Entorno Remoto de Desarrollo de Aplicaciones Paralelas en Ambiente PVM

Mauricio Gómez C.
Javier Cañas R.

Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, CHILE

email : {maurro, jcanas}@inf.utfsm.cl

Resumen

El presente trabajo extiende las funcionalidades de la máquina PVM Parallel Virtual Machine al mundo de los entornos de trabajo desktop (Windows NT / 95) a través de un ambiente remoto de desarrollo de aplicaciones paralelas que permite que un usuario utilice su computador personal para escribir y ejecutar algoritmos paralelos dentro del entorno PVM.

Se desarrolla el marco conceptual y la implantación de un sistema que permite el uso y acceso remoto por parte de varios usuarios a una única máquina virtual PVM. Como resultado de este desarrollo se puede, por ejemplo, establecer una conexión a la máquina virtual PVM desde una aplicación cliente en el entorno desktop, dentro de la cual se realizará la ejecución de un algoritmo paralelo y se obtendrá sus resultados.

Palabras claves : Sistemas Distribuidos, Computación Paralela, Prototipos Rápidos de Sistemas Paralelos

Trabajo parcialmente financiado por Proyecto USM N°95.24.21

1. Introducción

La tecnología PVM, o Parallel Virtual Machine [3] permite la integración de una red heterogénea de máquinas y arquitecturas en una sola gran máquina virtual que ofrece las potencialidades de una arquitectura paralela, cuyo acceso y uso es relativamente sencillo para un usuario con un conocimiento mediano de desarrollo de aplicaciones en ambiente Unix.

El presente trabajo extiende las funcionalidades de la máquina PVM al mundo de los entornos de trabajo desktop (Windows NT / 95) a través de un ambiente remoto de desarrollo de aplicaciones paralelas que permite que un usuario utilice su computador personal para escribir y ejecutar algoritmos paralelos dentro del entorno PVM.

Se desarrolla el marco conceptual y la implantación de un sistema que permite el uso y acceso remoto por parte de varios usuarios a una única máquina virtual PVM. Como resultado de este desarrollo se puede, por ejemplo, establecer una conexión a la máquina virtual PVM desde una aplicación cliente en el entorno desktop, dentro de la cual se realizará la ejecución de un algoritmo paralelo y se obtendrá sus resultados.

2. El Ambiente PVM.

Parallel Virtual Machine, mejor conocido como PVM, es un sistema de software que permite formar una arquitectura virtual de multicomputador a partir de la cooperación de una colección heterogénea de máquinas seriales, paralelas o vectoriales que operan sobre diferentes sistemas operativos. La idea principal de PVM es otorgar a un usuario final un entorno transparente sobre el cual se pueda desarrollar aplicaciones paralelas, sin que resulte un obstáculo las diferencias en cuanto a las arquitecturas, sistemas operativos, formatos de datos, rapidez de computación o carga de procesos de la red subyacente de las máquinas participantes.

PVM utiliza un modelo de computación basado en la comunicación de tareas (tasks) paralelas cooperativas a través de un mecanismo asincrónico de paso de mensajes. Cada tarea se construye como un proceso secuencial que se comunica con los demás a través de invocaciones a la biblioteca PVM, la cual constituye una interfaz de programación que provee primitivas de comunicación, sincronización y control de procesos concurrentes.

La Máquina Virtual en sí es una red de "daemons" (programas residentes) en la que cada cual se instala en un host específico y se responsabiliza por el control de las tareas que sobre dicho host se ejecutan. Un proceso especial, la consola de PVM, es el front - end desde el cual el usuario levanta la máquina PVM, agrega y elimina hosts, ejecuta programas y, en general, controla la ejecución de las tareas distribuidas en la máquina virtual a través de un sencillo intérprete de comandos.

