Implementación de un reconocedor distribuido de voz robusto para PDAs

José A. González, Ángel M. Gómez, Antonio M. Peinado, José L. Pérez-Córdoba, Victoria E. Sánchez, José L. Carmona *

Dept. de Teoría de la Señal, Telemática y Comunicaciones

E.T.S. Ingeniería de Informática y Telecomunicación, Universidad de Granada

c/Daniel Saucedo Aranda, s/n, 18071 Granada

{joseangl,jlpc,amp,amgg,victoria,maqueda}@ugr.es

12 de junio de 2007

Resumen

Este trabajo presenta una implementación de un sistema de reconocimiento distribuido del habla en tiempo real para dispositivos móviles del tipo PDA. El resultado ha sido una aplicación cliente/servidor basada en el *front-end* avanzado desarrollado por ETSI (AFE) y el reconocedor HTK. Se propone un algoritmo de ordenación de paquetes y un algoritmo de mitigación de errores, con el fin de robustecer la tarea de reconocimiento del habla frente a pérdidas y retrasos introducidos en la transmisión. El sistema implementado se ha probado con distintas trazas reales de Internet y la base de datos Aurora 2.

1. Introducción

El desarrollo de dispositivos portátiles de bajo coste con conexión a Internet, supone un cambio en el paradigma de interacción entre el usuario y los servicios a los que accede. Tradicionalmente este tipo de interacción se ha venido realizando a través de limitados periféricos (teclados pequeños, dispositivos apuntadores, joystick, ...), o a través de operadores humanos. En los últimos años, no obstante, se está apreciando una evolución del paradigma de interacción hasta dar lugar a lo que se ha venido a llamar interacción multimodal. En la interacción multimodal se da libertad al usuario para que pueda interaccionar con una misma aplicación utilizando diferentes modos de comunicación como teclado, ratón, stylus, voz, pantalla, etc.

Dado que la voz es el método más natural de comunicación para el hombre, el reconocimiento automático del habla (RAH) es especialmente adecuado para la interacción hombre-máquina y, en particular, para entornos móviles donde el tamaño y las funcionalidades de los dispositivos introducen serias restricciones a dicha interacción.

Existen dos posibles soluciones para la implementación de un sistema de reconocimiento del habla. La primera consiste en instalar el sistema de reconocimiento en el propio terminal. La segunda se basa en realizar el reconocimiento fuera del propio dispositivo, para ello el cliente tiene que enviar la voz (o una parametrización de la misma) a un servidor remoto, donde se realiza el reconocimiento. Esta segunda aproximación se conoce como reconocimiento remoto de voz (RSR, *Remote Speech Recognition*) y es muy apropiada para su uso en dispositivos móviles, donde la reducida capacidad de cómputo y/o memoria impiden la implantación de un sistema RAH como en la primera solución. Además, RSR tiene como ventaja adicional el permitir diseñar y mante-

*Este trabajo se ha realizado con la financiación de los proyectos MEC/FEDER TEC2004-03829/TCM y PROFIT/AVANZ@/FEDER FIT-330503-2006-2 (www.mciudad.org).

ner de forma separada tanto el cliente como el servidor, por lo que el cliente no se tiene que preocupar por el proceso de reconocimiento.

Si bien RSR tiene una serie de bondades respecto a otras soluciones de RAH, se le pueden achacar dos problemas que, o bien no aparecen, o lo hacen en menor medida en otras aplicaciones de RAH. El primer problema es la degradación en el reconocimiento producido por errores del canal de transmisión. El segundo es la degradación por ruido ambiental consecuencia de la mayor libertad de uso. En la figura 1 se muestra un esquema de acceso a servicios mediante RSR.

Dentro de RSR pueden, a su vez, considerarse dos arquitecturas. En la primera de ellas, denominada reconocimiento basado en voz codificada (NRS, *Network-based Speech Recognition*) [1] , el cliente envía la voz codificada al servidor usando un codec tradicional, por ejemplo G.729 [2] o AMR [3]. En la segunda, denominada reconocimiento distribuido de voz (DSR, *Distributed Speech Recognition*) [4, 5, 6], la voz es procesada en el propio terminal cliente, extrayéndose y transmitiéndose únicamente los parámetros requeridos por el sistema de reconocimiento del servidor.

En este trabajo proponemos una implementación de un reconocedor distribuido del habla para dispositivos móviles, basado en nuestro trabajo previo [7]. Se ha elegido la arquitectura DSR frente a NSR dada su menor tasa de bits y su mayor robustez frente a errores en el canal. El sistema DSR se implementa como una aplicación cliente-servidor. La aplicación cliente se puede instalar en cualquier PDA o computador personal que posea un equipo de audio que permita adquirir la voz del locutor.

