UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

DIVISIÓN DE ESTUDIOS DE POSGRADO

FACULTAD DE INGENIERÍA

"HERRAMIENTA PARA ANÁLISIS DE RENDIMIENTO EN REDES"

TESIS

Como requisito para obtener el grado de

Maestro en Ingeniería

(Eléctrica-Informática)

Presenta

RAYMUNDO SUÁREZ MARTÍNEZ

DIRECTOR DE TESIS

Ing. MARIO RODRÍGUEZ MANZANERA

MÉXICO, D.F. julio 2000

a Dios

a mi madre

a la memoria de mi padre

ÍNDICE

CAPÍTULO V 48

CAPÍTULO VI 65

Capítulo I

I.1 INTRODUCCIÓN

Las redes de comunicaciones son el medio por el que es posible diseminar información entre un conjunto de sitios (nodos), transmitiendo mensajes a través de un conjunto de líneas que los interconectan **[1].**

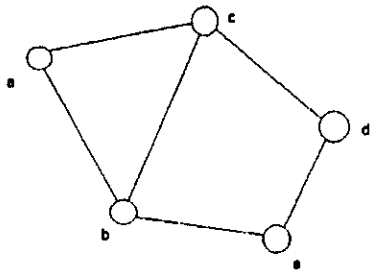


Fig. 1.1 Red de comunicación (nodos: a, b, c, d y e).

En las **redes de computadoras**, éstos sitios se encuentran formados por una o varias máquinas de distintos tipos y capacidad las que son proveedoras de servicios (servidores), o clientes de ellos, o los dos a mismo tiempo **[2].**

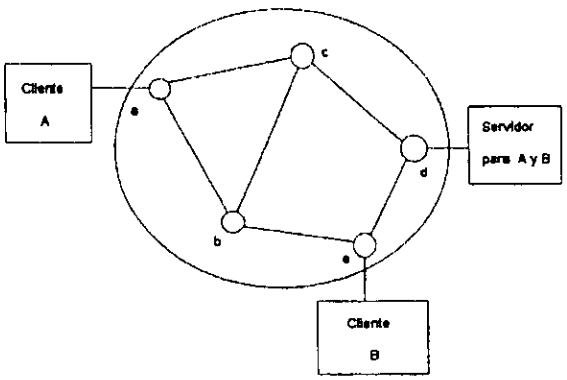


Fig. 1.2 Red de Computadoras (nodos: a, b, c, d y e).

En un principio, las redes se crearon como un intento para compartir el costo de adquisición y mantenimiento tanto del hardware como del software de grandes computadoras (mainframe), las que implantaron el esquema de funcionamiento de *tiempo compartido* **[2]**. Este tipo de funcionamiento permite que múltiples usuarios accedan a una sola computadora a bajo costo y aprovechar su capacidad de procesamiento y almacenamiento de información.

Actualmente se aprovecha la gran oferta y bajo costo de las PC, para integrarlas formando redes que incrementan la productividad de los usuarios, al utilizarse la computadora como herramienta de procesamiento y transferencia de información. Las redes cubren múltiples necesidades de transferencia de información, consulta a BD, transferencia de archivos, acceso a computadoras remotas, correo electrónico; y se les encuentra instaladas en escuelas, bancos, supermercados, oficinas de gobierno y empresas privadas, entre muchos otros lugares.

Es un hecho, que la rápida evolución en el poder de procesamiento de los servidores y las estaciones de trabajo, aunado al creciente número de aplicaciones (multimedia, cliente/servidor, interfaces gráficas y de misión crítica) que trabajan sobre las redes, generan gran demanda de recursos y consumen la capacidad de transmisión de información (ancho de banda). De aquí que la capacidad de las redes sea insuficiente en cortos intervalos de tiempo y necesiten adecuarse continuamente **[3]**.

El diseño y la construcción de redes implica una fuerte demanda de recursos humanos y materiales, y es fuente inspiradora de análisis e investigaciones teóricas para mejorar su desempeño y facilitar su construcción.

I.2 OBJETIVO

El presente trabajo ofrece una herramienta genérica para analizar el rendimiento de redes que utilizan tecnología de conmutación de paquetes. Esta herramienta esta formada por un conjunto de objetos que representan a los diferentes dispositivos, que de manera general, se presentan formando parte de las redes.

Como producto final, se ofrece esta Librería Genérica de Dispositivos, cuyo enfoque esta dirigido a facilitar el análisis del rendimiento de las redes.

I.3 RENDIMIENTO

Como rendimiento se entenderá el análisis de distintos parámetros de las redes **[4],[5]**, como por ejemplo:

- La velocidad de procesamiento de un dispositivo (paquetes/seg.)
- Paquetes transmitidos por unidad de tiempo.
- Paquetes recibidos por unidad de tiempo.
- Retardo en una línea de transmisión específica.
- Retardo en un dispositivo específico.
- Uso promedio de un dispositivo.
- Probabilidad de que un dispositivo este vacío (sin paquetes que procesar).
- Probabilidad de que exista un cierto número de paquetes en un dispositivo.
- El número promedio de paquetes en un dispositivo.
- El tiempo total que pasa un paquete en un dispositivo.
- La longitud promedio de las colas de entrada y salida de un dispositivo (bus).
- La probabilidad de que un paquete aguarde en la cola de un dispositivo, menos de cierto tiempo (t).
- Etc.

I.4 INTEGRACIÓN DE REDES

La integración de redes incluye todos los aspectos relacionados con el análisis, diseño y construcción del sistema **[6]**. Es recomendable realizar también, un análisis *costo-beneficio*, además de la presentación de la solución técnica que cumpla con los requerimientos del proyecto.

Se definen los siguientes temas para la integración de una red:

1) Análisis de los requerimientos
2) Diseño de la Red
3) Instalación
4) Mantenimiento

Éstos puntos incluyen los aspectos más importantes a considerar para el desarrollo de una red, variando el nivel de precisión utilizado en cada una, de acuerdo a las necesidades propias y requerimientos del proyecto.

I.5 DISEÑO

Aquí se establecen los recursos con los cuales satisfacer los requerimientos, incorporando técnicas de análisis que ofrezcan un sistema final completo, añadiendo las actualizaciones tecnológicas necesarias y considerando la creación de un sistema abierto, para cubrir las necesidades futuras de procesamiento y transferencia de información **[6]**.

El nivel de diseño se establece definiendo las abstracciones correspondientes a **la topología y la operación** de la red. La **topología** es la forma que presenta la red como medio de transmisión de información y la constituyen los nodos y las líneas de transmisión (ligas) que unen los nodos. El funcionamiento de la red, integra el comportamiento de todos los elementos que la forman en uno sólo, que define la **operación** completa de la red.

I.6 NIVEL DE ABSTRACCIÓN

Es posible describir y analizar las redes desde diferentes puntos de vista, dependiendo del nivel de abstracción que se requiera o se considere adecuado. Esto puede realizarse tomando en cuenta las condiciones físicas, de operación o lógicas, de la red **[7]**.

El diagrama de la figura 1.3, muestra algunos de los posibles niveles de abstracción que se pueden considerar para el diseño de una red nueva, o para modificar alguna ya existente. Se parte desde el Nivel 0, que representa un diagrama de red, no dirigido, con un servidor y dos clientes, con sus nodos y ligas correspondientes. Hacia arriba del diagrama (Niveles: 1, 2 y 3), aumenta el nivel de abstracción, hasta llegar a un diagrama que representa la interacción de dos clientes con una Base de Datos, sin más detalles. Hacia abajo (Niveles: -1 y –2), el nivel de abstracción disminuye, mostrando cada vez más detalles de la red. Por ejemplo, los puertos, el direccionamiento de las líneas, la potencia de salida, etc.

Cada una de éstas representaciones, define un nivel de abstracción, propio para determinada aplicación u objetivo de diseño **[8]**.

Figura 1.3. Diferentes Niveles de Abstracción
Para el Diseño de la Red

En el presente trabajo, se aplican los **Niveles de Abstracción 0 y 1** para definir la representación más adecuada de los dispositivos en las redes, por que a éstos niveles, se consideran únicamente las características de los dispositivos que influyen directamente en operaciones de creación, enrutamiento y recepción de paquetes de información, que es la parte medular de este estudio, tomando en cuenta características de capacidad, velocidad y rendimiento, de dichos elementos **[7],[9],[10]**.

En general, los elementos a considerar son:

1) Servidores
2) Clientes
3) Dispositivos de interconexión (enrutadores, puentes, multiplexores, etc.)
4) Líneas de transmisión
5) Topologías
6) Tecnologías empleadas
7) Distancias entre nodos

Es necesario que dichos elementos sean descritos con el mayor detalle posible, dado que el nivel de abstracción utilizado los convierte en los actores principales, para la definición de esquemas de transferencia de información en las redes a diseñar y analizar.

El nivel de abstracción definido de esta manera, permite dejar fuera otras características de la red, como los niveles de potencia de los dispositivos, los tipos de conectores, o el lugar de instalación, que no se consideran directamente implicados en los esquemas de transferencia de información, y por lo tanto, se dejan fuera del estudio.

I.7 EL MODELO

En este trabajo se aplica modelos matemáticos para representar a los elementos que forman las redes y los algoritmos que definen el comportamiento de cada uno de ellos, de tal manera que es posible generar diseños rápidamente, analizarlos y modificarlos **[11]**.

Una vez definido el nivel de abstracción al que se trabajará, a continuación se describe el modelo para representar el sistema de red en estudio. El modelo incluye:

1) Una Base de Datos, para almacenar las características de los elementos que forman las redes (marca, capacidad, velocidad, etc.).
2) Una tabla que describe la topología de la red (mapa de nodos y ligas).
3) Una tabla que describe los medios de transmisión utilizados (fibra óptica, par trenzado, cable coaxial, etc.)

Se definen éstos, para describir la red. Las relaciones son las siguientes:

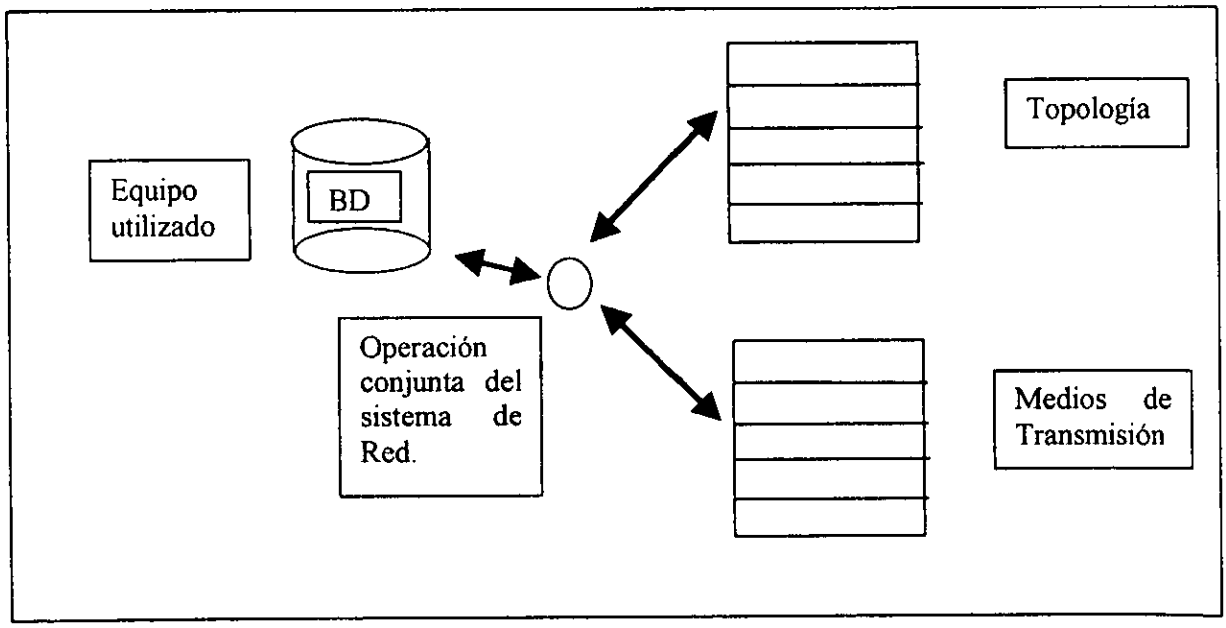


Fig. 1.4 Diagrama de relaciones operativas del proyecto.

I.8 MOTIVACIÓN

En sistemas de red reales, la instalación implica un cierto período de tiempo, que puede variar desde unas semanas a un par de meses, dependiendo del lugar de instalación y del personal que lo realice. Dicho período de tiempo, se relaciona directamente con los costos de ejecución del proyecto, siendo gravemente afectado si se realizan adecuaciones posteriores o si se presentan errores en la instalación. Por tal motivo, son cada vez más utilizadas herramientas virtuales que permitan predecir el comportamiento del sistema bajo esquemas específicos de funcionamiento, que generen ahorros en cuanto a definir el mejor esquema de trabajo de una red, desde su mismo diseño **[12]**.

Las herramientas de ayuda en el diseño de redes han sido utilizadas desde la década de los 60's **[12]** y se han sofisticado hasta alcanzar altos niveles de confianza en la detección de problemas, la predicción de comportamiento y el análisis de modificaciones a las redes. Pero aún no se ha llegado a un planteamiento o esquema de diseño final, dada la rápida evolución tecnológica en esta área y la explosiva proliferación de redes de todos tipos y tamaños. Por lo tanto, esta es un área en la que existe un gran interés y donde los desarrollos implican una gran inversión de recursos y donde principalmente se busca la certeza de los resultados.

El uso de estas herramientas de simulación, se torna sustantivo debido a la importancia de la información que se distribuye a través de las redes y en las que los cambios y adaptaciones son constantes y sus resultados vitales.

La simulación de Eventos Discretos se utiliza desde hace tres décadas con esta finalidad, por lo que es necesario mostrar algunos de los motivos de su aplicación y sus ventajas para sistemas de este tipo.

Capitulo II

II.1 SIMULACIÓN DE EVENTOS DISCRETOS

La simulación de eventos discretos se lleva a cabo para diseñar sistemas nuevos y evaluar el rendimiento de los existentes. Se aplica en sistemas de manufactura, evaluación de armamento, sistemas de transporte, evaluación de hardware y software para sistemas de cómputo, entre otros.

Dado el gran número de redes existentes y la complejidad que presentan, cada vez es más frecuente utilizar simuladores para el análisis de redes de comunicaciones **[12]**. Pero difícilmente se encuentra una herramienta de simulación que contemple de manera precisa el fenómeno de la transferencia de información por las redes, por la propia naturaleza estocástica del funcionamiento de este tipo de sistemas. Más aún, cuando se hace referencia a una red de varios miles o millones de dólares, resulta poco razonable actuar solamente con dicha información para realizar su análisis independientemente de su finalidad ya sea para efectuar cambios, mejoras o ampliaciones.

Continuamente surgen nuevas tecnologías. Nuevas aplicaciones en comunicaciones y redes son exploradas. La pregunta constante que se presenta en las distintas empresas es: ¿qué combinación de tecnologías y aplicaciones son las ideales, para que funcione adecuadamente la red?. En la medida en que las necesidades crecen, la inversión crece y el costo de operación se incrementa. Entre mayor sea la dependencia de la empresa en la red para soportar sus diferentes operaciones de negocios, las fallas causadas por un deficiente desempeño y disponibilidad de la red puede tener serias consecuencias.

Cada vez con mayor frecuencia los diseñadores de redes buscan herramientas de soporte para ayudarlos a decidir cuál dará el mejor desempeño. Ellos utilizan este tipo de herramientas para crear sus diferentes casos de diseño y mostrarlos a sus clientes.

El objetivo es predecir el comportamiento, a través de la simulación mediante un modelo aproximado del sistema **[12]**. Un modelo puede utilizarse para crear diferentes alternativas de diseño o para aplicar diferentes políticas de operación. El analista puede explorar el comportamiento de un sistema propuesto sin construirlo, dado que es poco práctico, caro o imposible, construir diferentes alternativas de un sistema, elegir la mejor y desechar las otras. Además, las modificaciones a un sistema real pueden ser evaluadas a través de la simulación, sin interrumpir o poner en riesgo su funcionamiento.

Lo anterior, estimula el desarrollo de paquetes para simulación, diseñados específicamente para sistemas de comunicaciones, los cuales reducen de manera considerable los costos y tiempos de diseño y análisis, aumentando la confiabilidad en el funcionamiento del diseño o en las adecuaciones que necesiten las redes.

El presente capítulo, analiza los diferentes tipos de simuladores disponibles en el mercado para diseñar redes y algunas características deseables en este tipo de sistemas. Se hace énfasis en las facilidades que ofrecen para construir modelos de simulación y las diferencias entre ellos.

Es necesario indicar, que a pesar de las facilidades que ofrecen los sistemas de simulación, para su aplicación se requiere tomar en cuenta todos los detalles que existen para realizar las simulaciones, como por ejemplo, verificar la correcta formulación del problema, la adecuada recolección de datos, la validez del modelo, el diseño y análisis de las corridas; y por otra parte el estar dispuesto a invertir una gran cantidad de tiempo y esfuerzo en el proceso de aprendizaje y en la aplicación del sistema de simulación.

II.2 CARACTERÍSTICAS DE LOS SIMULADORES

Las siguientes son algunas características de desempeño y aplicación necesarias en paquetes de simulación, para cumplir con los requerimientos en cada proceso de diseño, de acuerdo a la opinión de algunos autores **[12]**.

Características Generales

- La flexibilidad en la creación de modelos es de las características más importante en el software de simulación, dado que cada red es única. Si el paquete no tiene la capacidad requerida en una aplicación específica, entonces es necesario aproximar el sistema, resultando un modelo de dudosa validez.
- La facilidad en la creación de modelos, es la otra característica más importante para el software de simulación, dado lo corto del tiempo con que normalmente se cuenta para realizar el análisis de una red. Las interfaces de usuario pueden facilitar la captura de parámetros del modelo, normalmente mediante el uso de menús y el ratón. También es deseable que exista una jerarquía en el modelo (de tal manera que puedan construirse nuevas estructuras de modelado a partir de constructores base) y la reutilización de modelos.
- Se recomienda contar con herramientas adecuadas para análisis de código (debugg), que efectúe una verificación adecuada de que la estructura de los modelos sea completa y consistente.
- La velocidad de ejecución de los modelos es importante, especialmente cuando un gran número de "eventos" diferentes deben ser modelados. Importa también, que se puedan modelar redes grandes y que el software corra tanto en PC´s y WS´s, con compatibilidad a través de las distintas plataformas.
- La animación del proceso de simulación del modelo, es útil para encontrar problemas y para mostrar la esencia del modelo. En la animación, se representan los elementos clave de modelo (mensajes, nodos, ligas, etc.) mediante iconos que cambian de posición o de color, cuando hay cambios en el estado de la simulación, mostrándolos gráficamente. Algunos simuladores muestran en tiempo real la corrida de la simulación y otros la muestran fuera de línea (play back), donde la animación se muestra después de que se realizó la simulación, tomando la información de los cambios de estado de un archivo almacenado en disco.

Módulos preconstruidos

El tiempo necesario para construir un modelo, para simular una red, se puede reducir de manera considerable, si el software de simulación cuenta con módulos preconstruidos (de calidad) para ciertos dispositivos clave **[12]**, tanto de hardware como de protocolos. Algunos ejemplos de dichos módulos serian para representar una Ethernet, token ring, FDDI, puentes, enrutadores, TCP/IP, paquetes o satélites.

Características Estadísticas

Dado que todas las redes de comunicaciones presentan algún tipo de comportamiento aleatorio, un paquete de simulación debe contar con excelentes características estadísticas **[12]**, como por ejemplo:

- Contar con un excelente generador de números aleatorios, capas de generar múltiples secuencias de números aleatorios que se asignen, por ejemplo, a diferentes fuentes aleatorias de paquetes, por que de esta manera uno puede obtener mayor precisión estadística en la comparación de diferentes configuraciones de redes.
- En general, cada fuente de datos hacia la red, debe ser modelada escogiendo cuidadosamente la distribución de probabilidad que la represente de manera adecuada, y no sólo por valores promedio (como por ejemplo, en los tiempos entre envío de paquetes, el tamaño de los paquetes, etc.). Si hay un sistema existente, los datos deben ser recolectados y colocados dentro de una fuente aleatoria, en forma de una función de distribución de probabilidad aleatoria (por ejemplo una exponencial o Poisson); esta distribución es utilizada como modelo para las diferentes fuentes aleatorias en la simulación. Si ninguna distribución estándar representa adecuadamente a los datos, entonces se utiliza una distribución empírica creada a partir de los datos observados.

Generadores de Tráfico

La fuente más importante de datos aleatorios para la simulación de redes, esta normalmente asociada con la generación de tráfico **[12]**. Los siguientes métodos de generación de tráfico, son generalmente utilizados:

- El tiempo entre envío de mensajes y su longitud en un nodo, dependen de la aplicación a la cual representan (transferencia de archivos, e-mail, etc.), y puede depender de la recepción de mensajes de confirmación (acknowledge).
- El tiempo entre envío de mensajes y su longitud para un nodo particular, deben ser muestras independientes de su respectiva distribución de probabilidad (normalmente exponencial para tiempos entre llegadas).
- Los datos de tráfico se pueden leer al simulador por medio de herramientas para recolección de datos y análisis de protocolos, como por ejemplo: Sniffers, HP NetMetrix, RadCom y Bay Networks Trend Man.

