DISEÑO Y DESARROLLO DE UNA APLICACIÓN DE IMPRESIÓN REMOTA PARA USUARIO FINAL A TRAVÉS DE INTERNET, BASADA EN EL MODELO DE COMUNICACIÓN CLIENTE/SERVIDOR.

WILMER PEARSON ARRIETA

GLENDYS B. BELTRÁN SEPÚLVEDA

CORPORACIÓN UNIVERSITARIA TECNOLÓGICA DE BOLÍVAR

FACULTAD DE INGENIERÍA DE SISTEMAS

CARTAGENA

2000

DISEÑO Y DESARROLLO DE UNA APLICACIÓN DE IMPRESIÓN REMOTA PARA USUARIO FINAL A TRAVÉS DE INTERNET, BASADA EN EL MODELO DE COMUNICACIÓN CLIENTE/SERVIDOR.

WILMER PEARSON ARRIETA

Código: 9505063

GLENDYS B. BELTRÁN SEPÚLVEDA

Código: 9405086

Trabajo de Grado presentado como requisito para optar al título de Ingenieros de Sistemas

Director:

CARLOS A. VALENCIA MARTÍNEZ

Ingeniero Electricista

CORPORACIÓN UNIVERSITARIA TECNOLÓGICA DE BOLÍVAR

FACULTAD DE INGENIERÍA DE SISTEMAS

CARTAGENA

2000

Cartagena, 19 de Abril de 2000

Señores:

COMITE DE EVALUACIÓN DE PROYECTOS

CORPORACION UNIVERSITARIA TECNOLÓGICA DE BOLÍVAR

L.C.

Respetados Señores:

A petición de los estudiantes **WILMER PEARSON ARRIETA Y GLENDYS BEATRÍZ BELTRÁN SEPÚLVEDA**, he aceptado participar como director de tesis, para la elaboración del proyecto **"DISEÑO Y DESARROLLO DE UNA APLICACIÓN DE IMPRESIÓN REMOTA PARA USUARIO FINAL A TRAVÉS DE INTERNET, BASADA EN EL MODELO DE COMUNICACIÓN CLIENTE/SERVIDOR"**, como trabajo de grado para optar el título de Ingenieros de Sistemas.

Atentamente,

Ing. CARLOS A. VALENCIA M.
CC# 73.148.708 de Cartagena

Cartagena, 19 de Abril de 2000

Señores:

COMITE DE EVALUACIÓN DE PROYECTOS

CORPORACION UNIVERSITARIA TECNOLÓGICA DE BOLÍVAR

L.C.

Respetados Señores:

Por medio de la presente nos permitimos presentar a ustedes, para que sea puesto a consideración, el estudio y aprobación del trabajo de grado que lleva por nombre **"DISEÑO Y DESARROLLO DE UNA APLICACIÓN DE IMPRESIÓN REMOTA PARA USUARIO FINAL A TRAVÉS DE INTERNET, BASADA EN EL MODELO DE COMUNICACIÓN CLIENTE/SERVIDOR"**, como trabajo de grado para optar el título de Ingenieros de Sistemas.

Agradeciendo de antemano la atención prestada.

Atentamente,

WILMER PEARSON ARRIETA	**GLENDYS B. BELTRÁN SEPÚLVEDA**
9505063	**9405086**

Cartagena, 19 de Abril de 2000

Ingeniero

GONZALO GARZÓN

Decano de la Facultad de Ingeniería de Sistemas.

Corporación Universitaria Tecnológica de Bolívar.

Respetado Ingeniero:

Por medio de la presente le hacemos entrega formal del proyecto de grado titulado **"DISEÑO Y DESARROLLO DE UNA APLICACIÓN DE IMPRESIÓN REMOTA PARA USUARIO FINAL A TRAVÉS DE INTERNET, BASADA EN EL MODELO DE COMUNICACIÓN CLIENTE/SERVIDOR",** para su aprobación.

Cordialmente,

WILMER PEARSON ARRIETA
9505063

GLENDYS B. BELTRÁN SEPÚLVEDA
9405086

Nota de aceptación

_____

_____

_____

_____
Presidente del jurado

_____
Jurado

_____
Jurado

Ciudad y fecha: Cartagena, Abril 19 de 2000

La Corporación Universitaria Tecnológica de Bolívar, se reserva el derecho de propiedad intelectual de todos los trabajos de grado aprobados y no pueden ser explotados comercialmente sin su autorización.

DEDICATORIA

A mis padres, Hesterlyna y Adalberto, por su gran dedicación, sacrificio y confianza en mis capacidades. Sin su apoyo nada de esto hubiese sido posible. Gracias por ayudarme a realizar este hermoso sueño.

A mi hermana, Mirsshan y su familia, por ese apoyo incondicional que tanto necesité en momentos difíciles.

Los ama,

Glendys Beatríz Beltrán Sepúlveda

DEDICATORIA

A mis padres, Julia Arrieta y Fernando Pearson, por todo su amor y apoyo, por mantener siempre viva la esperanza de este triunfo, y por creer siempre en mis capacidades.

A mis Hermanos, Harry, Fernando y Erick. Que este sueño que hoy se hizo realidad sea el impulso para el inicio y culminación de sus metas.

Que Dios los bendiga

Wilmer Pearson Arrieta

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a:

- *Carlos Alberto Valencia Martínez. Director del presente proyecto.*

  Gracias por ese apoyo constante e incondicional que siempre nos brindaste.

- *Gonzalo Garzón, Juan Carlos Mantilla y Moisés Quintana. Profesores de la CUTB. Gracias por todo el tiempo y el esfuerzo invertido en nosotros.*

- *Y a todas aquellas personas que de una u otra forma colaboran con la culminación de este proyecto de grado.*

Gracias a todos, que Dios siempre los acompañe y permita que sigan colaborando con todos los que un día necesitamos de ustedes.

Gracias a DIOS, por permitir esta dicha en nuestras vidas.

GLENDYS BELTRÁN SEPÚLVEDA **WILMER PEARSON ARRIETA**

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

RESUMEN

Los ambientes computacionales basados en la arquitectura Cliente/Servidor, ofrecen la posibilidad de optimizar el manejo de la información y compartir recursos de red con otros usuarios. Una definición de la arquitectura Cliente/Servidor puede ser la siguiente: Distribución de los recursos computacionales a lo largo y ancho de una organización, pero con una administración central, como un todo único.

En el esquema Cliente/Servidor la computadora de cada usuario (llamada cliente) sirve para preparar una demanda de información a cualquiera de las computadoras que proporcionan información (llamadas servidores). Esta información se transmite a través de la red y es recibida y procesada por el servidor, quien responde la demanda del cliente que la solicitó. Los clientes y los servidores pueden estar conectados a una red local o a una red amplia, como la que se puede establecer entre dos redes de una empresa, o a una red mundial como Internet.

La comunicación Cliente /Servidor es realizada a través de Sockets, que no son mas que puntos de comunicación entre procesos por el cual se puede emitir o recibir información. Las características principales de los sockets están referidas al dominio y tipo del mismo. En relación con el dominio, UNÍX ofrece dos dominios para sockets, el dominio Internet y el dominio UNÍX; el primero, puede ser usado por comunicaciones en todo el mundo, mientras que el segundo es solamente aplicable en un sistema UNÍX. En cuanto al tipo de sockets, los más comúnmente utilizados son los de flujo y trama.

Para que dos computadores conectados en cualquier parte del mundo puedan comunicarse entre sí estos deben cumplir con ciertas convenciones que especifican como debe realizarse el intercambio de datos; dichas convenciones son conocidas como Protocolos de comunicación. Los protocolos establecen una descripción formal de los formatos que deberán presentar los mensajes para poder ser intercambiados por equipos de cómputo; además definen las reglas que ellos deben seguir para lograrlo. Estos están presentes en todas las etapas necesarias para establecer una comunicación entre computadoras, desde aquellas de más bajo nivel (la transmisión de un flujo de bits a un medio físico) hasta aquellas de más alto nivel (compartir o transferir información desde una computadora a otra en la red).

Una de las ventajas de las redes de computadoras fue el poder compartir los recursos de software y hardware con lo cuál se diseñaron aplicaciones que aprovecharan estos beneficios, como por ejemplo los servidores de Impresión remota.

La impresión remota permite manipular impresoras localizadas en otros computadores, como si estas fueran locales. En la red UNIX, la impresión remota es completamente invisible o transparente. De este modo, el usuario no sabe con exactitud a que máquina UNIX está conectada la impresora por la

cual se va a enviar la impresión deseada. Aunque por supuesto, si se tiene que saber donde esta la impresora. Todo esto es manejado a través de un demonio de impresión llamado lpd, quien se encarga de gestionar los procesos de impresoras remotas.

Al igual que otros sistemas operativos, el sistema operativo UNIX y en consecuencia LINUX, es un conjunto de programas de utilidad y un conjunto de instrumentos que permiten al usuario conectar y utilizar esas utilidades para construir sistemas y aplicaciones. Al conjunto de programas que componen UNIX y que se encargan de proporcionar los recursos del sistema y de coordinar todos los detalles internos de la computadora se les llama en conjunto *Sistema Operativo* o *Kernel*. UNIX se caracteriza por ser un sistema "*Multiusuario*" porque permite que dos o más personas utilicen la computadora al mismo tiempo. Esta es una de las grandes ventajas que se aprovechan de este sistema operativo y que le han brindado un gran auge mundial.

LINUX, maneja la impresión remota a través de comandos llamados comandos lp*, La forma más simple de imprimir en el sistema operativo Linux es enviar el fichero a ser impreso directamente al dispositivo de impresión.
Para el propósito de la seguridad, sólo el usuario root y los usuarios de su mismo grupo como el demonio de impresión son capaces de escribir directamente a la impresora. Es por esto por lo que se tienen que usar comandos como lpr, lprm y lpq para acceder a la impresora.

Todos estos comandos de impresión anteriormente mencionados, son los encargados de manejar los parámetros y características de impresión, pero solo para el sistema LINUX, para aprovechar estos servicios en otras plataformas es necesario crear una aplicación en un lenguaje multiplataforma que permita aprovechar odas estas ventajas de este sistema operativo desde el sistema operativo propio. Por todo esto y muchas otras razones se Diseñó e implementó **PrintSerX 1.0**, que es una aplicación basada en el modelo Cliente /Servidor que maneja las características de impresión de LINUX en Windows y que trabaja a través de Internet o en una red LAN. Para que este proyecto se llevara a cabo fue necesario programarlo en lenguaje JAVA, aprovechando así una de sus múltiples ventajas, Lenguaje Multiplataforma.

A través del tiempo, la tecnología y en particular los computadores, han significado un cambio radical en la vida del hombre. Anteriormente, todos los procesos de las entidades o empresas se realizaban de manera manual; esto hacía que los procesos fueran largos y aburridos. La información era desorganizada y de muy difícil manipulación. Hoy en día, los procesos son manejados a través de modernos sistemas que organizan la información y permiten al usuario tenerla en el momento que la necesite.

Todas estas mejoras no se hubiesen logrado sin la incursión de las redes. Las redes brindaron a los usuarios de los sistemas la posibilidad de comunicarse, transmitir información y compartir recursos de red con otros usuarios. Es más fácil justificar el costo de la adquisición de recursos compartidos de alta calidad, como las impresoras, cuando un gran número de usuarios puede acceder simultáneamente a ellos mediante la red. Aunque las redes brindaban múltiples ventajas a los usuarios, era claro que inicialmente estos solo podían comunicarse si compartían el mismo sistema operativo. Con el paso del tiempo y otros avances tecnológicos, algunos sistemas operativos brindaron a los usuarios opciones de interoperabilidad; los sistemas operativos podían comunicarse entre ellos si contaban con algunas características que los hacían compatibles.

Hoy en día, la mayoría de sistemas operativos permiten al usuario del sistema comunicarse entre sí, sin importar en que sistema operativo estén trabajando. Todo esto gracias a software o programas que proporcionan ventajas de comunicación e interacción entre diferentes sistemas operativos.

Con la presencia de diversos sistemas operativos, es posible encontrar características y/o ventajas en un sistema operativo ausente en otros; lo que genera la necesidad de desarrollar mecanismos para aprovechar dichas ventajas sin necesidad de cambiar de plataforma.

Aprovechando las ventajas de impresión presentes en el sistema operativo LINUX, se originó la idea de explotar dichas ventajas desde otro sistema operativo, debido a que LINUX es nuevo y la mayoría de la gente no trabaja con él. De este modo se desarrolló PrintSerX 1.0, que es una aplicación de Impresión remota para usuario final a través de Internet que aprovecha las características de impresión de LINUX en la plataforma Windows.

En el desarrollo del presente proyecto de investigación se presentan una serie de aspectos que describen de manera detallada que es PrintSerX 1.0. Este software permite la integración e interacción de dos sistemas operativos, que hasta la fecha parecía incompatibles, pero a través de algunas herramientas ha sido posible comunicarlos.

El presente trabajo de investigación consta de seis capítulos conformados de la siguiente manera:

Capítulo 1: Descripción del proyecto. Contiene toda la información relacionada con los aspectos preliminares de la investigación. Hace una descripción general del proyecto, con temas de gran importancia para la buena comprensión de los siguientes capítulos, como son Antecedentes, El problema de investigación, Objetivos, entre otros.

Capítulo 2: Estrategias metodológicas de la investigación. Presenta la información teórica de la investigación. Trata todos los temas, que de una u otra forma, intervinieron en la elaboración del presente proyecto. Algunos temas relevantes en el desarrollo de la investigación y tratados en este capítulo son: Modelo Cliente/Servidor, Impresión remota, LINUX y Java. Además contiene una descripción de las fuentes de investigación utilizadas.

Capítulo 3: Diseño de la aplicación. En este capítulo se realiza una descripción detallada de los diferentes módulos que integran la aplicación y su interacción entre sí. Todo esto acompañado de gráficos que permiten al lector un mejor entendimiento del mismo.

Capítulo 4: Diseño computacional. Trata los aspectos básicos e iniciales para el diseño y desarrollo del Software, como son estructuras lógicas, diccionario de datos y diagramas de flujo.

Capítulo 5: Recomendaciones. Este capítulo contiene una serie de sugerencias y proyectos futuros que pueden desarrollarse a partir del presente proyecto.

1. DESCRIPCIÓN DEL PROYECTO

1.1 ANTECEDENTES

Los orígenes de las redes de computadoras se remontan a los primeros sistemas de tiempo compartido[1], al principio de los años sesenta, cuando una computadora era un recurso caro y escaso. La idea que encierra el tiempo compartido es simple, puesto que muchas tareas requieren solo una pequeña fracción de la capacidad de una gran computadora, se sacara mayor rendimiento de esta, si presta servicios a mas de un usuario al mismo tiempo. Del tiempo compartido a las redes hay solo un pequeño escalón.

Una vez demostrado que un grupo de usuarios mas o menos reducido podía compartir una misma computadora, era natural preguntarse si muchas personas muy distantes pudiesen compartir los recursos disponibles (discos, terminales, módems, fax, impresoras, e incluso programas especializados y bases de datos) en sus respectivas computadoras de tiempo compartido.

La forma en que las redes son usadas ha estado cambiando y han afectado la forma de trabajo, incluso a los académicos. El antiguo modelo de una gran computadora centralizada, ya es cosa del pasado. Ahora la mayoría de las instalaciones tienen diferentes tipos de computadoras, desde computadoras personales y estaciones de trabajo, a super computadoras.

Las computadoras, por lo general, están configuradas para realizar tareas particulares. Aunque la gente suele trabajar con una computadora especifica, las computadoras pueden llamar a otros sistemas en la Red para servicios especializados. Esto ha dado origen al modelo de servicios de Red "CLIENTE/SERVIDOR" y a que los recursos puedan ser compartidos de una manera mucho más amplia y a mayores distancias.

[1] Época inicial de las redes. Se empezaban a compartir los recursos computacionales.

El Trabajo a distancia entre instituciones y personas muy diversas, separadas geográficamente, ha recibido un gran impulso gracias a la introducción del fax, del correo electrónico, y servicios como transferencia de archivos, conexión remota (TELNET) e impresión remota. La impresión remota permite que un usuario pueda accesar impresoras sobre otras computadoras como si ellas estuviesen directamente conectadas a su computadora. Esto está acelerando el ritmo del intercambio a tal punto que podemos plantearnos acciones concretas e investigaciones de todo tipo coordinadas a distancia.

Las nueva tecnologías permiten trabajar sin salir de nuestras casas o lugares de trabajo. El teletrabajo[2] ha dejado de ser un mito lejano. Hoy en día, en muchas partes del mundo, el trabajo a distancia es la mejor prueba de que las redes y con ellas el Internet, se han convertido en una necesidad y no una obligatoriedad.

1.2 EL PROBLEMA DE LA INVESTIGACIÓN

1.2.1 Problema

Diseño y desarrollo de una aplicación de impresión remota para usuario final de Internet, basada en el modelo de comunicación Cliente/Servidor[3].

1.2.2 Descripción del problema

La aplicación posee las siguientes características:

[2] Nueva forma de trabajo. Trabajo virtual,
[3] Modelo de comunicación e intercambio de datos entre dos entes, uno Cliente y otro Servidor.