El servidor lleva a cabo tres tareas principales: la gestión de la comunicación con los clientes, la mitigación de los errores producidos en la transmisión y el reconocimiento del habla. Para esta aplicación se proponen dos algoritmos para ordenar y mitigar, respectivamente, los errores en los paquetes enviados por el cliente. Estos algoritmos imponen un retardo máximo en el procesamiento de cada paquete, lo que permite su aplicación a tareas de reconocimiento en tiempo real.

La tarea implementada para mostrar el buen funcionamiento del sistema DSR ha consistido en el reconocimiento continuo de dígitos en inglés. Para ello se ha utilizado un reconocedor basado en modelos ocultos de Markov (HMM, *Hidden Markov Model*) [10] implementado mediante HTK (*Hidden Markov Toolkit*) [11], entrenado con la base de datos Aurora 2 [12].

Este trabajo se organiza como se indica a continuación. En la sección 2 se describe de forma general las característica del sistema de reconocimiento remoto implementado. Las secciones 3 y 4 describen los detalles de implementación del cliente y el servidor, respectivamente. Los resultados de reconocimiento se presentan en la sección 5. Finalmente, la sección 6 contiene las conclusiones de este trabajo.

2. Descripción general del sistema DSR

A continuación se describe, a grandes rasgos, el funcionamiento del sistema DSR implementado. Cada uno de los puntos siguientes será descrito en detalle en las secciones 3 y 4:

- Para acceder al servicio de reconocimiento remoto de voz, la aplicación cliente se conecta al servidor mediante una conexión TCP fiable.

- Una vez establecida la conexión, el cliente procede a adquirir la voz del usuario, extrayendo y codificando los parámetros necesarios para reconocimiento de acuerdo con el *front-end* AFE [4]. Los parámetros codificados se envían usando el protocolo RTP.

- La información enviada por el cliente será procesada por el servidor de reconocimiento de voz, que generará a partir de ella una representación en texto. Adicionalmente, el servidor hará uso de un algoritmo de mitigación de errores para reducir la tasa de error producida por pérdidas en la transmisión.

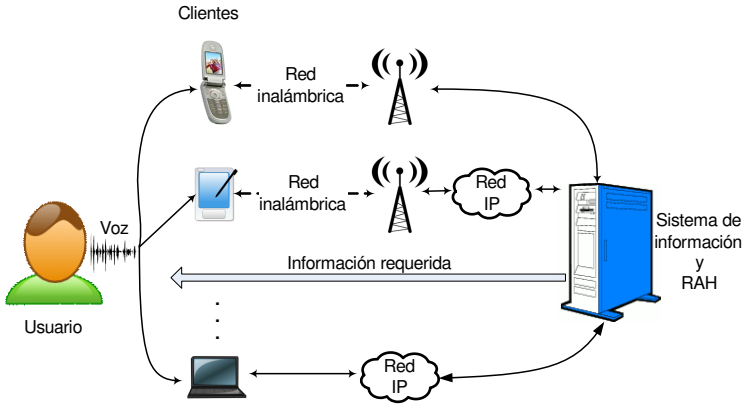

Figura 1: Acceso a servicios mediante voz usando RSR.

- El texto reconocido en el servidor es enviado al cliente a través de una conexión TCP, procediéndose finalmente al cierre de la conexión entre las dos aplicaciones.

3. Cliente DSR

Las tareas básicas de la aplicación cliente consisten en leer las muestras de voz del micrófono, extraer los parámetros necesarios para el reconocimiento, codificarlos , enviarlos al servidor y, por último, recibir y presentar al usuario los resultados del reconocimiento del habla que envía el servidor.

El cliente DSR ha sido evaluado en una PDA con procesador Intel® PXA270 416 MHz y 64 MB de memoria.

En los siguientes subapartados se detalla cada una de las partes que conforman el cliente.

3.1. Grabación de muestras de voz

Para permitir el funcionamiento en tiempo real de la aplicación cliente, se han definido dos hebras. La primera hebra se encarga de gestionar la obtención de muestras de voz del micrófono y almacenarlas en un buffer común. El formato de grabación utilizado es 8 KHz con 16 bits por muestra.

La segunda hebra se encarga de leer las muestras del buffer común, parametrizar la voz con el AFE y enviar los parámetros obtenidos al servidor según el RFC 3557 [9].

3.2. Procesamiento y transmisión de las características de la voz

De los diferentes *front-ends* estandarizados propuestos por ETSI, el que se utiliza en nuestro sistema es el AFE (ETSI ES 202 050 [4]), puesto que incorpora módulos de filtrado que lo hacen más robusto ante el ruido acústico. Dentro de las implementaciones de este estándar, se ha utilizado la que trabaja en punto fijo [8] por estar especialmente diseñada para ser usada en sistemas de comunicación móvil.

