XX (147074,1)

CINVESTAV IPN
USB INFORMACION Y DOCUMENTACION
SERVICIO DOCUMENTAL

Centro de Investigación y de Estudios Avanzados
del I.P.N.
Unidad Guadalajara

Gestión y Control de Mecanismos de Calidad de Servicio en Redes IP

Tesis presentada por:
Edson Gallo Contreras

para obtener el grado de:
Maestro en Ciencias

en la especialidad:
Ingeniería Eléctrica

CINVESTAV
IPN
ADQUISICION
DE LIBROS

Director de Tesis:
Dr. Mario Angel Siller González Pico

Guadalajara, Jalisco, Mayo de 2008.

Gestión y Control de Mecanismos de Calidad de Servicio en Redes IP

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Edson Gallo Contreras
Ingeniero en Computación
Universidad de Guadalajara 2001-2004

Becario de CONACYT, expediente no. 199534

Director de Tesis:
Dr. Mario Angel Siller González Pico

CINVESTAV del IPN Unidad Guadalajara, Mayo de 2008.

CLASIF.: TK165.G8 .G35 2008
ADQUIS.: BC - 510
FECHA: 11-XI-2008
PROCED.: Don. - 2008
$

U. 144199 - 1001

Resumen

En esta tesis se presenta un marco de trabajo para la gestión y control de mecanismos y protocolos de Calidad de Servicio mediante agentes, utilizando un enfoque racional, para la mejora de la Calidad de Experiencia del usuario en la transmisión de servicios de multimedia en redes IP Dicho marco tiene su origen en el marco de trabajo *QoE Framework*. En la presente propuesta se incrementa la capacidad de gestión de mecanismos de Calidad de Servicio (*Quality of Service QoS*) con los mecanismos de Servicios Integrados y MPLS. Se presenta una extensión de la ontología QoE, que es la ontología utilizada por los agentes. Esta extensión se hizo partiendo de la idea de que las ontologías de *QoS Management* se requieren para métricas de QoS, unidades de medición, unidades monetarias, propiedades medibles y métodos de medición. Además, deben de permitir decidir que mecanismo QoS satisface de mejor manera las necesidades del usuario, así como el monitoreo y detección de violaciones en los SLAs y llevar a cabo adaptación de QoS. También se propone un algoritmo para la toma de decisiones de los mecanismos QoS y su configuración, en función de que se provea una mejor Calidad de Experiencia (*Quality of Experience – QoE*) percibida por el usuario, y optimizando la asignación de recursos de la red durante una sesión multimedia. Este algoritmo se basa en la Teoría de la Decisión, la cual comprende la Teoría de la Utilidad y la Teoría de la Probabilidad. Se muestra el diagrama de decisión con el que se hace el cálculo de la Máxima Utilidad Esperada, utilizando así un enfoque racional. Además se elaboro un modelado mediante redes de Petri del proceso de gestión y control de QoS para proporcionar una representación formal de este proceso y mostrar algunas de sus propiedades. Finalmente se muestran los resultados de los experimentos conducidos en el laboratorio para la evaluación subjetiva de la QoE experimentada por el usuario bajo este marco de trabajo. Para esto se utilizaron métodos de evaluación estándar como SSCQE. Los resultados muestran un desempeño favorable en términos de la red y percepción del usuario final.

Abstract

This thesis presents an agent based framework for QoS Management. It uses a rational decision approach to improve the end user Quality of Experience for multimedia transmission over IP networks. This framework is based on a previous *QoE Framework*. This framework is extended to support additional QoS protocols such as Integrated Services and MPLS. The QoE ontology is also extended based on the idea that ontologies in QoS Management are required support QoS metrics, measurement units, currency units, measured properties and measurement methods. Besides, a QoS ontology should allow to decide which QoS mechanism is better to fit the user requirements, perform QoS monitoring and adaptation, and detect SLA violations.

An algorithm for the decision taken of the QoS mechanism and its configuration is proposed. This algorithm provides a better QoE perceived by the end user. It optimizes the allocation of network resources during a multimedia transmission. The algorithm is based on the Decision Theory which encompasses Utility Theory and Probability Theory. The decision network related with this algorithm is shown too. This decision network is used to calculate the Maximum Expected Utility, which gives the rational approach. A Petri net model for the QoS Management process is depicted in order to give a formal representation of this process and to prove some of their properties. Finally the experimental results are shown. A subjective evaluation of the perceived QoE under the framework is analyzed. Standard evaluation methods such as SSCQE are employed. The results show a favorable performance of the network and the end user perception.

Agradecimientos

A Dios, por darme la luz que me ha guiado a través de toda mi vida.

A mis padres, por darme la libertad de elegir mi camino y el soporte que me ha llevado por él.

A mi novia, por su comprensión y la motivación que significa para mí.

A mi asesor, por sus enseñanzas y el aprendizaje que hemos compartido.

A mis sinodales y profesores, de los cuales también aprendí y por su buena convivencia.

A mis compañeros, por su apoyo y amistad.

A Cinvestav, por ofrecerme la oportunidad de una maestría.

A CONACYT, por su beca.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Debido al gran crecimiento de transmisiones multimedia a través de la red han surgido varios mecanismos para garantizar la Calidad de Servicio, incluso existen combinaciones entre ellos que forman diferentes arquitecturas para dar solución al problema, pero, ¿cuándo utilizar cada uno de ellos?

Debe existir una entidad que se encargue de gestionar y controlar estos mecanismos de Calidad de Servicio para obtener un máximo desempeño de la red y ofrecer al usuario la mejor calidad posible en las transmisiones multimedia.

De las transmisiones más comunes que hoy en día se pueden encontrar en las redes de conmutación de paquetes son las teleconferencias, tráfico en tiempo real tal como audio y video, etc. A pesar del incremento de ancho de banda disponible mediante la implementación de nuevas tecnologías de red, este sigue siendo un recurso limitado. Por lo que, mientras más aumente el ancho de banda disponible, más aplicaciones nuevas que consuman tales cantidades de ancho de banda surgirán.

Como resultado de esta situación, las aplicaciones requieren de diferentes niveles de servicio provisto por la red. Esto se puede hacer mediante la implementación de Mecanismos de Calidad de Servicio como Servicios Diferenciados (*DiffServ*) [1], Servicios Integrados (*IntServ*) [2] en conjunto con el Protocolo para la Reservación de Recursos (*RSVP*) [3] y el Multiprotocolo de Conmutación por Etiquetas (*MPLS*) [4]. La gestión de estas soluciones para ajustar la red conforme a las condiciones requeridas se le conoce como *Gestión de Calidad de Servicio (QoS Management)*. La *Gestión de Calidad de Servicio* se puede lograr a través de la variación de los parámetros de configuración de los diferentes mecanismos de Calidad de Servicio basándose en las métricas de la red y en los Acuerdos de Nivel de Servicio (*SLA*).

El problema principal que esta tesis toma es hacia Calidad de Servicio en transmisiones multimedia, en donde se desea proveer una transmisión con el menor retraso posible y una mínima perdida de paquetes. Esto con el fin de mejorar la Calidad de Experiencia que el usuario percibe durante una sesión multimedia. Los parámetros de la red pueden ser configurados con los distintos mecanismos de Calidad de Servicio para proveer el tipo de servicio deseado. Se utiliza el paradigma de

agentes para que de acuerdo las condiciones de la red estos manipulen y decidan que mecanismo y configuración utilizar.

Dentro de los objetivos de esta tesis se encuentra el construir una red basada en código libre que cuente con varios mecanismos de Calidad de Servicio, los cuales puedan ser manipulados y gestionados. Para la manipulación y gestión de estos, existe una plataforma basada en agentes a la cual se le hará una extensión para robustecer áreas como selección, base de conocimiento y capacidad de soporte de mecanismos QoS de la red. Como partes de esta extensión se considera complementar la ontología utilizada por los agentes e implementar un algoritmo de selección para elegir el mecanismo de Calidad de Servicio y su configuración óptima de acuerdo a las condiciones de la red. También se modelara mediante redes de Petri una descripción formal del proceso de Gestión de Calidad de Servicio que resulte de esta extensión. Finalmente, se desarrollarán una serie experimentos para evaluar la Calidad de Experiencia percibida por el usuario durante una sesión multimedia.

Esta tesis se organiza de la siguiente manera. En el capítulo 1 se resumen de manera breve los conceptos de Calidad de Servicio, Calidad de Experiencia y Gestión de Calidad de Servicio, además, se explica de manera breve como funcionan algunos de los mecanismos de Calidad de Servicio. En el capítulo 2 explica cómo construir una red que soporte los mecanismos de Calidad de Servicio, y cuyo propósito es ser gestionados. En el capítulo 3 se exponen diversas ontologías para la Gestión de Calidad de Servicio y se propone una ontología para el marco de trabajo de Calidad de Experiencia, la cual utilizará esta solución multiagentes. En el capítulo 4 se muestra la arquitectura del Sistema Multiagentes para llevar a cabo la Gestión de Calidad de Servicio y el algoritmo para la toma de decisiones de los mecanismos QoS y su configuración. En el capítulo 5 se hace una descripción formal mediante Redes de Petri del proceso de Gestión de Calidad de Servicio. En el capítulo 6 se detallan el escenario de experimentación, las pruebas realizadas y los resultados obtenidos de los experimentos. Finalmente se mencionan las conclusiones de esta tesis y el trabajo futuro que esta conlleva.

CAPITULO 1

1 Marco Teórico

En este capítulo se resumen de manera breve los conceptos base de esta tesis. Entre ellos se encuentran los conceptos de Calidad de Servicio, Calidad de Experiencia y Gestión de Calidad de Servicio. También se hace una breve explicación de los diferentes mecanismos de Calidad de Servicio como Servicios Diferenciados, Servicios Integrados y el Multiprotocolo por Conmutación de Etiquetas, los cuales son gestionados por la plataforma propuesta. Con ello se identifican las propiedades relevantes para su control.

1.1 Calidad de Servicio (Quality of Service – QoS)

En la actualidad las redes proveen el servicio *best-effort*. Este es, que los paquetes son procesados de la mejor y más rápida manera posible para hacerlos llegar hasta su destino, aunque no se tiene certeza de que estos finalmente serán entregados.

Se podría pensar que si el ancho de banda de las conexiones a Internet se incrementara de manera ilimitada, por ejemplo, con el uso de conexiones de fibra de óptica, entonces tan solo con servicios como el de *best-effort* bastarían para satisfacer la demanda que solicitasen los usuarios. Sin embargo, inevitablemente surgirían aplicaciones que requirieran de todo ese ancho de banda causando que una vez más este sea insuficiente. Por lo tanto, se deben proponer soluciones a este problema.

El objetivo principal de Calidad de Servicio es proveer de un trato preferencial al flujo de una transmisión en particular. Esto significa que una transmisión entregará exitosamente todos sus paquetes.

Es de pensar que habrá usuarios que demanden diferentes clases de servicio basados en sus necesidades de negocio. Por tanto, las empresas que proveen de servicios de Internet pueden prestar varias clases de servicio debido a que ya existen equipos (ruteadores/switches) que proveen de algún tipo de QoS. Así, quien desee una mejor transmisión para la transferencia de información pagará más

para obtener tal clase de servicio, mientras que para quienes sus necesidades no sean tan ambiciosas y solo necesiten de conectividad pueden adquirir servicios cuyo precio no sea elevado.

En la IETF (*Internet Engieneering Task Force*) se han propuesto varios mecanismos que permiten proveer Calidad de Servicio. Entre ellos destacan Servicios Diferenciados, Servicios Integrados, MPLS e Ingeniería de Tráfico.

En Servicios Diferenciados los paquetes son clasificados con un código para crear diferentes clases de paquetes. Los paquetes de diferentes clases reciben diferente servicio, de tal modo que se crean diferentes niveles de servicio.

Servicios Integrados propone la reservación de recursos de la red en lugar de la clasificación de paquetes. Para esto utiliza un protocolo de señalización llamado RSVP (*Resource Reservation Protocol*) el cual establece una ruta para la transmisión de paquetes y reserva recursos de la red para dicha transmisión.

Cuando los paquetes de datos ingresan a un dominio que soporta MPLS (*Multiprotocol Label Switching*) se les asigna una etiqueta con la cual son reenviados a través de este. Todo procesamiento futuro dentro del dominio MPLS se basa únicamente en la etiqueta, lo cual provoca un avance de paquetes más rápido.

Es necesario dar una definición estándar de Calidad de Servicio y para ello se utiliza la que propone la ITU-T (*International Telecommunication Union – Telecommunication*) en su recomendación E.800:

"*Calidad de Servicio, es el efecto colectivo del desempeño de un servicio el cual determina el grado de satisfacción de un usuario con respecto a dicho servicio*"

De esta definición podemos comentar que todo esfuerzo que se haga para lograr un impacto positivo en el desempeño de la red y la satisfacción del usuario forma parte de Calidad de Servicio.

1.2 Servicios Diferenciados

Servicios Diferenciados (*DiffServ*) está definido en el RFC 2474. El objetivo primordial de Servicios Diferenciados es permitir que haya diferentes niveles de servicio para proveer a los distintos flujos de tráfico en una infraestructura de red en común, basándose en el marcado de los paquetes.

1.2.1 Arquitectura de Servicios Diferenciados

La arquitectura de Servicios Diferenciados se define el RFC 2475. Esta se basa en un modelo en donde los paquetes que ingresan a un domino de red son clasificados y acondicionados. Esto se lleva a cabo en los nodos situados a los extremos del domino, con lo cual se les asignan a los paquetes diferentes comportamientos conforme a ciertas políticas.

Clasificación y acondicionamiento de tráfico

En la clasificación de paquetes, lo que se hace es identificar el tráfico que recibirá un servicio diferenciado a través del acondicionamiento que se le den a los paquetes. En el acondicionamiento de paquetes se realiza la medición, moldeado, remarcado y descarte de estos para asegurar que el tráfico que ingresa al dominio de DiffServ cumpla con las reglas especificadas en el *Acuerdo de Acondicionamiento de Tráfico* (*TCA – Traffic Conditioning Agreement*) [1], conforme a la política de provisión de servicios.

El *clasificador de paquetes*, selecciona los paquetes del flujo de tráfico basándose en el contenido de partes del encabezado de los paquetes, como el campo DS. El *perfil de tráfico* especifica las propiedades de un flujo que está siendo transmitido y estas pueden ser utilizadas por el clasificador para su labor. Un perfil puede estar basado en un *Token bucket*.

---

[1] En el TCA están especificadas las reglas de clasificación, medición, marcado, moldeado y descarte de paquetes que serán a aplicadas a los flujos de tráfico.

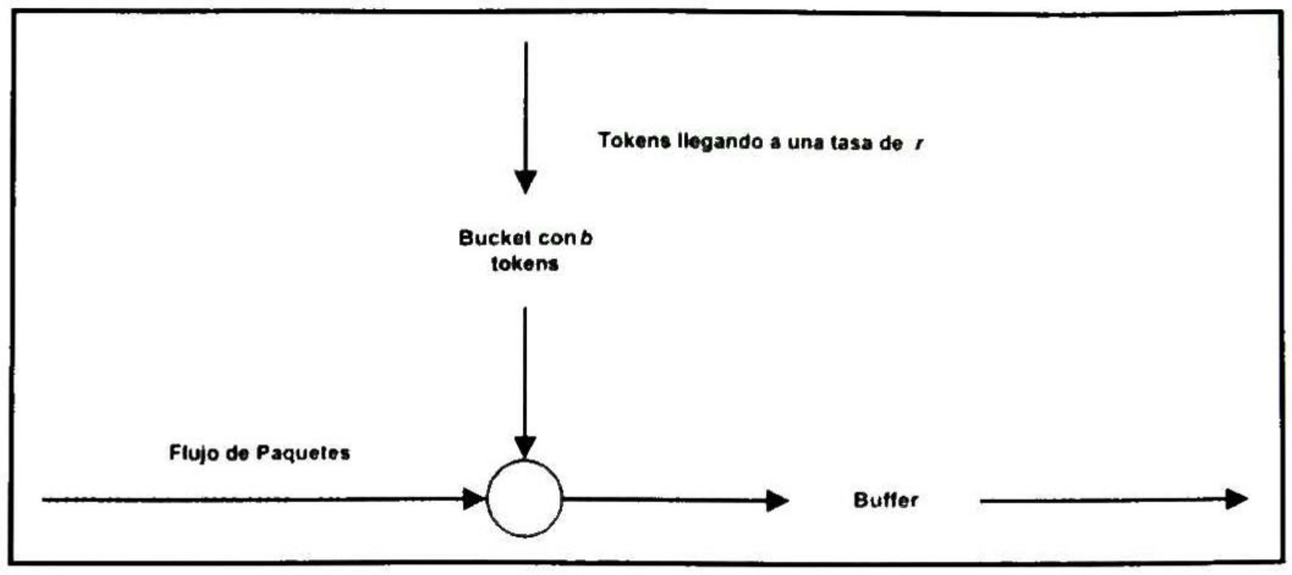


Figura 1.1 Filtro de tráfico token bucket.

Los paquetes que se encuentran dentro del perfil se les permiten entrar al dominio de DiffServ sin la necesidad de acondicionamiento posterior. Sin embargo, los paquetes que llegan fuera del perfil son almacenados hasta que se encuentren dentro del perfil (son moldeados), descartados, marcados con un nuevo código (remarcados) o inmediatamente reenviados sin ningún cambio.

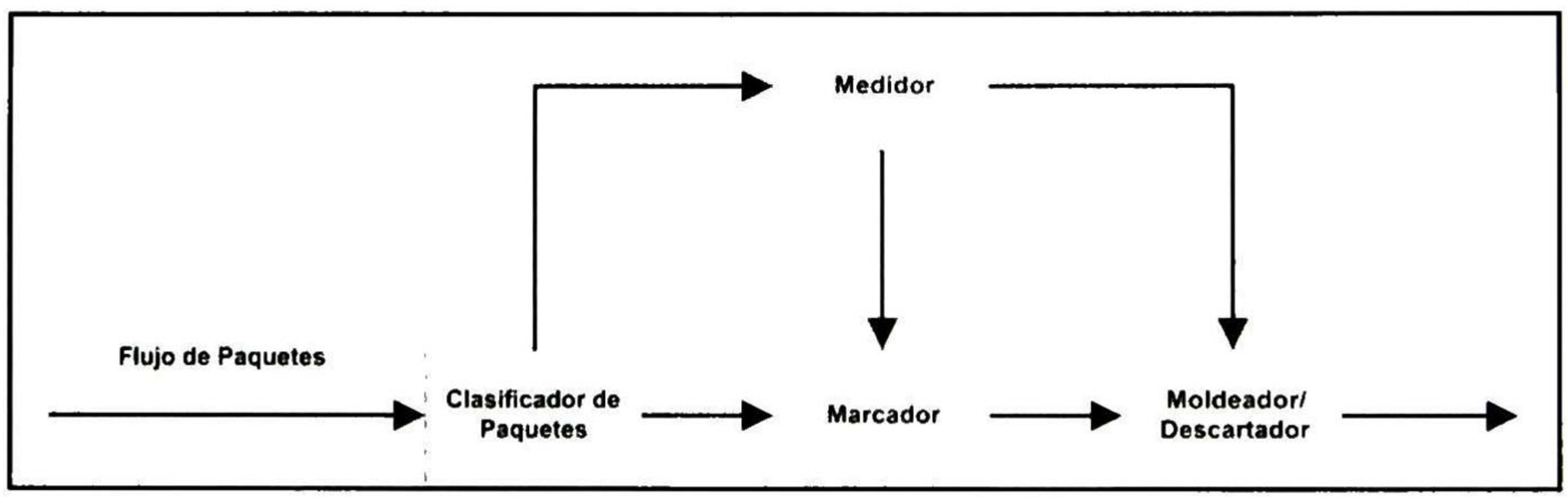


Figura 1.2 Acondicionamiento de tráfico en DiffServ.

1.2.2 Assured Forwarding (AF)

Assured Forwarding está definido en el RFC 2597. Es un comportamiento que soporta el reenvío de paquetes en diferentes clases. Cada clase permite que los paquetes tengan diferentes niveles de precedencia para ser descartados. Actualmente, para uso general, están definidas cuatro diferentes clases, cada una con tres niveles de precedencia para descarte de paquetes. Dentro de una clase, un nodo que soporta DiffServ debe de reenviar paquetes con mayor probabilidad para aquellos que tienen un menor nivel de precedencia y así descartar también con mayor probabilidad de aquellos paquetes que tienen un mayor nivel precedencia. Los valores de los códigos para AF recomendados por la IETF son:

AF11 = '001010'. AF12 '001100', AF13 '001110'
AF21 = '010010', AF22 '010100'. AF23 '010110'
AF31 '011010'. AF32 '011100', AF33 '011110'
AF41 '100010', AF42 '100100', AF43 '100110'

Prioridad de Descarte	Clase 1	Clase 2	Clase 3	Clase 4
Bajo	001010	010010	011010	100010
Medio	001100	010100	011100	100100
Alto	001110	010110	011110	100110

1.2.3 Expedited Forwarding (EF)

Expedited Forwarding se definió en el RFC 2598. Este comportamiento permite soportar una transmisión baja en perdida de paquetes, *jitter* y latencia con un ancho de banda garantizado dentro de un dominio de DiffServ. EF se define como el tratamiento para reenvío de paquetes en un flujo de DiffServ en particular, donde la tasa de partida de los paquetes de dicho flujo en cualquiera de los nodos que soportan DiffServ, debe ser mayor o igual a una tasa de transmisión configurable. Esto quiere decir que, la tasa con la que parten los paquetes de EF de un nodo, debe ser mayor o igual a la tasa con la que ingresan a ese mismo nodo. El valor del código recomendado por la IETF para EF es: '101110'

1.2.4 Simulación con Servicios Diferenciados

Para una mejor comprensión y entendimiento de este mecanismo de Calidad de Servicio se presenta una simulación de DiffServ implementada en el simulador de redes NS2 [5]. NS2 incluye a NAM (*Network AniMator*) el cual reproduce una animación de la simulación implementada y permite utilizar herramientas como Gnuplot y XGraph para obtener estadísticas y representarlas gráficamente. NS2 por defecto incluye ya un modulo para implementar DiffServ. NS2 es utilizado por compañías de gran prestigio en cuanto a telecomunicaciones se refiere, entre ellas, Nortel Networks.

La simulación fue implementada en la versión 2.29 de NS2. El escenario consiste de una red en la que se encuentran siete nodos. Tres de ellos se encargan de enviar tráfico UDP por sus respectivos enlaces hacia un nodo destino con una tasa de transmisión constante de 2Mb, 3Mb y 3Mb respectivamente. De estos tres nodos emisores, el flujo de los dos primeros cuenta con DiffServ, mientras que el flujo del tercero solo cuenta con *best-effort*. Los flujos de tráfico atraviesan por un

dominio de DiffServ compuesto de tres nodos ruteadores previos al nodo destino. Dentro de este dominio de DiffServ existe un cuello de botella debido a un enlace de 5Mb que se ve saturado al momento en que los tres nodos emisores transmiten sus flujos al mismo tiempo. El nodo ruteador que se localiza en el cuello de botella implementa la disciplina RED (*Random Early Detection*) para manejar los paquetes marcados, los cuales deberán recibir un trato preferencial.

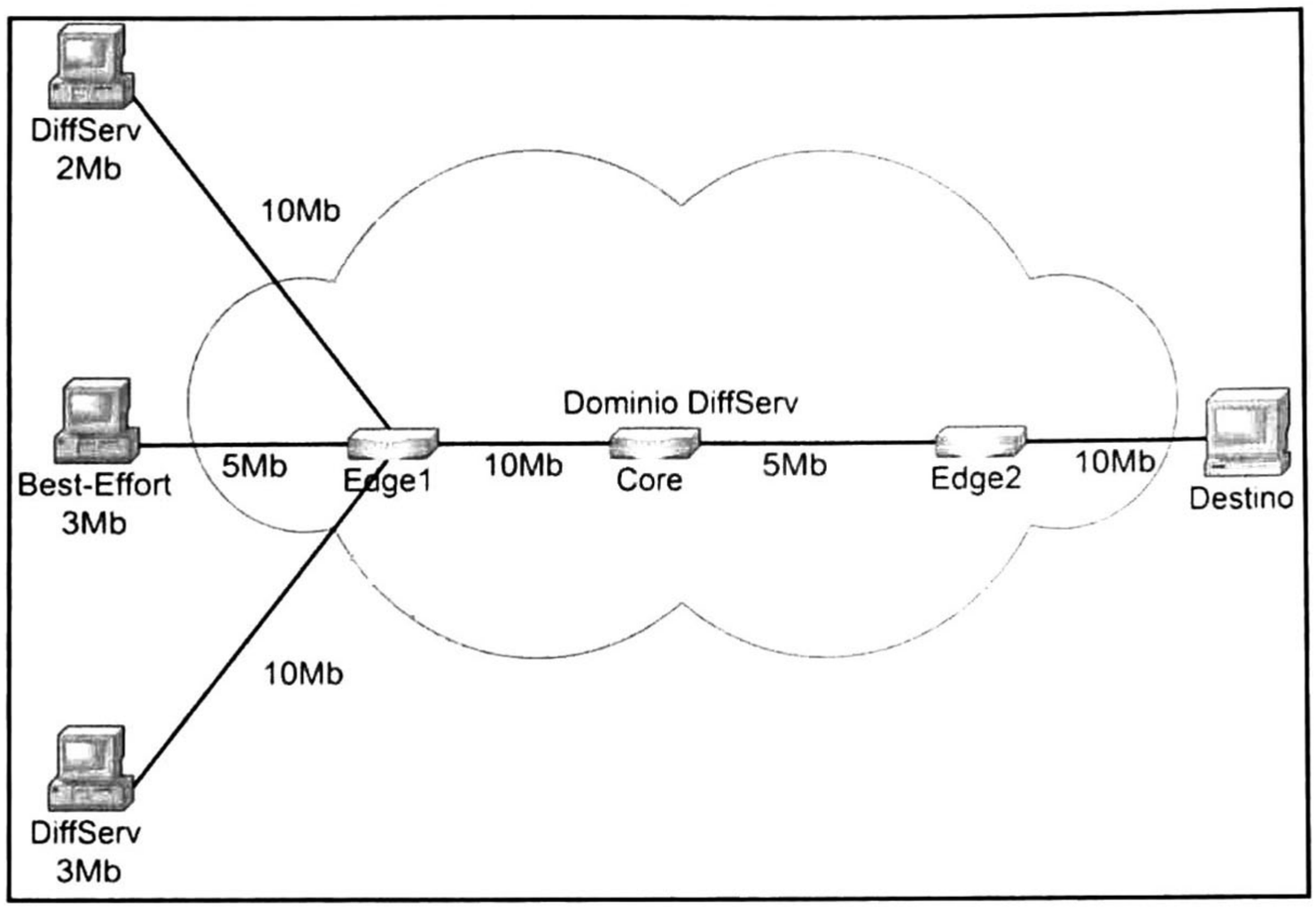


Figura 1.3 Escenario de red para la simulación de DiffServ.

NS2 antes de iniciar la simulación muestra una serie de tablas en las cuales se basan las políticas y el comportamiento con las que procederá la simulación. Las tablas mostradas para esta simulación son las siguientes:

```
Policy Table(3):
Flow (0 to 5): Token Bucket policer, initial code point 10, CIR 2000000.0 bps, CBS 3000.0 bytes.
Flow (1 to 5): Token Bucket policer, initial code point 10, CIR 2000000.0 bps, CBS 10000.0 bytes.
Flow (6 to 5): Token Bucket policer, initial code point 0, CIR 1000000.0 bps, CBS 1000.0 bytes.
Policer Table:
Token Bucket policer code point 10 is policed to code point 11.
Token Bucket policer code point 0 is policed to code point 0.
PHB Table:
Code Point 10 is associated with Queue 0, Precedence 0
Code Point 11 is associated with Queue 0, Precedence 1
Code Point 0 is associated with Queue 0, Precedence 2
```

En la *Policy Table* se pueden observar los valores del código DS que toma cada uno de los flujos de tráfico que partirán hacia el nodo destino. Se identifican los flujos con DiffServ que de entrada tienen el código DS `10` y el flujo *best-effort* que cuenta con el código DS `0` por defecto. Además, se muestra que los flujos de tráfico serán medidos contra un *token bucket* cuyos parámetros *r* y *b* son también desplegados. La *Policer Table* indica las políticas a seguir por el *token bucket* en donde el código DS `10` se puede degradar con un valor de `11` en los momentos de congestión. Mientras tanto el tráfico *best-effor* permanecerá con el mismo valor bajo cualquier condición. En la *PHB Table* se puede apreciar cómo queda conformada una clase junto con sus niveles de precedencia. En este caso, los paquetes con código DS `10` son los que tienen el nivel de precedencia más bajo, por lo que serán los últimos en ser descartados. Los paquetes con código DS `11` tienen un nivel de precedencia medio y podrán ser descartados con una probabilidad un poco mayor. En último lugar, los paquetes *best-effort* con código DS `0` serán descartados indiscriminadamente en cuanto ocurra una condición de congestión en la red, esto debido a que poseen el nivel de precedencia más alto.

Ilustrándose con la animación proporcionada por NAM en el instante de congestión, se observa que los paquetes provenientes del nodo 6 cuyo tráfico es *best-effort*, son descartados con una probabilidad mucho mayor que el resto. Los paquetes provenientes de los nodos 0 y 1, que obviamente cuentan con DiffServ, arriban a su destino, mientras que los del nodo 6 no.

