno a su tarea correspondiente. Permiten realizar un procedimiento parecido al anterior con la salvedad de que no se guarda información sobre las tramas perdidas sino que sólo se recogen ciertos estadísticos. Gracias a esto se consigue reducir aún más los tiempos de simulación. Sin embargo, nos obliga a evaluar la calidad mediante un modelo paramétrico, como se explica en el siguiente epígrafe.

El conjunto de parámetros de simulación se tratan mediante los ficheros ***options.cpp*** y ***options.h***. Permiten recoger los valores de los distintos parámetros, el nombre de los ficheros *ftype*, aplicar los valores por defecto en caso de que no se especifiquen y verificar que se hallan dentro del rango válido. Una vez conocido los ficheros que componen la batería de conversaciones codificadas de entrada al multiplexor, se procede a cargar el contenido de dichos ficheros, dividir su contenido según la duración de las conversaciones indicadas en las opciones, elegir aleatoriamente el número de conversaciones deseadas, acceder a su contenido para clasificar y empaquetar las tramas en función de su tipo y crear una lista de eventos ordenada temporalmente. Estas operaciones están definidas y se llevan a cabo en ***ftype.cpp*** y ***ftype.h***.

A continuación se procede al núcleo del multiplexor, cuyo funcionamiento es el reflejado en la figura 16, implementado en ***voipmux.cpp*** y ***voipmux.h***, donde se ejecuta la multiplexión de la lista de eventos y se contabilizan los paquetes perdidos, el retraso que sufren y las ráfagas de pérdidas. Gracias a estos datos recolectados es posible efectuar, en el mismo fichero, la evaluación de la calidad tal como se detalla en el siguiente epígrafe. Hay que tener en cuenta, a diferencia del modelo de multiplexión anterior, que cada vez que se realiza una llamada se ejecuta una serie de simulaciones donde se varía los valores de tamaño de buffer y capacidad de salida en el intervalo definido y se repiten un número

significativo de veces. Aún así, no se superan los 300 segundos en el peor de los casos. Además, se dispone del fichero ***stats.cpp*** que proporciona como valores de retorno los estadísticos media y desviación típica del conjunto de resultados obtenidos tras las simulaciones.

Por último, en el fichero ***main.cpp*** se produce la llamada secuencial a cada una de las funciones y presenta los estadísticos de los valores resultantes en un fichero que sigue la nomenclatura ***NTcNfpp.lang.códec.txt***. Este fichero contiene el valor de R y K seguido de la media y la desviación típica de la pérdida de paquetes totales, de paquetes SID y de paquetes de voz en tanto por ciento; el retardo de los paquetes en segundos; la calidad en escala MOS y el valor de BurstR.

Comentar que los rangos de los parámetros de simulación se mantienen según la tabla 11, a excepción del factor de reducción de la capacidad de salida R. El rango de validez va a ser dependiente del códec utilizado pues está intrínsecamente relacionado el valor de R con la pérdida de paquetes y, consecuentemente, con la calidad de la conversación. De tal manera que para obtener unos resultados plausibles y relevantes se ha optado por estimarlo de la siguiente manera.

Para los códecs que no hagan uso de VAD se ha establecido el rango de valores entre $(100 - Bpl)$ y el 100 por ciento de la capacidad total demandada en la entrada del multiplexor. El parámetro Bpl se define como la robustez que presenta un códec frente a la pérdida de paquetes y se tiene en cuenta en la evaluación de la calidad. En cambio, para códecs con VAD el límite inferior viene dado por la expresión $(\rho_{act} \cdot 100) - Bpl$, mientras que el superior es idéntico. Como ya he es sabido, $\rho_{act}$ es el factor de actividad vocal cuyo valor se especifica en la tabla 13 para cada uno de los códecs.

No obstante, con el objetivo de homogeneizar los resultados, se redondean los límites inferiores a números enteros múltiplos de diez. Además, el incremento para códecs sin VAD es del 5%, pues el rango de valores es más amplio, y del 2% para el resto de códecs, para obtener un conjunto de resultado mayor.

La causa de que se hayan elegido estos rangos reside en el comportamiento de las fuentes de tráfico. Para el caso en el que no se

disponga de VAD, las fuentes se comportan como fuentes de tipo CBR, es decir fuentes generadoras de tráfico a una tasa de bit constante. El hecho de que su comportamiento sea de este tipo supone que a la salida del multiplexor se pierden un ratio de paquetes proporcional a la diferencia entre las capacidades de entrada y salida de los enlaces del multiplexor. De forma tal que si se impone una capacidad de salida del 80% respecto de la entrada (R=80), en media se perderán aproximadamente el 20% de los paquetes. Para R=90, se pierden el 10%, y así sucesivamente. Por el contrario, para el caso de VAD activo la fuente se considera VBR, tasa de bit variable. Entonces el ancho de banda consumido no es el mismo para fragmentos de voz que de silencio, pues se persigue ahorrar su consumo. Es por ello que se busca conocer el factor de actividad vocal.

