Capítulo 6.
Prototipo

En este capítulo se describen las herramientas utilizadas para la implementación del modelo de videoconferencia VicFDL, los requerimientos de hardware utilizados, los componentes y la funcionalidad de VicFDL.

6.1 Herramientas

El prototipo fue implementado en los lenguajes de programación Java y C++. Estos lenguajes se encuentran en el paradigma orientado a objetos. Java por su parte tiene la ventaja de ser portable a través de plataformas y sistemas operativos, pero ¿porqué C++ es utilizado para la implementación de este sistema en ves de java en su totalidad?, primero, porque un programa compilado siempre será más rápido que un programa interpretado, la velocidad de ejecución es un factor primordial, el acceso a al sistema operativo es otro, y por razones de seguridad los applets de java no pueden realizar tareas como la escritura a disco y el acceso a los dispositivos de video o a los puertos serie, un programa en C++ es más flexible porque puede llamar a cualquier función del sistema operativo y manipular los dispositivos de captura de video a bajo nivel, mientras que java aún esta limitado.

También se utilizaron las clases de JMF (java media framework) que son una interfaz de programación que permite la manipulación de multimedios en una red a través de sus API´s. JMF proporciona las herramientas necesarias para la creación y manipulación de elementos multimedia. JMF es parte de los API´s de java y de esta forma permite que sea accesible en todas las plataformas e Internet. JMF obtiene ideas de otros API´s existentes y de nuevas tecnologías. El video de JMF sintetiza las mejores ideas encontradas en API´s como son DirectX, JPG, MPG, AVI, MOV, QuickTime, Direct3D, OpenGL, XGL, QuickDraw3D [OpenGL 1998].

También se utilizo el lenguaje de programación Tcl/Tk que trabaja en combinación con C++, este lenguaje proporciona las herramientas necesarias para la manipulación de dispositivos secundarios como lo son cámaras o tarjetas de video, tarjetas de audio entre otras, además para la manipulación directa de video proveniente de la cámara, se utilizaron API´s propios de SUN Microsystems, librerías para la creación aplicaciones

multicast vic, vat y sdr proporcionadas por Ernest Orlando Lawrence Berkeley National Laboratory.

Para el caso de las estaciones de trabajo con sistema operativo solaris y windows se tuvieron que reprogramar y reestructurar las librerías, siguiendo estrictamente los estándares de la interfaz dictados por dicho laboratorio, pues se encuentran aún en proceso de investigación y desarrollo. Desafortunadamente para la manipulación de video en ambos programas nos muestra la información del API, por esta razón al levantarse los servicios en el módulo de aplicación aparecerá información sobre el desarrollo de estos componentes.

6.2 Requerimientos de Hardware

La utilización de las librerías JMF tienen ciertos requerimientos de hardware como el chip S3 el cual hace uso del API XIL y del API OpenGL de UNIX para la manipulación de cámaras de video y acelerador de propósitos gráficos de rango medio. [JMF 1998]. La utilización de C++ tiene ciertos requerimientos para la manipulación de audio y video, pues es necesario que el equipo donde se esté desarrollando al sistema cuente con los API´s necesarios para la manipulación de elementos de hardware como una videocámara o tarjeta capturadora de video y una tarjeta de sonido.

6.3 Módulos y componentes del sistema

El componente *Win32* inicializa los sockets de windows en conjunción con la clase *tclc++,* eventualmente se llaman a las otras clases de *Str_c++, ctl y tkstripchart* para la manipulación de los dispositivos de audio y video.

El componente *Assistor* incrementa el tamaño del buffer para el envío de video, donde a través de la clase *inet* envía paquetes de datos utilizando las *clases XIL_apis, win32x, confbus y mkversion,* si fracasa este envío de datos, cancela la operación y la codificación del video no se lleva a cabo.