De acuerdo con el estándar AFE, para una frecuencia de muestreo de 8 KHz la señal de voz de entrada es dividida en segmentos de 25 ms (200 muestras) obtenidas cada 10 ms (80 muestras). Para cada segmento de 200 muestras, el *front-end* calcula 13 coeficientes MFCC (*Mel Frequency Cepstral Coefficients*), incluyendo el de orden 0, además del logaritmo de la energía del segmento de señal y un bit de VAD (*Voice Activity Detection*) que informa si dicho segmento contiene voz o no.

El vector de 14 parámetros obtenido en el paso anterior se codifica usando una cuantificación SVQ (*Split Vector Quantization*) [4]. En esta codificación los 13 coeficientes MFCC y el logaritmo de la energía (no se tiene en cuenta el bit de VAD) se agrupan en parejas,

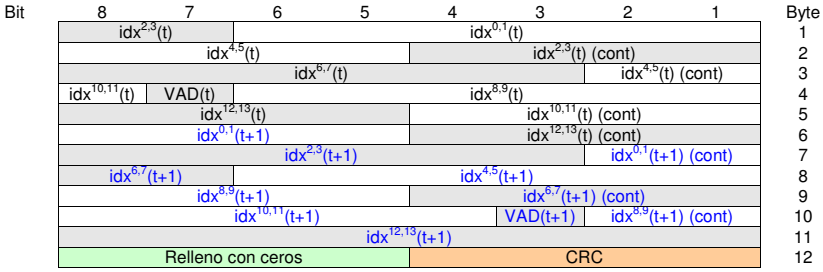

Figura 2: Carga útil (*payload*) del paquete RTP.

y cada pareja se cuantifica usando su propio diccionario. El conjunto de índices representa el segmento de voz. El número de bits que se emplean para codificar cada par de MFCCs del 1 a 10 es 6, mientras que se emplean 5 bits para los coeficientes MFCC 11 y 12, y 8 bits para codificar el coeficiente de orden 0 y el logaritmo de la energía.

Los vectores de parámetros comprimidos se agrupan en pares y son enviados al servidor usando el protocolo RTP sobre UDP. En la figura 2 se detalla la información que el cliente envía al servidor. Suponiendo que t es el índice temporal del segmento de voz actual y que $t+1$ el índice del siguiente segmento, entonces $idx^{i,i+1}(t)$ representa el índice SVQ asociado al par de parámetros $(i, i+1)$ para el segmento de voz t. El flag de VAD del segmento t se nota como $VAD(t)$. Cada par de vectores de parámetros codificados que el cliente envía al servidor se denomina *frame-pair*.

Como se observa en la figura, a continuación de los parámetros codificados el cliente añade 4 bits para el control de errores (notados como CRC en la figura), así como otros 4 bits de relleno puesto a cero, lo que supone una tasa de envío final de 4,6 kbps.

4. Servidor DSR

La aplicación servidor lleva a cabo diferentes tareas, siendo la principal el reconocimiento de la voz. Otra tarea importante es la gestión de la comunicación con los clientes DSR. Dado que múltiples clientes pueden solicitar a la vez el servicio de reconocimiento de voz al servidor, éste necesita definir un conjunto de hebras esclavas que interactuarán con cada cliente.

La comunicación entre los clientes y el servidor se realiza utilizando dos protocolos de transporte. El protocolo TCP se utiliza en aquellas tareas en las que exigimos fiabilidad en la transmisión a costa del retardo introducido. Ejemplos de esto son el establecimiento y cierre de la conexión, y el envío del texto reconocido. Por otro lado, el protocolo UDP se utiliza en aquellas tareas en las que exigimos un retardo acotado en la transmisión, aunque se puedan producir pérdidas. En nuestro sistema este protocolo se ha utilizado en la transmisión de las características de la voz.