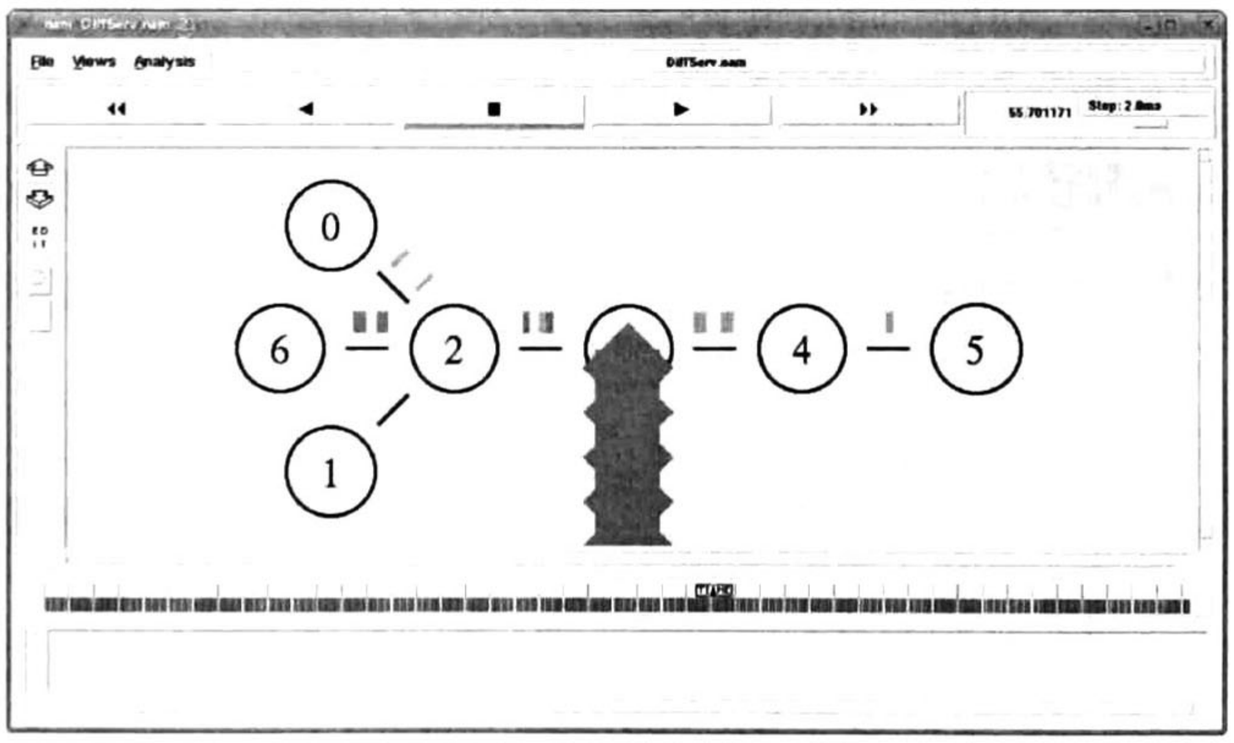


Figura 1.4 Animación reproducida por NAM en el instante de congestión.

En la grafica obtenida por XGraph a partir de los archivos de rastreo generados por NS2, se observa que en el instante 10, el flujo *best-effort* de 3Mb comienza su transmisión sin ningún problema.

En el instante 20, la primera transmisión DiffServ de 3Mb comienza y disminuye a 2Mb la transmisión *best-effort*, esto debido a que el enlace donde ocurre el cuello de botella soporta solo 5Mb. En el instante 50, la segunda transmisión DiffServ de 2Mb comienza y disminuye a 0Mb la transmisión *best-effort*, ya que las dos transmisiones DiffServ ocupan los 5Mb del enlace. Hasta el instante 75, el flujo *best-effort* se recupera porque el primer flujo DiffServ termina su transmisión y así se conserva hasta terminar la simulación. En esta simulación los flujos con DiffServ han recibido un trato preferencial sobre el de *best-effort*, provocando que este se degrade hasta el punto de una tasa de recepción nula.

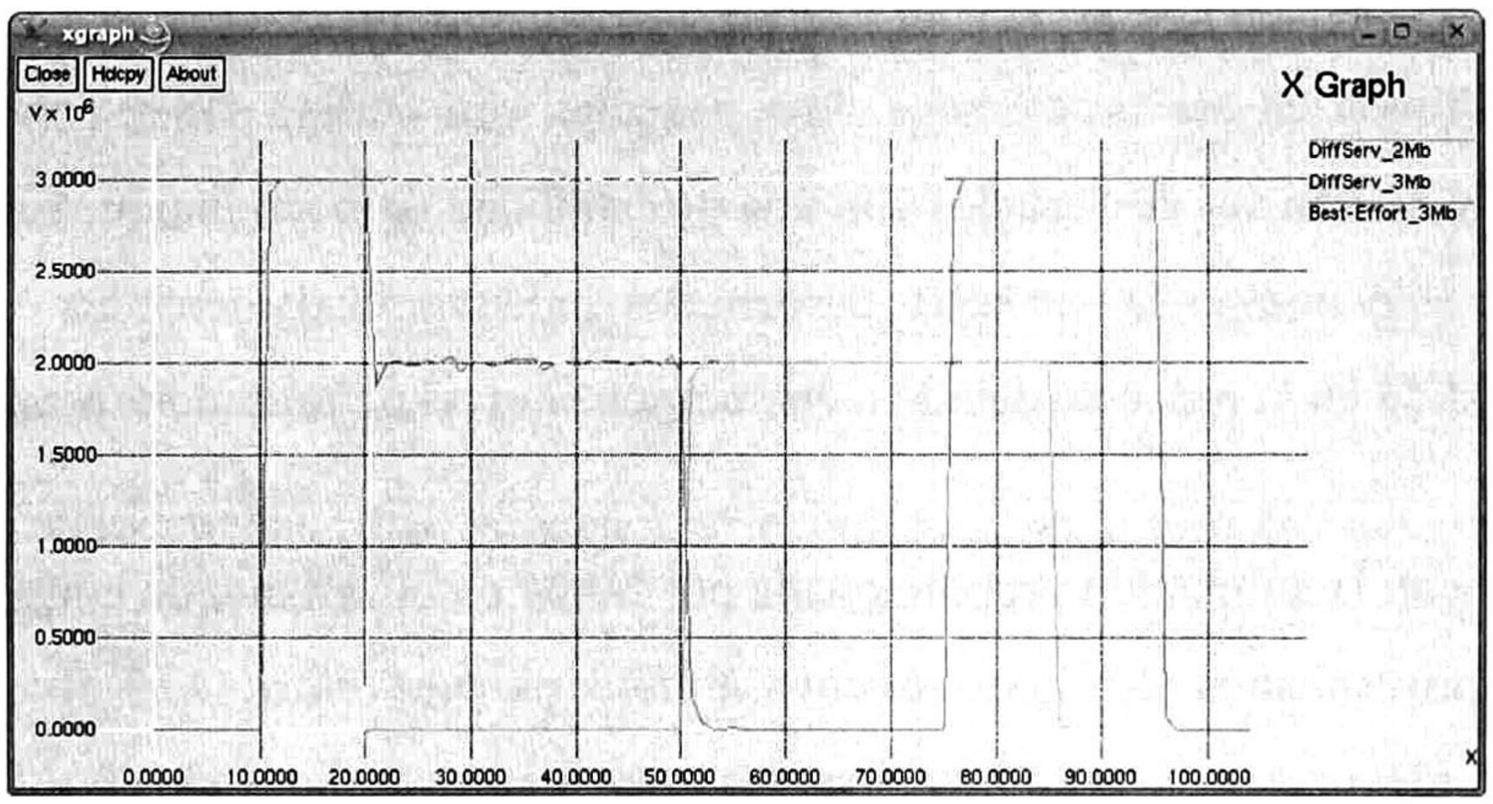


Figura 1.5 Tasa de recepción de cada flujo en el nodo destino contra el tiempo.

Por último, NS2 en su modulo para DiffServ puede arrojar estadísticas de los paquetes con el siguiente formato:

```
Packets Statistics
=======================================
 CP  TotPkts   TxPkts   ldrops   edrops
 --  -------   ------   ------   ------
All    59371    46219    13137       15
  0    29996    16859    13137        0
 10    22509    22509        0        0
 11     6866     6851        0       15
```

Las estadísticas mostradas corresponden a las de la simulación previa. Las nomenclaturas son las siguientes: *CP*: Código DS. *TotPkts*: Total de paquetes recibidos. *TxPkts*: Paquetes transmitidos. *ldrops*: paquetes descartados debido a la saturación del enlace. *edrops*: paquetes descartados debido a la disciplina de encolamiento.

1.3 Servicios Integrados

Debido al gran auge de las aplicaciones de tiempo real como las transmisiones multimedia, se propuso una clase de servicio que garantice cierta cantidad de ancho de banda además de un retraso y pérdida de paquetes mínimo en una transmisión. Este modelo se llama Servicios Integrados (*IntServ*) y se define en el RFC 1633. Está basado en la reservación de recursos de la red y un control de admisión a estos mismos.

1.3.1 Arquitectura de Servicios Integrados

La arquitectura de Servicios Integrados está formada por cuatro elementos. El primero de ellos es el *planificador de paquetes* que maneja el reenvío de los paquetes de los diferentes flujos utilizando disciplinas de encolamiento. Su función básica es reordenar los paquetes del buffer de salida. El *clasificador*; que tiene el propósito de hacer un mapeo de cada paquete que ingresa hacia alguna clase. Todos los paquetes que pertenecen a una misma clase reciben el mismo trato por el *planificador de paquetes*. La clasificación del paquete se puede basar en el contenido de algunos de los campos de su encabezado o agregando algún código para su clasificación. El *control de admisión* implementa el algoritmo de decisión que un nodo usa para determinar cuando a un nuevo flujo se le pueden otorgar recursos sin causar un impacto negativo en las reservaciones previas. El *control de admisión* se invoca en cada nodo para tomar una decisión de otorgar o rechazar la reservación, al tiempo que el nodo propaga esta solicitud a lo largo de la ruta. El *protocolo de reservación* es necesario para crear y mantener un estado específico del flujo desde el emisor y el receptor a través de los nodos intermedios de la ruta del flujo. A este protocolo se le conoce como RSVP (*ReSerVation Protocol*).

1.3.2 Protocolo para la Reservación de Recursos (RSVP – Resource Reservation Protocol)

RSVP está definido en el RFC 2205. RSVP establece reservación de recursos de manera *iniciada por el receptor*, es decir, el receptor es quien solicita la transmisión, para flujos *unicast* o *multicast*, con buenas propiedades de escalabilidad y robustez. RSVP fue diseñado para IntServ. Este protocolo es utilizado por un host para solicitar cierta Calidad de Servicio de la red para el flujo de una de sus aplicaciones. En los ruteadores, es utilizado para entregar las solicitudes de Calidad de Servicio

a todos los nodos a lo largo de la ruta del flujo, y establecer y mantener el estado de la reservación para proveer el servicio solicitado. Las solicitudes de RSVP por lo general resultan en recursos que se reservan.

Una solicitud elemental de reservación en RSVP consiste de una especificación de flujo y una especificación de filtrado. A estos dos juntos se les llama descriptor del flujo. La especificación del flujo indica la Calidad de Servicio deseada, mientras que la especificación de filtrado define el flujo de paquetes que recibirá la Calidad de Servicio descrita en la especificación del flujo. La especificación de flujo establece parámetros en el planificador de paquetes del nodo y la especificación de filtrado establece parámetros en el clasificador de paquetes. Los paquetes que no concuerdan con alguna especificación de filtrado se procesan como tráfico *best-effort*.

Hay dos mensajes fundamentales en RSVP: *Resv* y *Path.* Cada nodo receptor envía un mensaje de solicitud de reservación (*Resv*) hacia el emisor. Este mensaje debe de seguir exactamente la ruta inversa que seguirán los paquetes de datos. También crea y mantiene el estado de la reservación a lo largo de toda la ruta. Cada nodo emisor transmite un mensaje RSVP *Path* de regreso por la ruta que se reserva, la cual seguirán los paquetes de datos.

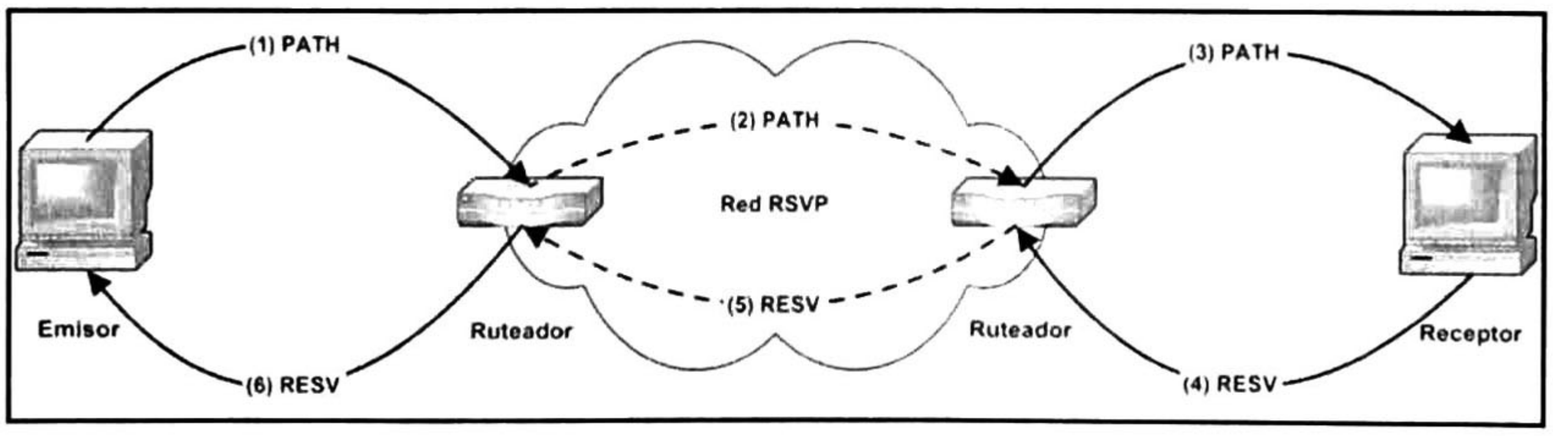


Figura 1.6 Proceso de señalización de RSVP

1.3.3 Controlled Load Services

El servicio de Carga Controlada se definió en el RFC 2211. Este provee a un flujo de datos una Calidad de Servicio cercana a la que un elemento de red proporcionaría a un flujo bajo el servicio de tipo *best-effort* en condiciones de no congestión. Para esto utiliza un control de admisión que asegure que tal servicio será provisto aun cuando el elemento de red se encuentre en condiciones de congestión. Así, un gran porcentaje de los paquetes transmitidos en la red serán entregados con éxito a su destino.

El servicio de Carga Controlada está pensado para soportar aplicaciones que son altamente sensibles a las condiciones de congestión. A este tipo de aplicaciones se les llama *aplicaciones de tiempo real adaptables*. Estas aplicaciones trabajan en óptimas condiciones en redes no congestionadas, pero pierden notablemente su rendimiento cuando la red está en condiciones de congestión. Algunos ejemplos son: aplicaciones para la transmisión de audio y video digitalizado, y aplicaciones para la transmisión continua de datos multimedia.

1.3.4 Guaranteed Services

Servicios Garantizados está definido el RFC 2212. Este servicio provee de una cota en el retraso de paquetes garantizada, así como también garantiza una cierta cantidad de ancho de banda. La cota en el retraso de paquetes se garantiza únicamente en cuanto al tiempo de almacenamiento de estos en un elemento de la red. Servicios Garantizados, como su nombre lo indica, garantiza que los paquetes llegarán dentro del tiempo de entrega garantizado, y que no sufrirán de pérdidas debido al desbordamiento en los buffers de almacenamiento de los equipos de red. Todo esto, si el flujo de tráfico se mantiene dentro de los parámetros de tráfico especificados.

1.3.5 Simulación con Servicios Integrados

Ahora se muestra una simulación en NS2 que implementa IntServ. Para poder implementar IntServ en NS2 es necesario aplicar previamente el parche RSVP/NS hecho por Marc Greis. Esta simulación fue implementada en la versión 2.29 de NS.

El escenario está conformado por cinco nodos. Tres de estos nodos transmiten un flujo de paquetes a una tasa de transmisión de 500Kb cada uno hacia un nodo destino. El tráfico de estos nodos pasa por un ruteador que tiene soporte para IntServ el cual tiene el enlace hacia el nodo destino. En este enlace se forma un cuello de botella ya que solo soporta un ancho de banda de 1Mb y cuando los tres emisores transmiten al mismo tiempo la tasa de transferencia total suma 1.5Mb. En algún instante uno de los emisores solicita una reservación para su flujo de paquetes y es así como este flujo se ve privilegiado con respecto a los demás.

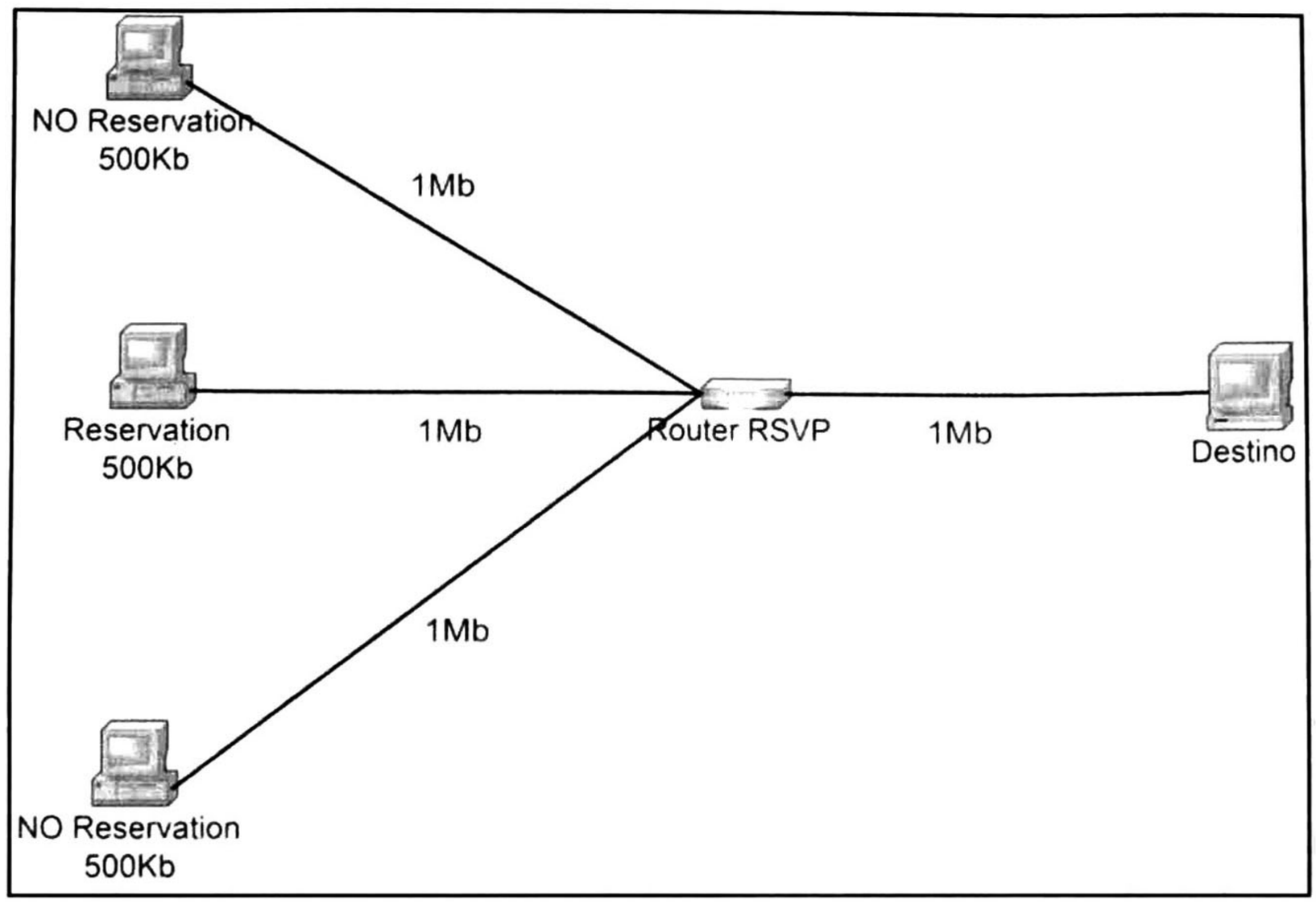


Figura 1.7 Escenario de red para la simulación de IntServ.

De la animación provista por NAM durante el congestionamiento se puede observar que el flujo de paquetes que hizo la reservación de recursos no ve afectada su transmisión ya que sus paquetes arriban al nodo destino sin ningún contratiempo. En los flujos que no cuentan con una reservación sus paquetes son descartados de igual manera ya que son tratados como tráfico de tipo *best-effort*, por lo que no todos ellos llegan a su destino.

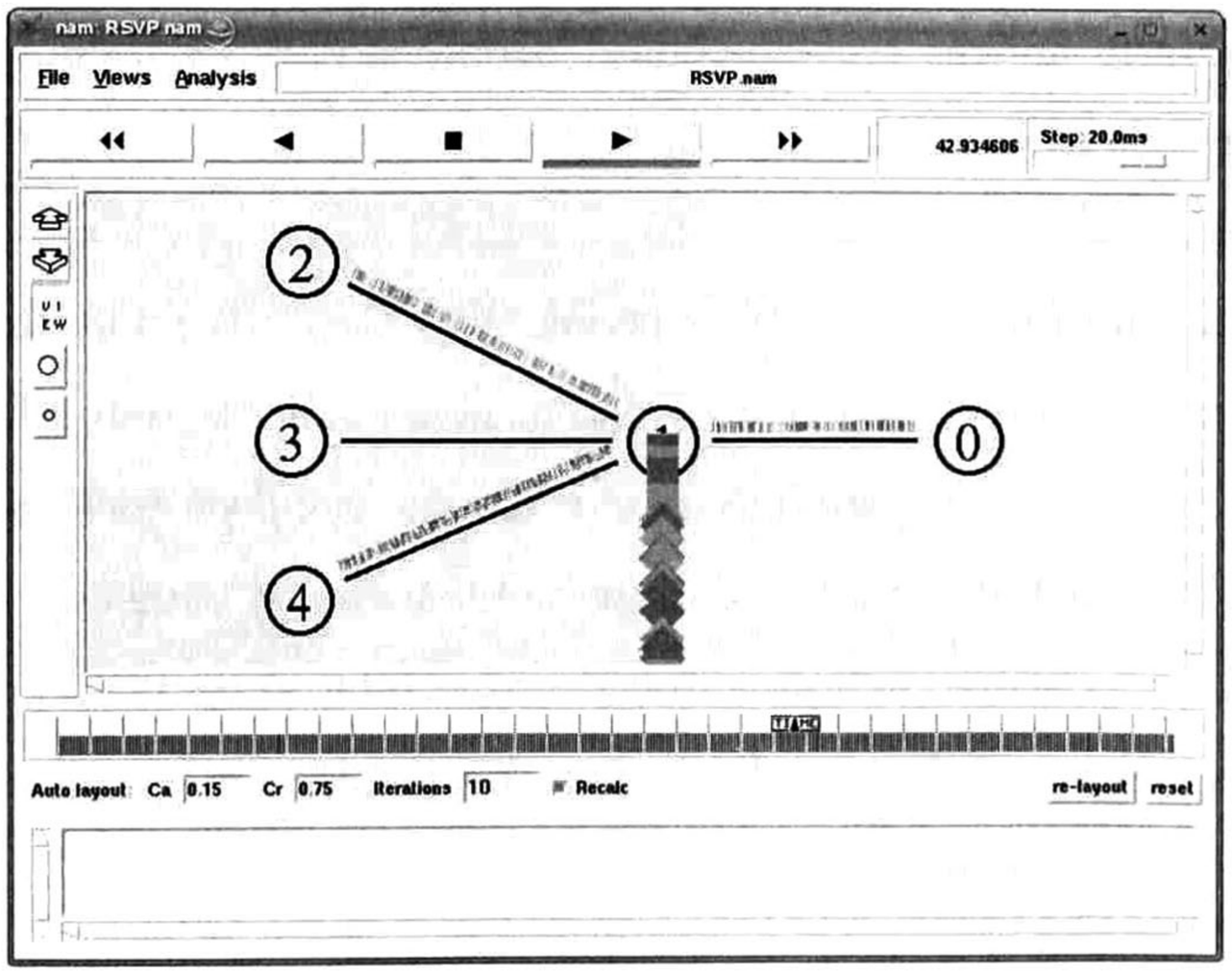


Figura 1.8 Animación mostrada por NAM en el instante de congestión con una reservación ya hecha.

De la grafica generada se puede observar que en el instante 12 los tres nodos emisores comienzan a transmitir simultáneamente. Al saturar de manera inmediata el enlace y al contar todos los flujos solo con servicio de tipo *best-effort*, estos tienen que compartir el ancho de banda disponible, por lo que todos ellos ven afectada su tasa de transmisión. En el instante 32, uno de los flujos ha establecido una reservación de recursos, por lo que se incrementa su tasa de transmisión hacia el nodo destino. Mientras tanto los flujos restantes ven disminuida aun más su tasa de transferencia debido a que el ancho de banda disponible para compartir ahora es menor. Después de la reservación, solo el flujo que pertenece a ella entrega en tiempo y forma sus paquetes al nodo destino. Los paquetes del resto de los flujos no llegan por completo a su destino y así permanecen hasta terminar la simulación. El flujo que estableció la reservación de recursos vio cumplidas sus expectativas de transmisión, mientras que para el resto de los flujos no fue así.

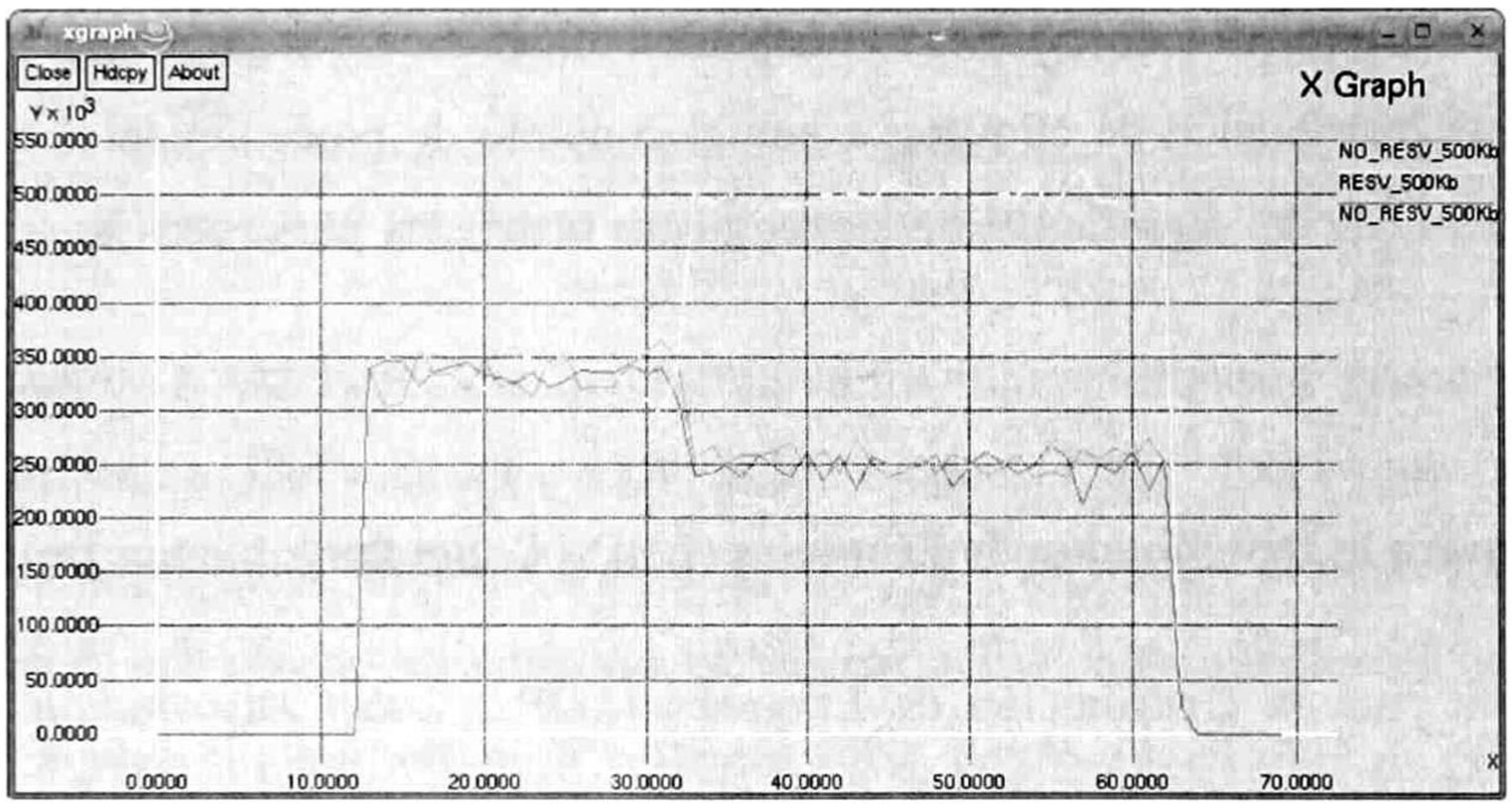


Figura 1.9 Tasa de recepción de cada flujo en el nodo destino contra el tiempo.

1.4 Multiprotocolo por Conmutación de Etiquetas (Multiprotocol Label Switching - MPLS)

Cuando un paquete viaja por la red, este va de un ruteador a otro, y en cada uno de ellos se tiene que tomar la decisión de hacia dónde reenviar el paquete para que llegue a su destino. Para eso, el ruteador lee y analiza la información contenida en el encabezado del paquete. Pero, en muchas ocasiones la información que viene dentro del encabezado es más de la necesaria para elegir cuál será el siguiente salto. En MPLS a los paquetes se les asigna una clase de reenvío en particular solo una vez, y esto es cuando el paquete entra en el dominio. La clase de reenvío a la cual pertenece el paquete se codifica en un valor corto y de longitud fija llamado *etiqueta*. MPLS está definido en el RFC 3031.

1.4.1 Arquitectura para MPLS

Cuando el paquete se reenvía hacia su siguiente salto este es enviado junto con su etiqueta, es decir, el paquete es etiquetado antes de ser reenviado. En los saltos siguientes ya no existe la necesidad de analizar el encabezado del paquete. En su lugar la etiqueta se utiliza como índice de una tabla la cual especifica el siguiente salto y una nueva etiqueta para el paquete. La vieja etiqueta es reemplazada por la nueva y se vuelve a reenviar el paquete hacia su siguiente salto. El paradigma de reenvío en MPLS es que una vez que al paquete se le ha asignado una clase de reenvío ya no existe la necesidad de que los ruteadores subsecuentes analicen el encabezado del paquete. De esta manera el reenvío de los paquetes esta guiado por las etiquetas.