Es posible definir de antemano errores en la red (breakdowns), en componentes del modelo como los nodos o las ligas. Por ejemplo, puede ser de interés analizar como puede reconfigurarse a sí misma una red, después de que ocurre un desastre natural.

Se requiere que los reportes de resultados del simulador sean generados automáticamente durante la simulación, para ahorrar tiempo. Deben incluir estadísticas de rendimiento, como la capacidad, la utilización, los retardos de punto-a-punto y el retardo en los buffers **[12]**. La presentación de resultados puede ser por medio de archivos con un cierto formato o mediante gráficas, de tal manera que estos datos faciliten análisis posteriores.

II.3 HERRAMIENTAS DE SIMULACIÓN (estado del arte) [12]

Lenguajes de propósito general

Los lenguajes de simulación de propósito general se aplican al modelado de diferentes tipos de sistemas, y algunos de ellos cuentan con módulos especiales para modelar específicamente sistemas de comunicaciones. Algunos ejemplos son: Arena, BONeS (Block Oriented Network Simulator) DESIGNER, GPSS/H, MODSIM II, SES/Workbench, SIMAN/Cinema V, SIMSCRIPT II.5 y SLAMSYSTEM. De estos, solamente BONeS y SES tienen módulos para representar sistemas de comunicaciones **[12]**.

El modelo se crea en el lenguaje de simulación, escribiendo un programa (empleando sintaxis o gráficos), utilizando los constructores propios del lenguaje que incluye entidades (mensajes), atributos (tipo de mensaje o destino), recursos (nodos y ligas) y buffers. Su mayor ventaja es la facilidad con que permiten modelar cualquier tipo de red, sin importar su complejidad o tamaño. Una de sus desventajas es la necesidad de que sean programados los modelos, lo que se complica si la red es grande.

Lenguajes de simulación orientados a las comunicaciones

Son lenguajes de simulación orientados completamente a las redes de comunicaciones. Un ejemplo de ellos se encuentra en el OPNET Modeler. Las ventajas que presentan son la reducción en el tiempo de programación y los constructores con que cuentan para el desarrollo de modelos, los cuales representan a los elementos que conforman dichos sistemas.

Su desventaja es que los modelos se complican conforme aumenta el tamaño de la red. Es importante contar con cierta experiencia en el manejo de dichos lenguajes para crear modelos de simulación adecuados **[12]**.

Simuladores orientados a las comunicaciones

En su forma más básica, son paquetes que permiten simular redes de comunicaciones específicas sin la necesidad de programar. Algunos ejemplos son: BONeS PlaNet, Comnet III, L·NET II.5 y NETWORK II.5 **[12]**.

El procedimiento de aplicación de los simuladores es el siguiente: se selecciona el tipo de red a modelar, dentro de las disponibles en el paquete, y los demás dispositivos necesarios, eligiéndolos de menús, cajas de diálogo o mediante gráficos. Se configura la red de acuerdo a los requerimientos a simular, dado que se tienen constructores para simular una LAN (Ethernet, Token Ring, etc.), para estaciones (PC o WS), para dispositivos de interconexión (puentes o enrutadores) y para generadores de tráfico. Se establecen los parámetros iniciales para cada dispositivo y se realiza la simulación.

Las ventajas que presentan dichos sistemas, son la reducción del tiempo de programación (comparado con los lenguajes de simulación) y sus constructores, los cuales son muy parecidos a los dispositivos reales y fáciles de utilizar.

Su principal desventaja es que están limitados a simular las redes que solamente permiten los bloques de construcción estándar del paquete. Hay sin embargo, algunos simuladores que permiten ya sea modificar los módulos existentes o bien crear nuevos módulos incrementando la flexibilidad de la modelación.

II.4 SOFTWARE PARA SIMULACIÓN

Existen sistemas de software para simulación de redes, tanto comerciales como públicos. Los simuladores comerciales, son herramientas orientadas a las comunicaciones y a las redes, con gran cantidad de módulos preconstruidos y múltiples servicios al cliente, los cuales tienen un costo y se adquieren en el mercado **[12]**. Los sistemas de software públicos son a los que se puede tener acceso libremente (sin cargo), que han sido probados exhaustivamente y se ofrecen tanto para uso académico como de investigación **[13]**.

II.5 SIMULADORES COMERCIALES

Estos sistemas ofrecen ventajas como la reducción en tiempo de programación de modelos, bloques preconstruidos para representar los elementos de la red y fácil utilización. Sin embargo, presentan inconvenientes importantes, como lo son su limitación a modelar solamente los tipos de redes definidas en el paquete y lo referente a los costos de los sistemas, los cuales varían desde U$129.00 que cuesta el LANModel para Windows, que pertenece al Network Performance Institute, hasta los U$40,000.00 que cuesta el COMNET III, para Windows 95, NT o UNIX, de CACI Products Co. Además, algunos módulos de elementos prediseñados se adquieren de manera independiente y sus precios varían desde U$1,000.00 hasta U$10,000.00 dólares, cada uno **[12]**.

La diferencia en precio es debida a su funcionalidad. Algunos son orientados al manejo de LAN´s y otros a WAN´s. Algunos ofrecen diagramas de red y simulaciones limitadas, mientras que otros ofrecen modelos globales más sofisticados. Pero la realidad es que ningún paquete lo hace todo. Si se requiere modelar, analizar y simular una red, es necesario adquirir múltiples productos. Hay notables diferencias entre productos que se supone realizan el mismo trabajo.

Comparación de simuladores comerciales

Prácticamente todos los simuladores se ofrecen con librerías de elementos de red, dispositivos y protocolos, pero no todos modelan lo mismo. Por ejemplo, CANE de ImageNet Inc., modela 9,000 dispositivos y estaciones, mientras que SimuNet de Telenix Corp., solamente se ofrece con librerías de ruteadores de Cisco Systems Inc. Otras herramientas de simulación pueden modelar ruteadores de Cisco, hubs, gateways y switches. Pero no todas pueden modelar enlaces WAN y LAN, al mismo tiempo **[12]**.

NetArchitec de Datametrics System Corp., incluye modelos para procesadores, controladores de discos y discos en sus librerías de elementos.

Protocolos

Desde el punto de vista de los protocolos para redes, los simuladores pueden representar tanto el IP, como el IPX (Novell), entre otros. También modelan los protocolos de la capa de red, como el IEEE 802.3 y 802.5, ATM y frame relay.

El simulador adecuado

Para elegir el simulador adecuado se debe examinar que tipo de elementos de red simula cada herramienta que constituye el paquete y a que nivel de abstracción lo realizan [12]. Aquí se han encontrado algunos resultados interesantes: la mayoría de los productos pueden simular los elementos de red que modelan, pero CANE de ImageNet´s, no puede simular discos o controladores de red; mientras que el simulador Virtual Agente de Network Tool´s Co., no puede mostrar el rendimiento de las colas de paquetes ni la velocidad media en ellos; por otra parte, SimuNet de Telenix, no puede mostrar detalles a nivel de dispositivos, como la arquitectura del hardware. Lo anterior muestra como los sistemas se encuentran limitados en algunas de sus funciones de simulación.

Con la excepción de NetArchitect, de Data-metric, ninguna de las herramientas mencionadas anteriormente simulan dispositivos a nivel sistema, esto es, que están limitadas en su habilidad para determinar cómo el rendimiento de cada uno de los dispositivos que la forman puede impactar al rendimiento global de la red.

Es interesante saber que los simuladores suelen estar limitados para determinar el impacto de esquemas de prioridad o de calidad de servicio, en el rendimiento de la red.

La mayoría de los simuladores proveen ejemplos de redes prediseñados, que ayudan al usuario para familiarizarse con cada producto y cuentan con demostraciones previas que dan alguna idea del funcionamiento de los sistemas, pero lo mejor es contar con el producto y aplicarlo.

La elección de la herramienta adecuada es difícil, y se complica cada día por la gran competencia existente en este rubro tecnológico, por lo que lo más recomendable es acudir a los distribuidores, solicitar la información detallada de cada sistema y de ser posible, obtener el paquete para su análisis por un periodo de tiempo, antes de la compra. Algunos distribuidores ofrecen los paquetes sin cargo alguno por períodos hasta de 30 días.

Ejemplos

La tabla muestra algunos ejemplos de simuladores comerciales **[12]**:

Nombre	Características	Precio o Localización
BONeS DESIGNER (Block Oriented Network Simulator)	❖ Lenguaje de simulación de propósito general, de ambiente gráfico. ❖ Permite modelar redes de comunicaciones. ❖ No requiere programación, excepto si se quieren diseñar nuevos elementos (C, C++). ❖ Sus bloques de construcción son estructuras de datos y diagramas de bloques. ❖ Los resultados que ofrece son medidas de utilización y desempeño, y se presentan como numéricos o gráficos. ❖ Se puede utilizar en plataformas: SUN-4(SPARC), RAM 32/64, 175 MB, SWAP 80 MB, HP 9000/700 O SUN Solaris.	❖ Licencia por usuario: U$15,000.00 ❖ Nuevos módulos: U$1,000.00 hasta U$10,000.00 ❖ Sistems & Networks, Sta. Clara CA ❖ Comdisco Systems, Inc. 1993.
Comnet III	❖ Herramienta para diseñadores de redes Orientada a Objetos (nodes, links, protocols and traffic generators). ❖ Predice el comportamiento de LAN's, WAN's, redes de datos y de voz. ❖ No requieren de programación (interfaz gráfica). ❖ Permite especificar opciones y detalles para cada uno de los distintos tipos de objetos mediante cajas de diálogo y menús. ❖ Los resultados que ofrece son gráficos y de datos. ❖ Algunos reportes no son claros y es difícil interpretar algunas configuraciones grandes. ❖ Se puede utilizar en plataformas: PC (486 o mayor), RAM 24, Windows 3.X, 95 o NT, Silicon Graphics, HP 9000/700, SUN (SPARC), IBM RS6000 y DEC.	❖ Licencia por usuario: U$29,000.00 ❖ http://www.caciasl.com ❖ Caci Products Company. ❖ La Jolla, CA,1993.
L·NET II.5	❖ Simulador para LAN's (estaciones y gateways). ❖ Contiene módulos prediseñados de Ethernet, Token Ring, 10BASE T y □DI. ❖ Produce reportes de mensajes transmitidos, colisiones, uso de canales y retardos promedio.	• Caci Products Company • La Jolla, CA,1993
NetMaker XA	❖ Conjunto de herramientas para toma de decisiones en redes. ❖ Simplifica el seguimiento, generación de reportes, análisis, diseño y planeación. ❖ Necesita de programación (C). ❖ Incluye librerías de elementos, específicas por vendedor. ❖ Requiere una estación SPARC.	❖ Licencia para una configuración típica: U$40,000.00 ❖ Precio base del producto: U$10,000.00. ❖ Precio de módulos opcionales: U$7,000.00-U$15,000.00. ❖ http://www.makesys.com
Network II.5	❖ Simulador para sistemas de cómputo y redes. ❖ Sus módulos de construcción son dispositivos de hardware y módulos de software. ❖ Los dispositivos son elementos de procesamiento, almacenamiento y transferencia. ❖ Los detalles y opciones para los dispositivos se especifican a través de cajas de diálogo. ❖ Los reportes incluyen la utilización de dispositivos, retardos promedio e información de la ejecución de los módulos de software.	• Caci Products Company • La Jolla, CA,1993

OPNET Modeler	❖ Lenguaje de simulación de propósito general. ❖ Utiliza editores para la red, los nodos y procesos, que generan el modelo. ❖ Permite modelar redes de comunicaciones. ❖ Requiere programación (C, C++), pero cuenta con módulos preconstruidos. ❖ Los resultados que ofrece son medidas de utilización y desempeño. ❖ Requisitos: SUN-4(SPARC), RAM 32, 120 MB, Compilador C, HP 9000/700, DEC, Alpha y Silicon Graphics.	❖ Licencia por usuario: U$16,000.00. ❖ Soporte y mantenimiento: U$3,000.00 anuales. ❖ Mil3 Inc. Washintong D.C. ❖ httphttp://www.mil3.com ❖
Optimal	❖ De Optimal Networks. ❖ Productos para Windows. ❖ Se utilizan para monitoreo, mapeo y modelado de redes corporativas. ❖ Soporta aplicaciones Java, HTTP, Oracle, Microsoft, FTP, SMTP, POP3, RPC V2, Novell NCPBurst y NFSv2. ❖ Requerimientos: Pentium 90 MHz., RAM 32, 100 MB +, Windows 95 o NT 4.0	❖ Optimal Network Corporation ❖ Mountain View, CA ❖ http://www.optimal.com
SES/Workbench (Scientific and Engineering, Inc.)	❖ Lenguaje de simulación de propósito general, gráfico. ❖ Su costo es bajo. ❖ Permite simular sistemas de cómputo y redes de comunicaciones. ❖ Sus módulos incluyen nodos, arcos o ligas y transacciones. ❖ Los parámetros de los elementos de la red pueden ser bien definidos. ❖ Se puede utilizar C para definir los detalles de nodos y arcos. ❖ Muestra resultados en formato numérico y gráfico, durante la simulación o al final de ésta, pero necesita que Excel este instalado.	❖ Scientific and Engineering Software, Inc. ❖ Austin TX, 1992 ❖ Licencia por usuario: U$9,995.00

II.6 SIMULADORES PÚBLICOS

El desarrollo de proyectos de investigación en un área tecnológica tan dinámica como las redes para comunicaciones, se ha convertido en un paradigma en centros de investigación públicos y privados, y forma parte ya de las matrículas educativas en prácticamente todas las Universidades del país y el extranjero, que trabajan en el área de ingeniería. De aquí, que sean las instituciones educativas las principales fuentes de sistemas para simulación desarrollados con el propósito de representar este tipo de sistemas.

Características

En general son gratuitos y se encuentran desarrollados en lenguaje C o C++. Aplican procedimientos generales de redes de comunicaciones, modelando los dispositivos más comunes, de acuerdo al nivel de diseño específico para cada uno de ellos (conmutadores, encauzadores, líneas de intercomunicación, etc.). Estos sistemas promueven y fortalecen la tendencia hacia el libre acceso al software, instituyendo nuevas formas para asegurar utilidades económicas a los autores.

Estos simuladores se adquieren de servidores públicos en la Internet, siendo generalmente propiedad de universidades e instituciones de investigación científica. Son ofrecidos en paquete junto con los programas fuente, los archivos de instalación y la documentación necesaria.

Objetivo

El objetivo de ofrecer dichos simuladores, es el de proporcionar métodos iniciales, exhaustivamente probados, para el desarrollo de sistemas de simulación aplicables a redes de cómputo, tanto de tipo experimental como redes reales.

Evolución

Los simuladores públicos presentan una evolución a través del tiempo en forma de diferentes versiones (como por ejemplo: 1.0, 1.1, 2.0, etc.), al igual que los simuladores comerciales. Dicha evolución, la establece el uso del sistema, adecuándose o mejorando gracias a la participación de su grupo de usuarios. Generalmente se crean estos grupos para cada herramienta, y ellos discuten los problemas y plantean soluciones generales, lo que se agrega al sistema enriqueciéndolo y permitiendo la evolución del mismo.

Los mismos simuladores, son versiones refinadas de otras herramientas, que han sido retomadas en sus planteamientos generales y se les han realizado adecuaciones para que abarquen las nuevas tecnologías y sus requerimientos de eficiencia.

Aplicación

Singularmente, la mayoría de estos simuladores han sido desarrollados para trabajar en estaciones de trabajo UNIX, como el Cnet, Scotty, Methusela, Smurph, Mars, etc.; algunos otros corren en PC´s, como el Simpack, el cual corre bajo DOS, Windows y OS2.

Tendencias

La tendencia en este tipo de simuladores, es claramente encaminada hacia la Programación Orientada a Objetos (OOP). Dado el tipo de análisis que se presenta para la aplicación de cada herramienta, en donde se resumen las características de cada dispositivo y se crea un elemento único (objeto), que interactúa con los demás que forman la red.

Ejemplos

La tabla presenta algunos ejemplos de estos simuladores:

Nombre	Características	Localización
Cnet	❖ Es un simulador de redes de comunicaciones. ❖ Su sintaxis es: cnet [opciones] {TOPOLOGY_FILE | \-r nodes} ❖ Este programa permite experimentar con varios protocolos para la capa de enlace, de red y de transporte. ❖ Provee diferentes aplicaciones y capas físicas, con diferentes características estáticas de generación de mensajes y de transmisión de datos. ❖ Acepta comandos de línea y posteriormente lee de un archivo de topología o genera una topología aleatoria para realizar la simulación. ❖ Compila y dinámicamente liga los archivos fuente que contienen la implantación de los protocolos para cada nodo. ❖ Los nodos pueden ser hosts o routers. ❖ Las ligas se conectan a los nodos mediante buffers y cada una tiene un costo (weight), para cada frame y byte transmitido. ❖ Cuenta con una librería de funciones de red. ❖ Utiliza un estilo de programación orientado al evento, que indica que nodo asociado con la red necesita atención. ❖ http://www.cs.uwa.edu.au	❖ Autor: Chris McDonald ❖ University of Western Australia ❖ chris@cs.uwa.edu.au.
MaRS (Maryland Routing Simulator)	❖ Es un simulador de eventos discretos creado para evaluar y comparar algoritmos de ruteo en redes. ❖ El sistema se caracteriza por trabajar por estados discretos y por cambios de estados (eventos), los cuales ocurren a instantes discretos de tiempo. ❖ Contiene componentes para representar ligas, dispositivos de enrutamiento, fuentes y consumidores de datos. ❖ Los eventos que maneja son de diferentes tipos como comando, evento, privado y registro. ❖ Utiliza una política de diseño Orientada a Objetos.	❖ University of Mariland, Florida, USA. ❖ Autor: System Design and Analisys Group ❖ Lista de usuarios: mars@cs.umd.edu
Methulsela	❖ Es un simulador para comparar entre varias políticas de migración de procesos, en redes balanceadas de computadoras. ❖ Recibe de entrada 9 argumentos en la línea de comandos, que indican el archivo de datos, la política de migración, el costo de cambiar el proceso (en segundos), etc. ❖ Las políticas de migración pueden ser: sin migración, con base en el nombre, con base en el tiempo transcurrido de ejecución y por la carga de trabajo en los hosts. ❖ Las políticas de depósito son: escoger el host con menor carga de trabajo y escoger el que tenga el menor tiempo total de ejecución. ❖ Ofrece como salidas de resultados: las entradas, los procesos ejecutados durante la simulación, el retardo en ejecución, la cantidad de tiempo de CPU utilizada, el número total de migraciones, etc.	❖ Berkeley University, USA. ❖ Autor: A. Downey (downey@cs.berkeley.edu)

Real	❖ Simulador para estudiar el comportamiento dinámico de esquemas de control de flujo y de congestión, en redes de switcheo de paquetes. ❖ Toma un escenario de entrada, que es la topología, protocolos, carga de trabajo y parámetros de control. ❖ Ofrece estadísticas como el número de paquetes enviados, el retardo en las colas, el número de paquetes rechazados y retransmitidos, etc. ❖ Los escenarios se describen utilizando NetLanguage, que es una representación ASCII simple de la topología de la red. ❖ Se define el control de flujo y la carga de trabajo de cada fuente. ❖ Existen diferentes tipos de fuentes: genéricas, telnet, jk_tahoe, jk_reno, dec, non_flow_controlled, poisson, background, controlled_rate, random_rate y mmpp. ❖ Los reportes que genera incluyen el número de paquetes transmitidos, los retardos en las colas, el número de paquetes retransmitidos, tirados, etc. ❖ Los reportes son numéricos y gráficos.	
Scotty	❖ Este es un interprete de código TCL con extensiones, para obtener el estado e información de configuración de redes TCP/IP.	❖ Technical University of Braunschweig. ❖ Autor: Juerguen Schoenwaelder (schoenw@cs.utwente.nl)
Simpack	❖ Es una colección de herramientas en C (programas y funciones), para simulación en computadora. ❖ El objetivo es proveer un punto de inicio para simular un sistema. ❖ Incluye temas como multimodelado, modelado declarativo, modelado diferencial, modelado funcional, librería de colas, eventos, etc. ❖ http://www.cis.ufl.edu/~fishwick	❖ University of Florida, USA. ❖ Autor: Paul A. Fishwick (fishwick@cis.ufl.edu).
Smurph (System for Modeling Unslotted Real-time Phenomena)	❖ Es un paquete para simular protocolos de comunicación a nivel MAC (Medium Access Control). ❖ Puede entenderse como la combinación de un lenguaje de especificación de protocolos en C++ y un manejador de eventos. Es un simulador de tiempo discreto que provee un ambiente virtual pero realista para la ejecución de protocolos. ❖ Se puede utilizar para diseñar protocolos de comunicación de bajo nivel y analizar su rendimiento y calidad. ❖ Requerimientos: corre en equipos SUN-3, SPARC-stations (bajo UNIX 4.1.1 y mayores), SGI (IRIX4.0.5), MIPS (UMIPS 5.0) y Apple. ❖ ftp://menaik.cs.ualberta.ca	❖ University of Alberta, CANADA ❖ Autor: Pawel Gburzynski (pawel@cs.ualberta.ca)
VISTA	❖ Es una herramienta configurable para visualización y simulación, para switches ATM. ❖ Soporta arquitecturas de switcheo tandem Banyan y knockout. ❖ Maneja tráfico geométricamente distribuido. ❖ Utiliza esquemas de buffers: pushout, sharing y selective discarding. ❖ Permite visualizar la arquitectura de switches, transferencia de datos y de video. ❖ Reporta la distribución dinámica de celdas perdidas, el retardo promedio y el rendimiento promedio. ❖ http://vip.cs.utsa.edu/systems/atm.html	❖ University of Texas ❖ Autores: Narayana Tummala & Kay A. Robbins.