En la figura 3 se muestra la arquitectura de red del servidor. Los clientes se conectan al servidor usando un puerto TCP conocido. Cada vez que se conecta un nuevo cliente al servidor, una hebra dedicada denominada *Maestro* acepta la nueva conexión, crea una nueva hebra denominada *Esclavo*, que se encargará de gestionar la comunicación con el nuevo cliente, y redirige el tráfico TCP a un nuevo puerto. Al contrario de lo que ocurre con la comunicación a través de TCP, donde cada par $(Cliente, Esclavo)$ tiene un puerto TCP único para comunicarse entre ellos, en el servidor sólo existe un puerto UDP donde todos los clientes deben mandar sus paquetes RTP/UDP. De la gestión de este puerto se encarga otra hebra, que copia los paquetes que le llegan a los buffers de recepción asociados a las hebras esclavas. Para determinar la hebra al que va dirigido cada paquete UDP, se consulta la dirección

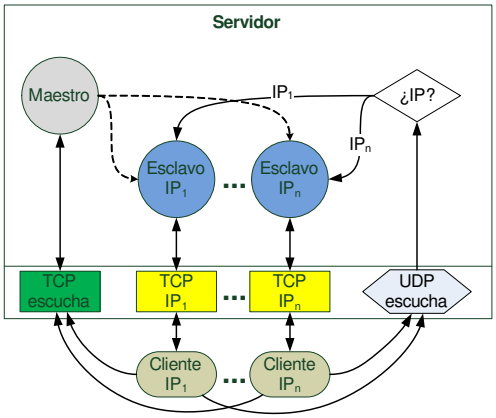

Figura 3: Arquitectura de red del servidor.

IP del emisor del paquete UDP entrante.

Como se ha dicho, cada vez que se recibe un nuevo paquete RTP/UDP de parte de un cliente, éste se copia en el buffer de recepción de la hebra que se encarga de gestionar la comunicación con dicho cliente, notificándole a ésta que tiene nuevos datos de entrada. Dado que los paquetes UDP pueden llegar desordenados, repetidos, con errores o, simplemente no llegar, es necesario definir en el servidor procedimientos para aliviar al máximo los errores que puedan provocar estos problemas en la tarea de reconocimiento. Para afrontar estos problemas proponemos dos algoritmos: uno de ordenación y otro de mitigación de errores.

4.1. Algoritmo de ordenación

Este algoritmo tiene dos fines. El primero es reordenar los paquetes RTP/UDP desordenados que llegan al servidor. El segundo es imponer un retardo máximo antes de dar por perdido un paquete que no haya llegado. Para satisfacer ambos objetivos utiliza una cola de espera de tamaño N. Como se dijo en la sección 3, y suponiendo que el tiempo de procesamiento es igual para todos los *frame-pairs*, cada 20 ms el cliente envía al servidor un par de vectores de parámetros codificados (un par de tramas). Disponer de una cola de espera permite al servidor no descartar paquetes retrasados, lo que se traduce en que para un tamaño de cola igual a N, como mucho se va a esperar $N \times 20$ ms a cada paquete. En la figura 4 se muestra un diagrama del funcionamiento de este algoritmo.

La recepción desordenada de paquetes conlleva la definición de dos modos de funcionamiento para el algoritmo de ordenación: uno síncrono y otro asíncrono, descritos a continuación.

4.1.1. Modo síncrono

En el modo síncrono se insertan y extraen paquetes de la cola de espera cada 20 ms. El esquema de funcionamiento de este modo es:

1. Inicialización

Supongamos que el primer paquete en llegar es P_n (con número de secuencia n). El paquete contiene dos tramas $P_n(1)$ y $P_n(2)$. Supuesto que los paquetes se empiezan a numerar desde 0, se insertan en la cola de espera n paquetes de relleno seguidos por el paquete P_n. Las posiciones de la cola de espera correspondientes a los paquetes previos no recibidos (del 0 al $n-1$) se rellenan con "paquetes vacíos"$\{V_0, V_1, ..., V_{n-1}\}$. El estado de la cola de espera después de esta operación sería $\{V_0, V_1, ..., V_{n-1}, P_n\}$, siendo P_n el último paquete de la cola.

Es posible que el primer paquete en llegar, P_n, tenga un número de secuencia mayor o igual que el tamaño de la cola (N). Si esto ocurriese, al finalizar la inicialización se extraerían de la cola de espera los $n-N$ primeros paquetes de relleno y se le pasarían al algoritmo de mitigación.

2. Caso general

Cada 20 ms a partir de la recepción del primer paquete, se inserta en la cola de espera un paquete de relleno numerado con el paquete que se espera recibir en ese instante de tiempo. Este paquete será sustituido posteriormente en el modo asíncrono.

Después de insertar el paquete de relleno, se comprueba si la cola de espera está llena (tiene más de N paquetes). En tal caso, se extrae de ésta el primer paquete y se le pasa al algoritmo de mitigación de errores.