1. La aplicación provee una interfaz gráfica amigable, basada en manejo de ventanas y menús que permita una fácil interacción con el usuario y fácil manejo de la misma.
2. La aplicación está basada en el modelo Cliente/Servidor.
3. La aplicación presenta facilidad para la conexión de clientes remotos, a través de Internet, a un servidor de impresión.
4. La aplicación maneja un servidor de tipo concurrente, es decir, atiende a más de un usuario a la vez, lo que origina que se maneje el concepto de colas de impresión para los usuarios que esperan por el recurso.
5. La aplicación posee métodos para el manejo de reportes de servicios, es decir, debe informarle al cliente el progreso de la impresión, presenta de manera clara los diferentes mensajes de error que puedan ocurrir, comunica al cliente si el envío de los datos o el proceso de impresión se realizó de una manera exitosa o no; esto con el fin de que el cliente tenga conocimiento de todos los detalles presentados en el proceso.
6. El proyecto está realizado en lenguaje JAVA[4], lo que nos brinda la ventaja de que la aplicación se pueda transportar a cualquier plataforma.
7. La aplicación provee ayudas en línea y respuesta a algunas preguntas frecuentes sobre Cliente/Servidor e impresión remota. Esto le da al usuario final la facilidad en el manejo de la aplicación y mejor entendimiento de la misma.

1.2.3 Justificación del problema

El Proyecto desarrollado se justifica por las siguientes razones:

[4] Lenguaje de programación multiplataforma.

1. No existe en el mercado una aplicación para usuario final que maneje servicios de Impresión Remota a través de Internet.
2. No existen en nuestro medio educativo personas que manejen la tecnología Cliente/Servidor aplicado a servicios remotos a través de Internet; es por esta razón, que a través del presente proyecto pretendemos apropiarnos de dicha tecnología y contribuir al fortalecimiento y desarrollo de grupos de investigación en ésta área.
3. El presente proyecto puede servir como base para aplicaciones o proyectos futuros para el uso de recursos remotamente (fax, módems, etc.); por ejemplo, puede ser parte de un sistema de control para el monitoreo periódico de un proceso que envíe sus resultados en forma impresa.
4. Este proyecto aplicado a grandes empresas disminuiría la dificultad de utilizar los servicios de impresión en áreas externas; por ejemplo, un usuario "satelital" (vendedor, técnico, etc.), para el que sea necesario utilizar constantemente medios de impresión, podría valerse de esta aplicación para subsanar ésta necesidad.
5. Este proyecto aplicado a instituciones educativas, entre ellas la CUTB, brindaría una alternativa diferente para el envío, recepción e impresión de trabajos o clases prácticas.
6. Este proyecto es un producto factible a ser comercializado debido a que no existe en el mercado una aplicación que presente sus características.
7. El trabajo a distancia entre instituciones y personas muy diversas, separadas geográficamente, ha recibido un gran impulso gracias a la red Internet; lo que está acelerando el ritmo del intercambio a tal punto que las

necesidades actuales en materia de impresión puedan ser coordinadas a distancia.

1.3 OBJETIVOS DE LA INVESTIGACIÓN

1.3.1 Objetivo General

Diseñar e implementar una aplicación de impresión remota para usuario final de Internet, basada en el modelo de comunicación Cliente/Servidor.

1.3.2 Objetivos Específicos

1. Desarrollar una aplicación con una interfaz gráfica amigable, basada en manejo de ventanas y menús que permita una fácil interacción con el usuario y fácil manejo de la misma.
2. Apropiación de la tecnología Cliente/Servidor a través de Internet.
3. Facilitar la conexión de clientes remotos, a través de Internet, a un servidor de impresión.
4. Implementar la comunicación entre procesos a través de dos (2) plataformas diferentes, como LINUX y Windows.
5. Proporcionar al usuario métodos para el manejo de reportes de servicios, es decir, informar al cliente el progreso de la impresión, presentar de manera clara los diferentes mensajes de error que puedan ocurrir, comunicarle si el

envío de los datos o el proceso de impresión se realizó de una manera exitosa o no.

6. Proveer ayudas en línea y respuesta a algunas preguntas frecuentes sobre Cliente/Servidor e impresión remota.
7. Contribuir a la formación de grupos de investigación en Colombia, generando información útil a otros proyectos.

1.4 RESULTADOS DE LA INVESTIGACIÓN

Al culminar nuestro trabajo de grado, se hace entrega a la Corporación Universitaria Tecnológica de Bolívar, una herramienta que permita la impresión remota a través de Internet, utilizando el modelo Cliente/Servidor. Esta herramienta ha sido desarrollada en Lenguaje Visual J++ 6.0[5] con interfaz el sistema operativo LINUX[6], utilizando el Kit de Herramientas de Desarrollo JDK para LINUX, que es un interprete de Java para LINUX.

Junto con ésta herramienta de desarrollo se entregará:

- Documento final del proyecto.
- Manual de usuario.
- Manual del sistema.
- Software debidamente documentado en formato CD.

1.5 TIPO DE INVESTIGACIÓN

[5] Lenguaje Java orientado a objetos.
[6] Sistema operativo basado en UNIX

El tipo de estudio llevado a cabo en la realización del presente trabajo corresponde a una investigación Descriptiva – Aplicada; debido a que su desarrollo conlleva o se dirige a la aplicación inmediata.

2. ESTRATEGIAS METODOLÓGICAS DE LA INVESTIGACIÓN

2.1 MARCO TEÓRICO

Desde sus inicios la tecnología de computadoras desarrolló una serie de herramientas sistematizadas que permitieron agilizar los procesos en las empresas; además, mejoraron en gran parte la administración de la información, que hasta estos momentos había sido una labor tediosa y aburrida. A pesar de los múltiples beneficios obtenidos con el uso de las computadoras, éstas presentaban el inconveniente de que resultaba muy costoso adquirir recursos para cada una de ellas.

Posteriormente con la necesidad de compartir recursos lógicos y físicos, como la impresora, el módem, etc., y la de integrar las diferentes dependencias de una organización, surgen las redes de computadoras; lo cuál hizo posible que docenas de estaciones de trabajo compartieran los recursos disponibles en la red.

Las computadoras, por lo general, están configuradas para realizar tareas particulares. Aunque la gente suele trabajar en una computadora específica, con el nacimiento de las redes, se han aumentado los servicios que se pueden prestar, esto ha dado origen a los modelos de comunicación, entre ellos al modelo Cliente/Servidor; dando una alternativa diferente y más óptima para el intercambio de información.

2.1.1 Modelo Cliente / Servidor

El modelo Cliente/Servidor alude a un proceso de intercambio colaborativo de datos entre dos o más computadoras conectadas entre sí.

Este modelo consta de tres componentes básicos: un proceso cliente, el cuál es el proceso que solicita un servicio; un proceso servidor, que es el encargado de gestionar el acceso a determinado recurso; y por último, una petición, que no es más que un mensaje del cliente solicitando el servicio.

Figura 1: Modelo Cliente/Servidor

Los términos *cliente* y *servidor* se usan tanto para referirse a los programas que cumplen estas funciones, como a los equipos donde son ejecutados dichos programas.

El programa ***cliente*** cumple dos funciones distintas: por un lado gestiona la comunicación con el servidor, solicita un servicio y recibe los datos enviados por aquel. Por otro lado, maneja la interfaz con el usuario: presenta los datos en el formato adecuado y brinda las herramientas y comandos necesarios para que el usuario pueda utilizar las prestaciones del servidor de forma sencilla.

El programa ***servidor*** en cambio, básicamente sólo tiene que encargarse de transmitir la información de forma eficiente. No tiene que atender al usuario.

Entre los procesos Cliente y Servidor, se debe establecer un protocolo de comunicaciones, el cuál debe definir:

1. Como se codifican las peticiones[7]. Los clientes y los servidores deben de estar de acuerdo en como se escriben los mensajes, en que orden van los posibles parámetros de la petición, cuántos bytes ocupan, etc.

[7] Requerimientos o tareas a ejecutar.

2. Como se sincronizan entre sí los procesos. La forma de sincronización nos dice si el cliente puede seguir adelante justo después de enviar la petición, o por el contrario tiene que esperar a que el servidor le envíe una respuesta.

La sincronización entre procesos se divide en:

- ***No bloqueante:*** el proceso cliente puede seguir trabajando, justo después de enviar la petición; por esta razón se debe definir un mecanismo para que el cliente pueda saber si la respuesta del servidor está disponible.

- ***Bloqueante:*** el cliente tiene que esperar a que el servidor envíe la respuesta.

El diálogo cliente / servidor es casi siempre bidireccional. Por un lado, el cliente envía información al servidor (el tipo de servicio solicitado más los parámetros); por otro, el servidor devuelve información al cliente (los resultados del servicio, códigos de error en caso de producirse, etc.)

Inicialmente para poder establecer la comunicación entre los procesos Cliente y Servidor se debe establecer una conexión entre los dos. El tipo de conexión indica el tipo de circuito que se va a establecer entre los dos procesos que se están comunicando; este puede ser virtual (orientado a la conexión) o datagrama (no orientado a la conexión).

- ***Conexión virtual (Orientada a la conexión):*** En este tipo de conexión se realiza el envío secuencial de bs datos debido a que existe una conexión permanente y una ruta única para todos los mensajes.

Los pasos a seguir por el Cliente y el Servidor, para realizar una conexión virtual son:

Servidor	Cliente
♦ Abre el canal de comunicaciones ♦ Publica su dirección. ♦ Espera por la conexión de algún Cliente ♦ Acepta la conexión del cliente. ♦ Lee la solicitud del cliente. ♦ Realiza la petición del cliente. ♦ Envía la respuesta al cliente.	♦ Abre el canal de comunicaciones ♦ Se conecta al servidor. ♦ Envía la solicitud al servidor. ♦ Espera la respuesta del servidor. ♦ Lee la respuesta del servidor.

Tabla # 1: Pasos para una conexión virtual (orientada a la conexión).

Estos pasos se ilustran mejor en la figura 2.

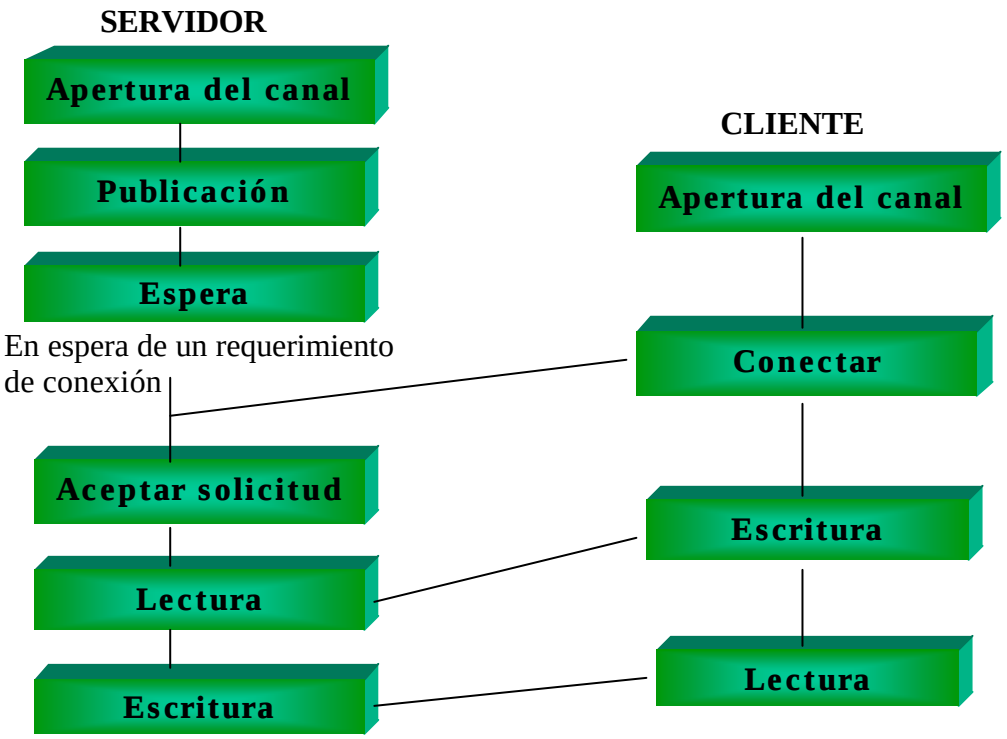


Figura 2: Pasos en la conexión virtual.

- ***Datagramas (No orientada a la conexión):*** En este tipo de conexión no necesariamente se trabajan con conexiones permanentes, y los mensajes pueden tomar rutas diferentes y llegar en un orden no secuencial.

Los pasos a seguir por el Cliente y el Servidor, para realizar una conexión virtual son:

Servidor	Cliente
♦ Abre el canal de comunicaciones. ♦ Publica su dirección. ♦ Espera por un requerimiento de conexión. ♦ Realiza la petición del cliente. ♦ Envía la respuesta al cliente.	♦ Abre el canal de comunicaciones ♦ Publica su dirección. ♦ Envía la solicitud al servidor. ♦ Espera la respuesta del servidor. ♦ Lee la respuesta del servidor.

Tabla # 2: Pasos para una conexión virtual (orientada a la no conexión).

Estos pasos se ilustran mejor en la figura 3.

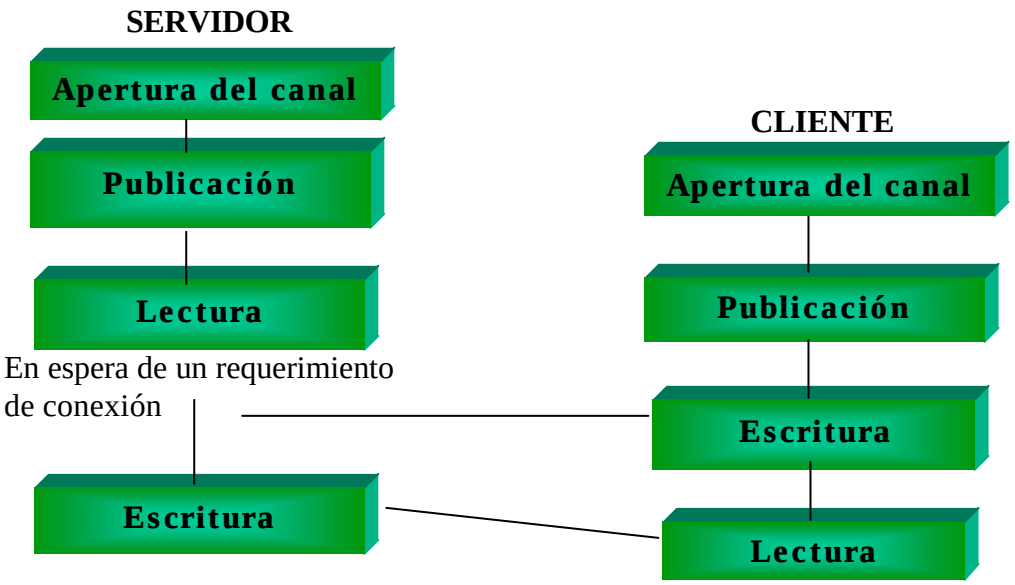


Figura 3: Pasos en la conexión por Datagramas.

De acuerdo con la forma de prestar el servicio, podemos considerar dos tipos de servidores:

- ***Servidores Interactivos:*** El servidor no solo recoge la petición de servicio, sino que él mismo se encarga de atenderla. Esta forma de trabajo presenta un inconveniente: si el servidor es lento en atender a los clientes y hay una demanda de servicios muy elevada, se originan unos tiempos de espera muy grandes.
- ***Servidores Concurrentes:*** El servidor recoge cada una de las peticiones de servicio y crea otros procesos para que se encarguen de atenderla. Este tipo de servidores solo es aplicable en sistemas multiprocesos. La ventaja que tiene este tipo de servicio es que el servidor puede recoger peticiones a muy alta velocidad, porque está descargado de la tarea de atención al cliente. En aquellas aplicaciones donde los tiempos de servicios son variables, es recomendable implementar servidores de tipo concurrente.

La comunicación entre procesos en la filosofía CLIENTE/SERVIDOR, se realiza a través de Sockets[8]: un proceso en esta comunicación actuará de proceso servidor creando un Socket cuyo nombre conocerá el proceso cliente, el cual podrá "hablar" con el proceso servidor a través de la conexión con dicho Socket nombrado.

2.1.2 Programación Con Sockets

Los *sockets* no son más que puntos o mecanismos de comunicación entre procesos que permiten que un proceso hable (emita o reciba información) con otro proceso, incluso estando estos procesos en distintas máquinas. Esta

[8] Puntos finales de enlaces de comunicaciones entre Procesos.

característica de interconectividad entre máquinas hace que el concepto de socket nos sirva de gran utilidad.