RESUMEN

En general, los sistemas descritos utilizan distintos modelos para formar y representar los diferentes dispositivos en las redes de computadoras. Algunos cuentan con "librerías" específicas para representar cada dispositivo, con parámetros predefinidos que incluyen las principales características tecnológicas de cada fabricante. Cada dispositivo es "llamado" para formar parte de la red y colaborar en conjunto, representando en un ambiente virtual el comportamiento global del sistema.

El avance tecnológico, demanda que las representaciones de los dispositivos sea actualizada constantemente, para hacer frente a los cambios que se generan con las nuevas demandas de transmisión de datos y transferencia de información en general.

Pero los modelos analizados para los dispositivos de simuladores públicos (de los comerciales no se tiene el código fuente), no representan de manera adecuada el fenómeno más importante en cuestión de rendimiento, el "retardo", causado por los dispositivos que intervienen a nivel de enlace y de red en la transmisión. En este sentido se analizarán en el siguiente capítulo algunos modelos para representar las redes, y se sugerirá un método para efectuar una descripción adecuada del fenómeno de retardo y obtener el rendimiento.

Capítulo III

III.1 El modelo

El modelo es la abstracción del sistema y su realismo esta estrechamente ligado al nivel de detalle utilizado en su definición [7]. Si conocemos todo acerca de un sistema e invertimos tiempo y esfuerzo para diluir su complejidad construyendo su modelo, dicha abstracción puede ser muy parecida a la realidad. En la mayoría de los casos, se desea construir un modelo que se enfoque a cierto(s) elemento(s) de interés, dejando el resto del sistema como una interfaz sin detallar sus propiedades.

III.2 El Sistema

El **"sistema"** como se ha denominado, es el dispositivo del mundo real que deseamos modelar (un cajero automático, un banco, o cualquier dispositivo tangible o proceso). Generalmente, un sistema puede ser considerado como un grupo unido de objetos para realizar algún conjunto de funciones o procesos, en donde el modelo de éste, es una abstracción del sistema, que plasma sus elementos más importantes y sus relaciones internas. La figura muestra las interacciones entre el modelo y el sistema real.

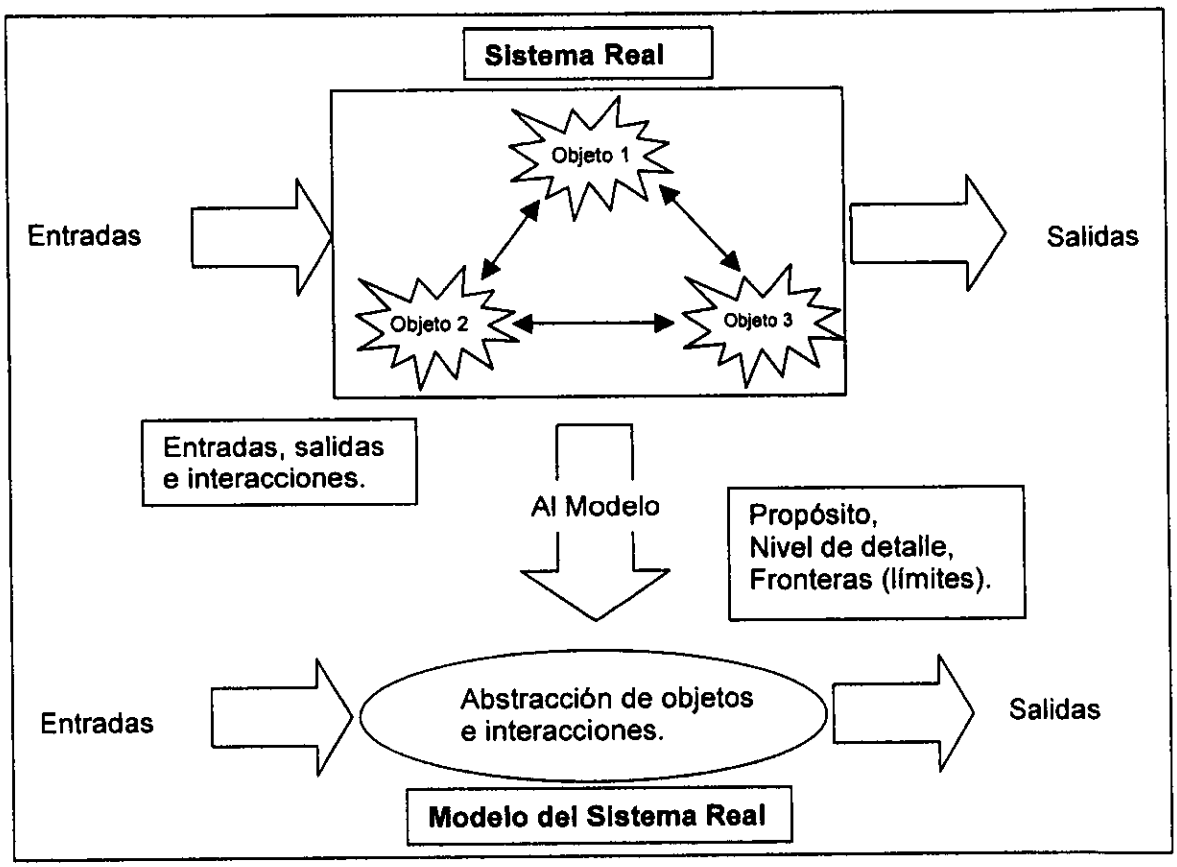


Figura 3.1. Descripción de la creación de un Modelo.

Lo importante de esta abstracción, es que el modelo es una representación subjetiva, desde algún punto de vista definido, del sistema. El punto de vista (nivel de abstracción), permite definir lo que es importante, el propósito u objetivo del modelo, los detalles y las fronteras o límites del mismo.

III.3 CONSTRUCCIÓN DEL MODELO

Es necesario seguir cierta metodología para construir un modelo, cuya representación sea correcta. La metodología utilizada comúnmente, es la del análisis "top-down", que es una descomposición del sistema orientada a conseguir el objetivo del modelo. El objetivo permite definir los límites y componentes del sistema, generando el modelo con el nivel de detalle requerido. Este es un método interactivo de generación de objetivos, límites y niveles de abstracción, donde se procura enfocarse a los elementos críticos del sistema.

Las entradas se derivan del sistema en estudio, así como de las mediciones que se desean extraer, es decir, de la utilización misma del modelo (de aquí la naturaleza experimental del modelo). La definición de las mediciones de rendimiento requeridas, redefinen el diseño del modelo. Este proceso de rediseño, continúa hasta que el analista ha alcanzado un nivel de detalle consistente con el definido en la planeación del objetivo del modelo. El nivel de detalle indica la importancia de cada componente a los ojos del analista, como un punto que debe ser evaluado.

La metodología para la construcción y uso del modelo, en general **[7]**, es la siguiente:

1. Definir el objeto a modelar según el criterio de análisis.
2. Definir y/o redefinir el modelo del sistema (incluyendo la creación de abstracciones del sistema en el campo matemático, lógico y de las relaciones procedurales).
3. Colectar los datos de entrada al modelo.
4. Seleccionar una herramienta de simulación – preparación del modelo para implementarlo en la herramienta.
5. Verificar que la implementación en la herramienta es un reflejo adecuado del modelo.
6. Validar que la implementación en la herramienta provee una adecuada correspondencia con el sistema del mundo real modelado.
7. Experimentar con el modelo para obtener las mediciones de rendimiento solicitadas.
8. Analizar los resultados de la herramienta.
9. Utilizar dichos resultados para crear diseños o mejoras para el sistema en el mundo real.

Un análisis en este sentido es útil, cuando el modelo ofrece información factible de aplicar en el producto final. Las salidas pueden consolidar los conceptos o nociones del sistema, definiendo diferencias, ofreciendo mejoras o corroborando con otra información del sistema.

III.4 HERRAMIENTAS DE MODELADO

Actualmente, existen diferentes herramientas para el modelado, como son las herramientas analíticas, las de simulación, las de análisis operacional y las de examen o evaluación de sistemas [7]. Cada una tiene su aplicación específica y sus razones para ser utilizadas. En este trabajo se utilizan las analíticas dada su generalidad y aplicabilidad y por las características propias de los sistemas de red.

Es necesario apuntar que las diferentes herramientas analizadas en este trabajo, utilizan un sistema de definición de elementos y procesos similares. Los cuales, dichos sea de paso, no varían de manera significativa.

Estas herramientas se utilizan como técnicas de implementación de modelos que requieren cierto tiempo de creación y la principal razón para utilizarlas es su generalidad. Las implantaciones analíticas dependen de la habilidad del analista para describir un modelo en términos matemáticos.

Ventajas al utilizar Modelos Analíticos

- Los modelos analíticos capturan los detalles más sobresalientes de los sistemas.
- Las suposiciones en el análisis de un sistema, son reales y fáciles de describir.
- Los algoritmos para resolver ecuaciones (por ejemplo, de las líneas de espera), están disponibles empaquetadas en forma de *máquinas de análisis*.

III.5 MODELO ANALÍTICO PARA LINEAS DE ESPERA (COLAS)

Algunos Sistemas del mundo real, pueden describirse como un conjunto de líneas de espera (colas), en donde se forman los clientes con forme llegan en espera de obtener un servicio. Para estos casos, lo recomendable es realizar una representación del Sistema utilizando la *Teoría de Colas* para resolver el problema de modelado. Dicha teoría se utiliza en el estudio de líneas de espera dinámicas y es ideal para Sistemas de Transferencia de Información, como las redes **[14],[15]**.

El Análisis de Colas de Espera al más simple nivel (una sola cola), se puede ver como una línea de espera que recibe clientes, por intervalos de tiempo (llegadas); la cuál es atendida por un servidor, que extrae clientes de la cola a una cierta razón o velocidad y los atiende (servicio). El siguiente diagrama muestra el modelo genérico de una cola.

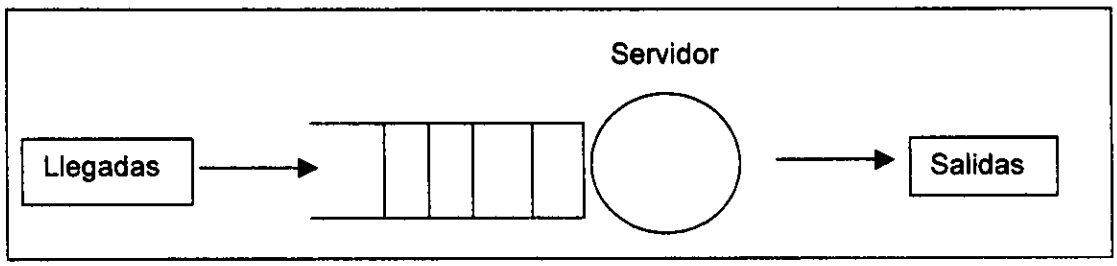


Figura 3.2. Cola o línea de espera

Las líneas de espera, operan de la siguiente manera: se recibe una entrada al sistema (cliente), si el servidor está ocupado el cliente es colocado en un lugar de espera (cola), tomando el lugar que le corresponda de acuerdo a la política de operación definida, esto sucede hasta que la cola se llena, en cuyo caso el cliente es rechazado (no hay espacio). Por otro lado, si la cola está vacía, o ha llegado el turno al cliente éste atendido. Entonces el cliente sale de la cola.

REPRESENTACIÓN

La notación utilizada para definir el fenómeno, es la siguiente: la distribución de llegadas, define el patrón de llegada de clientes a la cola **[15]**. Esta distribución, esta definida mediante una variable aleatoria, que establece el tiempo entre llegadas. Una función típicamente utilizada es la distribución Poisson, definida como: $P[\text{llegada} \leq \text{tiempo}] = 1 - e^{-\lambda t}$; donde λ es la razón de llegadas promedio (ej. tres clientes por minuto).

La cola o línea de espera, es definida como un reservorio de espera para los clientes. Adicionalmente, se define una política para aceptar y remover clientes de la cola. Algunos ejemplos de disciplinas de funcionamiento para las colas de espera son: el primero en llegar es el primero en salir (FIFO), el último en llegar es el primero en salir (LIFO), aleatorias o por prioridad. El último componente del Sistema de Cola de Espera, es la descripción de la política de servicio, que se refiere al método por el cuál los clientes son atendidos y la duración del servicio.

El tiempo de servicio es descrito por una distribución de probabilidad. Una típica distribución de servicio es la de Servicio Aleatorio, definida como $Ws(t) = 1 - e^{-\mu t}$, donde $t>0$, y el símbolo μ es utilizado para describir la relación promedio de servicio (ej. dos clientes atendidos por minuto).

Las distribuciones para describir las llegadas y el servicio promedio son muchas, algunos ejemplos son: la Distribución Exponencial, la Distribución General, la Distribución Erlang, la Determinística o la hiper-exponencial.

NOTACIÓN

La notación Kendall **[15]** se utiliza para describir el tipo de cola utilizada en un modelo, sus características de ingreso y servicio y sus capacidades respectivas. La forma de esta notación es:

A/B/c/K/m/Z

Donde ***A*** especifica la distribución del tiempo de llegada, ***B*** la distribución del tiempo de servicio, ***c*** el número de servidores, ***K*** la capacidad del sistema, ***m*** es el número en la fuente y ***Z*** es la disciplina en la cola.

Este tipo de análisis puede ser utilizado para generar estadísticas del tiempo de espera promedio, de la longitud promedio de la cola, del tiempo de servicio promedio, de la intensidad del tráfico, la utilización de los servidores, el tiempo promedio que pasa un cliente en el sistema, utilizando las probabilidades de sus tiempos de espera y tiempos de servicio medios.

III.6 DESCRIPCIÓN DE UNA RED EN TÉRMINOS ANALÍTICOS

Desde el principio, se estableció el nivel de abstracción a utilizar en el presente trabajo. En este momento, se utilizarán dichas definiciones para la creación del modelo para redes.

Desde el punto de vista analítico, se puede describir una red mediante una cola simple, de la siguiente manera:

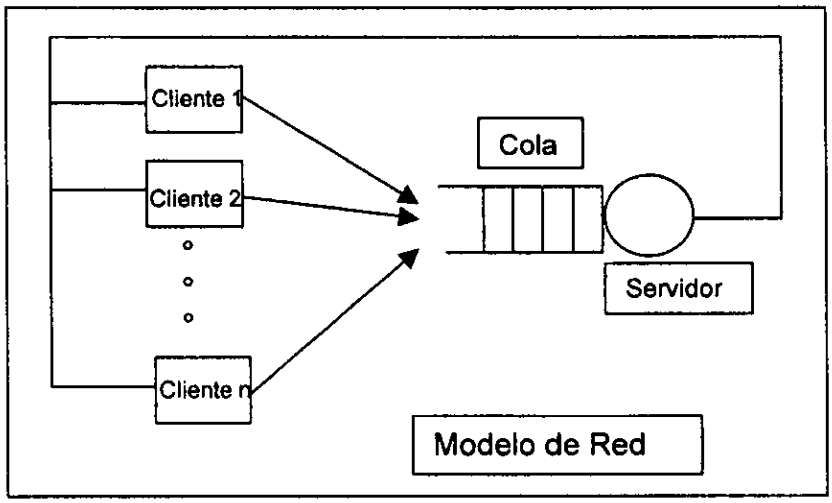


Figura 3.3. Modelo de una Red de Transmisión de Datos.

Como se muestra, la red se torna en un servidor que recibe paquetes de información que serán transferidos de una computadora a otra (Cliente 1, Cliente 2, etc.), entre las que forman la red.

Considerando un nivel de abstracción menor, es decir con mayor detalle, es posible modelar una red hasta el nivel de transferencia de bits. Con éste tipo de modelos, es posible examinar hasta el más mínimo detalle del diseño de una red nueva o existente. Sin embargo, el realizar éste tipo de modelado puede resultar muy costoso en términos de tiempo de desarrollo, codificación, pruebas, utilización y análisis de resultados.

De manera más realista, se eligió un nivel de abstracción situado entre el que considera gran detalle y el mostrado en el diagrama anterior, para esta investigación.

III.7 CALIDAD

Relacionada con el nivel de detalle, esta definida la calidad del modelo. Lo cual resulta un tanto complicado de entender. Se puede decir que un modelo con calidad satisface todos los requerimientos de validación y verificación, respecto del Sistema que representa. De cualquier manera, lo anterior se aplica a las Redes en términos de rendimiento, es decir, que si el modelo puede decir los detalles a del rendimiento de la red, o no. Los modelos de redes, deben ser herramientas útiles para determinar la capacidad de almacenamiento de sistemas secundarios, de canales de E/S, de las unidades de interfaz y de la red en sí. Los procesos o programas presentes en el sistema, demandan servicio de dichos recursos. El mayor problema de rendimiento que es posible estudiar con una herramienta de modelación como ésta, es la competencia por servicio de parte de los procesos que interactúan en la red.

Algunas mediciones típicas que se analizan en los Sistemas de Transferencia de Datos, están relacionadas con los retardos asociados a los *buffers* (colas) de los dispositivos que forman la red **[15]**.

Como se mencionó, la Teoría de Colas ofrece una estructura de fórmulas matemáticas para modelar, construir y resolver problemas del tipo **cliente-servidor**, que aplica actualmente al esquema de Transferencia de Información de las Redes actuales.

Algunas mediciones de rendimiento **[15]** típicas que es posible determinar mediante los modelos (que son precisamente las que se ofrecen), incluyen:

- El tiempo total en el sistema, de todos los mensajes.
- El rendimiento de las líneas de transmisión.
- La utilización de las líneas.
- El tiempo promedio de transferencia de los datos.
- La desviación estándar de la transferencia.
- El efecto de diferentes políticas de encolamiento en los retardos.
- El efecto de diferentes disciplinas de servicio.
- Los tiempos de ida y vuelta (para transferencia de mensajes) mínimo, máximo y promedio.
- El efecto de las relaciones entre dispositivos.
- Etc.

III.8 MODELO ANALÍTICO

Con los Modelos Analíticos se tienen las siguientes ventajas:

- Experimentación controlada.
- Expansión y compresión en tiempo real.
- Análisis de sensibilidad.
- Preajustes de los sistemas.
- Rendimiento y ajuste de componentes.
- Interacciones con el entorno.
- Análisis de "cuellos de botella".
- Herramientas de configuración.
- Análisis de rendimiento y ajuste.

En el presente Trabajo de Investigación se utilizan modelos analíticos para definir los componentes que forman la red, dado que ofrecen resultados en términos de rendimiento expresados mediante ecuaciones analíticas. Hasta el momento, lo mejor para evaluar Sistemas de Redes para Transferencia de Información se encuentra en la Teoría de Colas. En este tipo de análisis, el Sistema de Red se ve como un sistema de recursos múltiples con clientes y servicios.

III.9 MODELO DE LA RED

Una red puede ser considerada como la interconexión de una serie de elementos de cómputo (sistemas, terminales, etc.), que junto con una serie de facilidades de transferencia de información provee comunicación interna y externa a la red. Los atributos básicos de una red, que así la distinguen son su arquitectura o topología, su composición de dispositivos, tamaño o capacidad, tipos de canales, estrategias de uso y mecanismos de control **[2]**.

Utilizando la nomenclatura y taxonomía de Anderson para estructuras de interconexión de computadoras, un sistema específico puede ser caracterizado por su estrategia de transferencia (directa o indirecta), su mecanismo de control de transferencia (centralizado o descentralizado) y su estructura de rutas de transferencia (dedicada o compartida). Varias topologías de red, como anillos, bus y estrella, pueden ser vistas como combinaciones únicas de las características antes mencionadas.

Composición

La composición de las redes puede ser heterogénea u homogénea, dependiendo de la capacidad en los nodos o de los elementos de cómputo agregados. El tamaño de la red, generalmente se refiere al número de nodos o elementos de cómputo que la forman. Con respecto a los canales de comunicación, una red puede ser homogénea o emplear una variedad de medios (ej. fibra, par trenzado, microondas, etc). El control de la red y la administración puede ser centralizada o completamente distribuida, actuando en las diferentes topologías (BUS, ANILLO o ESTRELLA), de acuerdo a su propio funcionamiento.