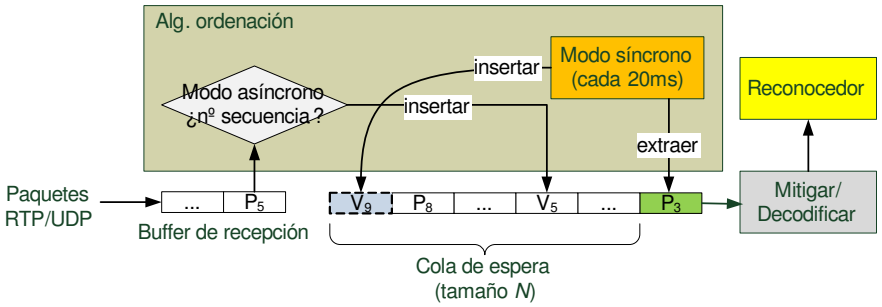

Figura 4: Diagrama de funcionamiento del algoritmo de ordenación.

4.1.2. Modo asíncrono

El modo asíncrono se dispara cada vez que llega un nuevo paquete al buffer de recepción. Si P_n es el paquete que acaba de llegar, el algoritmo de ordenación busca en la cola de espera el paquete V_n y lo sustituye por P_n.

Es posible, no obstante, que P_n no se pueda insertar en la cola de espera, bien porque haya llegado muy retrasado, bien porque haya llegado antes de lo esperado. Si el paquete ha llegado muy retrasado no se descarta, sino que se le pasa al algoritmo de mitigación de errores, puesto que podría ser útil a la hora de mitigar. Si el paquete ha llegado antes de tiempo, se mantiene en el buffer de recepción hasta que en el modo síncrono se introduzca V_n en la cola de espera.

4.2. Algoritmo de mitigación de errores

El algoritmo de mitigación de errores propuesto tiene como objetivo disminuir el efecto de los paquetes perdidos en la tasa de reconocimiento. A efectos de este algoritmo un paquete perdido o erróneo será aquél con CRC erróneo o un paquete de relleno.

A pesar de que el estándar AFE define un algoritmo de mitigación de errores, éste no garantiza un retardo máximo para la mitigación de pérdidas. Es por ello que resulta necesario modificar el algoritmo de mitigación de forma que el retardo máximo para la mitigación de cada paquete RTP/UDP esté acotado.

La mitigación sólo se aplica a los paquetes erróneos que salen de la cola de espera (los paquetes correctos no se mitigan). Mitigar consiste en sustituir las dos tramas del paquete erróneo, por las tramas más cercanas (en número de secuencia) de los paquetes recibidos correctamente, considerando tanto paquetes anteriores como posteriores. El paquete posterior que se utiliza para mitigar es el primero correcto en la cola de espera. El paquete anterior que se utiliza para mitigar se va actualizando durante la ejecución de dos formas. La primera de ellas consiste en actualizarlo cada vez que se extrae de la cola de espera un paquete correcto que no se tiene que mitigar. La segunda se realiza cuando el servidor recibe un paquete correcto con mucho retraso que ya se mitigó en su momento. Aunque este paquete no se inserta en la cola de espera, es posible usarlo para mitigar si es más reciente que el paquete anterior empleado para mitigar. Una vez mitigado el paquete, se decodifica y se le pasa al reconocedor de voz.

Para clarificar el funcionamiento de este algoritmo, en la figura 5 se presentan cuatro ejemplos de uso con un tamaño de cola de espera de $N = 3$ paquetes. En el ejemplo (A) se ha supuesto que el primer paquete que le llegó al servidor fue P_2, y que el algoritmo de ordenación insertó dos paquetes de relleno delante de él en espera de los paquetes P_0 y P_1. A los 20 ms de recibir P_2, el algoritmo de ordenación inserta un paquete de relleno con el número de secuencia del paquete que se espera recibir en ese instante (el 3). Como el tamaño de la cola de espera es de $N = 3$ paquetes, se extrae el primero de ésta, V_0, para ser mitigado.

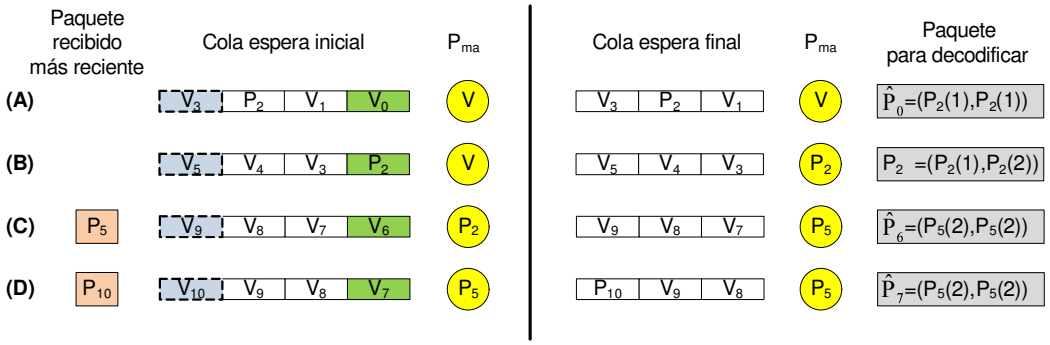

Figura 5: Ejemplos de uso del algoritmo de mitigación de errores.