La clase *XIL_apis* manipula los dispositivos de captura de audio y video (Sunvideo) tomando el video en vivo y mostrándolo en pantalla. Por otro lado la clase *net* determina el sistema operativo y el tipo de red que se está utilizando enviando bloques de datos para verificar la existencia de direcciones *ip* (protocolo de Internet) y puerto a utilizar. *Device_Input* , esta es la clase que utiliza los dispositivos de entrada, a través este, se determina con que clase de hardware se cuenta realizando una búsqueda de puertos para verificar cuál esta siendo utilizado por el dispositivo de video.

La clase *decoder_src* se comunica con los protocolos y decodificadores como el rtp (protocolo de transferencia real) el H.261 (estándar de videoconferencia), decodificadores de video H.261, JPEG y p64, mientras que la clase *renderer_window* realiza las operaciones de texturización y codificación del video para ser visualizado en una ventana o en varias. *Encoder-cell* transforma la señal de texturización y codificación para enviarla a las clases *grabber* , pues éstas contienen los controladores de las tarjetas de captura de video. Las clases *grabber* determinan si los dispositivos de captura de video son compatibles con los estándares utilizados por la cámaras quickcam, vivitar, sunvideo y sinrise, proporcionando los *API´s XIL* para la manipulación de video en UNIX y *capture* para windows visualizando en tiempo real el video proveniente de estos dispositivos a través de la clase *h261_play* .

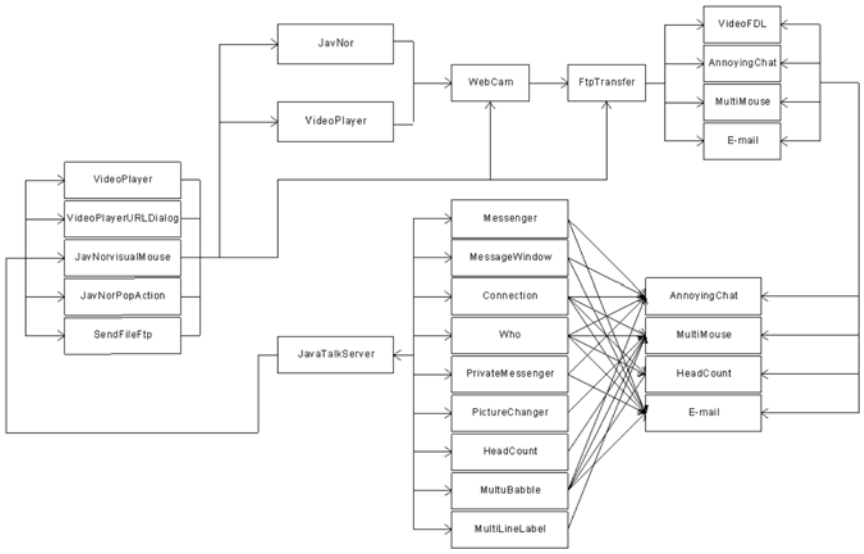

Figura 6.1 Componentes del Módulo Web

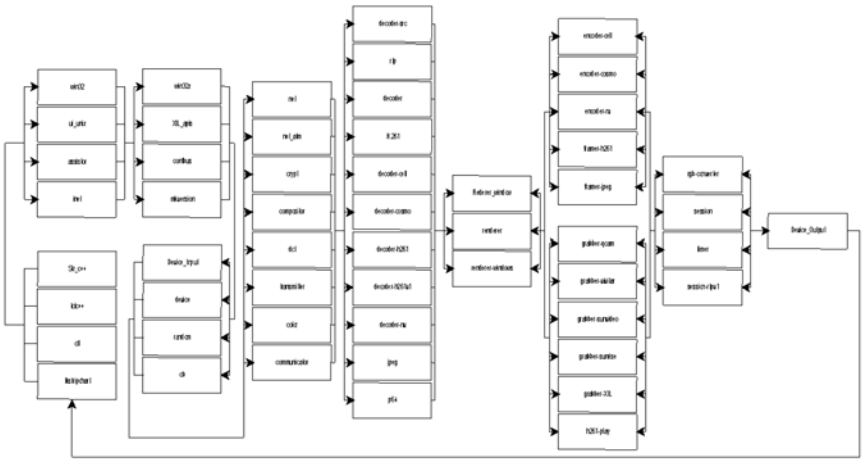

Figura 6.2 Componentes del Módulo de Aplicación

6.4 FUNCIONALIDAD DE VICFDL

Se describe de una forma más detallada la utilización del módulo de aplicación en función de cada uno de sus componentes (audio, video y pizarrón electrónico) en el apéndice C.