Para establecer y mantener la asociación de etiquetas entre dos LSRs (*Label Switched Router*) en una red MPLS se usa un protocolo para la distribución etiquetas. La arquitectura de MPLS define a un protocolo para la distribución de etiquetas como el conjunto de procedimientos con los cuales un LSR informa a otro LSR del significado de sus etiquetas utilizadas para reenviar tráfico a través de ellos.

1.4.2 Protocolo para la Distribución de Etiquetas (LDP - Label Distribution Protocol)

El Protocolo para la Distribución de Etiquetas (*LDP Label Distribution Protocol*) es un protocolo para la distribución de etiquetas. Este es el conjunto de procedimientos y mensajes con los cuales los LSRs establecen LSPs (*Label Switched Path*) a través de la red, mediante las asociaciones etiqueta/clase de reenvío que estos han hecho. LDP asocia una clase de reenvío con el LSP que crea. La clase de reenvío asociada con el LSP especifica que paquetes corresponden al LSP LDP se define en el RFC 3036.

1.4.3 Simulación con MPLS

Ahora se muestra una simulación de MPLS en NS2. Es importante mencionar que para poder implementar MPLS en NS2 es necesario instalar previamente el parche mns-for-2.26 (MPLS *for NS2.26*) cuyo autor es Christian Callegari en la versión 2.26 de NS2.

El escenario está compuesto por once nodos. Dos de ellos funcionan como emisores de un flujo de paquetes y otros dos como los receptores. El resto de los nodos son LSRs que pertenecen a un mismo dominio de MPLS. Un flujo de paquetes va del nodo fuente 1 al nodo destino 10, mientras que el otro flujo de paquetes va del nodo fuente 0 al nodo destino 9. Todos los enlaces en el escenario son de 1Mb y ambos flujos de paquetes tienen una tasa de 400Kb.

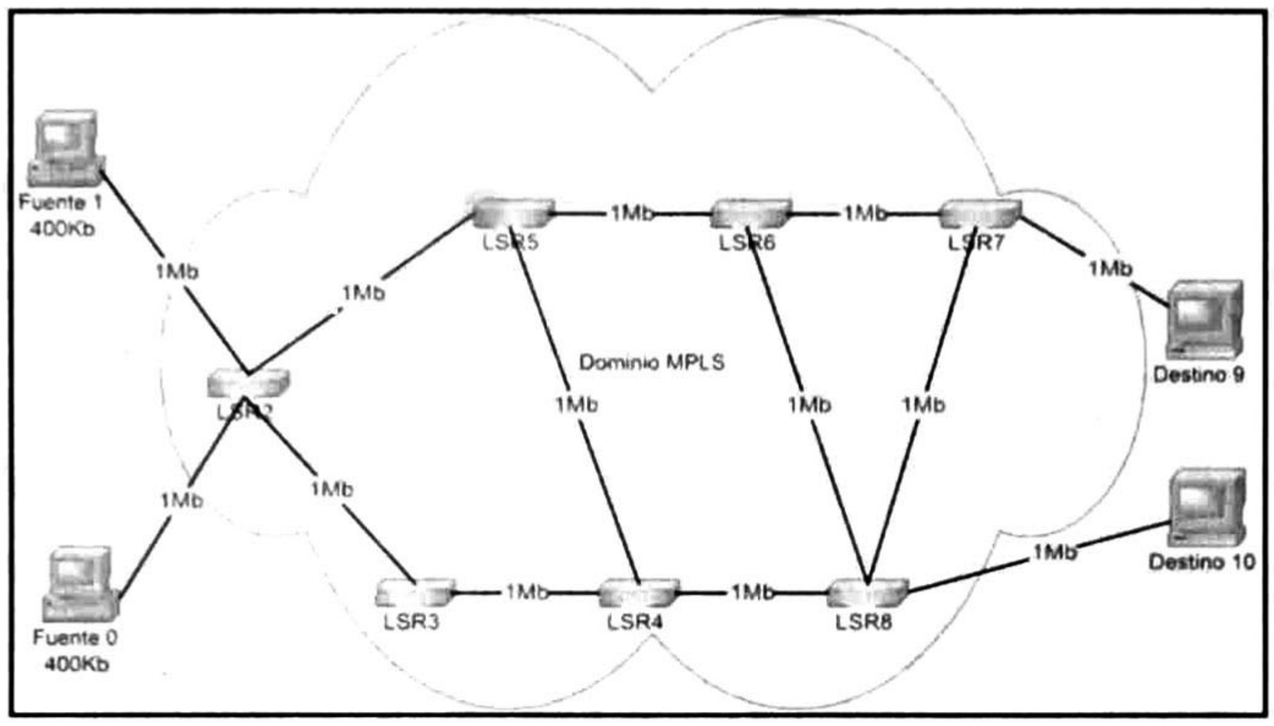


Figura 1.10 Escenario de red para la simulación de MPLS.

La animación de NAM explica la simulación en los siguientes pasos: Primero, figura 1.11 (a), comienzan a distribuirse los mensajes LDP para establecer los LSPs por los que viajaran respectivamente los paquetes del flujo del nodo 1 al nodo 10 y del nodo 0 al nodo 9. Enseguida, figura 1.11 (b), los flujos de los dos nodos emisores parten hacia sus correspondientes nodos destino. Los LSPs establecidos siguen el algoritmo de la ruta más corta, por lo que el flujo de paquetes que va del nodo 0 al nodo 9 sigue el LSP LSR2 – LSR5 – LSR6 – LSR7 y el LSP del flujo de paquetes que va del nodo 1 al nodo 10 es LSR2 – LSR3 – LSR4 – LSR8. Después, figura 1.11 (c), se efectúa una agregación de flujos. El flujo de paquetes que va del nodo 1 al nodo 10 sufre una agregación con el otro flujo. Esto provoca que en lugar de seguir el LSP LSR2 – LSR3 – LSR4 – LSR8 ahora seguirá el LSP LSR2 – LSR5 – LSR6 – LSR8. Con esto ambos flujos coinciden en un mismo segmento de sus LSPs, el segmento LSR2 – LSR5 – LSR6. En seguida, figura 1.11 (d), permanece solo un flujo de paquetes que va del nodo 0 al nodo 9. Se construye una ruta estática mediante un LSP la cual no es la más corta y se presume fue establecida por cuestiones de Ingeniería de Tráfico. El LSP está conformado por los nodos LSR2 – LSR5 – LSR4 – LSR8 – LSR6 – LSR7. Finalmente, figura 1.11 (e), se muestra un túnel por el cual viajan los paquetes del flujo del nodo 0 al nodo 9. El ingreso del túnel es por el LSR4, siguiendo el túnel por el LSR5 y LSR6, y egresar finalmente del túnel por el LSR8.

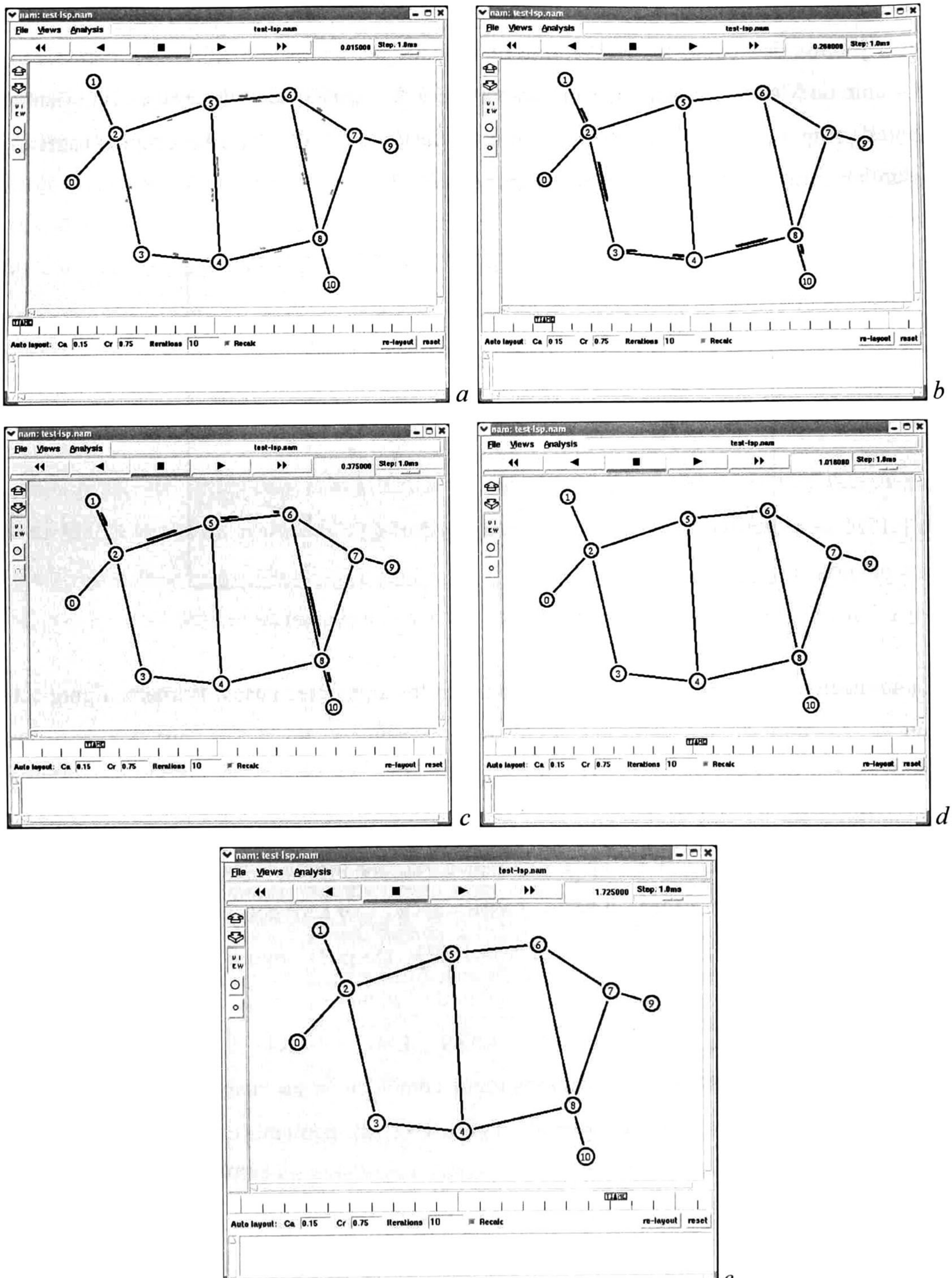


Figura 1.11 Animación en NAM de MPLS.

1.5 Gestión de Calidad de Servicio (QoS Management)

Network Management es una disciplina dentro de redes cuya función es la gestión de la de la red. Entre sus actividades están el manejo de configuraciones, rendimiento, seguridad, tolerancia a fallos y mantener registro de todos los sucesos importantes que ocurren en la red.

QoS Management es una de las áreas de *Network Management* que se encarga de obtener las características de tráfico en la red y con esta información validar cualquier configuración de Calidad de Servicio que se tenga, también a partir de esta información se pueden tomar medidas para una reconfiguración, es decir, modificar los parámetros para las seguir las políticas de Calidad de Servicio.

Cuando ya se tiene una configuración de Calidad de Servicio es importante evaluar los parámetros de las políticas que ésta usa para poder decidir cuando es necesario un incremento en el nivel de servicio. Para determinar si las políticas de Calidad de Servicio siguen siendo efectivas se realizan mediciones y comparaciones del estado de la red junto con los parámetros con los que se pretende cumplir la transmisión. El estado de la red debe de ser monitoreado constantemente ya que las características del tráfico varían continuamente.

1.6 Calidad de Experiencia (Quality of Experience – QoE)

Existen diferentes enfoques para la gestión de mecanismos de Calidad de Servicio. Estos varían desde la consideración de métricas de red objetivas hasta parámetros subjetivos que provienen del usuario final. Dentro de las primeras se pueden incluir métricas como *jitter, retraso, perdida de paquetes*, etc. Mientras que en las otras se podría incluir el *entendimiento*, la *satisfacción* y el *interés*.
El *entendimiento* muestra la compresión acerca del contenido que se percibe en una transmisión multimedia. La *satisfacción* indica que tan confortable se siente el usuario final con la calidad percibida. El *interés* denota que tanto le concierne al usuario final el contenido de la transmisión. Dos enfoques para la Gestión de Calidad de Servicio son: Calidad de Percepción (*Quality of Perseption QoP*) [6], y Calidad de Experiencia (*Quality of Experience – QoE*) [7].

Calidad de Experiencia se define como *"la experiencia que el usuario percibe por lo que se*

presenta en la Capa de Aplicación, en donde esta capa actúa como una interfaz la cual muestra el resultado final de toda la Calidad de Servicio provista individualmente por cada una de las capas previas"

Dentro del marco de trabajo de QoE se consideran dos factores subjetivos adicionales que son: el interés personal, y cuanto está dispuesto a pagar el usuario final por la transmisión. Cada usuario posee su propio grado de interés de lo que percibe, por tal motivo es que este es considerado un parámetro subjetivo. La experiencia del usuario se mide considerando dos métricas: Calidad de Servicio de la Aplicación (*Application Quality of Service – AQoS*), y Calidad de Servicio de la Red (*Network Quality of Service – NQoS*). A la primera métrica le concierne la codificación multimedia, la paquetización, etc. mientras que la otra involucra las estrategias de admisión, planificación, reservación y políticas de los ruteadores y demás equipos de red. Una parte fundamental de este marco de trabajo es la retroalimentación del usuario, ya que mediante las solicitudes que este realiza podrá mejorar o reducir la calidad de la transmisión que percibe en tiempo real. Esto debido a que las solicitudes de *mejorar* o *reducir* se pueden transformar en una serie de factores ponderados los cuales se pueden aplicar en las métricas *AQoS* y *NQoS*. El usuario basa sus decisiones de acuerdo a los parámetros subjetivos mencionados previamente. En este sentido, un usuario podría pagar a su proveedor de servicios multimedia de acuerdo al interés que este tenga en una transmisión en particular y de esa manera los requerimientos de calidad podrían también cambiar.

1.7 Conclusiones

En este capítulo se describieron conceptos fundamentales como el de Calidad de Servicio y se resumieron de manera breve los mecanismos con los cuales se puede proveer de esta. Entre ellos se encuentran Servicios Diferenciados, Servicios Integrados y Multiprotocolo por Conmutación de Etiquetas. También se mostraron simulaciones implementadas en NS2 de cada mecanismo de QoS. En ellas se pudo corroborar tanto el funcionamiento como las ventajas que los mecanismos de QoS proveen sobre otros tipos de servicio como *best-effort*. Finalmente se presentaron los conceptos de Gestión de Calidad de Servicio y Calidad de Experiencia que se ven involucrados en esta tesis. Con todo lo anterior se señalan los conceptos que dan origen a la plataforma propuesta.

CAPITULO 2

2 Construcción de una red experimental con los protocolos de Calidad de Servicio

En este capítulo se muestra como construir una red experimental basada en código abierto con soporte de diferentes protocolos de Calidad de Servicio. Dicha red puede ser considerada como cama de prueba (*test bed*) para la investigación en el área de QoS y transmisión de sistemas multimedia basadas en conmutación de paquetes.

2.1 Construcción de una red con el software de ruteo Zebra

GNU Zebra [8] es un software libre diseñado para sistemas operativos del tipo UNIX que maneja los protocolos de ruteo basados en TCP/IP Forma parte del proyecto GNU Project [9] y se distribuye bajo la licencia GNU General Public License [10]. Zebra tiene soporte para protocolos como BGP-4, RIPv1, RIPv2 y OSPF, además de soporte para IP versión 4 e IPv6. Zebra es único en su diseño ya que cada uno de estos protocolos corre en un proceso independiente.

2.1.1 Instalación

La instalación del software Zebra es muy sencilla. Lo primero que se tiene que hacer es descargar la versión más reciente desde el sitio web de Zebra y descomprimirlo en el directorio de preferencia. Para este laboratorio se utilizo la versión de Zebra 0.95a. Dentro del directorio donde se descomprimió Zebra se ejecutan los siguientes comandas para su instalación, no sin antes dejar de mencionar que se deben tener los privilegios del usuario root:

```
./configure
make
make install
```

La instalación por defecto colocara los demonios ejecutables en la ruta `/usr/local/bin` y los archivos de configuración en `/usr/local/etc`. Los demonios de Zebra tienen sus propias terminales de interfaz por lo que es necesario establecer los números de puerto por los cuales se debe conectar a ellos. Para ello, se deben agregar las siguientes líneas en el archivo `/etc/services`:

```
#Zebra routing ports
zebrasrv        2600/tcp                # zebra service
zebra           2601/tcp                # zebra vty
ripd            2602/tcp                # RIPd vty
ripngd          2603/tcp                # RIPngd vty
ospfd           2604/tcp                # OSPFd vty
bgpd            2605/tcp                # BGPd vty
ospf6d          2606/tcp                # OSPF6d vty
```

Para este laboratorio se utilizara el protocolo de ruteo BGP. Por lo tanto, los demonios que se tienen que arrancar son el de demonio de Zebra y el del protocolo BGP. Para ello es ejecutan simplemente los siguientes comandos:

```
zebra &
bgpd &
```

Finalmente se debe de cerciorar que la maquina Linux este habilitada para el reenvío de paquetes IP, ya que si no es así, a pesar de haber instalado el software Zebra no será posible que la maquina actúe como un ruteador. Para habilitar el reenvío de paquetes IP basta con ejecutar la siguiente instrucción:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

2.1.2 Configuración

Normalmente la configuración de ruteo en sistemas del tipo UNIX se hace mediante comandos como `ifconfig`, `route`, `netstat` o `ip`. Con Zebra estos quedan de lado dando paso a un modo de configuración que trabaja al estilo de una sesión telnet a un ruteador Cisco. De hecho, Zebra utiliza comandos al estilo de Cisco para su configuración.

Para conectarse al servidor Zebra, se establece una sesión telnet a la dirección IP del sistema en donde está corriendo Zebra y el número de puerto correspondiente de los ya mencionados. Después de conectarse y autentificarse, queda disponible una interfaz en línea de comandos estándar de tipo Cisco. También es posible editar los archivos de configuración (por ejemplo: zebra.conf y bgpd.conf) estableciendo todas las líneas necesarias para la configuración de ruteo deseada previo al arranque de los demonios de Zebra. De esta manera Zebra iniciara con la configuración ya establecida sin necesidad de hacerlo por la línea de comandos.

Ahora de manera breve se expondrán los comandos para configurar un ruteador BGP Por

simplicidad en este ejemplo se modificaran los archivos de configuración mediante un editor de texto, por lo que en ningún momento se utiliza la interfaz de línea de comandos. El archivo zebra.conf quedaría de la siguiente manera:

```
hostname Edge2
password zebra
enable password zebra
!
interface lo
 description Edge2 loopback
!
interface eth0
 description wan
 ip address 172.17.50.1/24
!
interface eth1
 description lan
 ip address 20.0.0.1/24
! Static default route sample.
!
ip route 0.0.0.0/0 172.17.50.2
!
log file zebra.log
```

Una línea de comentarios va precedida de un signo de admiración (`!`). En la línea `hostname Edge2` se establece el nombre de host. Con las líneas `password zebra` y `enable password zebra` se establecen las contraseñas para acceso y modo de configuración privilegiado respectivamente. Con la instrucción `interface eth0` estamos indicando que las líneas que prosiguen establecen algún parámetro para la interfaz en cuestión, en este caso la interfaz eth0. Con la línea `ip address 172.17.50.1/24` se establece la dirección IP de la interfaz como 172.17.50.1 y con la máscara de subred 255.255.255.0. Con la línea `description lan` solo estamos indicando que esa interfaz da hacia una red de área local. Finalmente, con la línea `ip route 0.0.0.0/0 172.17.50.2` se establece una ruta estática indicando que cualquier tráfico seguirá por la dirección 172.17.50.2. Ahora se verá una muestra del archivo bgpd.conf:

```
! Zebra configuration saved from vty
!   2007/05/14 18:50:56
!
hostname Edge2bgpd
password zebra
!
router bgp 3
 bgp router-id 3.3.3.3
 network 20.0.0.0/24
 network 172.17.50.0/24
 neighbor 172.17.50.2 remote-as 2
!
log stdout
```

Una vez más en la línea `hostname Edge2bgpd` se establece el nombre del host a Edge2bgpd y en la línea `password zebra` se establece la contraseña de acceso. Con la línea `router bgp 3` indicamos que

este será el ruteador BGP Sistema Autónomo 3 y las líneas que prosiguen establecen parámetros del ruteador. Con la línea `bgp router-id 3.3.3.3` se asigna un identificador al ruteador BGP. Con las líneas `network 20.0.0.0/24` y `network 172.17.50.0/24` se dan a conocer las redes a las cuales el ruteador BGP está conectado y tiene acceso directo. Finalmente con la línea `neighbor 172.17.50.2 remote-as 2` queda indicado que el ruteador BGP tiene por vecino al Sistema Autónomo 2 con la dirección IP 172.17.50.2. El resto de los ruteadores que conforman la red se configuran de manera similar. Para verificar si el intercambio de tablas de ruteo se está efectuando, es necesario ejecutar la siguiente instrucción dentro de la interfaz de línea de comandos del demonio bgpd:

```
show ip bgp
```

Las terminales que estén conectadas a un ruteador deberán de tener establecida una dirección IP que corresponda con la subred del ruteador. También deberán establecer como puerta de enlace predeterminada la dirección IP del ruteador al cual se encuentran conectado. Esto se pudiera hacer mediante las siguientes instrucciones respectivamente:

```
ifconfig eth0 10.0.0.2 netmask 255.255.255.0 up
route add default gw 10.0.0.1
```

2.1.3 Topología de la red

La topología de red que se utiliza en este laboratorio es la siguiente. Existen tres maquinas habilitadas como ruteadores conectadas de forma contigua. Los dos ruteadores de los extremos actúan con nodos de ingreso y egreso a un dominio de red. El ruteador central actúa como nodo intermedio del dominio. Conectados a los extremos de la red se encuentran tres maquinas que la hacen de terminales. Una de ellas funge como transmisor de video. Otra más la hace de receptor de video. La tercera se encarga de emitir ruido, es decir, trata de congestionar la red mandando paquetes indiscriminadamente a través de ella. Las tarjetas de red utilizadas en las maquinas son adaptadores para Fast Etheret por lo que pueden soportar 100Mbps y 10Mbps según sean configuradas. La siguiente figura ilustra la topología de la red construida en el laboratorio.

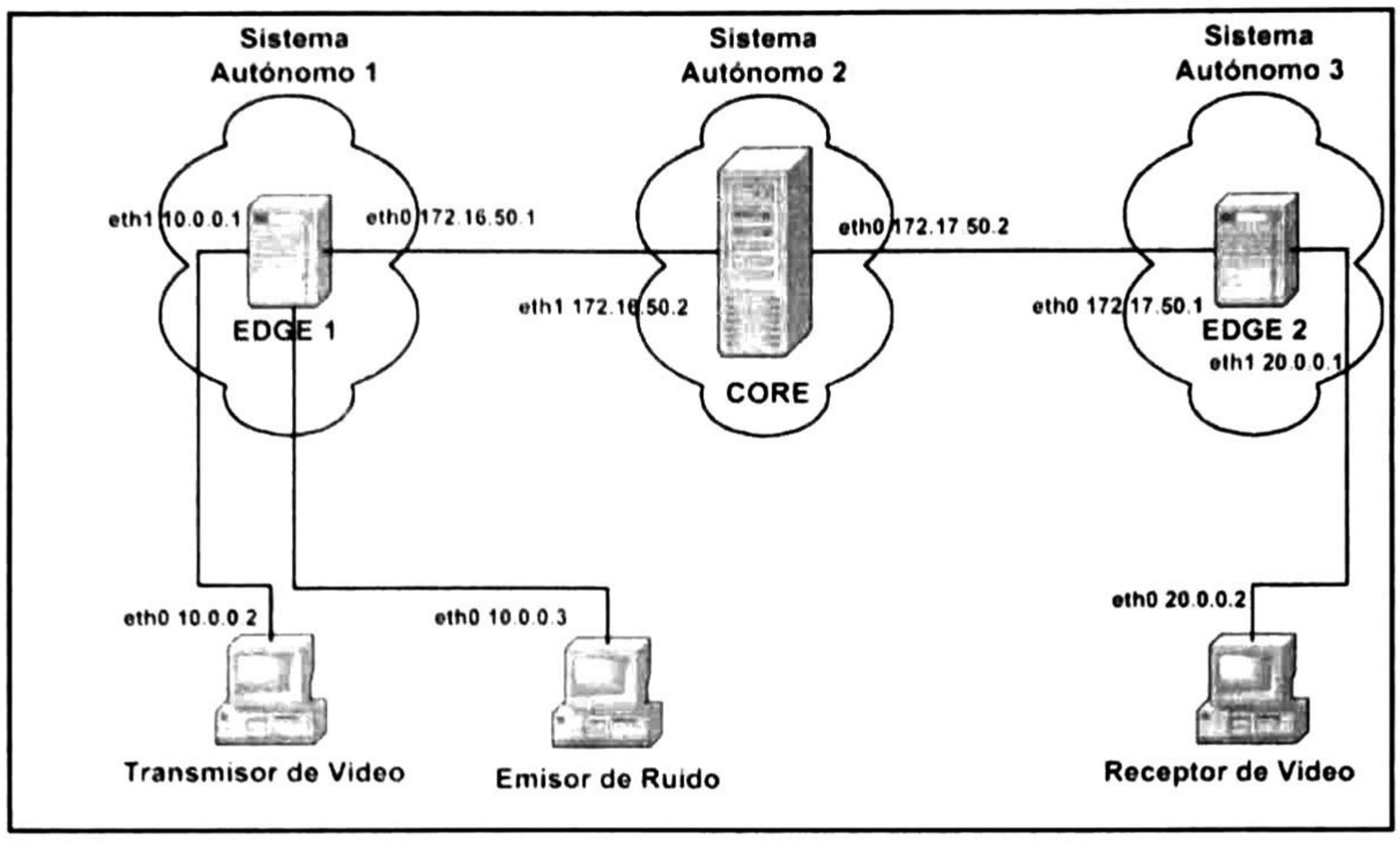


Figura 2.1 Topología de la red empleada en el laboratorio.

2.2 Habilitación de los mecanismos de Calidad de Servicio mediante Linux

Ahora que se tiene una red con la capacidad de operar protocolos de ruteo, es necesario habilitar los protocolos de Calidad de Servicio para poder proveer una variedad de servicios diferente a la de *best-effort*.

2.2.1 Servicios Diferenciados

Diffserv es el mecanismo de QoS más sencillo de proveer en una red Linux. Esto se debe a que se puede implementar utilizando el paquete Iproute2, el cual viene disponible en cualquier kernel de Linux a partir de la versión 2.2. También se puede encontrar en algunos kernels como Iproute solamente. Este paquete ofrece una interfaz que permite implementar una colección de herramientas de ruteo avanzado y gestión del tráfico de paquetes IP. Se pueden implementar túneles IP, tablas múltiples de ruteo, reserva de ancho de banda, etc. Las dos herramientas principales de Iproute son *ip* que permite la configuración de IPv4 e IPv6, como establecer direcciones etc., y *tc* que, como su nombre en inglés (*traffic control*) lo dice, permite control de tráfico proveyendo diferente ancho de banda mediante métodos para clasificar, priorizar, compartir y limitar el tráfico. El autor original de Iproute2 es Alexey Kuznetsov.

Una manera apropiada para configurar DiffServ con Iproute2 es adjuntar un marcador de

paquetes a una interfaz de salida, definir la estructura de la disciplina de encolamiento, numerar las clases que se van a utilizar e identificar que paquetes concuerdan con alguna clase para que de allí sigan algún comportamiento. La configuración de un dispositivo *edge,* es decir, un nodo de ingreso al dominio de DiffServ es la siguiente:

```
1. tc qdisc add dev eth0 handle 1:0 root dsmark índices 64
2. tc class change dev eth0 calssid 1:1 dsmark mask 0x3 value 0xb8
3. tc class change dev eth0 calssid 1:2 dsmark mask 0x3 value 0x68
4. tc class change dev eth0 calssid 1:3 dsmark mask 0x3 value 0x48
5. tc filter add dev eth0 parent 1:0 protocol ip prio 4 handle 1:u32 divisor 1
6. tc filter add dev eth0 parent 1:0 protocol ip prio 5 handle 2:u32 divisor 1
7. tc filter add dev eth0 parent 1:0 prio 4 u32 match ip dst 10.0.0.0/24 police rate 1Mbit burst 2k
continue flowid 1:1
8. tc filter add dev eth0 parent 1:0 prio 5 u32 match ip dst 10.0.0.0/24 flowid 1:2
9. tc filter add dev eth0 parent 1:0 prio 4 u32 match ip dst 10.1.0.0/16 match ip src 192.1.0.0/16
match ip protocol 6 0xff match ip dport 0x17 0xffff flowid 1:3
```

En la primera línea se adjunta un marcador DS al la interfaz eth0 en el nodo raíz. En la segunda línea se instruye al marcador DS que remarque el DSCP (*Differentiated Services Code Point*) de la clase 1:1 primeramente enmascarando los bits 6 y 7 y después haciendo una operación OR con el valor 0xb8. Hay que notar que esto equivale a ignorar los bits ECN (*Explicit Congestion Notification*) y establece el valor del código DS a 0x2e, el cual es el DSCP de EF De manera similar en la línea 3 se instruye al marcador DS para que remarque el código de la clase 1:2 con el valor 0x1a que corresponde al DSCP de AF31. La cuarta línea agrega un remarcado a la clase 1:3 con el código 0x12, el DSCP de AF21. Estas tres líneas también están registrando las clases 1:1, 1:2 y 1:3. En la línea 5 se agrega un clasificador u32 con prioridad de 4 al igual que en la línea 6 solo que este tiene una prioridad más baja, 5. En la línea 7 se hace un mapeo de todos los paquetes con destino a la subred 10.0.0.0/24 a la clase 1:1. Las líneas 7 y 8 muestran como adjuntar un medidor a un clasificador y su correspondiente acción en el caso de exceder la tasa de transmisión. Se definen dos filtros que correspondan a los mismos encabezados con más alta prioridad, pero uno de ellos se adjunta con un medidor y una política de acción. El operador *continue* se utiliza para permitir hacer una búsqueda de la siguiente prioridad inferior que corresponda con el filtro. En este caso, si se excede al medidor en la clase 1:1, el flujo es reclasificado a la clase 1:2. En la línea 9 se seleccionan todos los paquetes TCP provenientes de la subred 192.1.0.0/16 y cuyo destino es la subred 10.1.0.0/16 para que sean enviados a la disciplina de encolamiento de la clase 1:3. En resumen, todos los paquetes dirigidos hacia la subred 10.0.0.0/24 serán marcados con el DSCP 0x2e, que corresponde el comportamiento de EF, hasta el punto en donde comiencen a exceder su tasa de transmisión asignada, que es de 1Mbps con un tamaño de ráfaga de 2K. En tal caso, los paquetes son degradados a la clase 1:2 en donde son remarcados con el DSCP 0x1a que

corresponde a AF31. Finalmente, cualquier paquete TCP que dirigido hacia la subred 10.1.0.0/16 y que provenga de la subred 192.1.0.0/16 será remarcado con el DSCP 0x12, que corresponde a AF21.