De cara a la calidad, y como se verá reflejado en las ecuaciones, se obtienen resultados sensatos siempre y cuando la pérdida de paquetes sea igual o inferior al parámetro de robustez Bpl. Por ello modula el intervalo de manera que se ajusten los datos correctamente.

### 4.2.5. Evaluación de la calidad

Al igual que se ha diseñado dos modelos de simulación, también se hace uso de dos métodos de evaluación de la calidad que se ajustan al formato de los resultados proporcionados por cada uno de ellos.

En el primero de los casos, se utiliza un modelo comparativo, PESQ, que tiene únicamente en cuenta las tramas pérdidas. Motivo por el cual este escenario de simulación es más complejo pues requiere analizar y salvar información sobre cada una de las tramas multiplexadas. En contraposición, no valora el efecto del retardo de los paquetes y se considera que queda solventado con el buffer del receptor.

En el segundo de los modelos, se prefiere un método paramétrico, conocido como modelo E, pues ya no se dispone de la información de tramas pertinente y por tanto sólo podemos evaluar la calidad a través de los estadísticos obtenidos.

#### 4.2.5.1. PESQ

El modelo recogido en la norma ITU-T P.862 [31] propone un método para la evaluación de la calidad vocal basado en la comparación entre una señal de referencia y la señal multiplexada. Es por ello que este modelo se aplica a los resultados obtenidos mediante el escenario de simulación desarrollado sobre *Matlab*.

Pero antes de poder obtener la evaluación de la calidad se necesita decodificar los ficheros de las conversaciones multiplexadas teniendo en cuenta las tramas perdidas. Para tal efecto se dispone de un sencillo programa en *C*, ***eraser.c***, que toma los ficheros de tipo 1 generado por el simulador en *Matlab* y genera un fichero para cada conversación multiplexada con su identificador.

Una vez se dispone de los ficheros de tramas perdidas se procede a decodificar las conversaciones implicadas y llamar al programa ***p862*** que implementa el modelo PESQ para cada conversación. Se finaliza haciendo la media para todas las conversaciones y presentando los resultados en dos tipos de ficheros distintos: el primero contiene los resultados separados por batería de simulación, cuya extensión es ***.data***, y el segundo recopila todos los resultados, ***results.data***.

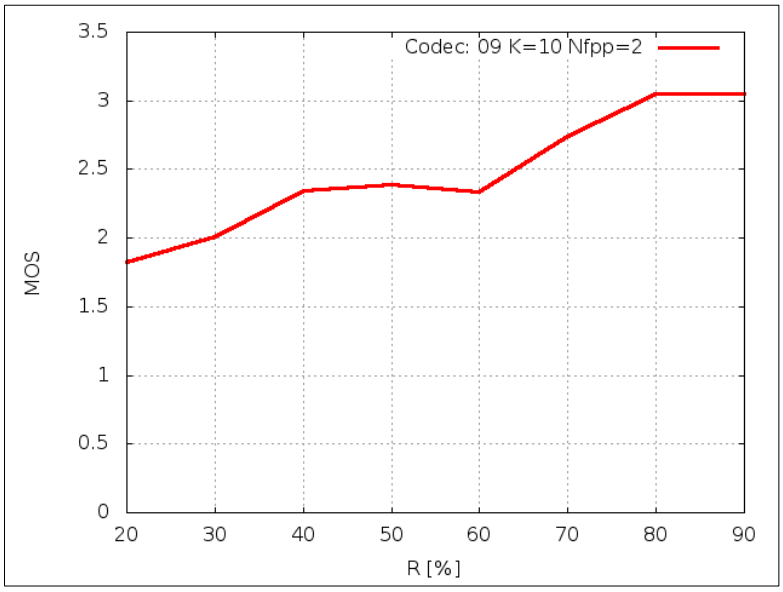
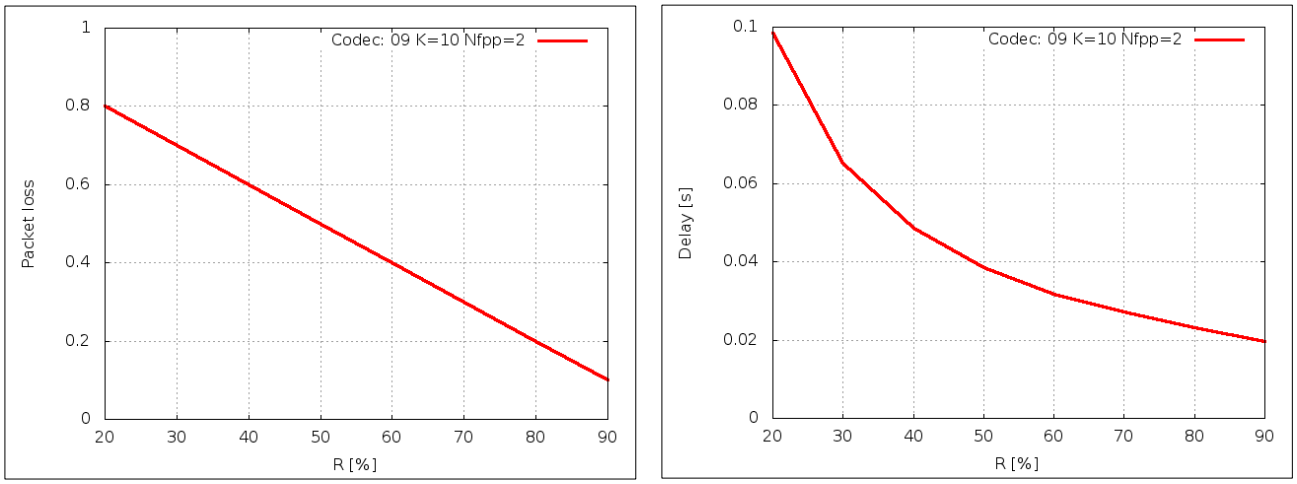