3. Requerimientos

El Ambiente de Desarrollo debe proveer el conjunto de funcionalidades que se detallan a continuación :

1. **Edición**. Proveer una forma sencilla para invocar un editor estándar de Windows en el cual se codificará los módulos de la aplicación paralela. Se mantendrá un área de trabajo del usuario, donde se almacenará el código fuente y el script de compilación (makefile) de la aplicación paralela.
2. **Generación de Código**. Ante la petición de generación de código por parte del usuario, el Ambiente Cliente de Desarrollo enviará automáticamente al Servidor las fuentes de la aplicación paralela y el script de compilación a un área reservada para tal usuario dentro del Servidor.
3. **Mapeo y ejecución de procesos sobre la red de máquinas de PVM**. El Ambiente proveerá la facilidad de elección de la(s) arquitectura(s) objetivo(s) o el (los) host(s) objetivo(s) donde se ejecutará la aplicación paralela.
4. **Depuración**. Luego de cada compilación en el Servidor, el usuario podrá ver por pantalla una lista de los errores encontrados durante esta fase.
5. **Control y Monitoreo de Entrada / Salida**. El Ambiente proveerá un subambiente independiente de comunicaciones asincrónicas que conectan al usuario directamente con su aplicación paralela en tiempo de ejecución.
6. **Amistosidad**. El Ambiente incorporará una Interfaz de Usuario Final gráfica, donde el usuario interactuará con una Consola a través de controles gráficos como campos de texto, botones, menús colgantes, y otros.

4. Arquitectura del Sistema.

El sistema se considerará "cliente / servidor". Esto se traduce en que un host central albergará al sistema servidor que se comunica con la máquina PVM, y al que se podrá conectar uno o varios clientes simultáneos a través de un enlace TCP/IP.

El sistema será diseñado para proveer el soporte básico para que un usuario desarrolle su aplicación PVM en un ambiente desktop remoto, lo compile y lo ejecute en el host que alberga el punto central de la máquina PVM, a través de un servidor (específicamente, un "Administrador de Sesiones").

4.1 Arquitectura

El Ambiente de Desarrollo se particionará en tres subsistemas globales (ver Figura 1), donde cada cual tendrá características claramente determinadas : el Administrador de Sesiones, que se encarga de dar servicios de compilación y ejecución en PVM de algoritmos, así como de

dar un servicio básico de mensajería; la Interfaz del Usuario, que presenta la consola de trabajo remota, y el Subsistema de Comunicaciones que enlaza los clientes con el servidor mediante un protocolo de comunicaciones de sesión y un sistema de mensajería asincrónica entre clientes para efectos de entrada/salida y monitoreo. En la Figura, "mensajes" denota el flujo asincrónico de mensajes entre clientes y "comandos de sesión" indica los comandos dirigidos al Administrador y las consiguientes respuestas.