La configuración de un dispositivo *core*, es decir, un nodo intermedio del domino DiffServ quedaría de la siguiente manera:

```
1. tc qdisc add dev eth0 handle 1:0 root dsmark indices 64 set_tc_index
2. tc filter add dev eth0 parent 1:0 protocol ip prio 1 tcindex mask 0xfc shift 2
3. tc qdisc add dev eth0 parent 1:0 handle 2:0 cbq bandwidth 10Mbit allot 1514 cell 8 avpkt 1000 mpu 64
4. tc class add dev eth0 parent 2:0 classid 2:1 cbq bandwidth 10Mbit rate 1500Kbit avpkt 1000 prio 1
bounded isolated allot 1514 weight 1 maxburst 10 defmap 1
5. tc qdisc add dev eth0 parent 2:1 pfifo limit 5
6. tc filter add dev eth0 parent 2:0 protocol ip prio 1 handle 0x2e tcindex classid 2:1 pass_on
7. tc class add dev eth0 parent 2:0 classid 2:2 cbq bandwidth 10Mbit rate 5Mbit avpkt 1000 prio 7 allot
1514 weight 1 maxburst 21 borrow
8. tc qdisc add dev eth0 parent 2:2 red limit 60KB min 15 KB max 45KB burst 20 avpkt 1000 bandwidth
10Mbit probability 0.4
9. tc filter add dev eth0 parent 2:0 protocol ip prio 2 handle 0 tcindex mask 0 classid 2:2 pass_on
```

En la línea 1 se adjunta a la interfaz eth0 del nodo raíz un marcador DS el cual copia el byte TOS (*Type of Service*) en tc_index. La línea 2 agrega un filtro al nodo raíz con el propósito de enmascarar los bits ECN y extraer el DSCP mediante un corrimiento de dos bits hacia la derecha. En la línea 3 una disciplina de encolamiento, en concreto CBQ (*Class Based Queue*), se adjunta al nodo 2:0 que es hijo del nodo raíz 1:0. En la línea 4 se define una clase a partir del nodo 2:0, la clase 2:1 que es de tipo CBQ y que esta acotada a una tasa de transmisión de 1.5 Mbits. En la línea 5 se adjunta una disciplina de encolamiento FIFO con tamaño máximo de 5 paquetes a la clase CBQ como esquema de manejo del buffer. En la línea 6 se agrega un clasificador que redireccionará todos los paquetes con DSCP 0x2e, es decir EF, a la clase 2:1, con el entendido de que los paquetes que no pertenezcan a EF podrán caer en el caso de que correspondan con otro filtro. En la línea 7 se define otra clase CBQ que también surge a partir del nodo 2:0, la clase 2:2 cuyo propósito es ser la clase BE (*best-effort*). La tasa de transmisión está limitada a 5 Mbps, aunque puede tomar prestado ancho de banda extra siempre y cuando no esté en uso, esto gracias al operador *borrow*. Como la clase EF no presta su ancho de banda, debido a que fue especificado en la línea 4 con el operador *isolated*, la clase BE solo puede tomar prestado como máximo un ancho de banda extra de 3.5 Mbps. En la línea 8 se adjunta una disciplina de encolamiento RED como esquema de manejo del buffer a ser usado por la clase BE. En la línea 9 se mapean el resto de los paquetes que no tienen DSCP 0x2e a la clase 2:2.

2.2.2 Servicios Integrados

Para la implementación de IntServ sobre Linux existe ya un software. Su nombre es RSVPd (*Resource ReSerVation Protocol daemon*) y fue desarrollado en el Instituto de Ciencias de la Información de la Universidad del Sur de California. RSVPd es un programa de tipo demonio que utiliza el protocolo de reservación de recursos RSVP para establecer reservaciones en ruteadores y hosts. Posee una API (*Application Programming Interface*) que permite a las aplicaciones hacer solicitudes de reservación. Incluye un modulo para adaptar los mecanismos de control de admisión y control de tráfico con los *drivers* de los dispositivos del kernel. Hay varias versiones públicas disponibles del código de RSVPd para sistemas como FreeBSD, SunOS, Solaris, IRIX y Linux. RSVPd se encuentra disponible en [11]. Sin embargo, la versión que existe para Linux es para kernel versión 2.2, la cual no es compatible con las versiones del kernel que se utiliza en los equipos del laboratorio que son 2.4 y 2.6

Existe otro proyecto que utiliza RSVP, su nombre es *RSVP-TE daemon for DiffServ over MPLS under Linux* y fue desarrollado por el grupo de investigación de redes de comunicaciones del Departamento de Tecnología de la Información de la Universidad de Ghent en Bélgica. Este se encuentra disponible en [12]. El principal objetivo de este proyecto es experimentar con Ingeniería de Tráfico utilizando a RSVP como protocolo de señalización para implementar DiffServ sobre MPLS en Linux. Para esto tomaron como base el código de la implementación de RSVPd de la Universidad de California y el código de MPLS-Linux de James R. Leu. Este código está hecho para kernel versión 2.4 y podría ser implantado en los equipos del laboratorio, pero su funcionalidad se encuentra fuera del alcance de esta tesis.

Por tales motivos se decidió utilizar como alternativa al mismo Iproute2 el cual permite la reservación de recursos, en este caso ancho de banda, lo cual es fundamental en IntServ y cubre la parte de control de tráfico. Sin embargo no se tiene un protocolo de reservación ni un control de admisión, por lo que estos se emularán de alguna manera mediante simple paso de mensajes entre los agentes de la plataforma y manteniendo el registro del estado de las reservaciones establecidas para determinar si se puede o no otorgar una nueva reservación.

No se puede dejar pasar el comentario de que implementar una versión compatible para kernel

2.4 o 2.6 de RSVPd a partir del código de la Universidad de California, sería un buen punto a considerar como trabajo futuro. De esta manera se tendría una implementación real de RSVP dentro del laboratorio de experimentación.

Para el control de tráfico dentro de Iproute2 se utilizan dos conceptos muy importantes que son los filtros que se encargan de clasificar los paquetes y depositarlos dentro de un buffer que implemente una disciplina de encolamiento, y las disciplinas de encolamiento que deciden que paquetes se envían primero que otros e incluso si es necesario descartarlos. Existen diferentes tipos de filtros, el más común es *u32*, que permite la selección del tráfico basándose en cualquier campo del encabezado del paquete.

Ahora se mostrara una configuración de ejemplo según el siguiente escenario de red. Se tiene un ruteador Linux con sus dos interfaces respectivas eth0 y eth1. La interfaz eth0 está conectada a una red de clientes y la interfaz eth1 es la conexión al *backbone* de Internet. Se procederá a limitar el ancho de banda de tal forma que se manejará una disciplina de encolamiento en la interfaz eth0 para decidir la velocidad con que se mandan datos provenientes de internet a la red de clientes. De manera similar, se manejará una disciplina de encolamiento en la interfaz eth1 para especificar la velocidad a la que la red de clientes puede mandar datos a internet. Con la disciplina de encolamiento CBQ se pueden crear clases e incluso subclases de usuarios por lo que se utilizara para la creación de dos de ellos:

- ISP, para proveer servicios de comunicación a los clientes, y
- Corp, para la red corporativa.

Se dispone de un ancho de banda de 10 Mbits que se dividirá en 8 Mbits para ISP y 2 Mbits para Corp.

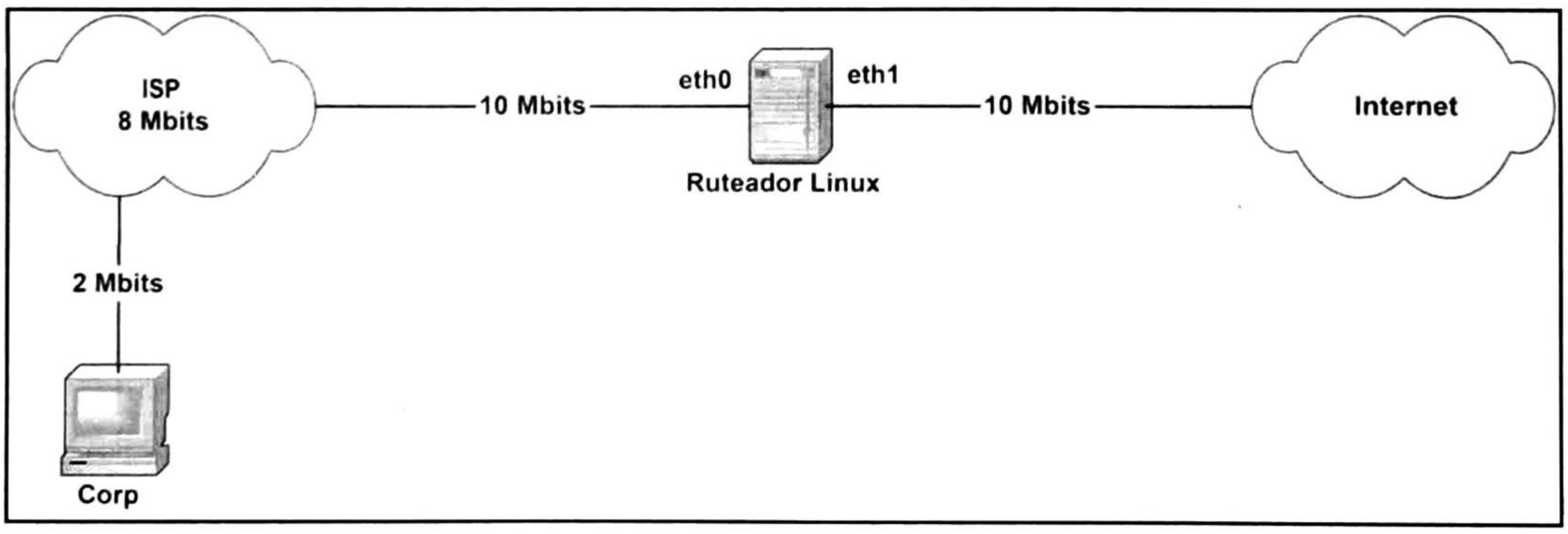


Figura 2.2 Escenario ejemplo para la configuración de reservación de ancho de banda con Iproute2.

El código de configuración es como sigue:

```
1. tc qdisc add dev eth0 root handle 10:0 cbq bandwidth 10Mbit avpkt 1000
2. tc class add dev eth0 parent 10:0 classid 10:1 cbq bandwidth 10Mbit rate 10Mbit allot 1514 weight
1Mbit prio 8 maxburst 20 avpkt 1000
3. tc class add dev eth0 parent 10:1 classid 10:100 cbq bandwidth 10Mb rate 8Mbit allot 1514 weight
800Kbit prio 5 maxburst 20 avpkt 1000 bounded
4. tc class add dev eth0 parent 10:1 classid 10:200 cbq bandwidth 10Mb rate 2Mbit allot 1514 weight
200Kbit prio 5 maxburst 20 avpkt 1000 bounded
5. tc qdisc add dev eth0 parent 10:100 sfq quantum 1514b perturb 15
6. tc qdisc add dev eth0 parent 10:200 sfq quantum 1514b perturb 15
7. tc filter add dev eth0 parent 10:0 protocol ip prio 100 u32 match ip src 150.151.23.24 flowid 10:200
8. tc filter add dev eth0 parent 10:0 protocol ip prio 25 u32 match ip src 150.151.0.0/16 flowid 10:100
```

En la línea 1 se define la regla para la disciplina de encolamiento de la interfaz eth0. Con el operador *root* se está indicando que esta es la disciplina raíz. Con el operador *handle* se indica el identificador de la regla. También se le está indicando al kernel que el ancho de banda disponible es de 10 Mbits y que el tamaño promedio de paquete es de 1000 Bytes. En la línea 2 se genera una clase hija con el identificador 10:1 y cuyo padre se indica con el operador *parent*. Se especifica también el MTU de 1514 Bytes. En la línea 3 se genera la clase para ISP con el identificador 10:100. Se le asigna 8 Mbits de ancho de banda y con el operador *bounded* se indica que esta acotado exclusivamente esa cantidad de ancho de banda, de otra forma podría absorber ancho de banda disponible de otras clases. De manera similar en la línea 4 se genera la clase Corp con el identificador 10:200 y con un ancho de banda de 2 Mbits el cual también esta acotado. En las líneas 5 y 6 se especifica que cada subclase se gestionará con la disciplina de encolamiento SFQ (*Stochstic Fair Queueing*). Existen más disciplinas de encolamiento como TBF (*Token Bucket Filter*), entre otras. Finalmente, en las líneas 7 y 8 se está indicando que paquetes pertenecen a cada clase. En el ejemplo se considera que Corp solo tiene una dirección IP para todo el tráfico. De manera similar se hace la configuración del tráfico de salida.

2.2.3 MPLS

La única herramienta conocida para implementar MPLS en una red Linux es MPLS-Linux. MPLS-Linux fue escrito por James R. Leu y consiste básicamente de un parche para el kernel de Linux que incorpora la capacidad de reenvío de paquetes conforme a MPLS permitiendo así construir redes por conmutación de etiquetas. El código de MPLS-Linux está hecho para kernel versión 2.6 y se encuentra disponible en [13]. Su instalación es muy sencilla ya que se encuentra disponible una serie de paquetes de tipo *rpm* para la distribución de Linux Fedora Core con lo cual se evita hacer la recompilación manual del kernel. La versión de MPLS-Linux que se utilizo en este laboratorio es la versión 1.955.

El escenario que servirá como ejemplo para una configuración de MPLS en Linux está conformado de la siguiente forma. Un LSR (*Label Switch Router*) como ruteador intermedio, y dos LER (*Label Edge Router*) como ruteadores de ingreso y egreso conectados entre sí formando un dominio de MPLS. Dos *hosts*, uno a cada extremo para llevar a cabo la transmisión de un punto a otro atravesando por el dominio de MPLS.

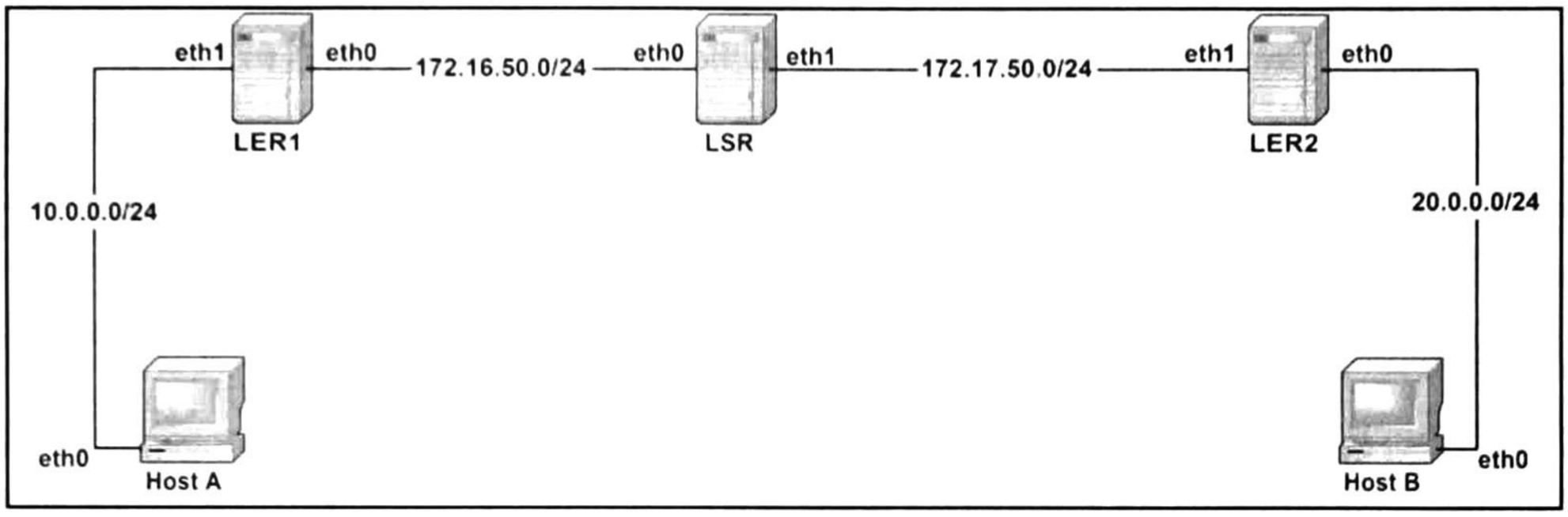


Figura 2.3 Escenario de ejemplo para la configuración de MPLS con MPLS-Linux.

La configuración de los equipos quedaría de la siguiente manera:

```
#Host A
1. ifconfig eth0 down
2. ifconfig eth2 10.0.0.2 netmask 255.255.255.0 up
3. ip route add 20.0.0.0/24 via 10.0.0.1 dev eth0
```

En el Host A se da de alta la interfaz de red y se establece la dirección de IP que esta tendrá (líneas 1y 2). Se establece una ruta que dirija el tráfico que va hacia el Host B por el LER1 (línea 3).

```
#Host B
1. ifconfig eth0 down
2. ifconfig eth2 20.0.0.2 netmask 255.255.255.0 up
3. ip route add 10.0.0.0/24 via 20.0.0.1 dev eth0
```

El Host B se configura de manera similar al Host A (líneas 1y 2), solo que el tráfico irá dirigido por el LER2 (línea 3).

```
#LER1
1. ifconfig eth0 down
2. ifconfig eth1 down
3. ifconfig eth0 172.16.50.1 netmask 255.255.255.0 up
4. ifconfig eth1 10.0.0.1 netmask 255.255.255.0 up

#Configuración MPLS de HostB -> HostA
5. mpls labelspace set dev eth0 labelspace 0
6. mpls ilm add label gen 2001 labelspace 0

#Configuración MPLS de HostA -> HostB
7. mpls nhlfe add key 0 instructions push gen 1000 nexthop eth0 ipv4 172.16.50.2 (regresa clave 0x2)
8. ip route add 20.0.0.0/24 via 172.16.50.2 mpls 0x2
```

De igual manera en el LER1 se dan de alta las interfaces de red y se establecen las direcciones de IP (líneas 1 a 4). Se comenzara por el tráfico que proviene del Host B hacia el Host A. Se establece un espacio de etiquetas del cual el ruteador pueda esperar paquetes por alguna interfaz (línea 5), en este caso esperan paquetes por la interfaz eth0. Enseguida se agrega el registro a la tabla ILM (*Incoming Label Map*) del valor con el que se espera alguna etiqueta (línea 6), en este caso se espera que arriben paquetes con la etiqueta 2001 y cundo lo hagan esta les será retirada. A continuación se hace la configuración para el tráfico del Host A al Host B. Se crea un registro NHLFE (*Next Hop Label Forwarding Entry*) que tendrá asociado la etiqueta con valor 1000 la cual se inserta en el paquete que se reenvía hacia su siguiente salto, en este caso hacia la dirección 172.16.50.2 por la interfaz eth0 que corresponden al LSR (línea 7). Esta instrucción arroja la clave del registro la cual se necesita en el siguiente comando. Finalmente, se hace el mapeo FEC (*Forwarding Equivalence Class*) a NHLFE usando la clave NHLFE recién creada (línea 8).

```
#LER2
1. ifconfig eth0 down
2. ifconfig eth1 down
3. ifconfig eth0 20.0.0.1 netmask 255.255.255.0 up
4. ifconfig eth1 172.17.50.1 netmask 255.255.255.0 up

#Configuración MPLS de HostB -> HostA
5. mpls nhlfe add key 0 instructions push gen 2000 nexthop eth1 ipv4 172.17.50.2 (regresa clave 0x3)
6. ip route add 10.0.0.0/24 via 172.17.50.2 mpls 0x3

#Configuración MPLS de HostA -> HostB
7. mpls labelspace set dev eth1 labelspace 0
8. mpls ilm add label gen 1001 labelspace 0
```

La configuración del LER2 es muy similar. Se dan de alta las interfaces de red y establecen las direcciones IP (líneas 1 a 4). A los paquetes del tráfico que va del Host B al Host A se les agrega la etiqueta con valor 2000 y se reenvían hacia el LSR (líneas 5 y 6). Con respecto al tráfico del Host A al Host B, se esperan paquetes con la etiqueta 1001 por la interfaz eth1 la cual les será retirada para posteriormente hacer la correspondiente búsqueda en las tablas de ruteo IP (líneas 7 y 8). Esto se debe a que el paquete abandonará el domino de MPLS y debe ser reenviado a su destino de manera convencional.

```
#LSR
1. ifconfig eth0 down
2. ifconfig eth1 down
3. ifconfig eth0 172.16.50.2 netmask 255.255.255.0 up
4. ifconfig eth1 172.17.50.2 netmask 255.255.255.0 up

#Configuración MPLS de HostA -> HostB
5. mpls labelspace set dev eth0 labelspace 0
6. mpls ilm add label gen 1000 labelspace 0
7. mpls nhlfe add key 0 instructions push gen 1001 nexthop eth1 ipv4 172.17.50.1 (regresa clave 0x2)
```

```
8. mpls xc add ilm_label gen 1000 ilm_labelspace 0 nhlfe_key 0x2

#Configuración MPLS de HostB -> HostA
9. mpls labelspace set dev eth1 labelspace 0
10. mpls ilm add label gen 2000 labelspace 0
11. mpls nhlfe add key 0 instructions push gen 2001 nexthop eth0 ipv4 172.16.50.1 (regresa clave 0x3)
12. mpls xc add ilm_label gen 2000 ilm_labelspace 0 nhlfe_key 0x3
```

También se dan de alta las interfaces de red y se establecen las direcciones IP (líneas 1 a 4). Debido a que es un ruteador intermedio y existen flujos de paquetes del Host A al Host B y viceversa, este espera paquetes por sus dos interfaces de red. Por la interfaz eth0 espera paquetes con la etiqueta 1000 (líneas 5 y 6) y por la interfaz eth1 espera paquetes con la etiqueta 2000 (líneas 9 y 10). Al igual que en los LER se genera una etiqueta, la 1001 con la cual se envían paquetes hacia el LER2 (línea 7) y la 2001 para los paquetes que se envían hacia el LER2 (línea 11). En seguida se hace el intercambio de etiquetas, sustituyendo la etiqueta 1000 por la 1001 (línea 8) y la 2000 por la 2001 (línea 12), esto utilizando la clave NHLFE recién creada. Finalmente la red MPLS queda configurada.

2.3 Dynamips / Dynagen

Otra alternativa para construir una red que incluya protocolos de Calidad de Servicio es el proyecto Dynamips [14] realizado en la UTC (*University of Technology of Compiegne, France*) [15] y cuyo autor es Christofer Fillot. Este proyecto tiene como propósito el emular ruteadores Cisco en PCs convencionales. Las versiones de ruteadores Cisco que soporta este emulador son las series 3600, 3700, 2600 y 7200. Dynamips corre imágenes reales de Cisco IOS. Dynagen [16] es una interfaz de usuario escrita en Python por Greg Anuzelli. Dynagen permite la simulación de un laboratorio de redes completo mediante la edición de un archivo de configuración inicial en el cual se establecen las configuraciones de los equipos de red emulados por Dynamips. También ofrece una interfaz en línea de comandos donde se puede listar las instancias de equipos que se están emulando, así como suspender y reiniciar dichas instancias, conectarse con las instancias de ruteadores, entre otras.

2.3.1 Cisco IOS

Cisco IOS es el software que usa la gran mayoría de equipos Cisco. IOS es un paquete con funciones de ruteo y conmutación de telecomunicaciones e interoperabilidad entre redes, integrado con un sistema operativo multitarea.

Cisco IOS cuenta con una interfaz de línea de comandos característica para la comunicación entre el equipo y el usuario. Esta interfaz provee un conjunto de comandos fijo, compuesto por múltiples palabras, para su configuración. El conjunto de comandos disponible depende del nivel de privilegio del usuario que se encuentra en sesión con el equipo, permitiendo así un cierto nivel de seguridad. Las imágenes Cisco IOS pueden ser descargadas desde el sitio web de Cisco [17].

2.3.2 Instalación

Dynagen funciona en cualquier plataforma con soporte para Python, entre ellas Windows y Linux. Existe un paquete de instalación para Windows, el cual está disponible en la sección de descargas del sitio web de Dynagen. Este paquete ya incluye Dynamips y una versión previamente compilada de Dynagen, por lo que se elimina la necesidad de instalar Python. También cuenta con integración al Explorador de Windows lo que permite correr los archivos *net* con tan solo un doble clic. Como prerrequisito para instalar Dynamips / Dynagen está la instalación de la librería WinPcap, la cual se puede descargar del sitio web de WinPcap [18]. Esta librería se utiliza para la captura de paquetes de la red y permite puentear interfaces virtuales de un ruteador emulado con las interfaces físicas del sistema, es decir, con las tarjetas de red de la PC. Para su instalación en Linux, se deben descargar los archivos de Dynamips / Dynagen de la sección de descargas del sitio web de Dynagen. Después se descomprimen en una ruta adecuada (ej. `/opt/dynamips/dynagen`) y se crean los enlaces a los archivos ejecutables Dynamips y Dynagen en `/usr/local/bin`, o se establecen en alguna parte de *PATH*. La siguiente tabla muestra línea por línea el proceso de instalación:

Instrucción en el shell de Linux	Descripción de la instrucción
`cd /opt/`	Cambiar al directorio /opt/
`mkdir dynamips`	Crear el directorio dynamips
`cd dynamips`	Cambiar al directorio dynamips
`ckdir images`	Crear el directorio images
`cp IOS_Images /opt/dynamips/images/`	Copiar las imágenes de IOS al directorio images
`cp dynagen.tar.gz /opt/dynamips`	Copiar el archivo dynagen.tar.gz al directorio dynamips
`tar -zxvf dynagen.tar.gz`	Descomprimir el archivo dynagen.tar.gz
`cp dynamips.bin /opt/dynamips/`	Copiar el archivo dynamips.bin al directorio dynamips
`ln -s /opt/dynamips/dynamips.bin /usr/bin/dynamips`	Crear el enlace al archivo ejecutable de Dynamips con el nombre dynamips en el directorio /usr/bin
`ln -s /opt/dynamips/dynagen/dynagen /usr/bin/dynagen`	Crear el enlace al archivo ejecutable de Dynagen con el nombre dynagen en el directorio /usr/bin
`dynamips -H 7200 &`	Emular un ruteador 7200 en modo Hypervisor
`cd dynagen/simple_labs/simple1`	Cambiar al directorio dynagen/simple_labs/simple1
`dynagen simple1.net`	Ejecutar dynagen con la configuración del archivo simple1.net

Tabla 2.1 Instrucciones para la instalación de Dynamip / Dynagen en Linux.

2.3.3 Configuración

Dynagen usa un archivo simple de configuración con la extensión *net* para establecer todas las configuraciones de los equipos de red y las interconexiones que habrá entre estos, todo ello para dar forma a un laboratorio de redes virtual. En este archivo se utiliza la sintaxis de un archivo *ini* en donde cualquier línea de comentario va precedida por el símbolo #. A continuación se muestra un ejemplo de archivo de configuración *net*:

```
# Simple lab
[localhost]

    [[7200]]
    image = \Program Files\Dynamips\images\c7200-jk9s-mz.123-7.T.bin
    # On Linux / Unix use forward slashes:
    # image = /opt/7200-images/c7200-jk9o3s-mz.124-7a.image
    npe = npe-400
    ram = 128

    [[ROUTER R1]]
    f0/0 = NIO_gen_eth:\Device\NPF_{568CD751-5F56-4DC7-9B84-9DC357B7CD95}
```

En la primera sección se especifica el *host* en el cual está corriendo Dynamips, encerrado por corchetes, ejemplo `[localhost]`. En la siguiente sección, encerrado por dobles corchetes, se indica la serie del ruteador, con la cual todas sus instancias de esa misma serie recibirán una misma configuración por defecto, ejemplo `[[7200]]`. Primero se especifica la configuración por defecto del ruteador y posteriormente se puede sobrescribir la configuración en la definición de cada instancia de ruteador en particular. La palabra clave *image* especifica la ruta del sistema en donde se localiza la imagen de IOS que se quiere utilizar, ejemplo `image = \Program Files\Dynamips\images\c7200-jk9s-mz.123-7.T.bin`. La palabra clave *npe,* del inglés *Network Processing Engine (NPE)*, indica el tipo de motor de proceso de la red, ejemplo `npe = npe-400` que es el valor por defecto. La palabra clave *ram* especifica la cantidad de memoria RAM del sistema que se reserva para uso de la instancia del ruteador virtual, ejemplo `ram = 128`. En las siguientes secciones se especifican los parámetros de configuración para cada instancia en particular. En doble corchete indicamos la instancia específica de ruteador que queremos configurar, ejemplo `[[ROUTER R1]]`. Se establece la interfaz de conexión, ejemplo `f0/0 = NIO_gen_eth:\Device\NPF_{568CE751-5FD6-4DC7-AB84-9DC35CB7CDEA}`. Esta línea indica que la interfaz virtual `f0/0` (FastEthernet0/0) de la instancia del ruteador está conectada a la interfaz física del adaptador de red cuyo descriptor aparece allí mismo. Los descriptores son consultados al sistema y proporcionados al usuario por la consola de Dynamips / Dynagen, esto para el caso de Windows. En sistemas Linux solo es necesario el nombre de la interfaz, por ejemplo eth0.