III.10 COMPONENTES

Como primer paso en el desarrollo de un modelo de simulación [7], se propone un modelo de red como el siguiente:

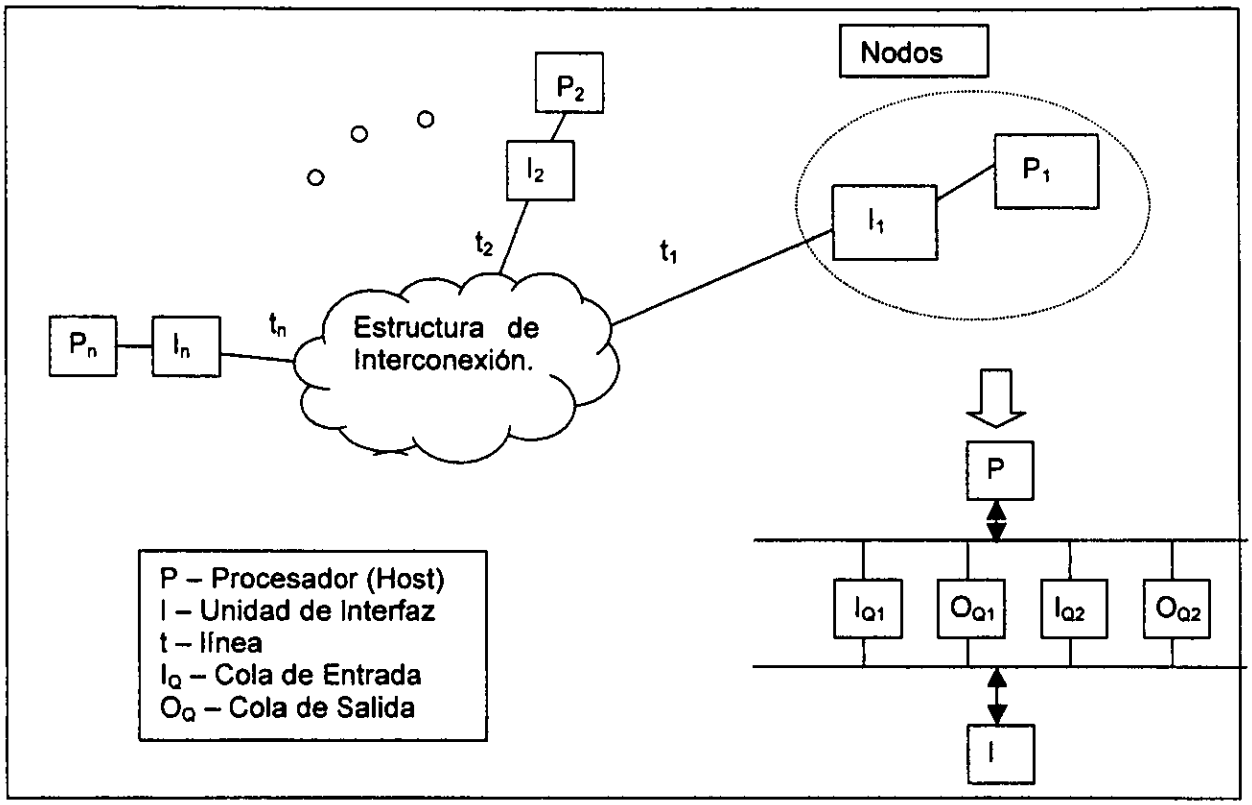


Figura 3.4. Modelo de Red.

Una red de Área Amplia (WAN), consiste de un número variable de nodos interconectados. Cada nodo consiste de una o más computadoras servidores, o varios procesadores conectados a un procesador central *(front-end)* independiente, ofrecido como una Unidad de Interface (UI).

Los servidores son los productores y consumidores de todos los mensajes y representan sistemas independientes, terminales, o puentes de enlace (gateways) a otras redes y otras instancias de elementos de cómputo. Las Unidades de Interfaz (UI), controlan todas las funciones de comunicación nodales y de la red, como manejo de mensajes, control de flujo y reconfiguración del sistema. Las líneas representan el medio físico de transmisión que interconectan los nodos. Las UI's junto con las líneas de interconexión, forman la estructura de comunicación de la subred.

RESTRICCIONES

En éste nivel de abstracción no hay distinción entre las instancias de mensajes, como lo son los bloques de datos y los mensajes de confirmación, todos se toman como el elemento genérico mensaje, que puede variar en longitud, únicamente.

III.11 ELEMENTOS DE LARED

A continuación, se describen los elementos a considerarse en este trabajo, con la finalidad de refinar la definición de cada uno de los componentes del modelo **[7]**.

Servidores

Los servidores son componentes que generalmente incluyen elementos de cómputo y control, con varios niveles de memoria y periféricos de entrada y salida. A lo que el modelo del sistema se refiere, se considera de manera independiente a cada procesador, reflejándolo con una función de distribución apropiada que describa la razón a la cuál el procesador produce y consume mensajes. Éstas funciones reflejan un poder de procesamiento inherente al procesador, establecido mediante la definición de los parámetros del procesador, los niveles exógenos de comunicación y las intercomunicaciones propias de la red.

Líneas de espera de servicio (colas)

Las líneas de espera de servicio comparten estructuras de almacenamiento de memoria (buffers), donde se deposita la información que se desea transferir entre los procesadores y las UI´s. Para cada nodo, puede existir una línea de salida, que almacene los mensajes que se encuentran en espera de ser transmitidos, así como una o más líneas de entrada de mensajes, conteniendo recepciones no procesadas. La memoria asignada a las líneas de espera, puede estar localizada en el procesador o en la UI, dependiendo de la implementación. Funcionalmente, ambas son equivalentes.

Asociado a las colas de espera, está el control de variables que son mantenidas y monitoreadas por ambos, el procesador y las UI's para ofrecer acceso a todas las colas, de manera simultanea o asíncrona. Los tipos más comunes son las colas lineales, las circulares y las ligadas. Las colas de espera lineales (buffers), son utilizadas cuando la longitud de un mensaje es conocida y se puede realizar una reservación de antemano. El uso de buffers circulares es apropiado cuando se manejan mensajes de longitud indeterminada, que son almacenados después de que otro es procesado. Un conjunto de colas ligadas, se utiliza si la longitud de los mensajes y los tiempos de llegada de los mismos, varían en rangos amplios que son imposibles de predeterminar de antemano, y los mensajes no son eliminados en el orden en que llegan.

Los mensajes son depositados (escritos) y extraídos (leídos) de las colas, utilizando varias estrategias como FIFO, LIFO, el mensaje más largo primero, aleatorios o por prioridad.

El acceso a las colas de espera, está controlado en atención a prevenir la escritura en una cola que se encuentre llena, le lectura de una cola que se encuentre vacía y la lectura de información que ya fue leída.

Unidades de Interfaz (UI)

En lo que a la red se refiere, la Unidad de Interfaz es la parte más compleja con respecto al hardware y al software. La función básica de la UI, es permitirle a su procesador comunicarse con los demás en la red, además de contribuir al funcionamiento general de la red. Esto incluye (re)inicialización del sistema, control de flujo, detección de errores y administración **[7]**.

Cuando la UI detecta que su procesador tiene un mensaje para enviar, adecua el mensaje (formato) para su transmisión, y trabaja para obtener de manera exclusiva el uso de los canal de comunicación. Una vez obtenido el control, el controlador transmite el mensaje y dependiendo de la implementación puede esperar una respuesta del procesador destino.

Utilizado el recurso (ej.. el Bus), la UI es capaz de pasar el control al siguiente candidato, de acuerdo al esquema de distribución. Si la UI falla, las otras unidades deben ser capaces de sustituirla, en cuanto a las responsabilidades del control de la red se refiere.

Líneas de Comunicación

Las líneas son las conexiones entre los nodos de la red, por donde viaja la información a transmitir y los datos de control. Son términos equivalentes, "canal" o "circuito". Un circuito particular puede ser unidireccional o bidireccional (por su naturaleza o su uso) y soportar transmisiones continuas obtenidas mediante técnicas analógicas o digitales.

Los circuitos son soportados utilizando una variedad de medios, como cables coaxiales, par trenzado, fibra óptica, enlaces de microondas, enlaces láser, enlaces infrarrojos, etc. Para el propósito del modelo, no es necesario profundizar en estas características de bajo nivel, excepto que estos medios están representados por un conjunto de características de canal como: la velocidad máxima a la que pueden viajar los datos, los parámetros de retardo y error, y las limitaciones direccionales.

Es importante considerar las características de configuración, como por ejemplo, si un circuito punto-apunto esta siempre dedicado a una red o no. Estas características pueden incluir los mecanismos de señalización y retardo, configuración del retardo en el circuito y el retardo por la ruptura de un circuito. Pero éstas características no son un factor determinante en el modelo.

La velocidad de transmisión, influye en la capacidad de transmisión de la línea, pero no es la velocidad de la red a la que efectivamente la información es transmitida. Siempre hay un tiempo extra de procesamiento (overhead). Existe degradación en la transmisión de datos debido a varios factores, como fallas lógicas en las interfaces electrónicas y daños físicos, que causan errores en la transmisión (repeticiones, retransmisiones, etc.), desde un bit hasta una falla total de la línea. Dependiendo del tipo de línea que se utilice, los errores varían desde 1 bit en 10,000 hasta 1 en 10,000,000 de bits, transmitidos.

Estructuras de Interconexión

Aspectos como el programa de actividades (schedule) de las redes, el enrutamiento de mensajes y la reconfiguración están íntimamente relacionados con la estructura de interconexión física de la red [7]. Por ejemplo, la elección de una línea para envío, esta relacionada con la posición en la red de los componentes que intervienen en la transferencia. El tiempo que tarda la transferencia de un mensaje depende de la posición de los procesadores en cuestión, y de los elementos de enrutamiento y redireccionamiento intermedios.

El funcionamiento continuo de la red, puede depender de que existan o no, líneas redundantes. Estas estructuras pueden ser representada mediante la organización topológica de tres aspectos de hardware: 1) los nodos que forman la red, 2) los conmutadores y enrutadores y 3) las rutas que se definen. Estos elementos se encuentran integrados en la transferencia de información entre procesos de nodos diferentes.

Esta transferencia se denomina "transmisión de mensajes" y no distingue entre instancias de mensajes, como son los bloques de datos, los mensajes de confirmación de servicios, de semáforos, etc.; sin embargo, en estricto apego a las consideraciones estructurales de la red es necesario distinguir entre los elementos de cómputo y su interface, ellos se juntan en este caso, como una sola entidad definida como un "nodo".

Los elementos de conmutación y enrutamiento, pueden ser vistos como "inteligencias interventoras" entre la fuente y el destino de un mensaje, los cuales afectan su ruta y puntos intermedios.

La figura 3.5, muestra un modelo general para los sistemas de interconexión **[17]**. Por simplicidad, se representa un sistema con un solo conmutador.

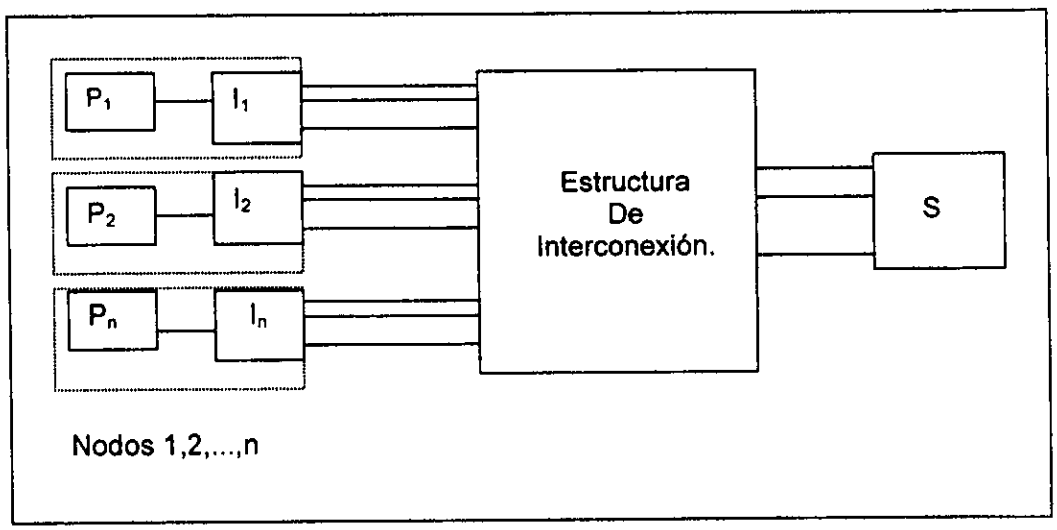


Figura 3.5. Interconexión de una red.

Hay una serie de rutas y ligas asociadas a cada nodo y conmutador. Cada nodo puede conectarse al resto de la red a través de una, dos o múltiples líneas, correspondientes por ejemplo, a un sistema de bus, a un anillo, a una estrella o a una red totalmente interconectada con líneas directas entre cada par de nodos.

La capacidad de abstraer dispositivos reales de red, en modelos de estudio, necesita de una metodología teórica como la Teoría de Colas de Espera, la cual esta dirigida a la evaluación del rendimiento de sistemas de proceso discreto. De aquí surge la necesidad de describir de manera general una tecnología de transmisión digital de datos, a la cual tomar como ejemplo, para probar los modelos desarrollados en este trabajo con dicha teoría. El siguiente capítulo se refiere a esta descripción general.

Capitulo IV

En este capítulo se describirán algunos de los fundamentos de la transferencia de información de manera digital y de la tecnología SONET. Esto tiene la finalidad de definir los rangos de funcionamiento de los dispositivos de enlace y enrutamiento propios de esta tecnología y de esta manera contar con la referencia real de operación para comparar los dispositivos que se ofrecen en este trabajo de investigación.

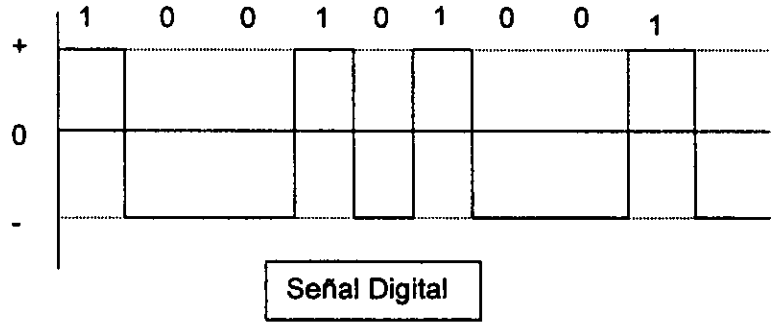
IV.1 SERVICIOS DE TRANSMISIÓN DIGITAL

Hay dos tipos de servicios de transmisión de datos utilizados generalmente en comunicaciones **[16]**, cada uno de ellos presenta un panorama distinto acerca de la manera de enviar información de un punto a otro dentro de una red. Estos servicios son:

- Servicios Analógicos
- Servicios Digitales

Señales Digitales

Una señal analógica es una onda senoidal electromagnética que cambia de manera continua. Por otro lado, una señal digital es una secuencia de pulsos de voltajes (u ópticos) que son transmitidos sobre algún medio. Un voltaje constante positivo indica un 1 binario, mientras que un voltaje constante negativo indica un 0 binario.



Señal Digital

Jerarquía Digital

La Jerarquía Digital, define el número de velocidades de transmisión digital que se utilizan en diferentes facilidades de transmisión **[16]**. Hay dos jerarquías definidas:

- North American Digital Hierarchy
- CEPT Digital Transmision Hierarchy

El elemento básico en la construcción de cada jerarquía es el canal de voz, que requiere de 64 kbps. Esta capacidad se obtiene aplicando una modulación de pulsos codificados (PCM). Actualmente, PCM se define como el estándar mundial para convertir una señal de voz analógica a su equivalente digital. La señal analógica se muestrea a una velocidad de 8,000 muestras/seg. La amplitud de cada muestra se representa mediante un valor que varía entre 0 y 255, es decir, se utilizan 8 bits para su representación.

North American Digital Hierarchy

En esta jerarquía, un canal simple de voz es conocido como la señal digital cero, o DS-0. La tabla muestra la jerarquía utilizada en Norte América. Cada nivel de señal digital tiene su propio formato y soporta diferente número de circuitos de voz.

Signal Level	Digital Bit Rate	Total Voice Circuits	Carrier System
DS-0	64 Kbps	1	(none)
DS-1	1.544 Mbps	24	T-1
DS-1C	3.152 Mbps	48 (or 2 DS-1s)	T-1C
DS-2	6.312 Mbps	96 (or 4 DS-1s)	T-2
DS-3	44.736 Mbps	672 (or 28 DS-1s)	T-3
DS-4	274.760 Mbps	4032 (or 168 DS-1s)	T-4

CEPT Digital Transmision Hierarchy

La *Conference on European Posts and Telecomunications* (CEPT), definió la jerarquía para transmisión digital que se utiliza en Europa. En esta jerarquía, un canal sencillo de voz se conoce como E-0. La tabla muestra la jerarquía CEPT.

Signal Level	Digital Bit Rate	Total Voice Circuits	Carrier System
0	64 Kbps	1	(none)
1	2.048 Mbps	30	E-1
2	8.448 Mbps	120 (or 4 E-1s)	E-2
3	34.368 Mbps	480 (or 16 E-1s)	E-3
4	139.264 Mbps	1920 (or 64 E-1s)	E-4

IV.2 Conmutación

La jerarquía digital se creó utilizando una serie de conmutadores de división de tiempo. Un conmutador es un dispositivo que combina dos o más señales para su transmisión simultánea sobre un solo canal de comunicación **[16]**.

Por ejemplo, cuando se utiliza la jerarquía Norte Americana, una señal DS-1 se crea con la combinación de 24 DS-0. Cuatro DS-1 se combinan para crear un DS-2. Siete DS-2 para formar un DS-3. Finalmente, seis DS-3 se combinan para crear un DS-4.

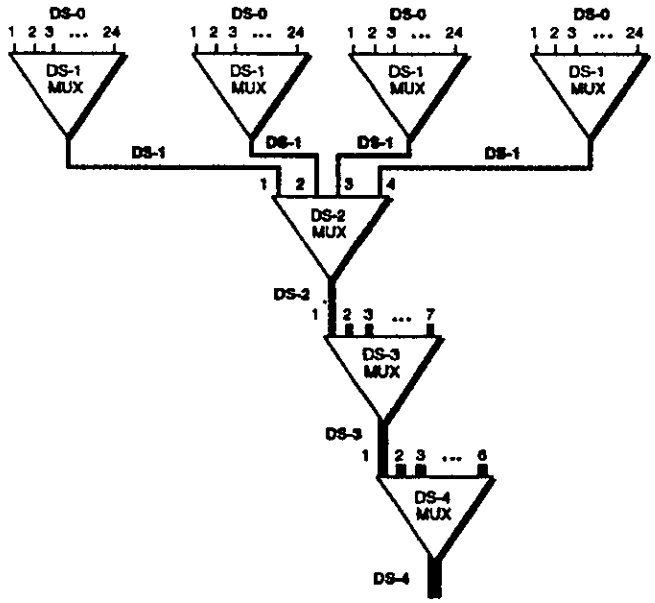


Figura 4.1. Esquema de Conmutación.

IV.3 Conmutación en Divisiones de Tiempo (TDM)

TDM es una técnica de transmisión digital que permite a muchos canales compartir una sola línea de comunicación. Utilizando TDM, una línea de transmisión de alta capacidad puede transmitir una mezcla de voz, video y datos.

Utilizando TDM, el ancho de banda de una línea de alta capacidad se divide en pedazos de tiempo sincronizados **[16]**. A cada canal se le asigna de antemano un numero definido de pedazos de tiempo. Por ejemplo, a un canal con bajo requerimiento de ancho de banda se le asigna solamente un pedazo de tiempo, mientras que a uno con alto requerimiento se le asignan varios.

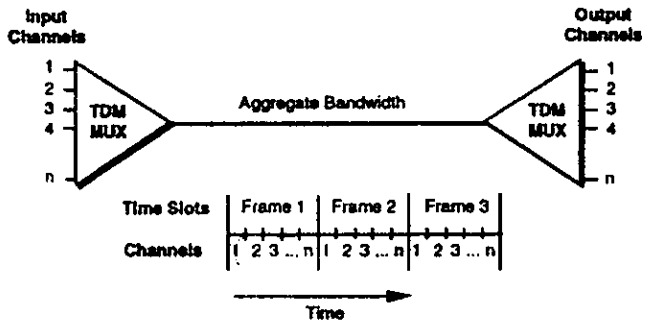


Figura 4.2. Conmutación en el Tiempo.

Dependiendo de la aplicación, cada pedazo de tiempo puede almacenar un bit, un carácter o múltiples caracteres. Los datos que se transmiten sobre una línea de alta capacidad son organizados en estructuras (frames). La estructura define como son organizados los bloques de bits de tal manera que el receptor los pueda entender. Cada estructura contiene un número fijo de pedazos de tiempo secuenciales. Cada canal utiliza los mismos pedazos de tiempo en cada estructura generada. El receptor descompone la estructura, y los datos de cada pedazo de tiempo son depositados en el buffer de salida apropiado.

IV.4 Codificación Bipolar

En las redes digitales, la información se transmite utilizando ésta técnica de codificación especial. La polaridad de una señal indica si es positiva o negativa. Cuando se utiliza codificación bipolar, la polaridad de un 1 binario se alterna entre un voltaje positivo y uno negativo. La polaridad es siempre opuesta a la polaridad que codifica el 1 previo. Un 1 binario es representado mediante un voltaje positivo o negativo durante la primera mitad del intervalo del bit. Un cero binario siempre se representa como un voltaje igual a cero durante todo el intervalo del bit.

La codificación bipolar ayuda a mantener la referencia a cero en la estación receptora. También permite que los pulsos de datos y señales ocurran a intervalos de tiempo iguales. Esto se denomina transmisión isócrona **[16]**, que por lo tanto, no requiere de una señal de reloj, por que cada nodo puede derivar su reloj de recepción, del paquete de bits recibido.

