4. La biblioteca AVM CAPI ADK.

En este capítulo se tratará la biblioteca de desarrollo AVM ADK: su funcionamiento, modificaciones realizadas a la misma, dependencias con otras bibliotecas y motivos que decidieron su utilización para el desarrollo del proyecto.

4.1. Dificultad de la programación usando CAPI 2.0.

Aunque la interfaz CAPI 2.0 es muy potente y permite el acceso a todos los servicios RDSI, la programación de una aplicación de red utilizando mensajes CAPI 2.0 es tediosa y compleja. Esto es debido principalmente a que los mensajes CAPI 2.0 se construyen como un flujo continuo de octetos con campos de longitud variable codificados al estilo ASN 1. Además, la mayoría de los campos no son comunes a todos los mensajes, y sólo aparecen en cierto tipo de mensajes o si se está utilizando una característica determinada del servicio ofrecido por la red RDSI.

Para crear este tipo de mensajes, y principalmente para analizar un mensaje de respuesta, es necesario recorrer el mensaje de principio a fin, incluso si estamos interesados sólo en cierto campo del mismo. Supongamos por ejemplo que tenemos una sola conexión activa y recibimos un mensaje de tipo `DATA_INDICATION`. Puesto que sólo tenemos una conexión activa, es completamente innecesario leer los campos de identificador de conexión, etc. y el único campo interesante es el que contiene los datos recibidos, pero el carácter secuencial de los mensajes CAPI 2.0 nos obliga a recorrer todos los campos anteriores hasta llegar a los datos.

Asimismo, resulta imprescindible mantener una máquina de estados de la comunicación para conocer en cada instante qué mensaje puede ser enviado o recibido y cómo hay que responder a cada mensaje CAPI 2.0. Implementar la máquina de estados CAPI 2.0 completa es un trabajo tedioso y complicado.

Todo esto resulta bastante engorroso, sobre todo para desarrolladores que hayan utilizado interfaces como la interfaz de `sockets` para comunicaciones mediante los protocolos de Internet.

4.2. Funcionalidad ofrecida y dependencia con c20lib.

La biblioteca de desarrollo de aplicaciones AVM CAPI ADK (*Application Development Kit*) facilita la creación de aplicaciones basadas en RDSI al mantener por nosotros la máquina de estados de la comunicación y ocuparse también de la creación de los mensajes CAPI 2.0. En realidad los mensajes no los crea esta biblioteca, sino otra biblioteca, llamada `c20lib`, una versión de la cual viene incluida en la distribución de la AVM ADK.

La biblioteca `c20lib` simplemente ofrece dos funciones en lenguaje C que permiten convertir un mensaje CAPI 2.0 en forma de flujo continuo de octetos en una estructura compleja de datos a base de `struct` y `union`, y viceversa. De forma que cuando se recibe un mensaje CAPI 2.0 sólo hay que convertirlo y acceder a los campos de las estructuras y uniones para hallar el tipo de mensaje recibido y sus opciones. Asimismo, para enviar un mensaje al controlador CAPI 2.0 basta rellenar los campos necesarios de estas estructuras y uniones y utilizar la función de conversión para obtener el mensaje CAPI 2.0 que hay que enviar. Aún así, programar una aplicación RDSI utilizando la biblioteca `c20lib` es bastante pesado.

La biblioteca AVM ADK ofrece un método sencillo para desarrollar aplicaciones CAPI 2.0. Su uso se basa en las siguientes facilidades proporcionadas:

- Definición de funciones sencillas para establecer conexiones vocales, de datos, de fax, etc., entrantes o salientes.
- Declaración de varias funciones que deben ser definidas en la aplicación final, y que señalan cambios de estado en la comunicación; lo que se conoce como funciones de *callback*.
- Mantenimiento de la máquina de estados para todas las comunicaciones posibles. En principio sólo permite mantener dos conexiones -equivalente a un acceso básico- aunque esto es fácilmente modificable.

En definitiva, el diseño de una aplicación RDSI mediante el conjunto CAPI 2.0 y AVM CAPI ADK se basa en definir las funciones de *callback* y utilizar las funciones de inicio de conexión para establecer el comportamiento de la misma, sin necesidad de tratar en ningún momento con la interfaz CAPI 2.0 ni con la biblioteca `c20lib`.

4.3. Implementación de AVM CAPI ADK.

La ilustración 5 muestra un diagrama de la implementación final de la AVM CAPI ADK, los módulos que la componen y la interrelación entre los mismos y la aplicación final.