Un *socket* es un punto de comunicación entre dos agentes (procesos o personas respectivamente) por el cual se puede emitir o recibir información. La forma de referenciar un socket por los procesos implicados es mediante un descriptor del mismo tipo que el utilizado para referenciar ficheros. Todo socket viene definido por dos características fundamentales: El tipo y el Dominio del socket. El tipo del socket, indica la naturaleza del mismo, el tipo de comunicación que puede generarse entre los sockets; y el dominio del socket especifica el conjunto de sockets que pueden establecer una comunicación con el mismo.

Tipos De Sockets

Definen las propiedades de las comunicaciones en las que se ve envuelto un socket, esto es, el tipo de comunicación que se puede dar entre cliente y servidor. Estas propiedades pueden ser:

- Fiabilidad de transmisión.
- Mantenimiento del orden de los datos.
- No duplicación de los datos.
- El "Modo Conectado" en la comunicación.
- Envío de mensajes urgentes.

Un socket puede pertenecer a uno de los siguientes tipos:

- **Sockets Stream (TCP, Transport Control Protocol):** Son un servicio orientado a conexión donde los datos se transfieren sin encuadrarlos en

registros o bloques. Si se rompe la conexión entre los procesos, éstos serán informados.

El protocolo de comunicaciones con streams es un protocolo orientado a conexión, ya que para establecer una comunicación utilizando el protocolo TCP, hay que establecer en primer lugar una conexión entre un par de sockets. Mientras uno de los sockets atiende peticiones de conexión (servidor), el otro solicita una conexión (cliente). Una vez que los dos sockets estén conectados, se pueden utilizar para transmitir datos en ambas direcciones.

- **Sockets Datagrama[9] (UDP, User Datagram Protocol):** Son un servicio de transporte sin conexión. Son más eficientes que TCP, pero no está garantizada la fiabilidad. Los datos se envían y reciben en paquetes, cuya entrega no está garantizada. Los paquetes pueden ser duplicados, perdidos o llegar en un orden diferente al que se envió.

  El protocolo de comunicaciones con datagramas es un protocolo sin conexión, es decir, cada vez que se envíen datagramas es necesario enviar el descriptor del socket local y la dirección del socket que debe recibir el datagrama.

- **Sockets Raw:** Son sockets que dan acceso directo a la capa de software de red subyacente o a protocolos de más bajo nivel. Se utilizan sobre todo para la depuración del código de los protocolos.

- **Sockets Seqpacet:** Corresponde a las comunicaciones que se encuentran circunscritas en el dominio *XEROX NS.*

[9] Paquetes de Datos de información.

Los Sockets Stream y los Datagrama son los más importantes y por ende más utilizados a nivel de aplicaciones. En ciertos casos se nos presenta un problema, ¿qué protocolo, o tipo de sockets, debemos usar - UDP o TCP? La decisión depende de la aplicación cliente/servidor que estemos escribiendo.

Podemos observar en la Tabla # 3 algunas diferencias entre los protocolos TCP y

UDP para ayudar en la decisión.

Diferencias entre Sockets Stream y Datagrama	
Sockets Stream	***Datagrama***
♦ Como son orientados a conexión, tenemos que establecer esta conexión entre los dos sockets antes de nada, lo que implica un cierto tiempo empleado en el establecimiento de la conexión, que no existe en UDP. ♦ TCP no tiene límite; una vez que se ha establecido la conexión, el par de sockets funcionan como los streams: todos los datos se leen inmediatamente, en el mismo orden en que se van recibiendo. ♦ TCP es un protocolo ordenado, garantiza que todos los paquetes que se envíen serán recibidos en el socket destino en el mismo orden en que se han enviado.	♦ Cada vez que se envía un datagrama, hay que enviar también el descriptor del socket local y la dirección del socket que va a recibir el datagrama, luego éstos son más grandes que los TCP. ♦ Existe un límite en el tamaño de los datagramas, establecido en 64 kilobytes, que se pueden enviar a una localización determinada. ♦ UDP es un protocolo desordenado, no garantiza que los datagramas que se hayan enviado sean recibidos en el mismo orden por el socket de recepción. ♦ Los datagramas son bloques de información del tipo lanzar y olvidar.

Tabla # 3: Diferencias entre Sockets Stream y Datagramas

En resumen, TCP parece más indicado para la implementación de servicios de red como un control remoto (rlogin[10], telnet[11]) y transmisión de ficheros (ftp); que necesitan transmitir datos de longitud indefinida. UDP es menos complejo y tiene una menor sobrecarga sobre la conexión; esto hace que sea el indicado en la implementación de aplicaciones cliente/servidor en sistemas distribuidos montados sobre redes de área local.

Dominios de los Sockets

Indica el formato de las direcciones que podrán tomar los sockets y los protocolos que soportarán dichos sockets. El mecanismo de sockets está diseñado para ser todo lo genérico posible. El socket por sí mismo no contiene información suficiente para describir la comunicación entre procesos.

Los sockets operan dentro de dominios de comunicación, entre ellos se define si los dos procesos que se comunican se encuentran en el mismo sistema o en sistemas diferentes y cómo pueden ser direccionados.

Entre los dominios de sockets existentes se encuentran:

- ***Dominio UNIX***

  Es utilizado en protocolos internos de UNIX. Es la familia de sockets empleada para comunicar procesos que se ejecutan en una misma máquina. Esta familia no requiere que este presente un hardware especial de red, puesto que en realidad no realiza accesos a ninguna red.

  Se permiten tanto los sockets stream como los datagrama, pero no se permiten sockets de tipo Raw.

[10] Comando de UNÍX para iniciar una sesión en una terminal o en un host remoto.
[11] Programa para establecer conexión con un servidor remoto.

- ***Dominio Internet***

  Es utilizado en protocolos Internet. Es la familia de sockets que se comunican a través de protocolos, tales como TCP o UDP.

  Las comunicaciones intersistemas proporcionan acceso a TCP, ejecutando sobre IP. De la misma forma que el dominio UNIX, el dominio Internet permite tanto sockets stream como datagrama, pero además permite sockets de tipo Raw.

- ***Dominio AF_NS***

  El servidor y el cliente deben estar en una red XEROX.

- ***Dominio AF_CCITT***

  Es utilizado en protocolos CCITT y protocolos X.25.

Filosofía Cliente-Servidor

Las funciones realizadas por el cliente y servidor para establecer una comunicación usando las abstracciones de sockets se ilustran en la figura #4.

Uso De Sockets

Podemos pensar que un *Servidor Internet* es un conjunto de sockets que proporciona capacidades adicionales del sistema, los llamados *servicios*.

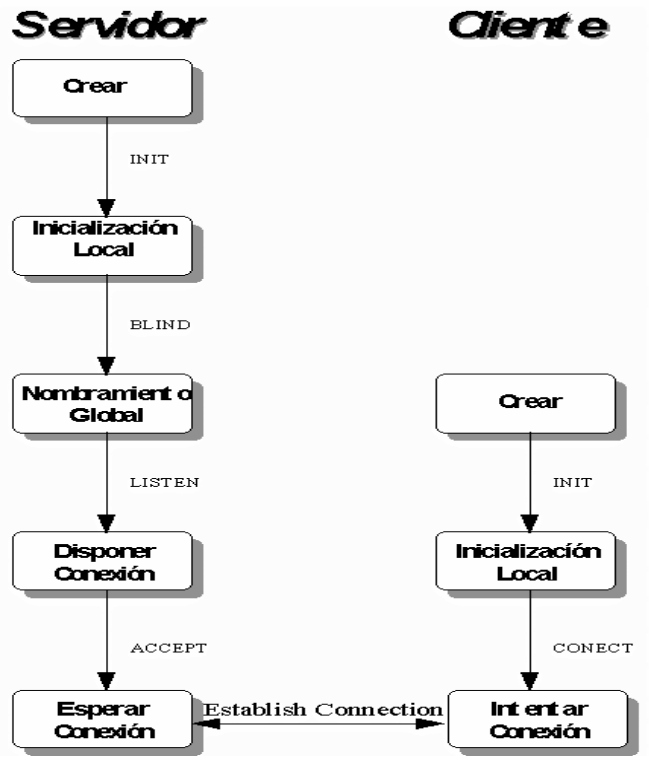


Figura 4: Esquema genérico de un Servidor y un Cliente usando Sockets.

Puertos y Servicios

Cada servicio está asociado a un *puerto*. Un puerto es una dirección numérica a través de la cual se procesa el servicio.

Las comunicaciones de información relacionada con Web tienen lugar a través del puerto 80 mediante protocolo TCP. Para implementar este servicio en Java se utiliza la clase **Socket**. Sin embargo, el servicio que coge la fecha y la hora del sistema, está ligado al puerto 13 utilizando el protocolo UDP. Un servidor que lo emule en Java usaría un objeto **DatagramSocket**.

2.1.3 Modelos De Protocolos De Comunicación

Rápidamente, con la aparición de las computadoras, surgió la necesidad de la conectividad entre equipos, con lo que las grandes compañías comenzaron a diseñar sus estándares de trabajo para permitir conectar diferentes equipos y plataformas a través de una red.

Debido a la gran diversidad de estándares diferentes, no compatibles entre si, surgieron organizaciones para intentar estandarizar todos los mecanismos de comunicación existentes, para lo cuál crearon modelos que empezaron a ser adoptados por la industria; entre dichos modelos se encuentran el modelo OSI y el modelos TCP/IP.

El modelo de comunicación OSI.

El modelo OSI (Open System Interconection) es utilizado prácticamente por la totalidad de las redes del mundo. Este modelo fue creado por el ISO (Organización Internacional de Normalización), y consiste en siete niveles o capas donde cada una de ellas define las funciones que deben proporcionar los protocolos con el propósito de intercambiar información entre varios sistemas.

Esta clasificación permite que cada protocolo se desarrolle con una finalidad determinada, lo cual simplifica el proceso de desarrollo e implementación. Cada nivel depende de los que están por debajo de él, y a su vez proporciona alguna funcionalidad a los niveles superiores. Los siete niveles del modelo OSI son los siguientes:

- ***Aplicación:*** El nivel de aplicación es el destino extremo de los datos donde se proporcionan los servicios al usuario.

- ***Presentación:*** Se convierten e interpretan los datos que se utilizarán en el nivel de aplicación.
- ***Sesión:*** Encargado de ciertos aspectos de la comunicación como el control de los tiempos.
- ***Transporte:*** Transporta la información de una manera fiable para que llegue correctamente a su destino.
- ***Red:*** Nivel encargado de encaminar los datos hacia su destino eligiendo la ruta más efectiva.
- ***Enlace:*** Enlace de datos. Controla el flujo de los datos, la sincronización y los errores que puedan producirse.
- ***Físico:*** Se encarga de los aspectos físicos de la conexión, tales como el medio de transmisión o el hardware.

El modelo de comunicación TCP/IP.

TCP/IP es el modelo común utilizado por las computadoras conectados a Internet, de manera que éstos puedan comunicarse entre sí. Hay que tener en cuenta que en Internet se encuentran conectadas computadoras de clases muy diferentes y con hardware y software incompatibles en muchos casos, además de todos los medios y formas posibles de conexión. Aquí se encuentra una de las grandes ventajas del TCP/IP, pues este protocolo se encargará de que la comunicación entre todos sea posible. TCP/IP es compatible con cualquier sistema operativo y con cualquier tipo de hardware. Actualmente, TCP/IP es un estándar en muchas plataformas, debido a que puede ser utilizado para prestar algunos servicios de la red de redes Internet.

Teniendo en cuenta que uno de los objetivos de las redes es compartir los recursos de las computadoras conectadas a ésta, se dio la necesidad de desarrollar aplicaciones a través de las cuáles se pudieran administrar los recursos remotamente, tales como fax, módem, discos, impresoras, etc.; lo que originó la creación de los servidores de impresión remota, entre otros.

2.1.4 Impresión Remota En UNIX

La impresión remota permite que un usuario pueda accesar impresoras sobre otras computadoras como si ellas estuviesen directamente conectadas a su computadora.

Los servicios de impresión en UNIX son soportados por intermedio de un manejador de spool (**lp**). Pueden configurarse con la impresora físicamente conectadas a la red o bien para impresoras conectadas a un proceso especifico. Los servicios de impresión incluyen librerías o filtros mediante las cuales los administradores pueden soportar sus necesidades. Los trabajos de impresión de un usuario final serán atendidos por estos servicios produciendo como resultado al requerimiento, la impresión sobre alguna impresora objetivo.

Cuando se arranca una máquina UNIX, también se arrancan estos servicios, activándose un demonio que atenderá las solicitudes de impresión. Cuando un usuario imprime un archivo o realiza una solicitud de impresión, esta no va inmediatamente a la impresora. En vez de ello, los archivos son encolados mediante el spooler (“Manejador de la cola”) para posteriormente ser manejados por un demonio de impresión llamado “**lpsched”**.

El spooler intercepta todos los trabajos de impresión y son alojados en un espacio en disco o directorio, cada impresora tiene un directorio asociado donde se alojaran todos los trabajos de impresión dirigidos a ella.

Además de los servicios básicos, los servicios de impresión le permiten a los administradores agregar impresoras del mismo tipo a una clase de impresora.

Esto provee una utilización óptima del recurso, haciendo que los usuarios utilicen una clase de impresora en vez de una impresora específica. Cuando esto pasa, **lpsched** envía el primer trabajo de impresión a la impresora disponible en esta clase. Otros servicios adicionales pueden ser programas de interfaces que notifiquen al usuario del estado de la impresión, y en el caso de impresora de red, se deben proveer los servicios de red que permitan una comunicación adecuada al dispositivo de impresión conectado a la red.

La forma más sencilla de imprimir bajo UNIX consiste en enviar los datos de impresión al dispositivo de impresión. El comando siguiente envía un listado de directorios a la primera impresora en paralelo (LPT1 en términos del DOS):

ls > /dev/lp0

Este método no aprovecha las capacidades de multitareas de UNIX, pues el tiempo que ocupa el comando para concluir es con relación a lo que tarda la impresora en imprimir físicamente los datos. En una impresora lenta o una a la que se desconecte, el lapso podría ser muy prolongado.

Un mejor método consiste en imprimir en forma indirecta los datos; esto es, reunir los datos de impresión en un archivo y después iniciar un proceso en

segundo plano para enviar los datos a la impresora.

En general, esta es la forma en que trabaja UNIX. Se define un área de impresión indirecta (también conocida como spool) para cada impresora. Los datos para la impresora se reúnen en el área de impresión indirecta: un archivo por cada trabajo de impresión. Un proceso en segundo plano (llamado demonio de impresión) revisa constantemente las áreas de impresión indirecta en busca de archivos nuevos que deban imprimirse. Cuando aparece uno, se envían los datos a la impresora apropiada. Si son varios los archivos que esperan, se les imprime en el orden que concluyen: el primero en llegar es el primero en salir. Por tanto, el área de impresión indirecta es de hecho una cola, y a menudo se dice que los trabajos que están en lista de espera están en *cola de impresión.* En el caso de la impresión remota, los datos se envían en forma local a un área de impresión indirecta igual que cualquier otro trabajo de impresión, pero al proceso en segundo plano se le indica que envíe los datos a una impresora particular en una máquina remota particular.

La información necesaria que requiere el demonio de impresión para efectuar su trabajo (el dispositivo físico por usar, el área de impresión indirecta en la que debe buscar, la máquina remota y la impresora para impresión remota, etc.) se guarda completa en un archivo llamado /etc/printcap.

Los Programas Importantes De Impresión

El sistema de impresión de UNIX comprende cinco programas, los que deben estar en la ubicación mostrada, ser propiedad de la raíz, pertenecer al demonio

de grupo y tener los permisos que se listan en la siguiente tabla:

Permisos de archivo	Ubicaciones de archivos
-rwsr-sr-x	/usr/bin/lpr
-rwsr-sr-x	/usr/bin/lpq
-rwsr-sr-x	/usr/sbin/lpc
-rwsr-sr-x	/usr/bin/lprm
-rwxr-s---	/usr/sbin/lpd

Tabla # 4: Ubicación de archivos de impresión

Los primeros cuatro archivos de la tabla se utilizan para someter, cancelar e inspeccionar trabajos de impresión. /etc/lpd, es el demonio de impresión.

Los aspectos importantes son que *lpr, lprm, lpc y lpq* por omisión operan en una impresora llamada *lp*. Si define una variable de ambiente llamada PRINTER (impresora), entonces se utilizará el nombre definido. Ambos nombres pueden ser sustituidos si se especifica el nombre de la impresora que ha de usarse en la línea de comandos.

COMANDO *lpr*

lpr – off line print

La sintaxis del comando lpr es:

lpr [-A] [-C clasificación] [- F formato de filtro] [-h] [-k] [-J trabajo] [-K, # copias] [-m mailto] [-P impresora] [-Q] [-r] [-R cuentas remotas] [-s] [- título] [-U usuario] [-w ancho] [filename...]

Este comando somete un trabajo a la impresora, o *pone en cola de espera un trabajo de impresión.* Lo que en realidad ocurre es que el archivo especificado por usted se copia al directorio de impresión indirecta. lpd, luego de encontrar

el archivo, se ocupa de transferir los datos a la impresora física. Si no especifica un archivo, lpr utiliza la entrada estándar.

Las opciones para el manejo de lpr son:

-A se usa para autentificar las transferencias si estas están disponibles.