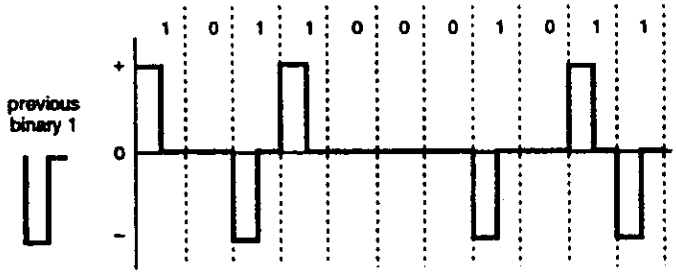


Figura 4.3. Codificación Bipolar

Este tipo de codificación representa un problema cuando se procesa una cadena larga de ceros, que provoca un estado constante de cero voltaje. Lo anterior causa que se pierda el reloj de sincronización y se desconozca en donde termina la posición de un bit e inicia la del siguiente. Lo anterior se corrige insertando un bit diferente de cero, dentro de la cadena de ceros. Para resolver este tipo de problemas existen diferentes técnicas como el *bit stuffing* y el *B8ZS*, que inserta un patrón conocido de violaciones bipolares, cuando se codifica una cadena de 8 ceros.

IV.5 Ventajas de la Transmisión Digital

Desde principios de los sesentas, los proveedores de servicio telefónico han incrementado la implantación de sistemas que utilizan tecnología digital **[16]**. La tecnología digital supera muchos problemas y limitaciones de ancho de banda asociados con los servicios analógicos. Algunos de los factores que intervienen para utilizar tecnología digital son:

- Se presentan bajos niveles de error, comparado con los servicios analógicos.
- Soportar mucho mayores velocidades de transmisión que los servicios analógicos.
- Permitir la mezcla de voz, video y datos sobre el mismo medio de transmisión, lo que significa que diferentes tipos de información pueden compartir los mismos conmutadores y cables .
- Los equipos que implementan tecnología digital son cada vez más baratos que sus contrapartes analógicas.

IV.6 Synchronous Optical NETwork (SONET)

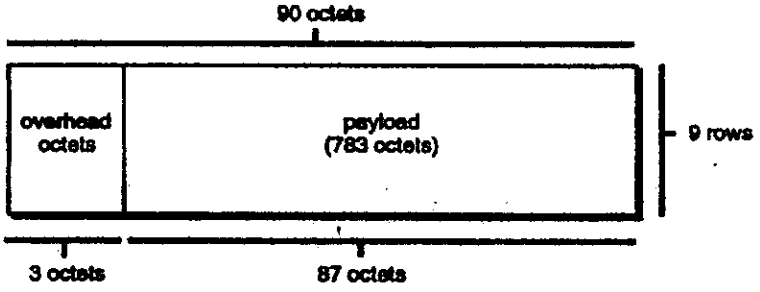
El SONET es un estándar internacional para transmisión y conmutación digital de alta velocidad **[16]**. Este fue originalmente propuesto por Bellcore y estandarizado por ANSI. SONET permite a los proveedores de servicio ofrecer un rango enteramente nuevo de servicios de conmutación y de líneas privadas de alta velocidad. SONET emerge como una nueva tecnología para WAN's; como también lo son, la de *Switched Multimegabit Data Service* (SMDS) y *Broadband ISDN* (B-ISDN).

SONET fue creada para ofrecer estándares de conmutación y para transmisión óptica, para velocidades sobre el nivel de DS-3. La tabla muestra la jerarquía de velocidades de SONET, utilizando el ancho de banda del transmisor óptico (OC).

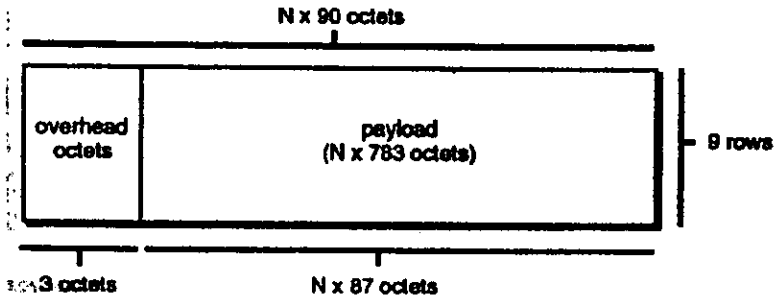
Signal Level	Data Rate	User Payload
STS-1/OC-1	51.840 Mbps	50.112 Mbps
STS-3/OC-3	155.520 Mbps	150.336 Mbps
STS-9/OC-9	466.560 Mbps	451.008 Mbps
STS-12/OC-12	622.080 Mbps	601.344 Mbps
STS-18/OC-18	933.120 Mbps	902.016 Mbps
STS-24/OC-24	1244.160 Mbps	1202.688 Mbps
STS-36/OC-36	1866.240 Mbps	1804.032 Mbps
STS-48/OC-48	2488.320 Mbps	2405.376 Mbps

IV.7 Formato del Frame de SONET

El bloque fundamental de las señales SONET, es la estructura (frame) STS-1 **[16]**. El STS-1 puede verse como una matriz que contiene nueve renglones, con 90 octetos en cada renglón, para un total de 810 octetos. Cada estructura contiene 27 octetos de *overhead* y 783 octetos para el usuario. Una estructura STS-1 es transmitida cada 125 microsegundos (8,000 frames/segundo) ofreciendo una velocidad de transmisión de 51.840 Mbps. (8 bits/octeto x 8,000 frames/segundo), con una capacidad disponible para el usuario de 50.112 Mbps. (8 bits/octeto x 783 octetos/frame x 8,000 frames/segundo). Los octetos en un frame son transmitidos por renglón, de izquierda a derecha y de arriba abajo.



Frames del tipo STS-N son creados con la unión de N frames STS-1. Cada frame STS-N se transmite uno cada 125 microsegundos, ofreciendo una ganancia en transmisión de datos de (N x 51.840) Mbps. La figura muestra un frame STS-N.



Por ejemplo, un frame STS-3 se crea con la unión de 3 frames STS-1. STS-3 ofrece una velocidad de transmisión de 155.520 Mbps (8 bits/octeto x 810 octetos/frame de STS-1 x 3 STS-1 frames/frame STS-3 x 8,000 STS-3 Frames/segundo).

Resumen

De esta manera se establecen las velocidades de transferencia de información a las que funciona esta tecnología. El siguiente paso es representar los dispositivos que forman las redes mediante modelos analíticos, en los que se aplicará la Teoría de Colas, la cual será descrita para fundamentar el desarrollo de los modelos y su aplicación de forma específica en este trabajo.

Capitulo V

V.1 MODELOS DE LOS DISPOSITIVOS

Los modelos definidos mediante líneas de espera (colas), permiten la asignación y modificación de sus parámetros para simular el comportamiento del sistema real, posibilitando la creación de una diversidad de escenarios operacionales, realizar suposiciones acerca del funcionamiento del sistema y ofrecer diferentes condiciones iniciales del sistema **[17],[18]**.

Sin embargo, los modelos desarrollados con esta teoría no han sido aplicados en su verdadero potencial, y distan aún de ofrecer sus ventajas al personal directamente involucrado en el procesamiento de datos, especialmente en la transferencia mediante redes **[20]**.

Ventajas

El modelado de sistemas mediante **Teoría de Colas,** ofrece un conjunto de técnicas para calcular cantidades específicas de rendimiento, como el tiempo de espera de servicio, la eficiencia de un servidor, el efecto de diferentes servidores actuando al mismo tiempo y la comparación de diferentes políticas de almacenamiento temporal (encolamiento) **[18]**.

Suposiciones

Lo que se debe suponer, para recibir las ventajas de las Técnicas de Análisis en Teoría de Colas, es que el sistema bajo observación pueda ser representado adecuadamente por un Sistema de Colas de Espera **[18],[22]**.

Objetivo

El objetivo de los Modelos de Sistemas de Colas es analizar el **rendimiento** de los Sistemas, considerando específicamente la capacidad de recepción y almacenamiento de clientes (colas) y la de atención (procesamiento), que se les ofrece en cada uno de los dispositivos que forman el sistema **[17],[18]**.

Rendimiento

Por Análisis de Rendimiento en Redes se consideran los siguientes aspectos genéricos **[18]**:

- La utilización del ancho de banda disponible en canales .
- El trabajo efectivo realizado en algún dispositivo.
- El tiempo promedio de espera, por parte de un cliente.
- El número promedio de usuarios en el sistema, en un momento específico.
- El análisis de la disponibilidad de recursos.
- Además de la capacidad de realizar análisis a futuro *(what if?)* de los sistemas, para establecer parámetros de rendimiento, como capacidad y disponibilidad.

V.2 DESARROLLO

El proceso de creación de Modelos Analíticos incluye el mapeo del sistema real dentro de un sistema abstracto relativamente más simple, la solución del sistema simple se dirige a los parámetros de interés definidos inicialmente y posteriormente se genera la extrapolación de resultados hacia el sistema real y se compara su comportamiento **[18],[19],[21]**.

Premisa Básica

La suposición básica detrás del uso de un Modelo de Colas para el análisis de un sistema de red, es que los componentes del sistema pueden ser representados por un conjunto de servidores de red (recursos) y de líneas de espera (colas) **[17],[18]**.

Descripción General del Sistema

Un servidor se define como una entidad que puede afectar, incrementar o hasta detener, el flujo de trabajo a través del sistema **[20]**. Éste puede ser un CPU, un canal de E/S, memoria o puerto de comunicación. Las líneas de espera son solamente el lugar donde los trabajos son encolados en espera de servicio.

Para que funcione un Modelo de Colas, los trabajos (o clientes, o paquetes de mensajes o cualquier otra cosa que necesita ser procesada por un servidor) se deben insertar en la red.

V.3 SERVIDOR

La descripción analítica para el servidor, que se propone en el presente trabajo, se plasma como la función de distribución de probabilidad apropiada que describe la razón a la cual el procesador **produce** y **consume** mensajes, de acuerdo con la Teoría de Colas **[20],[27]**.

El Objetivo es reflejar el poder de procesamiento inherente con el que cuenta el procesador, para establecer sus parámetros de funcionamiento acordes a los rangos externos de comunicación y de la intercomunicación propia de la red **[20]**.

La figura , muestra el modelo de líneas de espera que se propone para la representación del servidor.

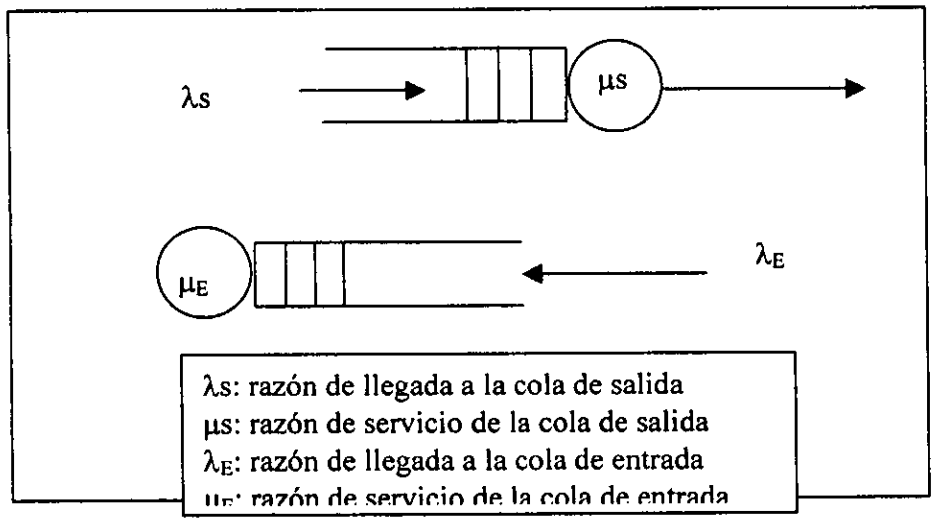


Figura 5.1. Modelo del Servidor

En el ejemplo de la figura , se muestra un servidor con un solo puerto de entrada y uno de salida, que en algunos sistemas reales es el mismo (ej. En UNIX el puerto para Telnet es el 23, utiliza protocolo TCP).

La figura refleja el nivel de generación de paquetes para los que, en términos generales, se cuenta con un espacio definido para realizar su envío, con una razón propia de dicha transferencia y adecuada al nivel de transmisión. De la misma manera se definen dichos parámetros en la recepción de mensajes. Se utilizan dos Sistemas de Colas del tipo M/M/1 para su representación.

Parámetros del Servidor

Se pueden definir las características del tipo de servidor que se desea modelar, en primer lugar, se deben definir las razones de recepción y envío de información (λ y μ). A partir de ellas, es posible determinar las características del sistema **[22]**, como son el tamaño promedio de la cola, el tiempo promedio de espera de un cliente, el total de tiempo que un cliente pasa en el sistema y la utilización del servidor, entre otras **[23]**.

Aplicando las técnicas de la Teoría de Colas, a continuación se describen los parámetros para el servidor **[7],[24],[25],[27]**:

1. Razón de salidas para el envío de paquetes:

$$\lambda_S \text{ y } \mu_S$$

2. Razón de llegadas y procesamiento para la recepción de paquetes:

$$\lambda_E \text{ y } \mu_E$$

3. Utilización promedio del servidor:

$$u = \frac{1/\mu}{1/\lambda} = \frac{\lambda}{\mu}$$

4. Probabilidad de que el servidor este vacío:

$$P_0 = 1 - \frac{\lambda}{\mu}$$

5. Probabilidad de que haya **n** clientes en el servidor:

$$P_n = (1 - \frac{\lambda}{\mu})(\frac{\lambda}{\mu})^n$$

6. El número promedio de clientes en el servidor:

$$E[N] = \frac{\lambda}{\mu - \lambda}$$

7. El tiempo promedio que un cliente espera en la cola de entrada del servidor:

$$E[w_q] = \frac{\frac{\lambda}{\mu^2}}{1 - \frac{\lambda}{\mu}}$$

8. El tiempo esperado total, de permanencia de un paquete en el sistema:

$$E[w] = \frac{1}{\mu - \lambda} \qquad \text{Por Little: } E[w] = \frac{E[N]}{\lambda}$$

9. La longitud promedio de la cola:

$$E[N_q] = \lambda E[w_q]$$

10. La probabilidad de esperar en la cola, menos de un tiempo dado (t):

$$P[w_q \leq t] = 1 - e^{\frac{-t}{E[w]}}$$

Ejemplo SONET

Para el ejemplo de SONET, es necesario fijar la función de generación de paquetes (de 810 octetos cada uno) a una velocidad constante de 125 μs. Con 783 octetos disponibles para el usuario y 27 octetos de *overhead* para control. Se aplica una política de encolamiento FIFO.

Los parámetros de transmisión para representar un servidor SONET de acuerdo al modelo propuesto serían:

$\lambda_S \leq 8{,}000$ paquetes por segundo.
$\mu_S = 8{,}000$ paquetes por segundo.
$\lambda_E = 8{,}000$ paquetes por segundo.
$\mu_E \geq 8{,}000$ paquetes por segundo para evitar perder información.

V.4 LÍNEA DE TRANSMISIÓN

La representación analítica de una línea de transmisión de información, se ofrece como dos líneas de espera, las cuales representan cada una, al medio de transmisión en una dirección de envío de información. Por supuesto, las líneas de transmisión pueden ser unidireccionales, por lo que solamente se utilizaría una de las colas, en la dirección apropiada **[7]**.

Se tomaron en cuenta las siguientes características, para definir el dispositivo:

- ❖ Recibe trabajo (paquetes, mensajes, etc.) que transmite hacia otro lugar de la red.
- ❖ Imprime un retardo definido por la tecnología de servicio.
- ❖ Esta limitado en cuanto a su capacidad de transmisión (tiene ancho de banda definido por la tecnología).

El modelo sería:

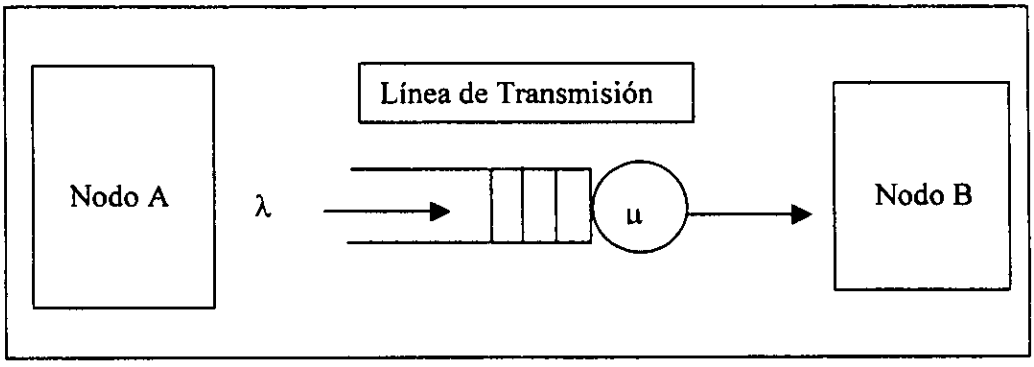


Figura 5.2. Modelo de una línea de transmisión

El parámetro λ marca la capacidad de transmisión del canal y el retardo en la transmisión de información se imprime mediante el parámetro de servicio μ, que indica la velocidad de salida de paquetes por la línea.

Cuando se tiene la condición de que $\lambda > \mu$, el canal se llena, y los paquetes que se reciben posteriormente son rechazados. En estas condiciones se define el sistema como inestable.

Parámetros de la Línea

Mediante los parámetros de ingreso y salida de la línea, es posible moldearla para simular diferentes tecnologías, velocidades y capacidad **[7],[24],[25],[27]**.

Aplicando las técnicas de la Teoría de Colas, se describen los parámetros para la línea de transmisión:

1. Razón de llegadas y transmisión de paquetes:

   λ y μ

2. El número promedio de paquetes en la línea:

$$E[N] = \frac{\lambda}{\mu - \lambda}$$

3. El tiempo esperado total, de un paquete en la línea (latencia):

$$E[w] = \frac{1}{\mu - \lambda} \qquad \text{Por Little: } E[w] = \frac{E[N]}{\lambda}$$

4. El tiempo promedio que un paquete espera en la cola de entrada a la línea:

$$E[w_q] = \frac{\frac{\lambda}{\mu^2}}{1 - \frac{\lambda}{\mu}}$$

5. El número promedio de paquetes en la línea:

$$E[N_q] = \lambda E[w_q]$$

Ejemplo SONET

Para el ejemplo de SONET, se definen los parámetros de recepción y transmisión de mensajes de la siguiente manera, la velocidad de recepción de paquetes es de 8,000 pps., de 810 octetos cada uno, aplicándose una política de encolamiento FIFO

Las λ se fijan con una velocidad de 8,000 pps., por restricciones de transmisión. Las μ se proponen con el retardo establecido para el sistema. En este ejemplo, se garantiza el servicio de velocidad de transmisión también a 8,000 pps., servidos.

V.5 CONMUTADOR

El modelo para representar un conmutador *(multiplexor)* de interconexión, se define como una entidad de recepción-transmisión de información, con una capacidad limitada de espacio de almacenamiento **[7],[24],[25],[27],[28]**. Las características principales de dicho elemento son:

- Opera a nivel de capa 2 de OSI.
- Trabaja entre una estación de un segmento y un servidor o estación de otro segmento.
- Establece un circuito directo temporal.
- El retardo que añade a los paquetes es mínimo, dado que no aplica esquemas complicados de decisión para la transferencia de información (latencia mínima).
- Su esquema de reenvío esta implementado en firmware, con lógica electrónica.
- No cuenta con espacio de almacenamiento temporal *(buffers)*.
- Su funcionamiento depende del tipo de tecnología que se implemente en la red. Se considera un elemento simple.

El modelo sería:

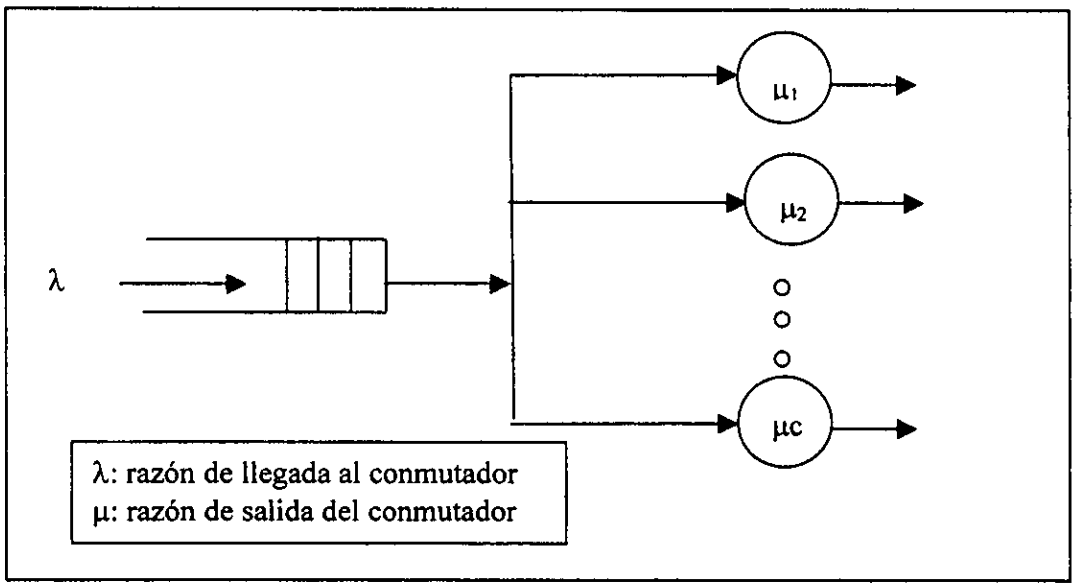


Figura 5.3. Modelo de un conmutador

Por lo tanto, se propone emplear un modelo de colas del tipo M/M/C, con retardo mínimo. El modelo cuenta con una línea de espera y varios servidores para procesar los mensajes, redireccionándolos hacia los diferentes puertos de salida.