2.3.4 Desempeño

Para probar el desempeño de los emuladores de ruteador Cisco Dynamips que componen la red se realizaron algunos experimentos. La topología de la red es la misma que se expuso en la sección 2.1.3. Con la ayuda del generador de tráfico Mgen se lanza un flujo de tráfico constante desde el transmisor y con el *sniffer* Ethereal y tcpdump se capturan los paquetes del flujo del lado del receptor. A partir de las mediciones tomadas se logró obtener varias estadísticas. Cada experimento se corrió durante 60 segundos. En cada experimento varía tanto el tamaño de paquete como la frecuencia con la que estos se transmiten. También se calcula la tasa de transmisión que el flujo transmitido debería alcanzar. De esta manera se podrá observar bajo qué condiciones los ruteadores Dynamips cumplen con un buen desempeño y bajo cuales no, para finalmente obtener así la tasa de transferencia máxima con la que se puede operar en la red y tener una buena comparativa de desempeño. A continuación se muestra el análisis de los experimentos realizados. En las tablas se encuentran los datos con que se corrieron los experimentos más significativos y les acompaña una gráfica que muestra el comportamiento del flujo de paquetes durante la transmisión.

Numero del Experimento	*1*
Paquetes por segundo enviados por Mgen	**75 pps**
Tamaño en bytes por paquete enviados por Mgen	**470 Bytes**
Tamaño total del paquete (encabezado incluido)	**512 Bytes**
Ancho de banda calculado en kilobits por segundo (sin encabezado)	**282 kbps**
Ancho de banda calculado en kilobits por segundo (con encabezado)	**307.2 kbps**
Tiempo de duración de la transmisión en segundos	**60 seg**
Cantidad de paquetes se calcula envía Mgen durante el tiempo de transmisión	**4500 paquetes**
Cantidad real de paquetes enviados capturados con Ethereal	**4501 paquetes**
Cantidad real de paquetes recibidos capturados con tcpdump	**4501 paquetes**

Tabla 2.2 Experimento 1 para el desempeño de Dynamips/ Dynagen.

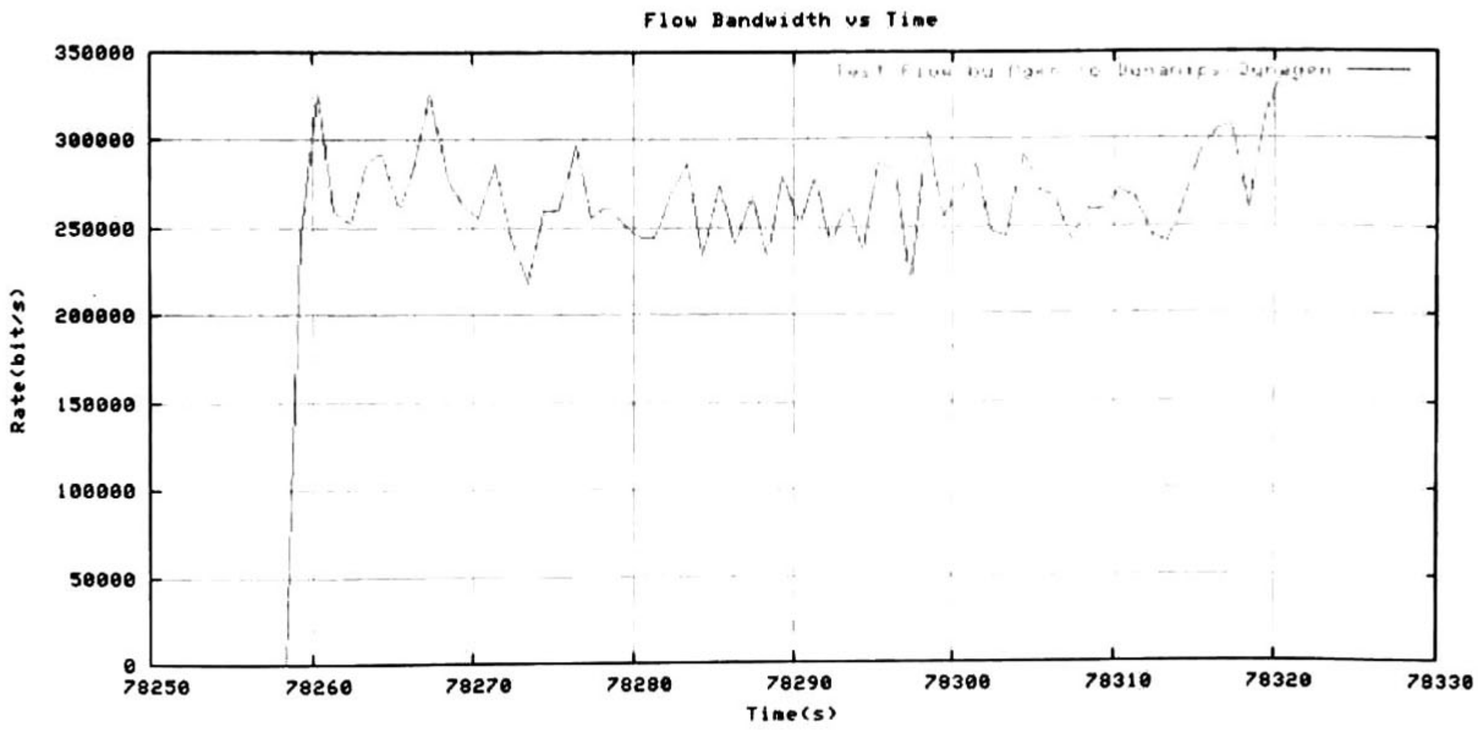


Figura 2.4 Tasa de transmisión del flujo de paquetes obtenida desde el receptor en el experimento 1.

Numero del Experimento	*3*
Paquetes por segundo enviados por Mgen	**75 pps**
Tamaño en bytes por paquete enviados por Mgen	**471 Bytes**
Tamaño total del paquete (encabezado incluido)	**513 Bytes**
Ancho de banda calculado en kilobits por segundo (sin encabezado)	**282.6 kbps**
Ancho de banda calculado en kilobits por segundo (con encabezado)	**307.8 kbps**
Tiempo de duración de la transmisión en segundos	**60 seg**
Cantidad de paquetes se calcula envía Mgen durante el tiempo de transmisión	**4500 paquetes**
Cantidad real de paquetes enviados capturados con Ethereal	**4501 paquetes**
Cantidad real de paquetes recibidos capturados con tcpdump	**941 paquetes**

Tabla 2.3 Experimento 3 para el desempeño de Dynamips/ Dynagen.

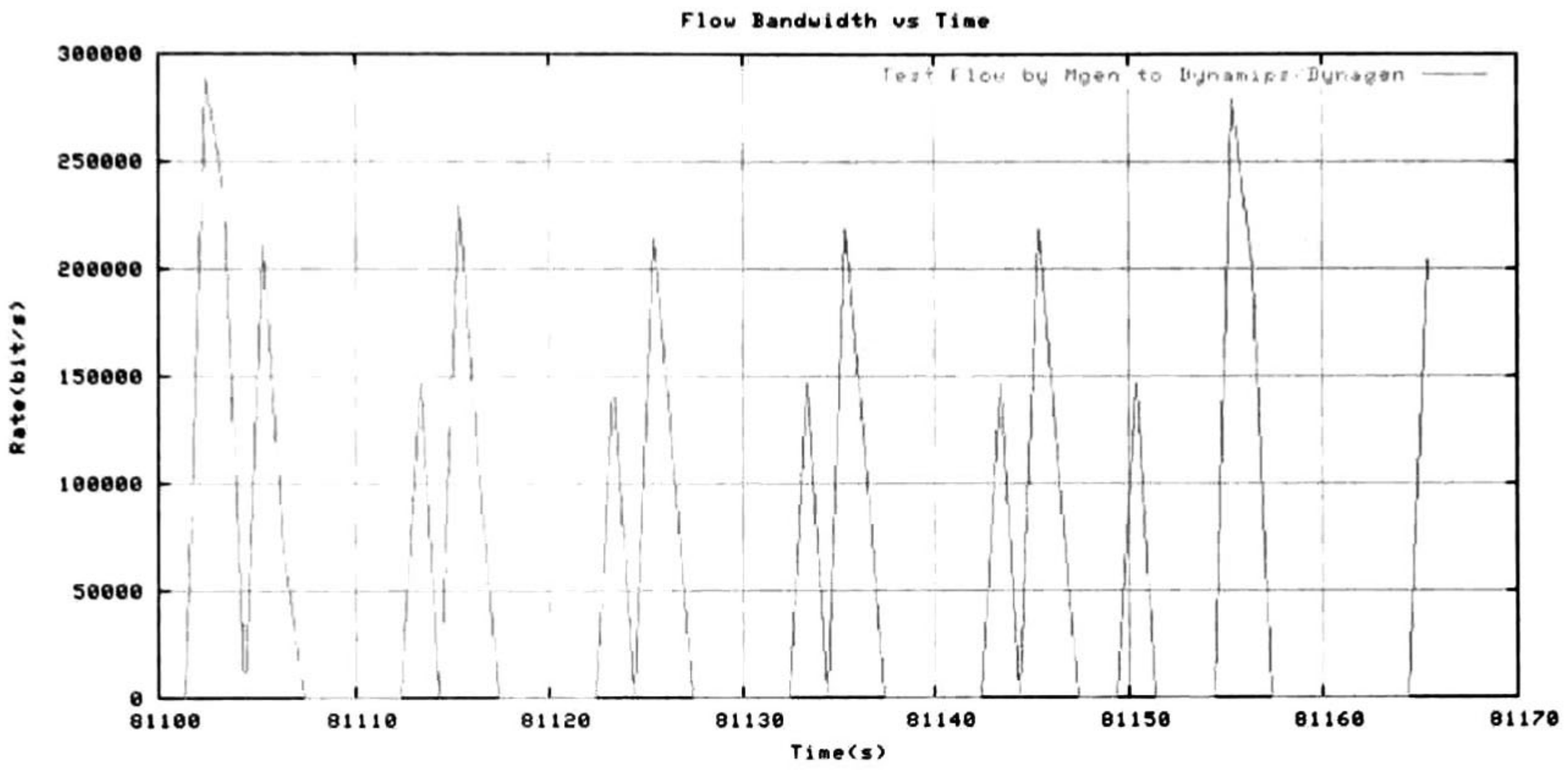


Figura 2.5 Tasa de transmisión del flujo de paquetes obtenida desde el receptor en el experimento 3.

De los experimentos se puede determinar que el máximo rendimiento de la red se obtiene con las características de experimento 1, es decir, cuando se transmiten paquetes de 512 Bytes con una frecuencia de 75 paquetes por segundo, lo que da una tasa de transmisión promedio de 307 kbps. Esto se puede concluir ya que se recibió en su totalidad la cantidad de paquetes generados en la transmisión, que fueron 4501 paquetes. En el experimento 2 se conserva el tamaño de paquete de 512 Bytes, pero se incrementa levemente la frecuencia de envío a 80 paquetes por segundo. Debido a esto se ve afectado el desempeño de la red ya que no se recibieron en su totalidad la cantidad de paquetes generados. En el experimento 3 se observa una pérdida de paquetes contundente cuando se conserva la frecuencia de envío de paquetes del experimento 1 (75 paquetes por segundo), pero se incrementa el tamaño de paquete tan solo en un byte, es decir, el tamaño ahora es de 513 Bytes. En los experimentos 4 y 5 se incrementa la frecuencia de envío a 100 paquetes por segundo, pero el tamaño de paquete es de 100 Bytes y 30 Bytes respectivamente. Aquí también se consigue una pérdida de paquetes significativa, por lo tanto, por más pequeño sea el tamaño de paquete si se excede levemente de los 75 paquetes por segundo, no será posible obtener un buen desempeño de la red. Finalmente en los experimentos 6 y 7 se

observa que aun si se sobrepasa el tamaño de paquete de 512 Bytes tal y como se hizo en el experimento 3, pero se tiene una frecuencia de envío de paquetes por segundo pequeña, se obtiene un comportamiento constante en la tasa de transmisión debido a que no existe perdida de paquetes. Sin embargo la tasa de transmisión que se puede alcanzar resulta ser demasiado pequeña. Es importante señalar que la tasa de transmisión máxima experimentada fue de 307 kbps con las características de tamaño de paquete y frecuencia de envío ya mencionadas, no cumple con las expectativas de desempeño que se tenían para la red experimental.

El desempeño de la red fue probado también con la herramienta llamada Pathload en su versión 1.3.2. Esta herramienta se utiliza para estimar el ancho de banda disponible de una ruta punto a punto desde un host emisor a un host receptor. El ancho de banda disponible es el *throughput*[2] máximo en la capa de red que un flujo puede obtener en la ruta del emisor al receptor, sin reducir la tasa de transmisión del resto del tráfico sobre la ruta. La descripción del algoritmo de estimación que utiliza Pathload se encuentra en [19] y un estudio más exhaustivo sobre ancho de banda se encuentra en [20].

Los resultados que esta herramienta arroja son contundentes. La prueba hecha en la red compuesta por emuladores Dynamips da un ancho de banda disponible de 0.09 Megabits por segundo. A continuación se muestra un extracto del archivo de resultados generado por Pathload:

```
Receiver host starts measurements at sender 20.0.0.2 on Thu Jun 14 13:10:04 2007

        *****  RESULT *****
User specified bandwidth resolution achieved
Available bandwidth range : 0.09   0.09 (Mbps)
Measurements finished at Thu Jun 14 13:12:25 2007
Measurement latency is 141.64 sec
```

Posteriormente se hizo una prueba a la red compuesta por ruteadores Zebra en donde el ancho de banda disponible fue de 95.30 a 96.80 Megabits por segundo.

```
Receiver host starts measurements at sender 20.0.0.2 on Wed Jun 13 19:55:06 2007

        *****  RESULT *****
User specified bandwidth resolution achieved
Available bandwidth range   95.30 - 96.80 (Mbps)
Measurements finished at Wed Jun 13 19:55:13 2007
Measurement latency is 6.77 sec
```

También a esta misma red se le hizo una prueba pero con los adaptadores de red configurados a 10Mbps. El ancho de banda disponible resulto de 8.56 a 9.17 Megabits por segundo.

```
Receiver host starts measurements at sender 20.0.0.2 on Wed Jun 13 10:01:21 2007

        *****  RESULT *****
```

---

[2] Cantidad de información digital por unidad de tiempo que es llevada por un enlace físico o lógico, o que pasa a través de un nodo de red. Generalmente se mide en bits por segundo.

```
Exiting due to grey bw resolution
Available bandwidth range : 8.56   9.17 (Mbps)
Measurements finished at Wed Jun 13 10:01:43 2007
Measurement latency is 22.25 sec
```

El desempeño tan raquítico que se observa en Dynamips es congruente. Ya que, aunque se puede construir una red con maquinas emulando IOS, sin ningún problema de conectividad ni de funcionamiento de los protocolos, una PC jamás podrá igualar el poder de computo que ofrece el hardware con el que está construido un ruteador Cisco autentico. Además, así se justifica también el propósito que tiene el emulador, el cual es el entrenamiento con el software IOS y el probar operatividad en configuraciones experimentales antes de ser implantadas en redes reales.

Aun sabiendo esto, se esperaba que el rendimiento de los emuladores fuera lo suficientemente poderoso para llevar a cabo los experimentos de que son propósito esta tesis, pero no fue así. Por lo tanto, a pesar de ser una opción muy interesante para conformar un laboratorio de experimentación de redes con los protocolos de Calidad de Servicio provistos por Cisco IOS, no queda más alternativa que implantar las soluciones que provee Linux para QoS.

En cuanto a simulación, una herramienta que resulta muy interesante es Packet Tracer la cual es desarrollada por Cisco Systems. Este software propietario ofrece un ambiente de simulación para poner a prueba nuevos diseños y configuraciones de red, con la disponibilidad de una gran variedad de dispositivos. Como ya se menciono, su propósito es la simulación y por lo tanto no sería posible construir una red con los requerimientos necesarios para la experimentación a partir de esta herramienta.

2.4 Conclusiones

En este capítulo se dieron a conocer diferentes herramientas ya existentes para la implementación de una red experimental que cuente con los mecanismos de Calidad de Servicio habilitados. Se partió desde los protocolos de ruteo básico como BGP, entre otros, los cuales pueden ser implementados con el software de ruteo Zebra. Se dio a conocer el paquete Iproute2 con el cual se pueden implementar estrategias de ruteo avanzado, desde marcado de paquetes (*DiffServ*) hasta reserva de ancho de banda (*IntServ*), etc. Posteriormente se mostro MPLS-Linux, una herramienta con la cual se puede implementar ruteo por conmutación de etiquetas (*MPLS*). Con estos tres mecanismos

habilitados se cumplen los requerimientos de la red experimental para el laboratorio. También se muestran otras soluciones muy interesantes para la construcción de la red las cuales fueron consideradas y analizadas durante la elaboración de este trabajo. Entre ellas está el proyecto Dynamips / Dynagen, el cual emula ruteadores Cisco en una PC convencional con imágenes reales del software IOS y que desafortunadamente no cumplió con el rendimiento necesario de transmisión para los experimentos que se proponen en esta tesis. Otro proyecto interesante fue RSVPd, un demonio que implementa IntServ con RSVP pero no fue compatible con las versiones del kernel de Linux que se utilizan en los equipos del laboratorio. Otro proyecto interesante fue *RSVP-TE daemon for DiffServ over MPLS under Linux*, el cual consiste en DiffServ sobre MPLS utilizando RSVP como protocolo de señalización para llevar a cabo experimentos de Ingeniería de Tráfico. A pesar de implementar RSVP basándose en el código del proyecto RSVPd y MPLS basándose en el código de MPLS-Linux, la funcionalidad de este proyecto está fuera del alcance de la tesis y solo se podría considerar como propuesta para incluirse en algún trabajo futuro. Finalmente se hace mención del software propietario Packet Tracer versión 4.1 hecho por Cisco Systems el cual solo puede fungir como simulador de redes.

CAPITULO 3

3 Ontología para el marco de trabajo QoE

Los agentes necesitan una representación formal del conocimiento y esta puede ser modelada mediante una ontología. Dado que en esta tesis se utiliza el enfoque de agentes para QoS Management se propone una ontología para el marco de trabajo QoE.

3.1 Ontologías para QoS

Una plataforma basada en agentes para el marco de trabajo QoE se propuso en [21]. La plataforma cuenta con una ontología para dar soporte al marco de trabajo. Esta ontología se creó usando el editor de ontologías Protege [22].

De acuerdo con [23] en ingeniería de software una ontología puede ser definida como *"la especificación de una conceptualización"* En otras palabras una ontología es la representación de objetos, conceptos, otras entidades y sus relaciones. En [24] se define como *"un conocimiento estándar compartido que define conceptos primitivos, relaciones, reglas, y sus instancias, los cuales pueden ser usados para capturar, estructurar y detallar de manera explícita y tacita tópicos de conocimiento a través de personas, organizaciones y sistemas de computo y software"*

Varias ontologías para QoS se han sido definidas, cada una con sus propias características. Para [25] una ontología para QoS debe de permitir decidir que mecanismo de QoS es mejor para satisfacer las necesidades del usuario, realizar monitorio de la QoS y detección de violaciones en los SLA's; y llevar a cabo adaptación de QoS.

En [26] se considera que las ontologías son requeridas para métricas de Calidad de Servicio, unidades de medición, unidades monetarias, propiedades medibles y métodos de medición. Como se acaba de mencionar, varias ontologías para QoS se han definido. Algunas de ellas son descritas brevemente y analizadas en [27] y [28].

DAMLQoS [29] es una ontología para describir servicios web. Está compuesta por tres capas: la capa de perfil QoS, cuyo propósito es realizar correspondencias de QoS; la capa de definición de propiedades QoS, la cual involucra la elaboración de dominios de propiedades y rangos de restricciones; y una capa para la definición de métricas y mediciones. En esta ontología las métricas se emplean sin importar las características de lo que estas miden. Como conceptos esta soporta *advertencias* y *solicitudes*. Debido a las restricciones de cardinalidad para expresar límites en propiedades QoS, el número de valores que una propiedad puede tener está restringido. El valor de los límites solo puede ser expresado en enteros positivos.

La ontología para QoS de FIPA [30] está diseñada para agentes FIPA y provee de un vocabulario básico para QoS. Esta ontología provee de dos métodos para obtener información de QoS: una *solicitud* simple, y una *suscripción*. Esta ontología está orientada hacia los aspectos de la red y es útil para la comunicación de agentes en escenarios de red. Debido a que está orientada solo a los estándares de FIPA, la implementación de esta ontología en otros contextos no podría ser adecuada.

La ontología WS QoS [31] está diseñada para el descubrimiento de servicios web con soporte QoS. Esta consiste de tres elementos: QoSInfo, WSQoSOntology , y QoSOfferDefinition. Al primer elemento le concierne el rendimiento del servidor y los protocolos de QoS. Los parámetros de QoS y las referencias de protocolos son soportados por el segundo elemento. El tercer elemento es una entidad de más alta jerarquía la cual se compone de uno o muchos elementos del tipo QoSInfo. Esta ontología tiene el propósito de selección de servicios web y las decisiones son tomadas considerando los requerimientos de la red o del servicio.

En [32] se propone QoSOnt. Esta ontología la forman también tres capas. A la capa inferior le corresponden los conceptos de QoS y las unidades de la ontología. En la capa media se cubren los atributos de QoS y sus métricas. La capa superior permite la interacción de las otras capas de la aplicación misma.

Otra ontología es la ontología QoS de [33] cuyo propósito es la selección dinámica de servicios web. Mediante esta ontología los agentes realizan la correspondencia entre los niveles de calidad disponible y las solicitudes de QoS. La ontología QoS se divide en: ontología superior la cual involucra

conceptos generales asociados a una calidad. La ontología media que se concentra en características de calidad de sistemas distribuidos, y la ontología inferior de la cual no se dan detalles a conocer.

La ontología QoE de [21] fue desarrollada para el marco de trabajo QoE. Esta ontología tiene la intención de soportar aspectos del usuario, la aplicación y QoS de la red que son considerados en el marco de trabajo QoE. También especifica el vocabulario y mensajes de comunicación de los agentes. La ontología consiste de conceptos QoS y acciones que realizan los agentes. Las acciones que se especifican son para la configuración de paquetización y depaquetización, la red y solicitudes de transmisión y recepción. Los conceptos se derivan de ahí para darle soporte a estas acciones. Cada concepto consiste de una serie de atributos. La ontología solo tiene dos performativos: *Consulta*, e *Informa*.

3.2 Ontología Extendida para QoE

Como se menciono, varias ontologías para QoS han sido propuestas. Todas ellas varían en cuanto a su propósito, arquitectura como número de capas o semántica, forma de representación abstracta que va desde lógica de primer orden hasta XML, y demás conceptos de QoS. A pesar de esto, la única ontología que considera los aspectos del usuario, la aplicación y QoS de la red es la ontología QoE. En otras palabras, a esta le conciernen ambas, QoE y QoS. También se puede observar que en lugar de excluirse, se pueden complementar una a la otra en el contexto de gestión de QoS y QoE. Basados en este análisis, se cree pertinente que la ontología QoE puede ser mejorada y complementada, por lo cual se le hace una serie de modificaciones. Estas modificaciones se derivan del principio de que las ontologías se requieren para métricas de Calidad de Servicio, unidades de medición, unidades monetarias, propiedades medibles y métodos de medición. En el caso de unidades medición, el tiempo y el espacio deben de ser considerados sin duda por la ontología QoE. A pesar de que fueron definidas métricas de desempeño como *retraso*, *ancho de banda* y *tamaño de buffer*, se requieren los conceptos para las unidades con las cuales estos se representan. De aquí que se incluyen los conceptos de *segundo*, *bit* y *byte*. En el marco de trabajo QoE el costo monetario se considera relevante para la QoE que está siendo medida. Por tal motivo, una unidad monetaria para representarlo es necesaria. A este concepto se le nombro *local_currency*. También se han propuesto otros conceptos asociados con las propiedades medibles de los protocolos de QoS que se proveen. Estos conceptos se muestran en la tabla

3.2. Entre estos, se incluyen para Servicios Diferenciados *Token Bucket*, para Servicios Integrados con RSVP *Flow Spec*, *Traffic Spec* y *Reservation Spec* y para MPLS *FEC, Label, LSP, LSR*. En términos de la retroalimentación subjetiva del usuario final se proponen dos nuevos conceptos: *user reduce request* y *user improve request*. También se proponen conceptos adicionales para la codificación de video y Gestión de QoS, entre ellos *SLA, network fault* y *application error.* También se han propuesto métodos nuevos para soportar acciones adicionales de los agentes. Para esquemas de codificación y gestión de la red se incluyen *network status query*, *poll performance metrics* y *coding configuration request*. La retroalimentación del usuario final se monitorea mediante la acción *monitor end user request*. La nueva y completa ontología QoE, acciones y conceptos, se muestra en las tablas 3.1 y 3.2 respectivamente.

Se cree que con las acciones y conceptos que han sido incluidos la ontología QoE cumple con los requerimientos de decidir que mecanismo de Calidad de Servicio es mejor para satisfacer las necesidades del usuario, realizar monitoreo de la QoS y detección de violaciones en los SLA's, y llevar a cabo adaptación de QoS.

Nombre de la acción del agente	*Descripción*
TxReq	Solicitud de transmisión
RxReq	Solicitud de recepción
NetConfReq	Solicitud de configuración de la red
NetDelConfReq	Solicitud de eliminación de la configuración de la red
NetStatusQuery	Consulta del estado de la red
PktConfReq	Solicitud de configuración del paquetizador
DepktConfReq	Solicitud de configuración del depaquetizador
PollPerformanceMetrics	Recolectar métricas de desempeño
MonitorEndUserReq	Monitorear solicitudes del usuario
CodecConfigReq	Solicitud de configuración de codificación

Tabla 3.1 Acciones de la Ontología QoE.

Nombre del Concepto	*Descripción*
Arbitrator	El agente arbitrador
Network	El ambiente de la red
Application	La aplicación
Packetiser	El paquetizador
Depacketiser	El depaquetizador
QoS	Calidad de Servicio
QoS_DiffServ	Mecanismo de Calidad de Servicio Servicios Diferenciados
QoS_DiffServ_AF	Servicios Diferenciados Servicio de Avance Asegurado
QoS_Diffserv_EF	Servicios Diferenciados Servicio de Avance Expedito
QoS_DiffServ_Token_Bucket	El medidor token bucket
QoS_DiffServ_Bucket_Rate	La tasa para el medidor token bucket
QoS_DiffServ_Bucket_Depth	La profundidad del medidor token bucket
QoS_IntServ	Calidad de Servicio Servicios Integrados
QoS_IntServ_CL	Servicios Integrados Servicio de Carga Controlada
QoS_IntServ_GS	Servicios Integrados Servicio Garantizado
QoS_IntServ_TSpec	Servicios Integrados Especificación de Tráfico
QoS_IntServ_RSpec	Servicios Integrados Especificación de Reservación
QoS_IntServ_FlowSpec	Servicios Integrados Especificación de Flujo

QoS_IntServ_FilterSpec	Servicios Integrados Especificación de Filtrado
QoS_IntServ_Token_Bucket_TSpec	Servicios Integrados Especificación de Tráfico del Token Bucket
QoS_IntServ_TBTSpec_Rate	Especificación de tasa de tráfico del Token Bucket
QoS_IntServ_TBTSpec_Depth	Especificación de profundidad de tráfico del Token Bucket
QoS_IntServ_TBTSpec_Peak_Rate	Especificación de tasa pico de tráfico del Token Bucket
QoS_IntServ_TBTSpec_Min_Size	Especificación del tamaño mínimo de paquete del Token Bucket
QoS_IntServ_TBTSpec_Max_Size	Especificación del tamaño máximo de paquete del Token Bucket
QoS_IntServ_TBRSpec_Rate	Especificación de tasa de reservación del Token Bucket Object
QoS_IntServ_TBRpec_ST	Especificación del término holgado del Token Bucket Object
QoS_MPLS	Multiprotocolo de conmutación por etiquetas
QoS_MPLS_FEC	Clase de equivalencia de avance de MPLS
QoS_MPLS_Label	Etiqueta MPLS
QoS_MPLS_LSR	Ruteador por conmutación de etiquetas de MPLS
QoS_MPLS_LSP	Ruta por conmutación de etiquetas MPLS
QoS_Class	Clase de servicio
QoS_Queue	Cola
QoS_Filter	Filtro de paquetes
QoE	Calidad de Experiencia
SLA	Acuerdo de Nivel de Servicio
BE	Servicio Best Effort
Bandwidth	Ancho de banda
Link	Enlace de la red
Packet	Paquete de red
Application_Error	Error (falla en la aplicación)
Network_Fault	Falta de la red (falla en algún elemento de red)
Router	Ruteador
Host	Host
Video	Video
Video_Horizontal_Size	Tamaño horizontal de una imagen de video
Video_Vertical_Size	Tamaño vertical de una imagen de video
Video_Codec	Codificación del flujo de video
Video_Data_Rate	Tasa de datos de un flujo de datos
Video_chromaticity	Calidad de color en un flujo de video
Audio	Flujo de audio
Audio_Codec	Codificación del flujo de audio
Still	Imagen estática
Text	Texto en un flujo multimedia
Media	Flujo multimedia
Net_Int	Interfaz de red
IP_Config	Configuración IP en la interfaz de red
Trans_Prot	Protocolo de transporte de red (por ejemplo BGP)
User	El usuario final
User_Improve_Req	Solicitud del usuario para mejorar la calidad de la transmisión
User_Reduce_Req	Solicitud del usuario para reducir la calidad de la transmisión
NetConfProfile	Perfil de configuración de la red
Problem	Cualquier problema no considerado
Message	Mensaje de comunicación del agente
Unit_Second	Unidad de medición para el tiempo
Unit_Bit	Unidad de medición de espacio en almacenamiento y volumen de transmisión de datos
Unit_Byte	Unidad de medición de espacio de almacenamiento y volumen de transmisión de datos
Unit_Local_Currency	Unidad de medición para valores monetarios
Metric	Métrica de desempeño
Metric_Net	Métrica de desempeño de la red
NQoS	La métrica de Calidad de Servicio de la Red
Delay	La métrica de desempeño retraso
Packet_Loss	La métrica de desempeño perdida de paquetes
Jitter	La métrica de desempeño jitter
Metric_Net_Utilisation	Métrica de utilización de la red
Metric_App	Métrica de la aplicación
Metric_App_AQoS	La métrica de Calidad de Servicio de la Aplicación
Metric_App_PSRN	Métrica PSNR
Weighting_Factor	Los factores ponderados del marco de trabajo QoE

Tabla 3.2 Conceptos de la Ontología QoE.

