UNIVERSIDAD NACIONAL DE INGENIERíA
FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

"ESTUDIO Y ANÁLISIS DE PROTOCOLOS Y TRÁFICO DE COMUNICACIONES EN REDES DE COMPUTADORAS"

INFORME DE INGENIERÍA

PARA OPTAR EL TÍTULO PROFESIONAL DE

INGENIERO ELECTRÓNICO

PRESENTADO POR

JOSÉ CARLOS BENÍTEZ PALACIOS

PROMOCIÓN
1988 - II

LIMA - PERÚ
2003

A mis padres José y Viviana,
por haberme inculcado
la ética de trabajo y superación.
A mi esposa Myrna,
por esas horas de compañía.
A mis hijos Andrew y Kylie,
que ponen luz a mi vida.
A mis hermanos
por su apoyo y ejemplo.
A mi familia,
por una huella en el terral del tiempo.
A Dios,
por mostrarme el camino.

ESTUDIO Y ANALISIS
DE PROTOCOLOS Y TRAFICO DE COMUNICACIONES
EN REDES DE COMPUTADORAS

SUMARIO

Se inicia la presente investigación realizando una explicación de conceptos generales básicos para entrar en los puntos centrales de discusión, donde se realiza un estudio y análisis de los protocolos de las redes de computadoras: especificación y validación, influencia en la estabilidad de las redes, verificación, y falla de seguridad. Asimismo se realiza un estudio y análisis de tráfico de comunicaciones en redes de computadoras, en la que se realiza una comparación de las distintas capacidades de tráfico. Al finalizar se presenta las conclusiones y recomendaciones de la investigación.

Las razones que justifican el presente proyecto son: El uso de los sistemas distribuidos va en aumento continuamente lo que conlleva a una ampliación permanente del ámbito de utilización de los protocolos, haciéndolos cada vez más complejos. Ha habido un crecimiento de varios órdenes de magnitud en la complejidad de los protocolos desde la aparición de los primeros terminales remotos hasta las actuales redes de computadoras. Como consecuencia, esto plantea nuevos retos de diseño, y la necesidad de una especificación precisa de protocolos. La especificación precisa de los protocolos es un problema de mucho interés, debido al objetivo de alcanzar una estandarización y a facilitar las realizaciones prácticas, para obtener una versión ejecutable de dicho protocolo en un computador. Se ha puesto poca atención a la caracterización del comportamiento del tráfico y la teoría de colas de los sistemas de redes de comunicaciones por lo que se cree conveniente realizarlas a través de la presente investigación.

Los objetivos que se han planteado para el desarrollo del presente proyecto son: a) Realizar una investigación de los protocolos en las redes de computadoras, descubriendo sus ventajas y desventajas; de los diferentes tipos de métodos de especificación, de las diversas técnicas de validación, determinar la influencia de éstos en la estabilidad de las redes de computadoras, y la falla de seguridad de los mismos. b) Obtener un análisis de las capacidades de tráfico de comunicaciones en las redes de computadoras, realizando un análisis matemático estadístico. c) Poder llegar a unas conclusiones que demuestren la calidad del estudio y análisis de nuestra investigación.

En la presente investigación se ha utilizado el método científico, definido como un procedimiento riguroso formulado lógicamente para lograr la adquisición, organización o sistematización y expresión o exposición de conocimientos tanto en su aspecto teórico como en su fase experimental. Se ha hecho uso de métodos cualitativos y cuantitativos. Haciendo un análisis del estado del conocimiento en el tema investigado que nos revela la revisión de la bibliografía y el enfoque que le pretendemos dar a nuestra investigación, los tipos de investigación que se utilizaron son: estudio exploratorio, estudio descriptivo y estudio explicativo. i) ***Estudio exploratorio*** porque nuestro objetivo es examinar un tema de investigación poco estudiado y que no ha sido abordado ampliamente. ii) ***Estudio descriptivo*** porque buscamos especificar las propiedades más importantes de los protocolos y tráfico de comunicaciones en redes de computadoras, que son sometidos a un análisis. Se ha seleccionado una serie de cuestiones y se mide cada una de ellas independientemente, para así describir lo que se investiga. Se mide de manera más bien independiente los conceptos con los que se analiza. iii) ***Estudio explicativo*** porque también vamos más allá de la descripción de conceptos, respondemos a las causas de algunos eventos de protocolos y tráfico.

INDICE

CAPÍTULO II

ESTUDIOS DE PROTOCOLOS

PROLOGO

Es bien conocido que el poder, la prosperidad y la democracia de los países, pueblos e individuos dependen de la posesión y distribución de la información, así como de la capacidad de comunicación de los hombres. Nadie puede dudar que a la revolución industrial a sucedido la revolución informática. Para que la revolución informática sea posible no es suficiente procesar los datos; sino también se debe tener la capacidad de acceder a los datos, de acumularlos, de transformarlos, de divulgarlos, compartirlos, etc.. Y para realizar todas éstas funciones se requiere la capacidad de comunicar datos.

La evolución de la tecnología ha acercado a los electrónicos a la informática, hoy es imprescindible el uso de la informática en la realización de un diseño digital. El profesional de la electrónica debe acercarse a la informática, convirtiéndose en poco(o en mucho) informático, y en el camino ha sido necesario adquirir y dominar conceptos de arquitectura de computadoras, y de ingeniería de software; conceptos antes reservados para los profesionales de la informática.

En los últimos años ha habido un crecimiento de varios órdenes de magnitud en la complejidad de los protocolos desde la aparición de los primeros terminales remotos hasta las actuales redes de computadoras. Como consecuencia esto plantea nuevos problemas de diseño, y la necesidad de una especificación precisa del protocolo. La especificación precisa de un protocolo es un problema de mucho interés, debido al objetivo de alcanzar una estandarización y a facilitar las realizaciones prácticas. El uso continuo y creciente de los sistemas distribuidos conlleva a una ampliación permanente del ámbito de utilización de los protocolos, haciéndolos cada vez más complejos.

La teoría de colas es la teoría de los procesos estocásticos aplicados al estudio de los sistemas de colas. Una subred de comunicaciones de una red consiste en un medio de transmisión y en un conjunto de interfaces para la conexión de los usuarios. La subred de comunicación de una red local es un ejemplo de sistema de colas. En este caso, un cliente representa un paquete de datos y el servidor es el medio de transmisión que presta el servicio de transmisión de paquetes entre las interfaces de la subred. La llegada de un cliente al sistema equivale a la sumisión de un paquete de datos para ser transmitido y la idea de éste corresponde a la finalización de una transmisión con éxito. El sistema de colas que modela la subred en este ejemplo tiene muchas colas (una para cada interfaz de la red) que están siendo atendidas por sólo un servidor. Sin embargo, otros sistemas de colas pueden tener más de un servidor que atienda a una misma cola. Algunos sistemas de colas están compuestos por un número de subsistemas interligados en red. En tales sistemas los clientes reciben servicio en más de un subsistema (o no) antes de tener atendidos todos sus requisitos de servicio. Normalmente, un cliente inicia su peregrinaje en un nodo dado, espera su turno en la cola y cuando es atendido va hacia otro nodo para obtener otro servicio más, y así sucesivamente hasta tener atendidos todos sus servicios.

CAPÍTULO I
CONCEPTOS PRELIMINARES

1.1. Introducción

1.1.1. Sistemas Informáticos

Un sistema informático está constituido por un conjunto de elementos de hardware y software, capaces de realizar conjuntamente, una determinada función orientada hacia la resolución de un problema objeto. Ver Figura 1.1. Los elementos de un sistema informático son tareas (programas) y recursos (archivos, periféricos, etc.), entre los que podrán establecerse determinadas relaciones. Ver Figura 1.2. Por ejemplo: Comunicaciones entre tareas con el objetivo de intercambiar información para la cooperación en la resolución de una determinada función; comunicaciones entre tareas y recursos con el objetivo de que las tareas utilicen los servicios de los recursos.

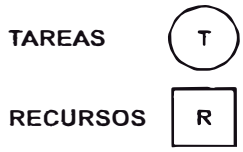


Figura 1.1. Elementos que constituyen un sistema informático.

Los sistemas informáticos se subdividen en sistemas centralizados y sistemas distribuidos. En un sistema centralizado, es decir basado en un solo computador, la responsabilidad del establecimiento de dichas relaciones corresponde generalmente a un sistema operativo, bien sea orientado a la manipulación de archivos o bien sea orientado hacia el control de aplicaciones en tiempo real, y estará constituido básicamente por un conjunto de programas que se ejecutan en el propio computador y que realizan dicho servicio de comunicación entre los elementos. (Ver Figura 1.3).

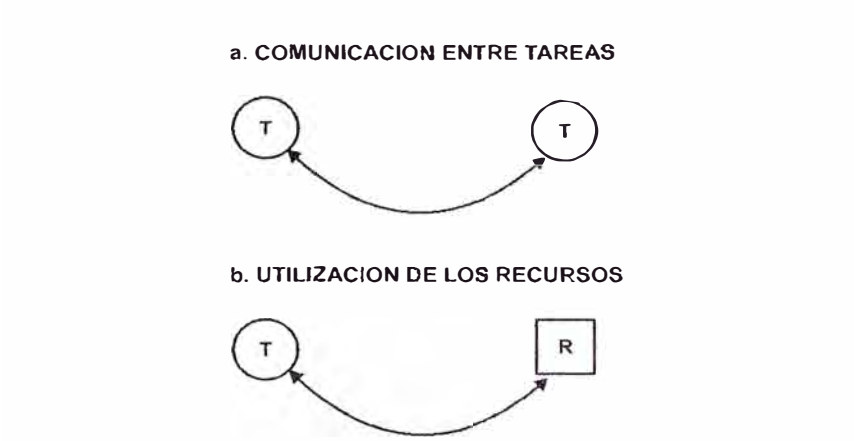


Figura 1.2. Relaciones entre los elementos de un sistema informático.

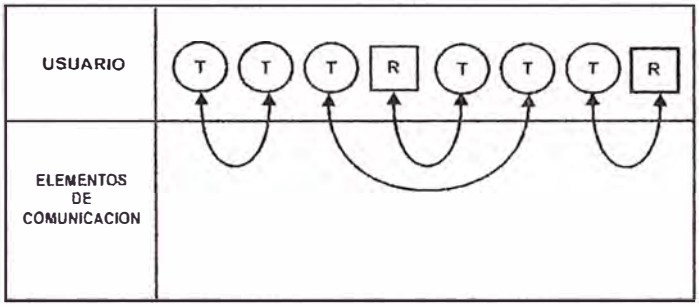


Figura 1.3. Relaciones en un sistema centralizado.

Un sistema informático distribuido está constituido por un conjunto de elementos (tareas y recursos) entre los cuáles se establecen relaciones. La particularidad consistirá en que los elementos que se comunican están ubicados en máquinas diferentes y distribuidas. (Ver Figura 1.4). Es evidente que el mecanismo que hace posibles dichas relaciones debe estar distribuido entre los diferentes componentes que constituyan el sistema. Dicho mecanismo está formado por: un conjunto de elementos de software (programas), residentes en las máquinas distribuidas, a los que se les denomina sistema operativo distribuido, y además un conjunto de elementos de hardware y/o software lo que constituye el mecanismo de comunicación e interconexión entre los elementos de tratamiento de la información.

Un sistema distribuido, es un sistema informático, en el que la potencia del tratamiento de la información se encuentra repartido (distribuido) en el espacio, entre todos los Elementos de Tratamiento de la Información (ETI) a través de un Mecanismo de Comunicación e

Interconexión (MCI). Un sistema distribuido presenta el esquema general mostrado en la Figura 1.5, en el que aparece un conjunto de Elementos de Tratamiento de la Información (ETI) interconectados mediante un Mecanismo de Comunicación e Interconexión (MCI).

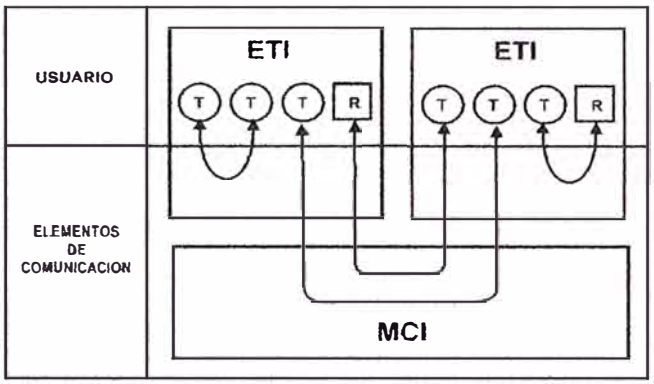


Figura 1.4. Relaciones en un sistema distribuido.

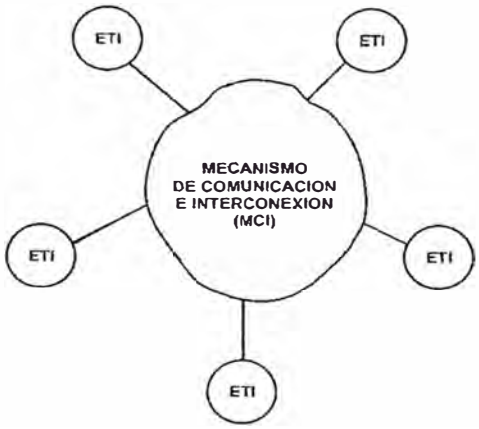


Figura 1.5 Esquema general de un Sistema Distribuido.

Una clasificación de los distintos tipos de Sistemas Distribuidos basados en una escala de distancias entre los ETI, es la siguiente (En la Figura 1.6 se muestra dicha clasificación):

a. Redes de Computadoras.
b. Redes Locales de Computadoras
c. Sistemas multicomputadoras
d. Sistemas multiprocesadores

Las redes de computadoras surgen a finales de los años 60 como una solución para la interconexión de computadoras situados en lugares remotos con el objetivo fundamental de compartir recursos, permitiendo a cualquier usuario de cualquier computador acceder y utilizar

los recursos ya sean hardware o software, del conjunto de máquinas que constituyen la red. Las redes de computadoras tuvieron su influencia decisiva en el desarrollo de las denominadas redes locales de computadoras. Las redes locales de computadoras se iniciaron a principios de los años 70 y trataron de aplicar, a escala reducida, soluciones experimentadas en las redes de computadoras, simplificando y optimizando dichas soluciones y sacando partido de las ventajas que reporta la disminución de la distancia entre los elementos del proceso.

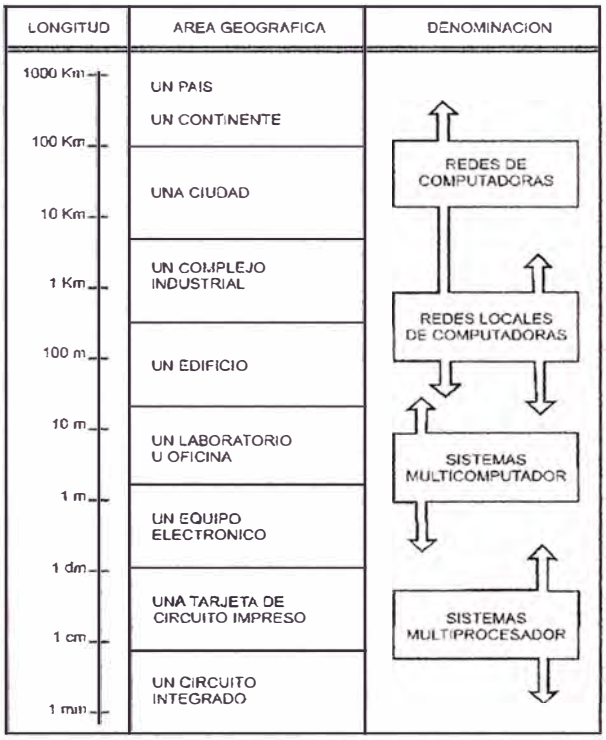


Figura 1.6 Clases de Sistemas Distribuidos.

Los sistemas multicomputadoras fueron desarrollándose durante la década de los años 60, aplicando la idea de la descentralización de funciones en un computador. Así aparecieron en los sistemas clásicos, unidades especializadas, en la manipulación de periféricos, en la gestión de comunicaciones, etc. Los sistemas multiprocesadores se refiere a máquinas potentes para el tratamiento de la información, basadas en la cooperación sistemática y ordenada de elementos de menor potencia, funcionando en paralelo. La realización de máquinas paralelas se han visto

superadas por la propia evolución de la tecnología al realizar máquinas secuenciales potentes y rápidas, limitando los sistemas paralelos a casos muy específicos donde la velocidad y las características del problema justifican tal desarrollo, como es el caso de los sistemas de tratamiento de señales en tiempo real. Actualmente existen componentes integrados muy complejos basados en la utilización de más de una unidad de proceso dedicada a la realización de funciones especializadas como solución para aumentar la potencia del tratamiento de la información del componente, sistemas que se podría clasificar en la categoría de los sistemas multimicroprocesadores.

1.1.2. Protocolos

Al conjunto de reglas que regulan el intercambio de información entre elementos de un sistema informático que cooperan, se le denomina protocolo. En un sistema informático distribuido un protocolo permitirá fundamentalmente iniciar, mantener y terminar un diálogo entre elementos del sistema, asimismo, un protocolo regulará la forma en que deberán generarse e interpretarse los elementos orientados al control de errores y la forma de recuperar las informaciones recibidas erróneamente, igualmente estarán previstas en un protocolo la forma de identificar el camino que se utiliza para el intercambio de la información y la identificación del tipo de mensajes. Los elementos del diálogo de un protocolo serán mensajes. Dentro de cada mensaje, además de los datos, objeto final del diálogo, existirán otras informaciones destinadas a permitir: la detección de errores, la identificación del camino, el control de flujo de información y la identificación del tipo de mensaje que se trate. Todas estas informaciones se materializarán en bloques con una determinada estructura que constituirá su formato. El establecimiento del diálogo implicará la existencia, en los entes que se comunican, de elementos que materializan los algoritmos de generación e interpretación de los mensajes, según las reglas que constituyen el protocolo.

Por lo tanto; un protocolo es un conjunto de reglas y normas, que aseguran que el intercambio de datos entre computadores y otros dispositivos en una red sea eficaz y fiable. Sin protocolos, el intercambio de datos entre puntos de una red es imposible establecerse y mantenerse. Es necesario disponer de protocolos muy detallados para definir el formato en el que se van a enviar los datos, y para controlar el tráfico de la red. Una transmisión de datos sólo puede efectuarse cuando los dispositivos de una red utilizan el mismo protocolo. Los protocolos no funcionan de forma aislada, sino que lo hacen como parte del juego de instrucciones que determinan las operaciones de un dispositivo o de una red. Los protocolos están diseñados para trabajar en condiciones diferentes y satisfacer distintas necesidades. El número de métodos (protocolos) posibles para el intercambio de datos y mensajes entre computadoras y dispositivos de una red es enorme.

1.1.3. Diseño y desarrollo de protocolos

Un protocolo es un conjunto de reglas que hacen posible y ordenan la comunicación entre entidades cooperantes en sistemas abiertos, posiblemente heterogéneos. Un protocolo no puede ser descrito informalmente porque existe el riesgo de no cubrir todas las posibles situaciones o estados de comunicación a controlar, porque también existe el riesgo de funcionar indebidamente bajo ciertas condiciones y de ser implementado en sistemas distintos por equipos diferentes, siguiendo interpretaciones particulares incompatibles. El resultado es un protocolo que, cuanto más complejo, más difícil será que haga posible alguna cosa, y mucho menos que haga posible la comunicación entre sistemas heterogéneos. Es imprescindible que la especificación o descripción de un protocolo sea concisa y precisa, totalmente ausente de ambigüedades; y esto sólo se obtiene a través de una especificación formal del protocolo. Esto no quiere decir que la descripción informal (asimismo casual) del protocolo deje de tener su propio mérito. Además, el desarrollo inicial de los protocolos utilizó la descripción informal

probablemente como resultado de la inexistencia o inadvertencia de técnicas formales de especificación de protocolos consolidadas. En conclusión, cuanto más sofisticados y complicados sean los protocolos, se hará más necesario dar especificaciones a través de técnicas formales. Por lo general la descripción informal de un protocolo se realiza en beneficio de su comprensión de funcionamiento, pero es necesario dar sus especificaciones formales, porque sirve de base para la validación, verificación, comprobación e implementación del protocolo. La validación y verificación de un protocolo son actividades importantes durante su diseño. Esa importancia viene reforzada cuando el protocolo se va a convertir en un producto comercial, y más aún, cuando el protocolo que esta siendo especificado va a ser propuesto como norma.

Por ***validación*** de un protocolo se entiende aquellas actividades necesarias para mostrar o asegurar que la especificación e implementación del protocolo van a satisfacer las necesidades de comunicación para las cuáles está siendo diseñado. Las actividades de validación, desempeñadas durante todo el diseño del protocolo, pueden incluir estudios de simulación, modelado analítico, etc. La validación de un protocolo también es conocido como evaluación de prestaciones.

La verificación de protocolos es un único aspecto de la validación de protocolos. La verificación de protocolos es la determinación de ciertas características lógicas de especificación del protocolo que indican si tiene defectos o no (como por ejemplo: la posibilidad de impasse –deadlock-). Las operaciones para la verificación de un protocolo son por lo tanto realizadas después de la especificación de protocolos. El uso de técnicas formales en la especificación contribuye incuestionablemente a facilitar y hacer posible esas operaciones. Una vez verificadas las especificaciones de un protocolo, el siguiente paso es implementarlo. Una tarea importante es realizar una prueba de consistencia de la implementación (o de parte de la implementación) con las especificaciones del protocolo.

Especificación formal de protocolos

El diseño de un protocolo se basa principalmente, en la especificación formal del protocolo. La especificación del protocolo debe incluir exactamente todas las condiciones que se deben satisfacer y nada más. Debe expresar lo esencial y omitir lo no esencial; debe ser clara y precisa. Es ahí donde reside la dificultad, ya que estos dos predicados, son a veces antagónicos. Las técnicas formales de especificación de protocolos existen para que se hagan especificaciones con esos dos predicados. Nosotros trataremos las técnicas formales relacionadas con la especificación de protocolos de comunicación para redes de computadoras. La arquitectura de red que adoptamos es la arquitectura para la interconexión de sistemas abiertos de la ISO, el RM-OSI está estructurado en siete capas jerárquicas de protocolos donde cada capa ofrece servicios a la capa inmediatamente superior. La capa 7 ofrece servicios de comunicación a los procesos que están siendo ejecutados en varios sistemas conectados a la red. Los servicios de cada capa son suministrados por el protocolo de la capa. Entonces surge la especificación de servicios y la especificación del protocolo.

Especificación de servicios

La especificación de servicios de un protocolo consiste en la descripción del comportamiento de entrada y salida de la capa del protocolo correspondiente, es decir, la especificación de los servicios del protocolo N describe el servicio de comunicación ofrecido por la capa N. La especificación de servicios del protocolo N debe definir las primitivas del servicio N, cuando se ordene la ejecución de esas primitivas y se ejecute cada una de ellas. Por tanto, no se debe de ocupar en definir detalles de implementación de las primitivas. Esos detalles son propios de la interfaz entre las capas N y N+1, y pueden variar en función del sistema operativo bajo el cual se va ha ejecutar la implementación final de la interfaz y del propio protocolo, del lenguaje de codificación empleado, etc. La utilización de técnicas para la

especificación formal de servicios es escasa. Se destacan el uso de métodos de ingeniería de software y en general de métodos descriptivos de la historia de las entradas y salidas de la capa del protocolo. La especificación formal de protocolos se centra más en los protocolos que en los servicios que ofrecen.

Especificación de un protocolo N

La especificación de los servicios de la capa N no define cómo suministra dichos servicios el protocolo N. La especificación del protocolo N define cada entidad N, pero sólo, con la extensión necesaria para asegurar la compatibilidad con las otras entidades de la capa. Pero después, la cuestión de cómo implementar la entidad es dejada abierta para dar libertad a la hora de escoger métodos de implementación. En la especificación del protocolo N, debe describirse la capa N. La propia ISO presenta directrices de la descripción de capas. De acuerdo con ellas, en la descripción de cada capa deben constar los siguientes puntos:

1. Una exposición general de los objetivos de la capa y de sus servicios
2. Una especificación exacta del servicio ofrecido por la capa.
3. Una especificación exacta del servicio suministrado por la capa N-1.
4. La estructura interna de la capa N, en términos de las entidades N y de sus relaciones.
5. Una descripción del (los) protocolo(s) entre las entidades N, que incluye:
   a) Una descripción general e informal de la operación de las entidades.
   b) Una especificación del protocolo que incluya:
      i) Una lista de los tipos y formatos de los mensajes transferidos entre las entidades N.
      ii) Las reglas que gobiernan la reacción de cada entidad N a las órdenes de las interfaces de otras entidades y sucesos internos.
   c) Detalles adicionales, tales como consideraciones para mejorar las prestaciones, sugerencias de implementación, o una descripción pormenorizada que se acerque a la implementación.

El punto 3 es redundante con la descripción de la capa N-1; se ha incluido para hacer que la especificación de protocolo N sea independiente de las demás. Se puede hacer una especificación formal del protocolo N mediante el uso de varias técnicas. Estas técnicas son

clasificadas en tres categorías: Modelos de transición, lenguajes de programación y modelos mixtos de las categorías anteriores.

A. Modelos de Transición

En esta categoría se clasifican los modelos de Maquina de Estado Finito (MEF), Redes de Petri (RP), lenguajes formales, los llamados gráficos de UCLA y coloquios, entre los más conocidos. De todos éstos los más populares son las MEF y las RP. Las RP son las más indicadas en la etapa inicial de especificación de protocolo; eso se debe a que están más próximas a un lenguaje natural y facilitan la identificación de nuevas situaciones o estados para los que trascurre el protocolo, en respuesta a la aparición de sucesos o transiciones. Por ello, en las etapas siguientes, cuando se retoca y se finaliza la especificación del protocolo la realización de MEF es más ventajosa, pues hace posible una especificación más compacta. En realidad, una RP es una MEF sin equivalencias y una puede ser transformada en otra sin mucha dificultad.

B. Lenguajes de Programación

A partir de un cierto grado de complejidad del protocolo la explosión de estados es tan grande, que se hace inviable la especificación a través de modelos de transición. Existen algunos métodos para facilitar esa dificultad. La especificación formal del protocolo entonces tiene que hacerse con otras técnicas, una de ellas utiliza lenguajes de programación. El interés en utilizar lenguajes de programación para especificar protocolos surgió con la observación de que un protocolo es en realidad un algoritmo y puede así describirse de forma clara y precisa con un lenguaje de programación de alto nivel. La especificación del protocolo toma entonces forma de programa escrito en un lenguaje tal como PROLOG, PASCAL, BASIC, Lenguaje C, y otros lenguajes..

C. Modelos Mixtos

Los modelos mixtos, también conocidos como modelos híbridos, realizan la especificación de un protocolo partiendo de modelos de transición (generalmente una MEF) y finalizando con lenguajes de programación. Los modelos mixtos combinan las ventajas de los modelos de transición y los lenguajes de programación. Los aspectos de control del protocolo son especificados con el uso de modelos de transición, mientras que las variables y otros parámetros de los estados son manipulados en un programa en un lenguaje de programación de alto nivel.

Normalmente las especificaciones de un protocolo mediante un modelo mixto utiliza una MEF pequeña (es decir una MEF con pocos estados) que sólo capta los detalles principales del protocolo. Esta pequeña MEF es ampliada a continuación con variables adicionales y rutinas de procesamiento de esas variables. Esto se hace asociando a cada transición de la MEF una rutina de procesamiento en un determinado lenguaje de alto nivel. Esa rutina puede probar y manipular las variables adicionales sin que estas tengan que ser incluidas en el estado de la MEF, reduciendo sustancialmente así, la complejidad de la MEF.

Por ello, si por un lado la inclusión de las variables en el estado de la MEF reduce su complejidad tenemos un problema en la especificación de las transiciones. La transición se dispara cuando se produce un suceso, pero ese suceso debe incluir valores para las variables auxiliares de la MEF. Si quisiéramos especificar todas las transiciones posibles, tendríamos una explosión del conjunto de entrada, E. Para eliminar este problema añadimos a cada transición un predicado de habilitación que es una expresión lógica que engloba a las variables de la MEF, y cuando es verdadera, posibilita el disparo de la transición. Se define entonces una MEF con variables auxiliares, rutinas de procesamiento y predicados de habilitación llamada MEFA (Máquina de Estados Finita Ampliada).

Verificación de Protocolos

La especificación formal de un protocolo, no garantiza por sí sola, que el protocolo no vaya a funcionar de manera no deseada o indebida. Porque pueden introducirse errores en la especificación o la especificación puede ser incompleta en el sentido de no cubrir todas las posibles situaciones de funcionamiento del protocolo. Surge entonces la necesidad que sea verificada la especificación para garantizar que esté libre de defectos o de propiedades anómalas. La verificación de un protocolo N se da a través del análisis de las interacciones entre las entidades de la capa N (interacciones que concuerdan con la especificación del protocolo N y que utilizan el servicio de comunicación de la capa N-1) para determinar si la operación conjunta de las entidades satisface la especificación del servicio de la capa N. La verificación del protocolo N es, por tanto dependiente de la verificación del (de los) servicio(s) de la capa N-1. Para eliminar esta dependencia y así atribuir los posibles defectos encontrados por la verificación, la especificación del protocolo N, supone que la capa N-1 funciona correctamente y suministra el servicio deseado. De esta forma la verificación del protocolo N se reduce básicamente a la demostración de que tiene ciertas propiedades necesarias para que su comportamiento sea el deseado, es decir, que ofrezca el servicio N, tal y como era esperado. Para verificar un protocolo primero se debe identificar las propiedades que el protocolo debe tener. A continuación, tenemos que probar que la especificación del protocolo satisface las propiedades identificadas.

Existen propiedades generales que son implícitas en todas las especificaciones, de las cuáles algunas de ellas, las más importantes, son:

- Ausencia de retardo (deadlock)
- Actividad (liveness)
- Realización de progreso
- Complitud.

- Terminación
- Corrección parcial
- Minimidad
- Estabilidad

A continuación se presenta una descripción resumida de cada una de ellas:

- ***Ausencia de retardo (deadlock)***: Propiedad que garantiza que el protocolo, bajo ninguna condición o circunstancia, llegará a un estado de inactividad total, permaneciendo allí por tiempo indefinido.
- ***Actividad (liveness)***: Propiedad que asegura el cambio del protocolo de un estado a otro de manera que, partiendo de cualquier estado, se alcancen (eventualmente), todos los demás estados.
- ***Realización de progreso***: Esta propiedad hace que el protocolo no presente comportamientos no útiles, o de forma equivalente, no permanezca en un estado de inactividad más que durante un tiempo finito.
- ***Complitud***: Propiedad que asegura que la especificación para cada estado dé una respuesta a todas las entradas posibles.
- ***Terminación***: Esta propiedad hace que cada operación del protocolo termina eventualmente en un intervalo de tiempo finito.
- ***Corrección parcial***: Esta propiedad hace que al término de una operación el protocolo produce el resultado correcto.
- ***Minimidad***: Esta propiedad hace que el protocolo engloba todas las situaciones que pueden producirse.
- ***Estabilidad***: Esta propiedad asegura que después de un fallo, el protocolo vuelve al funcionamiento normal dentro de un intervalo finito. Esta propiedad está relacionada con la propiedad de autosincronización.

1.2. Protocolos de redes locales

A continuación se enumeran los protocolos más adecuados a las redes locales:

- De contienda:
  - Protocolo de contienda simple
  - CSMA
    Carrier Sense Multiple Access
    Protocolo de Acceso múltiple por detección de portadora.
  - CSMA/CD
    Carrier Sense Multiple Access with Collision Detection
    Protocolo de Acceso múltiple por detección de portadora con detección de colisiones
  - CSMA/CA
    Carrier Sense Multiple Access with Collision Avoidance
    Protocolo de Acceso múltiple por detección de portadora evitando colisiones
- Polling

  Protocolo de Llamada selectiva.
- Token Passing

  Protocolo de paso de testigo.

1.2.1. Protocolos de contienda

En sentido ilustrativo, "Contienda" es lo que sucede en una reunión cuando varias personas comienzan a hablar al mismo tiempo. En los protocolos de contienda no hay nada que controle el uso de los canales de comunicación.

El protocolo de contienda es un método de acceso a la línea basado en que el primero que llega es el que la utiliza. Una clasificación de los protocolos de contienda es la siguiente:

A) Contienda simple.

B) CSMA.

C) CSMA/CD.

D) CSMA/CA.

A continuación se describe cada una de ellas.

A) Contienda Simple

En una red que utilice el protocolo de contienda simple todas las estaciones comparten un canal de transmisión común; los mensajes se envían a través de ese canal (Ver figura 1.7). Las estaciones sólo responden a los mensajes que incluyen su dirección; el resto de los mensajes se ignoran. Cuando las estaciones no están respondiendo a un mensaje, permanecen en estado de espera escuchando el canal, hasta recibir uno que lleve su dirección. En cada estación los mensajes que se van a transmitir se convierten en paquetes y se envían cuando están listos, sin mirar siquiera si el canal está disponible. Cuando un paquete de una estación coincide con el de otras estaciones, se produce una colisión. Los paquetes que chocan entre sí se destruyen automáticamente y las estaciones de donde éstos proceden han de enviarlos de nuevo. El protocolo básico de contienda no se preocupa de saber si ya hay otro mensaje en la línea, lo único que hace es avisar que la estación ha recibido el paquete. Si la estación emisora no recibe un "Acuse de recibo", supone que el mensaje no se ha recibido o que ha sido destruido. La estación emisora espera un cierto tiempo aleatorio y vuelve a trasmitir el paquete. El tiempo aleatorio transcurre y se vuelve a transmitir el paquete. El tiempo de espera ha de ser aleatorio o los mismos mensajes volverían a colisionar indefinidamente una y otra vez. En algunas ocasiones la estación receptora sólo recibe parte del paquete. Entonces el receptor envía al emisor un aviso de que no ha recibido el paquete y le pide que vuelva a transmitirlo.

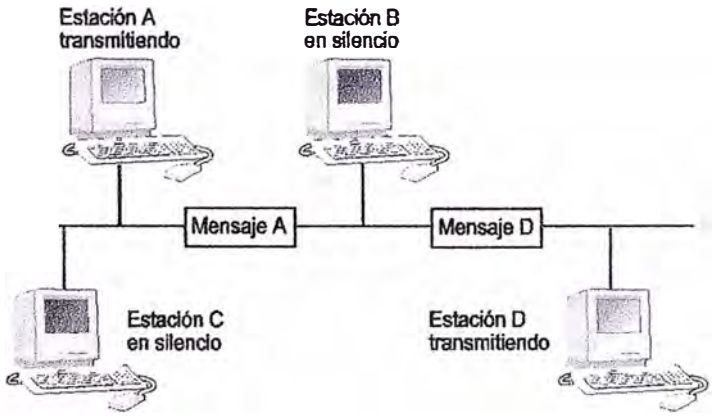


Figura 1.7 Protocolo de contienda simple.

Casi todas las redes locales actuales utilizan métodos de contienda más sofisticados. Algunos de estos métodos y los más importantes son: Acceso Múltiple por Detección de Portadora (CSMA), Acceso Múltiple por Detección de Portadora con Detección de Colisiones (CSMA/CD) y Acceso Múltiple por Detección de Portadora Evitando Colisiones (CSMA/CA). A continuación se hace un resumen de cada uno de estos protocolos. Las características generales que presentan cada uno de éstos son similares a las del método estándar de contienda.

B) CSMA (Acceso múltiple por detección de portadora)

Ilustrativamente podríamos decir que el acceso múltiple por detección de portadora (CSMA) sería el método empleado por los asistentes de una reunión cuando solicitan turno para hablar; cada uno de ellos hablaría cuando no hubiese nadie hablando. Al igual que en el método de contienda simple, las estaciones comparten un único canal de comunicaciones (Ver figura 1.8). Antes de enviar información, la estación se pone a la "escucha", normalmente en una frecuencia secundaria, para saber si otra estación está usando el canal principal de transmisión, es decir, la portadora. Cuando la línea queda libre, la estación comienza a transmitir. Si una estación está lista para transmitir y hay otra estación transmitiendo, la primera detecta la señal y no envía el mensaje hasta que la transmisión de la segunda haya terminado.

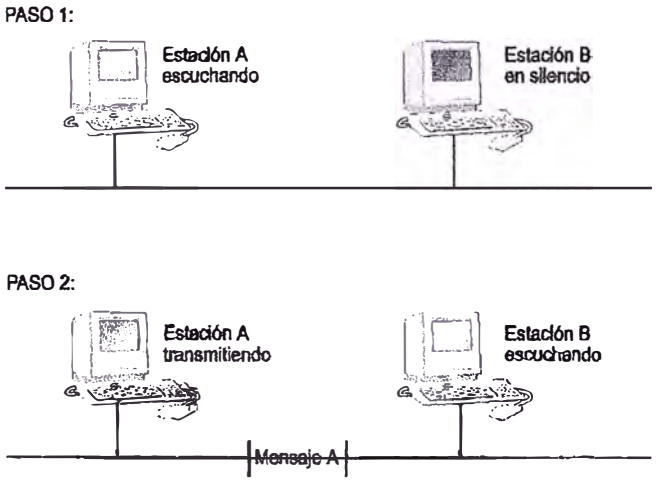


Figura 1.8. Protocolo CSMA (Acceso múltiple por detección de portadora).