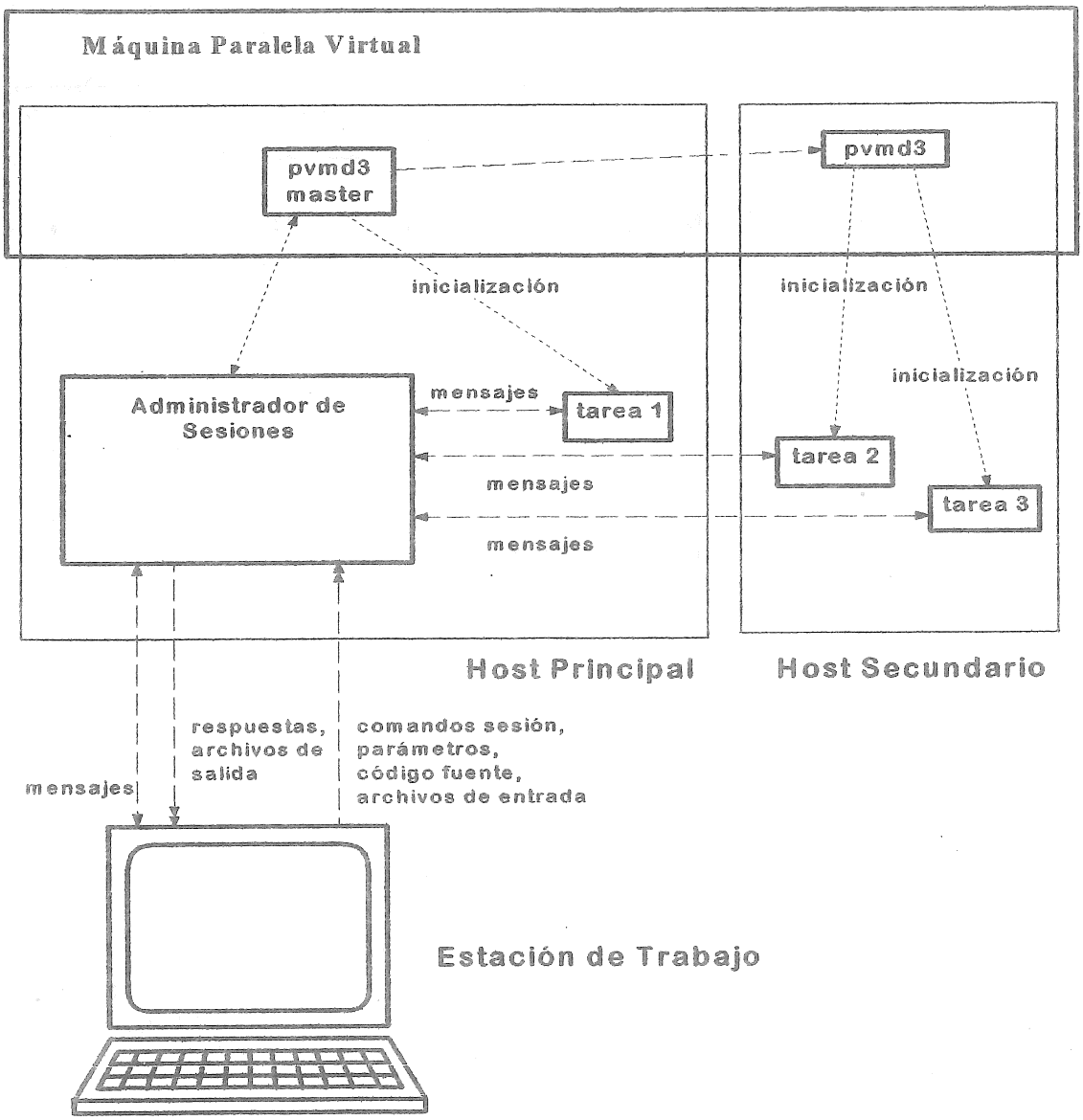

Figura 1: Esquema del Ambiente de Desarrollo.

4.2 Definición y establecimiento de funcionalidades.

El componente **Servidor** del sistema está formado por el Administrador de Sesiones y sus submódulos, los cuales proveerán los siguientes servicios durante cada sesión :

1. Inicio, mantención y término de Sesión
2. Mantención de un entorno privado para el cliente
3. Envío y recepción de archivos
4. Compilación y ejecución de las aplicaciones del usuario
5. Soporte básico de mensajería entre clientes

El Administrador de Sesiones utiliza dos submódulos principales: Interpretación de Protocolo y Gestión de Sesiones Concurrentes. La interacción entre estos componentes, los clientes y la máquina PVM puede apreciarse en la Figura 2.