**Figura 17.** ***Calidad medida tras la multiplexión.***

Todo este conjunto de tareas se mecanizan en el script ***PESQresult.sh*** y los resultados se almacenan en el directorio *PESQ* dentro de *results*. Por último se lleva a cabo la representación gráfica obteniéndose las figuras adjuntas.




***a)*** ***b)***

**Figura 18. a)** ***Pérdida de paquetes. b) Retraso medio de paquetes.***

### 4.2.5.2. Modelo E

La ITU-T provee de una opción alternativa al modelo anterior para la medida de la calidad del audio mediante la recomendación G.107 [28], comúnmente conocida como modelo E. La principal ventaja que otorga este método de evaluación de la calidad es la posibilidad de obtener resultados equivalentes con una menor capacidad de computación y volumen de datos.

Así, gracias a esta nueva propuesta es viable conseguir los valores de medición de calidad perseguidos a partir, únicamente, de los estadísticos de las medias de paquetes perdidos y su retraso. Para tal efecto se dispone de las ecuaciones que conforman el modelo y que se detallan a continuación.

La ecuación principal determina el factor $R$, que es la puntuación de calidad resultante de la aplicación del modelo a los datos de entrada. Posteriormente, dicha puntuación es traducible a una escala MOS. El factor $R$ está constituido por $R_0$, representa la señal a ruido básica que incluye fuentes de ruido tales como ruido de circuito y ruido de ambiente; $I_d$ representa las degradaciones producidas por el retardo, y el factor de degradación efectiva del equipo $I_{e\text{-}eff}$ incluye la degradación debida a pérdidas de paquetes.

$$R = R_0 - I_d - I_{e\text{-}eff} \tag{4.2}$$

El término $R_0$ se considera constante, cuyo valor se ha fijado a 93,2. Esto implica que, aunque las degradaciones fuesen nulas, el máximo valor alcanzable por el factor $R$ es igual a dicha constante. En cambio, los valores de $I_d$ e $I_{e\text{-}eff}$ se subdividen en parámetros más específicos. El factor de degradación por retardo posee una ecuación simplificada que se ajusta acordemente al propósito perseguido. Esta expresión se basa en evaluar el retraso, $d$, que sufren los paquetes enviados a través del multiplexor, expresados en milisegundos.

$$I_d = (0{,}024 \cdot d) + 0{,}11 \cdot (d - 177{,}3) \cdot H(d - 177{,}3) \tag{4.3}$$

Donde $H$ es la función de Heavyside tal que $H(x) = 0$ para $x < 0$ y 1 para $x \geq 0$.