Como se ha comentado, la mitigación consiste en sustituir las tramas de dicho paquete por las más cercanas de entre los paquetes recibidos. Como el único paquete que se ha recibido correctamente es P_2 (el paquete de mitigación anterior (P_{ma}) es de relleno V), el algoritmo de mitigación devuelve una estimación de P_0 consistente en sustituir sus dos tramas por la primera de P_2, esto es, $\hat{P}_0 = (P_2(1), P_2(1))$.

Desde la situación reflejada en el ejemplo (A) a la reflejada en el ejemplo (B), han pasado 2×20 ms sin haber recibido ningún paquete, de forma que se ha insertado en la cola el paquete V_4 y se ha mitigado las dos tramas de V_1 con $P_2(1)$. En el ejemplo (B) el paquete P_2 debe salir de la cola. Como no contiene errores, no se mitiga y se guarda una copia de él en P_{ma} para ser usado en posteriores mitigaciones.

Del ejemplo (B) al (C) tampoco se ha recibido ningún paquete, por lo que cada 20 ms el algoritmo de ordenación ha insertado un paquete de relleno al final de la cola de espera y ha sacado uno del principio de ésta, siendo mitigadas todas las tramas de $\{V_3, V_4, V_5\}$ por $P_{ma}(2) = P_2(2)$. En el ejemplo (C) se ha recibido el paquete P_5. Como este paquete ha llegado con mucho retraso (ya se mitigó anteriormente), no se inserta en la cola de espera, pero sí se le pasa al algoritmo de mitigación para que lo pueda usar durante la mitigación. De hecho, el algoritmo de mitigación sustituye el paquete P_{ma} por P_5, y mitiga las dos tramas de V_6 con $P_{ma}(2) = P_5(2)$.

En el ejemplo (D) el algoritmo de ordenación insertará V_{10} en la cola de espera (modo síncrono) y, después, sustituirá este paquete por P_{10} (modo asíncrono). Por último, las dos tramas del paquete V_7 son sustituidas por la segunda del paquete $P_{ma} = P_5$, pues está más cerca temporalmente que las del paquete P_{10}.

Aunque no se ha reflejado en los ejemplos, en el caso de que un paquete para mitigar se encuentre a la misma distancia de dos paquetes recibidos, como es el caso de V_{20} y los paquetes P_{19} y P_{21}, la primera trama de V_{20} se sustituye por la última de P_{19}, y la última de V_{20} se sustituye por la primera P_{21}, quedando finalmente un paquete de la forma $\hat{P}_{20} = (P_{19}(2), P_{21}(1))$.

4.3. Reconocimiento de la voz

4.3.1. Aurora 2

El entrenamiento de los modelos acústicos utilizados para reconocimiento se ha realizado con la base de datos Aurora 2 [12]. Esta es una base de datos en inglés de dígitos conectados dividida en dos conjuntos de entrenamiento y tres de *test*. Para entrenar nuestros modelos se ha usado el conjunto de entrenamiento denominado *Clean*, en el que los ficheros de audio no están contaminados por ruido acústico. Para probar el sistema se ha utilizado el conjunto de *test A*, que contiene tanto ficheros con ruido

	Traza 1		Traza 2	
Tamaño cola de espera	*WAcc (%)*	*Degradación relativa del WAcc*	*WAcc (%)*	*Degradación relativa del WAcc*
0	85,553	1,555	83,680	3,710
1	86,810	0,108	85,088	2,090
2	86,867	0,043	85,546	1,563
3	86,885	0,022	85,874	1,186
4	86,897	0,008	86,140	0,879
5	86,915	0,013	86,363	0,623
6	86,906	0,002	86,534	0,426
7	86,903	0,002	86,644	0,300
8	86,903	0,002	86,664	0,277
9	86,903	0,002	86,664	0,276
10	86,903	0,002	86,682	0,255
11	86,903	0,002	86,682	0,255
∞	86,903	0,002	86,682	0,255

Tabla 1: Resultados de reconocimiento para diferentes tamaños de cola de espera.

como sin ruido.

4.3.2. Modelado acústico

Para entrenar los modelos acústicos se utilizan vectores de parámetros de 39 coeficientes obtenidos a partir de los vectores de 14 parámetros que el cliente envía al servidor. Así, el nuevo vector se obtiene combinando el coeficiente de MFCC de orden 0 y el logaritmo de la energía en un solo parámetro, y añadiendo 13 coeficientes de velocidad y 13 de aceleración.