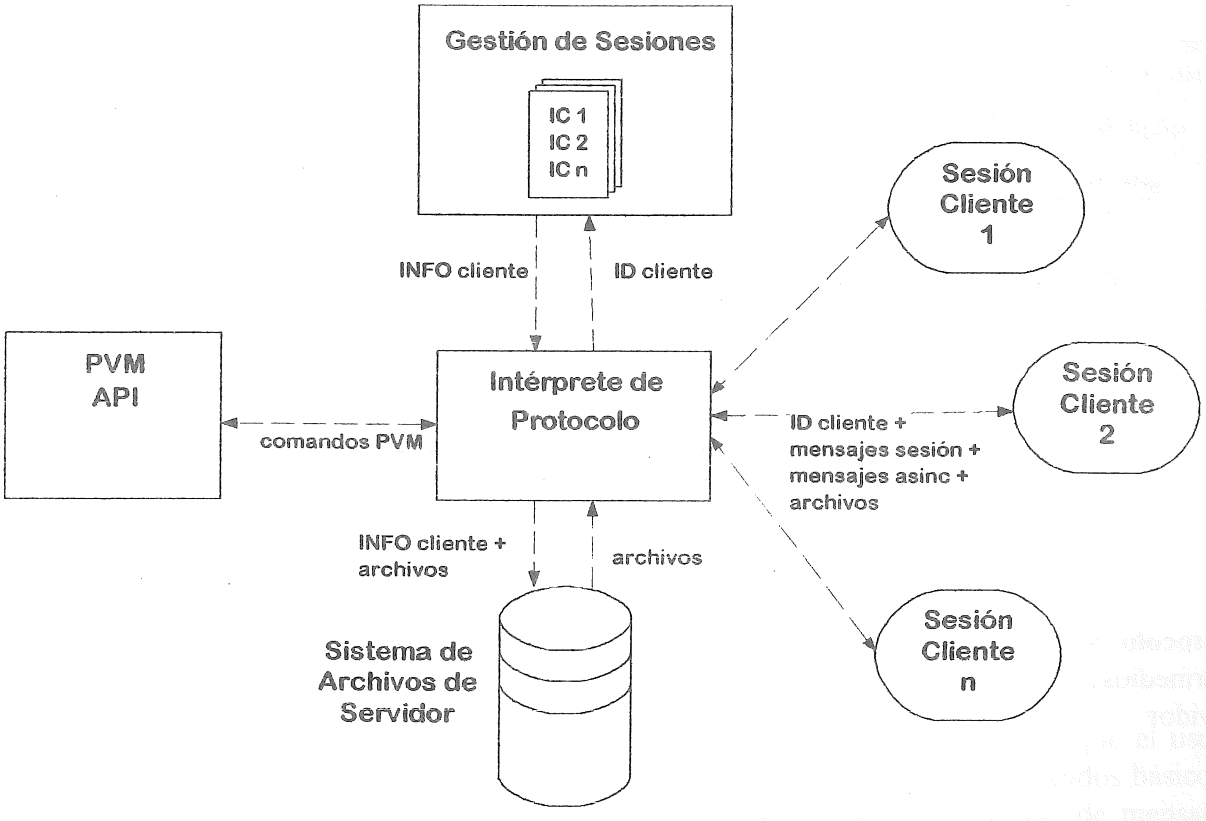

Figura 2: Módulos principales del Administrador de Sesiones.

Se optó por utilizar stream sockets de dominio Internet para comunicar el Administrador con sus clientes, debido a que este tipo de enlace ofrece un flujo de datos confiable, bidireccional, secuencial y sin duplicados.

Sobre este nivel primario de conexión de socket, se define un conjunto de protocolos de sesión, cuya jerarquía se puede apreciar en la Figura 3, y que al menos al nivel de conexión debe ser soportado por todos los elementos del sistema, tanto los clientes como el servidor.