Figura 6.3 Navegador taxonómico y herramientas de Ágora (adaptada de [Fernández 1998])

Figura 6.4 Opciones de comunicación (adaptada de [Fernández 1998])

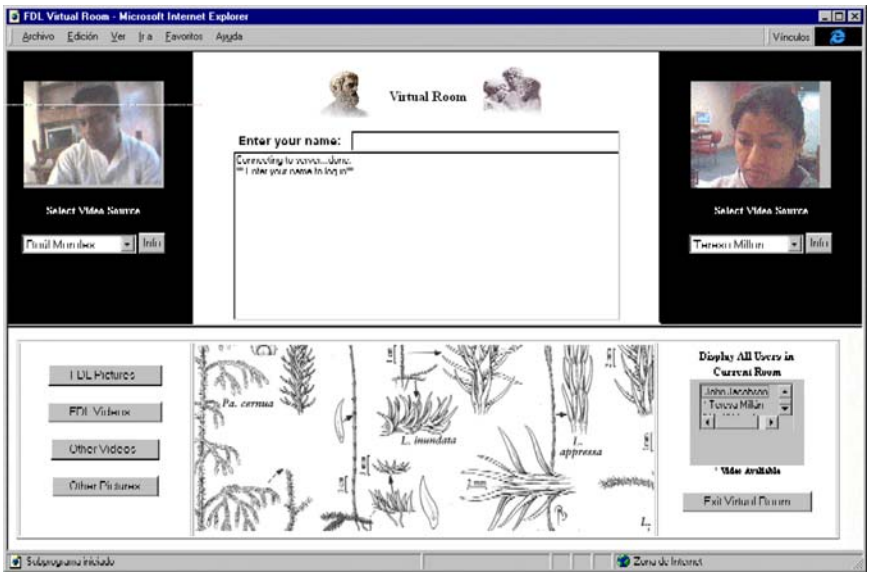

Figura 6.5 Salón Virtual VicFDL

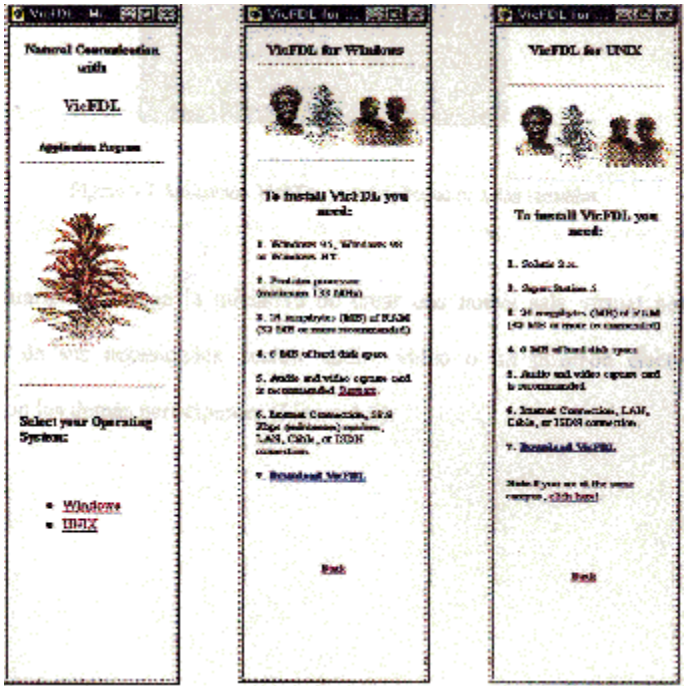

Figura 6.6 Opción de comunicación natural VicFDL y sus requerimientos