Cada módulo se representa mediante un cajón cuya pestaña contiene el nombre del fichero de cabecera `.h` asociado al mismo. Los nombres en el interior del cajón corresponden a los nombres de las funciones exportadas por el módulo.

Las pestañas pueden ser de varios colores:

- Rojo: indica un módulo añadido a la AVM CAPI ADK. Estos módulos no vienen incluidos en la distribución que puede obtenerse del sitio `ftp` de AVM.
- Verde: el módulo ha sido modificado para añadir nueva funcionalidad.
- Gris: el módulo se ha incluido en el proyecto tal cual, sin realizar ninguna modificación al mismo.

Los nombres de las funciones pueden aparecer en tres colores distintos:

- Negro: la aplicación no necesita invocar a estas funciones directamente; son funciones internas a la AVM CAPI ADK.
- Blanco: estas funciones son sólo prototipos que deben ser definidos en el cuerpo de la aplicación principal; son las funciones de *callback*.
- Verde: estas funciones deben ser invocadas desde la aplicación final para, por ejemplo, registrarse en CAPI 2-0, establecer conexiones o enviar datos.

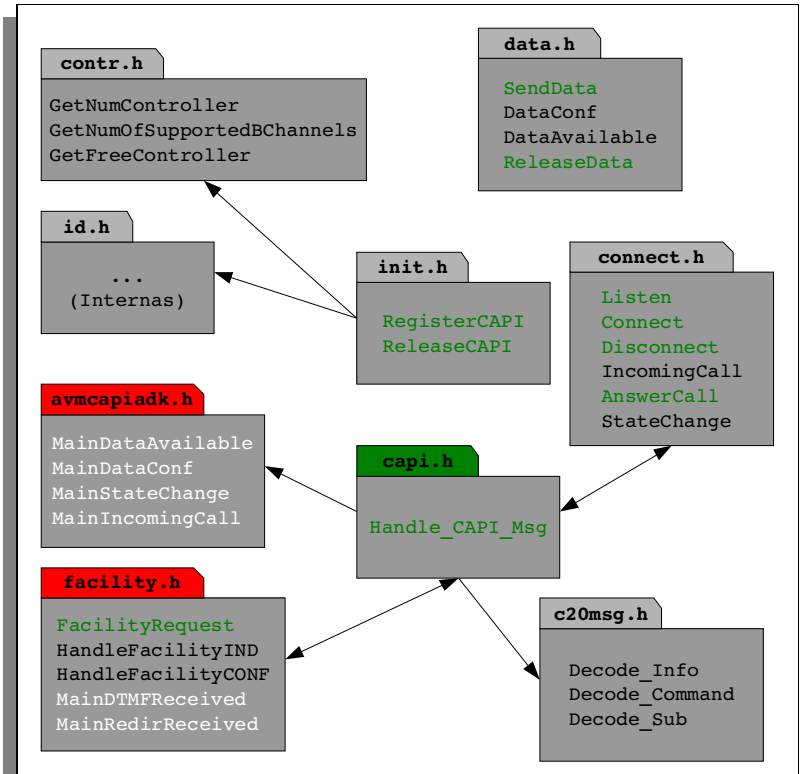

Ilustración 5. Implementación de AVM CAPI ADK.

El módulo `avmcapiadk.h` ha sido añadido para conseguir una biblioteca completa, puesto que la AVM CAPI ADK no se distribuye como tal, sino como una biblioteca que forma parte de un programa de ejemplo. Por lo tanto la distribución oficial de la ADK no es utilizable directamente, sino que es necesario modificarla en parte.

Por otro lado, el módulo `facility.h` ha sido añadido y el módulo `capi.h` ha sido modificado para conseguir incluir soporte a los servicios suplementarios de detección de tonos de teclado -DTMF- y transferencia explícita de llamada -ECT-. Estas modificaciones añaden los estados necesarios a la máquina de estados de la comunicación y crean los mensajes necesarios para el controlador de emulación CAPI 2.0.

En definitiva, la biblioteca junto con las modificaciones realizadas permiten crear una aplicación RDSI con acceso a los servicios de detección DTMF y ECT de forma sencilla y bastante elegante. Quizás sería deseable que dicha biblioteca no estuviese tan fragmentada en módulos pequeños, sino que al compilarse se obtuviera un sólo archivo de código objeto y sólo hubiera que incluir un fichero de cabecera. De cualquier modo, es una forma sencilla y efectiva de crear este tipo de aplicaciones.