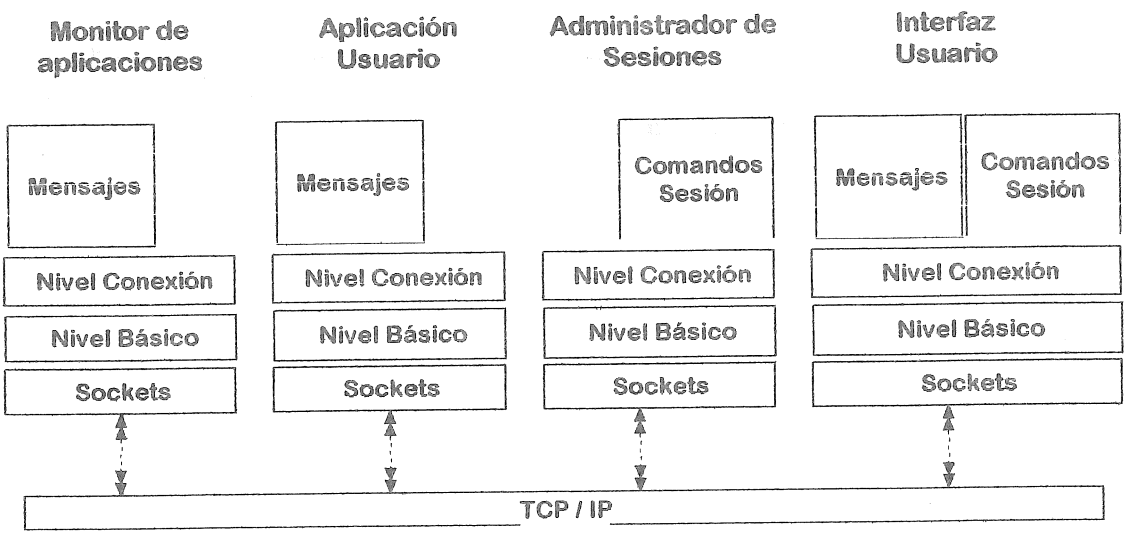

Figura 3: Jerarquía de los protocolos de Sesión.

Protocolo de Nivel Básico : establece como comando una palabra (string sin espacios intermedios) de un largo máximo de 255 caracteres, el cual se transmitirá desde el cliente hacia el servidor.

Protocolo de nivel de Conexión : establece que al momento de iniciar la conexión con el Administrador de Sesiones, el cliente debe enviar por el canal un string que identifique la sesión que está abriendo, luego de la cual el servidor responderá con el indicador de sesión iniciado. El fin de sesión se especifica con el comando único ***fin***, ante el cual el servidor también responde con el string ***fin***.

Protocolo de Comandos de Sesión : a continuación se detallarán los comandos para invocar las funciones del Administrador de Sesiones. Estos comandos se denotarán de la forma:

Cliente envía
```
Comando argumento1 argumento2 ... argumentoN
```

Para indicar que el cliente *recibe* respuesta por parte del servidor, se usará la notación

Cliente recibe
```
linea1 linea2 ... lineaN
```

Los comandos para invocar las funciones del Administrador de Sesiones son

Comando	Descripción
env	Envío de archivo de texto desde el ambiente del cliente hacia su área privada en el sistema de archivos del servidor
rec	Recuperación de archivo de texto desde el área privada del cliente en el sistema de archivos del servidor, hacia el ambiente del cliente
con	Construcción de la aplicación (compilación utilizando Makefile provisto por el usuario).
res	Resultado de la última construcción de la aplicación.
ini	Iniciación de la aplicación del usuario (ejecución)
cnf	Configuración de máquina PVM
prc	Procesos actualmente en ejecución dentro de la máquina PVM

Protocolo de Mensajes entre Clientes: determina los comandos para invocar las funciones de mensajería del Administrador de Sesiones.

mas Envío de mensaje asincrónico a otro cliente en sesión.

4.3 Interfaz con el Usuario Final.

La Interfaz con el Usuario Final es una aplicación en el entorno desktop que el usuario final utiliza para acceder al Ambiente de Desarrollo. Esta Interfaz provee los comandos básicos de envío y recepción de archivos, construcción, depuración, configuración y envío de mensajes a otros clientes, a través de módulos internos de lectura de comandos e interpretación de protocolo. Como se aprecia en la Figura 4, estos módulos utilizarán la API de Interfaz para la comunicación con el Administrador de Sesiones.