-C especifica la clasificación del trabajo y además coloca la prioridad, A(la mas alta) y Z(la mas baja).

-F filtro de especificación de formato. Por defecto, la entrada es asumida por un archivo de texto estándar y el formato f es usado para esto; el dispositivo de salida es asumido para ser una simple impresora en línea.

-h ningún estándar o título para este trabajo.

-J especifica el nombre del trabajo a imprimir en una página.

-k la opción –k causa que el trabajo sea enviado directamente al servidor. Si usted mata el trabajo en medio de la creación, entonces la parte del trabajo transferido será impresa. Esta opción no se puede ejecutar con trabajos muy grandes.

-K especifica el número de copias de cada archivo a ser impresos.

-m envía un mail después que un usuario ha fallado enviando un mail.

-p salida a la impresora especifica. El valor por defecto (en orden de prioridad) lo especifica el valor –P, la variable de ambiente de la impresora, la primera entrada en la información del printcap[12], y la entrada a la impresora por defecto desde el archivo de configuración.

[12] Archivo de configuración de impresoras.

-Q Pone el nombre de la cola del spool[13] dentro del archivo de trabajo. Esta información puede ser usada por el software spooling para controlar el formato de la salida.

-R Cuentas remotas. Especifican la información de las cuentas que pueden ser usadas por un sistema remoto que imprime su salida.

-T título. Especifica el título usado por pr(1); valor por defecto al nombre del archivo.

-U esta opción se usa para especificar un nombre de usuario para el trabajo.

-W ancho. Especifica el ancho de la página para imprimir en el trabajo.

COMANDO *lpq*

lpq - Programa examinador de la cola del spool.

La sintaxis para el comando lpq es la siguiente:

lpq [-l] [-Pprinter] [# del trabajo] [usuario...]

Este comando le muestra el directorio de impresión indirecta para una impresora determinada. Uno de los importantes componentes que despliega lpq es la ID (identificación) del trabajo, la que identifica un trabajo determinado. Es necesario que especifique este número si quiere cancelar un trabajo pendiente.

lpq también muestra una jerarquía para cada trabajo que esta en cola de espera: *Activo* significa que el trabajo se está imprimiendo (o por lo menos que lpd esta tratando de imprimirlo). De otra manera, un número le muestra en que

[13] Directorio de impresión indirecta.

lugar de la cola de impresión esta el trabajo.

lpq examina el área del spooling usada por lpd para imprimir archivos en la línea de impresión, y reportar el estado de los trabajos especificados o todos los trabajos asociados con un usuario. lpq puede ser invocado sin ningún argumento o sin ningún trabajo en la cola.

Las opciones disponibles de lpq son:

-P Especifica una impresora particular, o para usar la impresora predefinida. Se interpretan todos los otros argumentos proporcionados como nombres del usuario o el número del trabajo fuera del filtro.

-l Información sobre cada uno de los archivos que comprenden la entrada del trabajo que será impreso.

Para cada trabajo sometido (es decir, invocado por lpr) lpq informa el nombre del usuario, línea actual en la cola, los nombres de archivos que comprenden el trabajo, el identificador del trabajo (un número que puede proporcionarse a lprm(1) por quitar un trabajo específico), y el tamaño total en bytes. El trabajo ordenado es dependiente del algoritmo usado para examinar el directorio spool y se supone que es FIFO (Primero en entrar Primero en salir).g

COMANDO *lprm*

lprm Borra trabajos de la cola de impresión del spooling.

La sintaxis para el comando lprm es:

lprm [-Pprinter] [-] [#trabajo...] [usuario...]

Este comando retira un trabajo de la cola de impresión; esto es, retira los archivos no impresos aún del directorio de impresión indirecta. Puede especificar una ID (identificación) de trabajo (la que se obtiene mediante el uso del comando lpq) o especificar -como la ID de trabajo; en cualquier caso, todos sus trabajos se cancelan. Si hace esto como root, todos los trabajos direccionados a la impresora se cancelan. Si es root y quiere retirar todos los trabajos de un usuario particular, debe especificar el nombre del usuario.

Las Opciones y argumentos para lprm son:

-Pprinter Especifica la cola asociada con una impresora específica. Si solo se coloca '-', lprm quitará todos los trabajos que un usuario posee. Si el superusuario emplea esta bandera, la cola del spooling, es vaciada completamente.

user Causa que lprm intente quitar cualquier trabajo que se encuentre en la cola perteneciente a ese usuario (o usuarios). Esta forma de invocar lprm sólo es útil para el superusuario[14].

#*trabajo* Un usuario puede sacar de la cola de impresión un trabajo individual especificando el número del trabajo.

Si no se dan argumentos u opciones, lprm anulará el actual trabajo activo si este pertenece al usuario que invocó lprm.

lprm matará el demonio activo, si es necesario, antes de quitar cualquier archivo del spooling de impresión. Si un demonio muere, uno nuevo es automáticamente reiniciado.

[14] Llamase así al usuario supervisor del sistema UNÍX

COMANDO *lpc*

lpc – Programa control de líneas de impresión

La sintaxis del comando es la siguiente:

lpc [comando [argumentos]]

Este comando le permite revisar el estado de las impresoras y controlar algunos aspectos concernientes a su uso. En particular, le permite iniciar e interrumpir el vaciado del área de impresión indirecta, activar o desactivar las impresoras y volver a clasificar el orden de los trabajos en una cola de impresión.

lpc es usado por el administrador del sistema para controlar el funcionamiento del sistema de líneas de impresión. Por cada línea de impresión configurada en /etc/printcap, lpc puede ser usado para:

- Deshabilitar o habilitar una impresora,
- Deshabilitar o habilitar la cola de impresión.
- Reestructurar el orden de los trabajos de la cola de impresión,
- Revisar el estado de las impresoras, y sus colas de impresión asociadas y los demonios de las impresoras.

Sin ningún argumento, *lpc* se situará en el prompt a la espera de comandos desde la entrada estándar. Si se proporcionan argumentos, *lpc* interpreta el primer argumento como un comando y los argumentos restantes como parámetros para el comando. La entrada estándar puede ser redireccionada haciendo que *lpc* pueda leer comandos desde un archivo. Las órdenes pueden ser abreviadas; la siguiente es una lista de comandos reconocidos:

- **? [comando...]**

- **help [comando...]:** Imprime una descripción corta de cada comando especificado en la lista de argumentos, o, si no se dan argumentos, una lista de los comandos reconocidos.
- **abort {todos | impresora}:** Termina inmediatamente un demonio de impresión activo en el host local y entonces desactiva la impresora (impidiendo que el nuevo demonio pueda ser empezado por lpr) para las impresoras especificadas.
- **clean {todos | impresora}:** Borra cualquier archivo temporal, archivo de datos, y archivos de control que no pueda imprimirse (es decir, no forma un trabajo de impresión completo) de la cola(s) de impresión especificada en la máquina local.
- **disable {todos | impresora}:** Deshabilita la cola de impresión especificada. Esto previene que nuevos trabajos de impresión sean colocados en la cola por lpr.
- **down {todos | impresora} mensaje...:** Deshabilita la cola de impresión especificada, desactiva la impresión y pone el mensaje en el archivo status de la impresora. Esto es normalmente usado para Deshabilitar una impresora y permitir a otros conocer por qué lpq(1) deshabilitaría la impresora e imprime el mensaje.
- **enable {todos | impresora}:** Habilita el spooling en una cola local para las impresoras listadas. Esto le permitiría a lpr(1) poner nuevos trabajos en cola.
- **exit**
- **quit:** Salir de lpc.

- **restart {todos | impresora}:** Intenta empezar un nuevo demonio de impresión. Esto es útil cuando algunas condiciones anormales causan que el demonio muera inesperadamente, sacando los trabajos de impresión de la cola. lpq informará que no hay ningún demonio presente cuando esta condición ocurra. Si el usuario es el superusuario, intente primero abortar al demonio actual.
- **start {todos | impresora}:** Habilita la impresión y empieza un demonio de impresión para las impresoras listadas.
- **status {todos | impresora}:** Despliega el estado del demonio y colas de la máquina local.
- **stop {todos | impresora}:** Detiene el demonio de impresión después que el trabajo actual finalice y deshabilita la impresora.
- **topq printer [jobnum...] [user...]:** Coloca los trabajos en el orden listado en el extremo de la cola de impresión.
- **up {todos | impresora}:** Habilita todo y empieza un nuevo demonio de impresión. Deshace el efecto de down.

COMANDO *lpd*

lpd - line printer spooler daemon

La sintaxis para *lpd* es la siguiente:

lpd [-l] [#puerto]

lpd es el demonio del área de impresión (negociante de área de spool) y normalmente es invocado en el momento de arranque de los archivos rc(). Hace un solo paso a través del archivo printcap() el archivo para averiguar

sobre las impresoras existentes. Este usa entonces las llamadas al sistema *listen* y *accept* para recibir los requerimientos de los archivos de impresión en la cola, transfiere los archivos al área de spooling, despliega la cola, o quita trabajos de la cola.

En cada caso, esto crea un proceso hijo para manejar los requerimientos y así el padre puede continuar escuchando más requerimientos.

Las opciones disponibles para lpd son:

- **-l:** La bandera –l ocasiona que lpd consigne los requerimientos válidos recibidas de la red.
- **#puerto:** Los números de puertos usados en Internet para reunirse con otros procesos son usualmente obtenidos mediante getservbyname() pero pueden cambiarse con el argumento *port #.*

El control de acceso es proporcionado a través de dos formas. Primero, todas las demandas deben venir de una de las máquinas listadas en el archivo */etc/hosts.equiv* o */etc/hosts.lpd*. Segundo, si la capacidad del rs se especifica en la entrada printcap de la impresora que está siendo accesada, lpr sólo se honrarán demandas del *lpr* para esos usuarios con cuentas en la máquina con la copiadora.

Los Directorios Importantes

Hay un solo directorio importante: el área de impresión indirecta, donde se depositan los datos por imprimirse antes de que /etc/lpd los imprima. Sin embargo, un sistema por lo general se configura con varios directorios de

impresión indirecta, uno para cada impresora. Esto facilita la administración de impresoras. Por ejemplo, mi sistema está configurado para utilizar /usr/spool/lpd como área de impresión indirecta, en la que cada impresora tiene un directorio subordinado con el mismo nombre de la impresora. Así, hay una impresora llamada printer que tiene a /usr/spool/lpd/printer como directorio de impresión indirecta.

Los directorios de impresión indirecta deben pertenecer al grupo demonio y ser de lectura-escritura y de lectura global para usuario y grupo. Esto es, después de crear el directorio, asegúrese de que tenga los permisos –rwxrwxr-x (0775) con el comando chmod.

Los Archivos Importantes

Además de los programas analizados arriba, cada directorio de impresión indirecta contiene cuatro archivos: *.seq, errs, lock* y *status*, que tienen los permisos *-rw-rw-r-*. El archivo *.seq* presenta el contador de número de archivo para la asignación de *lpr*, y el archivo *status* contiene el mensaje que debe presentar *lpc stat.* El archivo *lock* lo usa *lpd* para evitar que el mismo trate de imprimir dos trabajos en una misma impresora al mismo tiempo. Por ultimo, el archivo *errs* es un registro de fallas de impresora.

El archivo *errs* no es necesario, pero puede llamarlo siempre que quiera. Aunque el nombre se especifica en etc/printcap, el archivo debe estar presente para que lpd pueda registrarse en el archivo. Por ello, al configurar el área de impresión indirecta, el archivo se crea en forma manual.

Información sobre */etc/printcap*

/etc/printcap es un archivo de texto que puede editar con su editor favorito, debe pertenecer a la raíz y tener los permisos *–rw-r-r-.*

Una entrada de printcap describe una impresora. En esencia, una página printcap proporciona un nombre lógico para el dispositivo físico y enseguida describe la forma en que se manejaran supuestamente los datos que se envíen a ese dispositivo. Por ejemplo, una entrada de printcap define que dispositivo físico se utilizara, en que directorio de impresión indirecta se deben guardar los datos para ese dispositivo, que procesamiento previo debe efectuarse sobre esos datos, que errores del dispositivo físico deben registrarse, etc. Puede limitar la cantidad de datos que habrán de enviarse en un solo trabajo o limitar el acceso a una impresora a cierto tipo de usuarios.

Es muy conveniente tener varias entradas printcap que definan distintas formas de manejar los datos destinados a la misma impresora física. Por ejemplo, una impresora física puede soportar formatos de datos para PostScript y HP Laserjet, de acuerdo con alguna secuencia de configuración que se envíe a la impresora física antes de cada trabajo. De este modo, tiene sentido definir dos impresoras una que procese previamente los datos para adjuntar de antemano la secuencia HP Laserjet y otra que adjunte de antemano la secuencia PostScript. Los programas que generan datos HP los envían a la impresora HP, mientras que los que generan PostScript imprimen hacia la impresora PostScript.

Los programas que modifican los datos antes de enviarlos a la impresora física se conocen como *filtros.* Un filtro es capaz de no enviar ningún dato a una impresora física; esto es, el filtro desecha todo.

Campos en */etc/printcap*

Hay tantos campos que sería imposible describirlos por completo; por ello, se describirán solo los más importantes. Todos los campos en /etc/printcap (salvo los que corresponden a los nombres de las impresoras) van encerrados entre dos puntos y se les denota mediante un código de dos letras, el que va seguido por un valor que depende del tipo de campo. Además, hay tres tipos de campo: de cadena, booleanos (o de Boole) y numérico. La tabla siguiente describe los campos más comunes e importantes.

Campo	Tipo	Descripción
Lp	De cadena	Especifica el dispositivo en el que se va a imprimir.
Sd	De cadena	Especifica el directorio de impresión indirecta para Esta impresora
Lf	De cadena	Especifica el archivo en el que van a registrarse los errores de esta impresora
If	De cadena	Especifica el nombre del filtro de entrada.
Rm	De cadena	Especifica el nombre de un anfitrión remoto de impresión
Rp	De cadena	Especifica el nombre de una impresora remota
Sh	Booleano	Especifica esto para suprimir encabezados (páginas bandera)
Sf	Booleano	Especifica esto para suprimir alimentación de formas al final del trabajo
Mx	Numérico	Especifica el tamaño máximo permitido para los trabajos de impresión (en bloques)

Tabla # 5: Campos en */etc/printcap*

Campo lp : Si especifica /dev/null como dispositivo de impresión, el procesamiento general se ejecuta en forma correcta, pero los datos finales van al deposito de vaciado; esto es, a ningún sitio.

Campo lf: Cualquier archivo que especifique, es necesario que ya exista, pues

de lo contrario no ocurre el registro de sesión.

Campo if: Los filtros de entrada son programas que toman datos de entrada en su entrada estándar y generan una salida en su salida estándar. El uso característico de un filtro de entrada es para detectar texto simple y convertirlo en PostScript, esto es, su entrada es texto burdo y su salida es PostScript.

Campos *rp y rm:* Enviar sus datos de impresión a una impresora conectada a su máquina es tan simple como especificar la rm de máquina remota y el rp de impresora remota y asegurarse de que el campo lp de dispositivo de impresión este vacío. Advierta que los datos aun se imprimen indirectamente de forma local antes de ser transferidos a la máquina remota y que también se ejecuta cualquier filtro de entrada que haya especificado.

Campos *sh* y *sf:* Si por lo general emplea su impresora para salida desde paquetes de procesamiento de palabras, es mas útil suprimir la alimentación de formas. La mayoría de los paquetes de procesamiento de palabras crea páginas completas de información, por lo que si el demonio de impresora añade una alimentación de forma al final de cada trabajo, obtendrá una página en blanco después de cada trabajo.

Campo *mx:* Este campo le permite limitar el tamaño de los datos de impresión que habrá de imprimir de manera indirecta. Advierta que el limite depende del tamaño de los datos impresos en forma indirecta y no de la cantidad de datos enviados a la impresora física.

2.1.5 ¿Porqué LINUX?

En sentido estricto, porque LINUX es el núcleo de un sistema operativo de tiempo compartido: un programa que controla los recursos de una computadora y los asigna entre los usuarios. Permite a los usuarios ejecutar sus programas; controla los dispositivos periféricos (discos, terminales, impresoras y otros) conectados a la máquina; y proporciona un sistema de archivos que administra el almacenamiento a largo plazo de información tal como programas, datos y documentos.

En un sentido más amplio, “UNIX” y por consiguiente “LINUX”, abarca no sólo el núcleo, sino que incluye también programas esenciales, entre ellos: compiladores, editores, programas para copiado e impresión de archivos, etc. Ampliando aún mas, “LINUX” puede incluir programas desarrollados por usuarios para ser ejecutados en el sistema; por ejemplo, herramientas para preparar documentos, rutinas para análisis estadísticos y paquetes gráficos. Cual de estos usos del nombre “UNIX” sea el correcto depende del nivel del sistema que se esté considerando.

Estas son algunas de las razones por las que el sistema operativo LINUX ha conseguido tanto éxito y popularidad:

- El sistema está escrito en un lenguaje de alto nivel, haciéndolo fácil de leer, comprender, cambiar y mover a otras máquinas.
- Tiene una interfaz con el usuario simple que tiene el poder de suministrar los servicios que quiere el usuario.
- Provee primitivas que permite la realización de programas complejos a partir de programas más simples.