El factor de degradación efectiva depende de valores tabulados mediante pruebas subjetivas para intentar aproximarse al valor real de la forma más precisa posible. Entre los valores tabulados se hallan el factor de degradación del equipo para pérdida de paquetes nula $I_e$ y el factor de robustez contra pérdida de paquetes $Bpl$. Así, eligiendo los valores definidos para cada códec [29], y recogidos en la tabla 15, junto con la pérdida de paquetes $Ppl$ se obtiene $I_{e\text{-}eff}$ mediante la siguiente ecuación.

$$I_{e\text{-}eff} = I_e + (95 - I_e) \cdot \frac{Ppl}{\frac{Ppl}{BurstR} + Bpl} \tag{4.4}$$

| Códec | $I_e$ | $Bpl$ |
|---|---|---|
| AMR 4,75 | 29 | 10 |
| AMR 5,15 | 27 | 10 |
| AMR 5,90 | 23 | 10 |
| AMR 6,70 | 20 | 10 |
| AMR 7,40 | 16 | 10 |
| AMR 7,95 | 15 | 10 |
| AMR 10,2 | 9 | 10 |
| AMR 12,2 | 5 | 10 |
| iLBC | 4 | 32 |
| G.729 | 11 | 19 |
| G.711 | 0 | 25,1 |

**Tabla 15.** ***Valores de Ie y Bpl para los distintos códecs.***

$BurstR$ es el *burst ratio* y es la última modificación añadida por la ITU-T para optimizar los resultados en base al comportamiento presentado por las pérdidas de paquetes. Relaciona la longitud de las ráfagas de pérdida de paquetes con la longitud de las ráfagas ante pérdidas aleatorias. Es por ello que siempre su valor es superior a la unidad, salvo cuando las pérdidas presentan una distribución aleatoria que entonces el resultado es uno.

Para medirlo de forma eficaz se define el inverso del tamaño medio de las ráfagas de paquetes perdidos $B$ como el número de paquetes de voz perdidos entre el número de ráfagas de pérdidas modulado por el número de tramas por paquete.

$$B = \frac{N_{pl} \cdot N_{fpp}}{N_{burst}} \tag{4.5}$$

Así, finalmente el parámetro $BurstR$ queda expresado de la siguiente forma:

$$BurstR = max\left\{1, B \cdot \left(1 - \frac{Ppl}{100}\right)\right\} \tag{4.6}$$

El hecho de que tengamos que corroborar que siempre su valor esté por encima de la unidad se debe a que las simulaciones constituyen un

conjunto de experimentos de carácter determinista y es posible que se obtengan resultados por debajo de uno. El segundo término se corresponde con la propia expresión proporcionada por la recomendación para distribuciones de pérdidas de paquetes en cadena de Markov de dos estados.

Así una vez se dispone del valor del factor $R$ sólo resta traducir dicho valor a una escala de valoración subjetiva como es la MOS. Para tal efecto se dispone de la siguiente ecuación de conversión.

$$MOS = \begin{cases} 1 & R < 0 \\ 1 + 0{,}035 \cdot R + 7 \cdot R \cdot (R - 60) \cdot (100 - R) \cdot 10^{-6} & 0 \leq R \leq 100 \\ 4{,}5 & R > 100 \end{cases}$$

A la vista de la expresión anterior a lo sumo es posible obtener una puntuación MOS de 4,5. No obstante, si consideramos que el factor $R$ no puede superar el valor de 93,2 entonces al máximo valor alcanzable en la escala MOS es de 4,4093.

## 4.3. Resultados: colección de estadísticos

A partir de este punto sólo se hace referencia al segundo modelo, pues la aproximación basada en *Matlab* se hace inviable para recopilar los estadísticos pertenecientes a todos y cada uno de los escenarios de simulación llevados a cabo.

En las secciones anteriores se ha explicado cómo se desarrolla la multiplexión de las conversaciones y el método para evaluar la calidad en base a los estadísticos del retardo y la pérdida de paquetes. Sin embargo, es imprescindible disponer de un script que lleve a cabo la ejecución de la batería de simulaciones de forma ordenada y automática. Esta batería consiste en abarcar todo el rango de valores para los distintos parámetros considerados y presentar los resultados obtenidos organizados en ficheros

para su posterior análisis. Todo ello está contenido en el fichero ***runsim.sh***. Así, cada uno de los parámetros involucrados es pasado como argumento al fichero ***main.cpp***, encargado de llamar a las distintas funciones que permiten la multiplexión de las conversaciones.

El fichero ***runsim.sh*** está diseñado para recorrer el rango completo de validez de cada uno de los parámetros, tal cual se recoge en la tabla 11. Sin embargo, a modo de presentar resultados ejemplificativos, no se hace uso de los rangos completos, sino que por el contrario, se efectúa un acotamiento de los mimos. Esta reducción se traduce en los intervalos presentados en la tabla 16.