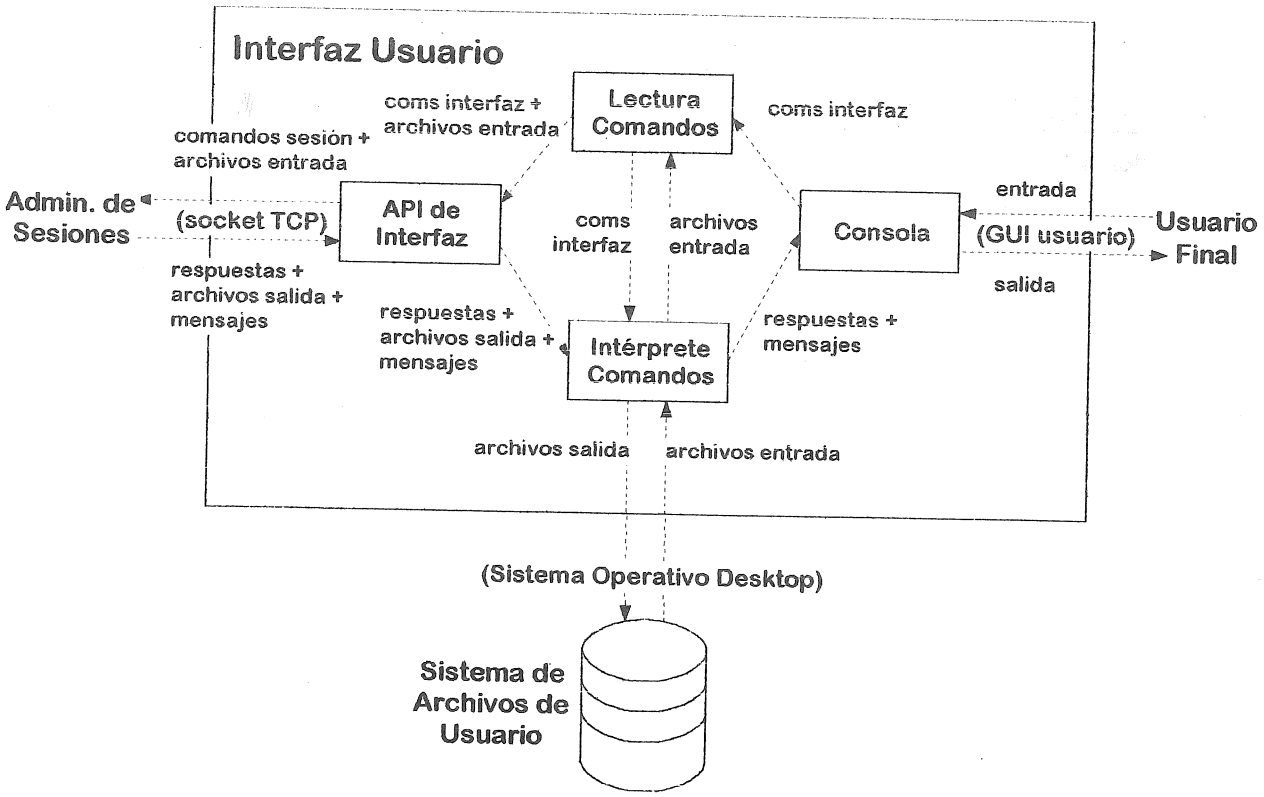

Figura 4: Diseño de la Interfaz del Usuario Final.

5. Experiencia

El sistema construido se le llamó V97. Para su funcionamiento, los hosts donde se ha de instalar deben cumplir los siguientes requisitos básicos:

- **Componente servidor**: Sistema Operativo Solaris 2.5.1, tener instalado PVM3, tener instalado un compilador de lenguaje C.
- **Componente cliente** (Interfaz Usuario): Sistema Operativo Windows 95 o superior, con soporte de TCP/IP, Visual Basic 3.0

Para la programación de aplicaciones, debe usarse un archivo "makefile" en el directorio de trabajo local del cliente que contiene el script de compilación de la aplicación. El Administrador de Sesiones, ante la orden de Construir, invocará sólo ese script, el que ya contiene todas las indicaciones de construcción de la aplicación.

La implantación actual tiene las siguientes limitaciones :

- El usuario debe generar el script de compilación.
- El servidor mantiene un área común de trabajo para todos los usuarios.
- Debido al uso de Visual Basic 3.0, los nombres de todos los archivos de usuarios deben tener el formato 8.3.
- La administración del servidor no ha sido soportada.
- No existe un control de seguridad interna del servidor.