En las redes de computadoras que utilizan este protocolo, la estación que está esperando tiene dos opciones.

a) Escuchar continuamente el canal, a la espera de que cese la señal de "ocupado", y entonces transmitir inmediatamente. Este método se conoce como "detección continua de portadora", puesto que la estación está comprobando continuamente si queda libre el canal para acceder a él y poder transmitir. Si coincide que hay alguna otra estación en la misma situación, se producirá una colisión al quedar libre el canal de comunicación.

b) Ver si el canal está ocupado, y si lo está dejar la transmisión para más tarde. Para determinar el tiempo que ha de transcurrir hasta que se vuelva a comprobar si el canal está libre, se utiliza un algoritmo aleatorio, y transcurrido ese tiempo vuelve a intentarlo. Este método se denomina "detección no continua de portadora". Con él se producen menos colisiones, y por tanto, aumenta el rendimiento general.

Cada estación además de transmitir el mensaje, emite otra señal a través del canal secundario para avisar al resto de las estaciones que la línea está ocupada. La estación emisora, una vez que ha transmitido el mensaje, espera hasta recibir una señal de aceptación (lo que hemos venido llamando "acuse de recibo"). Si no se recibe esta señal, o si se recibe una señal negativa, la estación supone que se ha producido una colisión; entonces espera un cierto tiempo antes de iniciar de nuevo el proceso. La colisión de mensajes entre estaciones en una red CSMA (Acceso Múltiple por Detección de Portadora), parece inevitable. Debido al tiempo que tarda la señal en propagarse a lo largo del canal; dos o más estaciones pueden encontrar al mismo tiempo libre la línea y, por tanto, intentar enviar un mensaje simultáneamente. Si el tiempo que tarda la señal en recorrer todo el canal de comunicación es corto, la información que la estación recoge de la línea es lo suficientemente actual como para tomar una decisión que no produzca una colisión, con lo que la probabilidad de acceder a la línea sin tener colisiones es

bastante más alta que con el método de contienda simple. Si, por el contrario, la información no está actualizada (el canal de comunicación es bastante largo), el método CSMA sólo ofrece una pequeña mejora en comparación con el método de contienda simple.

C) CSMA/CD (Acceso múltiple por detección de portadora con detección de colisiones)

Ilustrativamente el protocolo de acceso múltiple por detección de portadora con detección de colisiones (CSMA/CD) establece las normas para que cuando en una reunión dos personas comienzan a hablar al mismo tiempo, las dos personas dejan de hablar y esperan a que sólo una continúe hablando; la primera que comience a hablar será la que tenga la palabra, y hablará. Cada estación con el protocolo CSMA/CD antes de comenzar a transmitir, además de saber si alguien está usando el canal de comunicación comprueba si se ha producido una colisión y, si es así, se detiene la transmisión. Cuando no se tiene éxito en la transmisión de datos (al igual que en el resto de los protocolos de contienda), el mensaje se vuelve a enviar al cabo de unos instantes. En el caso del protocolo CSMA/CD el intervalo de retransmisión puede estar predefinido o ser aleatorio. Como la estación transmisora comprueba si la línea está libre antes y durante la transmisión, el número de colisiones es relativamente bajo y, por tanto, el rendimiento es mayor.

D) CSMA/CA (Acceso múltiple por detección de portadora evitando colisiones)

Ilustrativamente es lo que sucede en una reunión donde hay varias personas que quieren hablar al mismo tiempo y levantan la mano a la vez para solicitar el uso de la palabra. En este caso, el moderador de la reunión decide quién es la persona que va a hablar a continuación. Esto mismo es lo que hace el protocolo de acceso múltiple por detección de portadora evitando colisiones (CSMA/CA). En una red con protocolo CSMA/CA, cuando una estación desea enviar un mensaje, comprueba si la línea está libre y una vez que lo ha confirmado, indica que tiene intención de transmitir. Si hay varias estaciones esperando, el orden en que van a

transmitir se determina por medio de un esquema ya fijado (algoritmo). En las redes con protocolo CSMA/CA (al igual que en la reunión el moderador cede la palabra a la persona de más alto rango) cada estación tiene una prioridad y, de esta forma, la primera en acceder a la línea será la estación que tenga la prioridad más alta. En cuanto la estación termina de transmitir el mensaje, la estación que goce de mayor prioridad accede a la línea, y así sucesivamente. El protocolo CSMA/CA tiene un inconveniente: la estación receptora (estación a la que va dirigido el mensaje) tiene la máxima prioridad para transmitir. Si la estación receptora envía un mensaje a la estación emisora original, esta última será la que disponga de la máxima prioridad. De esta forma, las dos estaciones pueden bloquear el acceso del resto de las estaciones a la red.

1.2.2. Polling (llamada selectiva)

Ilustrativamente es lo que ocurre en un aula de escuela, donde la estación central, o estación principal, actúa como un maestro de escuela tomando la lección a sus alumnos: pregunta a una de las estaciones (alumnos) y cuando ésta ha respondido, pasa a la siguiente, repitiéndose el ciclo una y otra vez. El protocolo Polling consiste en llamar a una estación para que, transmita o se disponga a recibir un mensaje. Este método requiere un control centralizado de todas las estaciones de la red (Ver figura 1.9). Las redes que utilizan el protocolo Polling tienen dos tipos de estaciones: la estación principal y las estaciones secundarias conectadas a ella. Cada estación secundaria dispone de un área de almacenamiento temporal (o buffer). Cuando una estación secundaria desea transmitir un mensaje, lo envía a este buffer, donde permanece hasta que la estación central pide que le sea transmitido. En la red que utiliza el protocolo de Polling la estación central o principal llama a las estaciones secundarias de una en una para determinar si hay alguna que tenga un mensaje para transmitir. Si la respuesta es afirmativa, se autoriza a la estación secundaria para que lo transmita inmediatamente, o se le

asigna un determinado tiempo para que lleve a cabo la transmisión. Este tiempo viene determinado por ciertos parámetros establecidos en el sistema. Si la estación secundaria no tiene mensajes para transmitir, ha de contestar mediante un pequeño mensaje de control. En algunas redes de computadoras que utilizan este protocolo, cuando la estación secundaria no tiene datos para transmitir, pasan el control a la siguiente estación secundaria, ahorrando así algo de tiempo y haciendo que el rendimiento sea más alto. Cada vez que la estación central o principal llama a una secundaria, la estación principal ha de esperar a que la secundaria le responda; cuando ésta responde, la principal llama a la siguiente estación secundaria, y así se repite el proceso continuamente. La estación central decide qué estación tiene acceso a la red en un determinado momento.

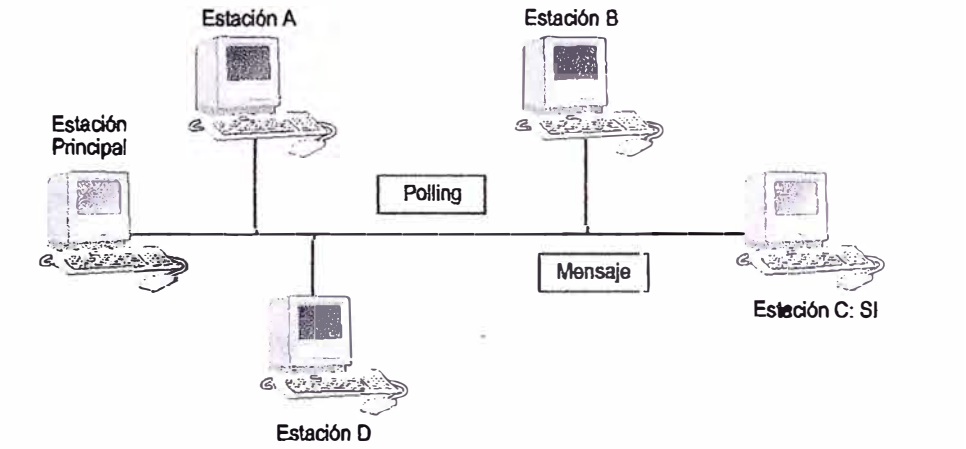


Figura 1.9. Protocolo de Polling (llamada selectiva).

En este tipo de redes que utilizan el protocolo de Polling el mensaje puede tomar dos direcciones para ir de la estación que emite el mensaje a la que lo recibe:

a) Todos los mensajes han de pasar por la estación central, la cual los reenvía a las estaciones de destino.

b) Cada estación puede enviar los mensajes directamente a su destino.

Pero siempre, la transmisión de datos sólo se realiza bajo la dirección de la estación central.

El protocolo de Polling presenta algunas variaciones.

- El protocolo básico establece que todas las estaciones tienen la misma prioridad, pero esto no siempre es así.
- En algunas redes de computadoras que emplean el protocolo Polling, las estaciones que tienen mucha actividad gozan de una prioridad más alta y han de ser llamadas a intervalos más cortos que el resto.
- En otras redes de computadoras que emplean el protocolo Polling , la estación central no llama a las estaciones que no están activadas.
- Y en otras redes de computadoras que emplean el protocolo Polling, la frecuencia de llamada a las distintas estaciones depende del nivel de actividad que hayan tenido en un determinado periodo de tiempo.

Es fácil darse cuenta que las redes que utilizan el protocolo Polling mantienen un mayor control sobre la red que los protocolos de contienda.

1.2.3. Token passing (paso de testigo)

En las redes que utilizan este tipo protocolo, el protocolo paso de testigo (token passing) hace circular continuamente un testigo o grupo de bits que habilita a la estación que lo posee el derecho a utilizar la línea. Únicamente la estación que posee el testigo puede enviar un mensaje a través de la red. Debe observarse que en las redes que utilizan este tipo de protocolo el control de la red no está centralizado. El testigo es un grupo de bits que contiene cierta información con una estructura predeterminada. Esta información está compuesta por una cabecera, un campo de datos y un campo final (Ver figura 1.10)

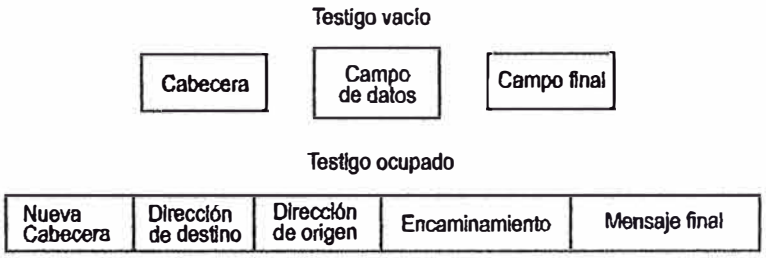


Figura 1.10. Estructura del testigo.

Cuando una estación de la red que desea transmitir recibe un testigo vacío, dicha estación inserta los datos y la información necesaria para que el mensaje llegue a la estación destino, y después envía el testigo a través de la red. La longitud máxima de los mensajes que puede enviar una estación con el testigo está preestablecida. Si no tiene nada para transmitir, pasa el testigo a la siguiente estación de la red. La figura 1.11 muestra el funcionamiento del protocolo de paso de testigo. El protocolo Token Passing presenta algunas variaciones; en la mayoría de las redes de paso de testigo, el testigo pasa de una estación a la estación que hay a continuación, pero en algunas implementaciones el testigo puede pasar de una estación a otra en un orden previamente establecido (no necesariamente a la estación que hay a continuación). En tales implementaciones, la estación que posee el testigo conoce la dirección de la siguiente estación a la que ha de enviarlo.

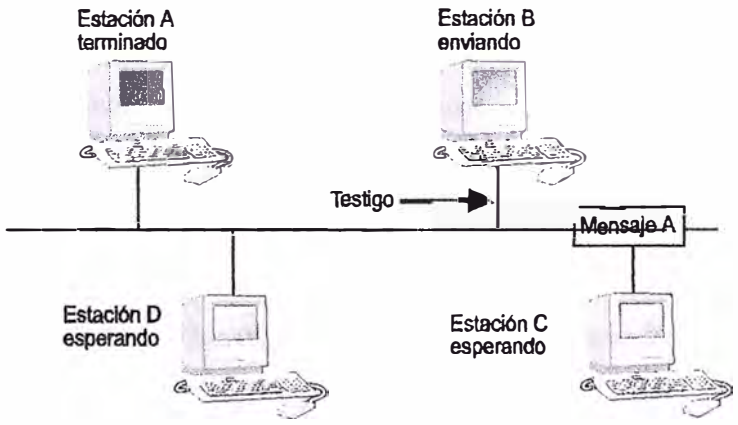


Figura 1.11. Protocolo Token Passing (Paso de testigo).

Todas las estaciones de la red leen la dirección que contiene el testigo; si no coincide con la de la estación que lo ha recibido, se pasa a la siguiente. Al llegar a su destino, la estación receptora lee el mensaje, pone una marca en el testigo indicando que lo ha aceptado o denegado, y vuelve a hacerlo circular por la red hasta hacerlo llegar a la estación emisora que ha enviado el mensaje. Cuando el testigo llega al emisor original, éste lee y borra el mensaje, lo marca como vacío y lo envía a la siguiente estación. El emisor puede guardar el mensaje y compararlo con el mensaje original para comprobar si se ha recibido correctamente. De esta

forma, se implementa un método de seguridad en la red. Si la estación emisora vuelve a recibir el mensaje sin la marca de "recibido" o con una marca que indique que la estación receptora no lo ha recibido correctamente, vuelve a transmitir el mensaje. Las redes de computadoras que utilizan el protocolo paso de testigo ofrecen un control muy estricto sobre toda la red. La mayor ventaja de este protocolo es que se elimina toda posibilidad de colisiones entre mensajes.

1.3. Factores de evaluación de protocolos

El tema de especificación de protocolos es amplio y profundo, y llenaría varios libros de fórmulas matemáticas muy complejas; en esta parte no vamos a tratar todas esas fórmulas, sino que nos centraremos a descripciones algo más generales, las fórmulas lo dejaremos para el siguiente capítulo. Para seleccionar un protocolo hay que tener en cuenta ciertos factores, los más importantes son:

- ***Longitud del mensaje***: Es la extensión de los mensajes que se van a transmitir
- ***Volumen de tráfico***: Es el número de mensajes que se pueden pasar.
- ***Tamaño de la red***: Es el tamaño que puede tener la red que va a usar el protocolo.
- ***Rendimiento***: Son las condiciones en que el protocolo funcionará bien.
- ***Carga***: Es la capacidad necesaria para pasar mensajes de control.
- ***Espera de acceso***: Se refiere a si el protocolo proporciona o no estados de espera antes de que la estación pueda acceder a la red.
- ***Fallos de estaciones***: Se refiere a la repercusión si falla una estación en la red.
- ***Expansión***: Se refiere a la posibilidad del protocolo de acoplar fácilmente estaciones adicionales.

1.3.1. Factores de evaluación del protocolo de contienda simple

Las redes que utilizan el protocolo de contienda simple dependen de la disponibilidad del medio de transmisión y del índice de colisiones. Dichas redes se caracterizan por lo siguiente:

- ***Longitud del mensaje.*** En este tipo de redes los mensajes se dividen en un número pequeño de paquetes para reducir así la cantidad de datos que se han de retransmitir después de las colisiones. Normalmente, el mensaje suele ser también bastante corto.

  Volumen de tráfico. Las redes con protocolos de contienda simple están pensados para redes con un tráfico bastante reducido; es decir, unas pocas estaciones que intentan hacer uso de la red al mismo tiempo. Un volumen de tráfico bajo implica que el número de estaciones conectadas es también bajo.
- ***Tamaño de la red.*** Cuanto más grande es la red, mayor es la posibilidad de que se produzcan colisiones. Las redes que utilizan protocolos de contienda simple tienen una limitación importante: el tiempo que tarda la señal en llegar a su destino y lo que tarda en volver la señal de acuse de recibo ("recibido" o "no recibido").
- ***Rendimiento.*** Las redes que utilizan protocolos de contienda simple son más eficaces cuando trabajan con cargas entre bajas y medias. En estas condiciones el rendimiento es excelente. Con cargas muy altas, la red tiende a ser muy inestable y los tiempos de servicio aumentan considerablemente.
- ***Carga.*** Las redes que emplean el protocolo de contienda simple conllevan una gran carga, debido a las colisiones (lo cual implica la retransmisiones del paquete) y a la necesidad de acusar recibo del paquete.
- ***Espera de acceso.*** En las redes que usan este protocolo la espera para acceder a la red está generalmente en un término medio, pero esto siempre depende del tráfico. Las esperas en condiciones de mucho tráfico pueden ser bastante superiores de lo que la carga actual haría suponer.
- ***Fallos de estaciones.*** Como en las redes que emplean el protocolo de contienda simple el funcionamiento de la red no depende de que haya o no estaciones en la red, el fallo

de una estación no afecta para nada el resto. Es muy difícil que el fallo de una sola estación pueda parar toda la red. En caso de que sucediese esto, el problema estaría en otra parte.

- ***Expansión.*** En las redes que usan el protocolo de contienda es relativamente fácil añadir una nueva estación o periférico, ya que lo único que hace falta es que ésta reconozca su dirección.

1.3.2. Factores de evaluación del protocolo Polling

Las redes que utilizan el protocolo Polling se caracterizan por lo siguiente:

- ***Longitud del Mensaje.*** En las redes que usan el protocolo Polling la longitud de los mensajes tiende a ser superior a la de las redes que utilizan los protocolos de contienda, pero si todas las estaciones transmiten mensajes muy largos, los tiempos de espera pueden ser muy altos.
- ***Volumen de tráfico.*** Las redes con protocolo Polling soportan un volumen de tráfico entre normal y alto, limitado principalmente por la obligación de esperar a que se reciba la autorización de transmitir. De esta forma se evitan los conflictos que se producen con los métodos de contienda y un gran número de estaciones puede compartir la línea.
- ***Tamaño de la red.*** En redes con protocolo Polling la distancia entre estaciones y la longitud total de la red sólo está limitadas por el medio de transmisión. Como en cualquier otra red, cuanto mayor es la distancia, más tiempo tardan los mensajes en ir de la estación emisora a la estación receptora.
- ***Rendimiento.*** Las redes de computadoras que utilizan el protocolo Polling trabajan mejor con una carga media. Con cargas altas, las esperas pueden resultar terriblemente largas. Este protocolo no es muy eficaz en redes con muy poco tráfico, puesto que la

mayor parte del tiempo se emplea en emitir señales de llamada y en recibir los "acuses de recibo".

- ***Carga.*** En las redes con protocolo Polling el tiempo empleado en llamar a las estaciones y en recibir las respuestas de éstas es una parte importante de la capacidad total de la red. En algunas redes de computadoras, la estación central (servidor dedicado) no se puede utilizar como estación de trabajo. En otras la estación central (servidor no dedicado) puede funcionar como estación de trabajo.
- ***Espera de acceso.*** En las redes que utilizan el protocolo Polling en general, los tiempos de espera son relativamente largos. La mayoría de los sistemas llaman una vez a cada estación en cada ciclo. Si la red es muy grande, el tiempo de espera puede ser grande.
- ***Fallos de estaciones.*** En las redes de computadoras con protocolo Polling el fallo de una estación secundaria no afecta al resto de la red. El sistema no tiene en cuenta la estación "averiada", es decir, no responde a la estación central. Sin embargo, cuando la estación "averiada" es la estación central, toda la red deja de funcionar.
- ***Expansión.*** En las redes Polling para poder ampliar la red es necesario avisar a la estación central que se va a añadir un nuevo dispositivo, y se ha de modificar el orden de llamada. Por tanto, en este tipo de red, añadir un nuevo dispositivo resulta algo más complicado que en las redes que utilizan métodos de contienda.

1.3.3. Factores de evaluación del protocolo Paso de testigo

Las redes que utilizan el protocolo Paso de testigo se caracterizan por lo siguiente:

- ***Longitud del mensaje.*** Las redes que usan el protocolo de paso de testigo pueden emplear mensajes muy largos. Puesto que los mensajes incluyen el testigo, se pueden trasmitir diversos tipos de datos.

- ***Volumen de tráfico.*** El tráfico en una red que emplee el método de paso de testigo puede ser bastante alto. Dado que cada estación puede emitir un solo mensaje a la vez, la disponibilidad de la red es bastante equitativa.
- ***Tamaño de la red.*** El tamaño de las redes que emplean el método de paso de testigo están limitadas principalmente por el medio de transmisión, no por el protocolo.
- ***Rendimiento.*** Las redes que emplean el método de paso de testigo ofrecen un buen rendimiento en casi cualquier condición. Con un tráfico entre normal y alto, el rendimiento general se puede calificar de excelente.
- ***Carga.*** Comparado con otros protocolos, las redes de paso de testigo emplean una gran cantidad de tiempo en la transferencia y control del testigo.
- ***Espera de acceso.*** Los tiempos de espera de las redes de paso de testigo suelen ser bastante constantes y se pueden calcular fácilmente con determinado volumen de tráfico. Con mucho tráfico, los tiempos de espera resultan aceptables, pero en condiciones normales de tráfico el tiempo es bastante corto.
- ***Fallos de estaciones.*** En las primeras redes que utilizaron el método de paso de testigo, el fallo de una de las estaciones bloqueaba por completo la red. Actualmente, las redes modernas que utilizan este protocolo han eliminado este problema.
- ***Expansión.*** Ampliar una red de paso de testigo es un proceso bastante complicado que puede hacer necesario reconfigurar completamente la red para incluir nuevas estaciones y definir una nueva secuencia de circulación del testigo. El proceso de expansión de la red puede inutilizarla durante períodos muy largos.

1.4. El modelo de referencia ISO y la norma IEEE 802

La necesidad de poner cierto orden en el proceso de diseño e implementación de las redes de computadoras, ha servido para el surgimiento de organizaciones que cumplan este objetivo.

Entre ellas tenemos la Organización Internacional de Normalización - ISO (International Standards Organization) y el Instituto de Ingenieros Eléctricos y Electrónicos - IEEE (Institute of Electrical and Electronics Engineers).

El Modelo ISO

Las redes de computadoras LAN y WAN surgieron para hacer posible compartir de forma eficiente los recursos informáticos (hardware, software y datos) de los usuarios. Esos recursos son sistemas heterogéneos. Los equipos tienen características diferentes de acuerdo cada fabricante, utilizan y ejecutan software con características específicas y distintas para las aplicaciones deseadas por los usuarios, y manipulan y producen datos con formatos incompatibles. También equipos idénticos de un mismo fabricante, que se integran en aplicaciones distintas, pueden presentar características heterogéneas. Esa heterogeneidad de los sistemas beneficia al usuario, que no está así limitado aun único tipo de sistemas para sus distintas aplicaciones. De esta manera, el usuario puede seleccionar el sistema que mejor se adapte a las condiciones de aplicación que le interesen y al presupuesto disponible. Por otro lado tal heterogeneidad dificulta considerablemente la interconexión de equipos de fabricantes diferentes. Hace algún tiempo los fabricantes dieron soluciones para la interconexión de sus propios equipos. La solución particular de cada fabricante la denominaron Arquitectura de Red, que es sinónimo de "un conjunto de convenios" para la interconexión de sus propios equipos. Así el fabricante Digital Equipment Corporation (DEC) habló de su "Digital Network Architecture" - DNA, e IBM habló de su "System Network Architecture" – SNA, etc.

La dificultad generada por la heterogeneidad se acentúa en el caso de la interconexión con equipos de otros fabricantes y/o en la interconexión de redes distintas. La interconexión de redes, contribuye a hacer más difícil el problema, ya que pueden haber redes diferentes con servicios de transmisión diferentes, que requieran interfaces diferentes, ... Entonces surge la

necesidad de una manera por la cual el problema de las heterogeneidades no haga inviable la interconexión de sistemas distintos. La incompatibilidad de equipos y/o redes fue inicialmente resuelto a través del uso de convertidores. El convertidor interpreta la información originaria de uno de los sistemas y transfiere la información al sistema destinatario, traduciéndola a una forma inteligible (compatible) para el destinatario. Pero el uso de los convertidores es deficiente. Son lentos e inadecuados para solucionar incompatibilidades a nivel de aplicaciones ya que sería necesario su integración en el propio sistema operativo de cada uno de los dos equipos involucrados. En 1977 la Organización Internacional de Normalización - ISO (International Standards Organization) vio la necesidad de normalizar la interconexión de sistemas heterogéneos y creó el subcomité 16 (SC16), la misma que a mediados de 1978 sugirió mediante la RM-OSI (The Reference Model of Open Systems Interconection) una arquitectura estratificada en categorías para que se desarrollase a partir de allí una norma para el modelo de arquitectura que sirviera de soporte para el desarrollo de protocolos normalizados. A finales del año 1979 el Comité Técnico 97 de la OSI (CT97) acató el RM-OSI como base para el desarrollo de protocolos, norma para la interconexión de sistemas abiertos. En mayo de 1983, el RM-OSI fue oficialmente aprobado por la ISO como una norma internacional para la interconexión de sistemas abiertos a través el documento ISO 7498. El Comité Consultatif International Telégraphique et Teléphonique (CITT) acató también el RM-OSI a través de la recomendación X200. La ISO aprobó el modelo de referencia de interconexión de sistemas abiertos (The Reference Model of Open Systems Interconection). Interconexión de sistemas abiertos significa el intercambio de información entre terminales, computadoras, redes, procesos, personas, etc. Este modelo define dónde se han de efectuar las tareas, pero no como se han de efectuar. No especifica servicios ni protocolos, pero si proporciona una base común para coordinar el desarrollo de estándares dirigidos a la conexión entres sistemas.

En el entorno OSI, un sistema es un conjunto de computadoras asociados con un software, periféricos, terminales, aperadores humanos, procesos físicos, medios para la transferencia de información, etc. capaz de procesar información.

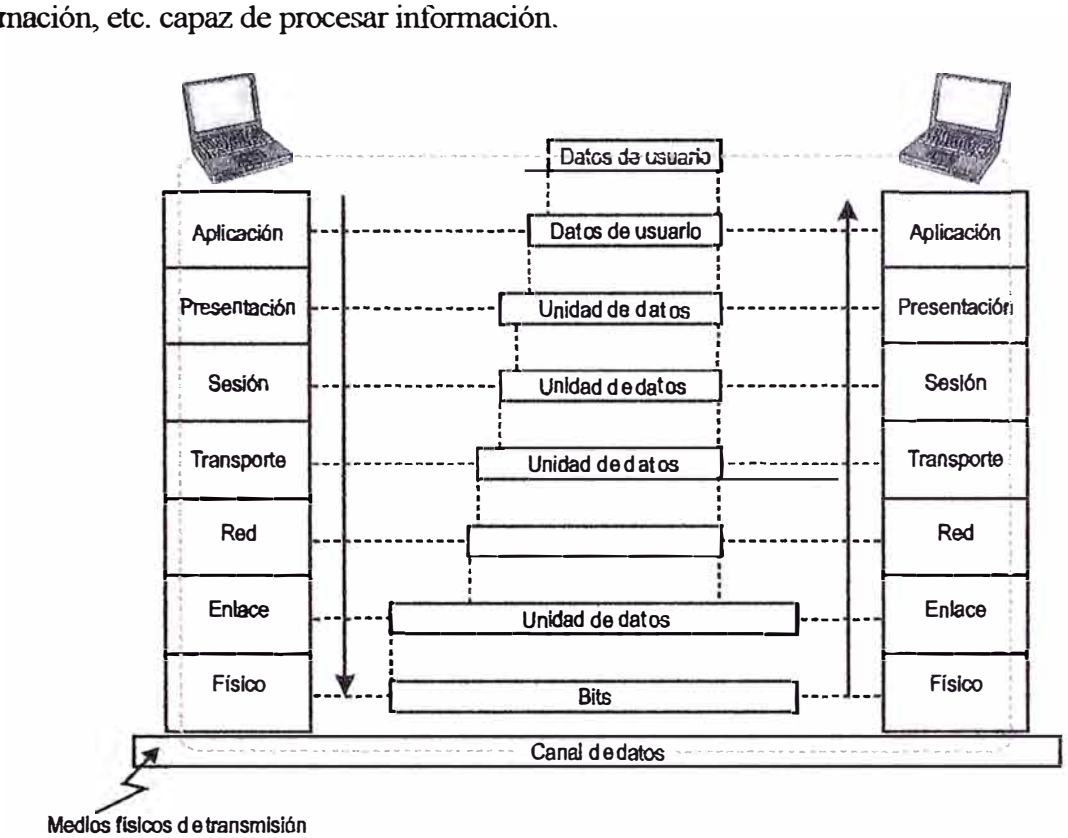


Figura 1.12. Transformación de un mensaje.

El modelo OSI está compuesto por un conjunto de siete niveles (Ver Figura 1.12), los mismos que están separados por interfaces cuyas funciones están bien definidas. Los niveles adyacentes se comunican entre sí por medio de un interfaz común. Cada nivel dispone de un conjunto de servicios para el nivel superior e inferior, las que están bien definidas, pero el formato utilizado para transferir datos entre niveles no lo está. Los protocolos asociados a los niveles 1, 2, 3, y 4 son iguales para todos los sistemas. Los protocolos de los niveles 5, 6 y 7 dependen del sistema.

Nivel 1: Nivel de medios físicos

Este nivel define las características mecánicas, eléctricas, funcionales y de procedimientos para activar, mantener y desactivar conexiones físicas para la transmisión de bits entre las

entidades del nivel de enlace de datos, a través de sistemas intermedios (interfaces), realizando cada uno de ellos transmisiones de bits a través del nivel físico. El nivel físico está pensado para atender a una gran variedad de medios físicos y procedimientos de control. Los medios físicos incluyen los cables y los conectores. Los procedimientos de control incluyen los métodos de transmisión, las computadoras y los equipos de comunicaciones. En resumen este nivel se encarga de la transmisión de cadenas de bits no estructurados sobre el medio físico, está relacionada con las características mecánicas, eléctricas, funcionales y de procedimiento para acceder al medio físico.

Nivel 2: Nivel de enlace de datos

Este nivel tiene el propósito de proveer los medios funcionales y de procedimiento para activar, mantener y desactivar una o más conexiones de enlace de datos entre unidades del nivel de red. Las funciones de este nivel son: a) Detectar y corregir errores en el nivel de medios físicos. b) Proporcionar un nivel de red con la capacidad de pedir el establecimiento de circuitos de datos en el nivel de medios físicos., es decir con la capacidad de controlar el cierre de circuitos. Este nivel establece y mantiene comunicaciones entre los usuarios. Es el responsable de mantener un canal sin errores, detectando y corrigiendo los que se puedan producir. Los protocolos relacionados con este nivel son los encargados del formato de los bloques de datos, de los códigos de dirección, de la detección y recuperación de errores y del orden de los datos transmitidos. En resumen el nivel de enlace de datos proporciona un servicio de transferencia de datos seguro a través del enlace físico; envía bloques de datos (tramas) llevando a cabo la sincronización, el control de errores y de flujo necesarios.

Nivel 3: Nivel de red

El nivel de red suministra los medios para establecer, mantener y terminar conexiones de red entre sistemas que contienen entidades de aplicación comunicantes. Suministra también los

medios funcionales y de procedimiento para la transferencia de la información, a través de conexiones de red, entre dos entidades del nivel de transporte. Este nivel establece y mantiene transmisiones entre sistemas. El nivel de red es el encargado de transmitir los datos por toda la red. En él los datos se convierten en paquetes y se envían a su destino. Los protocolos relacionados con el nivel de red se encargan de la administración y gestión de los datos, emisión de mensajes de estado, regulación del tráfico de la red y reparto del trabajo entre las distintas unidades de interfaz y la estación central. Se encarga también de que todos los paquetes lleguen correctamente a su destino. En resumen este nivel proporciona independencia a los niveles superiores respecto a las técnicas de conmutación y de transmisión utilizadas para conectar los sistemas; es responsable del establecimiento, mantenimiento y cierre de las conexiones.

Nivel 4: Nivel de transporte

El propósito de este nivel es proporcionar un servicio de transferencia transparente de datos (de fin a fin) entre entidades del nivel de sesiones. El nivel de red se encarga de todos los detalles de realización de paquetes, rotación y transferencia de paquetes de una subred a otra. Por tanto la complejidad de las funciones en el nivel de transporte, que son responsables de la calidad del servicio ofrecido, depende de la calidad del servicio del nivel de red. Si la conexión ofrecida por el nivel de red es fiable y económica, las funciones necesarias en el nivel de transporte quedarán proporcionalmente reducidas. Este nivel es el encargado de la transferencia de los datos entre la estación emisora y la estación receptora, y también es el encargado de mantener el flujo de la red. Su función básica es aceptar datos del nivel de sesión, dividirlos en mensajes y pasar éstos al nivel de red. Comprueba también si los mensajes llegan correctamente a su lugar de destino. El nivel de transporte representa la línea de separación entre la transmisión de datos y el proceso de datos. Los protocolos de este nivel controlan la

distribución de los mensajes y evitan que se pierdan o se dupliquen los mensajes. En resumen este nivel proporciona seguridad, transferencia transparente de datos entre los puntos finales; proporciona además procedimientos de recuperación de errores y control de flujo origen-destino.

Nivel 5: Nivel de Sesión

El objetivo del nivel de sesión es el organizar, sincronizar y se encarga del diálogo entre los usuarios; es decir, es el interfaz entre el usuario y la red. También se encarga de gestionar la transferencia de datos entre entidades del nivel de presentación de comunicantes. Para ello el nivel de sesión suministra servicios para el establecimiento de una conexión de sesión entre dos entidades de presentación, a través del uso de una conexión de transporte.

Los servicios del nivel de sesión se clasifican en dos categorías:

a) Servicio de Administración de sesiones, que une dos entidades para una relación y que más tarde los desune. Ejemplo de unión: Login, ejemplo de desunión: Logoff.
b) Servicio de diálogo de sesión, que controla una transferencia de datos, delimita y sincroniza operaciones con los datos entre dos entidades de presentación. Por ejemplo se puede abrir una conexión de sesión para transferir informaciones en half dúplex, full dúplex, etc.

Los servicios ofrecidos por el nivel de sesión son los primeros a realizarse con las aplicaciones propiamente dichas, después del servicio de comunicación. Cada usuario de la red ha de dirigirse al nivel de sesión para establecer una conexión con otra estación. Una vez hecha la conexión, el nivel de sesión sincroniza el diálogo y se encarga del intercambio de datos. Los protocolos de este nivel incluyen reglas para establecer y dar por finalizadas las conexiones, verificando al mismo tiempo si está teniendo lugar la comunicación adecuada y comunicando la red con el sistema operativo. En resumen el nivel de sesión proporciona el control de la

comunicación entre las aplicaciones; establece, gestiona y cierra las conexiones (sesiones) entre las aplicaciones cooperadoras.

Nivel 6: Nivel de Presentación

Este nivel traduce la información del formato de máquina a un formato que pueda entender el usuario. Una de las funciones más importantes es la traducción de los distintos formatos de archivo y de terminal, y de los diferentes sistemas de codificación (por ejemplo, de ASCII a EBCDIC). El nivel de presentación realiza los servicios que pueden ser seleccionados por el nivel de aplicaciones para la interpretación de la sintaxis de los datos transmitidos. Esos servicios gestionan la entrada, transferencia, presentación y control de datos estructurados. El nivel de presentación resuelve los problemas de diferencia de sintaxis entre sistemas abiertos comunicantes. Las aplicaciones en el entorno OSI pueden comunicarse a través de los servicios del nivel de presentación sin costes excesivos que dependen de la variación de interfaces, transformaciones o modificaciones de las propias aplicaciones. En resumen este nivel proporciona a los procesos de aplicación independencia respecto a las diferencias en la representación de los datos (sintaxis).

Nivel 7: Nivel de Aplicación

Este es el nivel más alto de le RM-OSI, y todo comenzó a partir de este nivel. Todas las otras capas existen para dar soporte a este nivel. Los servicios de este nivel son utilizados por los propios usuarios en el ambiente OSI. El propósito de este nivel es servir de ventana, entre los usuarios comunicante en un entorno OSI, a través del cual se produce toda la transferencia de información significativa para esos usuarios. Cada usuario viene representado para los demás por su entidad de aplicación correspondiente. Este nivel se encarga del intercambio de información entre el usuario y el sistema. Los protocolos de este nivel se ocupan del soporte de los programas de aplicación, como claves de acceso, transferencia de archivos, etc. Para el

usuario aparentemente los mensajes van desde la estación de origen a la estación de destino directamente, pero en realidad este proceso es mucho más complicado. Cuando el usuario envía un mensaje, este pasa por un protocolo hasta llegar al interfaz del nivel que tiene más abajo. Este nivel envuelve a su vez el mensaje en su propio protocolo antes de enviarlo. Según va bajando, el mensaje va siendo rodeado por el protocolo propio del nivel. El mensaje ha de pasar por todos los niveles desde su punto de origen hasta el canal de datos. Una vez que el mensaje está en el canal de datos, se pasa hacia arriba por todos los niveles. Según va subiendo, los niveles van desprendiéndolos de los protocolos correspondientes y el proceso se invierte hasta que vuelve a aparecer el mensaje original. Ver figura 1.12.

Nivel 7 Aplicación	Funciones de usuario final y aplicación final, como transferencia de archivos (FTAM), servicio a terminales virtuales (VTP) y correo electrónico (X.400)
Nivel 6 Presentación	Traducción de datos para ser usados por el nivel 7, como conversión de protocolo, descompresión de datos, codificación y expansión de comandos gráficos.
Nivel 5 Sesión	Ofrece el establecimiento de una conexión de sesión entre dos entidades de presentación para soportar el intercambio ordenado de datos,
Nivel 4 Transporte	Transferencia transparente de datos entre entidades de sesiones que liberan al nivel de sesión e la necesidad de preocuparse por la confiabilidad y la integridad de los datos.
Nivel 3 Red	Ofrece el medio para establecer, mantener y poner fin a conexiones de redes entre sistemas abiertos, en particular enviando funciones a través de múltiples redes.
Nivel 2 Enlace de datos	Define la estrategia de acceso para compartir el medio físico, incluyendo los aspectos del enlace de datos y acceso a los medios.
Nivel 1 Físico	Definición de las características eléctricas y mecánicas de la red.