Parámetros del Conmutador

Se cuenta, por lo tanto, con un modelo de nacimiento-muerte **[7],[4]** que tiene una razón de entrada constante λ y de servicio dependiente del estado del sistema:

1) Razones de servicio

$$\mu_n = n\mu, \quad n=0,1,2,...,c$$
$$= c\mu, \quad n=c+1,c+2,...$$

2) El número esperado de servidores del conmutador ocupados:

$$E(B) = c\rho$$

2) El número esperado de servidores del conmutador libres:

$$E(I) = c(1-\rho)$$

3) El tamaño esperado de la cola de entrada al conmutador:

$$E(Q) = \frac{\rho}{(1-\rho)} * Pr(N \geq c)$$

4) El número promedio de clientes (paquetes) en el conmutador:

$$E(N) = c\rho + \frac{\rho C}{1-\rho}, \qquad \text{donde } C = Pr\{N \geq c\}$$

5) El tiempo esperado de estar en la cola de entrada del conmutador:

$$E(W_Q) = \frac{1}{c\mu(1-\rho)} \; Pr(N \geq c)$$

6) El tiempo de espera total en el conmutador:

$$E(W) = \frac{1}{\mu} + \frac{P_c}{c\mu(1-\rho)^2}$$

Ejemplo SONET

Para SONET, se define la velocidad de recepción de paquetes constante λ = 8,000. Las de salida quedan de la siguiente manera:

$\mu_n = n * 8000$ n = 0,1,2,...,c

$= c * 8000$ n = c+1, c+2,...

V.6 ENRUTADOR

El enrutamiento envuelve dos actividades básicas: la determinación de los caminos de enrutamiento y la transportación de paquetes de información a través de una interconexión de redes **[7],[4],[24],[25],[27],[28]**.

Su aplicación se realiza tanto en redes locales como en redes de área amplia y cuando existen más de una ruta entre los puntos finales de una red además, proporciona control de tráfico y filtrado de paquetes **[25]**.

Los enrutadores operan a nivel de la capa de red del modelo OSI **[9],[27]**. Hacen decisiones más complejas que las realizadas por un puente o un conmutador. Para realizar las decisiones adecuadas, necesitan contar con mayor información. Algunos datos que considera son el costo de transmitir un paquete sobre ciertas rutas particulares, la longitud que recorre, etc. Esta información es contenida en una tabla conocida como de enrutamiento. La tabla de enrutamiento incluye información sobre la ruta o rutas que cualquier paquete puede tomar, a través de la red para llegar a su destino.

Las funciones principales del enrutador son:

- Crear y mantener la tabla de enrutamiento.
- Comprobar que el paquete está libre de errores, mediante el código de paridad contenido en el mismo.
- Seleccionar el próximo salto del viaje para cada uno de los paquetes, utilizando la información contenida en el paquete y en la tabla de enrutamiento.

Aquí, se revisa el tiempo de latencia o retardo de enrutamiento por medio de la cantidad de tiempo requerido para mover el paquete desde su origen hasta su destino, a través de la red **[25],[27]**. El retraso depende de varios factores, del ancho de banda de los enlaces intermedios de encolamiento en los puertos de cada enrutador a lo largo del camino, del congestionamiento de la red sobre los enlaces intermedios y de la distancia física a recorrer.

Se propone utilizar el siguiente modelo para describir el funcionamiento general del enrutador:

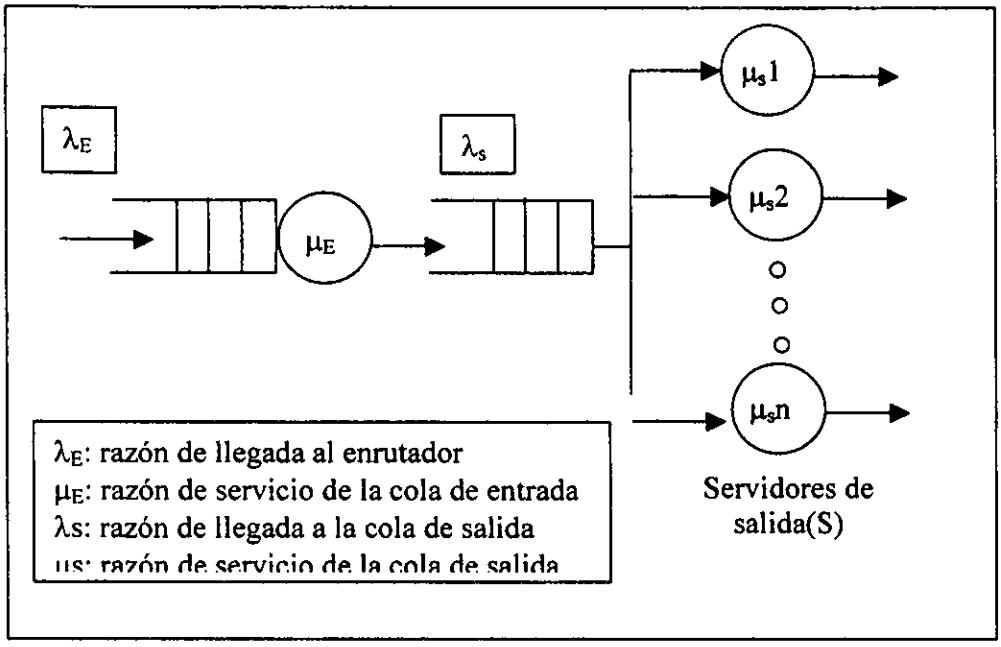


Figura 5.4. Modelo de un enrutador

Se considera que todos los paquetes se reciben en una sola línea y se distribuyen de acuerdo a su destino propio, hacia alguno de los servidores de salida (S). Existe la limitante del tamaño de la línea de espera a la entrada, ya que es necesario que el procesador decida hacia a donde va dirigido el paquete. Las salidas se encuentran restringidas a la dirección de destino del paquete (una vez resuelta ésta), por lo que se sugiere que exista una línea de espera intermedia para evitar perder los paquetes para los que su procesador de salida se encuentra ocupado.

Por lo tanto, se aplican dos modelos, uno del tipo de sistema M/M/1 **[4],[7]** para el servidor de decisión, que representa la primera parte del funcionamiento del dispositivo y se encarga de resolver la dirección siguiente de cada paquete. El modelo M/M/1 incluye una cierta capacidad de encolamiento, que se muestra restringiendo la entrada de paquetes mediante un almacén finito de información.

El modelo para la segunda parte, será del tipo M/M/C **[4],[7]**, que como en el dispositivo anterior, permite distribuir la razón de ingreso entre los diferentes servidores de salida (S). En éste modelo se cuenta con la capacidad para almacenar temporalmente los paquetes a los que se les ha resuelto la dirección, a la velocidad del procesador (servidor) de la primera parte del modelo. Por lo tanto, las estadísticas y parámetros del dispositivo serán asignadas por separado de acuerdo a las razones de ingreso definidas.

Parámetros del Enrutador

a) Primera parte

Aplicando las técnicas de la Teoría de Colas, se describen los parámetros para la parte de recepción de paquetes y procesamiento, que es la correspondiente a un sistema del tipo M/M/1 **[4],[7]**, cuyos parámetros son:

1. Razón de llegadas y transmisión de paquetes:

   λ_E y μ_E

2. El número promedio de paquetes en línea de entrada:

$$E[N] = \frac{\lambda_E}{\mu_E - \lambda_E}$$

3. El tiempo esperado total, de un paquete en la línea (latencia) de entrada:

$$E[w] = \frac{1}{\mu_E - \lambda_E} \qquad \text{Por Little: } E[w] = \frac{E[N]}{\lambda_E}$$

4. El tiempo promedio que un paquete espera en la cola de entrada:

$$E[w_q] = \frac{\frac{\lambda_E}{\mu_E^2}}{1 - \frac{\lambda_E}{\mu_E}}$$

5. El número promedio de paquetes en la línea de entrada:

$$E[N_q] = \lambda_E E[w_q]$$

b) Segunda Parte

La segunda parte, cuenta con un modelo de nacimiento-muerte del tipo M/M/C **[4],[7]**, cuya razón de entrada es constante λ_n y la de servicio, depende del estado del sistema:

1) Razones de servicio

$$\mu_n = n\mu, \quad n=0,1,2,...,c$$
$$= c\mu, \quad n=c+1,c+2,...$$

2) El número esperado de servidores ocupados del enrutador:

$$E(B) = c\rho$$

2) El número esperado de servidores libres del enrutador:

$$E(I) = c(1-\rho)$$

3) El tamaño esperado de la cola:

$$E(Q) = \frac{\rho}{(1-\rho)} * Pr(N \geq c)$$

4) El número promedio de clientes en el enrutador:

$$E(N) = c\rho + \frac{\rho C}{1-\rho}, \qquad \text{donde } C = Pr\{N \geq c\}$$

5) El tiempo esperado de estar en la cola:

$$E(W_Q) = \frac{1}{c\mu(1-\rho)} \; Pr(N \geq c)$$

6) El tiempo de espera total en el enrutador:

$$E(W) = \frac{1}{\mu} + \frac{p_c}{c\mu(1-\rho)^2}$$

V.7 PUENTE

Los puentes son dispositivos que pueden interconectar redes similares o distintas. Su función es unir dos redes separadas (segmentos de red) para formar una sola red lógica **[27],[28]**. Las redes originales llegan a ser segmentos de red que dan como resultado una red más compleja, la cual funciona como si fuese un solo cable o un dominio de envío *(broadcast)*.

Los puentes trabajan en el nivel de capa de enlace de datos (capa 2 de OSI) **[2]**. Por lo que solo pueden ver las direcciones MAC de los paquetes. De tal manera que aíslan el acceso al medio en los segmentos que el puente conecta. Con un puente, solo los paquetes *(frames)* que tienen la dirección destino a un servidor sobre un segmento diferente serán redireccionados o retransmitidos a través del puente (trafico remoto).

Se tomaron en cuenta las siguientes características, para definir el dispositivo:

- Recibe trabajo (paquetes, mensajes, etc.) que transmite hacia otro segmento de la red, si es que el destino se encuentra fuera del segmento dado.
- Imprime un retardo de servicio, definido por la tecnología.
- Esta limitado en cuanto a su capacidad de transmisión o retransmisión.

La figura , representa el modelo completo.

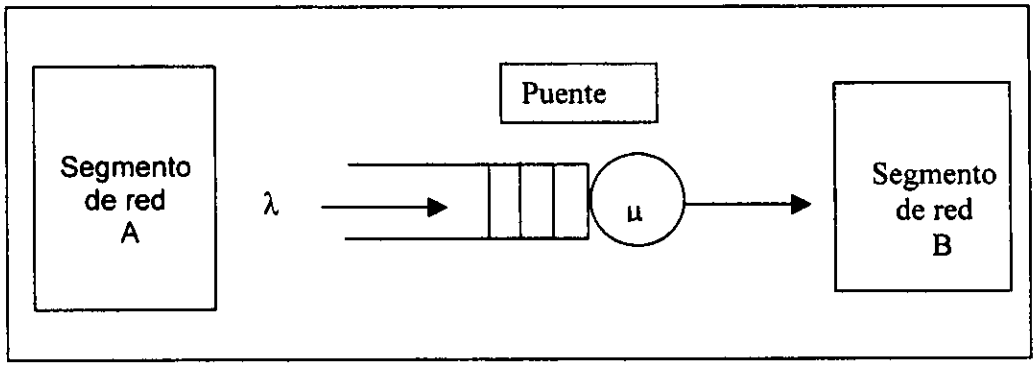


Figura 5.5. Modelo de un Puente

El parámetro λ marca la capacidad de transmisión del canal **[4],[7]** y el retardo en la transmisión de información se imprime mediante el parámetro de servicio μ, que indica la velocidad de salida de paquetes por el puente.

Cuando se tiene la condición de que $\lambda > \mu$, el canal de ingreso al puente se llena, y los paquetes que se reciben posteriormente son rechazados. En estas condiciones, se define el sistema como inestable.

Parámetros de el Puente

Mediante los parámetros de ingreso y salida del puente, es posible moldearla para simular diferentes tecnologías, velocidades y capacidad **[4],[5],[7],[20],[27],[28]**, para el mismo.

Aplicando las técnicas de la Teoría de Colas, se describen los parámetros para el puente de la siguiente manera:

1. Razón de llegadas y transmisión de paquetes:
   λ y μ
2. El número promedio de paquetes en el puente:

$$E[N] = \frac{\lambda}{\mu - \lambda}$$

3. El tiempo esperado total, de un paquete en el puente (latencia):

$$E[w] = \frac{1}{\mu - \lambda} \qquad \text{Por Little: } E[w] = \frac{E[N]}{\lambda}$$

4. El tiempo promedio que un paquete espera en la cola del puente:

$$E[w_q] = \frac{\frac{\lambda}{\mu^2}}{1 - \frac{\lambda}{\mu}}$$

5. El número promedio de paquetes, en el puente:

$$E[N_q] = \lambda E[w_q]$$

Ejemplo SONET

Para el ejemplo de SONET, se definen los parámetros de recepción y transmisión de mensajes de la siguiente manera, la velocidad de recepción de paquetes es de 8,000 pps., de 810 octetos cada uno, aplicándose una política de encolamiento FIFO.

Las λ se fijan con una velocidad de 8,000 pps., por restricciones de transmisión. Las μ se proponen con el retardo establecido para el sistema. En este ejemplo, se garantiza el servicio de velocidad de transmisión también a 8,000 pps., servidos.

Resumen

Una vez descritos mediante la Teoría de Colas, los dispositivos de redes seleccionados en el presente trabajo, el siguiente paso es plasmar en términos computacionales estos sistemas. Por lo que en el siguiente capítulo se ofrece el conjunto de dispositivos genéricos "programados", para ser aplicados en la representación de sistemas de redes y obtener los parámetros de rendimiento esperados.

Capítulo VI

VI.1 LIBRERÍA

Los modelos para los dispositivos, descritos anteriormente, son el conjunto de elementos abstractos que forman la librería que se propone como producto final de la presente investigación. Cada uno de ellos, describe el comportamiento específico del dispositivo al que representa, reflejando sus principales propiedades y características individuales, de acuerdo al nivel de abstracción definido en un principio.

El conjunto es denominado **"Librería"** dado que los elementos que modela tienen características similares, como es la de pertenecer a una misma área de aplicación (en este caso las redes de computadoras), y principalmente por que en conjunto permiten la descripción de este tipo de sistemas, con sus funciones y metodologías específicas **[29],[30],[31]**.

Los modelos son genéricos y pretenden describir lo mejor posible mediante abstracciones y suposiciones los fenómenos presentes en los sistemas reales.

VI.2 OBJETOS Y CLASES

En la Teoría de Orientación a Objetos, se analizan sistemas mediante su descomposición, obteniéndose abstracciones teóricas de sus partes, que trasladadas a un ambiente propio para la existencia de elementos abstractos (computadora), permite relacionarlos para formar abstracciones del sistema original para su análisis teórico.

La metodología de OO (Orientación a Objetos) es particionar el sistema en pequeños elementos únicos, cada uno representante de una parte específica del sistema. Cada parte es definida de tal manera que es independiente de la demás, con sus propios datos y las funciones que actúan sobre ellos, creándose por lo tanto, un tipo de objeto abstracto (**clase**) de cada una, que al interactuar con las demás, permiten formar la abstracción representativa del sistema original. Otra ventaja de estos tipos de **objetos** o **clases**, es que mediante ellos es posible describir un amplio rango de sistemas similares al original **[30],[31]**. La **clase** de un tipo de **objetos** es el molde de donde se obtienen objetos similares (lo que permite la reutilización de código). Las librerías pueden ser de objetos listos para ser utilizados o de clases para reutilizar código.

Las librerías de objetos son, en términos generales de computación, abstracciones referidas a una misma área de aplicación. En términos de programación de computadoras, las librerías son código preconcebido para ayudar a la creación de sistemas de una manera eficiente y práctica. Existe gran variedad de librerías de objetos para computación, que trabajan en el ambiente adecuado para resolver una diversidad de problemas como son los relacionados con la creación de gráficos, la animación, el sonido, la interacción con bases de datos, intercomunicación entre usuarios, procesamiento de información, control, entre muchos otros.

Algunas ventajas de utilizar librerías de objetos son **[7],[4]**:

- La disminución del tiempo de desarrollo de sistemas.
- Seguridad al utilizar código preevaluado.
- Eficiencia al reutilizar código.
- Contar con puntos de partida claramente definidos para crear sistemas nuevos.
- Localización eficiente de errores.
- Ampliar las capacidades de los elementos mediante mecanismos preestablecidos.
- Gran variedad y generalidad de temas y objetivos.
- Existencia en diferentes lenguajes y plataformas.

Las principales desventajas son:

- Su adquisición (costo).
- Para los creadores, el desarrollo original.
- Algunas cuentan con elementos definidos que no se pueden modificar o es difícil.

VI.3 LIBRERÍA PROPUESTA

Los modelos analíticos obtenidos para representar los dispositivos de interconexión en las redes, traducidos al lenguaje para Programación Orientada a Objetos C++ **[7],[4]**, forman el conjunto de **clases** que se ofrecen como **librería** para la creación de objetos que representarán los dispositivos de interconexión, producto final de ésta investigación.

Dada la generalidad de los modelos de la biblioteca, es posible manipularla para modelar diferentes tipos de dispositivos, configuraciones y capacidad del sistema de red.

Como se definió anteriormente, se modelaron los dispositivos más utilizados en sistemas de redes, siendo estos tanto entidades productoras o consumidoras de información (DTE, Data Terminal Equipment), como dispositivos que intervienen en el enlace, control y transferencia de información. Los elementos que se modelaron y que forman la librería, son los siguientes:

1. **Servidor**: el cuál modela una fuente o depósito de información.
2. **Línea**: que modela una línea de transmisión adaptable para diferentes capacidades.
3. **Puente**: que modela el dispositivo de enlace entre diferentes segmentos de red.
4. **Conmutador**: que cumple la función de combinar información de diferentes fuentes en una sola línea de transmisión.
5. **Enrutador**: que se encarga de analizar y definir las direcciones de envío de información a través de la red.

VI.4 CÓDIGO DE LAS CLASES

En POO, se definen las **clases** como los moldes que permiten la creación de los objetos de acuerdo a la definición establecida para cada una de ellas **[7],[4]**. A continuación se muestra el código de cada una de dichas clases.