- Usa un sistema de archivos jerárquico que permite un fácil mantenimiento y una implementación eficiente.
- Usa un formato para los archivos consistente, el flujo de bytes, haciendo que los programas de aplicación sean fáciles de escribir.
- Provee una simple y consistente interfaz con los dispositivos periféricos.
- Es un sistema multiusuario y multitarea, cada usuario puede ejecutar varios procesos simultáneamente.
- Oculta la arquitectura de la máquina del usuario, facilitando la tarea de escribir programas que corran en diferentes implementaciones de hardware.

Además de que el sistema operativo y muchos de los programas y comandos están escritos en lenguaje C, LINUX soporta otros lenguajes, incluyendo Fortran, Basic, Pascal, Ada, Cobol, Lisp y Prolog. LINUX soporta cualquier lenguaje de programación que tenga un compilador o intérprete y una interfaz con el sistema que permita transformar las peticiones del usuario de servicios del sistema al conjunto estándar de peticiones usadas en el sistema UNIX.

Hoy día, LINUX está siendo acogido por miles de usuarios en el mundo, se puede decir que está superando el auge y el poderío que representa Windows NT. También podemos decir que LINUX es un sistema muy diferente de lo que fue a principios de los años setenta; en aquella época, el sistema representativo era un solo procesador que servía a un conjunto de terminales de teletipo conectadas al procesador a través de líneas telefónicas directas o conmutadas.

El sistema representativo actual es una estación de trabajo con una pantalla de alta definición de mapa de bits que opera con un sistema de ventanas y participa activamente en una extensa red de computadoras.

En aquella época, UNIX era pequeño, sencillo y no comercial, destinado a un público reducido y selecto. Ahora, UNIX y en consecuencia LINUX, es un producto comercial importante, grande, complicado, que se utiliza en una amplia gama de aplicaciones, muchas veces por personas que no tienen experiencia de programación.

Características del Sistema operativo LINUX

Los siguientes conceptos son comunes para todos los sistemas UNIX y por consiguiente para LINUX, por lo cual se puede afirmar que éstos componen las características principales de LINUX.

- **Kernel:** Este es el componente principal del sistema operativo. Se encarga de asignar tareas y manejar el almacenamiento de datos. El usuario rara vez opera directamente con el kernel, que es la parte residente en memoria del sistema operativo.
- **Shell:** Esta es la utilidad que procesa las peticiones de los usuarios. Cuando alguien teclea un comando en la terminal, el shell interpreta el comando y llama el programa deseado. El Shell puede soportar múltiples usuarios, múltiples tareas, y múltiples interfaces para sí mismo. Los dos shells más populares son el **BourneShell** y el **Cshell**, debido a que usuarios diferentes pueden usar diferentes shells al mismo tiempo, entonces el sistema puede aparecer diferente para usuarios diferentes.

Existe otro Shell conocido como KornShell (así llamado en honor de su diseñador), que es muy popular entre los programadores.

- **Programas De Utilidad (Utilerías):** El Sistema Operativo LINUX incluye una gran variedad de programas de utilidad que pueden ser fácilmente adaptadas para realizar tareas específicas. Estas utilerías son flexibles, adaptables, portables y modulares, y pueden ser usadas junto con filtros y redireccionamientos para hacerlos más poderosos.
- **Sistema Multiusuarios:** Dependiendo del equipo disponible, un LINUX puede soportar desde uno hasta más de 100 usuarios, ejecutando cada uno de ellos un conjunto diferente de programas.
- **Sistema Multitareas:** UNIX permite la realización de más de una tarea a la vez. Pueden ejecutarse varias tareas en su interior, mientras se presta toda la atención al programa desplegado en la terminal.
- **Estructura De Archivos:** La estructura de archivos del UNIX está pensada para facilitar el registro de una gran cantidad de archivos. Utiliza una estructura jerárquica o de árbol que permite a cada usuario poseer un directorio principal con tantos subdirectorios como desee; UNIX también permite a los usuarios compartir archivos por medio de enlaces (*links*), que hacen aparecer los archivos en más de un directorio de usuario.

  Además, UNIX permite proteger los archivos del usuario contra el acceso por parte de otros usuarios.
- **Entrada Y Salida Independiente Del Dispositivo:** Los dispositivos (como una impresora o una terminal) y los archivos en disco son considerados como archivos por UNIX. Cuando se da una instrucción al UNIX puede

indicársele que envíe el resultado a cualquiera de los diversos dispositivos o archivos.

- **Comunicación Entre Procesos:** UNIX permite el uso de conductos y filtros en la línea de comandos. Un **conducto** (*pipe*) redirige la salida de un programa para que se convierta en entrada de otro. Un **filtro** es un programa elaborado para procesar un flujo de datos de entrada y producir otro de datos de salida. Los conductos y filtros suelen usarse para unir utilerías y realizar alguna tarea específica.

2.1.6 Java

Originariamente JAVA nació con la idea de crear un software que pudiese utilizarse en electrodomésticos con componentes electrónicos diversos, tales como equipos de vídeo o de audio. Este hecho inspiró el proyecto de desarrollo de un lenguaje independiente de la que en primera instancia recibió el ecológico nombre de *Oak* (Roble), simplemente porque desde el despacho del equipo de creación del lenguaje podía verse uno de estos árboles. El objetivo del equipo era conseguir evitar tener que desarrollar en C o C++ por la sencilla razón de que estos lenguajes obligan a recompilar el código para cada nueva plataforma en la que quiera implantarse.

Todos aquellos que hayan desarrollado o reflexionado en torno a la problemática de crear contenidos en *WWW* habrán pensado alguna vez en la naturaleza heterogénea de la red y los problemas que ello provoca a la hora de desarrollar programas. Si *Java* había nacido como un lenguaje que podía ejecutarse en cualquier máquina, qué podía ser más apropiado para un

entorno, el *web,* en el que jamás puedes saber qué equipo, ni tan siquiera qué sistema operativo estará utilizando el usuario de tu futuro módulo de *software.* Esta universalidad, que conlleva obvias repercusiones económicas en el ahorro de las empresas de desarrollo, se expresa en el adagio de *Sun "Write Once, run anywhere",* algo así como "escribe tu programa una vez, y ejecútalo en cualquier parte".

Algunas Ventajas de JAVA

- La principal ventaja de este lenguaje de programación es su capacidad de desarrollo y ejecución multiplataforma, la programación en Internet y World Wide Web.
- No se necesita volver a escribir el código si se quiere ejecutar el programa en otra máquina. Un solo código funciona para todos los browsers[15] compatibles con Java o donde se tenga una Máquina Virtual de Java (Mac's, Sun's, etc).
- Java es un lenguaje de programación orientado a objetos, y tiene todos los beneficios que ofrece esta metodología de programación.
- Java es un lenguaje y por lo tanto puede hacer todas las cosas que puede hacer un lenguaje de programación: Cálculos matemáticos, procesadores de palabras, bases de datos, aplicaciones gráficas, animaciones, sonido, hojas de cálculo, etc.
- Las páginas de Web, ya no tienen que ser estáticas, se le pueden poner toda clase de elementos multimedia y permiten un alto nivel de interactividad, sin tener que gastar en paquetes carísimos de multimedia.

[15] Buscador

Modelo de Comunicaciones con Java

En Java, crear una conexión socket TCP/IP se realiza directamente con el paquete java.net.

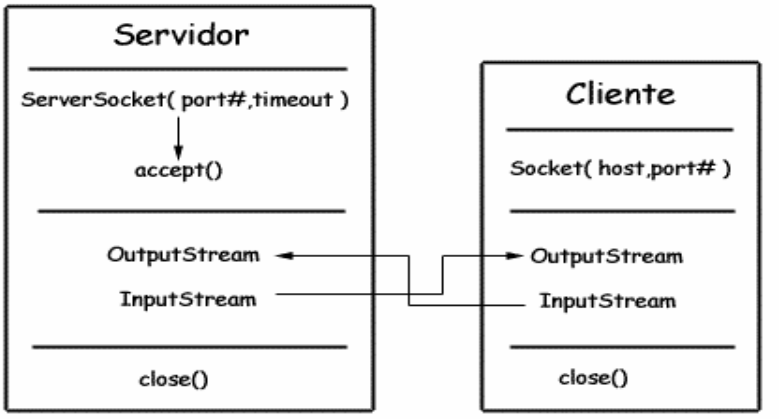


Figura 5: Comunicaciones en Java

El modelo de sockets más simple es:

El servidor establece un puerto y espera durante un cierto tiempo (timeout segundos), a que el cliente establezca la conexión. Cuando el cliente solicite una conexión, el servidor abrirá la conexión socket con el método accept().

El cliente establece una conexión con la máquina host a través del puerto que se designe en #puerto.

El cliente y el servidor se comunican con manejadores InputStream y OutputStream.

Clases Útiles en Comunicaciones:

Socket: Es el objeto básico en toda comunicación a través de Internet, bajo el protocolo TCP. Esta clase proporciona métodos para la entrada/salida a través de streams que hacen la lectura y escritura a través de sockets muy sencilla.

ServerSocket: Es un objeto utilizado en las aplicaciones servidor para escuchar las peticiones que realicen los clientes conectados a ese servidor.

Este objeto no realiza el servicio, sino que crea un objeto Socket en función del cliente para realizar toda la comunicación a través de él.

Otras clases utilizadas en Java para el manejo de Socket son:

- DatagramSocket
- DatagramPacket
- NetworkServer
- NetworkClient
- SocketImpl
- MulticastSocket

2.2 RECOLECCIÓN Y TÉCNICAS DE INFORMACIÓN

Las técnicas utilizadas para la recopilación y organización de información del presente proyecto están basadas en dos tipos de fuentes de información: Fuentes primarias y Fuentes secundarias.

2.2.1 Fuentes Primarias

Las fuentes primarias del presente trabajo se obtuvieron a través de libros, revistas, y trabajos de investigación con temas similares al que contiene el presente proyecto. Las consultas en Internet fueron de gran ayuda e importancia para la realización y desarrollo del presente trabajo.

2.2.2 Fuentes Secundarias

Las fuentes secundarias utilizadas en el presente trabajo están basadas esencialmente en las charlas y/o entrevistas realizadas a personas con conocimientos en el tema, así como la asesoría y sugerencias de profesores, y nuestro director de proyecto.

3. DISEÑO DE LA APLICACIÓN

3.1 DISEÑO FUNCIONAL DE LA APLICACIÓN

3.1.1 Diseño General de la Aplicación

Figura 6: Descripción General de la aplicación

3.1. 2 Definición de Módulos Principales

Cada uno de los módulos principales de PrintSerX corresponde a un proceso que ejecuta tareas específicas y que en conjunto forman lo que llamamos Aplicación de Impresión Remota para usuario final a través de Internet, basada en el Modelo Cliente/Servidor.

Los módulos principales del Cliente de PrintSerX son:

- ***Módulo de Usuario final:*** Contiene todas las herramientas y funciones de interacción con el usuario final. Maneja las funciones de la barra de herramientas, el menú principal de la aplicación y todo lo referente a la interfaz gráfica y al manejo de ventanas.
- ***Módulo de Comunicaciones:*** Es el encargado de establecer la comunicación con el servidor a través del socket, de transmitir y recibir

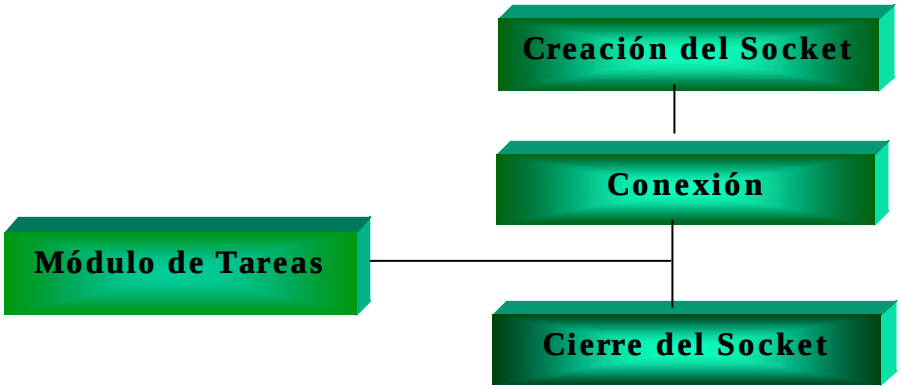
toda la información necesaria hacia y desde el cliente PrintSerX respectivamente.

- ***Módulo de Tareas:*** Realiza la captura de la información proporcionada por el usuario, envía dicha información al servidor por medio del módulo de comunicaciones, procesa y formatea la respuesta del Servidor de manera que esta sea entendible al usuario.

Los módulos principales del Servidor de PrintSerX son:

- ***Módulo de Comunicaciones:*** Es el encargado de establecer la comunicación con el cliente a través del socket, de transmitir y recibir toda la información necesaria hacia y desde el servidor PrintSerX respectivamente.
- ***Módulo de Resolución de Tareas:*** Ejecuta el comando enviado por el cliente, obtiene la respuesta del mismo y la envía al cliente a través del módulo de comunicaciones.

Módulo de Comunicaciones – Cliente:



Módulo de Usuario Final:

Figura 7: Módulo de Comunicaciones - Cliente

Figura 8: Módulo de Usuario Final

Módulo de Tareas:

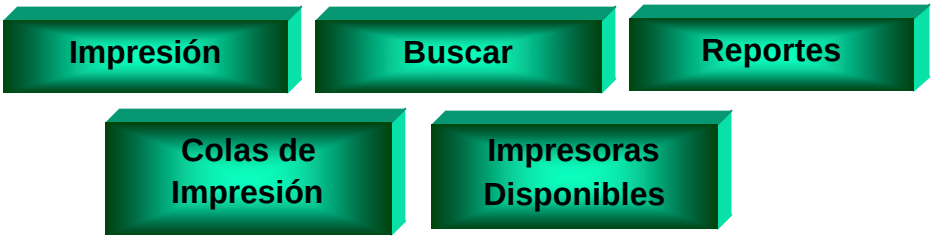
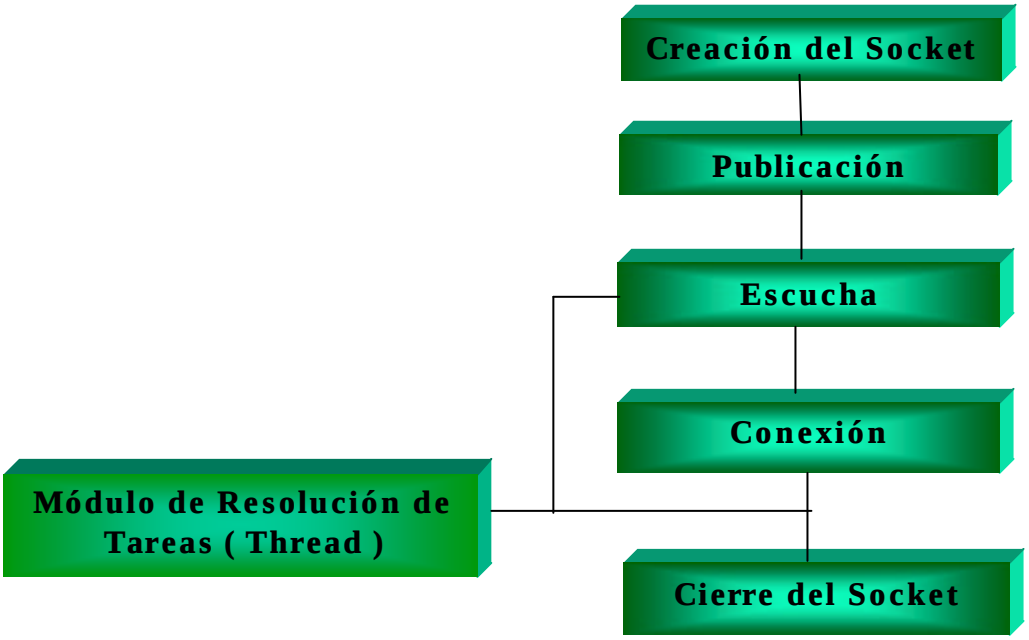


Figura 9: Módulo de Tareas



Módulo de Comunicaciones – Servidor:

Figura 10: Módulo de Comunicaciones - Servidor

Módulo de Resolución de Tareas:

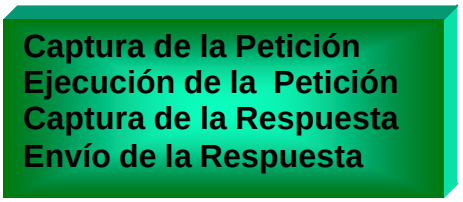


Figura 11: Módulo de Resolución de tareas

3.2 CARACTERÍSTICAS DEL PROCESO CLIENTE

3.2.1 Características de Ejecución

1. Los usuarios son independientes entre sí.
2. Al momento de iniciar este proceso, se lee la configuración básica de la aplicación del archivo de configuración PrintSerX.ini.
3. El proceso cliente permite almacenar las direcciones mas frecuentadas por el usuario en el menú de Favoritos, también guarda las direcciones de las dos últimas conexiones establecidas.