Figura 1.13. Modelo de referencia ISO para la interconexión de sistemas abiertos.

En resumen este nivel proporciona el acceso al entorno OSI para los usuarios y también proporciona servicios de información distribuida. La Figura 1.13 muestra los niveles del RM-ISO. Los protocolos de los tres primeros niveles (1, 2, y 3) son considerados como protocolos de

bajo nivel, y los protocolos de los niveles restantes (4, 5, 6 y 7) son considerados como protocolos de alto nivel.

La norma IEEE 802

Una red local (Según el Proyecto 802 del IEEE) es un sistema de comunicaciones que permite a varios dispositivos independientes comunicarse directamente entre sí, dentro de una determinada zona, a través de una línea de comunicaciones a velocidades de transmisión de datos moderadas. La IEEE define claramente el número y tipos de dispositivos que se pueden conectar. A continuación resumimos los requerimientos más importantes:

- ***Tamaño***: La red deberá soportar al menos 200 dispositivos y deberá poder cubrir un mínimo de 2 Km.
- ***Funciones de transmisión de datos***: Las funciones deberán incluir transferencia de archivos, acceso (manipulación) a archivos y acceso a bases de datos, soporte de terminales (inteligentes, no inteligentes de alta velocidad, etc.) correo electrónico y transmisión de voz.
- ***Dispositivos conectados***: Los dispositivos interconectados por la red deberán incluir computadoras y terminales, dispositivos de almacenamiento, impresoras, plotters, equipo de control, puentes (bridges) y puertas (gateways) a otras redes, teléfonos, cámaras de video y monitores, fotocopiadoras y unidades facsímile.
- ***Servicios***: La red deberá permitir que coexistan varios procesos.
- ***Ampliación***: Añadir o suprimir dispositivos ha de ser sencillo. Los cambios habrán de causar las mínimas molestias a los usuarios. Las esperas por esta causa no habrán de durar más de un segundo.
- ***Reparto de recursos***: Cuando los dispositivos necesiten compartir los recursos de la red, especialmente la línea, el reparto ha de ser equivalente para todos los dispositivos, incluso en condiciones de mucho tráfico.

- ***Velocidad de transmisión***: Los datos se deberán poder transmitir a una velocidad comprendida entre 1 y 20 Megabits por segundo.
- ***Fiabilidad***: La red ha de ser muy fiable. A lo más, sólo puede pasar desapercibido un error por año.

Al comenzar a desarrollar el estándar, los integrantes del Proyecto 802 reconocieron que no existía ninguna tecnología que reuniese todos los requisitos, todo dependía de la aplicación y de las necesidades. Debido a esto, el Proyecto 802 fue dividido en varios comités diferentes; los más importantes son los que a continuación detallamos:

- ***Comité 802.1***: Estándar del interfaz de nivel superior. El comité 802.1 no desarrolla estándares, sino que se encarga de temas comunes a todos los demás comités, como envío de mensajes, gestión de la red, etc.
- ***Comité 802.2***: Estándar del control de enlace lógico. El comité 802.2 se encarga del desarrollo de los estándares necesarios para que se establezca comunicación entre dos dispositivos.
- ***Comité 802.3***: Bus CSMA/CD. El Comité 802.3 se encarga de desarrollar una red en bus que utiliza el método de contienda CSMA/CD. Los estándares propuestos por este comité son prácticamente iguales a las especificaciones de Ethernet diseñado por Xerox; es decir, una red que transmita a 10 Megabits por segundo y que permita a un máximo de 1000 dispositivos compartir una línea formada por un cable coaxial de banda base.
- ***Comité 802.4***: Bus de paso de testigo. Este comité se ha encargado de definir una red lógica en anillo, de forma que se pueda usar el protocolo de paso de testigo.
- ***Comité 802.5***: Anillo de paso de testigo. El comité 802.5 ha definido una red de paso de testigo que usa una topología en estrella para acceder secuencialmente a las estaciones. Este comité ha desarrollado versiones de banda base y ancha en colaboración con IBM.

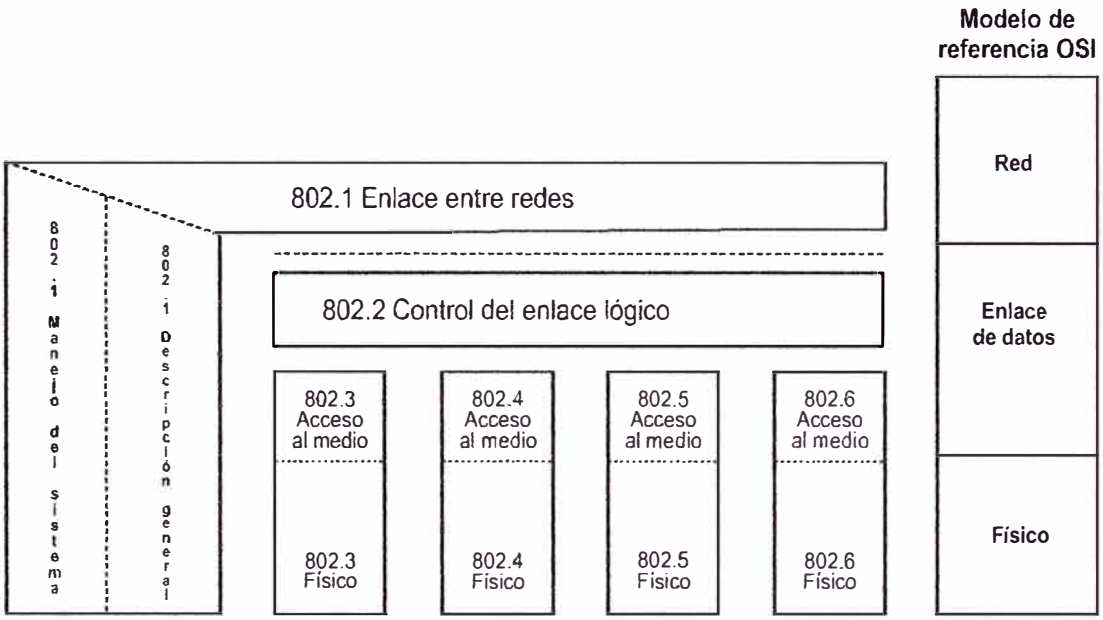


Figura 1.14. Relación entre los niveles de la ISO y la IEEE.

Existen algunos otros comités, pero los más importante son los que acabamos de enumerar. En la Figura 1.14 se muestra la relación entre el modelo de referencia OSI y la IEEE. En la Figura 1.15 se muestra de forma objetiva los sistemas operativos de red y los diferentes niveles de la ISO. En la Figura 1.16 se muestra los niveles de la ISO, la IEEE y sus protocolos.

OSI	MAP	IBM OS/2	IBM DOS	Banyan		Novell	3COM
Aplicación	ACSE, FTAM, MMS, Servicios de directorio	Aplicación	Aplicación	Appl. de VINES	OSI Appl.	Aplicación	Aplicación
Presentación	ISO 8822/8823	Servicios centrales de servidores de LAN Solicitante	NET-BIOS APPC			Conductos con nombres asignados de TLI	Conductos con nombres asignados de 3+Open
Sesión	ISO 8326/8327	NET-BIOS			Sesión de ISO		
Transporte	ISO 8072/8073			ISO 8072/8073		OSI TCP/IP IPX/SPX IBM SNA NETBEUI ATP	OSI TCP/IP XNS IBM SNA NETBEUI ATP
Red	ISO 8348/8473			ISO 8473 9542	X.25		
Enlace de datos	IEEE 802.2 LLC IEEE 802.4 MAC	802.2 LLC 802.3 MAC 802.5 MAC		802.2 802.3 802.5	LAPB	Interfase abierta de datos variadas	Interfases de enlace de datos variadas
Físico	IEEE B802.4 Banda ancha o banda de portador	IEEE 802.3, 802.5 Banda ancha o Otro		802.3 802.5	X.21	Físico	Físico

Figura 1.15. Los sistemas operativos de red y los niveles de la ISO.

NIVEL	NOMBRE DEL ESTÁNDAR	NÚMERO
Nivel 7 Aplicación	Arquitectura de documentos de oficina (ODA) Transferencia, acceso y manjeo de archivos (FTAM) Terminal virtual Manejo de la red Especificación del mensaje de manufactura Procesamiento distribuido de transacciones Archivado y recuperación de documentos Protocolo de acceso a base de datos distantes Transferencia y manipulación de trabajos Protocolo de transferencia, acceso y manipulación de documentos El directorio Servicio de manejo de mensajes Elementos de servicios comunes: Elementos de servicio de control de asociaciones (ACSE) Elementos de servicio de transferencia confiable	ISO 8613 ISO 8571 ISO 9040 ISO 9595/96 ISO 9506 ISO 10026 SC 18N 1264/5 ISO 9576 ISO 8832/33 CCITT T.431/33 CCITTT X.500, ISO9594 CCITT X.400, ISO 10020/21 ISO 9066 ISO 9072
Nivel 6 Presentación	Protocolo de presentación orientado a conexiones Protocolo sin conexiones	ISO 8823 ISO 9576
Nivel 5 Sesión	Protocolo de sesión orientado a conexiones Protocolo sin conexiones	ISO 8237 ISO 9548
Nivel 4 Transporte	Protocolo de transporte orientado a conexiones Protocolo sin conexiones	ISO 8073 ISO 8602
Nivel 3 Red	Protocolo sin conexiones X.25 Protocolo de intercambio de sistema final a sistema intermedio Propuesta de cómo utilizar ISDN en OSI y OSI en ISDN.	ISO 8473 ISO 8208 ISO 9542 ISO 9574
Nivel 2 Enlace de datos	Control del enlace lógico (LLC) Control de acceso a los medios (MAC) - CSMA/CD - Token Bus - Token Ring Interfase de datos distribuido por fibras	IEEE 802.2, ISO 8802/2 IEEE 802.3, ISO 8802/3 IEEE 802.4, ISO 8802/4 IEEE 802.5, ISO 8802/5 ISO 9314
Nivel 1 Físico	CSMA/CD Token Bus Token Ring Interfase de datos distribuido por fibras Anillo ranurado	IEEE 802.3, ISO 8802/3 IEEE 802.4, ISO 8802/4 IEEE 802.5, ISO 8802/5 ISO 9314 ISO 8802/7

Figura 1.16. Los niveles de la ISO, la IEEE y sus protocolos.

1.5. Fundamento matemático y estadístico

En este apartado se revisan algunas de las herramientas analíticas que se usan en los temas desarrollados. Las herramientas enfocadas son los Modelos de Redes, la teoría de probabilidades, variables aleatorias, procesos estocásticos y teoría de colas

1.5.1. Modelo de redes

Los flujos de redes, es un excitante campo con un gran rango de aplicabilidad, repartidos en variados temas tales como las redes de computadoras. Los modelos de redes, tienen su origen en los trabajos de Euler quien plantea la Teoría de Grafos en los años 1730. Un grafo es una estructura que está compuesta por dos conjuntos: puntos y líneas. Los puntos pueden representar cualquier ubicación, y las líneas juntan estos puntos, tal como se muestra en la Figura 1.17. A las líneas también se les denomina aristas y arcos.

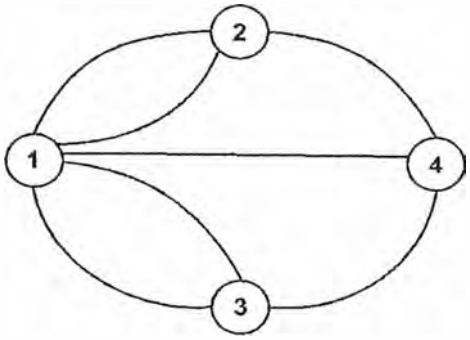


Figura 1.17. Representación de un Grafo

La notación para un grafo G representado por la Figura 1.17 es:

$G=(N,A)$, donde $N=\{1,2,3,4\}$ y $A=\{(1,2),(2,1),(1,4),(1,3),(3,1),(2,4),(3,4)\}$

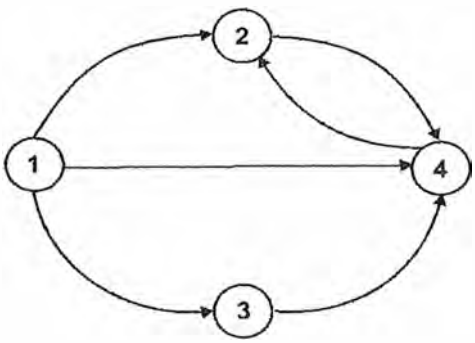


Figura 1.18. Representación de un Dígrafo.

En muchas situaciones un grafo se presenta con líneas orientadas, en este caso se denomina grafo dirigido o dígrafo, tal como se muestra en la Figura 1.18. La notación para el dígrafo *G* representado por la Figura 1.18 es:

$G=(N,A)$, donde $N=\{1,2,3,4\}$ y $A=\{(1,2), (1,3),(1,4),(2,4),(3,4),(4,2)\}$

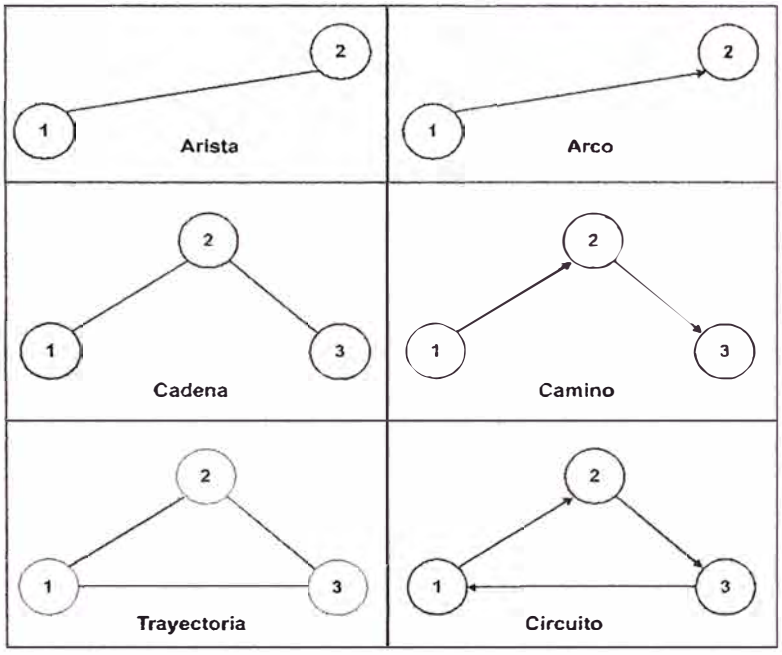


Figura 1.19. Representación de definición de redes

Una red es un grafo, con uno o más números asociados con cada arista o arco. Las redes de optimización, representan una de las más importantes áreas de la investigación de operaciones. Se denomina arista, a la línea que une dos puntos en un grafo; en cambio un arco es una línea para un dígrafo. Una secuencia de aristas forma una cadena, y una secuencia de arcos forma un camino, en donde el punto final de un arco es el punto inicial del siguiente. Cuando una cadena es cerrada se le conoce como trayectoria, y cuando un camino es cerrado se le conoce como circuito. Ver Figura 1.19.

Un grafo es llamado árbol, si satisface dos condiciones:

a. El grafo es conexo
b. El grafo es acíclico

Un árbol con n puntos (nodos o vértices) posee *(n-1)* arcos, los que se encuentran conectados, una propiedad de los árboles es que eliminando una arista, se pierde la conectividad. Por otro lado si se adiciona una arista, el árbol se convierte en un ciclo. Ver Figura 1.20.

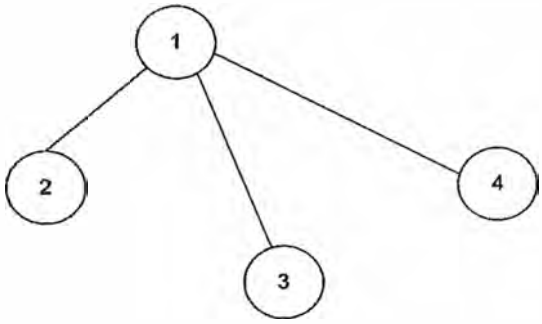


Figura 1.20. Representación de un árbol

Existen diversas maneras de representar numéricamente un grafo G, para propósitos computacionales; entre ellos destacan:

a. Matriz de adyacencia
b. Matriz de incidencia

Una matriz de adyacencia provee un camino muy simple para describir un grafo sin presentar vértices o arcos, para ello se define cada celda a_{ij} como:

$$a_{ij} = \begin{cases} 1 & si\ (i,j) \in A \\ 0 & en\ otro\ caso \end{cases}$$

Así para el dígrafo de la Figura 1.16 la matriz de adyacencia asociada es:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Para representar una matriz de incidencia, se hará uso del grafo $G=(N,B)$ mostrado en la Figura 1.21 y de la ley de conservación del flujo.

$Entrada_i - Salida_i = b_i$, donde b_i es una cantidad

Nodo 1	$-x_{12}$	$-x_{13}$			$= b_1$
Nodo 2	x_{12}	$-x_{23}$	$-x_{24}$		$= b_2$
Nodo 3	x_{13}	x_{23}		$-x_{34}$	$= b_3$
Nodo 4			x_{24}	x_{34}	$= b_4$

En el diseño de redes de computadoras, es necesario conectar los *n* nodos mediante *(n-1)* segmentos de cableado. Un arreglo que usa la menor cantidad de cableado es conocido como el árbol de extensión minimal.

Sea un grafo $G=(N,A)$ y (u,v) un par de componentes que definen una arista que pertenece a A. El peso de conectar (u,v) viene dado por $w(u,v)$, y T es un subconjunto de A, T está contenido en A que conecta todos los vértices, se tiene que:

$$W(T) = \Sigma w(u,v) \quad donde \quad (u,v) \in T$$

El que debe ser minimizado. T es acíclico y conectado a todos los pares de vértices; denominándose con el nombre de árbol de extensión al grafo G.

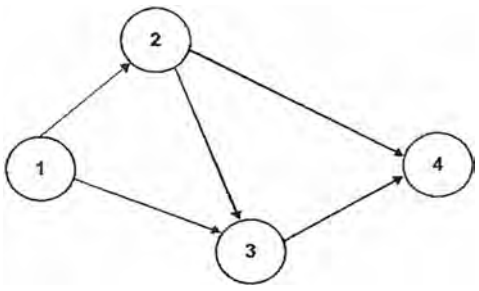


Figura 1.21. Dígrafo

Uno de los más simples y a la vez más importantes modelos de redes de optimización, es el problema de encontrar las rutas mas cortas. Considere un dígrafo $G=(N,A)$, con una longitud en cada arco (o un costo) asociado a $(i,j) \in A$. El problema consiste en uno de trasbordo, en donde el nodo inicial $(i=1)$ tiene una unidad para enviar y en el nodo final $(i=n)$ se recibe dicha unidad, es decir:

$$Min\ z = \sum_{(i,j)\in A} c_{ij}\ x_{ij}$$

$$\sum_{j=1}^{n} x_{ji} - \sum_{j=1}^{n} x_{ij} = \begin{cases} -1 & i = 1 \\ 0 & i = 2, \ldots, n-1 \\ 1 & i = n \end{cases}$$

$x_{ij} \geq 0 \quad \forall (i,j) \in A$

Existen varios tipos de problemas de la ruta más corta, y de acuerdo a ello, se han desarrollado diversos algoritmos.

	x_{12}	x_{13}	x_{23}	x_{24}	x_{34}	b
1	-1	-1	0	0	0	b_1
2	1	0	-1	-1	0	b_2
3	0	1	1	0	-1	b_3
4	0	0	0	1	1	b_4

Observe que:

[A] =						
	1	-1	-1	0	0	0
	2	1	0	-1	-1	0
	3	0	1	1	0	-1
	4	0	0	0	1	1
		(1,2)	(1,3)	(2,3)	(2,4)	(3,4)

Donde [A] es una matriz de $|N|$ filas, y por $|B|$ columnas,

siendo: $a_{ij} = e_j - e_i$ e_j, es el vector unitario con 1 uno en la fila j.

Entre los principales problemas de redes, se encuentran:

a. El problema del árbol minimal.
b. El problema de la ruta más corta.
c. El problema del flujo máximo.
d. El problema de flujo a costo mínimo.

Arbol de extensión minimal

Surgen los siguientes problemas:

i. Ruta más corta del origen al final.
ii. Ruta más corta de un nodo a los otros nodos.
iii. Ruta más corta entre cada par de nodos.

Flujo máximo

Considere una red que conecta dos nodos, uno de origen y el otro de destino, por varios caminos de nodos intermedios. Cada arco está asignado a un número conocido como capacidad de flujo. Asumiendo un estado estable, se encuentra un flujo de mayor valor que atraviesa la red desde el origen hasta el destino. Esta teoría conocida como el problema del flujo máximo fue desarrollado por Ford y Fulkerson en 1954.

Considere una red que envía flujo desde un origen s a un destino t, en donde:

x_{ij} = cantidad de flujo en el arco (i,j) y estando asociado a una capacidad, $0 \leq x_{ij} \leq c_{ij}$ que representa la máxima capacidad de flujo que atraviesa el arco en la unidad de tiempo.

Según las Leyes de Conservación se tiene:

- El Flujo que sale de i:

$$\sum_j x_{ij}$$

- El Flujo que llega a i:

$$\sum_j x_{ji}$$

- Se igualan en la siguiente expresión:

$$\sum_j x_{ji} - \sum_j x_{ij} = \begin{cases} -f & i = s \\ 0 & i \neq s,t \\ f & i = t \end{cases}$$

El problema del flujo máximo es formulado como:

$$\text{Max } z = f$$

$$\sum_{j=1}^{n} x_{ji} - \sum_{j=1}^{n} x_{ij} = \begin{cases} -f & i = s \\ 0 & en\ otro\ caso \\ f & i = t \end{cases}$$

$$0 \le x_{ij} \le c_{ij} \quad \forall (i,j) \in A$$

En el problema del transporte, los nodos representan localizaciones y los arcos representan rutas. El flujo natural de los bienes es unidireccional, esta consideración significa ser un dígrafo. El hecho que en un problema de trasbordo se conecte *n* puntos mediante *(n-1)* variables básicas corresponde a *(n-1)* arcos desde el árbol.

Teorema.- Una red con *n* nodos es un árbol si tiene *(n-1)* arcos y no posee ciclos.

El problema del flujo a costo mínimo es representado como:

$$Min\ z = \sum_{(i,j) \in A} c_{ij}\ x_{ij}$$

$$\sum_{j=1}^{n} x_{ji} - \sum_{j=1}^{n} x_{ij} = b \quad \forall i$$

$$x_{ij} \ge 0 \quad \forall (i,j) \in A$$

1.5.2. Teoría de probabilidades

La teoría de probabilidades se ocupa del estudio de sucesos imprevisibles (aleatorios), resultantes de la realización de procesos estocásticos (aleatorios cuyo valor varía con el tiempo). La idea central de esta teoría es que la ocurrencia de los sucesos aleatorios tiene una "regularidad estadística". Esta teoría es una parte de la matemática, y como tal está construida en forma axiomática.

Noción de Probabilidad, la probabilidad de un suceso E, es la razón (cociente) del número n de casos en los cuáles E tiene lugar; llamados casos favorables, al número total N de los casos elementales todos igualmente posibles:

$$p = \Pr(E) = \frac{n}{N}$$

Probabilidad inversa, probabilidad que ocurra el suceso contrario:

$$q = 1 - p = \frac{N - n}{N}$$

Probabilidad Compuesta, es la probabilidad de que se verifique A y B. Se denotará *Pr(A. B.)*

Probabilidad Total, es la probabilidad de realización de A y/o B (A solo, B solo o A y B conjuntamente). Se denotará: *Pr(A+B).*

Principio de las probabilidades compuestas:

$$Pr(A.B) = Pr(A).\ Pr_A(B) = Pr(B)\ .\ Pr_B(A)$$

Principio de las probabilidades totales:

$$Pr(A+B) = Pr(A) + Pr(B) - Pr(AB)$$

Si los sucesos se excluyen mutuamente:

$$Pr(A.B) = 0 \quad \text{y}$$

$$Pr(A+B) = Pr(A) + Pr(B)$$

1.5.3. Variable aleatoria:

Si una variable X puede tomar los valores $x_1, x_2, \ldots, x_n$ con una probabilidad para cada uno de esos valores: $p(x_1), p(x_2), \ldots, p(x_n)$ en donde:

$$\sum_{i=1}^{n} p(x_i) = 1$$

se le llamará variable aleatoria; se representará simbólicamente con una X y sus valores se representarán con x_i. Una variable aleatoria es una función que atribuye un valor real a cada uno de los posibles resultados de un experimento aleatorio, es decir, una variable aleatoria proyecta los elementos de S (espacio de prueba formado por todos los resultados posibles del experimento aleatorio E) en puntos de la recta de los reales (R). De esa forma si $r \in S$ y $x \in R$, entonces la variable aleatoria X es tal que $X(r)=x$. Un ejemplo de variable aleatoria nos viene dado por la propia función de probabilidad. Ver Figura 1.22.

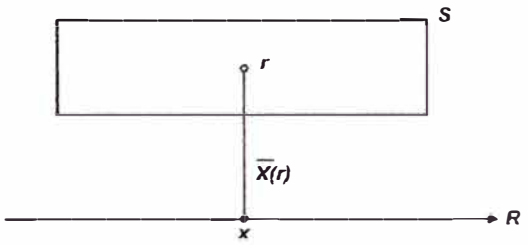


Figura 1.22. Definición de una variable aleatoria.

El dominio de una variable aleatoria X es el conjunto S y su imagen es el subconjunto de los reales formado por los valores que X asume, Una variable X es clasificada de acuerdo a las características de su imagen. Si la imagen consiste en un intervalo continuo finito o infinito en la recta de los reales, X es una variable aleatoria continua. Si la imagen de X está formada por un subconjunto de los reales con un número de elementos finitos o infinitos pero contables, X será una variable aleatoria discreta.

Esperanza matemática o valor probable: Es el valor medio ponderado de X:

$E(X)= x_1 p(x_1)+ x_2 p(x_2)+ \ldots + x_n p(x_n)$, escribiremos también: $\overline{X}$ o $\bar{x}$

Variable aleatoria reducida: $X' = X - \bar{x}$ se tiene: $E(X') = E(X - \bar{x}) = 0$

Función de una variable aleatoria:

Si $y=f(x)$, es una cierta función, entonces $Y=f(X)$ es una función de la variable aleatoria X y, a su vez, es una variable aleatoria.

***Media*:** Llamaremos esperanza de la función Y de la variable aleatoria X, la cantidad:

$$E(Y) = E(f(X)) = \sum_{i=1} f(x_i)p(x_i)$$

***Variancia*:**

$$\sigma_x^2 = E[(X - \bar{x})^2] = (x_1 - x)^2 p(x_1) + (x_2 - x)^2 p(x_2) + ... + (x_n - x)^2 p(x_n)$$

σ_x es llamada desviación estándar o desviación cuadrática media. Utilizaremos también la fórmula:

$$\sigma_x^2 = E(X^2) - [E(X)]^2$$

Función de probabilidad o distribución

La función $p(x)$ que toma valores discretos $p(x_1)$, $p(x_2)$, ..., $p(x_n)$ es llamada función de probabilidad o distribución . Se utiliza también la siguiente notación:

$$p(x_i) = \Pr(X=x_i)$$

Función de probabilidad acumulada o repartición:

Acumulando los valores de $p(x)$ se obtiene una nueva función llamada función de probabilidad acumulada o repartición. A esta función también se le conoce con el nombre de Función de Distribución de Probabilidad (FDP)

$$P(x) = \Pr(X \leq x) = \sum_{x_i \leq x} p(x_i)$$

Vemos que P(x) es una función no negativa, monotónicamente creciente con límites entre 0 y 1 en $-\infty$ e ∞, respectivamente.

La siguiente función es llamada repartición complementaria:

$$1 - P(x) = \Pr(X > x) = 1 - \Pr(X \leq x)$$

Función densidad de probabilidad (fdp)

A menudo es más cómodo trabajar con la derivada de la función de distribución de probabilidad $P(x)$ en vez de con la propia FDP. La derivada de $P(x)$ es llamada la función densidad de probabilidad.

$$f_{\bar{X}} = \frac{d}{dx} P(x)$$

Distribución de Poisson

Si X toma los valores enteros $r = 0, 1, 2, 3, \ldots$, con las probabilidades

$$p_r = \frac{a^r}{r!} e^{-a}$$

en donde a es un parámetro positivo, la distribución se llama Distribución de Poisson. La repartición correspondiente es:

$$P_r = \sum_{s=0}^{r} \frac{a^s}{s!} e^{-a} \text{ donde: } E(X)=a \quad \text{y} \quad \sigma_x = a$$

1.5.4. Proceso estocástico

Un proceso estocástico es una variable aleatoria cuyo valor varía con el tiempo. La parte dinámica de las probabilidades se denomina procesos estocásticos y corresponde a la incertidumbre de los procesos dinámicos. Un proceso estocástico no es otra cosa que una secuencia de variables aleatorias ordenadas por un conjunto índice.

$S_0, S_1, S_2, \ldots, S_n$

$\{S_n, n=0, 1, 2 \ldots\}$ donde n es discreto, y $\{Y_t, t \geq 0\}$ donde t es continuo, además n o t se encuentran referidos al tiempo.

El espacio de estados de un proceso estocástico, es una lista de los puntos posibles en cualquier punto del tiempo. Una variable aleatoria atribuye un valor real $X(r)$ a cada uno de los posibles resultados (contenidos en el espacio de prueba S) de un experimento aleatorio, E. Un proceso estocástico, a su vez, atribuye una función de tiempo real $X(r,t)$ a cada uno de los resultados $r \in S$. Por simplicidad de notación, representaremos un proceso estocástico simplemente con $X(t)$.

1.5.5. Teoría de colas

La teoría de colas fue desarrollada para proveer modelos que pronostiquen la conducta de los sistemas, que ofrecen servicios, productos de demandas aleatorias. A. K. Erlang, es considerado el padre de la Teoría de Colas, por su trabajo The Theory of Probability and Telephone Conversations en 1909. En 1927 Molina publica Application of the Theory of Probability to Telephone Trunking Problems y en 1928 Thornton Probability and its Engineering uses. En 1930 Pollazeck continuo los trabajos sobre ingreso Poisson, salida aleatoria y problemas de canales múltiples. Las más recientes contribuciones se deben a Lindley por sus ecuaciones integrales, a Bailey, Lederman y Reuter con soluciones dependientes del tiempo; a Kendall por las cadenas embebidas. También en la literatura de colas son significativos los trabajos de Benes, Bhat, Conway, Gaver, Little, Maxwell, Morse, Prabhu y Saaty.

Pérdida de Memoria

La propiedad Markoviana de la distribución exponencial es también conocida como la propiedad de la pérdida de memoria (memoryless).

Sea un intervalo de tiempo (t_0, t_1), $\Delta t = t_1 - t_0$ se cumple que:

$$P\{t \le t_1 \mid t \ge t_0\} = P\{t \ge t_0 \cap t \le t_1\} / P(t \ge t_0)$$

Desde el concepto de probabilidad condicional

$$P(B/A) = \frac{P(A \cap B)}{P(A)}$$

y la función exponencial con

$$f(t) = \lambda e^{-\lambda t} \quad t > 0$$

$$\int_0^T f(t) = 1 - \lambda e^{-\lambda t}$$

$$P\{t \le t_1 \mid t \ge t_0\} = \frac{\int_{t_0}^{t_1} \lambda\, e^{-\lambda t}\, dt}{\int_{t_0}^{\infty} \lambda\, e^{-\lambda t}\, dt} = \int_{0}^{(t_1 - t_0)} \lambda\, e^{-\lambda t}\, dt = P\{0 \le t \le (t_1 - t_0)\} = P\{0 \le t \le \Delta t\}$$

$$= 1 - \lambda\, e^{-\lambda \Delta t}$$

Es decir solo depende del intervalo de tiempo Exponencial en el caso continuo y Poisson en el caso discreto.

El proceso de Poisson

Sea la variable aleatoria X_t, definida para los valores 0, 1, 2, …, n con:

$$p_n(t) = P[X_t = n] \quad n = 0, 1, 2, \ldots, n$$

Condiciones de un proceso de Poisson:

- Las transacciones desde un estado Ej sólo son posibles a Ej+1.
- La probabilidad que en un tiempo pequeño Δt, la probabilidad de un evento es aproximadamente proporcional al intervalo de tiempo, siendo λ una constante.

  $$P_1(\Delta t) = \lambda\, \Delta t$$

- La probabilidad de dos o más eventos en un intervalo de tiempo pequeño Δt es despreciable.

  $$\sum_{k=2}^{\infty} p_1(\Delta t) = 0$$

De lo anterior resolviendo unas ecuaciones diferenciales y generalizando se puede obtener fácilmente:

$$p_n(t) = \frac{e^{-\lambda t}(\lambda\, t)^n}{n!} \quad n = 0, 1, 2, \ldots$$

que es la distribución Poisson con un valor medio ponderado:

$$E(x) = \lambda\, t$$

Fenómenos como flujo de paquetes de datos en redes de computadoras, desintegración radiactiva, llamadas telefónicas a una central, número de electrones que salen desde un CRT son modelados apropiadamente con procesos Poisson.

Características de un proceso de colas

Un sistema que está caracterizado por clientes que llegan por un recurso llamado servicio, es un sistema de colas. Diversos tipos de sistemas pueden ser vistos como un sistema de colas, bastando con reconocer al cliente y al servidor.

Los sistemas de colas tienen seis características:

1. Patrón de arribo de los clientes.
2. Patrón de servicio del servidor.
3. Número de canales en el servidor.
4. Disciplina de la cola.
5. Capacidad del sistema.
6. Tamaño de la población.
7. Adicionalmente existe el número de etapas.

Proceso Poisson y distribución exponencial

Los modelos de teoría de colas, asumen que el tiempo inter arribos y el tiempo de servicio obedecen a una distribución Poisson o que la tasa de arribos y la de servicio siguen una distribución exponencial.

Si el proceso de arribo sigue una distribución Poisson entonces:

$$p_n(t) = \frac{(\lambda t)^n e^{-\lambda t}}{n!}$$

La probabilidad de cero arribos en el tiempo t es:

$$p_0(t) = e^{-\lambda t}$$

y la probabilidad de existir arribos es:

$$1 - p_0(t) = 1 - e^{-\lambda t} \text{ equivalente a: } A(t) = p(T \leq t) = 1 - e^{-\lambda t}$$

de *A(t)* una función de probabilidad acumulada de *T* se tiene que su función de probabilidad *a(t)* es:

$$a(t) = A'(t) = \lambda\, e^{-\lambda t}$$

Esto significa que el número de arribos al sistema en el tiempo *t*, que viene dado por la variable aleatoria Poisson está asociado con el tiempo entre arribos que sigue una distribución exponencial.

Numero de canales, una decisión importante es el problema de decidir entre un sistema pool versus servidores separados. Suponga dos ejecutivos cada uno con una secretaria y la decisión que ellos cuenten con un mismo pool pero con dos secretarias. Otros ejemplos ocurren en un banco donde los clientes efectúan una sola cola y existen varios cajeros, como también en un supermercado que existen varias cajeras, y cada una tiene su respectiva cola. Hay sistemas de colas, como los de manufactura que consiste en una serie de facilidades, a estos se les denomina de estados múltiples.

Tamaño de la población, cuando la población de los clientes que vienen buscando servicio es muy grande (o infinita), no afecta a la tasa de arribos, porque la cantidad que se encuentra esperando es insignificante respecto de la cantidad que falta por llegar. Otro caso existe, cuando la tasa de arribos decrece cuando el número de clientes que esperan aumenta, existiendo el caso de llegar a ser cero, cuando todos los miembros de la población se encuentra en el sistema.

Disciplina de cola, existe un mecanismo que selecciona al cliente desde la cola, para ingresar al servicio, lo más común es el FIFO (First Input First Output) o el primero en llegar es el primero en salir. Otras disciplinas son: LIFO muy utilizadas en inventario,

SIRO servicio en orden aleatorio, y algunos de esquemas de prioridades son también útiles como mecanismo de selección..

Capacidad del sistema, en los sistemas de colas, existen muchas veces limitaciones sobre la cantidad de clientes que esperan, y cuando la longitud llega a su límite no se permite el ingreso por falta de espacio disponible. Una cola con límites en la longitud de cola, es conocida como cola capacitada, y es estudiada, como un proceso de bloqueo a los arribos si el sistema se encuentra lleno o full.

Metodología para modelaje en colas

Para construir un modelo de colas, se sigue paso a paso la siguiente metodología: Ver Figura 1.23.

1. Definir las tasas de arribo y de servicio, en función de n (el número de elementos en el sistema).

   Por ejemplo si la llegada de los clientes no se altera porque ya existen clientes, entonces:

   $\lambda_n = \lambda \qquad n \geq 0$

   Si la tasa de servicios, no varía por la existencia de clientes, entonces:

   $\mu_n = \mu \qquad n \geq 0$

2. Hacer uso de la ecuación de un Proceso de Nacimiento y Muerte para que incluyendo λ_n y μ_n se obtenga p_n. Otra forma consiste en derivarlo desde el diagrama de estados.
3. Desde la expresión

$$d_n = \prod_{i=1}^{n} \frac{\lambda_{n-1}}{\mu_1}$$

se encuentra: $p_n = \frac{d_n}{\sum_{n=0}^{\infty} d_n}$

El límite de la sumatoria se encuentra en el número de clientes de la fuente (infinito en muchos casos) y en M cuando la fuente es finita.