La metodología de trabajo expuesta, no obstante, es extensible a la totalidad del rango presentado originalmente.

| Parámetro | Descripción | Unidad | Valores |
|---|---|---|---|
| N | Número de llamadas cursadas simultáneamente | Llamadas | 2 – 80 |
| Tc | Duración de cada conversación | Minutos | 3 |
| K | Tamaño de la cola de entrada del multiplexor por llamada cursada | Paquetes | 1 |
| R | Factor de reducción del caudal de salida con respecto al de entrada | Tanto por ciento | 40 – 100 |
| Nfpp | Número de tramas por paquete | Unidades | 1 – 6 |
| Idioma | Idioma de los interlocutores | - | Inglés |
| Códec | Códec vocal usado para la codificación y compresión de la voz | - | G.729 VAD, AMR VAD |
| Iteraciones | Número de repeticiones para un mismo escenario | - | 50 |

**Tabla 16.** ***Rango efectivo de los parámetros de simulación.***

Una vez realizada la batería de simulación, se disponen de los ficheros de resultados devueltos por el ***main.cpp*** con el formato y contenido explicado en la sección pertinente. No obstante, la organización del contenido no es adecuada para realizar una representación gráfica del mismo. Es por ello que se diseña ***plotData.sh***. Se ocupa de recopilar toda la información disponible, acorde con los rangos de valores acotados, y organizarlas en ficheros designados por la nomenclatura ***nr.códec.K.Nfpp.data***. *códec* hace referencia al dígito de dos cifras que designa el códec usado, *K* es el tamaño del buffer del multiplexor, en esta ocasión siempre toma el valor 1; y *Nfpp* es el número de tramas por paquete. En esta nueva nomenclatura se han obviado algunos parámetros por considerar que no varían en el caso presentado. En caso de perseguir un nombre único e inconfundible sería necesario añadir el idioma y la duración de la conversación.

Los ficheros *.data* generados son almacenados en el directorio *data* contenido en la carpeta *results*. Su organización interna consiste en cinco columnas de datos. Las dos primeras hacen referencia a todas las posibles combinaciones de valores entre el número de conversaciones, *N*, y el factor de reducción del caudal, *R*. Las tres siguientes se corresponden con el ratio de pérdida de paquetes, el retraso medio sufrido y la media de la puntación MOS obtenida, respectivamente. A partir de estos ficheros sí es posible representar los resultados de forma gráfica y, por tanto, proceder a su análisis.

De manera representativa se ofrece a continuación las superficies de los valores medios para la calidad, el retraso y la pérdida de paquetes para el códec de audio G.729 VAD y una trama por paquete. En tales representaciones puede observarse como la calidad de las conversaciones, figura 19, varía inversamente proporcional a la variación del retardo y de las pérdidas, figuras 20 y 21; como cabía esperarse.

Destacar que los resultados mostrados representan la media de los valores obtenido tras repetir la simulación 50 iteraciones para cada caso de uso. También se estudian los valores de la desviación estándar, aunque no se incluyen porque no aportan información significativa. Para valores de *N* pequeños se tiene un 8% de desviación frente a una media de 40% de pérdidas de paquetes; 2ms para una media de 18 ms en retardo, y 0,1 para

una media de 1,4 de calidad en escala MOS. Conforme el valor de *N* aumenta, la desviación tiende a hacerse nula.