Cabe mencionar que la nueva ontología extendida para el marco de trabajo QoE también fue desarrollada utilizando el editor de ontologías protege en su versión 2.1.2 junto con el *Ontology Bean Generator* para Jade 3.1.

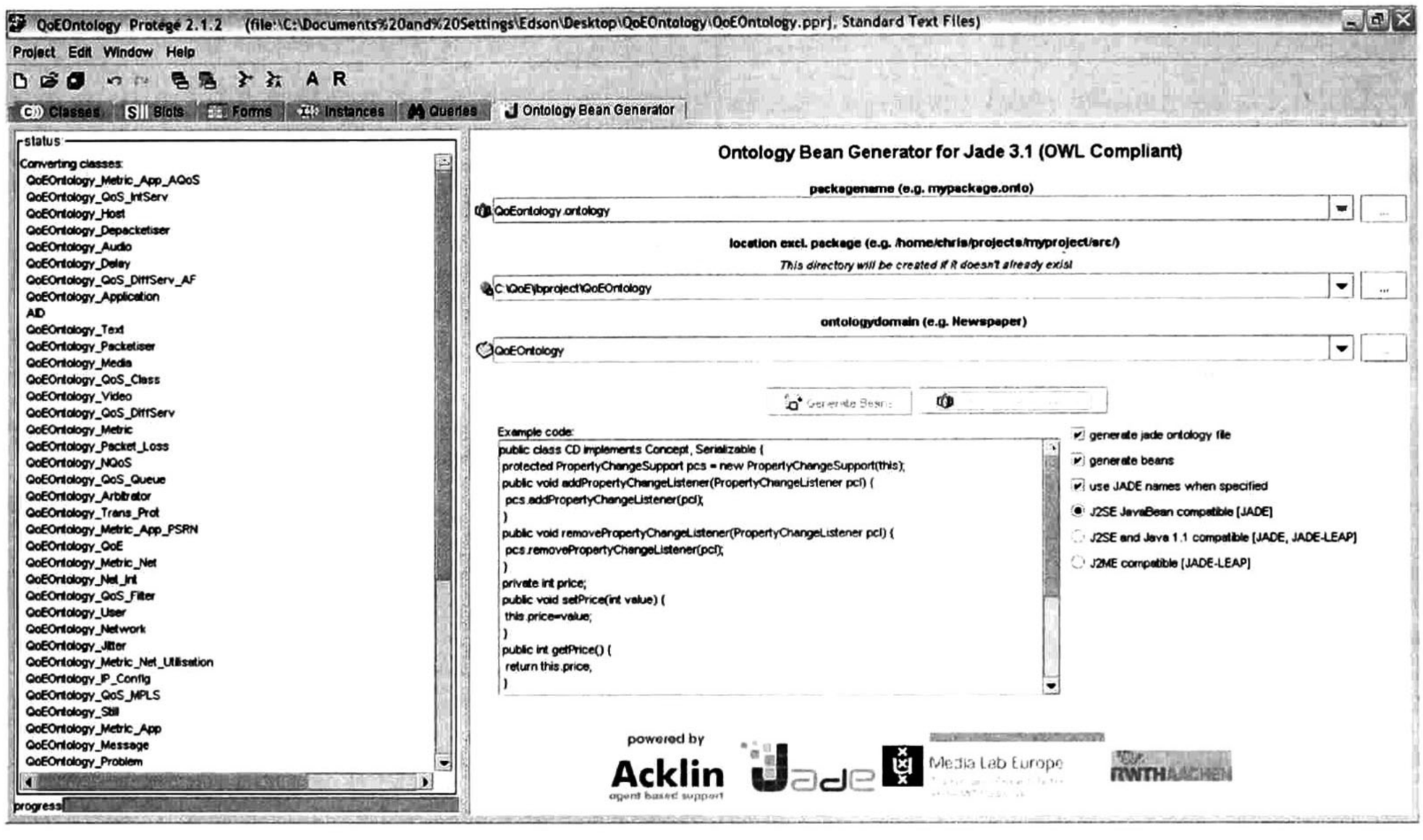


Figura 3.1 Entorno del editor de ontologías protege.

3.3 Conclusiones

Se examinaron varias ontologías de QoS. Se observo que estas varían desde propósito, arquitectura como número de capas o semántica, forma de representación abstracta que va desde lógica de primer orden hasta XML, y demás conceptos de QoS. Se identificaron ventajas y desventajas entre ontologías. La mayoría de las ontologías QoS toman en cuenta conceptos como métricas, protocolos y propiedades de la red. Además, estas son más orientadas a aplicación ya que su propósito es dar soporte a servicios web. De las ontologías examinadas, la única ontología que involucra tanto al usuario como a la aplicación y los aspectos de QoS de la red es la ontología QoE. En otras palabras, a esta le conciernen ambas QoE y QoS. Sin embargo se cree que esta puede ser mejorada y complementada. Por tal motivo se decidió hacer una extensión de la ontología QoE la cual acaba de ser presentada. Del análisis previo es razonable considerar a la ontología QoE como un enfoque más robusto para QoS Management.

CAPITULO 4

4 Gestión y Control de QoS - QoE mediante agentes

Utilizando el paradigma de agentes se presenta la arquitectura y algoritmo propuestos para la gestión y control de mecanismos de QoS de una sesión multimedia.

4.1 Agentes

El paradigma de agentes, por sus características como la de movilidad, es un paradigma único en contraste a paradigmas como el tradicional modelo cliente-servidor en un problema inherentemente distribuido y complejo, tal y como lo es QoS Management. Con este paradigma se pueden ahorrar cantidades considerables de ancho de banda y reducir el tráfico de la red [34].

4.1.1 Agentes, tipos de agentes, tipos de ambientes y Sistemas Multiagentes

No existe una definición universalmente aceptada del término agente, y más aún, existen todavía debates y controversias acerca de este tema. Sin embargo hay un consenso de que la autonomía es una de las características esenciales de un agente, pero más allá de esta característica se concuerda en poco.

En [35] se expresan de la autonomía de la siguiente manera: "*Un sistema es autónomo en el grado en que su comportamiento es determinado por su propia experiencia*"

En [36] se propone una definición para agente: "*Un agente es un sistema computarizado que se sitúa en algún ambiente, y que es capaz de acciones autónomas en este ambiente para así cumplir los objetivos con que fue diseñado*"

Un agente percibe su ambiente mediante sensores; ya sea sensores físicos o por software. Un agente tiene disponible un repertorio de acciones que pueden ser ejecutadas para modificar el ambiente, el cual podría reaccionar de manera no determinística al cabo de la ejecución de estas acciones.

En [35] se menciona que un agente racional es aquel que hace lo correcto. Una acción correcta es aquella que propiciara que el agente sea lo más exitoso posible en cuanto a su desempeño. Se debe de plantear cuando y como evaluar el desempeño de una agente y para ello se utiliza el término de *medida de desempeño* que determina el criterio de que tan exitoso es un agente.

La Inteligencia Artificial (*AI – Artificial Intelligence*) se encarga de diseñar un programa agente, el cual es un **algoritmo** que implementa el mapeo de percepciones a acciones del agente. La **arquitectura** es el dispositivo computacional en el cual este se ejecuta. En general la arquitectura hace que las percepciones de los sensores estén disponibles para el programa, ejecuta el programa, y alimenta a los efectores con las acciones que el programa elije conforme son generadas. Estos elementos forman la **estructura de un agente inteligente** y su relación se puede expresar como:

Agente = arquitectura + algoritmo

Se debe tener muy bien contemplados las posibles percepciones y acciones para el diseño de un programa agente, además de las metas o medidas de desempeño que este debe lograr, y el tipo de ambiente en el que se desenvuelve. Se consideran cuatro tipos de programas agente: agentes reflexivos, agentes reflexivos con estado interno, agentes basados en metas y agentes basados en utilidades.

De estos agentes el de interés es el agente basado en utilidades. En este tipo de agentes las metas no son realmente suficientes para generar comportamientos de alta calidad. Las metas solo proveen una simple distinción entre estados de "*felicidad*" e "*infelicidad*". Una medida de desempeño permite comparaciones entre diferentes estados del mundo o secuencias de estados de acuerdo a exactamente qué tan "*feliz*" harían al agente si tales estados fueran alcanzados. Los términos "*felicidad*" e "*infelicidad*" lo que en realidad quieren decir es que si algún estado del mundo se prefiere a otro, entonces este tiene una mayor utilidad para el agente. Utilidad es una función que mapea un estado a un número real, el cual describe un grado de "*felicidad*" asociado.

$u: E \rightarrow \Re$

u: función de utilidad.

E: conjunto finito de estados del ambiente ($E = \{e, e', e'', \ldots\}$.).

$\Re$: conjunto de los números reales.

La especificación de una función de utilidad permite al agente realizar acciones racionales.

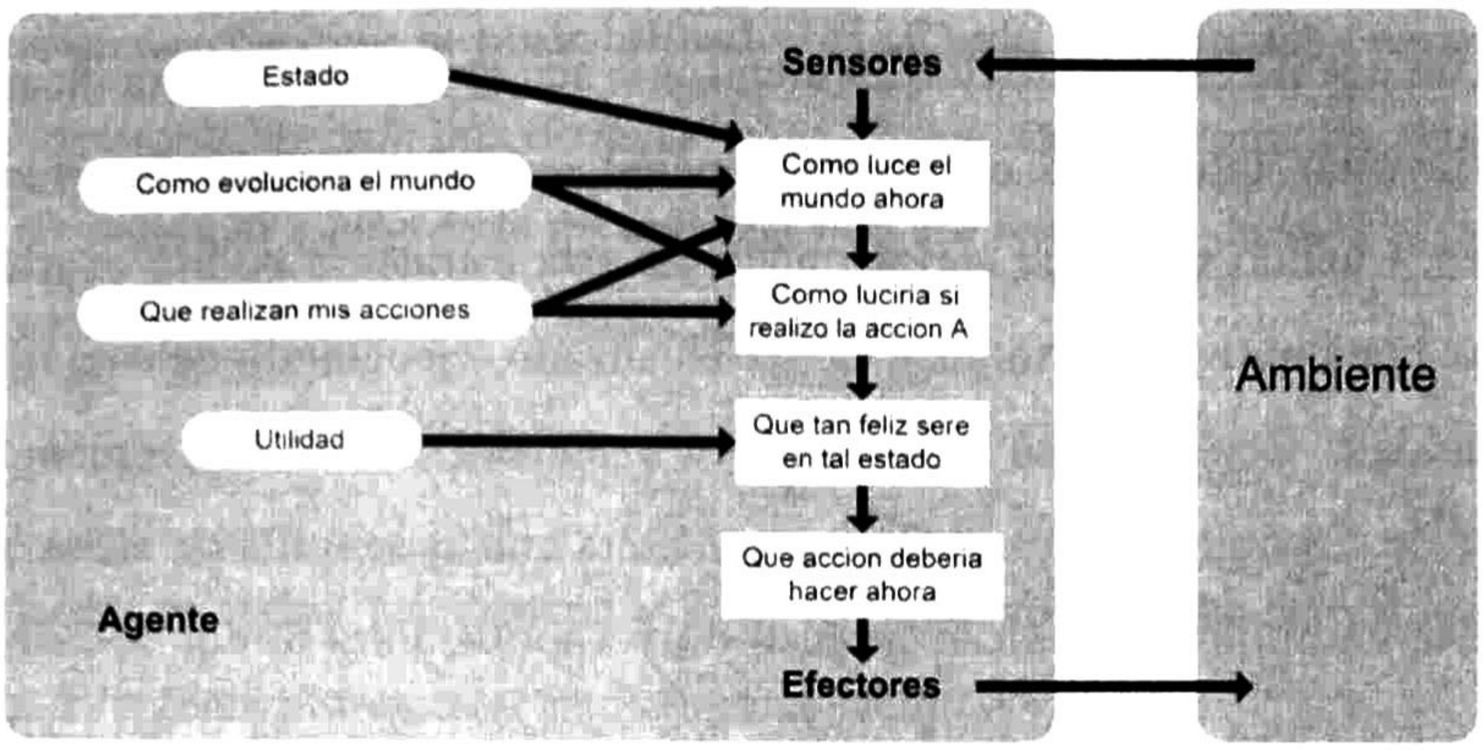


Figura 4.1 Agente basado en utilidades.

Según [35] los tipos de ambientes en los que se desenvuelve un agente pueden clasificarse de acuerdo a sus propiedades. Estos tipos de ambiente son: Accesible/Inaccesible, Determinístico/No determinístico, Estático/Dinámico, y Discreto/Continuo.

Un sistema multiagentes (*Multi-agent Systems - MAS*) es un conjunto de agentes que interactúan entre sí formando un sistema donde el control está distribuido. A este concepto también se le conoce como Inteligencia Artificial Distribuida.

4.1.2 Teoría de la decisión

Existen circunstancias en las cuales los agentes no pueden encontrar una respuesta categórica a una pregunta. En esos casos el agente debe actuar bajo incertidumbre. La incertidumbre proviene del entendimiento incompleto e incorrecto que el agente percibe de las propiedades de su ambiente. El conocimiento de los agentes en el mejor de los casos puede proveer un grado de creencia acerca de una proposición. La principal herramienta para manejar grados de creencia es la Teoría de la Probabilidad la cual asigna un grado de creencia con un valor numérico entre 0 y 1 a una proposición basándose en evidencias [35]. La probabilidad provee una forma de resumir la incertidumbre proveniente de la ignorancia y pereza. Pereza en este contexto indica demasiado trabajo para listar por completo todos los antecedentes y consecuencias necesarias para validar una regla, un problema demasiado difícil de tratar. Es importante no confundir grados de creencia con grados de verdad ya que estos últimos son tema de la Lógica Difusa.

Usualmente, un agente está interesado en elegir una acción que le permita alcanzar su objetivo, y posiblemente podría estar interesado en aquella opción que tiene mayor oportunidad de éxito de alcanzar tal objetivo. Cuando se tienen varios objetivos, esta estrategia podría ocasionar que el agente elija el objetivo que tiene mayor oportunidad de ser alcanzado, ejecutando de esta manera las acciones con mayor oportunidad. Construir un agente de esta manera no es conveniente ya que este no considerara el valor de sus objetivos, y así elegirá una meta que es sencilla de alcanzar pero no valiosa, en lugar de una meta complicada de alcanzar pero muy valiosa. Para tomar decisiones de ese tipo el agente debe tener preferencia sobre alguna opción con respecto de otras y para ello se hace uso del concepto de *utilidad*. La Teoría de la Utilidad es la herramienta que se utiliza para representar y razonar acerca de preferencias [35]. Una utilidad es un valor que se asocia con un estado del mundo, y que representa el valor que el agente le da a establecerse en ese estado del mundo. Las utilidades proporcionan un medio conveniente para representar las preferencias de un agente, por lo que es posible definir una función de utilidad que represente de manera confiable las preferencias de modo que un estado S_i se prefiere a S_j, si y solo si este tiene una utilidad mayor para el agente.

Ahora, considérese que un agente tiene un conjunto **A** de posibles acciones, de las cuales cada acción A_i tiene un rango de posibles resultados debido que son no determinísticas. El valor de tomar una acción en particular depende de en qué estado del mundo se encuentre y también de la acción que se elige tomar. El agente necesita buscar un valor de $U(S_j)$ en donde S_j es el estado en el que se encontrará después de realizar la acción. Haciendo esto para cada acción posible, el agente podrá entonces elegir la acción que lo lleve al estado que valore más. Un agente construido de esta manera, sin equivocación elegirá la acción con el mayor valor proporcionado por su función de utilidad, y así lograr el objetivo más valioso sin importar la dificultad de este.

Las preferencias (expresadas por medio de utilidades) son combinadas con las probabilidades (grados de creencia) en la teoría general de decisiones racionales llamada Teoría de la Decisión [35]. Esta teoría expresa que:

Teoría de la Decisión = Teoría de la Probabilidad + Teoría de la Utilidad

La Teoría de la Decisión [37] provee de una herramienta poderosa con la cual analizar escenarios en los cuales una agente debe tomar decisiones en un ambiente impredecible. Para construir agentes de

este tipo más sensibles se combina la probabilidad y el cálculo de utilidades para cada acción y así calcular la *utilidad esperada* de cada una de estas. Esto es calcular un promedio ajustado de la utilidad de cada resultado, en donde el ajuste es la probabilidad de ese resultado dada la acción a ser realizada. Ya que cada resultado es por sí mismo un estado, se tiene:

$$EU(A_i) = \sum_{S_j \in \mathbf{S}} \mathrm{P}(S_j | A_i, E) U(S_j)$$

donde **S** es el conjunto de todos los estados y E es la evidencia previa que el agente pudiera tener acerca del ambiente. El agente entonces seleccionara la acción A^* donde:

$$A^* = arg \max_{A_i \in \mathbf{A}} \sum_{S_j \in \mathbf{S}} \mathrm{P}(S_j | A_i, E) U(S_j)$$

La idea fundamental de esta teoría es que un agente es racional si y solo si este selecciona la acción que produce la máxima utilidad esperada, ajustada sobre todos los posibles resultados de una acción. Este es llamado el principio de Máxima Utilidad Esperada (*Maximum Expected Utility - MEU*). Así, probabilidades y utilidades son combinadas en la evaluación de una acción, midiendo la utilidad de un resultado en particular junto con la probabilidad de que este ocurra. En resumen la teoría de la probabilidad describe lo que un agente debería creer en base a evidencia, la teoría de la utilidad describe lo que el agente desea, y la teoría de la decisión pone a estas dos juntas para describir lo que un agente debería hacer.

Dado que los mecanismos básicos de la teoría de la decisión encajan adecuadamente en el contexto de agentes inteligentes, es quizás de sorprenderse que no sean empleados de gran manera en este campo, por lo que sistemas multiagentes de este tipo son muy escasos [38]. Algunas investigaciones en las cuales se utiliza actualmente la Teoría de la Decisión se mencionan también en [38]. Entre ellas destacan las concernientes a comercio electrónico en donde los agentes son responsables de la compra-venta de productos y servicios a través de internet. En [39] se ha investigado el impacto de shopbots en mercados, modelando el comportamiento de compradores y vendedores mediante técnicas de teoría de la decisión y teoría del juego. En los sistemas multiagentes, los agentes usualmente tratan de maximizar su utilidad esperada. Como resultado, surge el interés de calcular soluciones bajo racionalidad limitada. Este enfoque intenta ser racional en el sentido de calcular la máxima utilidad esperada, pero con el conocimiento de que existe un límite de recursos para hacer el

cálculo. En [40] realizan la búsqueda de una solución optima hasta que el costo de continuar la búsqueda sobrepase el mejoramiento de la solución, que de haber continuando con la búsqueda se había encontrado. Esto lo hacen introduciendo la noción de utilidad praxeológica, una medida que modela de manera explícita los recursos consumidos, y permite un balance de estos contra el deseo de obtener la mejor solución. En [41] comienzan con una meta del agente y consideran como este podría razonar acerca de sus metas. Utilizan esto para definir su función de utilidad mediante un enfoque basado en lógica. Estos dos artículos hacen combinación de creencias y utilidades.

4.2 QoS Management en transmisiones multimedia

Según [42] desde el punto de vista del usuario, se puede establecer QoS en término de los siguientes parámetros: *resolución*, *distorsión*, *sincronización* e *interactividad*. Desde el punto de vista de la red los parámetros de QoS más comunes son: *retraso*, *jitter*, *Errores (BER - Bit Error Rate)* y *Pérdida de paquetes (PER – Packet Error Rate –Packet Loss)*. En términos generales el retraso afecta la interactividad; el jitter afecta la sincronización y la distorsión se ve afectada por la perdida y errores de los paquetes. Además de la ITU, en [43] establecen límites de estos parámetros en aplicaciones multimedia distribuidas, 100-250 milisegundos para retraso, 5-10 milisegundos para jitter, 0.001-0.01% para perdida de paquetes y 0.01-0.1% para errores. Es obvio que, mientras menores sean los valores de estos parámetros, mayor será la QoS.

En [44] se mencionan las principales funciones de QoS Management y en [45] se argumenta que existen tres fases durante el tiempo de vida de una sesión multimedia. En cada una de estas fases intervienen diferentes funciones de QoS Management las cuales se ilustran en la figura 4.2.

El puno clave de QoS Management es como implementar estas funciones de tal manera que se optimice el uso de los recursos de la red.

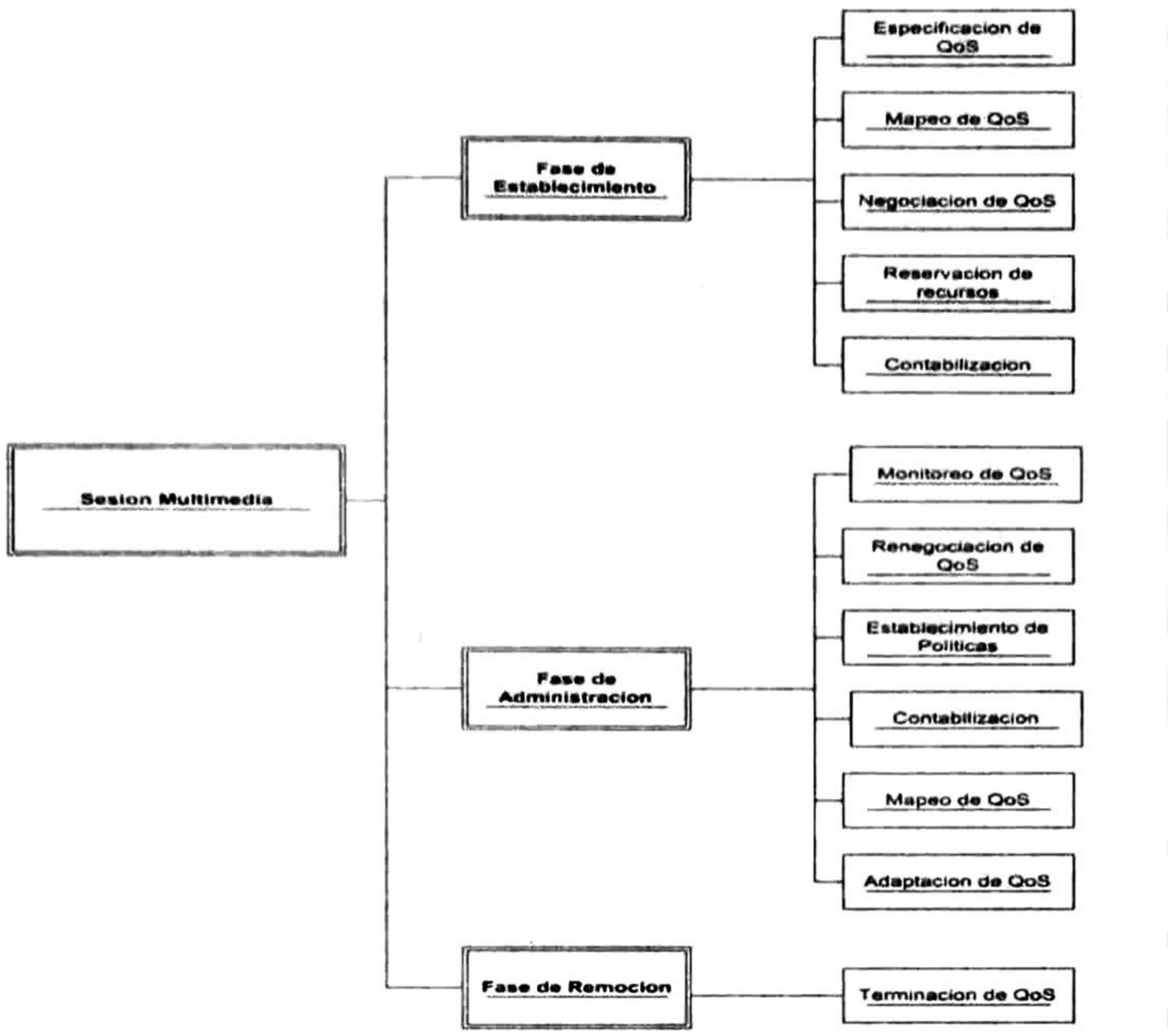


Figura 4.2 Fases de una sesión multimedia y sus respectivas funciones de QoS.

4.3 Arquitectura para QoS - QoE Management

Innumerables arquitecturas de agentes para QoS Management han sido propuestas, entre ellas se mencionan las siguientes.

En [46] se propone una arquitectura para QoS Management de manera distribuida, en donde se pueda monitorear el estado de la red a diferentes niveles de detalle y tener una mejor percepción de las situaciones que ocurren de manera local.

En [42] presentan una arquitectura para la negociación y gestión de QoS en sistemas multimedia distribuidos. Esta se basa en el concepto de agencia. Una agencia se encuentra situada en cada nodo y es allí en donde los agentes fijos residen y los agentes móviles son recibidos, procesados y despachados. La estructura básica de comunicación entre estos agentes es un contrato. Este contiene descriptores de recursos, como flujos, buffers, etc.; restricciones de negociación, como límites de tiempo, márgenes aceptables, etc.; y la QoS deseada. Los agentes interactúan mediante el procesamiento de estos contratos y enviando estos de una agencia a otra dentro de agentes móviles.

En [47] se presenta una arquitectura de agentes para QoS Management cooperativo. Se propone gestión de QoS de manera descentralizada y cooperativa. Los agentes son colocados en los componentes del sistema, es decir en los ruteadores y hosts. Cuando el usuario requiere de una QoS específica comienza la cooperación entre agentes para satisfacer de mejor manera posible al usuario.

En [48] se propone una arquitectura de agentes por capas. Cada capa representa límites conceptuales de los diferentes tipos de servicios que presta el agente. Esto obedece a que los agentes de más alto nivel, que se suponen más sofisticados y complejos, dependen de las capacidades de los agentes de más bajo nivel, que son más simples y elementales.

En [49] presentan un sistema multiagentes que pretende cumplir con las características de: *Subjetivo*; que maneje a detalle las preferencias del usuario para el manejo de recursos de la red. *Genérico*; que opere en una gran variedad de ambientes de telecomunicaciones. *Dinámico*; que reaccione con diferentes mecanismos de QoS a la variación de parámetros del ambiente. *Semiautónomo*; debido a su dinamismo, que este tome la mayoría de las decisiones de manera autónoma. *Adaptable*; que re-distribuya los recursos disponibles para adaptarse a nuevos servicios ofrecidos o modificaciones en las restricciones de la red. Y *Basado en XML*; para manejar la comunicación del agente mediante XML.

En [34] proponen un sistema multiagentes que pretende cumplir con ciertas características que consideran *principios funcionales de una arquitectura QoS*. Estas son: *Integración*; mediante la definición de agentes en diferentes niveles de abstracción (usuario, sistema y red). *Separación*; mediante la autonomía de los agentes, ya que estos no interfieren con otras funciones del sistema ni la transmisión del flujo. *Transparencia*; ya que tanto usuarios y aplicaciones pueden delegar de manera transparente tareas de negociación y gestión a los agentes. *Manejo de recursos de manera asíncrona*; ya que los agentes se ejecutan de manera concurrente y asíncrona. *Desempeño*; debido a que los agentes solo realizan funciones de gestión de manera que no afectan el desempeño de los protocolos de red. Utiliza agencias para que sus agentes fijos residan en ella y para despachar a los agentes móviles.

En [50] proponen un conjunto de agentes especializados que dinámicamente decidan la cantidad de recursos asignados a un enlace virtual. Este enfoque utiliza IntServ como único mecanismo de QoS.

En [51] se propone una arquitectura, también por capas, para ambientes multimedia distribuidos. El proceso de QoS Management es dividido en sub-tareas, las cuales forman las capas, y para cada una de ellas existe un agente que se encarga de realizarla.

En [52] se presenta una arquitectura para el control y valuación de servicios multimedia dentro de un dominio. Esta arquitectura basada en agentes es genérica, distribuida e independiente de la capa de transporte. Se enfoca en el desarrollo de componentes de software inteligentes móviles y distribuidos que interactúan entre ellos y sirvan de interface con capas de más bajo nivel que controlan los mecanismos de QoS para dinámicamente administrar y valuar a una sesión multimedia.

Finalmente en [21] se presenta una arquitectura para el marco de trabajo de QoE cuya arquitectura está dictada por las tareas que realiza cada uno de los agentes que la componen.

Como ya se pudo apreciar, existe una gran diversidad de arquitecturas para QoS Management mediante agentes en sistemas distribuidos multimedia. Aunque cada una de ellas tiene un enfoque particular por su propósito con QoS Management, se observan dos principales perspectivas de arquitectura que son (i) por capas o sub-tareas, y (ii) por las características que posee. La arquitectura que aquí se propone se extiende de [21] de tal manera que cumpla con el marco de trabajo de QoE, además de incluir ciertas características deseables para QoS y que las funciones de QoS Management marquen las capas o sub-tareas que deben ser realizadas por los agentes que la integran.