Un sistema es estable cuando:

$$0 < \sum_{n=0}^{\infty} d_n < \infty$$

4. Si p_n es la probabilidad en el estado estable de encontrar n clientes, entonces el número esperado de clientes en el sistema resulta:

$$L = \sum_{n=0}^{\infty} n p_n$$

Si únicamente s clientes, pueden ser servidos simultáneamente y los otros clientes ($n > s$) tienen que esperar, entonces el número de clientes en la cola resulta:

$$L_q = \sum_{n=0}^{\infty} (n - s) p_n$$

Modelos de línea de espera

Existen muchos modelos de línea de espera entre ellas tenemos:

- Modelo de cola simple
- Modelo de cola finita
- Modelo de cola simple de población finita
- Modelo de autoservicio
- Modelo de estaciones en paralelo.
- Modelo de servicio de máquinas
- Modelo de cola con canales en paralelo y truncada
- Modelos de colas con servicios no exponenciales

Modelo tipo Erlang, sea la familia de distribución de probabilidad Erlang:

$$f(x) = \frac{(\mu k)^k}{(k-1)!} x^{k-1} e^{-k\mu x} \qquad 0 < x < \infty$$

Siendo k un número positivo arbitrario. La media y la varianza, vienen dadas por: $E(x) = \frac{1}{\mu}$ y $Var(x) = \frac{1}{k\,\mu^2}$ y considerando un tiempo de servicio E(t), que se descomponen en k fases: luego cada tiempo es: $\frac{1}{k\,\mu}$ Con esto se obtiene los mismos valores de la media y la varianza. La figura 1.24 ilustra el comportamiento de E(t).

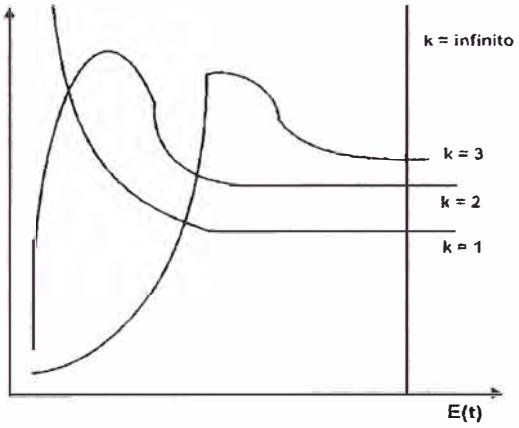


Figura 1.24. Tiempo de servicio en función de k.

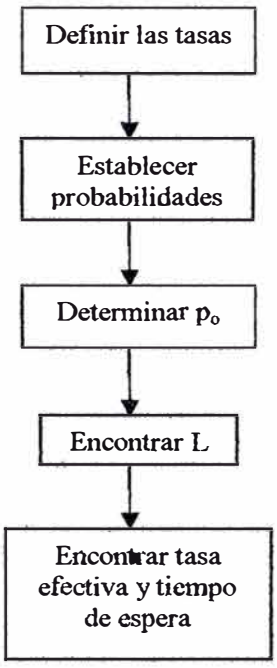


Figura 1.23 Metodología para modelaje en colas.

CAPÍTULO II
ESTUDIOS DE PROTOCOLOS

2.1. Especificación y validación de protocolos.

El uso continuo y creciente de los sistemas distribuidos conlleva a una ampliación permanente del ámbito de utilización de los protocolos, haciéndolos cada vez más complejos. Ha habido un crecimiento de varios órdenes de magnitud en la complejidad de los protocolos desde la aparición de los primeros terminales remotos hasta las actuales redes de computadoras. Como consecuencia esto plantea nuevos problemas de diseño, y la necesidad de una especificación precisa del protocolo. Una especificación precisa de un protocolo debe permitir verificar si un protocolo cumple el servicio para el que ha sido diseñado, además de facilitar al máximo la realización del protocolo en forma ejecutable. Si la especificación no es ambigua, siempre será posible encontrar un procedimiento mecánico (determinista) para obtener una versión ejecutable en un computador de dicho protocolo. Este paso de especificación a programa ejecutable puede ser automatizado con la ayuda de un compilador si el lenguaje de especificación se ha diseñado adecuadamente. Existen dos facetas dentro del problema de la especificación de protocolos. La primera; es necesario especificar el servicio que un protocolo debe dar a unos usuarios que quieran comunicarse entre sí; y la segunda, especificar el protocolo que debe dar dicho servicio. En la especificación del servicio se suele incluir una especificación del interfaz a través del cual se va a dar el servicio a los usuarios. Una vez hecha la especificación, el diseñador del protocolo debe enfrentarse con dos problemas. En primer lugar se debe verificar que el protocolo cumple la especificación del servicio.

Esta parte del problema suele llamarse verificación. En segundo lugar debe generarse el protocolo en forma ejecutable para la o las computadoras. Esta segunda tarea suele realizarse a partir de la segunda especificación del protocolo y de la especificación del interfaz. El diseño de un protocolo, siguiendo estos pasos, debe llevar a una realización muy fiable, y además fácilmente transportable a diversas computadoras.

2.1.1. Conceptos sobre especificación de Protocolos

La especificación precisa de un protocolo es un problema de mucho interés, debido al objetivo de alcanzar una estandarización y a facilitar las realizaciones prácticas. Hasta ahora se han estudiado diversos métodos, algunos de los cuáles permiten modelar de una forma abstracta y a la vez precisa, diversos tipos de protocolos. Especificar un protocolo consiste en especificar un algoritmo distribuido de tiempo real que debe responder a un entorno, compuesto por varios usuarios que quieran comunicarse entre sí y por unas conexiones a través de las cuáles deben comunicarse las diversas partes (entes) del protocolo. El sistema con memoria es la base de la mayoría de métodos de especificación de protocolos. Un sistema con memoria se puede representar de muchas formas diferentes, cada una de las cuáles resalta con mayor claridad ciertas propiedades del sistema. Las primeras especificaciones de protocolos que se hicieron fueron en lenguaje ordinario, por ser el método más inmediato. Este método es poco preciso y no facilita en absoluto el problema de la verificación ni el de la realización a programa ejecutable. Cuando el número de estados internos es pequeño, un diagrama de estados con su correspondiente tabla de transiciones muestra muy claramente los estados y las posibles transiciones en cada momento.

Cuando el número de estados posibles se hace muy grande, existen otras formas más compactas. Todas las secuencias de posibles entradas, estados y salidas pueden ser

representadas con una gramática cuyas expresiones se pueden representar de forma gráfica con grafos sintácticos. Una de las primeras alternativas de especificación de protocolos que apareció fue la de los grafos de control de comunicación. Se utilizaron con diversas variantes. Son equivalentes a los grafos sintácticos utilizados en la definición de lenguajes. En éstos el protocolo se especifica como un grafo orientado, cuyas ramas tienen asociadas el envío o la recepción de alguna información (Figura 2.1). Los diversos caminos que es posible recorrer a lo largo del grafo son las secuencias de envíos de información permitidas. Son adecuados para representar el intercambio de información en protocolos no muy complejos y se utilizaron fundamentalmente para representar protocolos orientados a carácter.

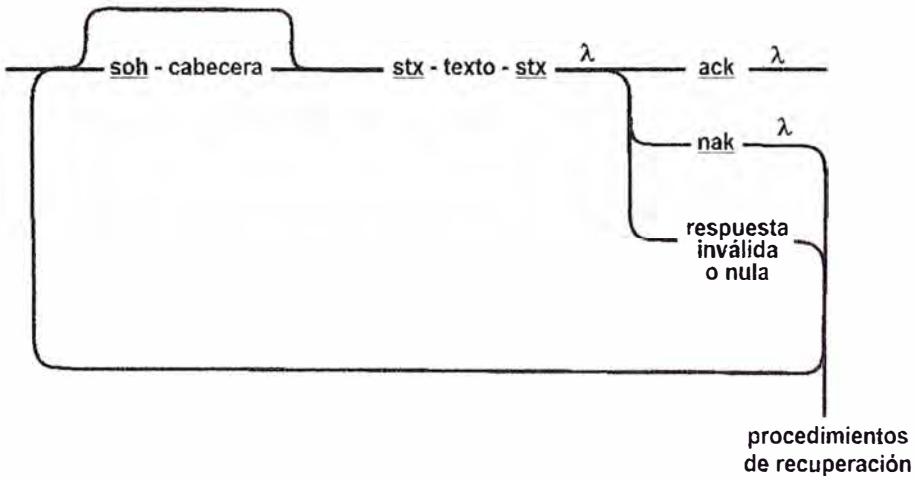
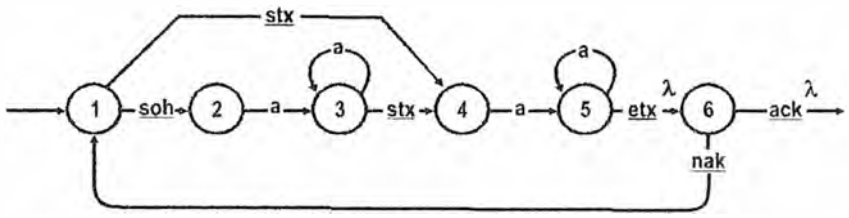


Figura 2.1. Representación de un protocolo mediante un grafo.

En la Figura 2.1. se representa un protocolo sencillo con un grafo del tipo de los utilizados por ANSI. Los símbolos SOH, STX, ETX, ACK y NACK representan caracteres de control del enlace y "λ" indica un cambio de estado del protocolo; paso de envío a escucha y viceversa. Por último, conviene resaltar que este tipo de especificación es incompleto. Es necesario añadir algunos detalles adicionales como controles de error, transformaciones de código, etc., que no son fácilmente representables en el grafo.

Otra metodología de especificación de protocolos está basada en representar el protocolo como un autómata con su diagrama de estados y su tabla de transiciones. Las respuestas del protocolo a las diversas situaciones pueden modelarse fácilmente como acciones asociadas a las transiciones de un autómata, cuyos estados representan las diferentes situaciones en que puede encontrarse el protocolo. El disparo de una transición está determinado por el entorno (recepción de una trama válida, de un ACK, de un NACK, etc.). En la Figura 2.2 se representa el mismo protocolo de la Figura 2.1 pero ahora modelado como un autómata. No se incluyen los procedimientos de recuperación. Este tipo de especificación plantea el inconveniente de la explosión de estados. La utilización de tramas con números de secuencia, el piggybacking, y la gran cantidad de tipos de tramas utilizadas en la mayoría de los protocolos actuales, crea una cantidad de estados muy grande y no es posible utilizar este tipo de especificación.



SUCESOS

ESTADO	soh	stx	a	etx	ack	nak	respuesta inválida o nula
1	A,2	B,4					
2			C,3				
3		E,4	D,3				
4			F,5				
5			G,5	H,6			
6					I,1	J,1	

A-J - Rutinas a ejecutar en la transición
a - Caracteres

Figura 2.2. Representación de un protocolo como un autómata.

Otro tipo de especificación de protocolos son las redes de Petri. Estas se suelen utilizar para verificar ciertas propiedades del protocolo representando habitualmente los

intercambios de información. Una red de Petri está compuesta por lugares (redondeles), transiciones (arcos) y marcas (puntos negros). Las transiciones tienen flechas de entrada y de salida, y su disparo se puede producir cuando en todos los lugares de entrada (flechas dirigidas de lugar a transición) hay una marca. El disparo de una transición consume todas las marcas de entrada y produce una marca en cada salida. El estudio de la evolución de la red se hace partiendo de un marcaje inicial y siguiendo todas las posibles secuencias de disparos de transiciones. En la Figura 2.3 se modela de una forma simplificada el intercambio de información del protocolo.

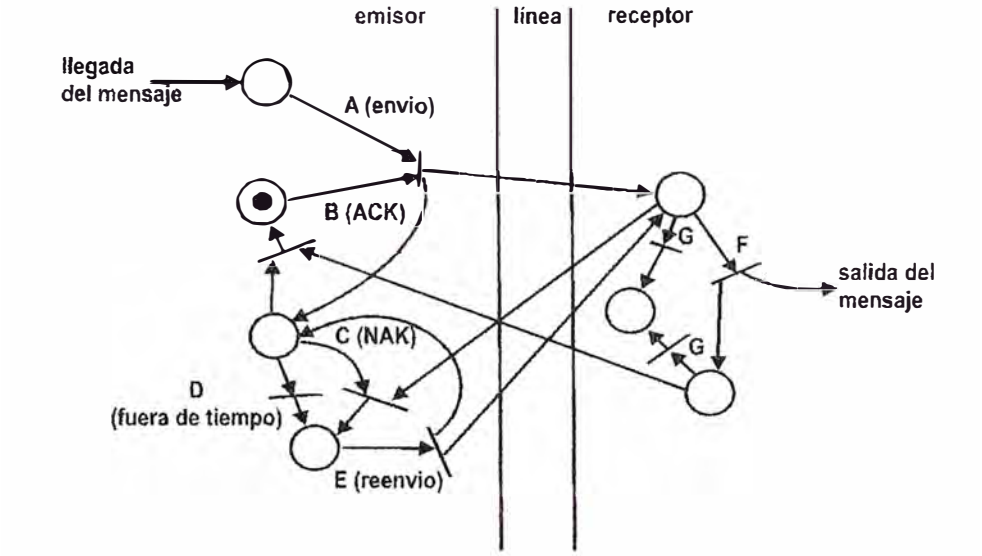


Figura 2.3. Representación de un protocolo mediante redes de Petri.

La transición A representa el envío de una trama. Una vez enviada, el transmisor se queda esperando la respuesta. La recepción del reconocimiento se representa con la transición B. La transición C representa la pérdida de la no aceptación del mensaje por parte del receptor, la transición D representa la recuperación del transmisor utilizando un temporizador cuando no recibe respuesta, la transición E representa un reenvío, la transición F representa la aceptación del mensaje por parte del receptor y la transición G representa la pérdida de la respuesta (ACK o NACK). Este tipo de representación se adapta bien a algunos de los problemas que presenta la validación de protocolos. Las

redes de Petri es otra forma de descripción de un autómata y sus representaciones resultan muy gráficas a la hora de representar concurrencia Existen técnicas fácilmente automatizables para detectar interbloqueos, viveza y otras características que implican una interrupción del servicio que el protocolo debe dar. Otros tipos de propiedades de la especificación del servicio no son validables con estos métodos. Por ejemplo, la duplicación o la pérdida de tramas no es posible detectarla a través de las propiedades de la red de Petri. En el ejemplo de la Figura 2.3 pueden duplicarse mensajes (transiciones A, F, D, E, F, B), no siendo posible deducirlo directamente de las propiedades de la red de Petri. El factor tiempo tampoco puede ser tenido en cuenta y además se produce también la explosión de estados cuando se trata de modelar un protocolo de cierta complejidad.

Muchas limitaciones de las especificaciones han llevado a utilizarse variantes de la red de Petri que permitan mejorar sus capacidades de especificación de protocolos. Se ha introducido el tiempo. También se han especificado condiciones de disparo. Una de las generalizaciones de mayor amplitud es la Red Numérica de Petri (Numerical Petri Net – NPN). Este tipo de red tiene una capacidad de representación muy grande y permite representar prácticamente cualquier tipo de algoritmo. Una particularización de las redes NPN denominada Máquina de Estados Extendida es considerado actualmente como el método más adecuado para representar protocolos. Son menos generales, pero permiten representar todo lo necesario para especificar un protocolo.

Los lenguajes de programación son lenguajes diseñados para escribir un autómata muy complejo, un computador, de forma mucho más adaptada a la resolución de ciertos problemas que un diagrama de estados. Los Lenguajes de programación también han sido utilizados para especificar protocolos, aunque no suelen representar de una forma clara e flujo

de control y el estado de un protocolo. En cambio, con ellos es fácil representar operaciones sobre los mensajes y variables de control de una forma clara y precisa. Esto ha llevado a especificaciones que simulan una máquina de estados o su equivalente, en el lenguaje utilizado. Tankoano, codifica en Pascal concurrente los niveles 2 y 3 (enlace y red) de la norma X.25 simulando una máquina de estados extendida. Sunshine describe las experiencias realizadas de especificación y verificación de protocolos utilizando las herramientas de especificación y verificación de software que posee el sistema AFFIRM. Los protocolos de la red SNA de IBM fueron modelados y verificados utilizando un lenguaje, el FAPL, basado en el PL1. García Hoffmann utiliza un lenguaje tipo CSP. Estas representaciones tienen la ventaja de no ser ambiguas, pero llevan a las especificaciones a un grado de abstracción que no suele considerarse suficiente para una representación abstracta del protocolo. Existen algunas excepciones a esta regla. Una de las más interesantes es el lenguaje desarrollado especialmente para especificar protocolos por Blumer y Tenney. En realidad es un lenguaje con una sintaxis diseñada para representar máquinas de estados extendidas de forma legible por un computador.

2.1.2. Especificación de las Interfases

Generalmente la especificación del interfaz entre dos niveles suele derivarse a partir de la especificación del servicio, aunque debe utilizarse junto con la especificación del protocolo. Esta se compone del conjunto de primitivas a través del cual el nivel superior va a solicitar los servicios que le da el nivel inferior. Esto tiene sus ventajas e inconvenientes. Por un lado permite determinar el interfaz del nivel, siguiendo la línea de los tipos abstractos de datos, tan fructífera en el campo del software. A cambio hay que concretar aspectos que quizá no sea necesario especificar para un protocolo en forma abstracta. Es un enfoque del problema que facilita el paso de especificación a programa ejecutable.

La especificación de las interfaces suele hacerse siguiendo técnicas parecidas a las utilizadas en tipos abstractos de datos, adaptándolas al entorno particular de los protocolos. Considerando al ente de un nivel como una caja negra, tenemos que la parte visible desde el exterior está formada por las interfaces. La respuesta a las diversas activaciones desde el interfaz, las determina el protocolo. Si el protocolo se ha especificado con una máquina de estados extendida, tenemos que el disparo de las transiciones estará sincronizado generalmente con la activación de las primitivas del interfaz.

Dentro de las posibles primitivas se pueden distinguir dos tipos de interacciones diferentes. Las primitivas con las cuales el usuario solicita un servicio al nivel (solicitudes), y las primitivas con las que el protocolo indica algo al usuario (indicaciones); estas últimas generalmente causadas directa o indirectamente por una solicitud en algún punto remoto. Los usuarios solicitan servicios al protocolo activando puntos del interfaz que deben producir las indicaciones correspondientes en los lugares remotos cuando sea necesario. Por ejemplo, en protocolos en los que es necesario establecer la conexión antes de intercambiar mensajes se necesitan cuatro tipos de interacciones entre niveles, para el establecimiento de una conexión. La secuencia de activaciones habitual es:

1) Solicitud de conexión (de usuario solicitante al protocolo)

2) Indicación de solicitud de conexión (de protocolo a usuario solicitante)

3) Solicitud de envío de reconocimiento (o no) (de usuario solicitado al protocolo)

4) Indicación de reconocimiento (o no) (de protocolo a usuario solicitante)

Un lenguaje de especificación de un interfaz debe distinguir los dos tipos de interacciones (solicitudes e indicaciones). Además debe representar el intercambio de mensajes o informaciones que se produce en la llamada a dicho punto del interfaz. Siguiendo con el ejemplo anterior, la "solicitud de conexión" debe dar al protocolo las direcciones del solicitante

y del destinatario, además de las informaciones necesarias acerca del "tipo de conexión" solicitado, como el número de circuito virtual a través del cual intenta establecer la conexión.

Si se tiene un sistema de procesos en cooperación, de modo que tal cooperación se realiza a través de un intercambio de mensajes, un protocolo es el conjunto de reglas las cuales gobiernan este intercambio. Limitando la interacción a intercambio de mensajes, significa que la información acerca del estado de un proceso puede ser conocida por otros sólo si esta información es explícitamente liberada por el proceso (por ejemplo cuando un mensaje se envía). Los sistemas distribuidos naturalmente emplean protocolos debido a que si la interacción de entidades es físicamente remota una de otra, el intercambio de mensajes es sólo la única manera posible de coordinar sus actividades. El uso de protocolos no está limitado a sistemas distribuidos físicos, sino a cualquier sistema en el cual la interacción entre entidades se haga por intercambio de mensajes. Un mensaje representa una letra, una secuencia finita de bits, una señal o un pulso.

Actualmente los sistemas distribuidos modernos pueden requerir protocolos extremadamente complejos, y pueden ser tan sutiles como un método sistemático sea necesario para garantizar que esta definición sea completa y no ambigua, y que el propósito del protocolo sea correctamente realizado. Las técnicas de especificación de protocolos basadas en modelos formales que son usadas para definir protocolos, y las técnicas de validación son usadas para asegurar su correcta y apropiada operación. Por lo tanto, las diferentes clases de protocolos requieren diferentes técnicas de especificación y diferentes técnicas de validación, y no hay un método único que pueda ser convenientemente usado para modelar y validar todos los protocolos. Sin embargo, hay relación entre las diferentes clases de protocolos y las técnicas de especificación y validación, algunas veces estas coincidencias son prácticas, mientras que otras veces,

aunque teóricamente las técnicas dadas pueden ser aplicadas a protocolos dados, en la práctica esto puede ser imposible.

Seguidamente se presenta un ejemplo del uso de una técnica de especificación de protocolos (modelación) y validación simple de protocolos; que ilustra sus ventajas, y demuestra lo sencillo de ésta técnica para mantener que los protocolos tengan ciertas características. Así, las características que determinan la aplicabilidad de una técnica a un protocolo son discutidas, y los diferentes modelos y técnicas se presentan y evalúan de acuerdo a los tipos de características que ellos pueden mantener.

2.1.3. El protocolo del bit alternante:

Este protocolo se describe usando el modelo de las redes de Petri. Como se ha explicado anteriormente, las redes de Petri son modelos de especificación de protocolos donde las "condiciones" se representan por nodos, y los "eventos" son representados por barras de transición. El mantenimiento de una condición es representada colocando una señal en ese nodo. Los arcos directos conectan nodos a barras y barras a nodos. Una barra de transición (un evento) puede disparar (suceder) si todos los nodos (condiciones) que ingresan a dicha barra de transición tienen señales (sostienen). Cuando una barra de transición dispara, remueve la señal de cada nodo de entrada y coloca una señal en cada nodo de salida. Para ilustrar esto, en la Figura 2.4, si sólo A1 mantiene una señal y B1 mantiene una señal (un estado denotado como A1B1) la única barra que puede disparar es la barra 11. Cuando 11 dispara, remueve la señal de A1 y coloca una señal en W1 y una señal en M1 (la red de Petri alcanza el estado W1M1B1). En este estado, entonces la barra 12 puede disparar alcanzando el estado W1C1, luego de lo cual la barra 13 puede disparar alcanzando W1K1D1. En este estado la barra 14 o la barra 15 pueden disparar,

representando eventos cuyo orden de ocurrencia es inmaterial. Las redes de Petri permiten a los nodos mantener más de una señal.

Todos los estados posibles que una red de Petri puede alcanzar y las posibles transiciones entre ellos definen una máquina de estado llamada Máquina Token. La máquina token de una red de Petri de la Figura 2.4 asumiendo un estado inicial A1B1, se muestra en la Figura 2.5. En la Figura 2.4 se representa el hecho de una versión simplificada del protocolo de bit alternante. Este protocolo involucra dos partes (un par de partes), el transmisor y receptor, conectados a través de un enlace. El transmisor envía un mensaje al receptor, y el receptor responde con reconocimiento. Cada mensaje lleva un bit de control (0 o 1) cuyo valor se alterna para mensajes consecutivos, y cada reconocimiento lleva un bit igual al bit de control llevado por el mensaje reconocido.

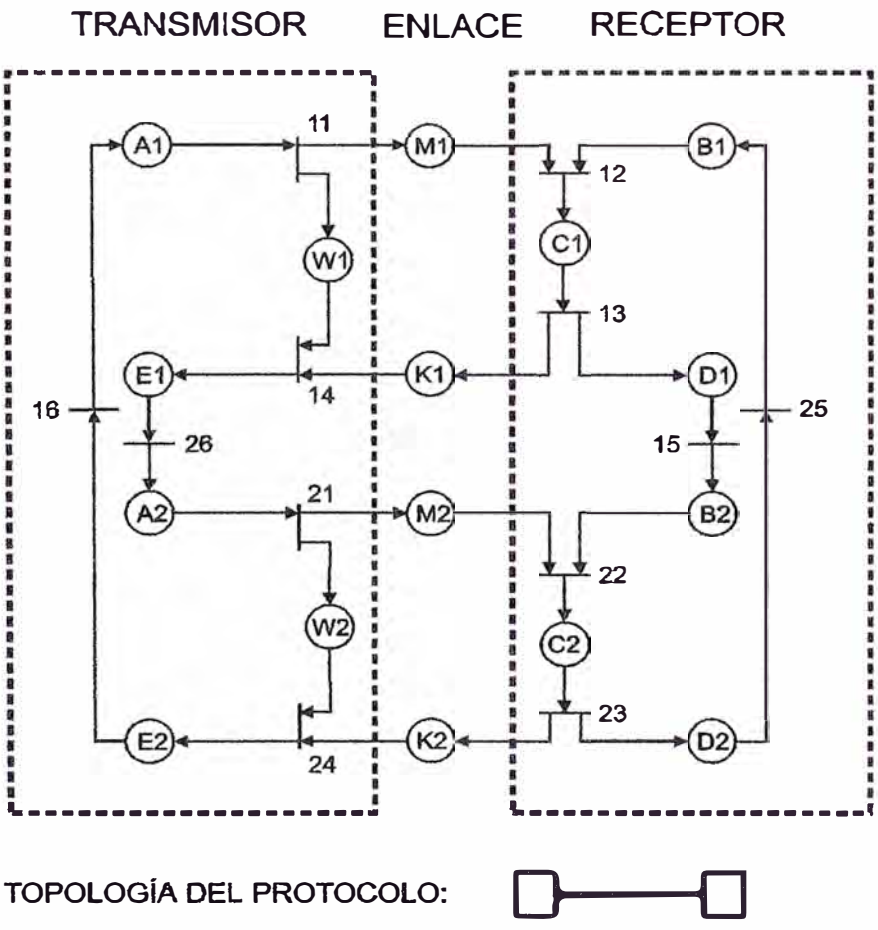


Figura 2.4 Representación del protocolo de bit alternante mediante una Red de Petri.

Los nodos de la Figura 2.4 representan las siguientes condiciones:

A1 : listo para enviar mensaje con bit de control 0

B1 : listo para recibir mensaje con bit de control 0

M1 : mensaje con bit de control 0 en tránsito

K1 : reconocimiento de mensaje con bit de control 0 en tránsito

W1 : esperando para el reconocimiento del mensaje con bit de control 0

C1 : mensaje con bit de control 0 fue recibido

E1 : reconocimiento al mensaje con bit de control 0 fue recibido

D1 : mensaje con bit de control 0 está siendo procesado

A2, B2, M2, K2, W2, C2, E2, D2 tienen el mismo significado pero para mensajes o reconocimientos que lleven el bit de control 1.

Las barras de transición de la Figura 2.4 ejecutan los siguientes eventos:

11 : envía mensaje 0

12 : recibe mensaje 0

13 : envía reconocimiento del mensaje 0

14 : recibe reconocimiento del mensaje 0

15 : mensaje 0 procesado

16 : mensaje 0 producido

21, 22, 23, 24, 25, 26 ejecutan los mismos eventos pero para mensajes o reconocimientos que lleven bit de control 1.

Desde que en este caso, la máquina token es finita, se puede ver fácilmente que después que un mensaje es enviado será recibido, y que mensajes consecutivos son enviados llevando el bit de control alternante, que no es una clave que no tiene sentido, demostrando así que el protocolo incluye los mecanismos de recuperación y estadísticas para posibles fallas. La topología de un protocolo es el gráfico cuyos nodos son las partes del protocolo y los arcos denotan enlaces que permiten las interacciones entre las partes (Figura 2.4). El incremento en la complejidad de la topología puede también impedir la aplicabilidad de una técnica. Por ejemplo, un protocolo con partes simples puede no estar validado por generación de estados globales exhaustivos si incluyen demasiadas partes.

La complejidad del comportamiento del enlace de la topología del protocolo puede también impedir la aplicabilidad de una técnica de especificación o una técnica de validación de protocolos. Por ejemplo, los enlaces pueden perder mensajes, de modo que un enlace complejo puede ser representado como una parte que modela su comportamiento, y así los problemas causados por la complejidad de enlaces no requieren tratamiento especial.

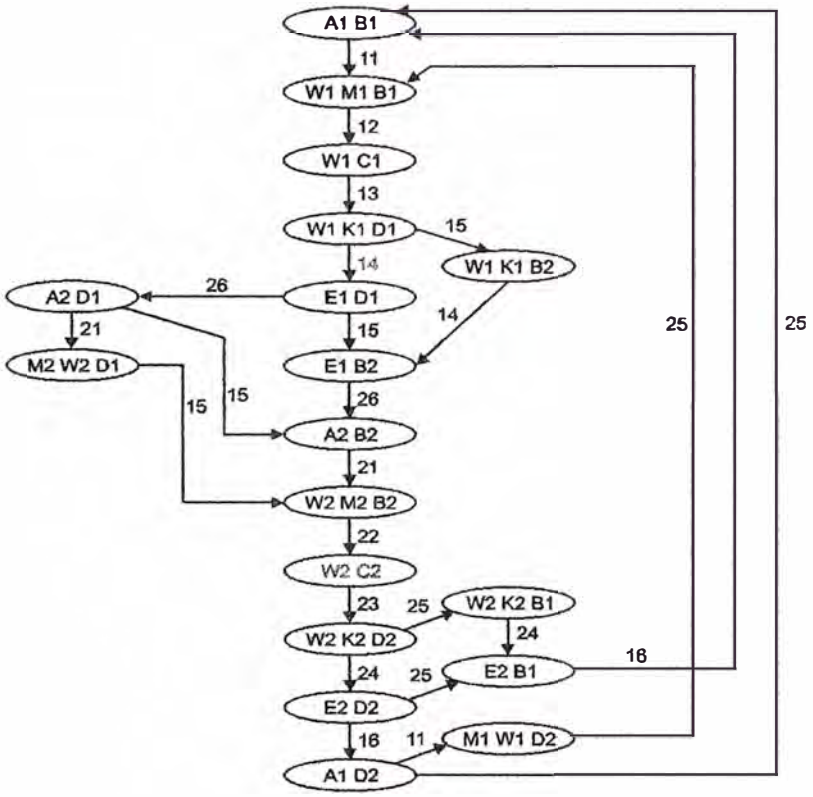
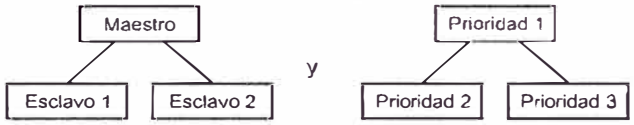


Figura 2.5. Maquina Token del protocolo de bit alternante.

2.1.4. Características de los protocolos:

La facilidad para la aplicación de una técnica de especificación y validación de protocolos es afectada por dos características del protocolo relativamente independientes: Características de parte y característica de topología. La característica de parte de un protocolo se da por el conjunto de todos los posibles pares de secuencias de mensajes que llegan y salen. La característica de topología de un protocolo es el conjunto de topologías

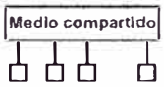
que son diseñadas para trabajar en el protocolo. La característica de topología más simple es el par de partes (Figura 2.4) tal como un conjunto de una topología única. Un lazo de hasta 64 partes es también un ejemplo de topología única. Un lazo de más de 64 partes es una característica que incluye muchas topologías, y un protocolo con tal característica es aquel que puede trabajar en alguna de estas topologías. Notar que estos son protocolos que tienen una característica de topología sin límite, tal como un número infinito de topologías permitidas. El cuadro 2.1 muestra ejemplos de tipos de característica de topología. Las entradas en la tabla son ordenadas por incremento de la generalidad, así cada entrada es un subconjunto de entradas que aparecen más tarde.

TIPO	EJEMPLO
Par de pares	
Muestra de una pequeña topología	
Una clase limitada de topologías finitas (por ejemplo, un conjunto de un número limitado de topologías)	"cualquier lazo de hasta 64 pares"
Una clase ilimitada de topologías finitas (por ejemplo, un conjunto de un número ilimitado de topologías)	"cualquier lazo" Otro ejemplo: "cualquier topología"

Cuadro 2.1. Ejemplos de tipos de característica de topología.

Un protocolo también puede ser diseñado para trabajar en una topología envolvente. Por ejemplo, un protocolo de enrutamiento puede trabajar en alguna topología de redes de computadoras donde los nodos operativos y los enlaces operativos pueden llegar a ser inoperacionales, y nuevos nodos y enlaces pueden llegar a ser operacionales. Esto puede ser considerado como si la topología fuera envuelta durante la operación. Otro ejemplo de la evolución de topología se da por el progreso de una conversación de multipartes telefónicas en un intercambio telefónico avanzado. En estos casos, la característica de

topología incluyen, en suma al conjunto de topologías, las transiciones permitidas de una topología en otra. El Cuadro 2.2 lista unos cuantos ejemplos de protocolos reales y su característica de topología.

PROPOSITO DEL PROTOCOLO	CARACTERISTICA DE TOPOLOGIA	EJEMPLOS
Transferencia de datos Punto-a-punto		SDLC [SDLC] X.25 [X.25] Host-Host [STEN]
Transferencia de datos con medio compartido	Topología limitada o ilimitada de la forma: 	Alohanet [ABRA] Ethernet [METC]
Sincronización de reloj de tiempo en una red de computadoras	cualquier topología (fija o envolvente) Cualquier grafo completo	ver: [FINN] ver: [LAMP]
Actualización de múltiples copias de un archivo	Cualquier grafo completo Cualquier lazo	ver: [MULL] ver: [ELLI]

Cuadro 2.2. Característica de topología del protocolo–Ejemplos de protocolos reales.

Un modelo de especificación de protocolos es teóricamente aplicable a un protocolo dado, sí y sólo sí es bastante potente para representar cada una de las características de parte y las características de topología de un protocolo. Para ilustrar esto, la característica de parte del transmisor (o receptor) en la Figura 2.4 puede ser expresado como repeticiones de las cadenas: M1 K1 M2 K2. Pero tales expresiones pueden ser representadas por un modelo de especificación de protocolos como una red de Petri, y de hecho una máquina de estados finito podrá ser usada para representar esta característica. La topología simple del protocolo de bit alternante es también representada por la conectividad de la red de Petri de la Figura 2.4.

En el modelo usado para describir el protocolo, la aplicabilidad teórica o práctica de una técnica de validación depende de las características del protocolo (característica de parte y característica de topología) y las propiedades del protocolo. Por ejemplo la

descripción de cual validación es aplicada para ser validada. La aplicabilidad práctica de un modelo de especificación de protocolos o de una técnica de validación es difícil para formalizar. La aplicabilidad práctica implica aplicabilidad teórica, pero también brevedad, facilidad de entender, conveniencia para usar, etc., las cuales son difíciles de cuantificar y algunas veces dependen de experiencia personal o prueba, y fuentes disponibles. Inicialmente, la mayoría de trabajos en modelado (especificación) y validación de protocolos se hace para protocolos de características de partes simples, tal como todas aquellas que pueden describirse por máquinas de estado finito. De modo que, desde el punto de vista de la topología, con unas cuantas excepciones la mayoría de trabajos formales realizados son aplicables sólo a topología extremadamente simples, usualmente un par de partes.

2.1.5. Modelos para especificación de protocolos:

Como se explicó anteriormente, existen muchos modelos de especificación de protocolos, de los cuales analizaremos unos cuantos, y realizaremos las extensiones necesarias para especificar protocolos de característica de topología compleja. En esta parte vamos a tratar los siguientes puntos:

A. El modelo de máquina de estado finito.
B. Redes de Petri y modelos relacionados.
C. Representación de un número ilimitado de topologías.

A. El modelo de máquina de estado finito:

Una máquina de estado finito puede ser usada para describir el estado global del protocolo, o alternativamente, una máquina de estado finito puede ser usada por cada parte, como se describe en el ejemplo de un transmisor y un único buffer receptor de la Figura 2.6. El transmisor envía un mensaje de dato y espera por la confirmación del mensaje enviado por el receptor cuando el buffer está vacío otra vez. En la aproximación

multimáquina, una transición marcada en una máquina con ENVIO y una transición en una máquina diferente marcada con RECIBO tienen los mismos parámetros (por ejemplo el mismo mensaje) y son ejecutadas simultáneamente; las máquinas se dice que están acopladas. Si el retardo de la transmisión del mensaje no es importante, las máquinas de transmisión y recepción pueden ser directamente acopladas y la máquina para el enlace omitida. El modelo de máquina simple y los modelos de máquinas acopladas son teóricamente equivalentes y son teóricamente aplicables a cualquier protocolo que tenga partes finitas (por ejemplo caracterizables por expresiones regulares) y un número limitado de topologías.