Para validar el modelo, se realizó el algoritmo numérico paralelizado que calcula el valor de π. Este algoritmo tiene la ventaja de que puede ser fácilmente configurado en la forma de un maestro y varios esclavos, y sirve para ver cómo se distribuye una gran cantidad de procesos sobre una red de computadores.

Se comprobó que Visual Basic 3.0 impone serias limitaciones a la velocidad de comunicaciones vía sockets. Manualmente hubo que calibrar la rutina de envío de comandos desde Visual Basic para que no superara los 3.6 Kbytes/segundo, de lo contrario se perdían paquetes de comandos. Para la recepción de respuesta no hubo problemas de comunicación.

Se probó el caso de que un cliente enviara strings aleatorios hacia el servidor, mientras otro cliente mantenía una sesión normal. El servidor presentó un funcionamiento normal, y mientras la primera sesión presentó un comportamiento aleatorio, fue posible ingresar un tercer cliente al sistema.

6. Conclusiones

El enfoque Cliente / Servidor resultó ser de gran utilidad como base conceptual para guiar el proceso de diseño. Facilitó la descomposición funcional y la asignación de tareas específicas a componentes determinados del sistema, produciendo como resultado un modelo cuyo nivel de abstracción le permite materializarse de varias formas, manteniendo en esencia su característica básica de sistema de acceso múltiple (multiusuario).

El proceso de desarrollar el diseño de un sistema de software que contiene múltiples funcionalidades, características y perspectivas de crecimiento, y que involucra el nivel de complejidad propio de un sistema de arquitectura cliente / servidor, es factible de realizar en un período de tiempo y con una cantidad de esfuerzo previsibles sólo si se ha determinado en forma exacta los requerimientos básicos que se exigirán al sistema.

El desarrollo conceptual que llevó al documento final de diseño, y la construcción del producto de software pueden servir como apoyo para otros trabajos de interés, entre los cuales se

puede mencionar:

- La formulación de nuevos protocolos de comunicación.
- La generación de un sistema de monitoreo que use como estándar de comunicación el protocolo de pasada de mensajes entre clientes desarrollado en el presente trabajo.
- La construcción de una Interfaz de Usuario Final en un ambiente distinto de Windows. Por ejemplo, puede desarrollarse la Interfaz en Java, pues este lenguaje incorpora el uso de sockets TCP. Así, puede existir un repositorio centralizado para aplicaciones Java, por ejemplo a través de un sitio WWW dedicado a tal efecto, y entre tales aplicaciones se puede almacenar la Interfaz de Usuario del Ambiente de Desarrollo.

Referencias

1. Douglas E. Comer, "Internetworking with TCP/IP: Principles, Protocols, and Architecture", Addison-Wesley, 1987

2. Peterson y B. Davie, "Computer Networks : A Systems Approach", Morgan Kaufmann Publishers, 1996

3. Al Geist, Adam Beguelin, Jack Dongarra, et al, "PVM User's Guide and Reference Manual", Oak Ridge National Laboratory. Oak Ridge, Tennessee 37831, Septiembre de 1994. Correo Electrónico: mailto://pvm@msr.epm.ornl.com

4. Samuel J. Leffler, Robert S. Fabry, et al, "Computer Systems Research Group. Deparment of Electrical Engineering and Computer Science". University of California, Berkeley. Berkeley, California 94720.

5. Steve Miller, Chris Torek, ""An Advanced 4.3BSD Inter Process Communication Tutorial", Heterogeneous Systems Laboratory, Department of Computer Science University of Maryland, Publicado en Diciembre de 1991.

6. Ling Shi, "Development of A Structured and Distributed Performance Monitoring System", Thesis for the research Degree of Master of Science in the Department of Computer Science at James Cook University of North Queensland, Australia, 1995.

7. Cañas, J. Atkinson, M. Solar," Soporte de Software de Comunicación para la Programación de Multicomputadores sobre Redes de Hipercubos", XX Conferencia Latinoamericana de Informática, Ciudad de México, México, 1994.