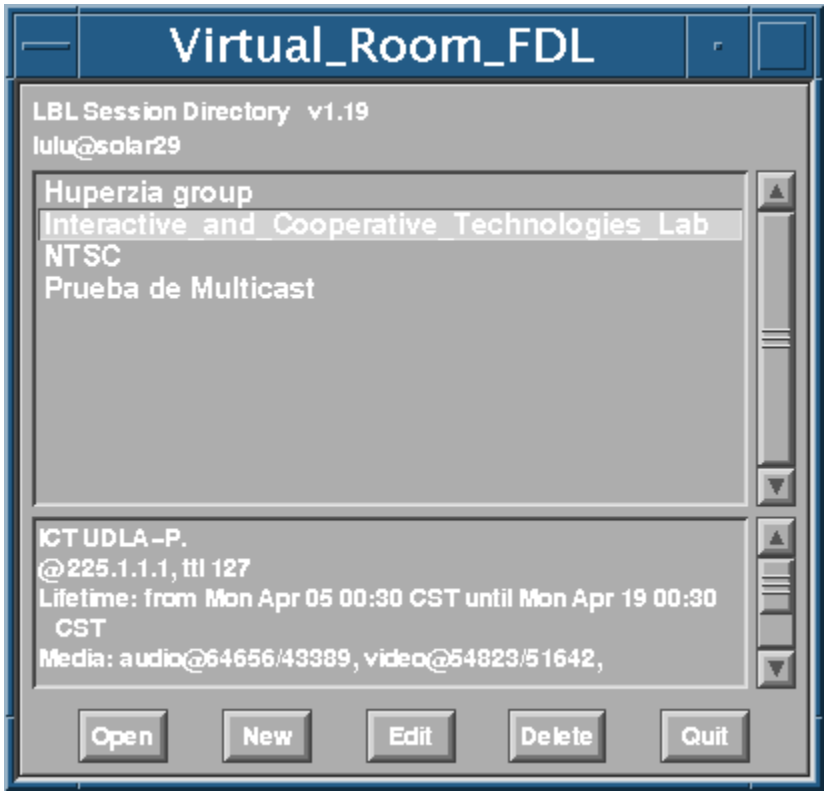

Figura 6.7 Aplicación VicFDL y el despliegue de salas virtuales

Figura 6.8 Aplicación VicFDL. Creación de salas virtuales

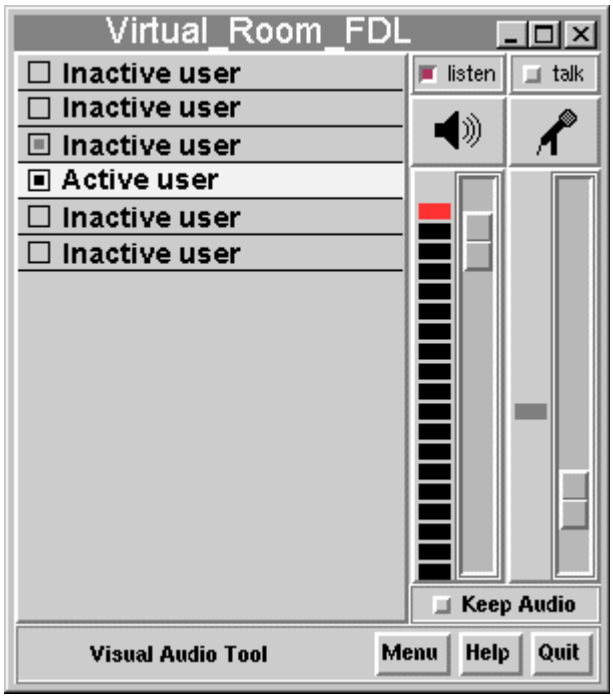

Figura 6.9 Aplicación VicFDL. Nivel de participación

Figura 6.10 Aplicación El VicFDL. Recepción de video

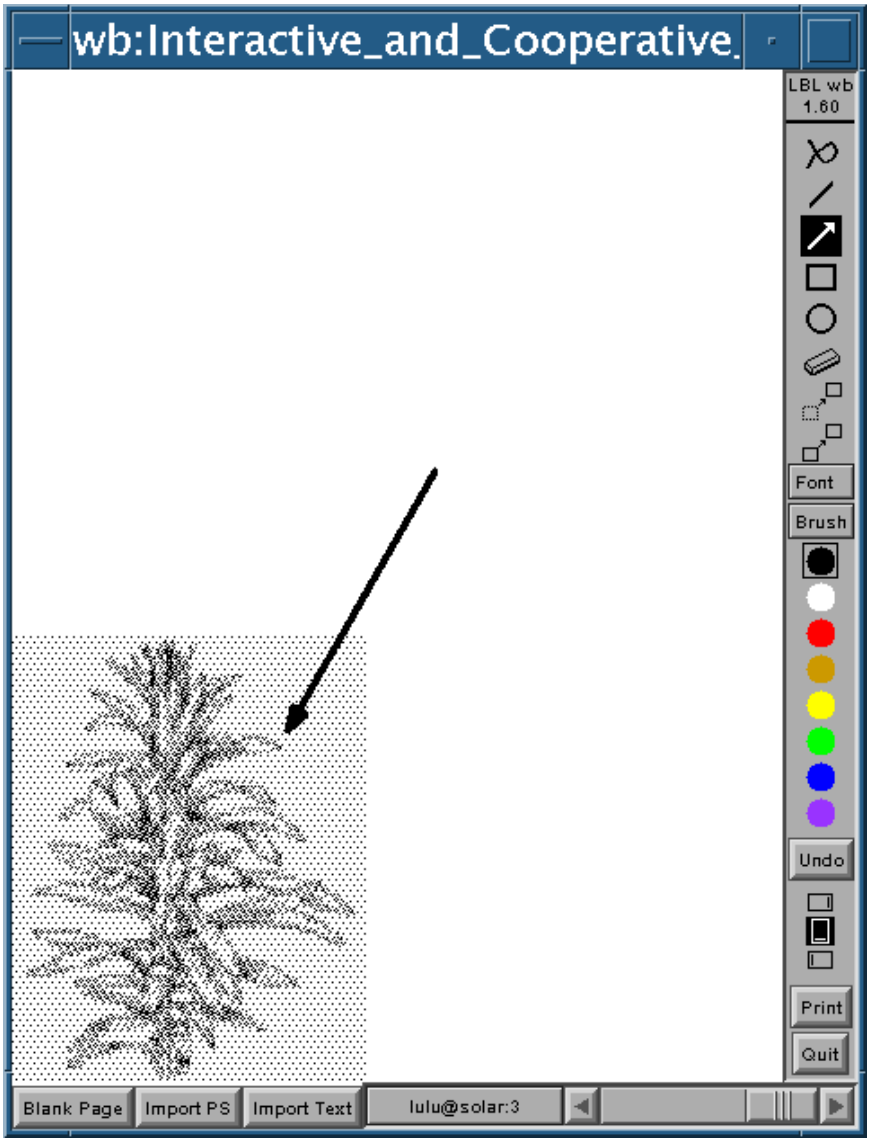

Figura 6.11 Aplicación VicFDL. Pizarrón electrónico compartido

Morales Salcedo, R. 1999. **Aplicaciones de la Videoconferencia en Bibliotecas Digitales**. Tesis Maestría. Ciencias con Especialidad en Ingeniería en Sistemas Computacionales. Departamento de Ingeniería en Sistemas Computacionales, Escuela de Ingeniería, Universidad de las Américas Puebla. Mayo. Derechos Reservados © 1999.