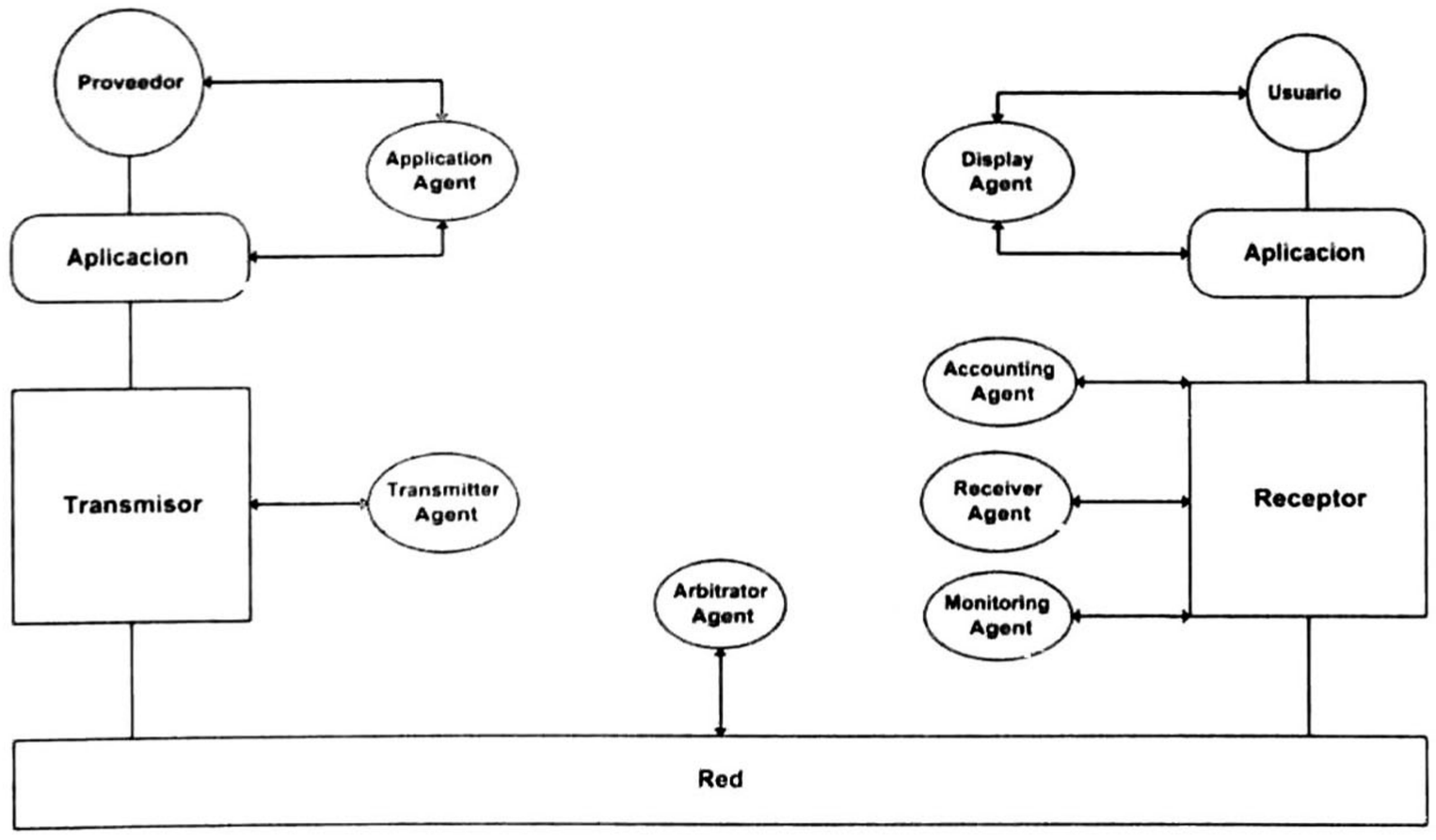


Figura 4.3 Arquitectura de Agentes para el marco de trabajo QoE.

Esta arquitectura para sistemas distribuidos multimedia cuenta con las siguientes características: *Subjetivo*; dado que toma en cuenta las preferencias del usuario para la gestión de los recursos de la red. *Genérico*; ya que está pensado para que funcione en redes heterogéneas. *Dinámico*; porque cuenta con diferentes mecanismos de QoS y pueden ser activados conforme a las variaciones de los parámetros del ambiente. *Autónomo*; debido a que los agentes toman las decisiones de manera autónoma. *Adaptable*; reasigna los recursos disponibles para adaptarse a nuevas solicitudes y modificaciones en las sesiones ya existentes. *Integración*; porque existen agentes para realizar las funciones de QoS Management dentro de las fases de la sesión multimedia que componen todo un proceso. *Separación*; debido a que los agentes actúan de manera autónoma estos no interfieren en funciones del sistema ni de la transmisión. *Transparencia*; ya que los agentes realizan de manera transparente sus tareas sin que el usuario o la aplicación deban intervenir. *Manejo de recursos de manera asíncrona*; puesto que los agentes se ejecutan de manera concurrente y asíncrona. *Desempeño*; porque los agentes no afectan el desempeño de los protocolos de red puesto que solo realizan funciones de gestión. Los agentes que lo integran son los siguientes:

- *Arbitrator Agent*: se encarga de la negociación y renegociación de QoS. Cuando se ha realizado una negociación, reserva y asigna los recursos acordados, y en caso de una renegociación debe llevar a cabo la adaptación correspondiente. Realiza el mapeo de métricas de aplicación a red.
- *Monitoring Agent*: este se encarga de monitorear la red y generar estadísticas. Cuando las métricas de la red acordadas en el SLA no se cumplen informa al *Arbitrator Agent* para que realice las acciones pertinentes.
- *Application Agent*: sirve como interfaz de usuario del lado del transmisor. Se encarga de establecer los parámetros de especificación de QoS para iniciar la transmisión y crea a los agentes *Monitoring Agent*, *Agent*, *Receiver Agent* y *Transmitter Agent*.
- *Accounting Agent*: este se encarga de valuar la transmisión para dar a conocer la tarifa y posteriormente expedir el recibo correspondiente.
- *Display Agent*: este se encarga de proyectar la transmisión además de mostrar el menú en el cual el usuario puede solicitar la reducción o mejoramiento de la calidad. También muestra la tarifa actual y realiza el mapeo de métricas del usuario a métricas de la aplicación.
- *Transmitter Agent*: este se encarga de generar la transmisión multimedia.
- *Receiver Agent*: este se encarga de recibir la trasmisión multimedia. Crea el *Display Agent* y el *Accounting Agent*.

4.4 Algoritmo para QoS - QoE Management

Existen varias estrategias que se utilizan para el manejo inteligente de QoS. A continuación se mencionan algunas de ellas.

En [53] se habla acerca de la aplicación de algoritmos basados en lógica difusa y redes neuronales para mejorar la QoS en redes de comunicaciones. Debido a que los algoritmos basados en lógica difusa y redes neuronales usualmente tienen la habilidad inherente de aprender, estos se adaptan bien a comportamientos causados por ráfagas y mezclas complejas de tipos de tráfico. La capacidad de cómputo requerida es relativamente baja, lo cual es benéfico en términos de operaciones de tiempo real. Algoritmos neuro-difusos para el mejoramiento de QoS se han reportado desde los 90s. Su aplicación se enfoca al control de tráfico para la puesta a punto de flujos y evitar congestiones. Los algoritmos de predicción de tráfico pueden mejorar la precisión de controladores cuando se considera el retraso de paquetes. Los algoritmos de predicción neuro-difusos generalmente se pueden adaptar a patrones de tráfico y no dependen en un modelo matemático específico para operar. En el trabajo citado el autor extiende un algoritmo de predicción de un solo paso a uno de múltiples pasos y lo aplica en redes de comunicaciones para la predicción de tráfico multimedia. Muestra resultados prometedores para mejorar el desempeño de QoS. El modelado y predicción de tráfico basado en lógica difusa puede ser utilizado para caracterizar tráfico multimedia de manera más precisa. Como el tráfico sigue generalmente una distribución no gaussiana y no estacionaria, hablando estadísticamente, los métodos tradicionales podrían tener un desempeño no satisfactorio o incurrir en un gran costo computacional.

En el modelo presentado por [51] un usuario puede especificar la QoS de un flujo seleccionando a través de un rango de parámetros de QoS de nivel aplicación y la utilidad de los parámetros. Esto es, el usuario puede seleccionar varias especificaciones QoS como candidatas para un flujo al comenzar su aplicación y de esta manera evitar una interacción constante entre este y la aplicación. A ese tiempo, para cada candidata QoS del flujo el usuario también establece un valor de utilidad en el rango de 1 a 100 que representa la satisfacción que tendrá cuando dicho flujo tenga esa QoS candidata. Mientras mayor sea el valor de utilidad mayor será su satisfacción. Además establece algunos otros parámetros para indicar prioridades. Para la negociación de QoS el emisor y el receptor se intercambian entre sí sus funciones de utilidad para definir una relación entre QoS y utilidad, y otorgar los recursos solicitados, de otra manera comienza una renegociación.

En la propuesta de [52] a un usuario se le permite seleccionar la prioridad relativa de su sesión multimedia que la red habilitada con QoS y el proveedor del servicio deberán soportar. Las preferencias del usuario se definen en términos de una función de utilidad y voluntad de pagar por el servicio. La utilidad es seleccionada por el usuario de un número discreto de curvas costo-beneficio y es ajustada por el agente mientras este va aprendiendo del usuario de manera adaptiva. Se requiere que el usuario programe sus agentes con sus propias preferencias y perfiles de servicio. El *agente usuario* comienza a negociar con los *agentes servicio* utilizando la función de utilidad del usuario para decidir cuando los recursos asignados son aceptables al precio ofrecido por el *agente servicio*.

Haciendo un análisis de las propuestas anteriormente mencionadas se pueden observar dos alternativas para el manejo inteligente de QoS. La primera es mediante redes neuronales, llevando a cabo predicciones. Esta alternativa es muy prometedora pero bastante compleja. El inconveniente es realizar un entrenamiento adecuado de la red neuronal que además podría resultar costoso al tener que enseñarle formas como patrones de tráfico, situaciones de congestión, etc. Cabe señalar que no existe duda que posteriormente al entrenamiento adecuado los resultados de esta sean exitosos. La otra alternativa toma en cuenta preferencias mediante funciones de utilidad ya definidas. La propuesta de esta tesis toma esta alternativa pero incorporando creencias, es decir probabilidad, para llevar la toma de decisiones dentro de QoS Management a un contexto aun más racional. El agente que se encarga de esto es el *Arbitrator Agent*. Su adaptación a una agente basado en utilidades se muestra a continuación.

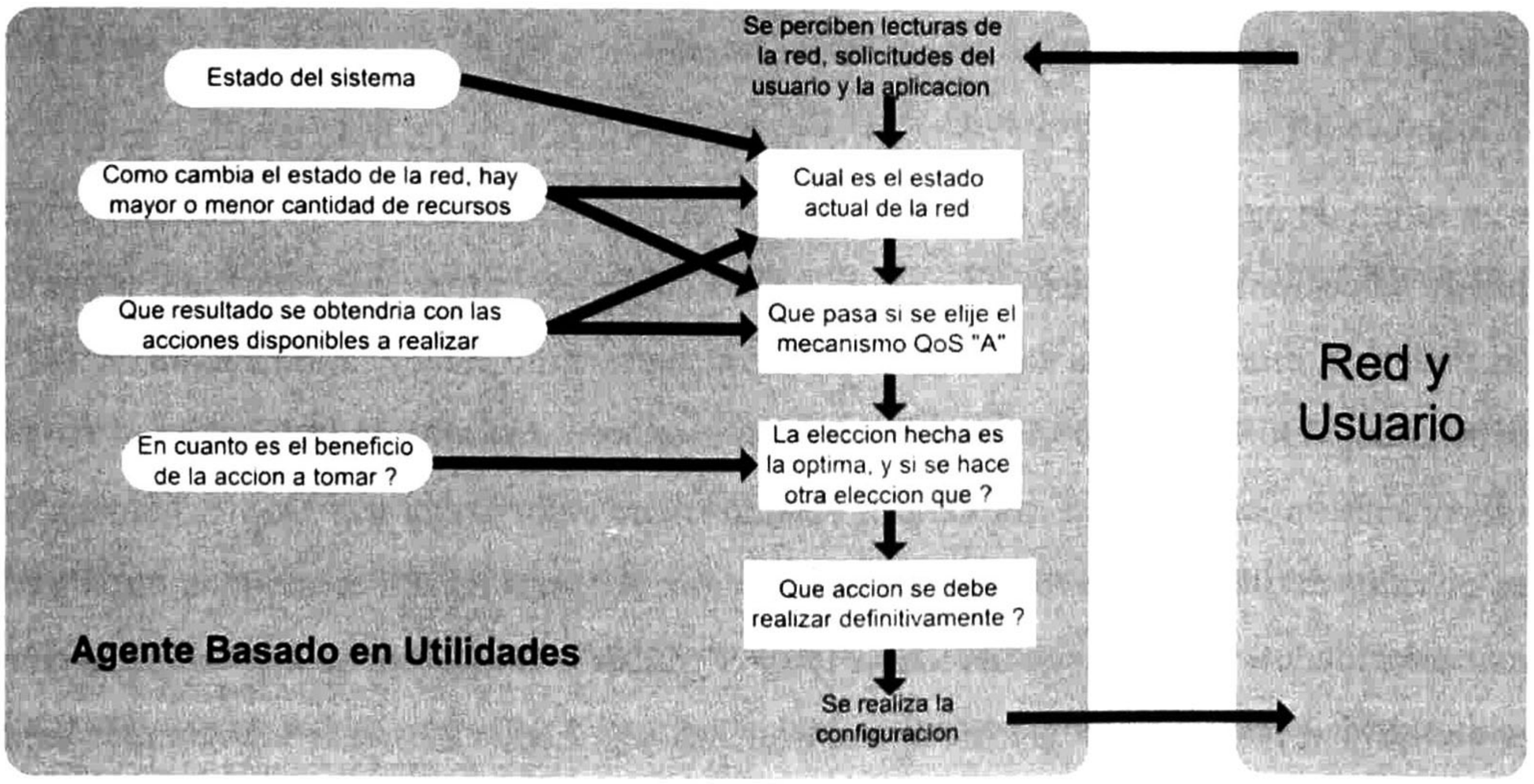


Figura 4.4 Agente basado en utilidades para el marco de trabajo QoE.

Se identifica que el ambiente en el que se encuentran los agentes tiene las siguientes características: *Dinámico*; ya que en cada instante puede cambiar el tipo y la cantidad de tráfico que transita por esta, no es posible asegurar en qué momento se encontrara saturada o libre para efectuar una transmisión. *Inaccesible*; ya que los agentes no están al tanto de todas sesiones que existen entre aplicaciones a través de la red tales como navegación web, correo electrónico, mensajería instantánea, entre otras. Todo este tráfico generado por otras aplicaciones se considera ruido (*background traffic*), sin embargo se mantiene un registro de las sesiones que sí son de interés. *No determinístico*; porque al efectuar el agente sus acciones para administrar la red no se garantiza en su totalidad el resultado por lo que el monitoreo continuo del ambiente es necesario junto con la adaptación de QoS. *Discreto*; se tiene un número finito de acciones a realizar por parte del agente que esta dado por las posibles configuraciones QoS, así como por las percepciones, que en este caso son las solicitudes del usuario y de otros agentes. También se tiene un conjunto finito de posibles estados.

Suponga que se cuenta con una cantidad M de recursos existentes para su uso en un dispositivo de red, ya sea ancho de banda, tamaño de buffer, etc. Se desea dividir esta cantidad de recursos en fragmentos con una misma medida K. De esta manera se obtendrían M/K fragmentos de recursos de proporciones iguales para ser asignados o retirados a un servicio dado, es decir M/K posibles asignaciones de recursos.

El conjunto de posibles configuraciones para DiffServ está dado por:
Config_DiffServ = {Config_DS$_1$, Config_DS$_2$,..., Config_DS$_n$}, donde Config_DS$_i$ es una tupla formada por la frecuencia de transmisión, tamaño de buffer, disciplina de encolamiento y mecanismo QoS (Rate, Buffer, Queue_Discipline, DiffServ) y n indica el número de posibles servicios DiffServ que a lo más son M/K. Aunque AF tiene 4 clases con 3 diferentes niveles de prioridades cada una que forman un total 12 servicios diferentes, un mismo servicio puede ser modificado con una configuración distinta pudiendo ser cualquiera de las M/K mencionadas. Rate y Buffer toman valores de los fragmentos de recursos hechos.

El conjunto de posibles configuraciones para IntServ está dado por:
Config_IntServ = {Config_IS$_1$, Config_IS$_2$,..., Config_IS$_m$}, donde ConfigIS$_i$ es una tupla formada de la misma manera que ConfigDS, solo que le mecanismo QoS es IntServ, y m esta dado por el número

de servicios (GS) deseados que lo sumo también son *M/K*, en los cuales se reserva cierta cantidad de recursos de manera garantizada.

MPLS no establece una frecuencia de transmisión, ni un buffer para el almacenamiento temporal de paquetes, tampoco una disciplina de encolamiento para manejar el comportamiento de estos. MPLS simplemente utiliza etiquetas para acelerar la transmisión de los paquetes a través de un LSP. Por lo tanto solo existe una configuración única para MPLS que es Config_MPLS.

BE evidentemente solo cuenta con una configuración única Config_BE, que equivale a la ausencia de configuración o configuración vacía. Esta no se toma en cuenta como una acción del agente ya que no se necesitan establecer parámetros de configuración para proveer del servicio BE.

Las anteriores configuraciones forman el conjunto de acciones que intervienen en la decisión del agente. Sea **A** el conjunto de acciones posibles a realizar del agente, entonces:

A = {Config_DiffServ, Config_IntServ , Config_MPLS}.

La cantidad de recursos, las reservaciones llevadas a cabo y las sesiones se caracterizan de la siguiente manera:

- Registro_Capacidad(Link_Band_Width, Device_Buffer_Size)
- Registro_Reservacion(Reserved_Band_Width, Buffer_Size, Session_Id, QoS_Mechanism)
- Registro_Sesion(Id_Session, Source, Dest, Request)

A través de estos registros se puede calcular la cantidad de recursos disponibles y demás para ser utilizados en la toma de decisiones.

Se nombra **S** al conjunto de posibles estados a los que se puede llegar en el instante en que se toma una decisión dada la cantidad de recursos disponibles, recursos otorgados y recursos solicitados el cual se define así: $\mathbf{S} = recursos_{disponibles} \times recursos_{otorgados} \times recursos_{solicitados}$

donde $recursos_{disponibles}$ tiene cardinalidad uno ya que solo existe una única cantidad de recursos disponibles en ese instante y es el mismo caso para $recursos_{solicitados}$.

La cardinalidad de $recursos_{otorgados}$ es: $recursos_{disponibles} / K$

donde K es el valor de la medida que divide la cantidad de recursos existentes.

Ya que se han definido las acciones y estados se procede a definir la función de utilidad. Esta función debe de arrojar como resultado una mayor utilidad cuando se satisface por completo la demanda de transmisión dejando la mayor cantidad de recursos disponibles.

Para determinar que tan preferible es un estado del ambiente respecto a otros, primero se utiliza la función *Result*(), la cual recibe como parámetro una acción, es decir una posible configuración QoS, y devuelve la caracterización del ambiente correspondiente al estado que resultaría después de realizar la acción a_i.

$$Result(a_i) = s_j$$

Con la información de la caracterización s_j se toma de los recursos disponibles los posibles recursos a ser otorgados a la sesión multimedia y los recursos solicitados por esta para obtener una razón entre ellos, a esta se le llama r.

$$r = \frac{s_j.recursos_{otorgad\ os}}{s_j.recursos_{solicitados}}$$

Una vez obtenida r, se calcula la utilidad que esta tendrá con la siguiente función de utilidad:

$$u(r) = \begin{cases} 1/r, & r > 1 \\ r, & 0 \leq r \leq 1 \end{cases}$$

La función de utilidad pretende satisfacer en lo más posible la solicitud de la sesión multimedia de manera que se otorgue la menor cantidad de recursos disponibles, esto para optimizar los recursos. La función de utilidad devuelve r cuando esta se encuentra entre los valores 0 y 1, esto quiere decir que la proporción de recursos es menor o igual a la cantidad de recursos solicitados. Mientras más se acerque r a 1 la utilidad será mayor, sin embargo si r sobrepasa ese valor no quiere decir que se está obteniendo aun más utilidad, por el contrario, mientras r se aleja más de 1 entonces la utilidad disminuye, ya que esto significa que efectivamente se está satisfaciendo por completo la solicitud, pero otorgando más recursos de los necesarios lo cual no es optimo. Por esta razón cundo r sobrepasa 1 entonces la función de utilidad devolverá su valor inverso ($1/r$) el cual nos dirá de manera normalizada la verdadera utilidad. De esta manera se toma en cuenta el estado del la red en donde con la menor cantidad posible de recursos se satisfacen las demandas de transmisión para las sesiones multimedia.

Dadas las condiciones de la red, existen circunstancias en las que es menos probable establecer una ruta reservada o un circuito virtual para una sesión, a que los paquetes de esa sesión viajen marcados con prioridades quizás por rutas alternas. El establecimiento de estas depende de la utilización de la red, es decir, de la disponibilidad de recursos. La utilización de la red no comprende únicamente tráfico BE del cual se puede prescindir para otorgar QoS a una nueva sesión multimedia, también puede comprender tráfico de otras sesiones las cuales ya han reservado recursos y disponen de ellos para su propia QoS. Será más probable establecer una ruta reservada o un circuito virtual mediante señalización cuando la utilización de la red sea menor, es decir existe mayor disponibilidad, y por lo tanto se cree son preferibles, de otra manera se cree preferible utilizar prioridades en los paquetes. Se pudiera determinar la utilización/disponibilidad de la red de las siguientes maneras:

- Mediante estudios previos de red.
- Mediante monitoreo exhaustivo, sin embargo tiene un alto costo para el sistema. El agente necesitaría ser accesible totalmente a su ambiente, lo cual es muy complejo, para así determinar la utilización/disponibilidad de la red al instante.
- Agente que aprende. Manteniendo un registro o historial estadístico de la red a través del tiempo para predecir la utilización/disponibilidad de esta.

Mediante estas estrategias se podría obtener una evidencia a priori que llevara a determinar ciertas probabilidades de utilización/disponibilidad de la red bajo tales condiciones. Se propone establecer tres rangos de utilización de la red que son Baja, Media y Alta, dentro de los cuales se estima una probabilidad de factibilidad de establecer una sesión multimedia bajo un mecanismo QoS dada la utilización de la red. Estas probabilidades se muestran en la siguiente tabla:

Utilización de la Red	*DiffServ*	*IntServ*	*MPLS*
Alta	Media	Baja	Baja
Media	Media	Media	Media
Baja	Alta	Alta	Alta

Tabla 4.1 Probabilidad de factibilidad de establecer el mecanismo QoS dada la utilización de la red.

Por su definición en los RFCs, se sabe de antemano el desempeño que ofrece cada uno de los mecanismos QoS. Por lo tanto se puede asignar la probabilidad de desempeño que tiene cada uno de estos en base a las características del servicio. Servicios Integrados garantiza la cantidad de recursos

otorgados y de esa amanera cumple con sus métricas de desempeño por lo que se le asigna una probabilidad de 1. Una vez hecha la reservación esta cumplirá cabalmente con la QoS solicitada, garantizando así el servicio. Servicios Diferenciados provee diferentes clases de servicio, cada una con cierta probabilidad de descarte de paquetes bajo circunstancias de congestión con lo cual se asegura al menos un porcentaje de paquetes exitosamente entregados bajo las métricas de desempeño acordadas. La probabilidad asignada a DiffServ está dada por 1 menos la probabilidad de descarte de paquetes de la clase, que resulta en la probabilidad de paquetes no descartados. Si la probabilidad de descarte es cero se puede considerar una probabilidad de 1. MPLS por su arquitectura permite crear circuitos virtuales de manera que se acelera la transmisión de los paquetes, sin embargo no garantiza una cantidad de recursos en particular por lo cual se considera tiene una probabilidad menor a 1.

Mecanismo QoS	*Probabilidad de desempeño*
Servicios Integrados	= 1
Servicios Diferenciados	≤ 1
MPLS	< 1

Tabla 4.2 Probabilidad desempeño asignada a cada mecanismo QoS.

De manera implícita se muestra una relación de orden entre los mecanismos QoS, que indica la preferencia de un mecanismo sobre otro bajo circunstancias de igual utilidad. La relación de orden es la siguiente: $BE \prec MPLS \prec DiffServ \prec IntServ$

En base a estas probabilidades y las condiciones en las que ocurren se construye la siguiente red de creencias (*belief network*):

Figura 4.5 Red de creencias para el marco de trabajo QoE.

Una red de creencias es una estructura de datos que representa la dependencia entre variables y facilita una especificación concisa de la distribución de probabilidad conjunta. La red de creencias es un grafo en el cual un conjunto de variables aleatorias forman los nodos de la red; un conjunto arcos dirigidos conectan pares de nodos. El significado intuitivo de un arco del nodo X al nodo Y es que X tiene influencia directa sobre Y. Cada nodo tiene una tabla de probabilidad condicional que cuantifica los efectos que los padres de ese nodo tienen sobre él. Los padres de un nodo son aquellos nodos que tienen arcos apuntando hacia él. El grafo no tiene ciclos dirigidos, por lo que se puede clasificar como un grafo dirigido acíclico.

La red de creencias indica que la utilización de la red afecta directamente en la factibilidad de establecer una sesión multimedia bajo un mecanismo QoS en particular, pero el desempeño de la transmisión durante la sesión depende únicamente del mecanismo QoS que se haya establecido.

Una vez que se ha especificado la topología de la red de creencias, solo se necesita especificar la probabilidad condicional para cada nodo que participa en dependencias directas, y usar estas para calcular cualquier otro valor de probabilidad. Estas probabilidades se establecen en una tabla de probabilidad condicional. Cada fila de la tabla contiene la probabilidad condicional de cada valor del nodo para un caso de condición. Un caso de condición es una posible combinación de valores de los nodos padre, que representan un evento atómico. Un nodo sin padres tiene solo una fila, representando la probabilidad a priori de cada posible valor de la variable. La topología de la red de creencias se puede ver como una base de conocimiento abstracta ya que representa de manera general la estructura del proceso causal en el dominio especifico. Ahora se procede a complementar la red de creencias así.

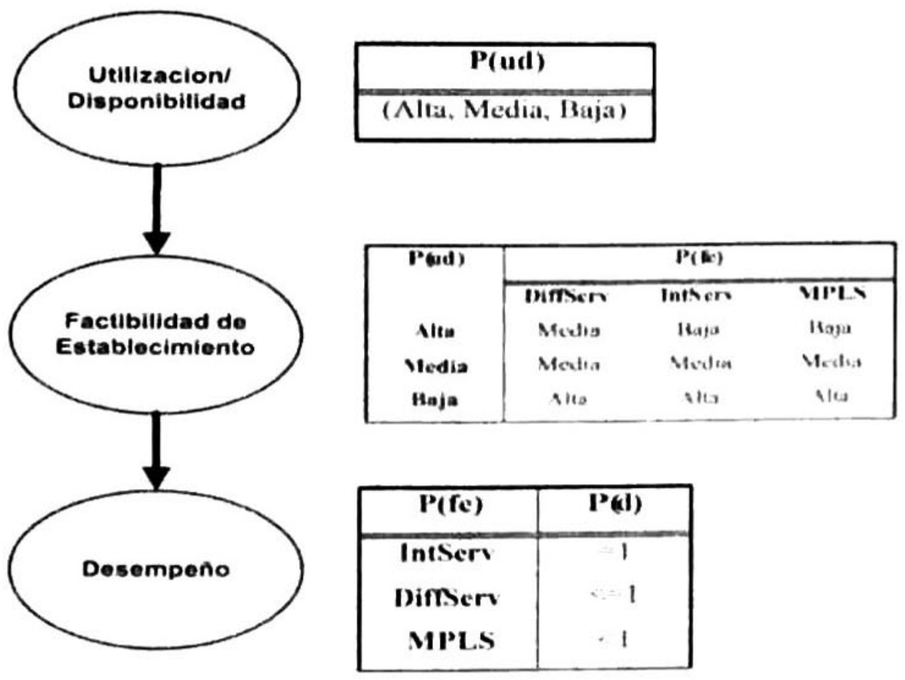


Figura 4.6 Red de creencias con sus tablas de probabilidad condicional.

La Distribución de probabilidad conjunta (*Joint Probability Distribution*) especifica por completo las asignaciones de probabilidad de un agente a todas las proposiciones en el dominio. Un modelo probabilístico del dominio consiste de un conjunto de variables aleatorias que pueden tomar valores particulares con cierta probabilidad. Sean las variables $X_1 \ldots X_n$, un evento atómico es una asignación de valores particulares a todas las variables, en otras palabras, una especificación completa del estado del dominio. La distribución de probabilidad conjunta $\mathbf{P}(X_1,\ldots,X_n)$ asigna probabilidades a todos los eventos atómicos posibles.

Una de las maneras de entender la semántica de una red de creencias es verla como una representación de la distribución de probabilidad conjunta. Una red de creencias provee una descripción completa del dominio. Cada elemento en la distribución de probabilidad conjunta puede ser calculado con la información de la red. Un elemento genérico en la distribución de probabilidad conjunta es la probabilidad de una conjunción de asignaciones particulares a cada variable, tal que $P(X_1 = x_1 \wedge \ldots \wedge X_n = x_n)$. Usando $P(x_1,\ldots,x_n)$ como abreviación de esta notación, el valor de cada elemento esta dado por la siguiente fórmula:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | Padres(X_i))$$

Así, cada elemento en la distribución de probabilidad conjunta es representado por el producto de los elementos apropiados de las tablas de probabilidad condicional en la red de creencias. Las tablas de probabilidad condicional proveen de una representación descompuesta de la distribución de probabilidad conjunta.

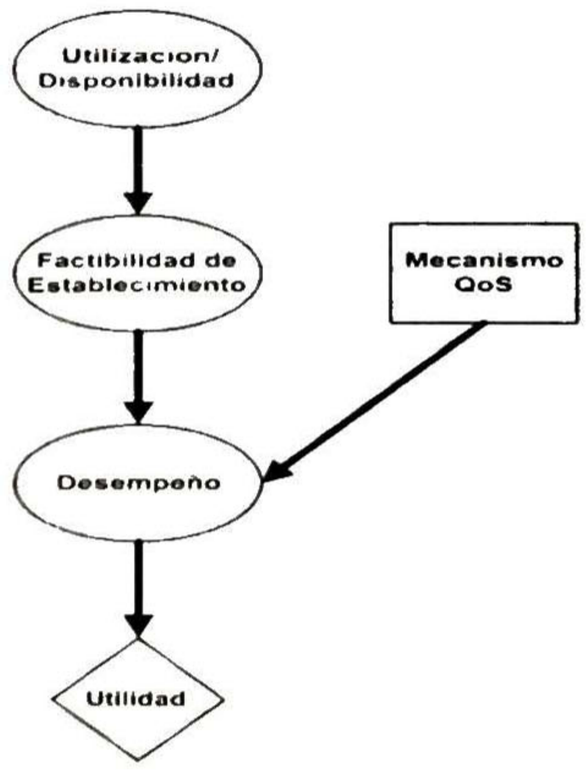


Figura 4.7 Red de decisión para el marco de trabajo QoE.