Propuesta

Librería Genérica de clases de Dispositivos para Intercomunicación en Redes

```
#ifndef SERVI_H
#define SERVI_H 1

//
// Clase que permite la creación de objetos tipo servidor genérico
// Desarrollada por: Ing. Raymundo Suarez Martinez
//

#include <stdio.h>
#include <math.h>

class SERVIDOR                                          // clase servidor genérico
{
 private:
        float lamda_ent, lamda_sal, mu_ent, mu_sal;     // parámetros del servidor
                                                        // (bidireccional) para una entrada
                                                        // y una salida.
        float uso;                                      // utilización del servidor (puede ser de entrada
                                                        // o salida, dependiendo de
                                                        // los parámetros que se utilicen).

        float A_la_n(int n=0, float esto=0.0)           // función que eleva un
        {                                               // valor (esto) a la n
                float aux=1.0;
                int i;
                for (i=1; i<= n; i++) aux=aux*esto;
                return aux;
        }

 public:
        SERVIDOR(float l_ent=0.0, float m_ent=0.0, float l_sal=0.0,  float m_sal=0.0)
                {
                  printf ("  Servidor Bidireccional \n");
                  lamda_ent = l_ent;
                  printf (" l (ent) = %2.2f (cli/seg) \n", lamda_ent);
                  mu_ent = m_ent;
                  printf (" mu (ent) = %2.2f (cli/seg) \n", mu_ent);
                  lamda_sal = l_sal;
                  printf (" l (sal) = %2.2f (cli/seg) \n", lamda_sal);
                  mu_sal = m_sal;
                  printf (" mu (sal) = %2.2f (cli/seg) \n", mu_sal);
                }
                                                        // Constructor de un servidor
                                                        // Unidireccional si solo se dan dos parámetros
                                                        // Bidireccional si se dan los cuatro
```

```
float Utilizacion(float lamda, float mu) // utilización del servidor
{                                        // (puede ser bidireccional,
        uso = 0;                         // dependiendo de los parámetros
        if (mu != 0) uso=lamda/mu;       // que se le pasen -lamda y mu-).
        return (uso*100);
}

float Prob_Sistema_Vacio(float lamda=1.0, float mu=1.0)
{
        float p=0.0;
        if (mu != 0) p=1.0-(lamda/mu);
        return fabs(p*100);
}
                                         // Probabilidad de no tener clientes en
                                         // el sistema

float Prob_N_Clientes(int n=0, float lamda=1.0, float mu=1.0)
        {
        float p=0.0;
        if (mu != 0) p=(1.0-lamda/mu)*A_la_n(n,(lamda/mu));
        return fabs(p*100);
}
                                         // Probabilidad de que se encuentren
                                         // exactamente n clientes en el sistema

float Numero_De_Clientes(float lamda=1.0, float mu=0.0)
{
        float p=0.0;                     // El número esperado en promedio,
        if (lamda-mu)                    // de clientes activos en el sistema
          p=lamda/(lamda-mu);
        return fabs(p);
}

float Tiempo_Espera_Cola(float lamda=1.0, float mu=0.0)
        {
        float p=0.0;                     // Tiempo en promedio, que un cliente
        if (mu != 0)                     // espera en la cola, para recibir servicio
          p=(lamda/(lamda-A_la_n(2, mu)))/(1.0-(lamda/mu));
        return fabs(p*60/100);
}

float Tiempo_Total(float lamda=1.0, float mu=0.0)
{
        float p=0.0;                     // Tiempo total en promedio, que un cliente
        if (lamda-mu)                    // pasa dentro del sistema, desde que llega
          p=1.0/(lamda-mu);              // a la cola hasta que sale después de ser
        return fabs(p*60/100);           // atendido.
}
```

```
        float Longitud_Cola(float lamda=1.0, float mu=0.0)
        {
                float p=0.0;
                if (mu != 0) p=lamda*Tiempo_Espera_Cola(lamda, mu);
                return fabs(p);
        }
                                                        // Número de clientes en promedio,
                                                        // que se espera que tenga en promedio
                                                        // la cola.

        double Tiempo_Espera(double t=0.0, float lamda=1.0, float mu=0.0)
        {
                double p=0.0;
                if (mu != 0) p=1.0-exp(-t*Tiempo_Total(lamda, mu));
                return fabs(p*100);
        }
                                                        // Probabilidad de esperar en la cola
                                                        // menos de un tiempo (t), dado.
}; // SERVIDOR

#endif
```

```
#ifndef PUENTE_H
#define PUENTE_H 1

//
// Clase que permite la creación de objetos tipo puente genérico
// Desarrollada por: Ing. Raymundo Suarez Martinez
//


#include <stdio.h>
#include <math.h>

class PUENTE                                          // clase puente genérico
{
 private:
        float lamda_ent, lamda_sal, mu_ent, mu_sal;   // parámetros del puente
                                                      // (bidireccional) para una entrada
                                                      // y una salida.
        float uso;                                    // utilización del puente (puede ser de entrada
                                                      // o salida, dependiendo de
                                                      // los parámetros que se utilicen).

        float A_la_n(int n=0, float esto=0.0)         // función que eleva un
  {                                                   // valor (esto) a la n
                float aux=1.0;
                int i;
   for (i=1; i<= n; i++) aux=aux*esto;
   return aux;
  }

 public:
        PUENTE(float l_ent=0.0, float m_ent=0.0, float l_sal=0.0,  float m_sal=0.0)
                {
                 printf ("  Puente Bidireccional \n");
                 lamda_ent = l_ent;
                 printf (" l (ent) = %2.2f (cli/seg) \n", lamda_ent);
                 mu_ent = m_ent;
                 printf (" mu (ent) = %2.2f (cli/seg) \n", mu_ent);
                 lamda_sal = l_sal;
                 printf (" l (sal) = %2.2f (cli/seg) \n", lamda_sal);
                 mu_sal = m_sal;
                 printf (" mu (sal) = %2.2f (cli/seg) \n", mu_sal);
                }
                                                      // Constructor de un puente
                                                      // Unidireccional si solo se dan dos parámetros
                                                      // Bidireccional si se dan los cuatro
```

```
float Utilizacion(float lamda, float mu)                    // utilización del puente
{                                                           // (puede ser bidireccional,
        uso = 0;                                            // dependiendo de los parámetros
        if (mu != 0) uso=lamda/mu;                          // que se le pasen -lamda y mu-).
        return (uso*100);
}

float Prob_Sistema_Vacio(float lamda=1.0, float mu=1.0)
{
        float p=0.0;
        if (mu != 0) p=1.0-(lamda/mu);
        return fabs(p*100);
}                                                           // Probabilidad de no tener clientes en
                                                            // el sistema

float Prob_N_Clientes(int n=0, float lamda=1.0, float mu=1.0)
        {
        float p=0.0;
        if (mu != 0) p=(1.0-lamda/mu)*A_la_n(n,(lamda/mu));
        return fabs(p*100);
}                                                           // Probabilidad de que se encuentren
                                                            // exactamente n clientes en el sistema

float Numero_De_Clientes(float lamda=1.0, float mu=0.0)
{
        float p=0.0;                                        // El número esperado en promedio,
        if (lamda-mu)                                       // de clientes activos en el sistema
          p=lamda/(lamda-mu);
        return fabs(p);
}

float Tiempo_Espera_Cola(float lamda=1.0, float mu=0.0)
        {
        float p=0.0;                                        // Tiempo en promedio, que un cliente
        if (mu != 0)                                        // espera en la cola, para recibir servicio
          p=(lamda/(lamda-A_la_n(2, mu)))/(1.0-(lamda/mu));
        return fabs(p*60/100);
}

float Tiempo_Total(float lamda=1.0, float mu=0.0)
{
        float p=0.0;                                        // Tiempo total en promedio, que un cliente
        if (lamda-mu)                                       // pasa dentro del sistema, desde que llega
          p=1.0/(lamda-mu);                                 // a la cola hasta que sale después de ser
        return fabs(p*60/100);                              // atendido.
}
```

```
        float Longitud_Cola(float lamda=1.0, float mu=0.0)
        {
                float p=0.0;
                if (mu != 0) p=lamda*Tiempo_Espera_Cola(lamda, mu);
                return fabs(p);
        }
                                                        // Número de clientes en promedio,
                                                        // que se espera que tenga en promedio
                                                        // la cola.

        double Tiempo_Espera(double t=0.0, float lamda=1.0, float mu=0.0)
        {
                double p=0.0;
                if (mu != 0) p=1.0-exp(-t*Tiempo_Total(lamda, mu));
                return fabs(p*100);
        }
                                                        // Probabilidad de esperar en la cola
                                                        // menos de un tiempo (t), dado.
}; // PUENTE

#endif
```

```
#ifndef LINEA_H
#define LINEA_H 1

//
// Clase que permite la creación de objetos tipo línea genérica
// Desarrollada por: Ing. Raymundo Suarez Martinez
//

#include <stdio.h>
#include <math.h>

class LINEA                                          // clase línea genérica
{
 private:
        float lamda_ent, lamda_sal, mu_ent, mu_sal;  // parámetros de la línea
                                                     // (bidireccional) para una entrada
                                                     // y una salida.

        float uso;                                   // utilización de la línea (puede ser de entrada
                                                     // o salida, dependiendo de
                                                     // los parámetros que se utilicen).

        float A_la_n(int n=0, float esto=0.0)        // función que eleva un
            {                                        // valor (esto) a la n
              float aux=1.0;
              int i;
                    for (i=1; i<= n; i++) aux=aux*esto;
              return aux;
             }

 public:
        LINEA(float l_ent=0.0, float m_ent=0.0, float l_sal=0.0,  float m_sal=0.0)
                {
                  printf ("  Linea Bidireccional ");
                  lamda_ent = l_ent;
                  printf (" l (ent) = %f \n", lamda_ent);
                  mu_ent = m_ent;
                  printf (" mu (ent) = %f \n", mu_ent);
                  lamda_sal = l_sal;
                  printf (" l (sal) = %f \n", lamda_sal);
                  mu_sal = m_sal;
                  printf (" mu (sal) = %f \n", mu_sal);
                }
                                                     // Constructor de una línea
                                                     // Unidireccional si solo se dan dos parámetros
                                                     // Bidireccional si se dan los cuatro
```

```
float Utilizacion(float lamda=1, float mu=1) // utilización de la linea
{                                                    // (puede ser bidireccional,
        uso = 0;                                     // dependiendo de los parámetros
        if (mu != 0) uso=lamda/mu;                   // que se le pasen -lamda y mu-).
        return uso;
}


float Prob_Sistema_Vacio(float lamda=1.0, float mu=1.0)
{
        float p=0.0;
        if (mu != 0) p=1.0-(lamda/mu);
        return fabs(p);
}
                                                     // Probabilidad de no tener clientes en
                                                     // el sistema

float Prob_N_Clientes(int n=0, float lamda=1.0, float mu=1.0)
{
        float p=0.0;
        if (mu != 0) p=(1.0-lamda/mu)*A_la_n(n,(lamda/mu));
        return fabs(p);
}
                                                     // Probabilidad de que se encuentren
                                                     // exactamente n clientes en el sistema

float Numero_De_Clientes(float lamda=1.0, float mu=0.0)
{
        float p=0.0;                                 // El número esperado en promedio,
        if (lamda-mu)                                // de clientes activos en el sistema
          p=lamda/(lamda-mu);
        return fabs(p);
}

float Tiempo_Espera_Cola(float lamda=1.0, float mu=0.0)
{
        float p=0.0;                                 // Tiempo en promedio, que un cliente
        if (mu != 0)                                 // espera en la cola, para recibir servicio
          p=(lamda/(lamda-A_la_n(2, mu)))/(1.0-(lamda/mu));
        return fabs(p);
}

float Tiempo_Total(float lamda=1.0, float mu=0.0)
{
        float p=0.0;                                 // Tiempo total en promedio, que un cliente
        if (lamda-mu)                                // pasa dentro del sistema, desde que llega
          p=1.0/(lamda-mu);                          // a la cola hasta que sale después de ser
        return fabs(p);                              // atendido.
}
```

```
        float Longitud_Cola(float lamda=1.0, float mu=0.0)
        {
                float p=0.0;
                if (mu != 0) p=lamda*Tiempo_Espera_Cola(lamda, mu);
                return fabs(p);
        }
                                                        // Numero de clientes en promedio,
                                                        // que se espera que tenga en promedio
                                                        // la cola.

        double Tiempo_Espera(double t=0.0, float lamda=1.0, float mu=0.0)
        {
                double p=0.0;
                if (mu != 0) p=1.0-exp(-t*Tiempo_Total(lamda, mu));
                return fabs(p);
        }
                                                        // Probabilidad de esperar en la cola
                                                        // menos de un tiempo (t), dado.
};// LINEA

#endif
```

```
#ifndef ENRUTA_H
#define ENRUTA_H 1

//
// Clase que permite la creación de objetos tipo enrutador genérico
// Desarrollada por: Ing. Raymundo Suarez Martinez
//

#include <stdio.h>
#include <math.h>

class ENRUTADOR                                  // clase enrutador genérico
{
 private:
        float lamda_sal, mu_sal, lamda_ent, mu_ent;   // parámetros del enrutador
        int c;                                   // numero de procesadores en el enrutador
        float servicio;                          // razón de servicio del enrutador
        float uso;                               // utilización de los servidores
        float rho;                               // razón de uso del enrutador

   float A_la_n(int n=0, float esto=0.0)         // función que eleva un valor
        {                                        // (esto) a la n
    float aux=1.0;
    int i;
    for (i=1; i<= n; i++) aux=aux*esto;
    return aux;
   }

   float Fact(int n)                             // función para calcular el factorial de n
   {
    int i;
    float aux=1;
    for (i=1; i<=n; i++) aux= aux*i;
    return aux;
   }

   float PCero(void)                             // func. para calcular la probabilidad
{                                                // cero (P0) del sistema
    float paux=0.0;
    int i;
    if (rho<1)
    {
     for (i=0; i<c ;i++)
       paux= paux+(A_la_n(i,(lamda_sal/mu_sal))/Fact(i));
     paux = paux + A_la_n(c,(lamda_sal/mu_sal))/(Fact(c)*(1-rho));
                if (paux != 0) return (1/paux);
     else return 0;
    }
    return 0;
   }
```

```
float Prob_NmayorC(void)                    // Función para calcular la probabilidad de tener un
{                                           // numero mayor de clientes en el sistema,
        float aux, aux2;                    //  que el numero de servidores
        aux2=lamda_ent/mu_sal;
        aux=(A_la_n(c,aux2)/(Fact(c)*(1-rho)))*PCero();
        return aux;
}

public:
        ENRUTADOR(float l_ent=0.0, float m_ent=0.0, float l_sal=0.0,
                                float m_sal=0.0, int serv=1)
                {
                 lamda_ent = l_ent;
                 printf (" l de entrada = %2.1f (cli/seg)\n", lamda_ent);
                 mu_ent = m_ent;
                 printf (" mu de entrada = %2.1f (cli/seg)\n", mu_ent);
                 lamda_sal = l_sal;
                 printf (" l de salida = %2.1f (cli/seg)\n", lamda_sal);
                 mu_sal = m_sal;
                 printf (" mu de salida = %2.1f (cli/seg)\n", mu_sal);
                 c=serv;
                 printf (" servidores de salida = %2d \n", c);
                 rho=lamda_sal/(mu_sal*c);
                }                                                  // Constructor del enrutador

        // primera parte

        float Utilizacion1(void)                                   // utilización del enrutador
        {                                                          // (entrada)
                uso = 0;
                if (mu_ent != 0) uso=lamda_ent/mu_ent;
                return (uso*100);
        }


        float Prob_Sistema_Vacio1(void)                            // Probabilidad de no tener clientes en
                {                                                  // la entrada del enrutador
                 float p=0.0;
                 if (mu_ent != 0) p=1.0-(lamda_ent/mu_ent);
                 return fabs(p*100);
                }

        float Numero_De_Clientes1(void)                            // El número esperado en promedio,
        {                                                          // de clientes activos en la entrada
                float p=0.0;                                       // del enrutador
                if (lamda_ent-mu_ent)
                  p=lamda_ent/(lamda_ent-mu_ent);
                return fabs(p);
        }
```

```
float Tiempo_Espera_Cola1(void)                    // Tiempo en promedio, que un cliente
{                                                  // espera en la cola, para recibir
        float p=0.0;                               // servicio
        if (lamda_ent-mu_ent)
          p=(lamda_ent/(lamda_ent-A_la_n(2, mu_ent)))/(1.0-(lamda_ent/mu_ent));
        return fabs(p*60/100);
}

float Tiempo_Total1(void)                          // Tiempo total en promedio, que un cliente
{                                                  // pasa dentro del sistema, desde que llega
        float p=0.0;                               // a la cola hasta que sale después de ser
        if (lamda_ent-mu_ent)       // atendido.
          p=1.0/(lamda_ent-mu_ent);
        return fabs(p*60/100);
}

float Longitud_Cola1(void)                         // Número de clientes en promedio,
{ float p=0.0;                                     // que se espera que tenga en promedio
        if (mu_ent != 0) p=lamda_ent*Tiempo_Espera_Cola1();   // la cola.
        return fabs(p*60/100);
}

// segunda parte

float Utilizacion2(int n = 1)                      // razón de servicio del enrutador cuando
{                                                  // trabaja con un número de procesadores n
        servicio=0.0;
        if (n<=c) servicio=lamda_sal/(n*mu_sal);
        else servicio = lamda_sal/(c*mu_sal);
        return (servicio*100);
}

float Servidores_Ocupados2(void) {return (c*rho);}
                                                   // Número esperado de servidores ocupados

float Servidores_Libres2(void) {return (c*(1-rho));}
                                                   // Número esperado de servidores libres


float Numero_De_Clientes2(void)                    // El número esperado en promedio,
{                                                  // de clientes activos en el sistema
        float p=0.0;
        p=c*rho+(rho*Prob_NmayorC())/(1-rho);
        return fabs(p);
}
```

```
        float Tiempo_Espera_Cola2(void)                    // Tiempo en promedio, que un cliente
        {                                                  // espera en la cola, para recibir
                float p=0.0;                               // servicio
                p=(1/(c*mu_sal*(1-rho)))*Prob_NmayorC();
                return fabs(p*60/100);
        }

        float Tiempo_Total2(void)
        {
                float p=0.0;
                p=Numero_De_Clientes2()/lamda_sal;
                return fabs(p*60/100);
        }
                                                           // Tiempo total en promedio, que un cliente
                                                           // pasa dentro del sistema, desde que llega
                                                           // a la cola hasta que sale después de ser
                                                           // atendido.
}; // ENRUTADOR

#endif
```

```
#ifndef CONMUTA_H
#define CONMUTA_H 1

#include <stdio.h>
#include <math.h>

//
// Clase que permite la creación de objetos tipo conmutador genérico
// Desarrollada por: Ing. Raymundo Suarez Martinez
//

class CONMUTADOR {                      // clase conmutador (multiplexor) genérico
 private:
        float lamda_ent, mu_sal;        // parámetros del conmutador
        int c;                          // número de procesadores en el conmutador
        float servicio;                 // razón de servicio del conmutador
        float uso;                      // utilización de los servidores
        float rho;                      // razón de uso del conmutador

        float A_la_n(int n=0, float esto=0.0)       // función que eleva un valor
        {                                           // (esto) a la n
                float aux=1.0;
                int i;
                for (i=1; i<=n; i++)
                  aux=aux*esto;
                return aux;
        }

        float Fact(int n = 1)                       // función para calcular el factorial de n
        {
                int i;
                float aux=1;
                for (i=1; i<=n; i++) aux= aux*i;
                return aux;
        }

        float PCero(void)                   // func. para calcular la probabilidad cero
        {                                   // (P0) del sistema
                float paux=0.0;
                int i;
                if (rho<1)
                {
                  for (i=0;i<c;i++)
                          paux= paux+(A_la_n(i,(lamda_ent/mu_sal))/Fact(i));
                  paux = paux + A_la_n(c,(lamda_ent/mu_sal))/(Fact(c)*(1-rho));
                  if (paux != 0) return (1/paux);
                  else return 0;
                }
                return 0;
        }
```

```
        float Prob_NmayorC(void)          // Función para calcular la probabilidad de tener
        {                                 // un número mayor de clientes en el sistema,
                float aux, aux2;          // que el número de servidores
                aux2=lamda_ent/mu_sal;
                aux=(A_la_n(c,aux2))/((Fact(c)*(1-rho))*PCero());
                return aux;
        }

public:
        CONMUTADOR(float l_ent=0,  float m_sal=0, int serv=1)
                { lamda_ent = l_ent;
                  printf (" l = %2.2f \n", lamda_ent);
                  mu_sal = m_sal;
                  printf (" mu = %2.2f \n", mu_sal);
                  c=serv;
                  printf (" servidores : %3d \n", c);
                  rho=lamda_ent/(mu_sal*c);
                  printf (" rho = %2.2f \n", rho);
                }

                                          // Constructor del conmutador

        float Utilizacion(int n = 1)              // razón de servicio del enrutador cuando
        {                                 // trabaja con un número de procesadores n
                servicio=0.0;
                if (n<=c) servicio=lamda_ent/(n*mu_sal);
                else servicio = lamda_ent/(c*mu_sal);
                return (servicio*100);
        }

        float Servidores_Ocupados(void) {return (c*rho);}   // Número esperado de
                                                            // servidores ocupados

        float Servidores_Libres(void) {return (c*(1-rho));} // Número esperado de
                                                            // servidores libres

        float Tamano_Cola(void)           // Tamaño esperado de la cola (num clientes en prom)
        {
                if (rho != 1)
                  return ((rho/(1-rho))*Prob_NmayorC());
                else return -1;
        }

        float Numero_De_Clientes(void)            // El número esperado en promedio,
        {                                 // de clientes activos en el sistema
                float p=0.0;
                p=c*rho+(rho*Prob_NmayorC())/(1-rho);
                return fabs(p);
        }
```

```
        float Tiempo_Espera_Cola(void)          // Tiempo en promedio, que un cliente
        {                                       // espera en la cola, para recibir servicio
                float p=-1;
                if (rho != 0) p=(1/(c*mu_sal*(1-rho)))*Prob_NmayorC();
                return fabs(p*60/100);
        }

        float Tiempo_Total(void)                // Tiempo total en promedio, que un cliente
        {                                       // pasa dentro del sistema, desde que llega
                float p=0.0;                    // a la cola hasta que sale después de ser
                p=Numero_De_Clientes()/lamda_ent;       // atendido.
                return fabs(p*60/100);
        }
}; // CONMUTADOR

#endif
```

Resumen

Una vez descrito el código para cada uno de los dispositivos desarrollados, solo resta mostrar la aplicación de este Sistema de Objetos para redes. Es muy importante resaltar la generalidad de los dispositivos, puesto que pueden representar casi cualquier tecnología para transmisión de datos sobre redes digitales.

En este trabajo se considerará el ejemplo de transmisión de datos sobre una red tipo SONET, con fines descriptivos únicamente, sin tratar de indicar que este Sistema de Objetos sea específicamente dirigido a la representación de esta Tecnología, ni restringir este trabajo en ese sentido.

Capítulo VII

VII.1 EJEMPLO

A continuación se verifica el rendimiento de una red formada por cinco servidores, tres líneas de comunicación y dos puentes, utilizando las definiciones establecidas y la Librería de Clases Propuesta **[27],[28],[30],[31]**.

Este ejemplo aplica las definiciones de rendimiento mínimo, establecido para los dispositivos de red SONET. Esto no restringe el uso de los modelos a éste tipo de tecnología. SONET se utiliza únicamente como referencia tecnológica.

Descripción del ejemplo

El diagrama de la figura 7.1 muestra la configuración de la red **[27]** a modelar utilizando la librería de clases propuesta, con los parámetros definidos de acuerdo al estándar SONET para comunicación digital de alta velocidad.

Se presenta una red con 3 PCs para usuarios, 2 del tipo NetBIOS y una TCP/IP, enlazadas a servidores de estos tipos (NetBIOS y TCP/IP), mediante 2 anillos FDDI y 2 puentes que segmentan la red.