3.2.2 Características del Diseño

1. Posee una interfaz gráfica amigable que permite una fácil interacción con el usuario.
2. Posee métodos para el manejo de reportes de servicios.

3. Este proceso brinda la oportunidad de obtener ayuda en línea acerca del manejo del mismo, visitar la página web de la aplicación PrintSerX 1.0 y enviar correo electrónico con sugerencias y/o problemas acerca del software.

3.3 CARACTERÍSTICAS DEL PROCESO SERVIDOR

3.3.1 Características de Ejecución

1. El servidor se ejecuta en segundo plano y no requiere una monitorización constante.
2. Al momento de ejecutar el comando enviado por el cliente, muestra en pantalla el resultado de ejecutar dicho comando.
3. Este proceso permite realizar un control de todas las tareas ejecutadas por cada cliente, debido a que genera un archivo de log especificando las acciones realizadas por cada uno de ellos.

3.3.2 Características del Diseño

1. El servidor presenta facilidad para la conexión de clientes remotos.
2. Este proceso es de tipo concurrente, es decir, a medida que se conecta un proceso cliente, ejecuta un proceso hijo el cual se encargara de ejecutar las tareas de dicho cliente.
3. Está diseñado para ejecutarse en plataforma LINUX.

4. DISEÑO COMPUTACIONAL

4.1 ESTRUCTURAS LÓGICAS

La aplicación desarrollada cuenta con cinco (5) estructuras de datos dinámicas, de las cuales cuatro (4) son de tipo Vector y una de tipo Archivo. Las primeras pueden redimensionarse automáticamente a sí mismas cada vez que necesiten crecer para contener más elementos.

Las estructuras dinámicas son las siguientes:

Nombre	Descripción
Impr	Almacena las impresoras configuradas en el Servidor
Disp	Almacena los dispositivos físicos de las impresoras
Spol	Almacena los directorios de spool de las impresoras configuradas
Desc	Almacena una descripción de las impresoras.

Tabla # 6: Estructuras de Datos Dinámicas Tipo Vector

La estructura de tipo archivo contiene la información referente a las conexiones realizadas y los archivos enviados a través de la aplicación. Cuenta con siete (7) campos descritos como sigue:

Nombre	Descripción
Fecha	Contiene la fecha en la que realizó la acción especificada.
Hora	Contiene la hora en la que realizó la acción especificada.
Dirser	Contiene la dirección IP del servidor remoto.
Nomser	Contiene el nombre de dominio del servidor remoto.
Puerto	Contiene el puerto a través del cual se estableció la conexión.
Acción	Especifica si se realizó una conexión o se envío un archivo.
Archivo	Contiene el archivo enviado a imprimir al servidor remoto.

Tabla # 7: Estructuras de Datos Dinámicas Tipo Archivo

4.2 DICCIONARIO DE DATOS

4.2.1 Definición de los Flujos de Datos del Cliente

- **Nombre :** Tipo de servicio.

  Descripción: Especifica el servicio o acción que el usuario desea realizar.

  Proveniente de: Usuario.

  Para los procesos: Establecer conexión.

- **Nombre :** Trabajo impreso.

  Descripción: Trabajo de impresión enviado por el usuario.

  Proveniente de: Ejecutar comando.

  Para la fuente: Administrador de trabajos impresos.

- **Nombre :** Propiedades de las impresoras.

  Descripción: Especifica las características de las impresoras configuradas.

Proveniente de: Printerdb.

Para el proceso: Mostrar propiedades de impresoras.

- **Nombre :** Comando a ejecutar.

  Descripción: Especifica el comando remoto enviado por el cliente para ejecutar en el servidor.

  Proveniente del proceso: Realizar comando a ejecutar.

  Para el proceso: Comunicación cliente.

- **Nombre :** Registro de log.

  Descripción: Contiene un registro con la fecha, hora, nombre y dirección del servidor, etc.; generado cuando se establece una conexión o se envía un archivo.

  Proveniente de: PrintSerX.log

  Para el proceso: Generar reportes.

- **Nombre:** Configuración inicial de PrintSerX

  Descripción: Contiene la configuración por defecto de PrintSerX.

  Proveniente de: PrintSerX.ini

  Para el proceso: Configurar PrintSerX.

- **Nombre:** Configuración establecida por el usuario.

  Descripción: Contiene la configuración fijada por el usuario.

  Proveniente del proceso: Salir

  Para la fuente: PrintSerX.ini

- **Nombre:** Mensaje de inicio.

  Descripción: Mensaje de iniciación de la aplicación.

  Proveniente del proceso: Configurar PrintSerX 1.0

  Para el proceso: Ejecutar servicio.

- **Nombre:** Dirección.

  Descripción: Contiene la dirección remota a la que el usuario se desea conectar.

  Proveniente de la fuente: Usuario

  Para el proceso: Establecer conexión

- **Nombre:** Puerto.

  Descripción: Contiene el número del puerto remoto por el que se establecerá la conexión.

  Proveniente de la fuente: Usuario

  Para el proceso: Establecer conexión

- **Nombre:** Conexión establecida.

  Descripción: Mensaje para informar a los demás procesos que se ha establecido comunicación con un servidor.

  Proveniente del proceso: Establecer conexión.

  Para el proceso: Obtener configuración de impresoras.

- **Nombre:** Petición de conexión.

  Descripción: Mensaje de solicitud de conexión.

  Proveniente del proceso: Establecer conexión.

  Para el proceso: Comunicación cliente.

- **Nombre:** Mensaje de cierre.

  Descripción: Mensaje para informar al servidor que se desea cerrar la comunicación.

  Proveniente del proceso: Ejecutar herramientas de PrintSerX.

  Para el proceso: Desconectar.

- **Nombre:** Petición de ayuda.

Descripción: Mensaje para informarle al sistema que se desea ingresar al módulo de ayuda.

Proveniente del proceso: Ejecutar herramientas de PrintSerX.

Para el proceso: Mostrar ayuda.

- **Nombre:** Spol.

  Descripción: Vector con el directorio de spool de las impresoras configuradas.

  Proveniente del proceso: Obtener configuración de impresoras.

  Para el proceso: Realizar comando a ejecutar (1.3.5.1).

  Realizar comando a ejecutar (1.3.6.1).

  Mostrar propiedades de las impresoras.

- **Nombre:** Impr.

  Descripción: Vector para las impresoras configuradas en el servidor.

  Proveniente del proceso: Obtener configuración de impresoras.

  Para el proceso: Realizar comando a ejecutar (1.3.5.1).

  Realizar comando a ejecutar (1.3.6.1).

  Mostrar propiedades de las impresoras.

  Realizar comando a ejecutar (1.3.9.1).

- **Nombre:** Desc.

  Descripción: Vector con la descripción de las impresoras configuradas.

  Proveniente del proceso: Obtener configuración de impresoras.

  Para el proceso: Realizar comando a ejecutar (1.3.5.1).

  Mostrar propiedades de impresoras.

- **Nombre:** Disp.

  Descripción: Vector con los dispositivos físicos de las impresoras.

Proveniente del proceso: Obtener configuración de impresoras.

Para el proceso: Realizar comando a ejecutar (1.3.5.1).

Mostrar propiedades de impresoras.

- **Nombre:** Opciones de Reportes.

  Descripción: Contiene las opciones especificadas por el usuario para la generación de un determinado reporte.

  Proveniente de la fuente: Usuario.

  Para el proceso: Generar reportes.

- **Nombre:** Petición de reporte.

  Descripción: Mensaje para informarle al sistema que se desea obtener un reporte.

  Proveniente del proceso: Ejecutar herramientas de PrintSerX.

  Para el proceso: Generar reportes.

- **Nombre:** Reporte.

  Descripción: Informe solicitado por el usuario.

  Proveniente del proceso: Generar reportes.

  Para el proceso: Mostrar reportes.

- **Nombre:** Opciones y archivos de impresión.

  Descripción: Contiene todos los parámetros de impresión y el archivo que se desea imprimir.

  Proveniente de la fuente: Usuario.

  Para el proceso: Realizar comando a ejecutar.

- **Nombre:** Nombre del archivo.

  Descripción: Nombre del archivo que se desea buscar en las impresoras.

  Proveniente de la fuente: Usuario.

Para el proceso: Realizar comando a ejecutar (1.3.9.1)

- **Nombre:** Impresora.

  Descripción: Nombre de la impresora con la que se desea trabajar.

  Proveniente de la fuente: Usuario.

  Para el proceso: Mostrar propiedades de las impresoras.

  Realizar comando a ejecutar (1.3.6.1).

- **Nombre:** Trabajo de Impresión.

  Descripción: Trabajo de impresión que se desea colocar al principio o borrar de la cola de impresión.

  Proveniente de la fuente: Usuario.

  Para el proceso: Realizar comando a ejecutar (1.3.6.1).

4.2.2 Definición de Flujos de Datos del Servidor

- **Nombre :** Comando a ejecutar.

  Descripción: Especifica el comando enviado por el cliente para ejecutar en el servidor.

  Proveniente del proceso: Comunicación Servidor.

  Para el proceso: Ejecutar comando.

- **Nombre :** Respuesta del comando ejecutado.

  Descripción: Contiene la respuesta del comando ejecutado.

  Proveniente de: Ejecutar comando.

  Para el proceso: Comunicación servidor.

- **Nombre :** Trabajo impreso.

  Descripción: Contiene el trabajo de impresión enviado por el cliente.

  Proveniente de: Ejecutar comando.

Para la fuente: Administrador de trabajos impresos.

4.2.3 Definición de los Almacenes de Datos del Cliente

- **Nombre:** Printerdb

  Descripción: Contiene las características de las impresoras

  Flujos de Datos Proporcionados: Propiedades de las impresoras.

- **Nombre:** PrintSerX.log

  Descripción: Contiene todos los parámetros de las conexiones realizadas y archivos enviados a un servidor remoto.

  Flujos de Datos Recibidos: Registro de log.

  Flujos de Datos Proporcionados: Registro de log.

- **Nombre:** PrintSerX.ini

  Descripción: Contiene los parámetros de iniciación de la aplicación y guarda la configuración realizada por el usuario.

  Flujos de Datos Recibidos: Configuración establecida por el usuario.

  Flujos de Datos Proporcionados: Configuración inicial de PrintSerX.

4.2.4 Definición de los Almacenes de Datos del Servidor

- **Nombre:** PrintSerX.log

  Descripción: Contiene todos los parámetros de las conexiones realizadas y comandos ejecutados en el servidor.

  Flujos de Datos Recibidos: Registro de log.

  Flujos de Datos Proporcionados: Registro de log.

4.2.5 Definición de los Procesos del Cliente

- **Proceso:** Configurar PrintSerX 1.0

  Descripción: Configura la aplicación con los parámetros básicos para su ejecución y la configuración establecida por el usuario la última vez que ejecutó la aplicación.

  Entrada: Configuración inicial de PrintSerX

  Salida: Mensaje de inicio

- **Proceso:** Comunicación cliente

  Descripción: Es el encargado de la comunicación con el servidor a través del socket.

  Entrada: Comando a ejecutar

  Salida: Respuesta del comando ejecutado

  Archivo printcap

- **Proceso:** Salir.

  Descripción: Guarda la configuración establecida por el usuario y finaliza la aplicación.

  Entrada: Mensaje de cierre.

  Salida: Configuración establecida por el usuario.

- **Proceso:** Establecer conexión.

  Descripción: Establece la conexión con el servidor remoto.

  Entrada: Tipo de servicio

  Dirección y puerto

  Archivo printcap.

  Salida: Petición de conectar.

- **Proceso:** Obtener configuración de impresoras.

Descripción: Obtiene los parámetros de configuración de las impresoras del archivo de printcap enviado por el servidor.

Entrada: Conexión establecida

Salida: Impr, Spol, Disp, Desc.

- **Proceso:** Ejecutar herramientas de PrintSerX.

  Descripción: Ejecuta la herramienta de la aplicación seleccionada por el usuario.

  Entrada: Tipo de servicio

  Salida: Impr, Spol, Disp, Desc.

  Mensaje de desconectar

  Petición de reporte, Petición de ayuda

- **Proceso:** Desconectar.

  Descripción: Finaliza la comunicación a través del socket entre el cliente y el servidor.

  Entrada: Mensaje de desconectar

- **Proceso:** Imprimir trabajos.

  Descripción: Permite enviar a impresión un archivo y establecer los parámetros del mismo.

  Entrada: Impr, Spol, Disp, Desc.

- **Proceso:** Manipular colas de impresión.

  Descripción: Permite visualizar las colas de impresión de las impresoras configuradas, borrar uno o todos los archivos y poner un archivo al principio de la cola.

  Entrada: Impr, Spol, Disp.

- **Proceso:** Manipular reportes.

Descripción: Permite generar reportes de conexiones realizadas y archivos enviados a un servidor en particular ó a todos los servidores con los que se estableció conexión. Además, permite especificar la fecha en la que se realizó la acción.

Entrada: Petición de reporte.

- **Proceso:** Mostrar impresoras disponibles.

  Descripción: Visualiza las características de las impresoras disponibles en el servidor.

  Entrada: Impr, Spol, Disp, Desc.

- **Proceso:** Buscar trabajos de impresión.

  Descripción: Busca en la cola de impresión de la impresora especificada el trabajo de impresión establecido.

  Entrada: Impr.

- **Proceso:** Mostrar ayuda.

  Descripción: Visualiza la ayuda de la aplicación PrintSerX.

  Entrada: Petición de ayuda.

- **Proceso:** Realizar comando a ejecutar.

  Descripción: Genera el comando que se va a ejecutar en el servidor a partir de las opciones establecidas por el usuario.

  Entrada: Impr, Spol, Disp, Desc.

  Salida: Comando a ejecutar

- **Proceso:** Mostar respuesta.

  Descripción: Convierte la respuesta del comando ejecutado en el servidor a un formato más entendible para el usuario.

  Entrada: Impr, Spol, Disp, Desc.

Salida: Comando a ejecutar

- **Proceso:** Generar reportes.

  Descripción: Genera un reporte a partir de las opciones especificadas por el usuario.

  Entrada: Impr, Spol, Disp, Desc.

  Salida: Comando a ejecutar

4.2.6 Definición de los Procesos del Servidor

- **Nombre:** Comunicación Servidor.

  Descripción: Permite establecer la comunicación con el proceso Cliente a través de sockets.

  Entrada: Comando a Ejecutar

  Salida: Respuesta del comando ejecutado o Archivo Printcap.

- **Nombre:** Ejecutar Comando.

  Descripción: Ejecuta el comando enviado por el proceso cliente.

  Entradas: Comando a Ejecutar.

  Salidas: Trabajo impreso.