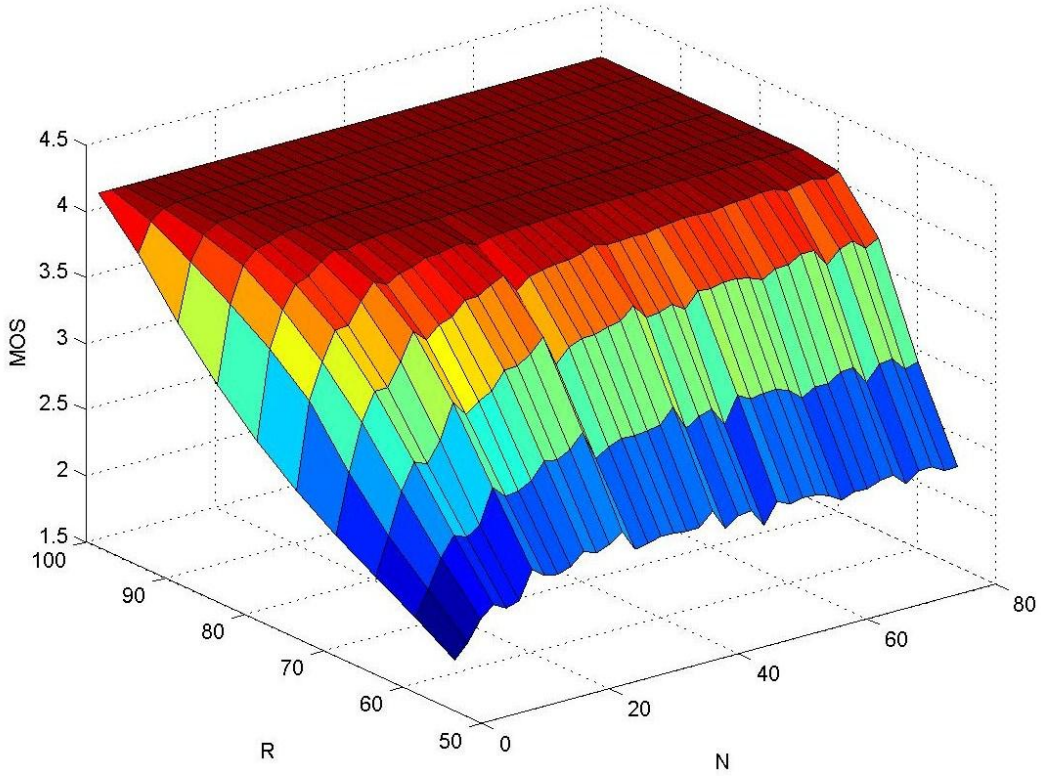



**Figura 19.** ***Calidad códec G.729 VAD (Nfpp=1).***

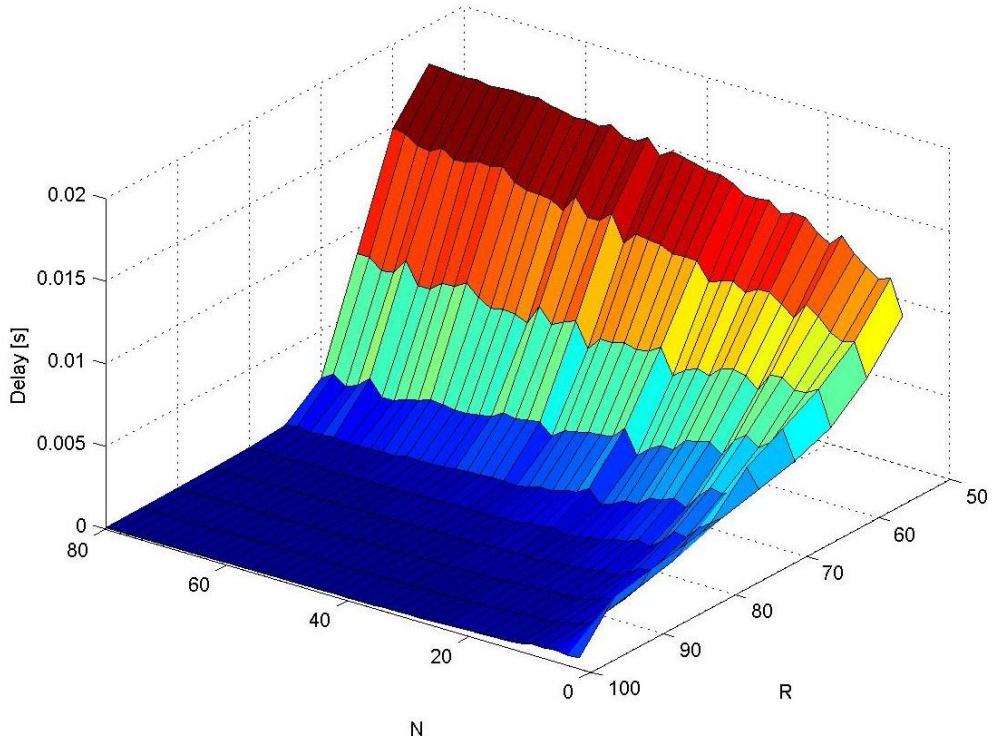



**Figura 20.** ***Retraso códec G.729 VAD (Nfpp=1).***

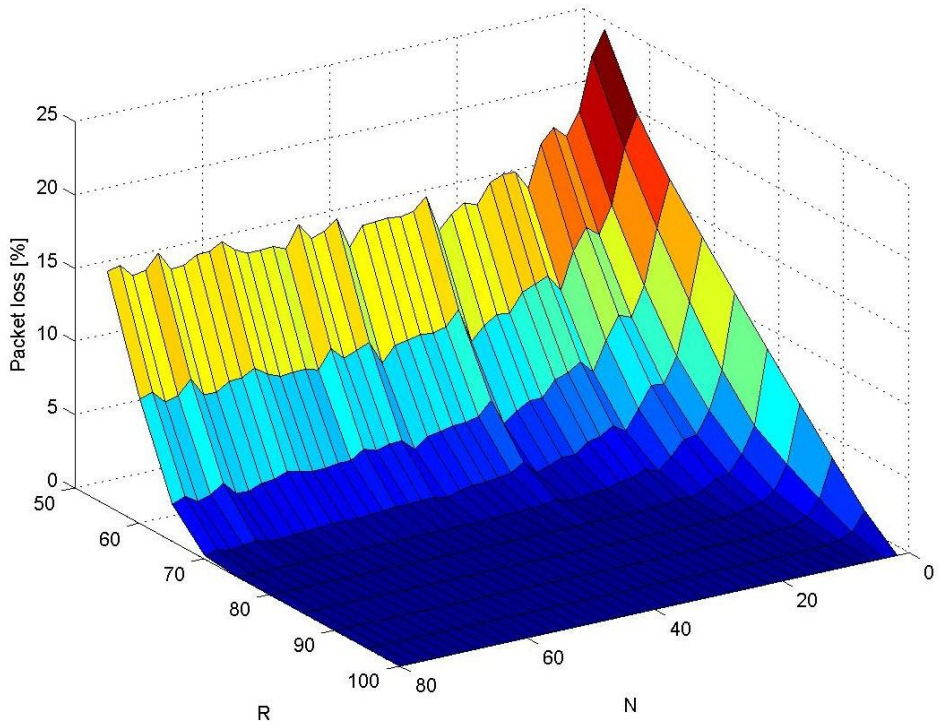



**Figura 21.** ***Pérdidas códec G.729 VAD (Nfpp=1).***

En todas las representaciones gráficas se observa un fenómeno de rizado. Este efecto parece estar íntimamente relacionado con el número de conversaciones multiplexadas, *N*. Sin embargo, el origen que lo desencadena no está muy claro. Una posible explicación puede ser la dependencia existente entre el tamaño de la cola de salida $K_{out}$ y el mencionado parámetro *N*. Esta dependencia reside en la manera de computar dicha cola en el algoritmo de simulación: $K_{out} = N \cdot K$, donde *K* es el tamaño del buffer de entrada al multiplexor para cada conversación.

A continuación, se añaden tres ilustraciones, figuras 22-24, correspondientes a la calidad obtenida para la codificación mediante G.729 VAD y para AMR 10,2 VAD. En vista a la similitud encontrada en la forma de las superficies, se justifica la búsqueda de una expresión analítica con una forma común para todo el paradigma estudiado.