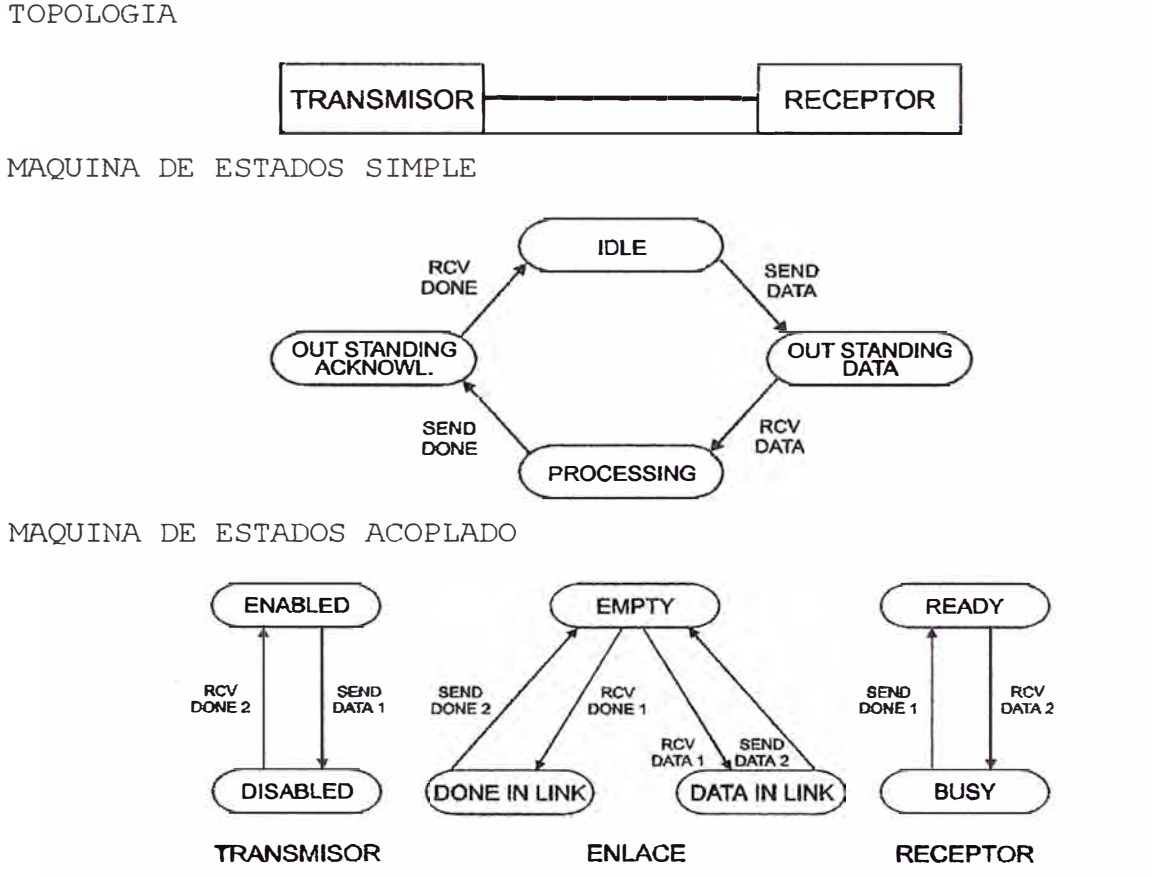


Figura 2.6 Representación mediante una Máquina de Estados de un protocolo de característica de topología simple o única.

Cuando un protocolo con una característica de topología múltiple es representado por este modelo, para cada topología permitida hay una correspondencia a conjuntos

diferentes de estados globales. Luego, la evolución de topologías puede ser representado por transiciones permitidas entre estados de diferentes conjuntos, y desde que el número de estados globales es finito, sólo un número finito de topologías puede ser representada. En la práctica, ambas aproximaciones son aplicables a protocolos de características de topología simples o únicas (usualmente un par de partes) y para partes que tienen no más de unas cuantas docenas de estados.

Una ventaja práctica de la aproximación de máquina de estados única es que las propiedades globales pueden ser directamente chequeadas o designadas en el modelo. Una ventaja de la aproximación de máquinas acopladas es que puede ser directamente implementada en cada parte sin los problemas de descomponer la descripción de máquinas únicas entre las partes. Tal descomposición puede hacerse de diferentes maneras, permitiendo la posibilidad de implementaciones no compatibles del mismo protocolo. La aproximación de máquina de estados única es usada para describir actualmente varios estándares adoptados (por ejemplo X.21, X.25, y otros protocolos.).

B. Redes de Petri y modelos relacionados:

El uso del modelo de especificación de protocolos mediante la red de Petri ha sido explicado anteriormente. La aplicabilidad teórica de redes de Petri como un modelo descriptivo es amplia en máquinas de estado finito, y algunos protocolos tienen un número infinito de posibilidades de estados que pueden representarse por una red de Petri en el cual el número de llamadas puede crecer sin límite. Para ilustrar esto, en el protocolo de la Figura 2.7, la barra PRODUCE no tiene condiciones de entrada y por lo tanto puede disparar arbitrariamente, así un número arbitrario de llamadas puede acumularse en el nodo S. Esto permite que un número arbitrario de disparos de la barra ENVIO provoquen que un número arbitrario de llamadas se acumulen en el nodo M. Una

situación similar puede ocurrir con respecto al nodo R. Por definición de las redes de Petri, las llamadas no necesariamente se quedan en el nodo en el mismo orden que llegan, y así la Figura 2.7 representa un protocolo que permite que algún número de mensajes en espera de salir puedan ser enviados y recibidos sin ningún orden, de modo que, el modelo de red de Petri no es universal debido a que ciertas características de parte no son representables por este modelo. Para ilustrar esto, si en el ejemplo anterior se requiere un número arbitrario de mensajes en espera de salir que serán recibidos en el mismo orden en que son enviados, esto no será representable por una red de Petri.

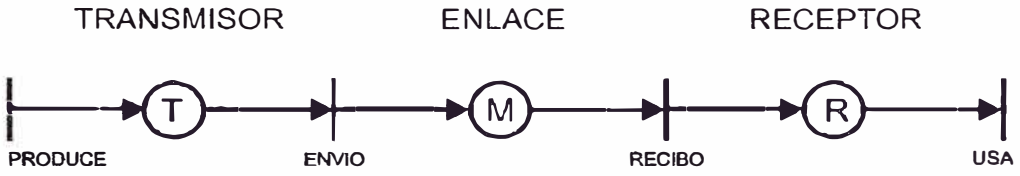


Figura 2.7 Representación de un protocolo mediante una Red de Petri, protocolo que no es representable por una máquina de estados finitos.

La aplicabilidad práctica de las redes de Petri es limitada a las máquinas de estado finito, pero en muchos casos se amplía, por ejemplo, el protocolo de la Figura 2.7 es representado en la práctica por una red de Petri, pero dicho protocolo no puede ser representada por una máquina de estado finito teóricamente uniforme. Las redes de Petri son convenientes para representar protocolos que pueden operar con varias cantidades de unos cuantos recursos (por ejemplo número de buffers, etc.). En este caso, una sola red de Petri basta, y la cantidad actual de recursos será representada por el número inicial de llamadas colocadas.

Las redes de Petri también serán convenientes para representar partes en la que varios eventos pueden ocurrir en orden arbitrario, por ejemplo, la red de Petri de la Figura 2.8 será completamente compleja para representarse por una máquina de estado finito, pero la representación con la red de Petri es relativamente compacta.

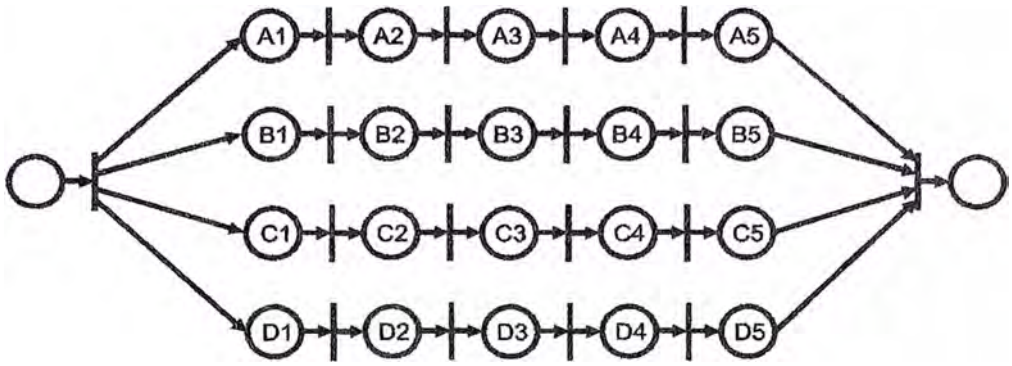


Figura 2.8 Representación de un protocolo mediante la red de Petri, protocolo difícilmente representado por una máquina de estado finito.

El principal defecto práctico de la red de Petri (así como de una máquina de estado) es el rápido aumento del gráfico con la complejidad del protocolo. Para aliviar esto, las mejoras y generalizaciones de los modelos básicos son propuestos y usados para representar protocolos, estas mejoras resultan de una notación más compacta, y esto también tiene una amplia aplicabilidad teórica como una herramienta descriptiva que las redes de Petri básicas, que son más difíciles de analizar.

En la práctica los lenguajes de programación estándares de alto nivel son convenientes para representar números, datos, variables, contadores, etc., pero no estructuras de control complejas. Así, este modelo es usado principalmente para representar los aspectos de transferencia de datos de protocolos, mientras que los modelos gráficos (máquinas de estado y redes de Petri) son principalmente usadas para representar los aspectos de control, tales como sincronización, inicialización, etc., para los cual ellos son más convenientes

C. Representación de un número ilimitado de topologías:

El problema básicamente es: ¿Como se pueden representar protocolos que tienen un número ilimitado de topologías permitidas, por una expresión finita?. Esto se puede hacer dando un número limitado de partes básicas y una regla de conectividad en respuesta de

las partes básicas en topologías permitidas, por ejemplo, un lazo de algún número arbitrario de partes idénticas, puede representarse por la representación de una copia de las partes, mostrando sus conexiones en las fronteras, y asumiendo un número finito de partes en alguno de los lazos resultantes, como se muestra en el ejemplo de la Figura 2.9.

Cualquiera de los modelos de especificación de protocolos puede ser usado para describir las partes básicas, con tal que aquella señale alguna otra manera de expresar la regla de conectividad de las partes que sean agregadas. Algunas veces, como en la Figura 2.9, los índices i, j, k son usados sólo como enlaces para mostrar la conectividad de la topología y los valores diferentes de puntos de señalización son usados sólo para diferentes ubicaciones en la red, pero no para diferentes propiedades de las partes, otras veces, los valores de los señalizadores son usados para denotar diferencias entre partes, por ejemplo, el mayor índice denota la mayor prioridad.

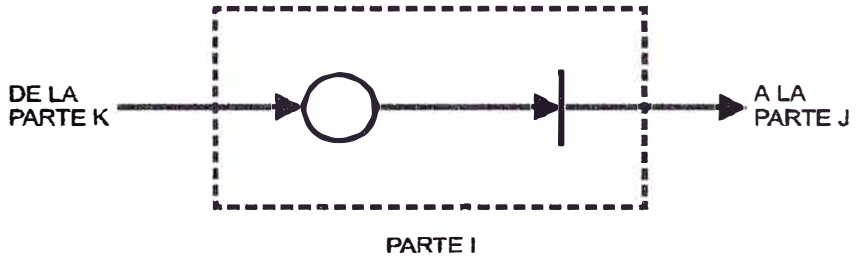


Figura 2.9. Representación de un protocolo con un número ilimitado de topologías.

Un número ilimitado de topologías puede incluir no sólo un número ilimitado de partes sino también un número ilimitado de enlaces para otras partes. En tal caso, cada parte puede tener un número arbitrario de vecinos, los cuales pueden ser representados por las partes básicas permitidas para tener una lista de vecindades de longitud arbitraria. Claramente estas técnicas de modelado son aplicables no sólo a un número ilimitado de topologías, sino también pueden llegar a ser de un gran valor práctico en la especificación de protocolos con un número de topologías limitado pero grande. Para ilustrar esto, si el lazo generado por la parte básica de la Figura 2.9 está limitado a 1024 elementos, puede

ser mejor representado como un caso ilimitado en el cual $0 < i, j, k <$ infinito es cambiado a $0 < i, j, k < 1023$.

Si cada parte tiene una lista o tabla de vecinos, una topología evolutiva puede ser representada como un cambio en estas tablas, posiblemente incluyendo la creación o destrucción de tablas, lo cual puede ocurrir durante la operación del protocolo. La descripción de protocolos involucra topologías evolutivas que requieren operaciones que causan cambios en la interconexión de las partes, y el modelo usado para describir tales protocolos incluirá estas operaciones.

2.1.6 Técnicas de validación de protocolos:

Existen muchas técnicas de validación de protocolos, pero la mayoría de ellas hace uso de una las cuatro técnicas básicas de validación. Aunque dos técnicas que usen la misma técnica básica de validación pueden mostrar algunas diferencias prácticas, en realidad son teóricamente equivalentes para cada protocolo que es representable por las dos técnicas en el que las técnicas de validación son respectivamente aplicadas.

Las técnicas básicas de validación de protocolos son:

A) Generación de estados globales
B) Prueba de afirmación
C) Inducción sobre la topología
D) Adherencia a condiciones suficientes

Se describen a continuación las técnicas básicas de validación.

A) Generación de estados globales:

Una de las técnicas de validación de protocolos más comunes es la generación exhaustiva de estados globales como se ejemplifica por la máquina Token de la Figura 2.5. La aplicabilidad teórica de esta técnica está limitada a protocolos de característica de topología con número limitado de topologías y características de parte de estado finito. La

aplicabilidad práctica está limitada a protocolos de característica de topología muy simples (tal como media docena de partes). Varios protocolos actuales, o partes de ellos son validados usando esta técnica. Una ventaja de esta técnica es que la generación de estados globales puede ser fácilmente mecanizada, y varias propiedades pueden ser automáticamente probadas, de modo que, desde que ahí existen protocolos donde algunas propiedades serán usualmente mantenidas en casos excepcionales que sean permitidos, la falla pasa una prueba automática que no necesariamente implica que el protocolo no sea correcto, y por ello la interpretación humana de los resultados sea necesaria. Algunas veces las propiedades del espacio de estado total puede ser validado generando un pequeño subconjunto de los estados, y esto puede incrementar enormemente la aplicabilidad de la técnica.

B) Prueba de la afirmación:

Otra técnica común de validación de protocolos es la prueba de la afirmación, la cual es aplicada a la descripción del protocolo, como si la descripción fuera un programa paralelo. Esta técnica es usualmente aplicada a modelos de especificación de protocolos mediante lenguajes de programación de alto nivel, pero teóricamente puede ser aplicada a otros modelos de especificación de protocolos. La forma usual de aplicar esta técnica es por la conexión de los valores de un atributo de las variables para ciertos puntos en un programa y probar que cuando el programa alcanza estos puntos el atributo es correcto. Esto puede ser generalizado a una colección de programas cooperantes por la conexión del atributo a conjuntos de puntos, tal que en cada conjunto haya al menos un punto de cada programa, entonces esto prueba que cuando todos los puntos de uno de los conjuntos son alcanzados por los correspondientes programas, el atributo se mantiene. El método puede ser generalizado a protocolos con un número ilimitado de topologías con tal que el

atributo deseado y los conjuntos de puntos puedan ser expresados por expresiones limitadas. En la práctica, la técnica básica de la prueba de la afirmación es principalmente aplicada a protocolos de característica de topología simple, pero algunas veces también son aplicadas a protocolos de característica de topología y de parte complejas, un ejemplo de este método es el protocolo de bit alternante. Como la construcción de pruebas puede requerir creatividad, esta técnica no es completamente automatizada. Mientras que la generación de estados globales es más conveniente en la prueba del control de propiedades (tal como ciertos eventos que podrán o no ocurrir), la prueba de la afirmación es principalmente usada en pruebas de transferencia de propiedades de datos, particularmente en protocolos que involucran partes con espacios de estado grandes o infinitos. Las dos técnicas pueden ser combinadas en suma para capitalizar las ventajas de cada uno.

C) Inducción sobre la topología:

Por esta técnica de validación de protocolos, el mantenimiento de una propiedad o la ocurrencia de un evento se prueba mostrando que ciertas condiciones se propagarán a través de la topología. El uso de la inducción sobre la topología es teóricamente aplicable a protocolos de alguna característica, probando que si la topología es ilimitada puede ser representada como se explicó anteriormente. La técnica de inducción sobre la topología es exitosamente aplicada en la práctica a varios protocolos con característica de topología ilimitada (fijas o envolventes).

D) Adherencia a condiciones suficientes:

En esta técnica de validación de protocolos, el protocolo es diseñado de manera que el diseño de cada paso es hecho satisfaciendo condiciones las cuales son suficientes para garantizar las propiedades requeridas. Esto es, en lugar de diseñar un protocolo y

posteriormente probar su corrección, esta técnica de validación es dirigida a diseñar directamente el protocolo y corregirla en el instante mismo de su construcción. Este es un concepto similar al utilizado en el desarrollo de software. Esta técnica puede ser usada en cualquier protocolo de característica de topología y de parte, y su principal ventaja es la facilidad para aplicar, y la corrección es directamente garantizada. Su principal defecto es que las condiciones suficientes pueden ser demasiados exigentes; por ejemplo, puede haber muchos protocolos correctos que serán rechazados debido a que ellos no satisfacen las condiciones suficientes. Por otro lado, las condiciones suficientemente mínimas (o preferiblemente condiciones necesarias y suficientes) son usualmente complejas y difíciles de hallar.

2.1.7. Protocolo de enrutamiento:

Los ejemplos de modelado (especificación) y validación de protocolos descritos anteriormente, son una versión simplificada del protocolo de enrutamiento. El protocolo de enrutamiento es ejecutado por una red de computadoras de N nodos arbitrarios y L enlaces, donde $L \leq N \times N$. Se supone que los arcos son no dirigidos, por lo tanto, si el enlace de los nodos i, j pertenecen a L, entonces el enlace de los nodos j, i también pertenecen a L, también se cumple que $L = N (N-1) / 2$. Para cada enlace i, j de L se asigna una constante positiva d_{ij} llamada distancia. La distancia representa el costo de uso del enlace, donde uno de los N nodos es llamado SINK. Cada nodo i de N tiene una variable d_i la cual almacena la distancia estimada del nodo i al nodo SINK y una variable p_i llamada "vecino preferido" que apunta a uno de los vecinos del nodo i. El conjunto de todos los puntos p_i es denotado por P. Inicialmente, los puntos de P forman sobre la red entera un árbol dirigido enrutado al nodo SINK. Un ejemplo de tal red se muestra en al Figura 2.10. Los puntos de P proveen un camino de lazo libre desde cada nodo al nodo

SINK y el propósito del protocolo es actualizar los puntos de forma que:

* en cada instante ellos formen un árbol enrutado al nodo SINK, y
* se minimice la longitud del camino (la suma de los d_{ij} a través del camino) desde cada nodo al nodo SINK.

En situaciones como las definidas arriba, un nodo i enviará a su vecino un mensaje de comunicación de su distancia estimada a SINK, d_i. Tal mensaje se denota por MSG(d_i), y llegará al nodo receptor dentro de un tiempo arbitrario finito. El mensaje se refiere al mensaje de control MSG(d_i), y no se refiere al mensaje de dato ordinario, el cual es transmitido a través de la red.

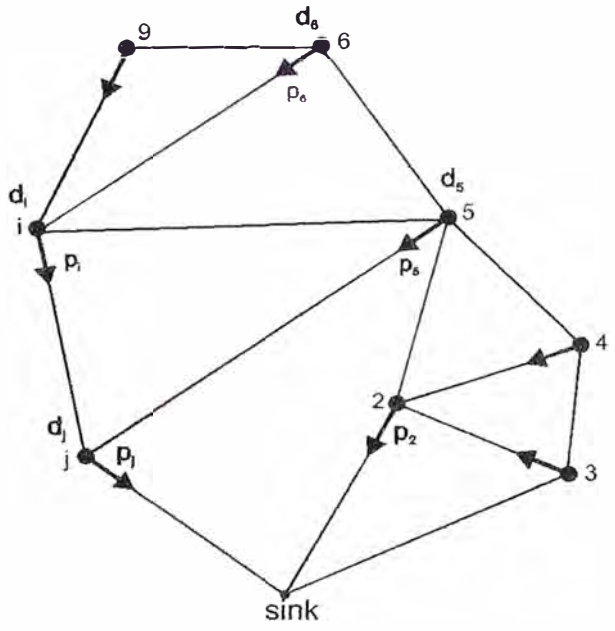


Figura 2.10. Ejemplo de una red usando el protocolo de enrutamiento.

El protocolo opera en ciclos de actualización los cuales son disparados por el nodo SINK, cada ciclo de actualización mejora el camino a SINK, y después que cada ciclo finaliza, el nodo SINK puede iniciar un nuevo ciclo de actualización (mediante un disparo). Después de un número finito de ciclos, los caminos convergen al camino mínimo desde cada nodo a SINK. Como se mencionó antes, cada ciclo de actualización procede en dos fases:

- Fase 1: Los mensajes de control son enviados por la parte superior del árbol, desde SINK a las hojas del árbol P actual, durante esta fase, las distancias estimadas d_i son actualizadas.
- Fase 2: Los mensajes de control son enviados por la parte inferior del árbol a SINK, durante esta fase, nuevos vecinos P preferidos son seleccionados.

En este ejemplo, la topología del protocolo corresponde a la topología de la red, y cada parte corresponde a un nodo de la red. Este protocolo trabaja en cualquier topología; a excepción de SINK, todas las partes ejecutan el mismo algoritmo, luego la descripción de sólo una parte y SINK son necesarias. Los detalles del protocolo de enrutamiento se describen en la Figura 2.11 donde el algoritmo ejecutado por un nodo arbitrario i se muestra. Cada parte (nodo) tiene dos estados: S1 (listo para el siguiente ciclo) y S2 (la fase 1 fue ejecutada, esperando por la fase 2). Cada parte (nodo) tiene dos variables, p_i (vecino preferido del nodo i), d_i (distancia estimada del nodo i al nodo SINK). La conectividad de la red está representada por una lista arbitraria de vecinos L, donde para cada vecino k de L hay:

- una distancia de enlace d_{ik}
- un señalizador $N_i(k)$ el que es:
  - inicializado a NIL,
  - seteado a RCVD cuando la parte i recibe un MSG de k,
  - y reseteado a NIL cuando la fase 2 es completada.
- una variable $D_i(k)$ almacena la última distancia estimada recibida de k.

Cuando un MSG con algún parámetro d es recibido del nodo k por el nodo i, la sentencia FOR es ejecutada, y nuevos valores son almacenados en $N_i(k)$ y $D_i(k)$. Entonces, las transiciones de la máquina de estado finito son ejecutadas si la parte está

en el estado correspondiente y la CONDITION asociada con la transición es cierta. Cuando la transición es ejecutada, el estado es cambiado y la ACTION asociada con la transición es ejecutada. T12 es ejecutada por la parte i (por ejemplo, la fase 1 para el nodo i) cuando recibe un MSG de su p_i, entonces un nuevo d_i se calcula y MSG (d_i) se envía a cada vecino excepto p_i. Como se prueba posteriormente, la estructura de árbol de P garantiza que luego que un ciclo es iniciado, T12 se ejecutará por cualquier nodo. T21 se ejecuta por la parte i (por ejemplo, fase 2 para el nodo i) cuando fue recibido un MSG de cada uno de sus vecinos, entonces i envía MSG(d_i) a sus p_i (permitiendo que la fase 2 se propague por la parte inferior del árbol), actualizando p_i e inicializando los señalizadores N_i. El ciclo termina cuando SINK ejecuta T21. Se dice que el protocolo puede estar en un estado ocioso si no hay mensaje en tránsito, y todas las partes están en S1, y para todos los i,k $N_i(k)$ = NIL. Es fácil mostrar que en un estado ocioso, el único evento que puede ocurrir es la transición de T12 a SINK. Inicialmente, el protocolo está en un estado ocioso y P forma el enrutamiento del árbol a SINK.

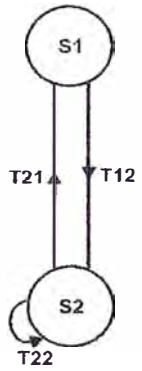


Figura 2.11 Máquina de estados finitos para SINK

2.2. Verificación de protocolos:

Los programas que implementan los protocolos de comunicaciones en computadores pueden exhibir comportamientos extremadamente complicados, puesto que ellos deben competir con agentes de cómputo asíncronos, y la posibilidad de falla en los agentes y en

el medio de comunicación. La mayoría de estudios previos para la verificación de protocolos en redes está basada en los argumentos alcanzados para modelos de estado finito de los protocolos. Esta técnica tiene la ventaja de ser fácilmente automatizada, encontrando dificultades cuando el estado de espacio del protocolo es muy grande. Por ejemplo, los modelos de estado finito presentan dificultades en comunicación con propiedades relativas a corregir transferencias de datos, debido a que la representación de cada valor a ser transmitido puede hacer que el estado de espacio sea extremadamente largo, y posiblemente infinito. Bochmann y Sunshine presentan algunas técnicas para reducir el estado de espacio, con una verificación parcial, combinando o ignorando ciertos estados, usando afirmaciones para clasificar estados, y localizando la investigación, sin chequear todas las partes. Todas estas técnicas consisten en ignorar algunos estados o usar algunas herramientas de estados no finitos, los cuales en general no pueden ser automatizados.

El sistema red/protocolo es modelado por un conjunto de módulos interactivos que representan unidades lógicas del sistema, tales como el medio de comunicación, transmisor y receptor. Hay dos tipos de módulos a ser considerados: proceso y monitoreo. Un proceso es un componente de programa activo, y el monitoreo es una abstracción de datos con sincronización. Las propiedades de proceso y monitoreo son verificadas por un examen de su código. La construcción del sistema de prueba puede usar las propiedades de verificación y puede ignorar la estructura interna de las implementaciones modulares. Por ejemplo, los buffers son un tipo de datos abstractos que pueden ser implementados de diversas maneras. Cualquier implementación encuentra los requerimientos del tipo de datos que pueden usarse en el protocolo sin afectar la prueba de corrección del resto del sistema.

Dos tipos de propiedades, seguridad y actividad, son importantes para sistemas paralelos. Las propiedades de seguridad tienen la forma de "cosas malas no ayudarán", ellas son análogas a correcciones parciales y son expresadas por afirmaciones invariantes las cuales deben ser satisfechas por el estado del sistema en todo instante. Las propiedades de seguridad son expresadas en términos de variables auxiliares que registran la historia de las interacciones de los módulos. Debido a que las variables auxiliares no son implementadas, ellas pueden registrar historias de longitud infinita y son un elemento importante de prueba.

Las pruebas de seguridad son construidas primero verificando las invarianzas de los módulos de nivel más bajo directamente del código, luego mostrando que en conjunción estas invarianzas implican las invarianzas de componentes grandes, llegando a la prueba de la invarianza del sistema completo. Las propiedades de actividad tienen la forma "cosas buenas ayudarán", esto incluye los requerimientos finales en programas secuenciales y propiedades recurrentes en programas sin término, semejante a los sistema operativos. Debido a que la actividad se refiere a la ocurrencia futura del estado deseado, las fórmulas lógicas convencionales, que sólo se refieren a un único estado, son inadecuadas para expresar y razonar acerca de la actividad. Para gestionar con la actividad, se usa la notación de lógica temporal, la cual provee operadores para realizar afirmaciones acerca de los estados futuros del programa. Fórmulas temporales que expresan propiedades de actividad son llamadas comisiones. Las comisiones son verificadas en el mismo estilo modular como invariantes: Primero se verifica las comisiones del módulo de nivel más bajo directamente de su código, y entonces se muestra que, en conjunción, ello implica las comisiones de módulos de niveles más altos.

2.2.1 Herramientas de verificación

Las herramientas de verificación que se estudiarán son las siguientes:

A) Lógica temporal.

B) Historia.

C) Especificaciones modulares.

A) Lógica temporal:

La lógica temporal provee operadores para razonamiento acerca del pasado y futuro, aunque sólo se necesitará operadores para el futuro. En el contexto del programa de verificación, el "futuro" es un programa de computación, esto es, una secuencia de estados que pueden elevarse durante la ejecución del programa. Informalmente, el primer estado en un cálculo representa el presente, y los siguientes estados representan el futuro. Los cálculos no son restringidos para arrancar el comienzo del programa, de modo que el estado "futuro" en un cálculo puede ser el estado "presente" en otro.

Hay dos operadores temporales básicos que son:

□ (futuro) y

◊ (eventualidad).

La fórmula □P (futuro P) significa "P es verdadero para todos los estados del cálculo" (P es verdadero ahora y quedará verdadero siempre).

La fórmula ◊P (eventualmente P) es interpretada como "hay algún estado en el cálculo en el cual P es verdadero" (P es verdadero ahora o llegará a ser verdadero alguna vez).

Las modalidades □ y ◊ son duales, esto quiere decir que se cumple:

$\Box P \equiv \sim \Diamond \sim P$ y

$\Diamond P \equiv \sim \Box \sim P$

Cuando se dice que una fórmula temporal es verdadera para un programa, significa que es verdadera para todos los cálculos del programa.

Los operadores temporales pueden ser usados para expresar las propiedades de seguridad y actividad,

Por ejemplo, el término de un programa, una propiedad de actividad puede expresarse por la fórmula:

a P $\supset$ $\Diamond$ después P, donde "a P" y "después P" son afirmaciones que son verdaderas de establecer en el cual el control está al comienzo o final del programa.

Un ejemplo de una propiedad de seguridad es una afirmación inductiva, esto es, una afirmación que será verdadera si siempre llega a ser verdadera. La siguiente fórmula establece que I es una afirmación inductiva: $\Box$ (I $\supset \Box$ I).

Las combinaciones de las dos modalidades son también usuales, por ejemplo, la fórmula: $\Box \Diamond$ P (P frecuentemente infinito), implica que hay un número infinito de estados futuros para los cuales P es verdadero.

Para entender mejor estas interpretaciones, notar que la expresión:

$\Diamond$ P , implica que P será verdadero en algunos estados futuros.

La fórmula: $\Box \Diamond$ P , establece que esto siempre será verdadero, en particular, si P siempre llega a ser falso, está garantizado que llegará a ser verdadero nuevamente en algún instante posterior, y esto significa que debe ser verdadero un número infinito de veces. El operador $\Box \Diamond$ es especialmente usual para propiedades recurrentes de un programa, por ejemplo: $\Box \Diamond$ (el buffer no está lleno).

También usarán los cuantificadores universal y existencial, así la afirmación:

$\exists$ n ($\Diamond$ (x = 2.n)) , establece que x eventualmente llegará siempre. Notar que n en esta fórmula no es un programa variable, y no se usarán fórmulas en las cuales las

cantidades variables son también usadas como variables del programa en el programa bajo consideración.

Por otro lado, consideremos la afirmación:

$$\forall i (x = i \supset \Box x \geq i)$$

Esta afirmación establece que x sólo toma un valor al menos tan grande como el valor inicial, sin importar cual fue su valor inicial.

B) Historia:

La prueba usa la historia de las variables para guardar la secuencia de mensajes que están en la entrada y salida de los módulos del sistema. La historia de las variables tienen frecuentemente usos en los argumentos acerca de los sistemas de comunicaciones. El valor inicial de la historia de una variable es una secuencia completa, y la única operación permitida está seguida de un nuevo valor.

Supóngase que A y B son las historias de variables, se denota que A es una sucesión inicial de B mediante la siguiente expresión:

$$A \prec B$$

esto significa que $|A| \leq |B|$, y las dos sucesiones son idénticas en sus primeros | A | elementos, donde | A | denota la longitud de la historia A.

Si X es una historia variable, la siguiente afirmación es verdadera para algún programa que contenga X:

$$\forall A \Box (A = X \supset \Box (A \prec X))$$

Esta afirmación establece que si hay algún punto en el cual X tiene el valor A, entonces todas las sucesiones de tiempos A están en una sucesión inicial de X. Esto sigue del hecho que la única operación en una historia de variables está esperando un nuevo

valor. Si ahora A y B son secuencias de historias arbitrarias, y si A tiene elementos u, v, y, z entonces se puede escribir:

$A = \langle uvyz \rangle$.

Si $| A | = n$ entonces se puede escribir: $A = \langle a_i \rangle_i^n = 1$

Se denota la concatenación de secuencias por yuxtaposición, esto es: $A = \langle uv \rangle \langle yz \rangle = \langle uvyz \rangle$. Finalmente, hay cierta afirmación temporal acerca de historias que se usan para razonar acerca de actividad. La primera es una afirmación que el tamaño de una historia dada aumentará con el límite. Esto se abrevia como u(A), donde:

$u(A) = \forall n (\Diamond (| A | > n))$, lo cual es equivalente a:

$u(A) = \forall n (\Box (| A | = n \supset \Diamond | A | > n))$

La segunda afirmación, establece que un valor particular ocurre en un número ilimitado de veces en la historia, donde c(A, m) es el número de ocurrencias de m en A, y se tiene: $uc(A, m) = \forall n (\Diamond (c (A, m) > n))$.

C) Especificaciones modulares:

Una especificación modular involucra tres tipos de información. Primero, las propiedades de seguridad están dadas por invariantes: afirmaciones acerca de los módulos variables que son verdaderos todo el tiempo en el cálculo. Segundo, las propiedades de actividad son especificadas por afirmaciones lógicas temporales llamadas comisiones, las cuales describen condiciones que el proceso causa para ser verdadero. Finalmente, los servicios provistos por el módulo a otros módulos son descritos por pre, post y afirmaciones en vivo acerca de cada operación. Las pre y post afirmaciones dan seguridad (corrección parcial) de las propiedades de las operaciones. Si la precondición se mantiene cuando la operación es invocada, y si termina, entonces un término de post condición debe mantenerse. Las variables en estas afirmaciones deben ser privadas para

el proceso que invoca la operación, y detectan que ningún otro proceso puede modificar sus valores. Esto permite la complejidad que puede llevarse cuando la comunicación con variables son distribuidas en varios procesos. La afirmación describe el efecto que la operación causa cuando esta es invocada, y esto puede involucrar variables que pueden ser modificadas por otros procesos.

Verificando que un sistema que recibe estas especificaciones es hecho en varias fases. Una fase es para examinar el código de bajo nivel de los módulos, y probar que satisface las especificaciones. En la otra fase, las especificaciones de módulos componentes se verifican de las especificaciones de sus componentes. La verificación de las propiedades de seguridad a partir del código, es una tarea muy sencilla. Ahora, para mostrar que una afirmación es invariante, uno debe mostrar que es verdadera inicialmente, y que es preservada por cada acción del módulo bajo consideración (debido a que todas las invarianzas de salida involucran sólo variables locales y privadas, y no hay necesidad para considerar interferencia de otros módulos). Así, cuando probamos que P es invariante, tenemos que probar que la afirmación temporal cumple: $Int' \supset \Box P$, donde Int´ es una afirmación que describe el estado inicial del programa. Probando que la pre y post afirmación de operaciones está esencialmente verificando las correcciones parciales de los códigos de operación.

Las propiedades de actividad de módulos (comisiones y afirmaciones) son probadas desde el código usando axiomas y reglas de interferencia acerca de las propiedades de actividad de las sentencias del programa, expresadas en lógica temporal. Notar que cuando se dice que P es una comisión, significa que: $Int \supset \Box P$, semejante que con invariantes. De modo que, las comisiones tienen fórmulas temporales más complejas que las invariantes. Por ejemplo, las comisiones tienen la forma: $P \supset \Diamond Q$.

Para establecer que una afirmación es una comisión, significa que en algún cálculo de partida, en un estado inicial legítimo, mientras que P llega a ser verdadero, Q será verdadero en el mismo instante o posteriormente. Los módulos componentes son verificados mostrando que las invarianzas y comisiones del módulo son implicadas por las invarianzas y comisiones de sus componentes. En esta etapa no hay necesidad para considerar el código.

2.2.2 Protocolo de transferencia de datos de Stenning:

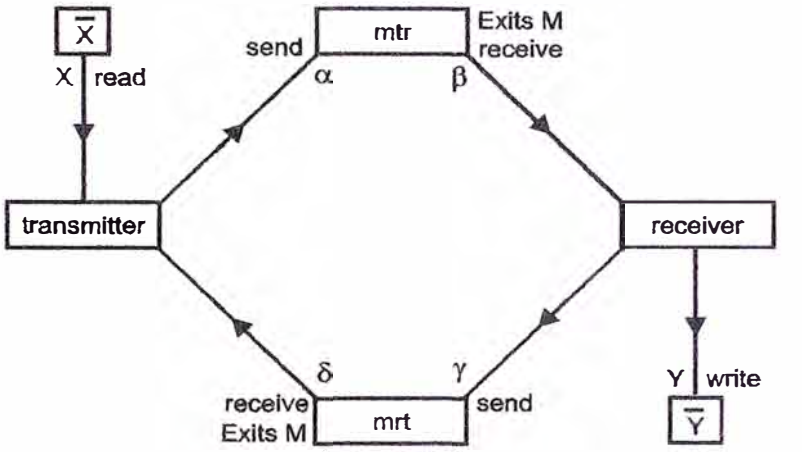


Figura 2.12 Diagrama del sistema para el protocolo de Transferencia de datos de Stenning.