4.3 DIAGRAMA DE FLUJO DE DATOS

4.3.1 Diagrama de Contexto

4.3.2 Diagrama de Flujo de Primer Nivel

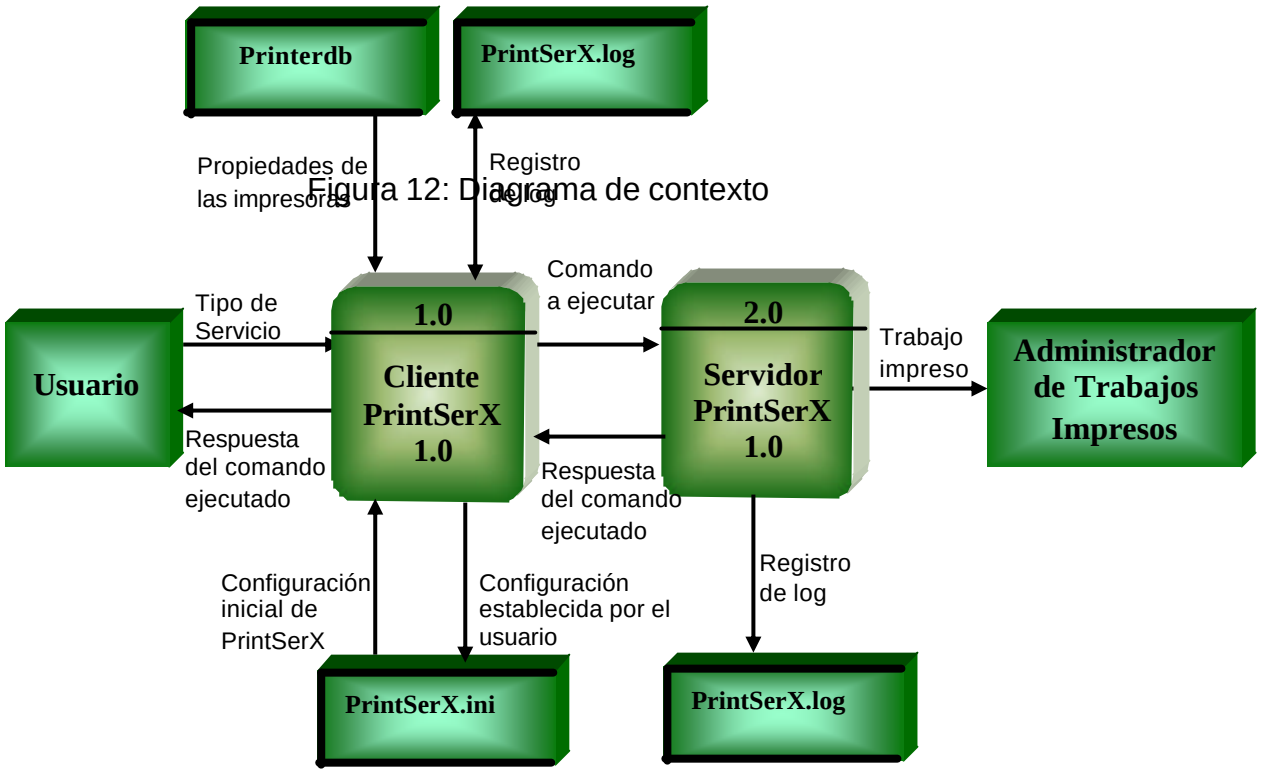


Figura 12: Diagrama de contexto

Figura 13: Diagrama de Flujo de Primer Nivel

4.3.3 Diagrama de Flujo de Segundo Nivel

Cliente PrintSerX

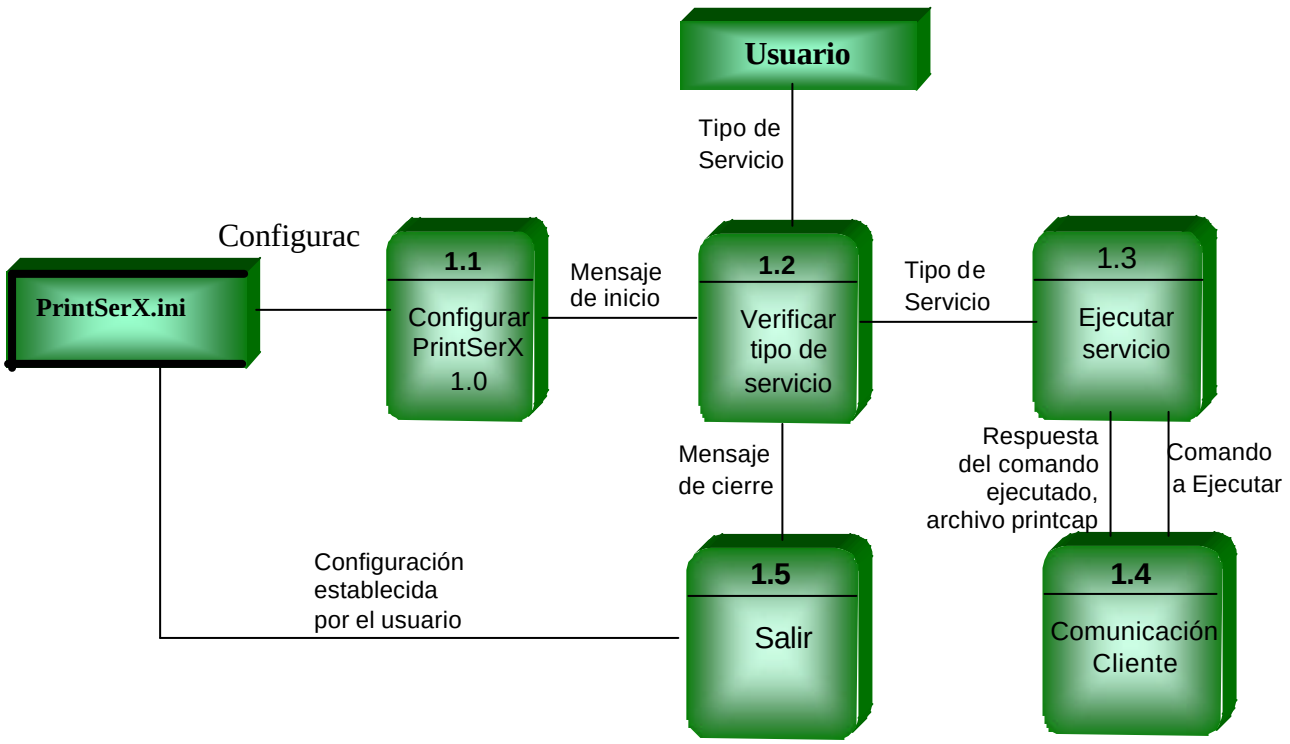


Figura 14: Diagrama de Flujo de Segundo Nivel - Cliente

Servidor PrintSerX

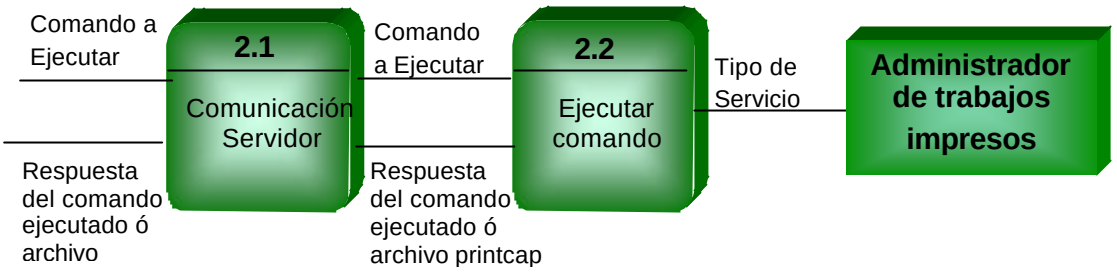


Figura 15: Diagrama de Flujo de Segundo Nivel - Servidor

4.3.4 Diagrama de Flujo de Tercer Nivel

Ejecutar Servicio

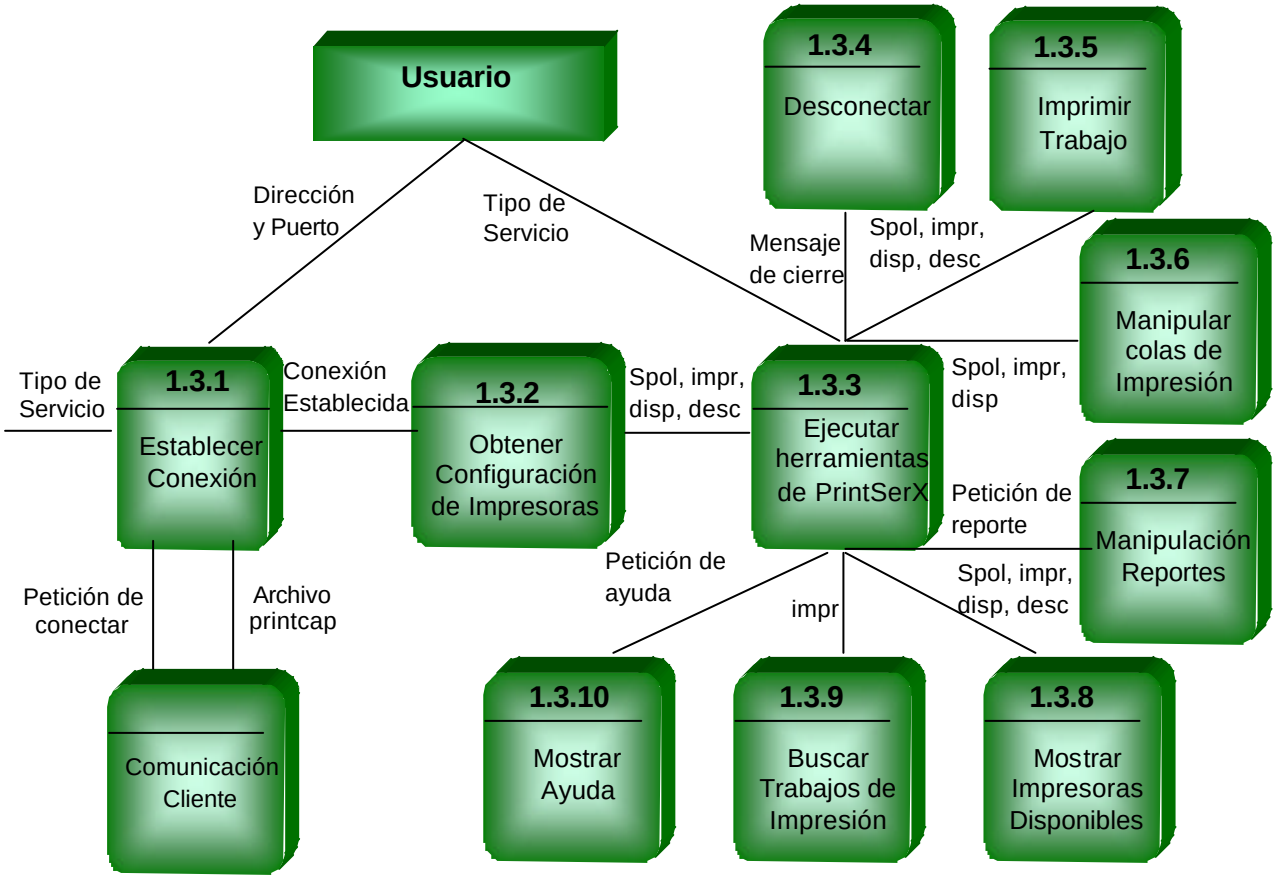


Figura 16: Diagrama de Flujo de Tercer Nivel - Ejecutar Servicio

4.3.5 Diagrama de Flujo de Cuarto Nivel

❖ **Mostrar Reportes**

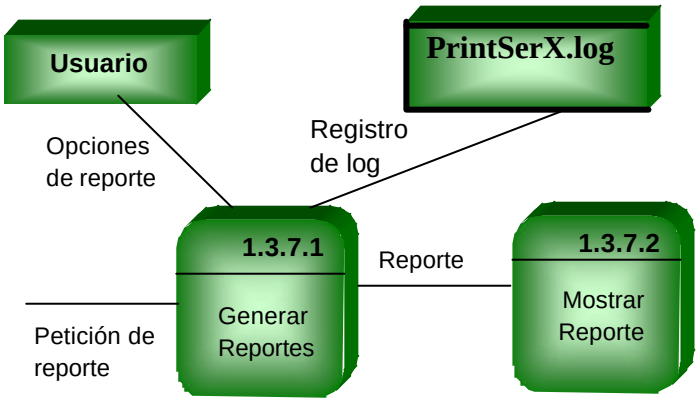


Figura 17: Diagrama de Flujo de Cuarto Nivel – Mostrar Reportes

❖ **Imprimir Trabajo**

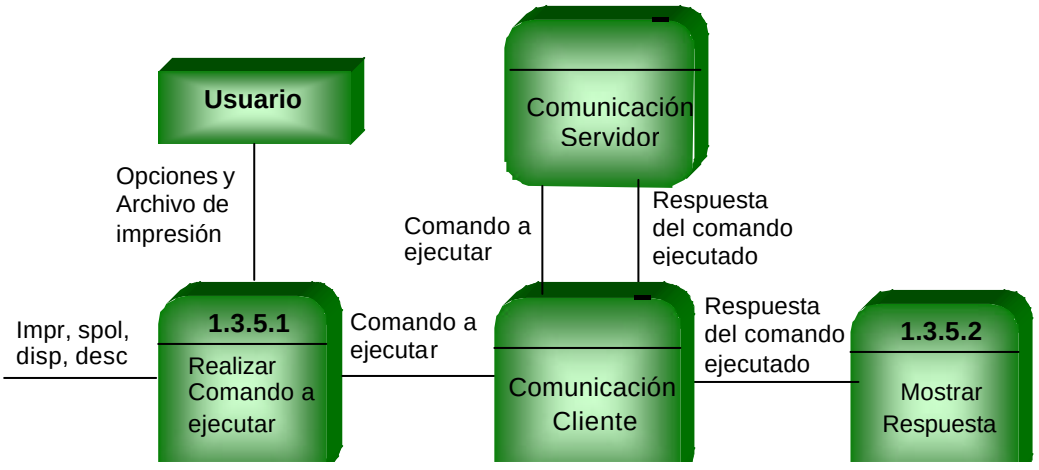


Figura 18: Diagrama de Flujo de Cuarto Nivel – Imprimir Trabajo

❖ **Buscar Trabajo**

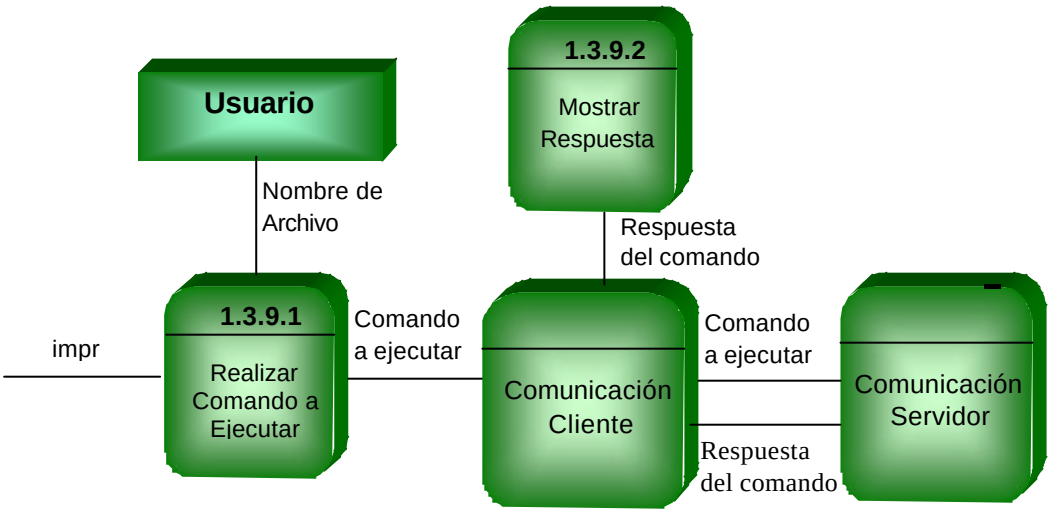


Figura 19: Diagrama de Flujo de Cuarto Nivel – Buscar Trabajo de Impresión

❖ **Manipular Colas de Impresión**

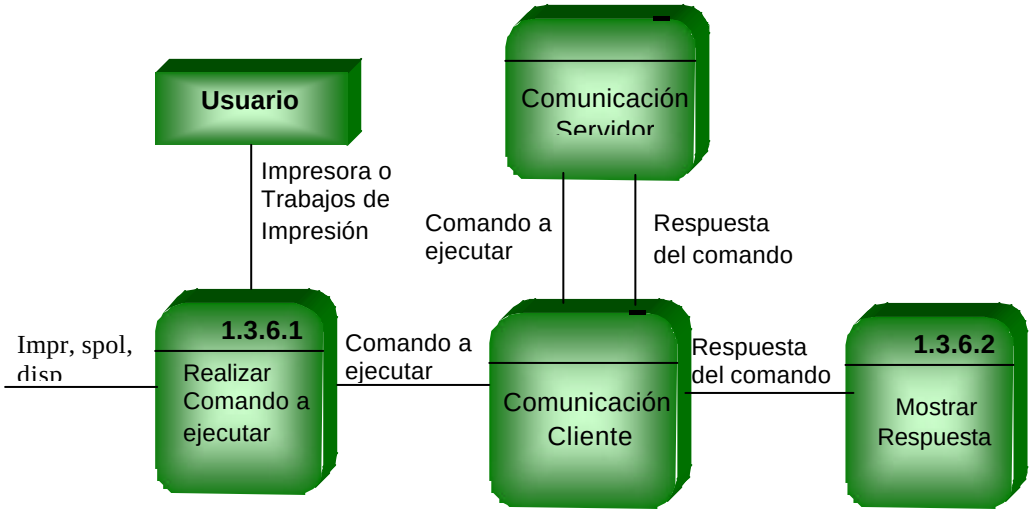


Figura 20: Diagrama de Flujo de Cuarto Nivel – Manipular Colas de Impresión

❖ **Impresoras disponibles**

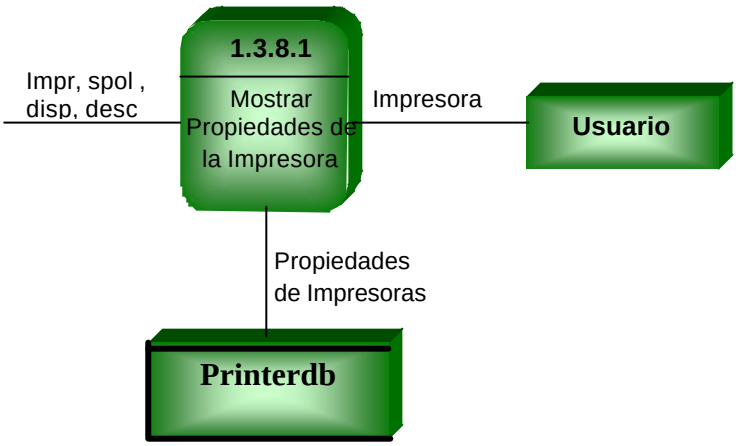


Figura 21: Diagrama de Flujo de Cuarto Nivel – Impresoras disponibles

5. RECOMENDACIONES

Para el enriquecimiento del presente proyecto, los autores brindan las siguientes recomendaciones:

- En la parte de impresión en LINUX, sería de mucha ayuda extender los tipos de archivos que se pueden imprimir, para esto es necesario implementar filtros mágicos capaces de manipular archivos con extensiones Doc, Pdf, Xls, Ppt, entre otros; esto con el fin de enriquecer las capacidades del software.