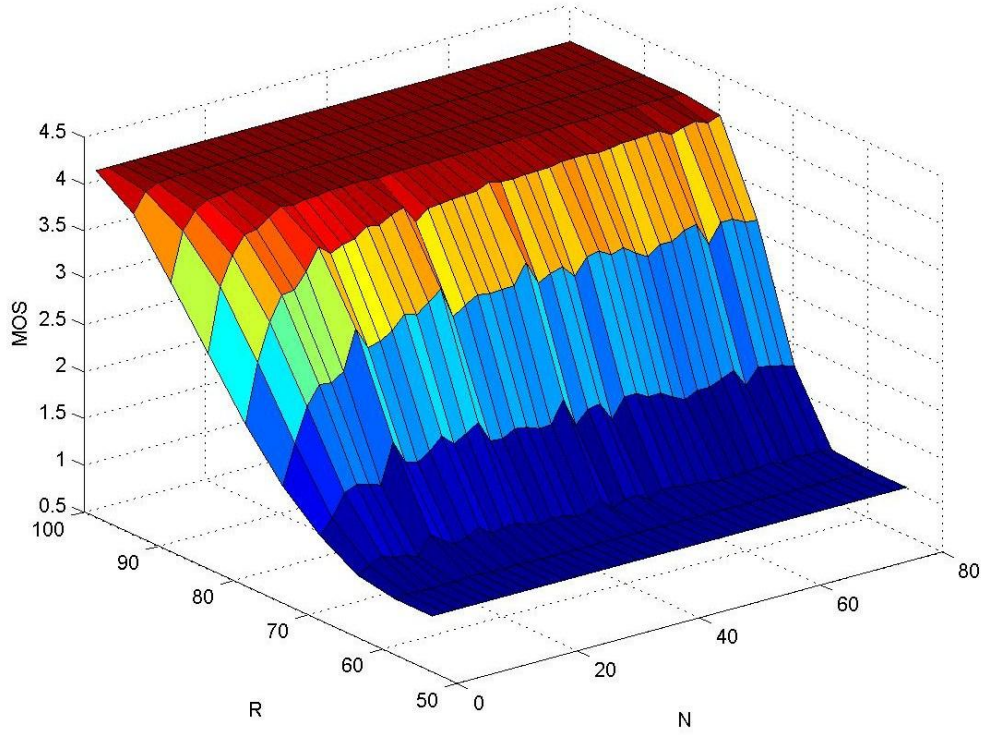



***Figura 22. Calidad códec G.729 VAD (Nfpp=6).***

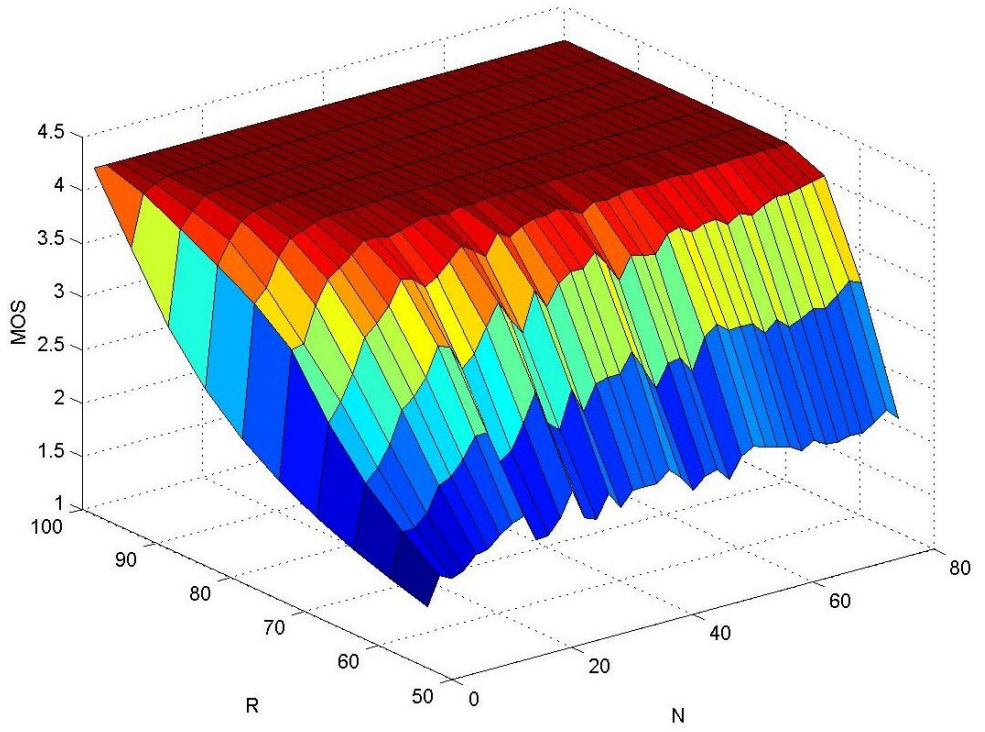



**Figura 23.** ***Calidad códec AMR 10,2 VAD (Nfpp=1).***

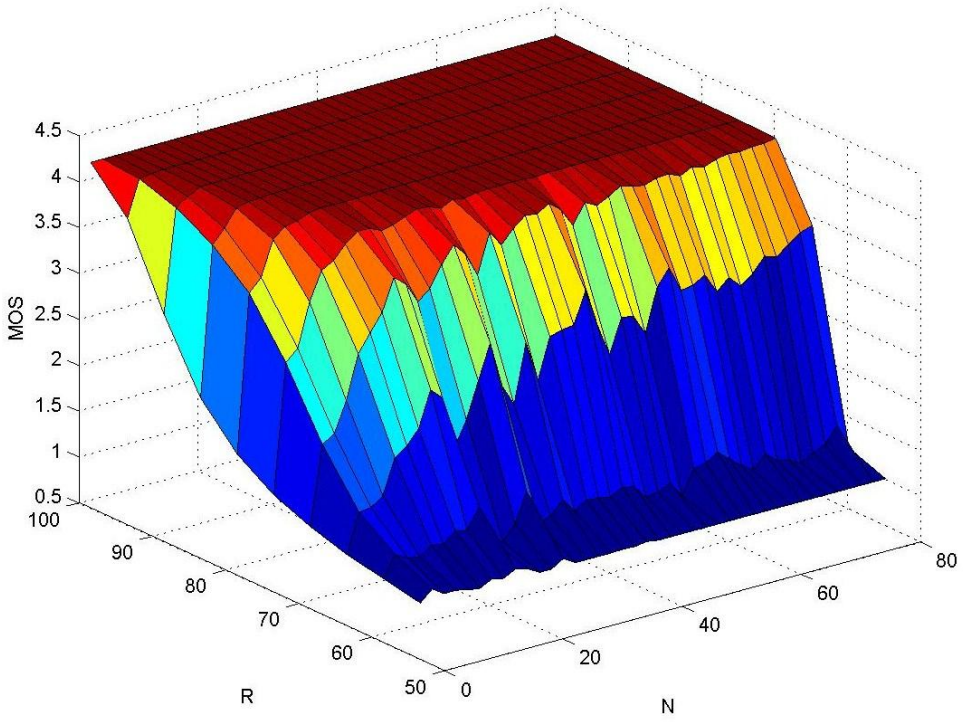



**Figura 24.** ***Calidad códec AMR 10,2 VAD (Nfpp=6).***