Para ilustrar la aplicación de estas técnicas de verificación de programas para protocolos de comunicaciones, se discutirá una versión simplificada del protocolo de transferencia de datos de Stenning. El protocolo se requiere para liberar todos los mensajes de entrada en el orden en el cual son presentados. Stenning verifica las propiedades de seguridad del algoritmo, usando una técnica de prueba no modular, sin considerar actividad. En la Figura 2.12 se ilustra el diagrama de la estructura de red del protocolo de Stenning. El protocolo se compone de tres procesos: un transmisor, un receptor y un medio de comunicación. El transmisor opera cuando una secuencia de mensajes ilimitado de entrada X desde la fuente X′, lo envía desde el receptor, a través del medio de comunicación *mtr*, el mensaje de salida desde el receptor Y′ (la sucesión de

salida asociada se denota por Y) que el receptor reconoce a través del medio de comunicación *mrt*. Las complicaciones llegan debido a que el medio de comunicación es irrealizable, los mensajes pueden perderse, duplicarse o reordenarse (se asume que puede ocurrir la corrupción de los mensajes, que es detectado por un mecanismo de verificación de bajo nivel, en el que los mensajes corruptos son descargados).

El protocolo debe asegurar que los mensajes están últimamente liberados en forma correcta a pesar de ser irrealizables, y esto es acompañado conectando una secuencia de números al mensaje enviado por el transmisor y el envío del reconocimiento por el receptor. El transmisor envía cada mensaje repetidamente hasta que recibe un reconocimiento de este mensaje, usando un mecanismo de temporización para disparar la retransmisión. Primero el receptor consigue un mensaje con una sucesión de números dados, lo almacena en la fuente de salida, también envía al transmisor un reconocimiento para cada mensaje recibido. Los nombres de las historias de las variables usadas en al prueba se indican en el diagrama de la red de la figura anterior, como se mencionó, X es la historia de entrada del transmisor (y del sistema entero), e Y es la salida del receptor (y del sistema). Las historias de entrada y salida del mensaje son α y β respectivamente, mientras que las historias de entrada y salida del reconocimiento son γ y δ. Se denotan las secuencias ilimitadas de ítems a ser transmitidos por: $D= d_1,d_2,...$

Estos son los valores obtenidos de X', y ellos aparecen en la historia de entrada de X. Notar que D es una secuencia constante global, la cual puede referirse en la prueba de algún módulo. Por otro lado, X es una variable local del módulo transmisor. Un mensaje conteniendo la secuencia de un número i y el ítem d_i es denotado por M_i, esto es: $M_i = [i,d_i]$. Esta es la forma de mensajes en α y β. Un reconocimiento para el mensaje i, el par

[i,"ack"] se denota por A_i , los mensajes en γ y δ tienen esta forma. Notar que d_i, M_i y A_i son constantes, cuyos valores no cambian a través de la ejecución del programa.

A continuación se analiza los siguientes tópicos del protocolo de Stenning:

A) Medio de comunicación.

B) Seguridad: transmisor y receptor.

C) Seguridad del sistema.

D) Actividad del transmisor y receptor.

E) Actividad del sistema.

A) Medio de comunicación.

El medio de comunicación usado por el protocolo no está definido por el código del programa, es esencialmente una caja negra alrededor de la cual se tiene la información limitada. De hecho, lo que se quiere conocer del medio son sus especificaciones, y estas especificaciones pueden verificarse examinando el código de los componentes de bajo nivel del sistema, así como las especificaciones del transmisor y receptor que puedan verificarse desde su código. Recalcando que la especificación del módulo involucra tres tipos de información: Invariantes, comisiones y especificaciones de servicio, debido a que el medio de comunicación considerado es irrealizable, tiene una invariante muy débil: *nada llega afuera, sino fue puesto dentro.* $m \in \beta \supset m \in \alpha$, $m \in \delta \supset m \in \gamma$.

Notar que m es una variable libre, no un programa variable, y observamos la convención usual que las variables libres están universalmente cuantificadas, así, *mtr1* establece que cada mensaje en β es también un mensaje en α. Estas afirmaciones de seguridad describen un medio en el cual puede perderse, duplicarse y reordenarse mensajes. Los invariantes anteriores serán satisfechos por un medio que nunca libera algún mensaje, y en este caso ninguna salida aparecerá. El medio asumido por protocolo de Stenning tiene dos comisiones independientes que garantizan que algún mensaje

último conseguirá pasar. El primero es una afirmación que si algún número de mensajes ilimitado son enviados, entonces los mensajes están ilimitadamente disponibles en el receptor: $u(\alpha) \supset \Box \Diamond$ mtr. Existe M, $u(\gamma) \supset \Box \Diamond$ mrt.Existe M .

En estas afirmaciones se usa la función media "Existe M", la cual retorna el valor "verdadero" si al menos un mensaje está disponible. Un módulo función en una afirmación es interpretada como verdadera de un estado si la función retorna "verdadero" cuando es invocada en este estado. La segunda comisión afirma que si el mismo mensaje es enviado una y otra vez, eventualmente será liberado (previendo que el receptor procesa los mensajes aceptados). Esta comisión se expresa por:

$$(uc(\alpha, m) \wedge u(\beta)) \supset \Diamond\, m \in \beta$$

$$(uc(\gamma, m) \wedge u(\delta)) \supset \Diamond\, m \in \delta.$$

Finalmente, se debe especificar las operaciones provistas por el medio, en este caso hay tres: enviar un mensaje, recibir un mensaje y chequear para observar mientras algún mensaje está esperando ser recibido. Las especificaciones de estos servicios se dan por el medio mtr, las especificaciones para mrt son esencialmente las mismas.

enviar (m)
pre: $\alpha = A$
post: $\alpha = A(m)$
listo: $\Diamond$ después mtr.enviar
recibir (var m)
pre: $\beta = B$
post: $\beta = B(m)$
listo: $\Diamond$ mtr. Existe M $\supset \Diamond$(después mtr. recibe)
Existe M
pre: verdadero
post: verdadero
listo: $\Diamond$ después mtr. Existe M.

Notar que una operación de envío siempre termina, y recibir termina si un mensaje está disponible. La pre y post afirmaciones de mtr. Existe M son ambas verdaderas, lo cual no da seguridad de información acerca de la operación, de hecho, sólo se usará mtr.

Existe M en el razonamiento acerca de la propiedad de actividad que una operación de recibir termina. El tiempo es otra caja negra, y se define sus propiedades de una manera similar.

B) Seguridad: transmisor y receptor.

Las especificaciones de seguridad de procesos están dadas por afirmaciones invariantes acerca de las variables del proceso. Para verificar un proceso invariante, se observa su estado inicial, y si este es preservado por cada operación del proceso, y este es un proceso de verificación secuencial progresiva. Aquí sólo se dará algunos pasos de verificación. La especificación de seguridad del transmisor consiste de dos invariantes expresadas en términos de M_i y A_i, definidas anteriormente:

$$\exists\, n(\, X = \langle\, d_i \,\rangle^{n}_{i=1} \wedge (m \in \alpha \supset \exists\, i \leq n(m = M_i))$$

$$|\, X \,| \geq k > 1 \supset A_{k-1} \in \delta$$

El primer estado invariante cuando n elementos ingresan al transmisor, la salida al medio contiene sólo mensajes que corresponden a estos n elementos con números conectados en secuencia. La invarianza de T1 puede ser provista, esto es mantenida inicialmente (cuando todas las secuencias están completas) y es preservada en cada operación del transmisor. El término de entrada k-ésimo del estado invariante T2, no se lee hasta después del reconocimiento que el mensaje (k-1)-ésimo ha sido recibido. Esto es obvio desde el código del transmisor. El receptor tiene dos invariantes, los cuales son similares a los del transmisor:

$$\forall\, m(m \in \beta \supset \exists\, j\ (m = M_j)) \supset (\exists\, n\ (Y \prec \langle d_i \rangle_{i=1}^{n}) \wedge d_k \in Y \supset M_k \in \beta)$$

$$A_i \in \gamma \supset (M_i \in \beta \wedge |\, Y \,| \geq i\ )$$

Hablando claramente, la salida Y del receptor en el estado invariante R1 será legitimada si su entrada β es legitimada. Más precisamente, si β contiene sólo

mensajes de la forma (i, d_i), entonces Y es una secuencia de datos $\langle d_i \rangle_{i=1}^{n}$, y cada dato en Y corresponde a un mensaje que aparece en β. Para lograr que el receptor satisfaga esta invarianza, se agrega el elemento i-ésimo a Y sólo después que se recibe un mensaje con secuencia del número i, y el valor que se agrega a Y es uno contenido en el mensaje. Se asume que cada mensaje en β tiene la forma (i, d_i), así el i-ésimo elemento de Y debe ser d_i. El segundo estado invariante i reconocido está en al historia de salida γ , luego el mensaje i está en la historia de entrada β, y por lo tanto el dato asociado d_i está en Y, esto es correcto analizando el flujo de control en el receptor, por lo que un reconocimiento se envía después que el correspondiente mensaje ha sido recibido y su dato adicional a Y.

C) Seguridad del sistema.

Las especificaciones de seguridad del sistema se dan en las afirmaciones invariantes: $Y \prec X$.

Esta afirmación establece que los valores de salida están en una secuencia inicial de los valores de entrada, lo cual no implica que algún valor de salida nunca sea producido, estos requerimientos están dados por las especificaciones de actividad. Se procede asumiendo la invarianza para el transmisor y receptor, y el medio de comunicación, y mostrando que el sistema invariante debe operar. Un primer paso, consiste en notar que la hipótesis de afirmación R1 del receptor es:

$$\forall\, m(m \in \beta \supset \exists\, i\, (m = M_i))$$

que sigue inmediatamente de las propiedades de seguridad del transmisor y del medio. Debido a que el transmisor sólo coloca mensajes legítimos en el medio (T1), y cualquier mensaje que sale del medio, debe haber sido puesto por el transmisor (mtr1), el receptor sólo puede obtener mensajes legítimos.

Dado $n = \text{máx}\ \{ i : M_i \in \beta \}$. Debido a la hipótesis de que R1 es satisfecha, se conoce que la conclusión de mantener R1, es llamada: $Y \prec \langle d_i \rangle_{i=1}^{n} = 1$. Pero por mtr1:

$M_n \in \beta \supset M_n \in \alpha$, y T1 implica que si $M_n \in \alpha$, entonces $|X| \geq n$, y así:

$M_n \in \alpha \supset (\langle d_i \rangle_{i=1}^{n} \prec X)$.

Así, se puede concluir que: $Y \prec \langle d_i \rangle_{i=1}^{n} \prec X$, lo cual implica la afirmación de la seguridad del sistema S1.

D) Actividad del transmisor y receptor.

Las propiedades de actividad del transmisor y receptor se dan por comisiones, verificando que un proceso satisface sus propiedades requeridas de especificaciones de actividad, razonando en base de suposiciones acerca de las propiedades de actividad de las sentencias del programa. Las reglas formales para proveer las propiedades de actividad del código de programa fueron dadas por Owicki y Lamport, y pruebas más detalladas por Hailpern. Se asume que el proceso se ejecuta suavemente, esto es, cada proceso progresa a menos que sea bloqueado. Más precisamente, si **s** es una acción no bloqueable en el programa, si **s** es la afirmación de que **s** está listo a ser ejecutado, y luego **s** es la afirmación que el control a terminado para **s**, y estas afirmaciones básicas de actividad pueden expresarse en lógica temporal por: en **s** $\supset \Diamond$ después **s**.

En el protocolo del sistema se discutió que un proceso sólo puede llegar a ser bloqueado mientras es difícil ejecutar una operación de recepción en un medio de comunicación que no está disponible para ser recibido. La especificación de actividad para el estado de operación de recepción (de mtr) que recibe, terminará si un mensaje está disponible, esto es: (en mtr. recibe $\wedge \Diamond$ mtr. Existe M) $\supset \Diamond$ después mtr. recibe.

Partiendo de que estas afirmaciones acerca de las acciones del programa, se pueden derivar reglas para proveer propiedades de actividad de sentencias de programas grandes. Por ejemplo, si se considera una sentencia de programa de la forma:

loop S end loop

donde **S** es una sentencia que no contiene algún lazo o acción que pueda ser bloqueada, y para tal sentencia se puede usar: en S $\supset \Box \Diamond$ en S, esto es, controlará ilimitadamente al comienzo de S, y esto es exactamente la forma del lazo en el programa transmisor. Por otro lado, se considera el programa receptor, y aquí nuevamente se tiene un lazo, llamado **S´** que contiene la sentencia mtr.recibir, la cual puede ser bloqueada. Para este lazo se tiene:

(en S´ $\wedge \Box \Diamond$ mtr. Existe M) $\supset \Box \Diamond$ en S´. La afirmación "$\Box \Diamond$ mtr. Existe M" garantiza que mientras mtr.recibir está empezando, un mensaje eventualmente estará disponible de modo que pueda terminar la ejecución. Así, el receptor no puede estar permanentemente bloqueado, y el lazo es ejecutado ilimitadamente. La mayoría de propiedades generales de actividad incluyen el efecto de las acciones del programa en las variables del programa. Por ejemplo, de la pre y post afirmación de envío, más el hecho que el envio nunca puede ser bloqueado, podemos concluir:

$$(\text{en mtr.envio} \wedge |\alpha| = k) \supset \Diamond (\text{después mtr.envio} \wedge |\alpha| = k+1)$$

Para el transmisor, en el cual mtr.envio está contenido en un lazo cuyo cuerpo es ejecutado indefinidamente, podemos concluir:

$$\Box (|\alpha| = k \supset \Diamond |\alpha| = k + 1)$$

el cual implica $u(\alpha)$.

Ahora vamos a considerar las especificaciones de actividad del transmisor, que consisten de tres comisiones. Primero, la historia de salida del transmisor α crece sin

límite: u(α). Estas comisiones son independientes de cualquier suposición acerca del entorno, y para ver que se satisface, notamos que el código del transmisor está en un lazo repetitivo que nunca puede ser bloqueado: la única operación que puede causar bloqueo es "recibir", y "recibir" sólo es ejecutada cuando una aclaración está disponible. Dado que no hay un bloqueo, el mecanismo de tiempo de garantiza que un mensaje se envíe fuera al menos una vez cada intervalo de tiempo.

La segunda comisión del transmisor es:

$\Box \Diamond$ (mrt.existe M) $\supset$ u(δ)

Este estado del transmisor incrementará el tamaño de δ proveyendo el entorno para asegurar que se realice un reconocimiento de disponibilidad en mrt. Esto sigue de la ausencia de bloqueo, y el hecho de que el transmisor aceptará un reconocimiento cada instante alrededor de su lazo (si está disponible).

La tercera comisión es una esperanza de iniciar el envío del siguiente dato así como el actual a sido reconocido:

$$\forall i(A_i \in \delta \supset | X | \geq i) \supset \forall j(A_j \in \delta \supset (uc(\alpha, M_{j+1}) \vee \Diamond (A_{j+1} \in \delta))).$$

La hipótesis de estas comisiones es una afirmación que el resto del sistema debe satisfacer: Un reconocimiento para el mensaje i no es recibido antes que el transmisor haya empezado a trabajar un mensaje i. Bajo esta suposición, una vez que el transmisor recibe un reconocimiento j, se inicia un mensaje de envío j+1, y se envía este mensaje un número ilimitado de veces a menos que eventualmente reciba un reconocimiento j+1.

A continuación consideremos las especificaciones de actividad para el receptor. Nuevamente tenemos tres comisiones, y ellas son completamente similares a las comisiones del transmisor. Primero, el receptor provocará β y γ para crecer ilimitadamente tan grande como esté habilitado para recibir mensajes de mtr.

$\Box \Diamond$ mtr.existe $M \supset u(\gamma)$

$\Box \Diamond$ mtr.existe $M \supset u(\beta)$.

El código del receptor satisface estas afirmaciones porque la disponibilidad repetida de mensajes implica que el receptor no puede ser bloqueado en la operación de recepción. Más aún, repetidamente ejecuta su lazo, y cada vez incrementa la longitud de β y γ. Notar que la comisión T3 , la cual corresponde a R3, no requiere asumir acerca del resto del sistema para garantizar que el tamaño de α siga creciendo. La diferencia entre T3 y R3 viene del hecho que el transmisor usa un mecanismo de tiempo y el receptor no.

La tercera comisión del receptor es un reconocimiento de cada mensaje recibido:

$(\forall i(M_i \in \beta \supset | Y | \geq i\text{-}1) \wedge u(\beta)) \supset \forall j\ [M_j \in \beta \supset (\Diamond(|Y| \geq j) \wedge ((uc(\gamma, A_j) \vee \Diamond(M_{j+1} \in \beta)))]$

Esta comisión es análoga a la comisión del transmisor T5.. Asumiendo que el mensaje i no llegue hasta que el receptor haya procesado el mensaje i-1, y que β crezca desmedidamente, el receptor reconocerá cada mensaje recibido hasta el siguiente arribo, y agregará d_k a la secuencia de salida Y.

(Esto es necesario para asumir $u(\beta)$ porque el receptor puede bloquearse si los mensajes no llegan, de modo que una suposición es innecesaria para el transmisor, debido a que nunca puede bloquearse)

E) Actividad del sistema.

Las propiedades de actividad del sistema finalmente requieren probar que cada mensaje está eventualmente en la salida. Debido a que las propiedades de seguridad dicen que cualquier salida producida está en un segmento inicial de la secuencia de entrada, se necesitará establecer que el flujo de salida tiene longitud arbitraria, esto es: $u(Y)$.

El primer paso es probar que todas las variables de la historia del medio crecen indefinidamente, y esto es consecuencia de la comisión y del medio.

$u(\alpha)$	(T3)
$u(\alpha) \supset \Box \Diamond$ (mtr.existe M)	(mtr2)
$\Box \Diamond$ (mtr.existe M) $\supset u(\gamma)$	(R3)
$\Box \Diamond$ (mtr.existe M) $\supset u(\beta)$	(R4)
$u(\gamma) \supset \Box \Diamond$ (mrt.existe M)	(mrt2)
$\Box \Diamond$ (mrt.existe M) $\supset u(\delta)$	(T4)

En combinación, estas afirmaciones implican que toda la historia secuencial crece sin límite, esto es: $u(\alpha) \wedge u(\beta) \wedge u(\gamma) \wedge u(\delta)$, y la entrada está infinitamente disponible para mrt.recibir y mtr.recibir.

2.3. Influencia de los protocolos en la estabilidad de las redes de computadoras

En una red de paquetes conmutados el flujo de datos debe mantenerse dentro de los límites compatibles con la cantidad de recursos disponibles. El conjunto de las tasas de entrada para los cuales todos los mensajes alcanzan su destino, determina el dominio de la estabilidad de la red. De la definición de Rudin, el control de flujo, es una colección de algoritmos los cuales son usados en una red para prevenir a un único usuario o a un único grupo de usuarios el uso de la tabla de anuncios de los recursos de la red para el perjuicio de los otros. Dentro del control de flujo se define el control de flujo End-to-End (E-E) y el control de congestión. El control de flujo E-E es un conjunto de mecanismos por lo cual un receptor de punto terminal mantiene el tráfico del transmisor dentro de límites compatibles con la cantidad de recursos; mientras que el control de congestión es el conjunto de mecanismos por el cual un subsistema intermedio mantiene el tráfico de entrada dentro de límites compatibles con la cantidad de recursos disponibles para el subsistema.

La palabra congestión será usada para denotar un estado del sistema donde haya una sobrecarga de demanda para el servicio disponible. No congestión y estabilidad pueden parecer sinónimos, pero la estabilidad se refiere al estado estable y la congestión al comportamiento transitorio. Para permitir comparaciones y evaluaciones, la medida de la performance de un sistema debe estar definido, el único principio usado aquí es la Máxima (más grande) Tasa de Entrada (MTE) que los recursos disponibles pueden servir. El conocimiento de la máxima tasa de entrada nos permite ser usada para evitar la congestión, y nos asegura que debajo de este valor (en un sistema sin bloqueo) la congestión desaparecerá después de un lapso de tiempo, y además se puede notar que la máxima tasa de entrada es igual a la máxima transferencia del sistema (MTE=MTS).

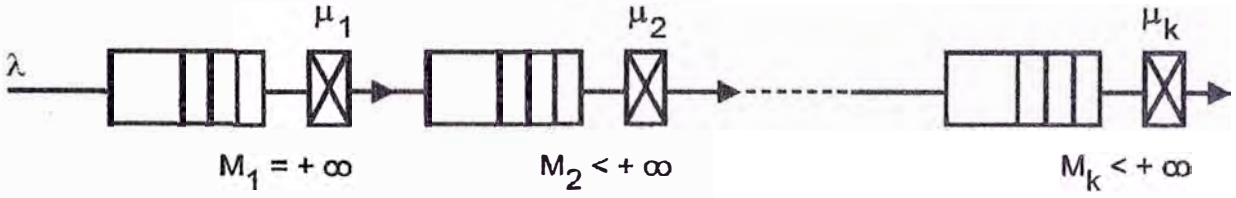


Figura 2.13. Colas en Tandem.

Hay varios estudios concernientes a la máxima tasa de entrada para un sistema general de colas en tandem como las mostradas en la Figura 2.13, pero todos ellos son para casos exponenciales. Las evaluaciones de performance usando otros criterios también están disponibles. Las medidas de la performance son frecuentemente expuestas en el retardo promedio de las redes por un paquete, y sobreflujo.

2.3.1 Modelo matemático: El propósito consiste en evaluar, con métodos matemáticos, la máxima tasa de entrada (MTE) en una red de computadoras. Como primer paso, se estudia un camino particular dentro de la red, siendo este un caso particular importante, y la MTE está definida. El sistema consiste de K colas (descritas en la Figura 2.13). Un cliente (correspondiente a un paquete de datos) va a través del sistema pasando la $(i+1)$-

ésima cola después del i-ésimo dispositivo para $1 \leq i \leq K$. Todas las estaciones, excepto la primera tienen una capacidad limitada a M_i, y los clientes van a través de las estaciones en el orden de su arribo (primero que llega-primero servido). Se asume que los tiempos de servicio tienen distribuciones arbitrarias, y la capacidad finita de la cola corresponde a las facilidades de almacenamiento en cada nodo. Se asume que la primera estación es la principal y es apta para su capacidad infinita (nunca hay pérdida de paquetes). La estructura de colas en tandem representa un camino peculiar en la red, reservada para la transmisión entre dos puntos; donde los arribos externos pueden permitirse en cada nodo, y ellos abandonan al sistema después del nodo. Se asume que el tráfico del enlace principal no se afecta por estos arribos externos, así estos arribos pueden ser despreciados. Se determinan dos clases de protocolos cuando una transmisión falla por falta de espacio de almacenamiento:

- El paquete es retransmitido desde la estación precedente, hasta que sea un éxito.
- El paquete es retransmitido desde la primera estación.

El primer protocolo es uno de nodo a nodo del tipo "envío y espera" con reconocimiento positivo y negativo. El paquete es retransmitido hasta la recepción de un reconocimiento positivo, este caso corresponde al control local de tráfico de Pennotti y Schwartz, donde el paquete es bloqueado en el nodo y en el tiempo principal el servidor estará permitido a servir a clientes externos. Los dos son idénticos si el tiempo de servicio es exponencialmente distribuido, donde se asume que el tiempo máximo y el tiempo de servicio del reconocimiento están incluidos en el tiempo de servicio. El segundo protocolo es uno de terminal a terminal, donde el tiempo se inicia con la entrada de un paquete en la red de paquetes conmutados y una copia es retransmitida al final del tiempo si un reconocimiento positivo no llega.

2.3.2. Máximo tráfico de entrada cuando las retransmisiones se realizan desde la estación precedente:

Se usarán especialmente aproximaciones, obtenidas por procesos de difusión, una estación (limitada por la capacidad de la cola con servicio y tiempos de interarribo que siguen leyes generales) se estudia por una equivalencia con un sistema de dos estaciones cerradas. En la conclusión, se asume que la probabilidad p_n de que el paquete sea rechazado en la entrada de la estación *n* es igual a la probabilidad de que haya M_n paquetes en la estación n. Tal condición también la hacen Pennotti y Schwartz, y Lam.

$\mathbf{p_n = P(M_n)}$ donde:

p_n es la probabilidad de que el paquete sea rechazado a la entrada del nodo *n*.

M_n es la capacidad de almacenamiento en el nodo *n*.

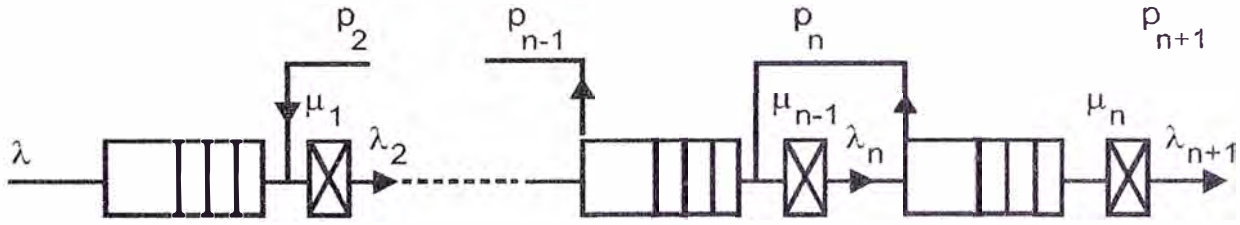


Figura 2.14. Colas en tandem con retransmisión desde la estación precedente.

Se denota por:

$f_n(x)$ la distribución de probabilidad de los tiempos de servicio en la estación *n*.

Esto será equivalente para definir la distribución de longitud de paquetes conociendo la capacidad del enlace.

Se denota:

La media de f(x) por E[f(x)],

La variancia de f(x) por Var[f(x)] y

El coeficiente de variación cuadrático (CVC) por Ks[f(x)].

Se denota:

$\mu_n^{-1} = E[f_n(x)]$ como la media de $f_n(x)$,

$Var_n = Var[f_n(x)]$ como la variancia de $f_n(x)$, y

$Ks_n = Ks[f_n(x)]$ como el coeficiente de variación cuadrático (CVC) de $f_n(x)$,

donde $f_n(x)$ la distribución de probabilidad de los tiempos de servicio en la estación *n*.

En la Figura 2.14, se muestra el modelo que se quiere estudiar.

Se denota por:

λ_n la tasa de entrada para la estación n (tasa de interarribo), antes del rechazo.

Ka_n el CVC de tiempos de interarribo para la estación *n*, antes del rechazo, el cual ocurre con la probabilidad p_n.

p_n es la probabilidad de que el paquete sea rechazado a la entrada del nodo *n*.

El arribo de clientes a la primera estación se asume que ocurre de acuerdo a un proceso general con una tasa de entrada λ. Para analizar el comportamiento de la estación n, se reemplaza por una cola equivalente en la cual no hay realimentación, y así un nuevo tiempo de servicio es el tiempo de residencia total del cliente en la estación de servicio.

Sea: $h_n(x)$ la distribución del tiempo de servicio equivalente,

se define: $\mu_n^{-1} = E[h_n(x)]$ como la media de $h_n(x)$, y

$Ks_n = Ks[h_n(x)]$ como el CVC de $h_n(x)$,

con lo cual se obtiene:

$$h_n(x) = (1 - p_{n+1}) \sum_{k=1}^{\infty} p_{n+1}^{k-1} f_n^{*k}(x)$$

donde: * es el producto convolución y p_{n+1} es la probabilidad de que el paquete sea rechazado en la entrada de la estación $(n+1)$.

$f_n(x)$ es la función de distribución de probabilidad de los tiempos de servicio en la estación *n*.

Esto corresponde al hecho que con la probabilidad $(1 - p_{n+1})\ p_{n+1}^{k-1}$ el cliente es servido *k* veces. De modo que se obtiene:

$$\mu_n = \mu_n \ (1 - p_{n+1})$$

$$Ks_n = p_{n+1} + Ks_n \ (1 - p_{n+1})$$

También es necesario conocer la tasa de interarribo en la estación n (λ_n) y el CVC de tiempos de interarribo Ka_n .

La tasa de interarribo de paquetes en la estación n (λ_n) es igual a $\lambda/(1-p_n)$.

$$\lambda_n = \lambda/(1-p_n)$$

donde λ es la tasa de entrada del arribo de clientes a la primera estación.

El número de paquetes que fluyen desde la estación (*n-1*) a la estación n (λ_n), es la suma de nuevas transmisiones desde la estación (*n-1*) y del número de paquetes retransmitidos desde la estación (*n-1*), luego se tienen tres posibilidades para aproximar el CVC de tiempos de interarribo Ka_n:

* El método de Kobayashi y Reiser.
* El método de Gelenbe y Pujolle
* El método de Sevcik

Considerando:

Ka_n como el CVC de tiempos de interarribo de paquetes para la estación *n*, antes del rechazo.

$Ks_n = Ks[h_n(x)]$ como el CVC de $h_n(x)$.

$h_n(x)$ como la distribución del tiempo de servicio equivalente.

p_{n-1} como la probabilidad de que el paquete sea rechazado en la entrada de la estación (*n-1*).

ρ_n es el tráfico en la estación *n* y ρ_{n-1} es el tráfico en la estación (*n–1*)

El método de ***Kobayashi y Reiser*** da una formulación sencilla:

$$Ka_n = Ks_{n-1}$$, pero es menos precisa que las otras.

El método de ***Gelenbe y Pujolle*** da:

$$Ka_n = -1 + \rho_{n-1}^2 \ (Ks_{n-1} + 1) + (2\rho_{n-1} + 1 \ Ka_{n-1})(1 - \rho_{n-1})$$

El método de ***Sevcik***:

$$\mathbf{Ka_n = -1 + \rho_{n-1}^2 (Ks_{n-1} + 1) + (2\rho_{n-1} + 1 + Ka_{n-1})(1 - \rho_{n-1}^2)}$$

Donde ρ_n es el tráfico en la estación n, y

$\rho_n = \lambda_n / \mu_n$.

Cuando ρ_n tiende a 1, las tres formulaciones son idénticas a: $Ka_n = Ks_{n-1}$.

Cuando se tenga una expresión pequeña, se usará la primera formulación, de otro modo la segunda formulación se tomará.

Como sabemos la probabilidad p_n que un paquete sea desechado a la entrada de la estación *n* es igual a la probabilidad de que haya M_n paquetes en la estación *n*. Este valor se obtiene de Gelenbe:

$$\mathbf{p_n = P(M_n) = \rho_n (1 - \rho_n) / [e^{-\gamma_n (M_n - 1)} - \rho_n^2]} \qquad \mathbf{(1)}$$

Donde:

$\rho_n = \lambda_n / \mu_n$

$\gamma_n = 2b_n / \alpha_n$

con $b_n = \lambda_n - \mu_n$

$\alpha_n = \lambda_n Ka_n + \mu_n Ks_n$

Luego el comportamiento de estación n estará definida por la ecuación (1) $P(M_n)$ - la probabilidad de que haya M_n paquetes en la estación *n* y, (2) λ_n - la tasa de interarribo de paquetes en la estación n.

$$\mathbf{\lambda_n = \lambda/(1-p_n)} \qquad \mathbf{(2)}$$

Las ecuaciones (1) y (2) tienen como variables λ_n y p_n en los cuales la probabilidad p_{n+1} de que el paquete sea rechazado en la entrada de la estación $(n+1)$ interfiere. Podemos estudiar el sistema, comenzando con la última estación, debido a que $p_{K+1} = 0$, así, se obtiene paso por paso λ_n y p_n desde $n = K$ a $n = 2$ para un λ dado.

Ahora se denota por λ_{max} la máxima tasa de entrada y por ρ_{max} el máximo tráfico, de $\rho_n = \lambda_n / \mu_n$ obtenemos:

$$\rho_{max} = \lambda_{max} / \mu_1 .$$

Despreciando las pérdidas en los nodos intermedios, las estaciones 2, 3, ... , *K* son estables, así, la máxima tasa en la entrada se determina precisamente cuando la estabilidad de la primera estación falla.

Esta es teóricamente formulada por:

$$\lambda_{max} = \mu_1 \qquad o \qquad \rho_{max} = 1$$

donde una formulación equivalente es:

$$\lambda_{max} = \mu_1 (1 - p_2) \qquad o \qquad \lambda_2 = \mu_1$$

Construido a partir de:

$$\lambda_n = \lambda/(1-p_n) \rightarrow \lambda = \lambda_n (1-p_n)$$

para $n = 2$:

$$\lambda = \lambda_2 (1-p_2) \quad y \quad \lambda_2 = \lambda_{max} = \mu_1 \rightarrow \lambda_{max} = \mu_1 (1 - p_2)$$

de modo que se hallará la máxima tasa de entrada por el siguiente algoritmo: así como λ_j y p_j , para j = 2, ..., K son incrementados con λ, la probabilidad p_2 que un paquete sea desechado a la entrada de la estación 2, es tal que la tasa de entrada es $\lambda=\mu_1 (1 - p_2)$, que son obtenidos por un procedimiento iterativo.

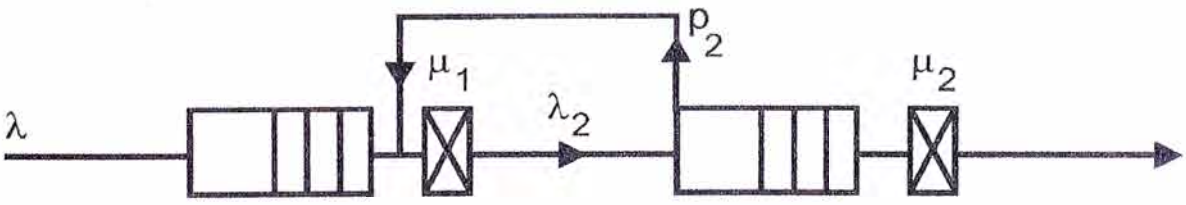


Figura 2.14b. Dos colas en tandem.

Por ejemplo si calculamos la máxima tasa de entrada en el caso particular de dos estaciones. La primera no es forzada en capacidad, la segunda tiene un límite de

capacidad M, donde se representa este modelo en la Figura 2.14b. La máxima tasa de entrada se obtiene para $\lambda_2 = \mu_1$, llegando al siguiente sistema:

De $\lambda_n = \lambda/(1-p_n)$ para n=2 → $\lambda_2 = \lambda_{max} / (1 - p_2)$

como $p_2 = \lim_{\lambda_2 \to \mu_1} \frac{\rho_2(1-\rho_2)}{e^{-\gamma_2(M-1)} - \rho_2^2}$ (*)

con $\rho_n = \lambda_n / \mu_n$ para n=2: $\rho_2 = \lambda_2 / \mu_2$

como $\lambda_2 = \mu_1$ se obtiene: $\rho_2 = \mu_1 / \mu_2$

además: $\gamma_n = 2b_n / \alpha_n$ con $b_n = \lambda_n - \mu_n$ y $\alpha_n = \lambda_n Ka_n + \mu_n Ks_n$

luego: $\gamma_n = 2b_n / \alpha_n = 2(\lambda_n - \mu_n) /(\lambda_n Ka_n + \mu_n Ks_n)$

para n=2: $\gamma_2 = 2b_2 / \alpha_2 = 2(\lambda_2 - \mu_2) /(\lambda_2 Ka_2 + \mu_2 Ks_2)$

y como $\lambda_2 = \mu_1$ se obtiene:

$\gamma_2 = 2(\mu_1 - \mu_2) /(\mu_1 Ka_2 + \mu_2 Ks_2)$ (**)

Utilizando el método de Kobayashi y Reiser que da una formulación sencilla:

$Ka_n = Ks_{n-1}$

para n=2:

$Ka_2 = Ks_1$

Luego (**) se transforma en:

$\gamma_2 = 2(\mu_1 - \mu_2) /(\mu_1 Ks_1 + \mu_2 Ks_2)$

Haciendo los reemplazos necesarios en (*) obtenemos:

$$p_2 = \frac{\frac{\mu_1}{\mu_2}\left(1 - \frac{\mu_1}{\mu_2}\right)}{\left\{\exp\left[-\frac{2(\mu_1 - \mu_2)}{\mu_1 Ks_1 + \mu_2 Ks_2}\right]\right\}^{M-1} - \left(\frac{\mu_1}{\mu_2}\right)^2}$$

y se tiene que $Ks_1 = Ks_2$, y como : $\lambda_{max} = \lambda_2/(1-p_2) = \mu_1/(1-p_2)$,

se obtiene:

$$\lambda_{\max} = \mu_1 \left[1 - \frac{\frac{\mu_1}{\mu_2}\left(1 - \frac{\mu_1}{\mu_2}\right)}{\left\{ \exp\left[-2\frac{\mu_1 - \mu_2}{\mu_1 Ks_1 + \mu_2 Ks_2} \right] \right\}^{M-1} - \left(\frac{\mu_1}{\mu_2}\right)^2} \right]^{-1}$$

la expresión anterior se calcula por técnicas recursivas, cuando la máxima tasa de entrada del primer servidor sigue un tiempo de servicio general. La comparación con su solución habilita la precisión de la fórmula anterior. En la práctica un máximo error de 5% para λ_{max} se ha hallado.