El motor de reconocimiento usado se basa en el toolkit HTK en su versión 3.4 [11]. Los dígitos usados en el aprendizaje se modelan mediante modelos ocultos de Markov (HMM). Cada HMM cuenta con 16 estados formados por mezclas de 3 gaussianas (matrices de covarianza diagonales) en una topología de izquierda a derecha.

Además de los dígitos, se definen dos modelos de pausa. El primero de ellos consta de 3 estados con 6 gaussianas cada uno. Este HMM modela las pausas al principio y al final de las frases. El segundo modela las pausas entre palabras. Consta de un solo estado ligado al estado central del primer modelo de pausa.

5. Resultados de reconocimiento

Con objeto de evaluar el rendimiento mostrado por la aplicación DSR implementada en un entorno real, se ha diseñado una prueba consistente en el reconocimiento del *test A* de Aurora 2, cuando los ficheros de éste se transmiten por un canal real.

La transmisión de los ficheros por una red IP se ha simulado usando dos trazas reales con pérdidas y retrasos en la transmisión de los paquetes. Las trazas se han obtenido midiendo el tiempo de ida y vuelta de paquetes ICMP enviados a diferentes localizaciones. En las dos trazas consideramos como paquetes perdidos aquellos con *checksum* erróneo. La primera traza, que denominaremos *Traza 1*, contiene un porcentaje de pérdidas de paquetes del 0,155 %, un retardo medio en la transmision de 0,356 s y una varianza de $4,195 \times 10^{-5}$ s. La segunda traza, que denominaremos *Traza 2*, contiene un porcentaje de pérdida de paquetes del 1,610 %, un retardo medio en la transmisión de 0,196 s y una varianza de $2,577 \times 10^{-4}$ s.

Una vez simulada la transmisión de los paquetes por la red con las dos trazas, en el servidor se ha evaluado la tasa de reconocimiento de palabras (WAcc, *Word Accuracy*) y la

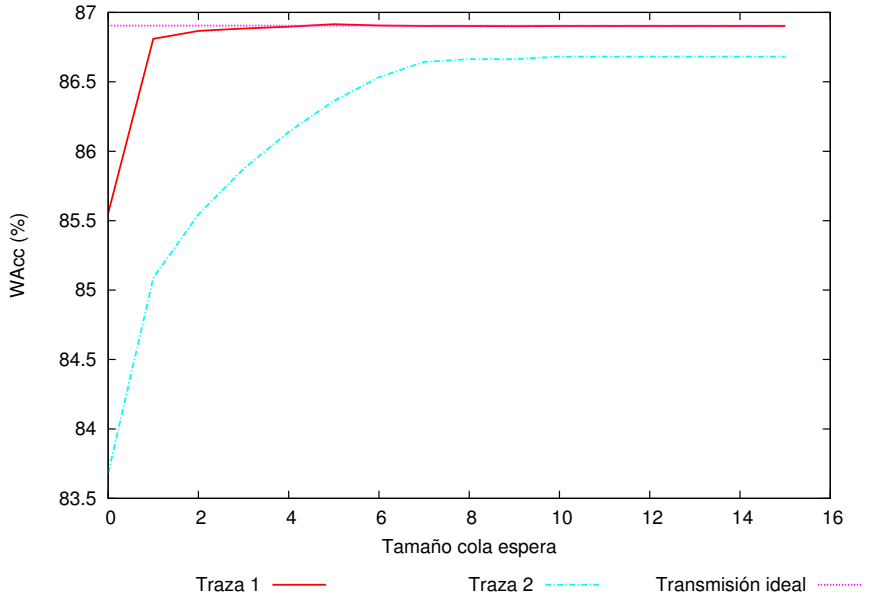

Figura 6: Mejoras en la tasa de reconocimiento gracias a los algoritmos de ordenación de paquetes y mitigación de errores.

degradación relativa del WAcc respecto a una transmisión ideal en la que no se producen pérdidas ni retrasos (la tasa de reconocimiento obtenida en este caso es de $WAcc = 86,904\,\%$). La tabla 1 recoge esta información para diferentes tamaño en la cola de espera, promediando los resultados obtenidos para el *test A* de Aurora 2 para condiciones de ruido de 0 a 20 dB. Como se puede observar la tasa de reconocimiento converge para un tamaño de cola igual a 7 en la *Traza 1* y de 10 en la *Traza 2*, esto es, disponer de una cola de espera más grande no reporta beneficios. El que no se produzcan mejoras en el reconocimiento a partir de un determinado tamaño de cola tiene explicación. En la *Traza 2*, por ejemplo, el retardo medio que sufren los paquetes es aproximadamente igual a 0,20 segundos, mientras que el retardo máximo es levemente inferior a 0,4 s, es decir, hay una diferencia de 0,2 s entre el retardo medio y el retardo máximo. Como cada nodo de la cola de espera equivale temporalmente con un retardo de 20 ms, es suficiente con una cola de tamaño $(0,4 - 0,2)s/20ms = 10$ paquetes, para que un paquete que llegue con el máximo retardo no se dé por perdido.