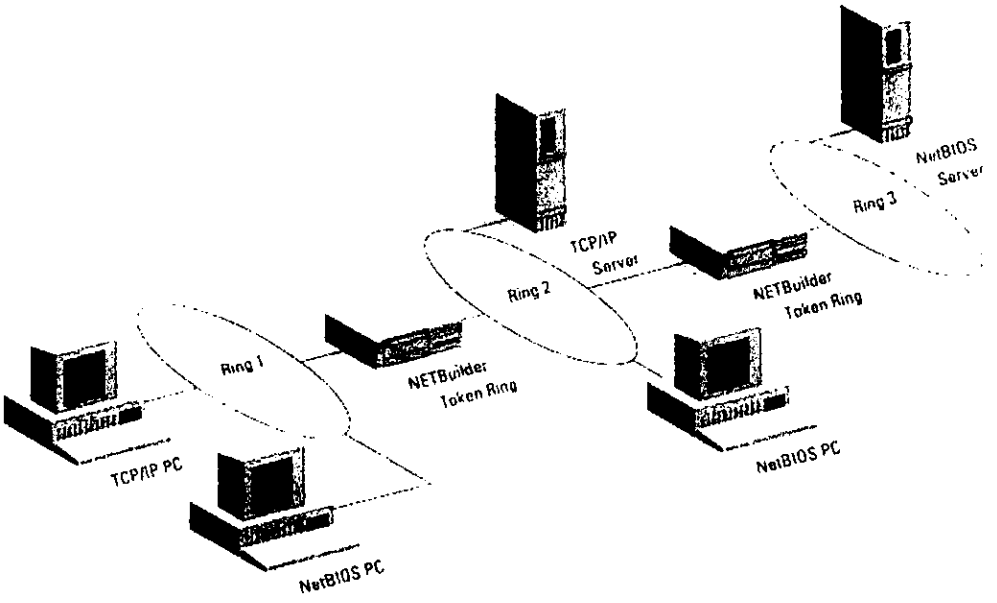


Figura 7.1. Descripción de la Arquitectura del Ejemplo 5s/3l/2p

Los parámetros para los dispositivos son los siguientes:

- PCs TCP/IP y NetBIOS: interfaz de entrada y salida de 8000 paquetes, acordes con la capacidad del canal (anillo FDDI tipo SONET STS-1 de 51.840 Mbps.).
- Servidores TCP/IP y NetBIOS: con interfaces de entrada y salida considerando capacidad de procesamiento de 100 Mbps. y el enlace a la línea de comunicación de acuerdo al estándar SONET STS-1.
- Anillos FDDI: velocidades de transferencia de 8,000 pps. (Nivel STS-1).
- Puentes: acorde con el estándar SONET STS-1, que garantiza una velocidad de transferencia de 8000 pps.

Código del Programa de Ejemplo 5s/3l/2p

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//              Librería de dispositivos para redes (C++)                                                                     //
//              Oct/1998                                                                                                      //
//              Autor: Ing. Raymundo Suarez Martinez                                                                          //
//                                                                                                                            //
//      Ejemplo de una red de computadoras, utilizando                                                                        //
//              el estándar SONET para transmisión y conmutación                                                              //
//              de alta velocidad.                                                                                            //
//                                                                                                                            //
//              Los dispositivos a utilizar son cinco DTE y dos DCE, distribuidos                                             //
//              de la siguiente manera:                                                                                       //
//                      PC de usuarios:         3 (Ethernet  de 10 Mbps.)                                                     //
//                      Servidores:             2 (1 TCP/IP y 1 NetBIOS, Fast Ethernet                                        //
//                                                      de 100 Mbps)                                                          //
//                      Líneas de transmisión:  3 (Anillos FDDI: Ring1, Ring2 y Ring3,                                        //
//                                                      tipo SONET STS-1, 51.840 Mbps)                                        //
//                      Puentes:                2 ( NETBuilder Token Ring)                                                    //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#include <stdio.h>
#include <c:\rayo\tesis\libreria\Conmuta.h>
#include <c:\rayo\tesis\libreria\Enruta.h>
#include <c:\rayo\tesis\libreria\Linea.h>
#include <c:\rayo\tesis\libreria\Puente.h>
#include <c:\rayo\tesis\libreria\Servi.h>

int main()
{
  // Se definen tres rutas de transmisión y recepción de información,
  // de las PC clientes a los servidores (TCP/IP y NetBIOS), correspondientes.

  float retardo_total_r1, retardo_total_r2, retardo_total_r3;

  // definición de los DTE (PCs y servidores)
  SERVIDOR PC_TCP1(8000, 8100, 1544, 8000),PC_NET1(8000, 8100, 1544, 8000),
                        SERV_TCP(8000, 15433, 15433, 8000),PC_NET2(8000, 8100, 1544, 8000),
                        SERV_NET(8000, 15433, 15433, 8000);
  // definición de las líneas
  LINEA R1(8000, 8100, 8000, 8100), R2(8000, 8100, 8000, 8100),
                R3(8000, 8100, 8000, 8100);
  // definición de los DCE (dos puentes)
  PUENTE P1(8000, 8100, 8000, 8100), P2 (8000, 8100, 8000, 8100);

  printf ("\n\n Programa que muestra el rendimiento de una red \n");
  printf (" tipo: 5s/3l/2p (5 servidores, 3 líneas y 2 puentes \n\n");
```

```
printf ("\n\n Rendimiento de los dispositivos: \n");

printf ("PC tipo TCP/IP: %3.1f %\n", PC_TCP1.Utilizacion(8000,8100));
printf ("PC tipo NETBios (1): %3.1f %\n", PC_NET1.Utilizacion(8000,8100));
printf ("PC tipo NETBios (2): %3.1f %\n", PC_NET2.Utilizacion(8000,8100));
printf ("Servidor tipo TCP/IP: %3.1f %\n",SERV_TCP.Utilizacion(8000,15433));
printf ("Servidor tipo NETBios: %3.1f %\n",SERV_NET.Utilizacion(8000,15433));

printf ("\n\n Retardos en las rutas: \n\n");

retardo_total_r1=PC_TCP1.Tiempo_Total(1544,8000)+ R1.Tiempo_Total(8000, 8100)+
                 P1.Tiempo_Total(8000, 8100)+ R2.Tiempo_Total(8000, 8100)+
                 SERV_TCP.Tiempo_Total(15433, 8000);
printf ("Ruta 1: PC_tcp/ip, Ring1, Puente1, Ring2, ServidorTCP \n");
printf ("       Retardo en la ruta: %f minutos \n",retardo_total_r1);

retardo_total_r2=PC_NET1.Tiempo_Total(1544,8000)+ R1.Tiempo_Total(8000, 8100)+
                 P1.Tiempo_Total(8000, 8100)+ R2.Tiempo_Total(8000, 8100)+
                 P2.Tiempo_Total(8000, 8100)+ R3.Tiempo_Total(8000, 8100)+
                 SERV_NET.Tiempo_Total(15433, 8000);
printf ("Ruta 2: PC_NETBios, Ring1, Puente1, Ring2, Puente2 ,");
printf ("Ring3, Servidor NETBios \n");
printf ("       Retardo en la ruta: %f minutos ", retardo_total_r2);

retardo_total_r3=PC_NET2.Tiempo_Total(1544,8000)+ R2.Tiempo_Total(8000, 8100)+
                 P2.Tiempo_Total(8000, 8100)+ R3.Tiempo_Total(8000, 8100)+
                 SERV_NET.Tiempo_Total(15433, 8000);
printf ("Ruta 3: PC_NETBios, Ring2, Puente2 ,");
printf ("Ring3, Servidor NETBios \n");
printf ("       Retardo en la ruta: %f minutos ", retardo_total_r3);

return 0;
}       // fin red1 5s/3l/2p.
```

Análisis de Resultados

El resultado de la ejecución del modelo se muestra en la figura , en donde se observan los resultados particulares de rendimiento de cada uno de los dispositivos, de acuerdo a los parámetros definidos para cada uno.

La figura muestra el rendimiento de las PCs y los servidores, considerando las velocidades de procesamiento a la entrada de cada una. Los retardos en las rutas se, refieren a las rutas definidas a través de al red para que las PCs tengan acceso al servidor que les corresponde:

- La ruta 1, va de la PC tipo TCP/IP hasta el servidor de TCP/IP, pasando por las líneas y dispositivos correspondientes.

- Las rutas 2 y 3, se refieren al camino que recorre la información para acceder desde las PCs tipo NetBIOS al servidor.

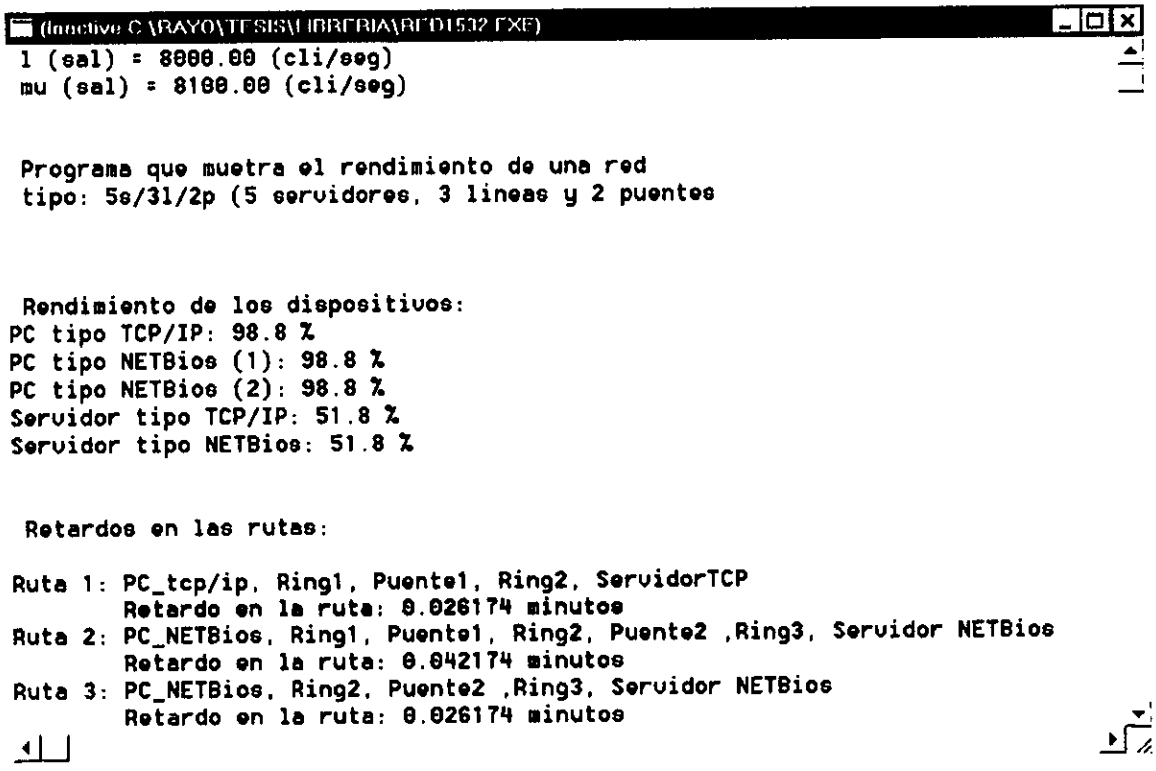


Figura 7.2. Resultado de la ejecución de los modelos.

Unos de los parámetros que es posible analizar mediante la aplicación de los modelos es el rendimiento de los dispositivos y los retardos en la transmisión de información, que indican algunos de los aspectos más importantes para el diseño de redes.

Como se muestra, los resultados de la simulación concuerdan en su comportamiento con el funcionamiento real de los sistemas de red. Por ejemplo, el retardo en la ruta dos, que va de la PC_NETBios que se encuentra en el anillo 2, hasta el Servidor NETBios, es la más larga y por lo tanto la que mayor retardo presenta (0.042174 minutos). Los servidores, por ejemplo, son también los que menos utilización presentan (51.8%) puesto que tienen mayor capacidad de procesamiento, esto se muestra en el rendimiento por dispositivo.

Comentarios

El ejemplo solo muestra una parte de los parámetros que es posible calcular mediante los modelos descritos, ya que como se menciona en la definición de los modelos, cada uno puede ofrecer una cantidad de parámetros además de los que se muestran en este ejemplo. La TOO permite además, adecuar y agregar funciones a cada uno de ellos de una manera sencilla, haciendo la librería extensible y adaptable.

Conclusiones

Como resultado de este trabajo, se ofrece una herramienta flexible que facilita la obtención de algunos de los parámetros más importantes del rendimiento de una red, con la finalidad de analizar su comportamiento actual y prever el futuro.

Se trabajó especialmente en la elección del nivel de abstracción como el punto de referencia básico para la definición de las redes. Del proceso de definición del nivel de abstracción, se derivaron los dispositivos que se consideran en este trabajo. Específicamente, se obtuvieron los dispositivos que imprimen algún retardo de tiempo al envío, redireccionamiento o transferencia de información, ya sea por sus características físicas o por la aplicación de algoritmos de decisión para el cumplimiento de su labor. De esta manera se obtiene que el retardo es el parámetro más importante en este trabajo, por que define en gran medida la eficiencia de los sistemas de red.

El retardo implica que las líneas de comunicación y los dispositivos de interconexión utilizan un cierto periodo de tiempo para procesar los paquetes de información, redireccionarlos o transferirlos. La definición del retardo en cada dispositivo, necesariamente involucra sus características físicas y lógicas propias. La implementación del retardo en los simuladores comerciales y públicos analizados, se realiza de una manera similar en todos ellos, mediante asignación manual o con algoritmos de espera que distan de coincidir con la realidad. Lo anterior provoca grandes diferencias de comportamiento entre los simuladores. La precisión que se presume en esta herramienta, se deriva de que los retardos no son asignados de manera aleatoria, sino mediante los datos ofrecidos por el fabricante, de los estándares tecnológicos o de los propios dispositivos durante su funcionamiento (aunque puede incluir valores aleatorios).

Se describieron los dispositivos de interconexión y los que generan la información que viaja por las redes, mediante modelos específicos de Líneas de Espera adaptados al funcionamiento de cada uno en particular, utilizando Teoría de Colas. Esta teoría es el instrumento analítico más desarrollado en cuanto a sistemas que definen su eficiencia mediante el cálculo de retardos de tiempo. Las definiciones obtenidas para los dispositivos, ofrecen una imagen instantánea de su rendimiento particular, y como parte integral de un Sistema de Red completo. Por lo tanto, como se demostró en el ejemplo, el uso de esta teoría aplicada a los dispositivos de intercomunicación para redes, permite efectivamente contestar algunas de las preguntas más comunes que surgen al efectuar el diseño de una red o al realizar un estudio de capacidad, de manera directa y simple.

Otra premisa básica en este trabajo es la *observabilidad* de los sistemas de red, donde todas las cantidades de rendimiento (utilización, velocidad de llegadas, tamaño promedio de las colas, velocidad promedio de procesamiento, distribución de llegadas, etc.) pueden obtenerse de acuerdo a datos reales, recolectados de los sistemas durante ventanas de tiempo finitas. Por lo tanto, el comportamiento de los dispositivos puede obedecer directamente al funcionamiento del dispositivo real o de acuerdo a la fuente generadora de sus parámetros de funcionamiento, incluyendo sus características tecnológicas propias, lo que provee la flexibilidad y adaptabilidad a esta herramienta. En particular los dispositivos modelados requieren de las razones de llegada y procesamiento de paquetes para calcular su eficiencia. Estos parámetros son comunes en los dispositivos de red y fáciles de obtener.

El uso de esta herramienta, permite eliminar la inestabilidad presente en los sistemas de simulación que manipulan los retardos, por que aplica funciones directas para el cálculo de los valores de rendimiento y capacidad.

Los resultados del ejemplo, muestran la correspondencia entre los valores de rendimiento ofrecidos por los dispositivos modelados y los obtenidos de sistemas reales. Resultados similares se obtienen al modificar los valores de rendimiento en los dispositivos, afectando el comportamiento global del sistema de red. Lo anterior confirma la versatilidad de los dispositivos, los cuales permiten implementar diferentes tecnologías, mediante el ajuste de los parámetros para cada uno y reafirman la capacidad de ahorrar en el análisis y diseño de Sistemas de Red, utilizando esta herramienta.

BIBLIOGRAFÍA

[1] A. Proskurowski, A. M. Farley, "Comunication Network Design Proyect; Proyect Description", www, http://www.cs.uoregon.edu/art/net.html, 1977.

[2] S. Misch, "Telecomunication Networks", Libro, Addison Wesley, 1993.

[3] H. Armbrüster, "The Flexibility of ATM: Suporting Future Multimedia and Mobile Communications", Artículo, IEEE Communications Magazine, February 1995.

[4] L. Kleinrock, "Queueing Systems; Volume I: Theory ", Libro, John Wiley & Sons, 1975.

[5] E. Cinlar, "Introduction to Stochastic Processes", Libro, Prentice Hall Inc., 1975.

[6] D. L.. Spohn, "Data Networks Design", Libro, McGraw-Hill, 1993.

[7] P. J. Furtier, G. R. Desrochers, "Modeling and Analysis of Local Area Networks", Libro, B & C, 1990.

[8] F. Halsall , "Data Comunication, Computer Networks and Open Systems Standars", Libro, New York, Addison-Wesley, 1992.

[9] H. Reynel, "Redes de Computadoras; I. Conceptos Básicos Generales", Revista, Redes de Computadoras, Año 4, Número 18, Verano 1994.

[10] D. L. Spohn, "Data Networks Design", Libro, McGraw Hill, 1993.

[11] P. J. Furtier, G. R. Desrochers, "Modeling Analysis of Local Area Networks", Libro, ISBN, 1990.

[12] A. M. Law, M. G. McComas, "Simulation Software for Communications Networks: The State of the Art", Artículo, IEEE Communications Magazine, March 1994.

[13] Páginas WEB, referencias:
ftp://ftp.cis.ufl.edu/p.../simdigest/tools/README
Paul Fishwick's Home Page
ftp://simserv.tuwien.ac.at/Help-and-News/Overview.txt
HotBot Results: Network Simulator
Directory of /pub/MaRS
MaRS Maryland Routing Simulator
Cengiz's Public Domain Software, MaRS + RA (routing analysis) toll
Welcome to MIL 3, OPNET
Maisie Programming Language
SMURPH and LANSF, System for Modeling Unslotted Real-time PHenomena
Simulator SMURPH University of Alberta, a Very Complete Simulator
NEST Network Simulation Tool
The REAL Network Simulator

Index of /ftp/global/pub/tenet/REAL/, REAL simulator tool
Packet Radio Network Simulator: Introduction
Welcome to Alta!
EUROSIM - ARGESIM HOME PAGE
The cnet network simulator on OSI model (full)
The Methusela Simulator
DQDBsim 802.6 Simulation package
TCP/IP Network Simulator from Australia
Installation and Use of the Caroline Simulator
SNMP Agent Simulator from VirtualDevices
SNMP Agent Simulator from Simplesoft
NetGuru Simulator
LBNL Network Simulator
Simulator from Naval Research Lab
Newbridge 46020 Network Simulator

[14] E. Cinlar, "Introducction to Stochastic Processes", Libro, Prentice Hall, 1975.

[15] L. Kleinrock, "Queueing Systems: Volume I: Theory", Libro, John Wiley & Sons, 1975.

[16] 3Com Corporation, "WAN Techn, rev.A; Digital Transmission Services", Manual, Cap. 4., 1992.

[17] G. Scott, "Queueing Network Models of Computer System Performance", Artículo, Computer Surveys, Vol. 10, No. 3, September 1978.

[18] P.J. Denning, J.P. Bunzen, "The Operational Analysis of Queeuing Network Models", Artículo, Computing Surveys, Vol. 10, No. 3, September 1978.

[19] J.W. Wong, "Queueing Network Modeling of Computer Communication Networks", Artículo, Computing Surveys, Vol. 10, No. 3, September 1978.

[20] C.A.. Rose, "A Measurement Procedure for Queueing Network Models of Computer Systems", Artículo, Computing Surveys, Vol. 10, No. 3, September 1978.

[21] Y. Bard, "The VM/370 Performance Predictor", Artículo, Computing Surveys, Vol. 10, No. 3, September 1978.

[22] K. Mani, C.H. Sauer, "Approximate Methods for Analyzing Queueing Network Models for MVS", Artículo, Computing Surveys, Vol. 10, No. 3, September 1978.

[23] J.P. Bunzen, "A Queueing Network Models of MVS", Artículo, Computing Surveys, Vol. 10, No. 3, September 1978.

[24] M. Reiser, "Mean-Value Analysis of Closed Multichain Queueing Networks", Artículo, Journal of the Association for Computing Machinery, Vol. 27, No. 2, April 1980.

[25] D. Towsley, "Queueing Network Models with State-Dependent Routing", Artículo, Journal of the Association for Computing Machinery, , Vol. 27, No. 2, April 1980.

[26] R.R. Muntz, "Queueing Networks: A Critique of the Art and Directions for the future", Artículo, Computing Surveys, Vol. 10, No. 3, September 1978.

[27] 3Com Corporation, "Bridgins and Roting, Thechnologies, Strategies and Benefits", Tutorial de Redes, 3Com Corporation, USA, 1994.

[28] 3Com Corporation, "Switches and Routers", Tutorial de Redes, 3Com Technical Papers, 3Com Corporation, USA, 1995.

[29] G. Byron S., "Programación en C", Libro, McGraw Hill, 1990.

[30] M. Katrib, "Programación Orientada a Objetos en C++", Libro, Informática Sistemas y Soluciones (Infosys), 1994.

[31] E. J. Hernández H., "Programación en C++", Libro, Paraninfo, Madrid, 1993.