Si ahora se asume que $\mu_1 = \mu_2 = \mu$, se obtiene:

$$\lambda_{max} = \mu \frac{Ks_1 + Ks_2 + 2(M-1)}{2(Ks_1 + Ks_2 + M - 1)} \qquad (3)$$

y $$p_2 = \frac{Ks_1 + Ks_2}{2(Ks_1 + Ks_2 + M - 1)} \qquad (4)$$

Se asume que se tienen tiempos de servicios con distribución e Erlang,:

con tasa $\mu=1$, y $Ks_1 = Ks_2 = 1/e$

De donde se obtiene de (3) y (4):

$$\lambda_{max} = \frac{e(M-1)+1}{e(M-1)+2} \qquad \text{y} \qquad p_2 = \frac{1}{e(M-1)+2}$$

Si e =1 (caso exponencial) , se tiene:

$\lambda_{max} = M/(M+1)$ y $p_2 = 1/(M+1)$

Si e $\rightarrow +\infty$ (caso más desfavorable), se tiene tiempos de servicios constantes, entonces $Ks_1 = Ks_2 = 1/e = 0$ de las ecuaciones (3) y (4)

$$\lambda_{max} = \mu \frac{Ks_1 + Ks_2 + 2(M-1)}{2(Ks_1 + Ks_2 + M - 1)} = \mu \frac{2(M-1)}{2(M-1)} = \mu$$

y $$p_2 = \frac{Ks_1 + Ks_2}{2(Ks_1 + Ks_2 + M - 1)} = 0$$

y como $\mu = 1$, se obtiene: $\lambda_{max} = 1$, y este es un resultado obvio.

Si se asume que el tiempo de servicio es hiperexponencial con $Ks_1 = Ks_2 = e$, entonces de (3) y (4) se obtiene:

$$\lambda_{max} = \frac{e + M - 1}{2e + M - 1} \qquad p_2 = \frac{e}{2e + M - 1}$$

Para e = 1 se determina el caso exponencial,

$$\lambda_{max} = \frac{M}{M + 1} \qquad p_2 = \frac{1}{M + 1}$$

y para $e \to +\infty$, se obtiene el caso más desfavorable :

$\lambda_{max} = 0.5$ y $p_2 = 0.5$.

Si se asume que $Ks_1 = Ks_2 = 1$ (proceso de servicio exponencial) se obtiene:

$$\lambda_{max} = \mu_1 \left[1 - \frac{\frac{\mu_1}{\mu_2}\left(1 - \frac{\mu_1}{\mu_2}\right)}{\left\{ \exp\left[-\frac{\mu_1 - \mu_2}{\mu_1 Ks_1 + \mu_2 Ks_2} \right] \right\}^{2.(M-1)} - \left(\frac{\mu_1}{\mu_2}\right)^2} \right]^{-1}$$

y puesto que : $$\exp\left[-\frac{\mu_1 - \mu_2}{\mu_1 + \mu_2} \right] \cong \frac{\mu_1}{\mu_2}$$ y $Ks_1 = Ks_2 = 1$

se obtiene el máximo tráfico de entrada λ_{max} :

$$\lambda_{max} = \mu_1 \left[1 - \frac{\frac{\mu_1}{\mu_2}\left(1 - \frac{\mu_1}{\mu_2}\right)}{\left\{ \frac{\mu_1}{\mu_2} \right\}^{2.(M-1)} - \left(\frac{\mu_1}{\mu_2}\right)^2} \right]^{-1}$$

2.3.3. Máximo tráfico de entrada cuando las retransmisiones se realizan desde la primera estación:

Si queremos estudiar el máximo tráfico de entrada cuando se tiene un protocolo del tipo 2; esto es, cuando un paquete es rechazado fuera de la red por falta de lugar, y es retransmitido desde la primera estación (Figura 2.15)

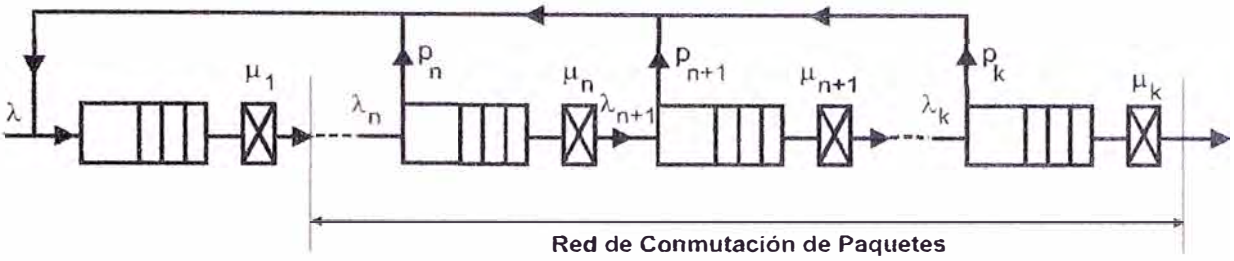


Figura 2.15 Colas en tandem con retransmisión desde la primera estación.

La primera estación representa el servidor con arribos externos, su capacidad se asume infinita, y este guarda una copia de los paquetes hasta que llega una señal ACK positiva. Al inicio del tiempo de salida con la partida del paquete una copia del paquete es generada, luego dicha copia es retransmitida al final del tiempo de salida cuando una señal de ACK positiva no es recibida, se asume un tiempo de salida suficientemente largo para evitar la retransmisión de paquetes inmóviles en la red. Se asume que el CVC del proceso de salida de una cola está dado por la formula de Kobayashi, donde la tasa de entrada a la estación n es ahora:

$$\lambda_n = \lambda_{n-1}(1 - p_{n-1}) \qquad n = 2, \ldots, K$$

$$\lambda_1 = \lambda + \sum_{j=2}^{K} \lambda_j p_j$$

Propio para la conservación del flujo de estado estacionario, la tasa de salida de la última estación es: $\lambda = \lambda_K(1-p_K)$

La probabilidad de rechazo se asume igual a la probabilidad que la cola esté llena.

$p_1 = 0$

$$p_n = \frac{\rho_n(1-\rho_n)}{e^{-\gamma_n(M_n-1)} - \rho_n^{\,2}} \quad \text{n= 2, ... ,K}$$

La máxima tasa de entrada se obtiene para $\lambda_1 = \mu_1$. Las otras colas son siempre estables, propios para pérdidas normales.

2.3.4. Extensión al caso de redes generales:

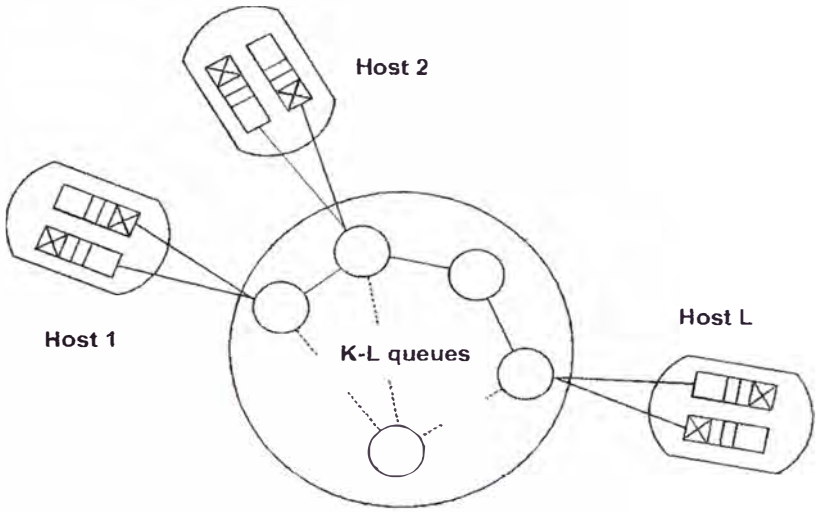


Figura 2.16. Una red General.

El modelo se describe en la Figura 2.16, y está compuesto de L colas de entrada con almacenamiento ilimitado (los servidores) y una red general con capacidad de colas finitas H (estaciones $L+1$, ... , H). La colas representan enlaces de salida y pueden aceptar sólo un número finito de paquetes, entonces cada nodo es descompuesto en enlaces de salida (los tiempos de conmutación se asumen despreciables). Finalmente se asume L colas de salida correspondientes a los L servidores. Se denota: $K = 2L + H$, donde K es el número total de colas en la red.

Se escoge el protocolo 1 de nodo a nodo (cuando las retransmisiones son realizadas desde la estación precedente), más aún, se asume fijo el enrutado y la red libre de bloqueos. El primer problema es definir la máxima tasa de entrada, llamada para definir el vector $\lambda_{max} = (\lambda_1, \ldots, \lambda_L)$. Este vector está en una superficie de orden L y se necesita

otra relación para definir un único vector, por ejemplo, tomamos: $\lambda_{max} = (\lambda_1, \ldots, \lambda_L)$, tal que $\sum_{i=1}^{L} \lambda_i$ es maximizado o tal que la componente λ_j es maximizada para un j.

Dadas dos estaciones ficticias 0 y $K+1$, la primera representa la fuente y la última la partida de un usuario. Dado las probabilidades de ramificación en la red q_{ij} , donde $0 \leq i \leq K$, $1 \leq j \leq K+1$. Estas probabilidades son calculadas desde la tabla de enrutados y las tasas de entradas. Se asume $\lambda_1 = \lambda_2 = \ldots = \lambda_L$, lo cual implica que las probabilidades de ramificación son independientes de los valores de estas tasas de entrada.

Dado el sistema lineal $e = q_o + eQ$, cuya solución es e_i con $i = 1, \ldots, K$

Donde $e = (e_1, \ldots, e_K)$, $q_o = (q_{o1}, \ldots, q_{oK})$ y Q es la matriz de los q_{ij}'s.

Dado $\lambda_i = \lambda e_i$, i = 1, ... , K,

Donde: λ_i representa la tasa de entrada de clientes a la estación *i*, si no hay límite de capacidad. λ_i es la tasa de entrada virtual de la estación *i* (igual a la tasa de entrada real si no hay realimentación propia de la capacidad limitada).

Si estudiamos la estación *i*, la entrada real es:

$$\lambda_i = \sum_{j=0}^{K} q_{ij} \left[\lambda_j / (1 - p_i) \right]$$

y el CVC del proceso de entrada real se asume que es igual a:

$$Ka_i = \frac{1}{\lambda_i} \sum_j \left[(Ks_j - 1) q_{ij} + 1 \right] q_{ij} \lambda_j$$

La tasa de servicio equivalente viene a ser:

$$\mu_i^{-1} = \mu_i^{-1} \sum_{j=1}^{K+1} \frac{q_{ij}}{1 - p_j}$$

y el CVC equivalente del proceso de salida puede calcularse de Ks_i ,

$$Ks_i = -1 + \mu_i^2 \sum_j \frac{q_{ij}\left[1 + p_j + Ks_i(1 - p_j)\right]}{\mu_i^2 (1 - p_j)^2} \quad \text{donde i = 1,..., K}$$

Donde se obtiene un sistema similar a las ecuaciones (1) y (2)

$$\lambda_i = \sum_{j=0}^{K} q_{ij} \frac{\lambda_j}{1 - p_i} \quad , \quad p_i = \frac{\rho_i (1 - \rho_i)}{e^{-\gamma_i (M_i - 1)} - \rho_i^2}$$

donde $\rho_i = \lambda_i / \mu_i$, $\gamma_i = 2b_i / \alpha_i$, y $b_i = \lambda_i - \mu_i$ $\alpha_i = \lambda_i Ka_i + \mu_i Ks_i$,

La solución del sistema no es tan fácil como en el caso anterior, de modo que para un λ dado y por un método iterativo se obtiene la solución para λ_i y p_i , donde $i = 1, ..., K$. Se inicializa con $p_i = 0$, entonces se calcula λ_i del sistema y luego p_i y así sucesivamente. Las condiciones de estabilidad de las L colas de entrada son: $\lambda_i = \mu_i$, para $i = 1, ..., L$. Si todos los λ_i son iguales, se halla una sola solución.

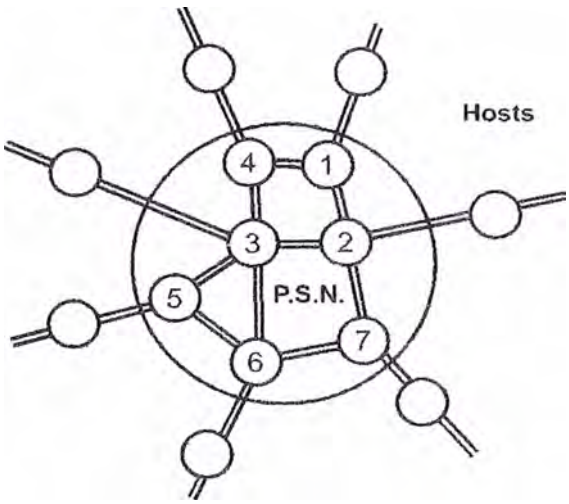


Figura 2.17. Red de Computadoras de 7 nodos.

Por ejemplo, si tomamos la red de computadoras con 7 nodos. Se asume un host (servidor) en cada nodo, y la descomposición en enlaces da 30 colas, donde 16 colas tienen una capacidad finita de almacenamiento, que se asumen idénticas e iguales a M.

- Se asume que todos los enlaces de la red tienen los mismos tiempos de servicio: 1 unidad de tiempo.

- Los servidores se asumen más rápidos: 0.1 unidad de tiempo.
- Las tasas de arribo externo son idénticas a cada servidor.

Luego los resultados para la máxima tasa de entrada para una red de computadoras de 7 nodos como se muestra en la Figura 2.17 es:

$$\lambda = \sum_{i=1}^{7} \lambda_i$$

2.4. Falla de seguridad en protocolos

La confianza de una red de comunicaciones es una propiedad de mayor importancia para su fácil operación. Esta propiedad es completamente dependiente de su habilidad para copar con cambios topológicos, significando que sin rupturas en la red entera o de grandes porciones de ella será activada por tales cambios y que en un tiempo finito luego de su ocurrencia, la red que queda estará habilitada para operar normalmente. Desafortunadamente, la recuperación de la red, luego de cambios topológicos es muy difícil. Los problemas de recuperación son difíciles de resolver, mientras se use un control de rutas centralizado o distribuido. Con ruteo centralizado, se tiene el problema de falla de nodo central más el problema del huevo y la gallina de rutas necesarias para obtener la información requerida de la red para establecer las rutas. El problema entonces es el cálculo asíncrono de información de rutas distribuidas a través de algoritmos los cuales se adapten a cambios arbitrarios en la topología de la red en ausencia del conocimiento global de la topología. Aquí analizaremos un protocolo distribuido que mantiene una ruta desde cualquier fuente a cualquier destino en la red. Desde que no interesa el dato mismo transferido, se usará el término mensaje para significar el mensaje de control especial empleado por el protocolo de ruteo. Se asume que los mensajes

enviados por un nodo a un vecino son procesados por el nodo receptor en el mismo orden en que son enviados.

2.4.1 El protocolo básico:

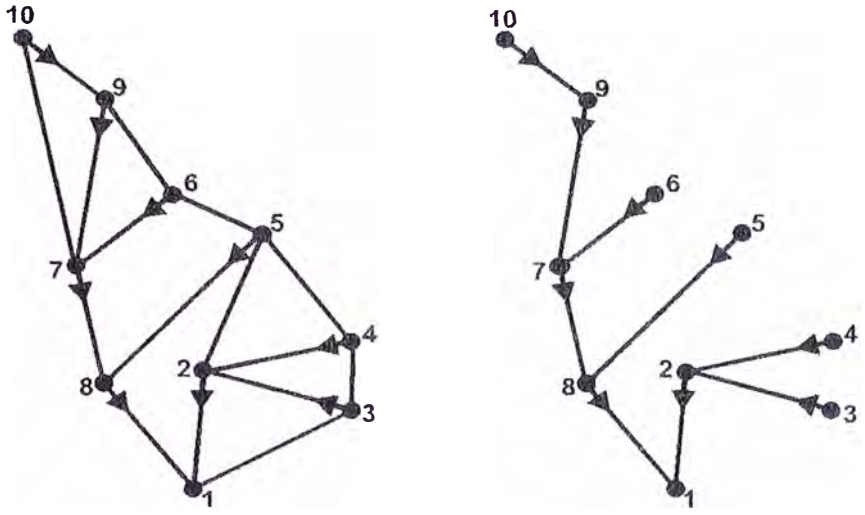


Figura 2.18. ***a) Ejemplo de Red*** ***b) Árbol dirigido correspondiente***

Cada nodo i en la red tiene en algún instante un vecino preferido. Así, se asume que cada nodo tiene una variable p_i la cual apunta a ese vecino. Para el protocolo básico se asume que después de la inicialización, el gráfico dirigido definido por los nodos i y los arcos (i, p_i) forman un árbol dirigido hacia SINK, como se muestra en el la Figura 2.18, donde los arcos dirigidos denotan al vecino preferido $\{p_i\}$. A continuación se describirá el protocolo con cambios topológicos manejables que mostrarán como esta suposición se justifica por el procedimiento de inicialización, donde cada nodo i tiene también una variable positiva d_i mantenida por el protocolo, denotando una distancia estimada desde i hasta SINK (d_{SINK} es por definición igual a cero). Durante un ciclo de actualización, el protocolo re-evalúa las distancias $\{d_i\}$ y los nodos escogidos de vecinos preferidos $\{p_i\}$ de tal manera que el gráfico dirigido dado por los arcos (i, p_i) queda en todo instante un árbol dirigido hacia SINK.

Un peso positivo denotado por d_{il} es asignado a cada enlace (i, l). Se asume que todos los enlaces son full duplex y que permiten a un enlace tener diferentes pesos en cada dirección. El peso d_{il} puede variar en cada instante y se necesita conocerlo (medido o estimado) sólo por el nodo i. El protocolo tiende a minimizar la distancia d_i desde cada nodo i a SINK, donde esta distancia representa un estimado de la suma de los pesos en el camino dirigido desde un nodo a SINK. La señal SINK puede iniciar ciclos de actualización asincrónicamente para cambiar las rutas de acuerdo a las nuevas distancias, y cada ciclo puede ser observado como un procedimiento en dos fases:

(a) Fase 1. Control de mensajes propagado árbol arriba desde SINK a las hojas del árbol actual , mientras se actualiza la distancia estimada { d_i }.
(b) Fase 2. Control de mensajes que proceden árbol abajo a SINK, mientras un nuevo vecino preferido { p_i } está siendo seleccionado.

Un control de mensaje llevando el contenido x será denotado por MSG(x). La Fase 1 de un ciclo de actualización es iniciado por SINK enviando una distancia reportada $MSG(d_{SINK})$ a cada uno de los vecinos (donde $MSG(d_{SINK}) = MSG(0)$ por definición). Un nodo arbitrario, digamos i, participa en la Fase 1 como sigue: Cuando el nodo i recibe un mensaje de su vecino preferido p_i, reevalúa su distancia estimada d_i y transmite $MSG(d_i)$ a cada uno de sus vecinos excepto p_i. Notar que asegurando la estructura del árbol mencionada arriba, se garantiza que luego que SINK ha iniciado el ciclo de actualización, cada uno de los nodos de la red eventualmente ejecutará este paso, más aún, esta es la orden dada por el árbol desde SINK hacia arriba. Mientras que el nodo i recibe un mensaje MSG(d) de su vecino l, estima y almacena su distancia a través de su vecino a SINK. Esta distancia es estimada como $d + d_{il}$, y como se dijo antes, la reevaluación de la distancia estimada d_i, es ejecutada cuando se recibe MSG del vecino preferido p_i. El nodo i calcula entonces la mínima de las distancias estimadas a SINK a

través de todos estos vecinos desde los cuales ha recibido MSG (durante el presente ciclo de actualización), entonces el nodo coloca d_i a su mínimo. Notar que d_i es sólo un estimado de la distancia mínima a SINK debido a que algunas veces el cálculo está basado en una parte de los vecinos de i. Un nodo i ejecuta su parte de Fase 2 de un ciclo de actualización, detectando que mensajes de todos sus vecinos han sido recibidos (durante el presente ciclo), entonces, transmite MSG(d_i) a su vecino preferido actual p_i, el cual provee la distancia estimada mínima desde i a SINK. Esta elección se hace entre todos los vecinos de i, y es tal que puede escoger un vecino diferente de uno de los que provea d_i. Desde que en la Fase 1 cada nodo i eventualmente enviará MSG(d_i) a todos sus vecinos excepto a uno preferido p_i , las hojas del árbol dirigido eventualmente recibirán un mensaje de cada uno de sus vecinos. Entonces la Fase 2 del ciclo de actualización será ejecutada por cada hoja i enviando MSG(d_i) a su vecino preferido p_i. Desde que SINK denota el destino, no tiene vecino preferido, y no se actualiza cuando recibe mensajes de todos sus vecinos. En lugar de esto, este evento sirve para notificar a SINK que el ciclo de actualización ha sido apropiadamente completado, entonces SINK no permite iniciar un nuevo ciclo de actualización hasta que el ciclo previo haya sido propiamente completado. Un nodo i siempre actualiza su vecino preferido p_i a un punto hacia un nodo k que tiene una distancia estimada $d_k < d_i$.

2.4.2. El algoritmo ejecutado por los nodos.

El algoritmo tiene variables p_i , d_i y d_{il} para cada vecino l de i, que almacena respectivamente, al "vecino preferido", "distancia estimada a SINK" y "peso del enlace (i, l)". En suma, también está la variable n_i que almacena el número del ciclo actualmente mantenido por el nodo i. Un control de mensaje enviado por el nodo i tendrá la forma MSG(n_i , d_i) y un control de mensaje recibido por el nodo i desde el nodo l será denotado

por MSG(m, d, l), donde m y d son los valores de n_l y d_l en el instante en que el mensaje fue enviado. La Tabla 1a muestra la lista de todas las variables usadas por el algoritmo en el nodo i y el dominio de sus valores. La Tabla 1b lista los mensajes que el algoritmo del nodo i puede recibir. Fail(l) y Wake(l) son mensajes que dan al algoritmo de i por el protocolo interno, responsabilidad para monitorear los enlaces adyacentes a este nodo, Fail(l) es recibida por i cuando una falla del enlace (i, l) se detecta, y Wake(l) cuando el enlace está operativo. Req(m) es un mensaje especial generado por el nodo i si un cambio topológico fue detectado por j en el instante en el cual $n_j = m$ y además intenta notificar a SINK acerca de este cambio.

Tabla 1a

(a) Variables del Algoritmo de nodo i. (Se asume que la red está compuesta de K nodos)

Nombre de la Variable	**Significado**	**Dominio de valores**
p_i	vecino preferido	nil, 1, 2, 3, ..., K.
d_i	distancia estimada desde SINK	∞, 1, 2, 3, ...
d_{il}	peso estimado del enlace (i, *l*)	1, 2, 3, ...
n_i	número de ciclo actual	0, 1, 2, ...
mx_i	mayor número m recibido por el nodo i	0, 1, 2, ...
CT	bandera de control	0, 1.
$N_i(l)$	ultimo número m recibido desde i después de la última actualización del i ciclo completado	nil, 0, 1, 2, ...
$D_i(l)$	$d + d_{il}$, para el último d recibido desde *l*	∞, 1, 2, ...
$F_i(l)$	estado del enlace (i,*l*)	DOWN, READY, UP
$z_i(l)$	número de sincronización usado por i hasta el enlace (i,*l*)	0, 1, 2, ...

Tabla 1b

(b) Mensajes recibidos por el Algoritmo de nodo i.

Formato del Mensaje	**Significado**	**Dominio de valores**
MSG(m,d,*l*)	actualizando mensaje desde *l*	m = 0, 1, 2, ... d = ∞, 1, 2, ... *l* = 1, 2, ..., K.
FAIL(*l*)	falla detectada en el enlace (i,*l*)	*l* = 1, 2, ..., K.
WAKE(*l*)	enlace (i,*l*) se vuelve operacional	*l* = 1, 2, ..., K.
REQ(*l*)	requerido para nuevo ciclo de actualización con $n_{SINK} > m$	m = 0, 1, 2, ...

Los estados en los cuales el nodo i puede ser S1, S2, S3, S2 (Figura 2.19), S1 es el i-ésimo estado y la transición al estado S2 es ejecutada por i en el instante de ejecutar la

Fase 1 del ciclo, y esperará en este estado hasta que la Fase 2 sea ejecutada, en el instante en que el nodo i retorne a S1 para esperar el siguiente ciclo. S3 es el estado de espera para la recuperación luego que el nodo i es informado que tiene perdida su ruta a SINK, S2 es un estado estancado, propuesto para prevenir que un ciclo de actualización basado en un dato obsoleto se complete y luego sea enterado por el nodo i mientras detecta una falla de un enlace.

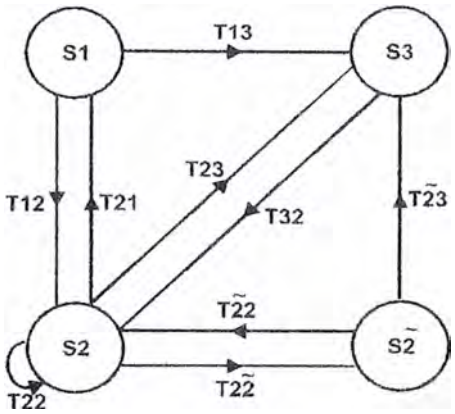


Figura 2.19 Máquina de estados finita para un nodo arbitrario i.

La Tabla 2 describe las condiciones que inducen una transición de estado y las acciones ejecutadas por el nodo i cuando tales condiciones toman lugar. Txy denota una transición del estado Sx al estado Sy, y la transición T12 corresponde a la Fase 1 y es ejecutada cuando recibe un mensaje de control del ciclo numérico más alto del mejor vecino p_i con $d \neq \infty$. Cuando T12 es ejecutado, un nuevo d_i se estima, n_i es actualizado y los nuevos enlaces pueden ser abiertos para mensajes marcados con UP, y MSG(n_i , d_i) y son enviados a todos los vecinos excepto p_i. T21 corresponde a la ejecución de la Fase 2 y es ejecutada cuando el mensaje de control con el número de ciclo más alto ha sido recibido por todos los vecinos después de la Fase 2 del ciclo previo y existe un candidato para el nuevo p_i . Mientras se está ejecutando T21, un MSG(n_i , d_i) es enviado a p_i, y un nuevo p_i se escoge, y los valores de $N_i(k)$ son puestos para nada a fin de distinguir los

mensajes recibidos en el presente ciclo desde los mensajes a ser recibidos durante el siguiente ciclo.

Tabla 2

Algoritmo para un nodo arbitrario manteniendo un mensaje-i.

```
1.1 For REQ(m)
        if p_i ≠ nil, then send REQ(m) to p_i
1.2 For FAIL(l)
        F_i(l) ← DOWN;
        CT ← 0;
        Execute FINITE-STATE-MACHINE;
        if p_i ≠ vacio, then send REQ(n_i) to p_i.
1.3 For MSG(m, d, l)
        if F_i(l) = READY, then F_i(l) ← UP
          (Comment: m > z_i(l));
    N_i(l) ← m;
        D_i(l) ← d + d_il ;
        mx_i ← max (m, mx_i);
        CT ← 0;
        Execute FINITE-STATE-MACHINE.
1.4 For WAKE(l)
            (Comment: Assuming F_i(l) = DOWN)
        if i and l se agregan al enlace abierto (i, l) then:
          z_i(l) ← max (n_i , n_l)
          F_i(l) ← READY;
          N_i(l) ← vacio;
          if p_i ≠ vacio, then send REQ(z_i(l)) to p_i.
```

T13, T23 y T2 3 son transiciones ejecutadas cuando el nodo i está notificado que tiene pérdidas en su ruta hacia SINK, entonces i pone d_i a ∞, actualiza n_i, abre nuevos enlaces, procede a informar a los vecinos que se ha perdido la ruta, y pone p_i con nada. T32 representa la recuperación, por ejemplo, hallando una nueva ruta, en el instante en que un vecino preferido p_i se escoge y la Fase 1 se ejecuta de una manera similar como para T12. T2 2 se ejecuta cuando una falla de un enlace diferente que p_i se detecta, y así T2 2 y T22 son ejecutadas cuando un nuevo ciclo se propaga antes de completar uno previo, y en este caso la Fase 1 para el nuevo ciclo es ejecutado y el ciclo previo es abandonado.

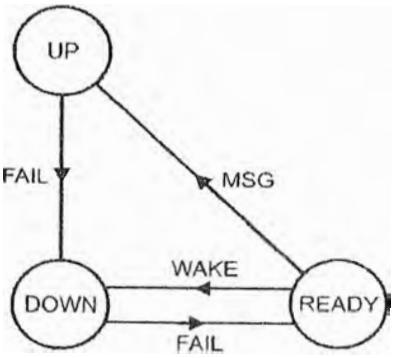


Figura 2.21. Posibles cambios de Fi(l)

La Tabla 2 lista las acciones que toman lugar para cada uno de los mensajes que son recibidos por el nodo i. La Tabla 3 define el protocolo para SINK, el cual es simple debido a que $d_{SINK} = 0$ y SINK no tiene un vecino preferido. El mensaje interno especial START provoca que el algoritmo SINK inicie un nuevo ciclo normal de actualización.

Tabla 3.
El Algoritmo para SINK.

For REQ(m)
    CT ← 0;
    Execute FINITE-STATE-MACHINE
For FAIL(*l*)
    $F_i(l)$ ← DOWN;
    CT ← 0;
    Execute FINITE-STATE-MACHINE.
For MSG(m, d, *l*)
    $N_i(l)$ ← m;
    CT ← 0;
    Execute FINITE-STATE-MACHINE.
For WAKE(*l*)
    (Comment: $F_i(l)$ = DOWN)
    if SINK and *l* agregan al enlace abierto (SINK,*l*), then
    $F_i(l)$ ← READY;
    CT ← 0;
    Execute FINITE-STATE-MACHINE.
For START
    CT ← 0;
    Execute FINITE-STATE-MACHINE.

T12 Condition 12 (CT =0) and (REQ(m = n_{SINK}) or FAIL or WAKE or START).
Action 12 if (REQ or FAIL or WAKE), then n_{SINK} ← n_{SINK} + 1;
∀ k , $F_i(k)$ = READY, then $F_i(k)$ ← vacio; transmit MSG (n_{SINK}, 0)
para todo k $F_i(k)$ = UP; CT ← 1.

T21 Condition 21 ∀ k $F_i(k)$ = UP, then $N_i(k)$ = n_{SINK}; MSG
Action 21 ∀ k $F_i(k)$ = UP, then $N_i(k)$ ← vacio. CT ← 1.

T22 Condition 22 (CT = 0) and (REQ(m = n_{SINK}) or FAIL or WAKE)
Action 22 Same as action 12.

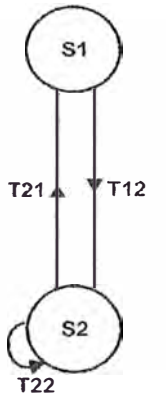


Figura 2.22. Máquina de estados finitos para SINK.

2.4.3 Propiedades del protocolo.

El protocolo de enrutamiento presenta las siguientes propiedades:

A) Propiedad 1: Lazo libre.

B) Propiedad 2: Operación normal.

C) Propiedad 3: Recuperación.

D) Propiedad 4: Rendimiento y eficiencia.

A) Propiedad 1: Lazo libre.

En algún instante, el gráfico dirigido definido por los nodos i y los arcos (i, p_i) es un lazo libre, y de hecho, es un conjunto de árboles dirigidos disjuntos, donde cada árbol es enrutado hacia SINK o al nodo en el estado S3. Para entender porque la propiedad 1 se mantiene, observe que el gráfico dirigido de arcos (i, p_i) cambia sólo si hay una falla de tal arco, y si el nodo i actualizado es un vecino preferido (en la transición T21), o si un nodo sin un vecino preferido escoge a uno nuevo (en T32).

El primer caso claramente no afecta la libertad del lazo, de modo que mientras un nodo i ejecuta la transición T21, todos sus predecesores j en el grafico dirigido deben estar en el estado S1 con $d_j < d_i$. Así, en la transición i no puede cerrarse un lazo. Para el tercer caso, T32 es ejecutado por i, los números de cuentas n_j de todos sus predecesores son $n_j \leq n_i(-)$, mientras que para todos los nodos k que llegan a ser sus sucesores se tiene

$n_k > n_i(-)$, donde $n_i(-)$ es el número de ciclos de i justo antes de la transición. Esto significa nuevamente que ninguno de sus predecesores es un potencial sucesor y así ningún lazo puede estar cerrado.

B) Propiedad 2: Operación normal.

Si un ciclo es iniciado con $n_{SINK} = m$, entonces dentro de un tiempo finito este ciclo será apropiadamente completado o un cambio topológico de orden m ocurre. Hasta completar este ciclo, y hasta que tales cambios ocurran, el conjunto de todos los nodos i potencialmente conectados a SINK quedarán constantes y sus arcos dirigidos (i, p_i) formarán un único árbol dirigido a SINK.

C) Propiedad 3: Recuperación.

Si un cambio topológico de orden m ocurre, un nuevo ciclo de actualización con $n_{SINK} = m + 1$ será o ha sido iniciado, y esta propiedad es garantizada por los mensajes REQ. Bajo el supuesto razonable que la frecuencia promedio de los cambios topológicos no son demasiado altos en comparación con el tiempo de propagación de los ciclos de actualización, las propiedades 2 y 3 garantizan la recuperación. Las propiedades aseguran que los ciclos con números en aumento serán disparados hasta un ciclo apropiadamente completado y todos los cambios topológicos previos serán tomados con cuidado. La siguiente propiedad muestra que cada ciclo mejora las rutas definidas por (i, p_i) y que después de un número limitado de ciclos, converge a la ruta más corta.

D) Propiedad 4: Rendimiento y eficiencia.

Supóngase que los pesos (d_{il}) y la topología de la red quedan fijos por la duración de un ciclo, y si al comienzo del ciclo el árbol de rutas no coincide con SR, entonces al final del ciclo todos los nodos tendrán rutas que no son más grandes que al comienzo del ciclo y hay un conjunto no vacío de nodos para los cuales las rutas son estrictamente cortas.

Hay un número finito L tal que si los pesos (d_{il}) y la topología de la red permanecen fijos por una duración de L ciclos, entonces después de terminar estos ciclos, las rutas provistas por los vecinos preferidos $\{(i, p_i)\}$ coincide con SR. El número L está limitado desde arriba por la mayor distancia de SINK en términos del número de saltos en SR. El protocolo expuesto provee la mejor ruta estimada desde cada nodo al destino y utiliza ciclos de actualización para comprobar periódicamente su ruta bajo cambios topológicos y carga de tráfico. El hecho de que el algoritmo provea una única ruta desde cada nodo a cada destino, no significa que en un instante dado todos los tráficos lleguen a un nodo y destinados a otro nodo serán enviados por medio de esta única ruta. La política correcta es sólo aumentar la fracción de tráfico enrutado por medio del "vecino preferido", mientras se facilita la carga en otras rutas, de esta manera, la ruta mejor estimada llega a ser algunas veces más cargada, mientras que a las otras les llega menos carga, probando así el rendimiento de la red. Una situación natural que puede ser implementada en la práctica, en una red de circuitos conmutados (físico o virtual).

CAPITULO III
ESTUDIOS DE TRAFICO

3.1 Comparación de Capacidades de Tráfico.

La teoría de colas es la teoría de los procesos estocásticos aplicados al estudio de los sistemas de colas. La subred de comunicación de una red local es un ejemplo de sistema de colas. En este caso, un cliente representa un paquete de datos y el servidor es el medio de transmisión que presta el servicio de transmisión de paquetes entre las interfaces de la subred. La llegada de un cliente al sistema equivale a la sumisión de un paquete de datos para ser transmitido y la idea de éste corresponde a la finalización de una transmisión con éxito. El sistema de colas que modela la subred en este ejemplo tiene muchas colas (una para cada interfaz de la red) que están siendo atendidas por sólo un servidor. Sin embargo, otros sistemas de colas pueden tener más de un servidor que atienda a una misma cola. Algunos sistemas de colas están compuestos por un número de subsistemas interligados en red. En tales sistemas los clientes reciben servicio en más de un subsistema (o no) antes de tener atendidos todos sus requisitos de servicio. Normalmente, un cliente inicia su peregrinaje en un nodo dado, espera su turno en la cola y cuando es atendido va hacia otro nodo para obtener otro servicio más, y así sucesivamente hasta tener atendidos todos sus servicios. Estos sistemas son llamados redes de colas. De forma informal, una red de colas es un conjunto de nodos interconectados entre sí. Cada nudo posee uno o más servidores y un lugar para que los clientes formen la(s) cola(s). Existen redes abiertas donde los clientes entran en el sistema y eventualmente salen del sistema; redes cerradas, las cuáles tienen un número fijo de clientes circulando por la red sin salir

y, desde el punto de vista del sistema, es como si no hubiese entrada ni salida de clientes, y redes mixtas, que son abiertas para ciertas clases de clientes y cerradas para otros.

En especial, se ha puesto poca atención a la caracterización del comportamiento de tráfico y colas de sistemas con la configuración a explicarse. La aplicación de la teoría de colas en la metodología para relacionar redes ha recibido alguna atención, y un estudio es para determinar las probabilidades del estado de sistemas y para calcular el rendimiento de sistemas medido desde estas probabilidades. La dimensionalidad requerida del estado de la probabilidad incrementa el estudio rápidamente con el tamaño y complejidad de la red. Una alternativa es para un modelo de sistema que se comporta directamente en términos del valor esperado de parámetros bajo condiciones de equilibrio estadístico. La demanda asignada de acceso múltiple puede ser vista alternativamente como un circuito de conmutación o técnica de conmutación de paquetes, y es particularmente apropiada para el tráfico en canales satelitales. En este estudio, la máxima capacidad de tráfico teórica de sistemas de demanda asignada de acceso múltiple con tipo de colisión de canales requeridos, es determinado como una función de los parámetros del sistema. Esto muestra que algunos sistemas simples de este tipo pueden lograr alta capacidad en situaciones realistas, y en tales esquemas el ancho de banda requerido máximo no depende directamente del número de usuarios del sistema como en la mayoría de sistemas convencionales, los cuales emplean canales requeridos FDM o TDM.