- La comunicación a través de sockets no solo es posible realizarla por medio de sockets tipo stream, también es posible con sockets datagram, raw y seqpacet, las cuales proporcionan diferentes características útiles para la comunicación entre procesos. Por lo que se recomienda estudiar los mecanismos de comunicación con sockets y/o implementar aplicaciones Cliente/Servidor utilizando los sockets anteriormente mencionados.

- Tomando como base el presente proyecto y teniendo en cuenta que el servicio de impresión no es el único servicio brindado por LINUX, es posible derivar otras aplicaciones multiplataformas basadas en el modelo Cliente/Servidor para manipular otros servicios de dicho sistema operativo; como por ejemplo, un servidor de FAX. Por lo tanto recomendamos a las personas interesadas en el estudio de este tema profundizar un poco en su investigación y tenerlo en cuenta como un futuro proyecto de grado.

- Un proyecto interesante que puede surgir a partir del presente proyecto sería el diseño e implementación de impresión remota para otros sistemas operativos de red, como son UNIX, Solarix, Windows NT, entre otros. Esto brindaría al usuario mayores posibilidades en la impresión de archivos y/o documentos, ya que podría despreocuparse del sistema operativo, debido a que el proyecto abarcaría los más importantes o comúnmente utilizados.

- Otro trabajo de investigación relacionado con el presente proyecto puede ser la investigación y desarrollo de un explorador remoto para LINUX, desde una plataforma Windows.

CONCLUSIONES

Al finalizar el presente proyecto podemos sacar las siguientes conclusiones:

- La aplicación desarrollada permite manejar el servicio de impresión de LINUX remotamente por medio de la plataforma Windows, a través de una interfaz gráfica basada en el manejo de ventanas y menús que permiten una fácil interacción con el usuario.

- PrintSerX 1.0 le brinda al usuario facilidades de conexión con un servidor remoto a través de Internet o por medio de una red LAN. Para esto basta con especificar la dirección y el puerto del servidor al que se desea conectar. Además permite almacenar las direcciones de los servidores mas frecuentados o favoritos con el fin de tener acceso directo a estos. También guarda las direcciones de los dos últimos servidores con los que se estableció conexión.

- El servidor PrintSerX esta diseñado para recibir múltiples peticiones de conexión de los clientes y crear procesos hijos que se encarguen de satisfacer las necesidades de estos. Cada proceso hijo posee las mismas

características y capacidades del servidor, lo cual amplia los beneficios de los clientes y mejora el desempeño del servidor debido a que este está descargado de la tarea de atender al cliente. Por tal motivo, el cliente obtendrá la información de una manera rápida y eficaz.

- La aplicación de impresión remota PrintSerX 1.0 implementa mecanismos de comunicación entre procesos de distintas plataformas, como LINUX y Windows, lo cual posibilita el aprovechamiento de los servicios de impresión de LINUX a través de la plataforma Windows. Esto genera beneficios al cliente ya que permite explotar características no disponibles en su sistema operativo.

- El presente proyecto esta basado en la tecnología Cliente / Servidor, la cual se considera un paradigma de comunicación de fundamental importancia en la época actual debido a que soporta múltiples ambientes, múltiples plataformas, múltiples manejadores de bases de datos; permitiendo la adquisición de hardware y software sin pensar en su compatibilidad. Otro concepto primordial utilizado en el desarrollo del presente trabajo de investigación son los sockets, los cuales son utilizados para la comunicación entre procesos. Es decir, dos procesos en distintos computadores establecen comunicación mediante sockets.

  Es de fundamental importancia manejar estos conceptos, ya que actualmente son los que tienen la mayor utilización en temas relacionados con redes y telecomunicaciones. Manejando estos conceptos se puede

saber como funcionan los computadores cuando están en red y como sacarles el mayor provecho.

GLOSARIO

- ***Address:*** Dirección. Este término se refiere a la dirección IP, o a una dirección de correo electrónico.
- ***ASCII:*** *American Standard Code for Information Interchange*. Es el código estándar del conjunto de caracteres que cualquier computador puede entender; es usado para representar las letras latinas, mayúsculas, minúsculas, números, puntuación, etc.
- ***Arpanet***: *Advanced Research Projects Agency Network*. Red militar norteamericana a través de líneas telefónicas, de la que posteriormente derivó Internet.
- ***Browser (Buscador):*** También llamado navegador. Programa que se utiliza para ver páginas Web. Netscape Navigator y Microsoft Explorer, son dos de los mas utilizados en la actualidad.
- ***Cliente***: programa que se usa para obtener y contactar datos de un programa servidor localizado en otro computador, a menudo a gran distancia. Cada programa cliente está diseñado para trabajar con uno o mas tipos de servidores específicos, y cada cliente requiere un tipo especial de servidor.

- ***Conexión***: Circuito virtual de transporte que se establece entre dos programas de aplicación con fines comunicativos.
- ***Contraseña***: *Password*. Palabra o cadena de caracteres, normalmente secreta, que se utiliza como herramienta de seguridad para identificar usuarios de una aplicación, archivo, o red. Puede tener la forma de una palabra o frase de carácter alfanumérico, y se usa para prevenir accesos no autorizados a información confidencial.
- ***Datagramas***: Paquetes de datos de información.
- ***Dirección IP***: Todos los computadores conectados a Internet tienen una dirección numérica e irrepetible llamada dirección IP (p.ej. 132.248.54.10), la cual sirve para identificar con quien o con que se va a conectar. En razón de que resulta mas sencillo recordar una cadena de palabras que una de números, se creó el DNS (Sistema de Nombres de Dominio) que contiene la equivalencia entre las dos series.
- ***Domain (Dominio)***: Sistema de denominación de Hosts en Internet. Los dominios van separados por un punto y jerárquicamente están organizados de derecha a izquierda. ej: arrakis.es . La parte de la derecha es la mas general.
- ***FTP*** : *File Transfer Protocol*. Protocolo para la transferencia de archivos entre computadores. Las páginas FTP se utilizan principalmente para que los usuarios puedan 'bajar' programas o información desde los sitios de Internet hasta sus equipos. FTP es un modo especial de entrar en otro servidor de web en Internet, para enviar o transferir archivos.

- ***Grupo de discusión*** : Sitio web (o parte de un sitio) que permite discusiones interactivas. Los usuarios envían sus preguntas y respuestas mediante formularios.
- ***Host*** : Literalmente anfitrión. En Internet, se llama así a un computador conectado a la red, que tiene su propio número IP y nombre de dominio, y que sirve información a través de WWW.
- ***HTML*** : Es el lenguaje que se utiliza para diseñar las páginas de la web. Hay diferentes versiones de *HTML* (2.0, 3.0, 3.1, 3.2, 4.0, etc.), cada vez más complejas, que van incorporando gradualmente a las nuevas versiones de los navegadores.
- ***HTTP:*** Acrónimo de HyperText Transport Protocol (Protocolo de Transporte de Hipertexto). Protocolo para mover archivos de hipertexto a través de la Internet. Requiere un programa cliente HTTP en un extremo y un programa servidor de HTTP en el otro. HTTP es el protocolo más importante usado en el WWW.
- ***Internet*** : La gran colección de redes interconectadas que usan protocolo TCP/IP y que evolucionó de *Arpanet* a finales de los 60 y principios de los 70. Internet conecta hoy por hoy a 60.000 redes independientes dentro de la red mundial global.
- ***Intranet*** : Se llaman así a las redes tipo Internet pero que son de uso interno, por ejemplo, la red corporativa de una empresa.
- ***IP:*** (ing.: Internet Protocol) Protocolo de Internet. Es un número dividido en cuatro parte separadas por puntos. Cada computador tiene un sólo número, si no lo tiene, no está realmente en Internet. La mayoría de computadores tienen uno o más nombres de dominio que son más fáciles de recordar.

- ***IRC***: *Internet Relay Chat*. Sistema para transmisión de texto multiusuario a través de un servidor IRC. Usado normalmente para conversar 'en línea'.
- ***Java Script***: Lenguaje derivado de Java, pero con instrucciones mucho más simples.
- ***Kernel*** : Este es el componente principal del sistema operativo. Se encarga de asignar tareas y manejar el almacenamiento de datos. El usuario rara vez opera directamente con el kernel, que es la parte residente en memoria del sistema operativo.
- ***LAN:*** Acrónimo de Local Area Network (Red de Área Local). Red de computadores limitada a un área inmediata, que es normalmente el mismo edificio o piso de un edificio.
- ***Link (Enlace)***: Sitio que al ser presionado le permite al usuario moverse dentro de la misma página o ir a una situada en otra parte de Internet. Normalmente es de un color diferente al texto, generalmente azul, y al colocar el cursor sobre él aparece una 'mano'.
- ***Linux*** : Versión Shareware del conocido sistema operativo Unix. Es un sistema multitarea multiusuario de 32 bits para PC.
- ***Marquesina*** : Sector de una página web que muestra un mensaje de texto que se desplaza horizontalmente.
- ***Multitarea***: UNIX permite la realización de más de una tarea a la vez. Pueden ejecutarse varias tareas en su interior, mientras se presta toda la atención al programa desplegado en la terminal.
- ***Multiusuario***: Dependiendo del equipo disponible, un LINUX puede soportar desde uno hasta más de 100 usuarios, ejecutando cada uno de ellos un conjunto diferente de programas.

- ***Network Information Center (NIC)***: Organismo responsable de la administración de las direcciones IP de toda la red.
- ***Nodo*** : Por definición punto donde convergen mas de dos líneas. A veces se refiere a una única máquina en Internet.
- ***OSI:*** Acrónimo de Open System Interconnection (Interconexión de sistemas abiertos). El modelo de referencia OSI proporciona la base para el desarrollo de estándares relativos a las redes. Este modelo enumera siete capas que definen las actividades que deben tener lugar cuando se comunican los dispositivos a través de una red. Estas siete capas (de arriba a abajo) son: aplicación, presentación, sesión, transporte, red, enlace y física
- ***Página man***: Páginas de ayuda de UNIX, o cualquier sistema que se derive de él. A través de estas páginas es posible encontrar ayuda de los comandos manejados en el mencionado sistema operativo.
- ***Página Web*** : Cualquier documento que se puede ver en el Web. Un sitio Web consta de una o varias páginas.
- ***Printcap***: Archivo de configuración de impresoras.
- ***Proceso***: Es una instancia de un programa en ejecución.
- ***Protocolo*** : Conjunto de instrucciones que permite la comunicación entre computadores.
- ***Protocolo de comunicaciones:*** Reglas y procedimientos utilizados en una red para establecer la comunicación entre las máquinas que disponen de acceso a la red.
- ***Proveedor de Servicios de Internet*** : Compañía que proporciona acceso a Internet mediante un servicio de suscripción.

- ***Puerto:*** (ing.: port) Se llama así a un lugar donde la información entra o sale de un computador o ambas cosas. Por ejemplo, el "puerto serie" de un computador es donde se conectaría un módem. En Internet, puerto también se refiere a menudo a un número que es parte del URL, apareciendo tras el signo ":", justo después del nombre de dominio.
- ***Red:*** Grupo de computadores y otros dispositivos periféricos conectados unos a otros para comunicarse y transmitir datos entre ellos.
- ***RFC:*** Acrónimo de Request for Comments (Petición de Comentarios). Resultado y proceso de creación de un estándar en Internet. Los nuevos estándares se proponen y publican en Internet, como RFC. El grupo de trabajo de Ingeniería de Internet (IETF) es un cuerpo de opinión que admite discusión a través de comentarios, en los que se establece un nuevo estándar.
- ***Servidor***: Computador que administra los recursos en una red de computadores.
- ***Sockets:*** Son puntos o mecanismos de comunicación entre procesos que permiten que un proceso hable (emita o reciba información) con otro proceso, incluso estando estos procesos en distintas máquinas.
- ***Spool:*** Directorio de impresión indirecta.
- ***Superusuario:*** Llamase así al usuario con todos los permisos de acceso al sistema Unix. Usuario Supervisor.
- ***TCP/IP:*** Acrónimo de Transmission Control Protocol / Internet Protocol
- (Protocolo de Internet / Protocolo de Control de Transmisión).
- Es el tipo de protocolos que define la Internet. Diseñado originalmente por el sistema operativo UNIX, el software TCP/IP no está disponible para la

mayor parte de los sistemas operativos. Para acceder a Internet, el computador debe tener software TCP/IP.

- ***Thread:*** también llamado contexto de ejecución o proceso ligero, es un flujo secuencial de instrucciones dentro de un proceso. Así, un proceso podrá estar formado por distintos threads que se ejecutan en paralelo. La capacidad de multithreading está cada vez más extendida entre los sistemas operativos.
- ***UDP :*** Acrónimo de User Datagram Protocol (Protocolo de datagrama a nivel de usuario), perteneciente a la familia de protocolos TCP/IP. Este protocolo no es tan fiable como TCP, pues se limita a recoger el mensaje y enviar el paquete por la red. Para garantizar el éxito de la transferencia, UDP hace que la máquina de destino envía un mensaje de vuelta. Si no es así, el mensaje se envía de nuevo. Con este protocolo no se establece una conexión entre las dos máquinas.
- ***UNIX:*** Sistema operativo diseñado para ser usado por mucha gente al mismo tiempo (es multiusuario) y tiene TCP/IP. Es el sistema operativo más común para servidores en Internet.
- ***WAN:*** Acrónimo de Wide Área Network (Red de Área Extendida. Una red que cubre un área más grande un sólo edificio.

BIBLIOGRAFIA

- MARTINEZ, Juan y ARCILA, Jaime. Fundamentos de UNIX.
- COMER, Douglas E. TCP/IP: Principios básicos, protocolos y arquitectura.
- TANENBAUM, Andrew S. Sistemas operativos modernos.
- MARQUEZ, Francisco Manuel. UNIX. Programación Avanzada.
- ABAD, Alfredo y MADRID, Mariano. Redes de área local.
- JENKINS, Neil y SCHATT, Stan. LAN: Redes de área local.
- GIBBS, Mark. Redes para todos.
- RIFFLET, Jean-Morip. Comunicaciones en UNIX
- HAHN, Harley / Stout. INTERNET: Manual de referencia...
- HUIDOBRO, José M. Comunicaciones: interfaces, módem, protocolos.
- TACKETT, Jack; GUNTER, David y BROWN, Lance. LINUX, Edición Especial.
- TANENBAUM, Andrew S. Redes de Computadora. Editorial Prentice Hall.
- BLANCO, Vicente J. LINUX, Instalación, Administración y uso del Sistema.
- MICROSOFT. Visual J++ 6.0. Manual del programador.
- DEITEL, H.M. y DEITEL, P.J. Cómo Programar en JAVA. Primera Edición.
- http://vgg.sci.uma.es/redes/bajos.html
- http://vgg.sci.uma.es/redes/index.html
- http://linux.lared.es/lucas/COMO-INSFLUG/html/c..ion-como-4.htmt
- http://www.astart.com/lprng/lprng-howto-14.html

- http://www.gth.die.upm.es/~juana/faq-5.html
- http://timc.imag.fr/doc/lprng_doc-3.4.5/lprng-howto-4.html
- http://www.ingula.ve/manual/impre.html
- http://www.dat.etsit.upm.es/~lagomez/tutorial.html
- http://www.microplex.com/microplex/support/bulletins/PRT-960607-04.html
- http://xray1.physics.sunysb.edu/doc/printing/printing-man-lpr.1.html
- http://www.ctv.es/USERS/xose/linux/linux_programas.html
- http://www.blackdown.org/java-linux/Mirrors.html
- ftp://ftp.cica.es/pub/java-linux/JDK-1.2/i386/pre-v1/
- ftp://wauug.erols.com/pub/java/JDK-1.0.2/
- http://www.monografias.com/stiles/trabajos/java.html
- http://www.hispan.com/eltaller/CursoVisualJ++i.html
- ftp://a.bitcom.net/pub/unix/
- http://gondor.gdl.iteso.mx/imike/unix/tutunix.html
- http://gondor.gdl.iteso.mx/imike/unix/indunix.html
- ftp://ftp.funet.fi/pub/Linux/java/JDK/
- http://linux.ncc.org.ve/LPD/
- http://www.uchile.d/lmater/Java/Apuntes/
- http://java.sun.com/docs/books/
- http://adex3.flycast.com/server/socket/
- http://sunsite.dcc.uchile.cl/SunSITE/java/docs/
- http://linux.box.sk/lunux/jdk/
- http://www.linux.org/linux/
- http://www.linuxplanet.com/linuxplanet/
- http://developer.java.sun.com/developer/codesamples/
- http://www.gamelan.com
- http://www.javasoft.com
- http://www.deitel.com
- http://www.brenhall.com/deitel