A partir de los resultados de la tabla 1 podemos determinar que un tamaño razonable para la cola de espera sería de 5 paquetes, puesto que el retardo introducido es corto (de $5 \times 20ms = 100$ ms), al tiempo que se consigue una buena tasa de reconocimiento en ambas condiciones de red.

La figura 6 compara las tasas de reconocimiento de la tabla 1, con el resultado obtenido suponiendo una transmisión ideal (sin retrasos ni pérdidas de paquetes). Como se observa, el uso de los algoritmos de ordenación y de mitigación en el servidor permiten salvar en gran medida los problemas introducidos por la transmisión de paquetes, al tiempo que se consiguen tasas de reconocimiento equiparables a las obtenidas en un canal ideal sin pérdidas.

6. Conclusiones y trabajo futuro

En este trabajo se ha presentado una implementación de un sistema de reconocimien-

to distribuido del habla en tiempo real para su uso en dispositivos móviles del tipo PDA. Desarrollado como una aplicación cliente-servidor, el cliente hace uso del *front-end* avanzado definido por la ETSI. El servidor implementa un algoritmo de ordenación de paquetes y un algoritmo de mitigación de errores, que permiten reducir los errores provocados en el reconocimiento de voz por problemas en la transmisión sobre UDP. Los resultados de reconocimiento han demostrado una alta robustez frente a problemas en el canal con un retardo mínimo en la mitigación.

Como trabajo futuro queda el estudio y la aplicación de otros métodos de mitigación de errores diferentes, como los propuestos en [6], que puedan aprovechar el retardo inevitable introducido por las colas de espera para poporcionar una mejor mitigación de paquetes perdidos de forma que la degradación producida por la pérdida de paquetes sea inapreciable.

Referencias

[1] A. Gómez, A.M. Peinado, V. Sánchez, A.J. Rubio. *Recognition of Coded Speech Transmitted Over Wireless Channels.* IEEE Trans. on Wireless Communications, no. 9, vol. 5, September 2006.

[2] Draft Recommendation G.729. *Coding of Speech at 8 kbit/s Using Conjugate-Structure Algebraic Code-Excited Linear-Prediction (CS-ACELP).* International Telecommunication Union, Telecommunications Standardization Sector, 1995.

[3] 3GPP technical report. *AMR speech Codec; General description.* 3GPP TS 26.071, April 1999.

[4] ETSI ES 202 050. *Speech Processing, Transmission and Quality Aspects (STQ); Distributed speech recognition; Advanced front-end feature extraction algorithm; Compression algorithms.* November 2005.

[5] A.M. Peinado, V. Sánchez, J.L. Pérez-Córdoba, A. de la Torre. "HMM-Based Channel Error Mitigation and its Application to Distributed Speech Recognition". Speech Communication, vol. 41/4, pp. 549-561, Nov. 2003.

[6] A. Gómez, A.M. Peinado, V. Sánchez, A.J. Rubio. *Combining media-specific FEC and error concealment for robust distributed speech recognition over loss-prone packet channels.* IEEE Trans. on Multimedia, no. 6, vol. 8, December 2006.

[7] Juan A. Morales Cordovilla, Timo Bauman, José L. Pérez-Córdoba, Antonio M. Peinado Herreros, Ángel M. Gómez García. *Implementación de un reconocedor distribuido de voz en tiempo real sobre IP.* IV Jornadas en Tecnología del Habla, Zaragoza, Noviembre 2006.

[8] ETSI TS 126 243 *Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); ANSI C code for the fixed-point distributed speech recognition extended advanced front-end.* December 2004.

[9] IETF. RFC 3557 *RTP Payload Format for ETSI European Standard ES 201 108 Distributed Speech Recognition Encoding*, 2003.

[10] L.R.Rabiner. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* In *Proc. of the IEEE* , volume 77, pages 257-285, Febrero 1989.

[11] S.Young, G. Evermann, T. Hain, et al. *The HTK Book - Version 3.2.1.* Cambridge University Engineering Department, November 2005, http://htk.eng.cam.ac.uk

[12] H.G. Hirsh, D. Pearce. *The Aurora Experimental Framework for the Performance Evaluations of Speech Recognition Systems under Noisy Conditions.* ISCA ITRW ASR 2000, November 2000.