3.1.1 Operación del sistema.

La operación básica del sistema asume que requiere para la comunicación, circuitos que son transmitidos sobre canales compartidos del tipo colisión a un controlador maestro. El controlador puede responder para recibir por medio de un canal de radiodifusión inverso, con alguno de los tres indicadores:

a) El llamado al grupo está ocupado

b) Los circuitos están bloqueados

c) Un circuito específico está asignado.

Los usuarios requieren ejecutar sus solicitudes en tres instantes:

En el primer instante los usuarios requieren ejecutar su solicitud actual y rehusarla un instante después.

En el segundo instante los usuarios repetirán los circuitos requeridos luego de un retardo aleatorio reprogramado.

En el tercer instante los usuarios sintonizarán al circuito asignado en el que la operación deseada tomará lugar.

Al final de estas operaciones el usuario señalará el controlador maestro y el circuito será abandonado. Si ninguna respuesta a la solicitud es recibida por el usuario, dentro de un tiempo especificado después de la señalización por el usuario, se repetirá la solicitud luego de un retardo aleatorio.

Se asume que el sistema tiene c_m canales de mensaje, cada uno de los cuales tiene un ancho de banda de B_m Hz, y c_a canales de colisión, teniendo cada uno un ancho de banda B_a Hz. Por ejemplo, supongamos que un ancho de banda de 250 Khz es dividido usando FDM en tres canales de 80 Khz y uno de 10 Khz , si cada uno de los canales de 80 Khz es entonces dividido por TDM en diez canales de mensajes, y el ancho de banda de 10 Khz es adicionalmente dividido por FDM en dos canales de acceso, entonces $c_a = 2$, $B_a =$ 5 Khz , $c_m = 30$ y $B_m = 8$ Khz .

El sistema puede tener también anchos de banda de fuentes adicionales los mismos aunque necesarias, no han sido consideradas en el presente estudio. Estas pueden incluir:

a) Fuentes de ancho de banda para agregar usuarios en el tiempo de adquisición y sincronización inicial.

b) Un canal de control para envío de información de control y asignación al sistema de usuarios en el modo de transmisión en la banda comercial.

c) Un canal abandonado para cada mensaje en el cual el usuario asignado al correspondiente canal de mensaje señala el circuito abandonado al controlador.

Aparte del ancho de banda necesario para estas funciones el sistema tiene un ancho de banda total *B:* $B = c_a B_a + c_m B_m$

Este es dividido entre los canales de acceso y los canales de mensaje de acuerdo a las fracciones. La fracción del ancho de banda permitido para el acceso de canales v_a definido por: $v_a \equiv c_a B_a / B = c_a B_a / c_a B_a + c_m B_m$ y $1 - v_a = c_m B_m / B = c_m B_m / c_a B_a + c_m B_m$

La tasa total a la cual una nueva solicitud para circuitos ingresa al sistema se denota por λ_T , y el tiempo medio de mantenimiento para un circuito asignado es T_m.

3.1.2 Equilibrio estadístico:

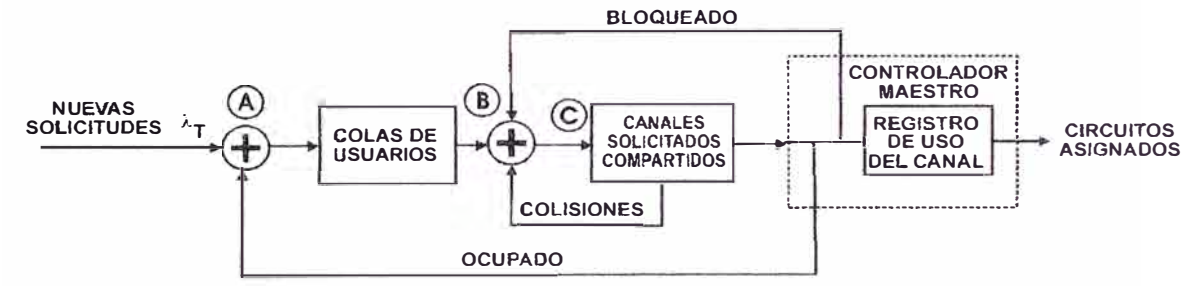


Figura 3.1 Flujo de solicitud para un sistema de demanda asignada por múltiple acceso, usando colisión tipo canales solicitado.

La Figura 3.1 muestra una representación esquemática del flujo de solicitudes a través del sistema. Si λ_T es la tasa a la cual las solicitudes son enviadas a los canales de acceso aleatorio (por ejemplo, punto C en la Figura 3.1). De todos estos:

Una fracción P_c choca y son confusos al controlador maestro.

Una fracción α de estos que el controlador recibe son enviados como caracteres ocupados. Estos reúnen el flujo en A en la Fig 3.1.

Del tráfico que queda, una fracción β recibe caracteres de circuitos bloqueados, y reúne el flujo a B.

Se define como el factor actividad de los canales solicitados compartidos en equilibrio estadístico, al factor $R_a = 1/ (1 - \alpha)(1 - \beta)(1 - P_c)$. Este factor es el número promedio de usos de canales de acceso aleatorio por una nueva solicitud y retransmisiones incorporadas convenientemente para todas las causas (ocupado, bloqueo, colisiones). Obviamente debemos tener $R_a \geq 1$, con la igualdad $R_a=1$ sólo cuando no haya retransmisiones de alguna calidad. En el equilibrio estadístico, la tasa promedio a la cual las nuevas solicitudes ingresan al sistema debe ser igual a la tasa a la cual los circuitos son asignados, con lo que los canales solicitados compartidos deben mantenerse juntos con una tasa: $\Lambda_r = \Lambda_T / (1 - \alpha)(1 - \beta)(1 - P_c) = \Lambda_T R_a$ solicitudes por segundo con una probabilidad de colisión de P_c. Que es la tasa promedio a la cual las solicitudes ingresan a todos los canales de acceso aleatorio.

Por conveniente normalización se denota por τ el tiempo de transmisión requerido por una solicitud cuando se envía en un canal de mensaje. De hecho, la solicitud se envía en un canal de solicitud en el cual el tiempo de transmisión requerido es:

$$\tau_a = \tau (B_m / B_a)$$

Se asume que cualquier usuario escoge un canal de acceso aleatorio, donde este proceso de selección se repite por cada retransmisión. En el promedio por consiguiente, el tráfico solicitado se divide uniformemente entre los canales solicitados c_a y la tasa promedio a la cual las solicitudes ingresan a un único canal de acceso aleatorio es:

$$\Lambda_r / c_a = \Lambda_T R_a / c_a$$

Con los canales solicitados tipo Aloha, cada solicitud tiene una duración τ_a , y la probabilidad de colisión está dada por:

$$P_c = 1 - e^{\left(\frac{-2\Lambda_T R_a \tau_a}{\eta c_a}\right)}$$

en la cual η es 1 o 2 dependiendo sobre si los canales de acceso aleatorio compartido son canalizados o no canalizados respectivamente.

Haciendo manipulaciones algebraicas simples al exponente de la función exponencial, hallamos:

$$\left(\frac{\Lambda_T R_a \tau_a}{\eta c_a}\right) = \left(\frac{\Lambda_T T_m}{c_m}\right)\left(\frac{\tau}{\eta T_m}\right)\left(\frac{c_m B_m}{c_a B_a}\right)(R_a)$$

Se define la ocupancia de un canal de mensaje (erlangs/canal) por:

$\rho_m = (\Lambda_T T_m)/c_m$

y en el segundo paréntesis de la derecha de la ecuación anterior, llamamos $\phi_\tau = \tau/T_m$. El tercer paréntesis es precisamente la razón del ancho de banda del sistema usado por tráfico de mensaje al usado por tráfico solicitado.

Notar que: $v_a \equiv c_a B_a / B$ y $1 - v_a = c_m B_m / B$

luego:

$(1 - v_a) / v_a = c_m B_m / c_a B_a$

y finalmente tomando : $\theta = \phi_\tau (1 - v_a) / \eta v_a$

se obtiene:

$$\left(\frac{\Lambda_T R_a \tau_a}{\eta c_a}\right) = \rho_m \theta R_a$$

por lo tanto la probabilidad de colisión P_c, puede expresarse como:

$$P_c = 1 - e^{\left(\frac{-2\Lambda_T R_a \tau_a}{\eta c_a}\right)} = 1 - e^{-2\rho_m \theta R_a}$$

reordenando se obtiene:

$$e^{-2\rho_m \theta R_a} = 1 - P_c$$

además sabemos que

$$R_a = 1/(1 - \alpha)(1 - \beta)(1 - P_c)$$

En conclusión, se puede obtener:

$$1 - P_c = 1/(1 - \alpha)(1 - \beta)(R_a)$$

remplazando P_c obtenemos:

$$e^{-2\rho_m \theta R_a} = 1/(1-\alpha)(1-\beta)(R_a)$$

multiplicando ambos miembros por un factor se obtiene:

$$2\rho_m \theta R_a e^{-2\rho_m \theta R_a} = 2\rho_m \theta /(1-\alpha)(1-\beta)$$

lo cual debe satisfacerse cuando el sistema está operando en equilibrio estadístico, y el tráfico total de mensajes que está siendo llevado por el sistema es:

$a_T = \Lambda_T T_m = \rho_m c_m$ sabiendo que $\rho_m = (\Lambda_T T_m)/c_m$

Asumiendo que este tráfico es dividido igualmente entre todos los receptores, la fracción de respuesta del controlador las cuales son señales de ocupado es:

$$\alpha = (\rho_m / K)$$

en la cual K es el número de receptores en el sistema por canal de mensaje.

3.1.3 Protocolo de solicitud borrada:

Siguiendo el protocolo básico de demanda asignada de múltiple acceso, se asume que el controlador no mantiene una cola de solicitudes, las cuales están aguardando circuitos. Más bien, cualquier solicitud que está bloqueada u ocupada es borrada del controlador inmediatamente después de enviar la respuesta apropiada. Esto minimiza los requerimientos de almacenamiento en el controlador, así como el usuario controla el hardware, pero a expensas de tráfico adicional en los canales solicitados los cuales llevarán repeticiones de las solicitudes bloqueadas u ocupadas. Bajo la suposición de

arribos de Poisson, la probabilidad de bloqueo está dada por la fórmula de pérdidas en Erlang: $\beta = B(c_m, a_o)$ donde:

$$B(s, a) = \frac{(a^s / s!)}{\sum_{k=0}^{s} (a^k / k!)} \qquad s = 1, 2, \ldots$$

El tráfico a_o es el que choca en el servidor del sistema de c_m (canales de mensaje) después que parte del tráfico que recibe una señal de ocupado a sido desviado. Desde que en el equilibrio estadístico, los canales de mensaje c_m están llevando un tráfico de:

$a_T = \rho_m c_m$ erlangs, y por lo tanto a_o y a_T están relacionados por la ecuación:

$$a_T = a_o (1 - B(c_m, a_o))$$

donde a_T y c_m son datos conocidos y esta ecuación puede resolverse para a_o, usando un esquema de iteración. El sentido completo de las soluciones requieren que $a_T < c_m$ de otro modo a_o no será limitado o no tendrá soluciones reales. Se denota la solución por $a_o = g(c_m, a_T)$, entonces sustituyendo por α y β se tiene:

$$2\rho_m \theta R_a e^{-2\rho_m \theta R_a} = 2\rho_m \theta / (1-\alpha)(1-\beta)$$

$$2\rho_m \theta R_a e^{-2\rho_m \theta R_a} = 2\rho_m \theta / ((1-(\rho_m / K))(1-B(c_m, g)))$$

siendo esta la relación entre los parámetros requeridos por equilibrio estadístico. Por conveniencia se define:

$$\gamma = 2\rho_m \theta / ((1-(\rho_m / K))(1-B(c_m, g)))$$

Asignando: $x = 2\rho_m \theta R_a$

obtenemos: $x e^{-x} = \gamma$ y desde que el lado izquierdo de la ecuación anterior alcanza el máximo de e^{-1} cuando $x = 1$, y es claro que el equilibrio estadístico requiere $\gamma \leq 1/e$. Donde la igualdad corresponde a la situación en la cual los canales compartidos requeridos, operan a su capacidad; con $\gamma < 1/e$, las soluciones con $0 \leq x < 1$ representan

la solución estable del sistema para equilibrio estadístico. Con γ dado, la solución para $xe^{-x} = \gamma$ para $0 \leq x < 1$ puede obtenerse usando un esquema iterativo. Denotamos la relación inversa por $x = h(\gamma)$, entonces la probabilidad de colisión P_c se determina por $P_c = 1 - e^{(-h(\gamma))}$.

El factor de actividad de los canales de acceso aleatorio es entonces determinado usando la definición de x, dando como resultado: $x = h(\gamma) = 2 \rho_m \theta R_a$ en consecuencia:

$$R_a = h(\gamma) / 2\rho_m \theta$$

se notará que γ está completamente especificada por cuatro parámetros c_m, ρ_m, θ y K, y que P_c y R_a están determinados por estos cuatro parámetros, y son suficientes para especificar el punto de equilibrio operativo del sistema para el protocolo de solicitud borrada.

3.1.4 Utilización de recursos y capacidad del sistema:

Con los requerimientos de equilibrio estadístico establecido, la habilidad del sistema para llevar tráfico de mensajes puede ser examinada. Supóngase que hay suficiente demanda de tráfico de modo que los parámetros del sistema limitan la cantidad de tráfico que puede mantenerse en equilibrio estadístico, y es de interés determinar cual es este límite y como efectivamente el sistema usa el ancho de banda permitido. La cantidad de utilización de este ancho de banda asume un sistema con c_m canales de mensaje, cada uno con ancho de banda B_m y suficiente demanda de usuarios para utilizar lo máximo de estos canales como sea posible. Desde que un erlang de tráfico de mensajes, es perfectamente planificada, ocupará B_m Hz, y una medida de utilización del ancho de banda es:

S = (*erlangs* de portadoras de tráfico de mensajes)B_m / B (ancho de banda total)

Si la fracción del ancho de banda total del sistema ubicado en los canales solicitados compartidos fueron cero, el ancho de banda total del sistema será lo menor posible, de

modo no se conseguirá ninguna solicitud, y el numerador será cero. Incrementando el ancho de banda de los canales solicitados compartidos, se incrementan el numerador y denominador de la ecuación anterior El tráfico no puede exceder c_m erlangs y así el denominador de la ecuación anterior eventualmente pasará al numerador y provocará la utilización a cero. Entre estos extremos, se debe estar en una ubicación óptima y la utilización para esta ubicación se define como su capacidad C_T , del sistema de demanda asignada de múltiple acceso.

Específicamente $C_T = \max_{v_a} S$

Renombrando que la portadora de tráfico de mensaje es $a_T = \rho_m c_m$ y que $(1 - v_a) = c_m B_m / B$, se puede escribir simplemente que la utilización del ancho de banda es:

$$S = \rho_m (1 - v_a)$$

La capacidad de tráfico del sistema de demanda asignada por múltiple acceso es entonces: $C_T = \max_{v_a} \rho_m (1 - v_a)$

Esto es importante, de hecho, para reconocer que en el resultado general anterior, la cantidad ρ_m depende de v_a por medio de las ecuaciones requeridas para equilibrio estadístico. Se notará que generalmente no hay ventajas para operar a la capacidad del sistema de demanda asignada por múltiple acceso cuando factores tales como la estabilidad marginal y características de retardo son consideradas. No obstante C_T representa un límite superior en la utilización del ancho de banda y así sirve como un marcador para comparación de sistemas.

3.1.5 Capacidad para el protocolo de solicitud borrada:

En el protocolo de solicitud borrada, nuevas solicitudes así como solicitudes bloqueadas, ocupadas y colisionadas aparecen en los canales solicitados compartidos.

Como el tráfico de mensaje se aproxima a c_m la probabilidad de bloqueo se aproxima a la unidad y R_a tiende al infinito, de modo que cuando existe suficiente demanda para canales de mensaje, la capacidad de tráfico del sistema estará limitada por los canales de acceso aleatorio, y desde que este límite es alcanzado con $\gamma = 1/e$, se determina que con suficiente tráfico de mensajes:

$$2\,\theta e = \rho_m^{-1}\,(1 - (\rho_m / K))(1 - B(c_m, g)) \equiv F(\rho_m)$$

eliminando θ y llamando $K_1 \equiv (2\,e\,\phi_\tau) / \eta$ se obtiene:

$$v_a = K_1 / (K_1 + F(\rho_m))$$

lo cual combinado con la ecuación de S da:

$$S = \rho_m F(\rho_m) / (K_1 + F(\rho_m))$$

Las ecuaciones anteriores definen a S como una función implícita de v_a por medio de su dependencia con ρ_m. Se puede mostrar que la capacidad C_T puede hallarse por un simple esquema numérico el cual maximiza la ecuación anterior con respecto a ρ_m, y su optimización es usada para hallar el correspondiente v_a óptimo.

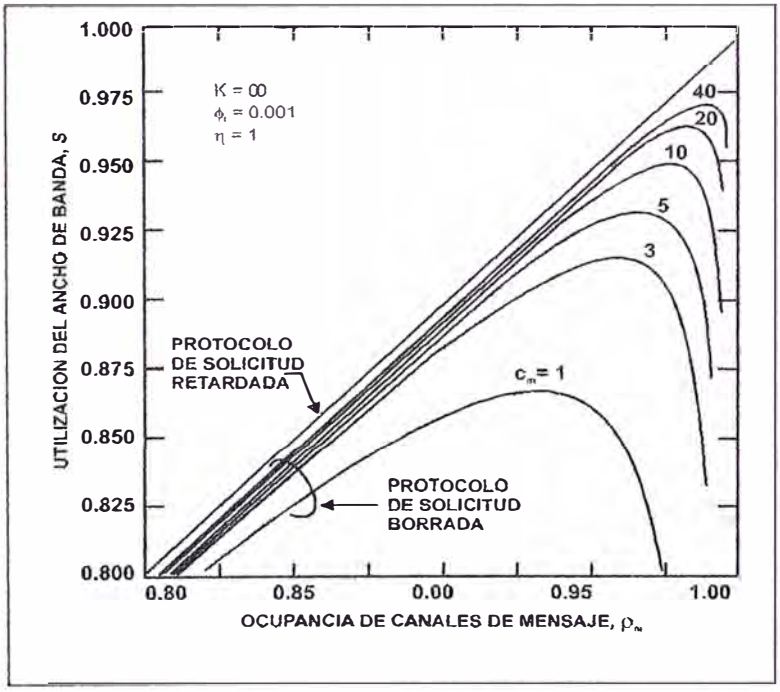


Figura 3.2 Utilización del ancho de banda para protocolos de demanda asignada por múltiple acceso, con colisión tipo canales solicitado.

Una gráfica de la utilización del ancho de banda S como una función de la ocupancia de canal ρ_m para varios c_m se muestra en la Figura 3.2 para canales solicitados sin dividir con $\phi_\tau = 0.001$ y $K = \infty$. Curvas completamente familiares se obtienen para otra elección de parámetros, donde para valores pequeños de ρ_m las curvas de cada una de las figuras son muy próximas, pero que difieren para valores grandes de ocupancia de canal, y la mayor utilidad accesible se obtiene cuando el número de canales de mensajes se incrementa.

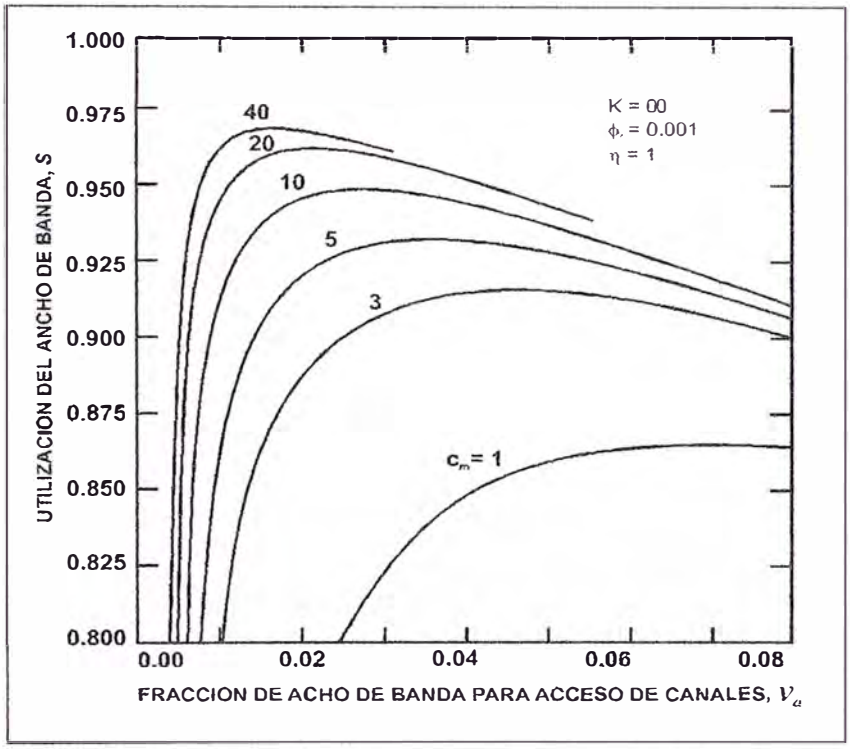


Figura 3.3 Utilización del ancho de banda versus fracción del ancho de banda permitido para el acceso de canales.

La Figura 3.3 muestra la utilización del ancho de banda graficada directamente como una función de la fracción del ancho de banda permitido para el acceso de canales v_a donde se observa que la capacidad próxima a la operación de un sistema es mucho más sensible para anchos de banda insuficientes permitidos para los accesos de canales que para anchos de banda excesivos.

Este efecto llega a ser más pronunciado conforme c_m se incrementa, y físicamente esto es debido a que a la izquierda de v_a óptimo, una disminución en el ancho de banda del canal de acceso incrementa ligeramente los canales de mensaje por una cantidad mucho mayor que la reducción del ancho de banda. Conforma K disminuye, el efecto de respuesta de ocupado viene a ser más significativa, y la presencia de repeticiones para respuesta de ocupado se reflejan en un mayor ancho de banda requerido para los canales solicitados y menor capacidad. Curvas similares a las figuras 3.2, 3.3, y 3.4 pueden obtenerse y se halla que para $K \geq 20$ las diferencias son despreciables.

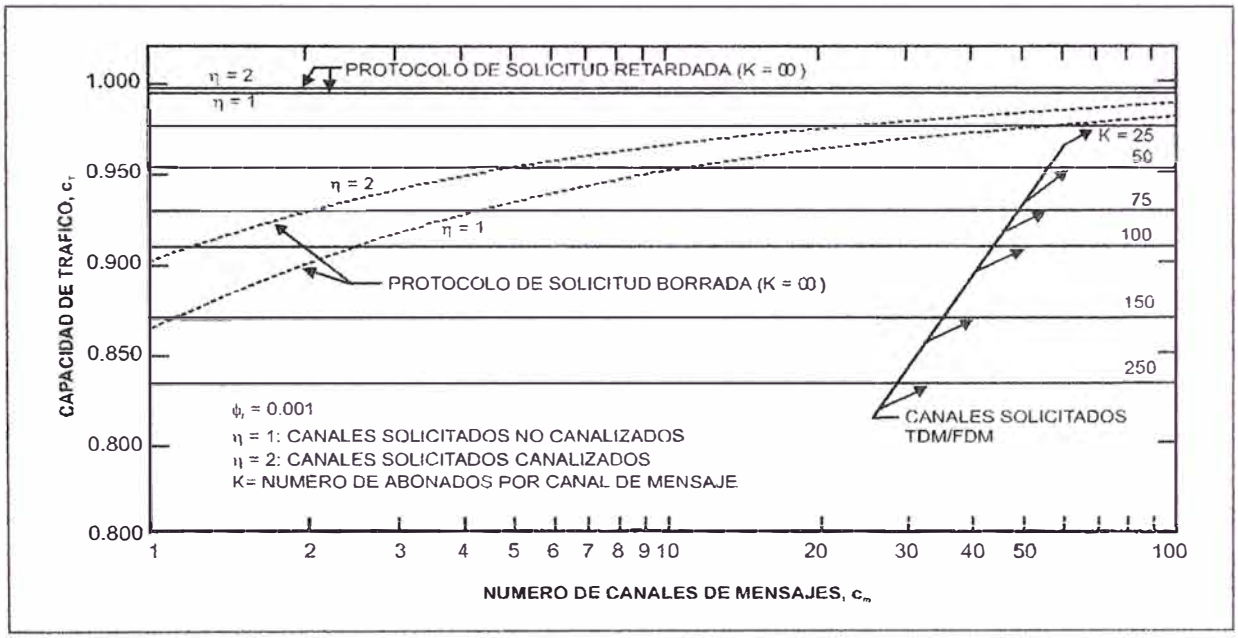


Figura 3.4 Comparación de capacidad de tráfico de demanda asignada por múltiple acceso versus el número de canales de mensaje.

Un gráfico de la capacidad de tráfico obtenible con el protocolo de solicitud borrada se muestra en la Figura 3.4 como una función del número de canales de mensaje, y para canales solicitados divididos y no divididos. Incrementando el número de canales de mensaje, se incrementa la utilización y capacidad que puede ser alcanzada. Con $c_m > 10$ capacidades mayores que alrededor de 0.95 se obtienen con canales divididos o no, y el uso de canales solicitados divididos puede mejorar la capacidad por un pequeño

porcentaje para c_m pequeños, y esta mejora disminuye cuando c_m se incrementa, , pero es sin embargo mayor que 1.0% para $c_m = 30$.

Capacidad aproximada. Una simple expresión puede obtenerse para $c_m = 1$ como sigue, cuando $c_m = 1$, la probabilidad de bloqueo β es precisamente ρ_m , donde tomando $K >> 1$, implica un mayor número de abonados del sistema de modo que la probabilidad de ocupado es esencialmente cero. Una interpretación alternativa es que todos los usuarios sean usuarios de la red, tal que no sean usadas señales de ocupado.

Por otro lado se puede hallar que:

$$F(\rho_m) = (1 - \rho_m) / \rho_m$$

y realizando cálculos simples se puede mostrar que el valor máximo de ρ_m es:

$$\rho_o = 1/(1+\sqrt{K_1}), \qquad \text{y} \qquad C_T = 1/(1+\sqrt{K_1})^2 \equiv \rho_o^2$$

Más aún, para este ρ_o , se halla que $v_{ao} = 1 - \rho_o$

Se sabe que la probabilidad de bloqueo para un grupo de servidores múltiple es menor que para un único servidor cuando ambos sistemas están llevando la misma carga por servidor, de esto se induce que:

a) para la ubicación sobre un grupo de servidores múltiple se tendrá mayor utilización del ancho de banda, y

b) desde que la utilización no es óptima para el sistema de servidores múltiples, mayores capacidades serán alcanzadas. Estos hechos se ilustran en las figuras 3.2 a 3.4, más aún estos cálculos simples pueden ser usados para estimados conservadores de ancho de banda y capacidad de canales solicitados.

La diferencia entre la fracción de recursos permitidos para tráfico de mensajes y la capacidad representa tiempo inútil en los canales de mensaje propios para el arribo de solicitudes aleatorias. Este tiempo inútil puede ser reducido a una cola de solicitudes para

los cuales ningún circuito está inmediatamente disponible Si el controlador maestro se usa, esto requiere programación adicional y/o capacidad de almacenamiento, y los requerimientos de señalización para este esquema también diferirán y se requerirá alguna complejidad mayor para los usuarios del sistema.

Este esquema es similar en concepto para ciertos esquemas de reservación de paquetes, y las diferencias son que:

a) Manteniendo tiempos necesarios, no se fijan o conocen tiempos de solicitud.

b) Múltiples canales de mensaje están disponibles para usos simultáneos.

c) Las señales de ocupado pueden ser usadas y las correspondientes solicitudes reaparecen.

d) Si hay almacenamiento finito, las señales de los circuitos bloqueados pueden ser usadas y las correspondientes solicitudes repetidas como en el protocolo de solicitud borrada. Aún con almacenamiento ilimitado todos los tiempos inútiles no podrán ser eliminados a menos que el tiempo de mantenimiento para cada solicitud sea conocida de antemano, de modo se que se alcance una planificación perfecta. Si se asume un tiempo de mantenimiento promedio que sea mayor comparado con la señalización requerida y un almacenamiento ilimitado de solicitudes, un simple estimado de capacidad de tráfico para este protocolo puede obtenerse aún así.

En este protocolo, la ecuación:

$$2\rho_m\theta\, R_a e^{-2\rho_m\theta R_a} = 2\rho_m\theta/(1-\alpha)(1-\beta)$$

que describe los requerimientos de equilibrio estadístico es válido, excepto que con almacenamiento ilimitado no hay tráfico con circuitos bloqueados en el sistema (por ejemplo $\beta = 0$). Entonces procediendo como antes, se llega a: $2\,\rho_m\,\theta e = (1 - (\rho_m / K))$

usando esta ecuación se puede resolver para v_a, y entonces se obtiene una expresión simple para la utilización del ancho de banda S. Con un gran número de usuarios del sistema (por ejemplo. $K >> 1$) el resultado es: $S \approx \rho_m (1 + \rho_m K_1)$

Esta ecuación es graficada en la Figura 3.2 para compararla con el protocolo de solicitud borrada, donde la capacidad para este protocolo es claramente: $C_T \approx 1 / (1+K_1)$

Los resultados numéricos (para $\phi_\tau = 0.001$) se muestran en la Figura 3.4 para canales solicitados divididos y no divididos, donde la comparación con el protocolo de solicitud borrada para $c_m = 1$ muestra que almacenar y/o programar pueden mejorar la capacidad del sistema por 14.7% con canales solicitados sin división, y por 10.4% con canales solicitados divididos. En cualquier situación la capacidad resultante excede 0.99, mientras que la diferencia de capacidad entre los protocolos de solicitud borrada y solicitud retardada puede ser completamente significativa para pequeños valores de c_m y puede verse que con canales de mensaje múltiple la capacidad del protocolo de solicitud borrada se incrementa sustancialmente, y el rango de posibles mejoras es algunas veces más pequeño y la diferencia en utilización entre los dos protocolos es considerablemente menos dramática para $c_m >> 1$.

CONCLUSIONES Y RECOMENDACIONES

1. Un protocolo es un conjunto de reglas y normas, que regulan y aseguran que el intercambio de información entre los elementos de un Sistema Informático sea eficaz y fiable. El número de protocolos posibles para el intercambio de información entre computadoras y dispositivos de una red es enorme. Los sistemas distribuidos modernos pueden requerir protocolos extremadamente complejos, y dichos protocolos pueden ser tan sutiles como un método sistemático sea necesario para garantizar que esta definición sea completa y no ambigua, y que el propósito del protocolo sea correctamente realizado.

2. Las técnicas de especificación basadas en modelos formales son usados para definir protocolos, y las técnicas de validación son usadas para asegurar su correcta y apropiada operación. Las diferentes clases de protocolos requieren diferentes técnicas de especificación y validación, y no hay un método único que pueda ser convenientemente usado para modelar y validar todos los protocolos.

3. En el modelo usado para especificar el protocolo, la aplicabilidad teórica o práctica de una técnica de validación depende de las características del protocolo (de parte y de topología) y las propiedades del protocolo. La aplicabilidad práctica de un modelo de especificación de protocolos o de una técnica de validación es dificil para formalizar. La aplicabilidad práctica implica aplicabilidad teórica, pero también brevedad, facilidad de entender, conveniencia para usar, etc., las cuales son dificiles de cuantificar y algunas veces dependen de experiencia personal o prueba, y fuentes

disponibles. El incremento en la complejidad de la característica de topología de un protocolo puede también impedir la aplicabilidad de una técnica; por ejemplo, un protocolo con partes simples puede no estar validado por generación de estados globales exhaustivos si incluyen demasiadas partes. La complejidad del comportamiento del enlace de la característica de topología del protocolo puede también impedir la aplicabilidad de una técnica de especificación o una técnica de validación de protocolos; por ejemplo, los enlaces pueden perder mensajes, de modo que un enlace complejo puede ser representado como una parte que modela su comportamiento, y así los problemas causados por la complejidad de enlaces no requieren tratamiento especial.

4. En la práctica los lenguajes de programación estándares de alto nivel son convenientes para representar números, datos, variables, contadores, etc., pero no estructuras de control complejas. Así, los lenguajes de programación son usados principalmente para representar los aspectos de transferencia de datos de protocolos, mientras que los modelos gráficos (máquinas de estado y redes de Petri) son principalmente usadas para representar los aspectos de control, tales como sincronización, inicialización, etc., para los cual ellos son más convenientes.

5. Mientras que la generación de estados globales es más conveniente en la prueba del control de propiedades (tal como ciertos eventos que podrán o no ocurrir), la prueba de la afirmación es principalmente usada en pruebas de transferencia de propiedades de datos, particularmente en protocolos que involucran partes con espacios de estado grandes o infinitos. Las dos técnicas pueden ser combinadas en suma para capitalizar las ventajas de cada uno.

6. Para el protocolo de solicitud borrada: Haciendo un análisis de la utilización del ancho de banda respecto a la ocupancia de canales de mensaje se obtiene que para valores pequeños de ocupancia de canales de mensaje las curvas de la utilización del ancho de banda son muy próximas, pero para valores grandes de ocupancia de canal difieren, y la mayor utilidad accesible se obtiene cuando el número de canales de mensaje se incrementa. Del análisis de la utilización del ancho de banda respecto a la fracción del ancho de banda para acceso de canales, se obtiene que la capacidad próxima a la operación de un sistema es mucho más sensible para anchos de banda insuficientes permitidos para los accesos de canales que para anchos de banda excesivos. Este efecto llega a ser más pronunciado conforme el número de canales de mensaje se incrementa, y fisicamente esto es debido a que a la izquierda de la fracción del ancho de banda para acceso de canales optimo una disminución en el ancho de banda del canal de acceso incrementa ligeramente los canales de mensaje por una cantidad mucho mayor que la reducción del ancho de banda. Por otro lado del análisis de la capacidad de tráfico respecto del número de canales de mensaje, se obtiene que incrementando el número de canales de mensaje, se incrementa la utilización y capacidad que puede ser alcanzada. Con un número de canales de mensaje mayores a 10, capacidad de tráfico de alrededor de 0.95 se obtienen con canales divididos o no, y el uso de canales solicitados divididos puede mejorar respecto de la capacidad de tráfico de los canales solicitados no divididos, por un pequeño porcentaje para un número pequeño de mensajes, y esta mejora disminuye cuando el número de canales de mensaje se incrementa, pero es sin embargo casi imperceptible cuando el número de canales de mensaje es mayor de 30.

7. Este trabajo de investigación se ha desarrollado en toda la extensión posible; sin embargo, no se ha logrado abarcar lo suficiente, porque tanto los protocolos como el tráfico en redes de computadoras son temas muy amplios, por lo que las futuras investigaciones respecto a estos temas deben presentar una mejor especificación del tema, que toda investigación científica requiere; es decir, se debe enfocar mejor el problema de investigación, y mejorar el diseño de la investigación.

BIBLIOGRAFÍA

1. Donald Gross, Carl M. Harris, ***Fundamentals of Queueing Theory***, John Wiley & Sons, Inc., USA 1974.
2. William Feller, ***Introducción a la Teoría de Probabilidades y sus aplicaciones***, Limusa, México, 1975.
3. Barry L. Nelson, ***Stochastic Modeling: Analysis and Simulation***, McGraw Hill, Inc., USA, 1995.
4. Don T. Phillips, A. Ravindran, James Solberg, ***Operations Research: principles and practice***, John Wiley & sons, Inc., USA, 1976.
5. Ravindra Ahuja, T. Magnanti, J. Orlin, ***Network, Flows: Theory, Algorithms and Applications***, Prentice-Hall, Inc., 1993
6. M. Bazaraa, J. Jarvis y H. Sherali, ***Linear Programming and Network***, Wiley, 1990.
7. James R. Evans y Edward Minieka, ***Optimization Algorithms for Networks and Graphs***, Marcel Dekker,Inc., 1992.
8. T. C. Hu, ***Combinatorial, Algorithms***, Addison-Wesley, Reading, Massachussets, 1982.
9. A. Kaufmann, ***Métodos y Modelos de la Investigación de Operaciones***, Compañía Editorial Continental S.A., México, 1972.
10. R. Thierauf y R. Grosse, ***Toma de decisiones por medio de la Investigación de Operaciones***, Limusa. México, 1981.

11. IEEE, ***IEEE 83***, Proceedings of the IEEE, diciembre, USA, 1983.

12. Tanenbaum A. S., ***Computer Networks***, Prentice Hall Inc, USA, 1981.

13. H. Zimmerman, ***OSI The Reference Model***, IEEE Transactions on Communications, USA, 1980.

14. P. E. Green, ***Computer Network architectures and protocols***, Plenum Press, USA, 1983.

15. C. A. Sunshine, ***Survey of protocol definition and verification techniques, Computer Networks***, USA, 1978.

16. A. Alabau y J. Riera, ***Teleinformática y Redes de Computadoras***, Boixareu Editores, Brasil, 1983.

17. C. A. Sunshine, ***Formal methods for communications protocol specifications and verification***, The Rand Corp., USA, 1979.

18. Thomas W. Madron, ***Redes de Area Local***, Megabyte, Grupo Noriega Editores, México, 1993.

19. W. Stallings, ***Comunicaciones y Redes de Computadoras***, Prentice Hall, España, 1997.

20. W. Stallings, ***A tutorial on the IEEE 802 Local Network Strandard***, Computer Science Press Inc., USA, 1986.

21. Anon, IEEE ***Committee Addresses LANs***, Computer Design, USA, 1985.