Agregando la función de utilidad a la red de creencias se forma la anterior red de decisión. Una red de decisión es un mecanismo general para hacer decisiones racionales. También se les llama diagramas de influencia. Las redes de decisión combinan redes de creencia con cierto tipo de nodos adicionales para acciones y utilidad. De manera más general, una red de decisión representa información acerca del estado actual del agente, sus posibles acciones, el estado que resultara por las acciones del agente, y la utilidad de ese estado. Por tal razón, las redes de decisión proveen una base para implementar agentes basados en utilidad. Las redes de decisión tienen tres tipos de nodos: Nodos de riesgo, que representan variables aleatorias, justo como en las redes de creencia. Cada nodo de riesgo tiene asociado con sigo una tabla de probabilidad condicional que esta indexada por los estados de los nodos padre. En las redes de decisión los nodos padre pueden ser nodos de decisión así como nodos de riesgo. Los nodos de riesgo se representan con óvalos. Los nodos de decisión representan puntos en donde el encargado de tomar la decisión tiene que elegir una acción. Este tipo de nodos se representa mediante un rectángulo. Los nodos de utilidad representan la función de utilidad del agente. El nodo de utilidad tiene como padres a todas aquellas variables que describen el estado resultante que afecta de manera directa a la utilidad. Este nodo se representa mediante rombos o diamantes. La tabla asociada con el nodo de utilidad representa la utilidad esperada asociada con cada acción, como se define en la ecuación de utilidad esperada. A estas tablas se les denomina tablas de utilidad-acción.

El algoritmo para calcular la máxima utilidad esperada MEU es el siguiente:

```
Para cada acción en el nodo de decisión
        Calcular el valor de utilidad esperada dada la acción y la evidencia
Regresar la máxima utilidad esperada y la acción
```

El proceso de QoS –QoE Management se puede resumir de la siguiente manera: Al comenzar la transmisión se establece la QoS inicial tomada de la Tabla de Referencia QoS cuyos valores provienen de la recomendación estándar Y.1541 de la ITU-T [54]. Después de iniciada la transmisión, se procede de manera concurrente a monitorear las métricas de desempeño en la transmisión y a escuchar las solicitudes que el usuario realiza a través del menú QoE además de las solicitudes de otros agentes. Las métricas son monitoreadas cada cierto intervalo de tiempo. Si no se cumple con las métricas establecidas en el SLA el *Monitoring Agent* le notifica al *Arbitrator Agent* para que este tome las medidas pertinentes. Cuando se recibe una solicitud de reducción de QoS se verifica que no sea una reducción a cero para evitar que se produzca una falta, siendo el caso debe de notificarse el error. Si se

trata de una solicitud de mejorar entonces se checa que haya un mínimo de recursos de manera que se pueda ofrecer el servicio, de otra manera se notifica el error. Si no ocurre error alguno, en ambos casos se procede a calcular la máxima utilidad esperada para de allí seleccionar el mecanismo QoS junto con su configuración. Ya hecha la elección se establecen los parámetros que producirán el nuevo nivel de QoS. Este procedimiento se repite hasta que la transmisión finalice.

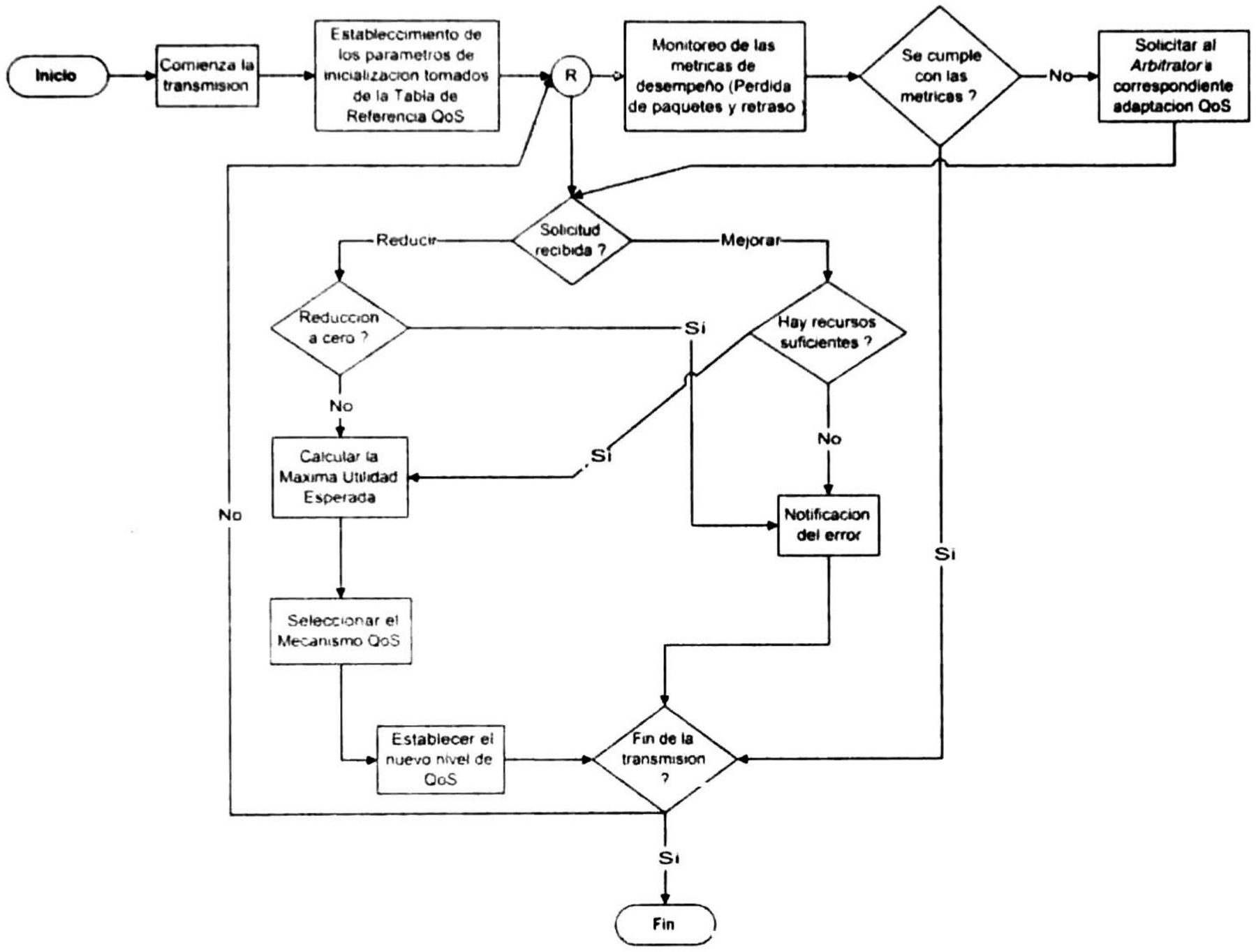


Figura 4.8 Ilustración del proceso de QoS Management.

Cualquier sistema de control se puede ver como un agente. [36]. Un sistema de control por retroalimentación se basa en un conjunto de variables de control relacionadas y un bucle de control por retroalimentación que continuamente monitorea el comportamiento del sistema controlado [55].

Como se puede apreciar, en la figura 4.9, el proceso de control es muy parecido a la estructura de un agente. Se desenvuelven dentro un ambiente, tienen percepciones del ambiente como entrada y como salida actúan sobre este después de haber computado en su interior la acción que llevaran a cabo. Semejante a un sistema de control, el proceso de QoS – QoE Management cuenta con un monitor, el *Monitoring Agent*, que se encarga de medir las métricas de desempeño de la red (estas son las variables

controladas y la red es el ambiente) contra las métricas acordadas en el SLA (la referencia). También cuenta con un controlador, el *Arbitrator Agent*, que con sus algoritmos y dependiendo de las solicitudes manifestadas (entradas al sistema) computa una acción a realizar (salida del sistema) sobre la configuración de los mecanismos de QoS (variable manipulada) en los protocolos de red (actuador). Además se cuenta con la retroalimentación del usuario que también funge como entrada del sistema. Por lo tanto se puede afirmar que se tiene un sistema de control de QoS – QoE por retroalimentación.

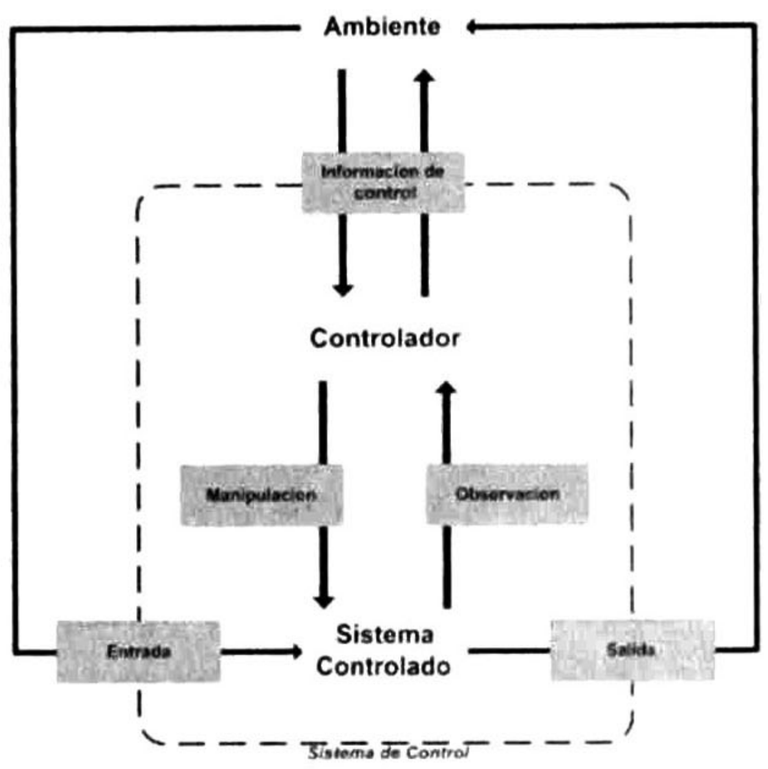


Figura 4.9 Diagrama de bloques del proceso de control.

4.5 Conclusiones

Se revisan de manera breve temas como el de agentes inteligentes y Teoría de la Decisión la cual está conformada por la Teoría de la Probabilidad y la Teoría de la Utilidad. En base a estos se propone el algoritmo para el proceso de QoS Management mediante agentes el cual sugiere una alternativa aun más racional que la de los trabajos previamente analizados. Con la utilidad se pretende satisfacer en lo más posible las necesidades del usuario con la menor cantidad de recursos posibles. Mientras que con las probabilidades se asignan grados de creencia para decidir que configuración de mecanismo QoS resultara más provechoso dadas las condiciones de la red. Se muestra el diagrama de decisión con el cual se calcula la Máxima Utilidad Esperada que sin duda arrojara la mejor acción a elegir para el *Arbitrator Agent*. También se justifica el control de QoS mediante control por retroalimentación. Las fases de una sesión multimedia y las funciones de QoS Management que intervienen en esta son revisadas para plantear la arquitectura del sistema, no sin antes analizar otros trabajos e incorporar características deseables que se observan de estos.

CAPITULO 5

5 Modelo formal del proceso de gestión de QoS -QoE

Ya que se ha mostrado el proceso de QoS Management es necesario plasmarlo de manera formal y que mejor metodología que las redes de Petri para el modelado de procesos. Es por ello que se presenta ahora un modelo de red de Petri como descripción formal del proceso de QoS Management.

5.1 Redes de Petri

Una red de Petri es un modelo matemático de un sistema con una representación gráfica cómoda y apropiada. Una red de Petri está compuesta de dos elementos: una *red*, y un *marcado inicial*. Una red es un grafo dirigido con dos tipos de nodos talque no existe algún arco entre dos nodos del mismo tipo, es decir, es un grafo bipartito. Los dos tipos de nodos son: *lugares*, y *transiciones*.

Los lugares se representan de manera grafica mediante un círculo, mientras que las transiciones son representadas con un cuadro o rectángulo. Los lugares pueden contener *tokens* los cuales se representan mediante puntos. Una distribución de tokens en los lugares de una red se le llama *marcado* y este corresponde al estado de la red de Petri. El marcado inicial corresponde a la distribución de tokens con que se inicializa la red. La transición de una red está *habilitada* en un marcado si todos su lugares de entrada, es decir, todos los lugar que tienen una arco apuntando a tal transición, contienen al menos un token. Una transición habilitada se puede disparar y tal disparo provoca un cambio en el marcado de la red de tal manera que se retira un token de cada lugar de entrada a la transición y se agrega un token a cada uno de sus lugares de salida. Esta *regla de disparo* trabaja para redes de Petri *ordinarias*, es decir, redes en las cuales el peso de todos sus arcos es igual a 1. Las redes de Petri generalizadas tienen diferentes pesos en sus arcos. En ellas la cantidad de marcas a retirar y agregar a los lugares de entrada y salida cuando se dispara una transición habilitada está determinada por el peso que indican sus respectivos arcos. De la misma manera la transición está habilitada solo cuando existen las marcas en los lugares de entrada que indican el peso de los arcos. Una red de Petri a la que se le han asociado a sus componentes un significado se le llama *red de Petri interpretada*. Una red de Petri interpretada corresponde al modelo de un sistema o un proceso. [56] y [57].

5.1.1 Definición de red de Petri y subclases de redes

Una red de Petri generalizada es una cuádrupla G = (P, T, Pre, Post), donde

- P es un conjunto finito de lugares ($|P| = n \geq 0$);
- T es un conjunto finito de transiciones ($|T| = m \geq 0$); $P \cap T = \emptyset$; $P \cup T \neq \emptyset$
- Pre: $P \times T \rightarrow \aleph$ es una función de incidencia previa la cual representa le pesos de los arcos de entrada a las transiciones;
- Post: $P \times T \rightarrow \aleph$ es una función de incidencia posterior la cual representa el peso de los arcos de salida de las transiciones.

Una red de Petri marcada se define como $G_M = (G,\mu)$; donde μ: $P \rightarrow \aleph$ es la función de marcado.

Las funciones de incidencia de una red de Petri pueden ser representadas mediante matrices de $n \times m$, a esta matriz se le llama *matriz de incidencia*. En ella cada elemento Pre (p_i, t_j) o Post (p_i, t_j) indica el peso del arco que relaciona p_i con t_j. Pre (t_j) indica un vector de la matriz que posee los pesos de los arcos que unen los lugares de entrada a t_j. El marcado es representado por un vector $[\mu(p_1)\ldots\mu(p_n)]^T$ donde $\mu(p_i)$ es el número de marcas en cada lugar p_i. Atendiendo a la regla de disparo se puede escribir la *ecuación de estado* de la red:

$$\mu_k = \mu_{k-1} + A\, v_k$$

donde A es la matriz de incidencia de la red $A = [a_{ij}]$; a_{ij} = Post (p_i, t_j) – Pre(p_i, t_j) y v_k es el vector de disparo, el cual representa las transiciones que se disparan en el instante *k*.

Mediante esta expresión se puede calcular el conjunto de marcados alcanzable desde el marcado inicial, al cual se le llama *conjunto de alcanzabilidad*. A partir de este se puede construir un grafo cuyos nodos son elementos del conjunto de alcanzabilidad y sus arcos representan las transiciones que al disparase ocasionan un cambio en el marcado. Este bosquejo se le llama *grafo de alcanzabilidad* de la red de Petri o grafo de estados asociado.

Existen diferentes subclases de redes de Petri que resultan de introducir restricciones estructurales en la red para dar solución a problemas en el análisis cuantitativo y cualitativo de los sistemas que modelan.

Una de estas subclases son las *máquinas de estados*, en las cuales cada transición tiene solo un lugar de entrada y un lugar de salida. Estas son útiles para modelar situaciones de elección y concurrencia, sin embargo, no permiten la sincronización de procesos.

Otra de estas subclases son los *grafos marcados*, los cuales en cada uno de sus lugares tiene solo una transición de entrada y una transición de salida. En estos es posible modelar configuraciones de sincronización y de distribución, pero no es posible modelar decisiones ni situaciones de conflicto.

Una subclase más son las redes de Petri de *libre elección*. En ellas, el resultado de la elección entre dos transiciones nunca debe ser influenciado por el resto del sistema, es decir, las elecciones son libres. Si existe un arco de un lugar p a una transición t, entonces sucede que t es la única transición de salida de p, lo que implica que t no puede estar en conflicto con otra transición, o p es el único lugar de entrada de t, lo que implica que no hay sincronización en t. De esta manera, siempre que una transición de salida del lugar p este habilitada, todas las transiciones de salida de p están habilitadas, y por lo tanto la elección que sucede en t es libre.

5.1.2 Propiedades y análisis de las redes de Petri

A continuación se mencionaran algunas de las propiedades que las redes de Petri poseen.

La primera de ellas es *vivacidad*. Una red de Petri es *viva* si cada una de sus transiciones siempre se puede disparar otra vez. De manera más precisa, si para cada *marcado alcanzable*, es decir, cada marcado que puede ser obtenido a partir del marcado inicial por el disparo sucesivo de transiciones, y cada transición t es posible alcanzar un marcado que habilita a t. Esta propiedad corresponde a sistemas en los cuales se puede alcanzar siempre todos los estados posibles modelados.

Otra propiedad es *libre de bloqueo*. Una red de Petri es libre de bloqueo si para cada marcado alcanzable se habilita alguna transición. Es propiedad es más débil que la vivacidad y corresponde a sistemas que garantizan alcanzar siempre algún estado y con ello que no se bloquearan.

Una red de Petri es *acotada* si existe un numero b tal que no hay un marcado alcanzable que

coloque más de *b* tokens en cualquier lugar. Si b = 1, es decir, la red es 1-acotada, entonces se trata de una red *a salvo*. Con frecuencia los lugares de una red de Petri se utilizan para modelar buffers y almacenamiento de datos. Si una red de Petri no es acotada el sistema que esta modela podría sufrir de un desbordamiento.

Un marcado es un *marcado de atracción* si este es alcanzable desde cada marcado alcanzable. Una red de Petri es *cíclica* si su marcado inicial es un marcado de atracción. Los sistemas que permanecen en un estado inicial hasta que algún usuario interactúa con ellos y después de esto regresan al mismo estado son característicos de esta propiedad.

Una red de Petri es *estrictamente conservativa* si y solo si para cualquier marcado alcanzable desde el marcado inicial, el numero de tokens de la red en ese marcado alcanzable es el mismo número de tokens en la red con el marcado inicial.

Una red de Petri es *conservativa respecto a un vector de ponderación* $w = [w_1\ w_2 \ldots w_n]$, $w_i \geq 0$ si y solo si para cualquier marcado alcanzable desde el marcado inicial, el numero de tokens en los lugares especificados por el vector w con el marcado alcanzable es el mismo número de tokens también en los lugares especificados por el vector w pero con el marcado inicial.

Una red de Petri es *repetitiva* si y solo si existe un marcado suficientemente grande y una secuencia de transiciones llamada secuencia de disparo cíclica tal que la secuencia aplicada desde dicho marcado regresa al mismo marcado y el vector de disparo *v*, construido a partir de la secuencia de disparo cíclica, no tiene elementos nulos. Si el vector de disparo tiene algún elemento nulo, lo que significa que algunas transiciones no se disparan en la secuencia de disparo cíclica, se dice que la red es *parcialmente repetitiva*.

Para el análisis de una red de Petri basta con resolver un conjunto de ecuaciones lineales simples, las cuales se obtienen a partir de la matriz de incidencia de la red de Petri en cuestión. La solución de tales ecuaciones se restringe a los enteros no negativos y lleva a particionar una red de Petri en componentes conservativas y repetitivas, de manera que es posible determinar propiedades estructurales las cuales son independientes del marcado inicial.

Sean x y y soluciones enteras positivas a los sistemas de ecuaciones:

$$x^{T} A = 0 \qquad A y = 0$$

el vector x de n × 1 se le llama *invariante de lugares* o *P-invariante*, y el vector y de m × 1 se le llama *invariante de transiciones* o *T-invariante*. A es la matriz de incidencia de la red de Petri. El *soporte* de un invariante es el conjunto de lugares o transiciones cuyos elementos correspondientes en un invariante son positivos.

Los soportes de P-invariantes de una red de Petri descomponen a esta en *componentes conservativas*. La suma de las marcas de los lugares indicados por un soporte permanece constante. Si cada uno de los lugares de una red de Petri pertenece a un soporte, es decir, hay un *P-invariante total*, entonces la red es estructuralmente acotada.

Los soportes de T-invariantes de una red de Petri descomponen a esta en *componentes repetitivas*. El disparo de las transiciones de un soporte a partir del marcado inicial, lleva a la red al mismo marcado, aunque en el no se especifica el orden de la secuencia. Si cada transición de una red de Petri pertenece a un soporte, es decir, hay un *T-invariante* total, entonces la red es repetitiva.

5.2 Modelo del proceso de gestión de QoS

Para la elaboración del modelo de red de Petri del proceso de gestión de QoS se utilizo la herramienta PIPE en su versión 2.4 [58]. PIPE es una herramienta *open source*, independiente de la plataforma en que se use, con la cual se construyen y analizan modelos de redes de Petri. PIPE está en constante desarrollo por un grupo del departamento de computación del Imperial College London.

En la red de Petri modelada se señalan las partes del proceso que corresponden a la funcionalidad de cada uno de los agentes que intervienen en el proceso de QoS Management. La inicialización de la transmisión y la solicitud en primera instancia de QoS se observan en la parte de la aplicación además del mapeo correspondiente de QoS. El monitor de la red que verifica las métricas de desempeño y notifica al arbitrador para que realice una adaptación de QoS. La retroalimentación del usuario que envía el agente desplegado en el host del cliente para la renegociación de QoS. La contabilización de QoS para su facturación. El mapeo de QoS, la negociación y asignación de recursos por parte del

arbitrador, en donde interviene el monitor de la red y el usuario mediante sus solicitudes.

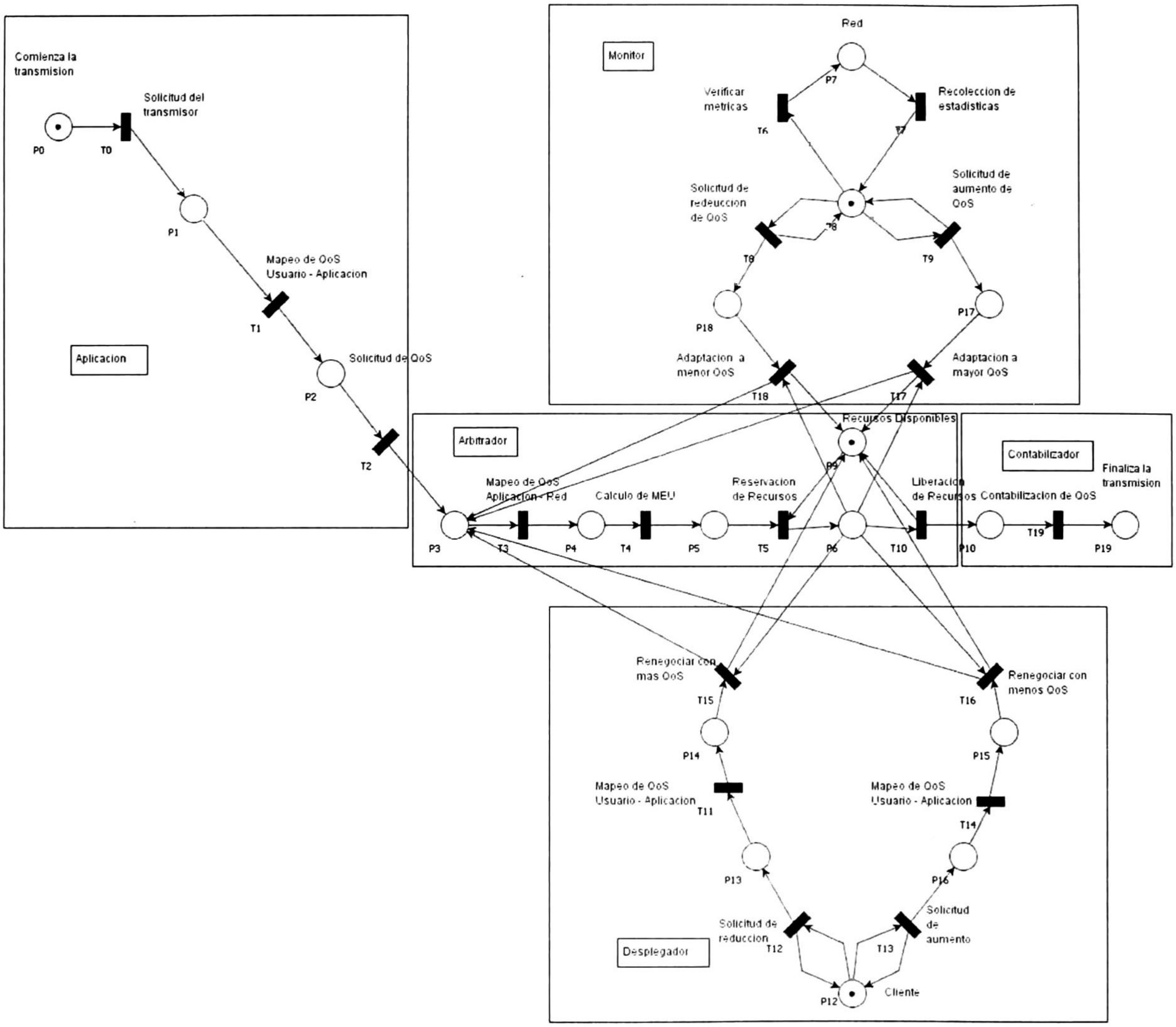


Figura 5.1 Modelo del proceso de QoS Management.

Petri net classification results

State Machine	false
Marked Graph	false
Free Choice Net	false
Extended Free Choice Net	false
Simple Net	true
Extended Simple Net	true

a

Petri net state space analysis results

Bounded	false
Safe	false
Deadlock	false

b

Petri net invariant analysis results

T-Invariants

0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
1 0 0 0 0
1 0 0 0 0
0 1 0 0 0
0 1 0 0 0
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 0
1 1 1 1 0
1 1 1 1 0
1 1 1 1 0
0 0 0 0 1
0 0 0 0 1
0 0 0 1 0
0 0 1 0 0
0 0 0 0 0

The net is not covered by positive T-Invariants, therefore we do not know if it is bounded and live

c

Petri net invariant analysis results

P-Invariants

1 0 0 0
1 0 0 0
1 0 0 0
0 1 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
1 0 0 0
1 0 0 0
1 0 0 0
1 0 0 0
1 0 0 1
0 0 1 0
0 0 1 0
0 0 0 1
1 0 0 0

The net is not covered by positive P-Invariants, therefore we do not know if it is bounded

d

Petri net invariant analysis results

P-Invariant equations

M(P0) + M(P1) + M(P10) + M(P2) + M(P3) + M(P4) + M(P5) + M(P6) + M(P19) = 1
M(P12) = 1
M(P7) + M(P8) = 1
M(P6) + M(P9) = 1

e

Forwards incidence matrix *f*

	T0	T1	T10	T11	T12	T13	T14	T15	T16	T17	T18	T2	T3	T4	T5	T6	T7	T8	T9	T19
P0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P10	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P12	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P13	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P14	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
P16	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
P2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P3	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
P5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
P9	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
P19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

f

Backwards incidence matrix *r*

	T0	T1	T10	T11	T12	T13	T14	T15	T16	T17	T18	T2	T3	T4	T5	T6	T7	T8	T9	T19
P0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
P12	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P13	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P14	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
P15	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
P16	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
P17	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
P18	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P6	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
P7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
P9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
P19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

g

Combined incidence matrix *I*

	T0	T1	T10	T11	T12	T13	T14	T15	T16	T17	T18	T2	T3	T4	T5	T6	T7	T8	T9	T19
P0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P1	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P10	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
P12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P13	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P14	0	0	0	1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0
P15	0	0	0	0	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0
P16	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
P17	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	1	0
P18	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0
P2	0	1	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0
P5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0
P6	0	0	-1	0	0	0	0	-1	-1	-1	-1	0	0	0	1	0	0	0	0	0
P7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0
P8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0
P9	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	-1	0	0
P19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

h

Marking

	P0	P1	P10	P12	P13	P14	P15	P16	P17	P18	P2	P3	P4	P5	P6	P7	P8	P9	P19
Initial	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
Current	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Enabled transitions

T0	T1	T10	T11	T12	T13	T14	T15	T16	T17	T18	T2	T3	T4	T5	T6	T7	T8	T9	T19
yes	no	no	no	yes	yes	no	no	no	no	no	no	no	no	no	yes	no	yes	yes	no

i

Figura 5.2 Resultados del análisis del modelo de la red de Petri del proceso de QoS Management.

Después de analizar la red de Petri del modelo de QoS Management con la herramienta PIPE, los resultados que arroja son los siguientes. No pertenece a las clases de Maquina de estados, Grafo Marcado, Red de Libre Elección y Red de Libre Elección Extendida. Sin embargo pertenece a las clases menos restrictivas que son Red Simple y Red Simple Extendida, figura 5.3 (a). No cumple con las características de ser Acotada ni A Salvo pero sí es Libre de Bloqueo, figura 5.3 (b). Se obtuvieron cinco T-invariantes, todos ellos semipositivos por lo cual no se puede asegurar que la red sea viva y acotada. A pesar de eso, con ellos se pueden identificar las componentes cíclicas de la red, figura 5.3 (c). Se obtienen cuatro P-invariantes, todos ellos también semipositivos por lo que no se puede afirmar que es una red acotada. Sin embargo a partir de ellos se pueden identificar sus componentes conservativas, figura 5.3 (d). También de este análisis se obtienen las ecuaciones de los P-invariantes, las cuales indican que las componentes conservativas inducidas por los P-invariantes son 1-acotadas, figura 5.3 (e). Tanto la matriz de incidencia, como la matriz de incidencia invertida y la matriz de incidencia combinada, que se forma a partir de las otras dos, se obtienen del análisis de incidencia, figura 5.3 (f), (g) y (h). Finalmente se muestran tanto el marcado inicial como el actual y las transiciones que están habilitadas bajo este marcado, figura 5.3 (i).

5.3. Conclusiones

Se presenta un breve resumen acerca de teoría de redes de Petri, en donde se menciona su definición, clases más comunes, algunas de sus propiedades y análisis de estas. Se presenta el modelo formal del proceso de gestión de QoS - QoE mediante redes de Petri y se analizan sus propiedades. De las propiedades a destacar está Libre de Bloqueo la cual nos indica que el proceso siempre terminara y este no caerá en un estado no deseado. No es acotada ya que puede recibir una cantidad infinita de solicitudes durante la transmisión que hará que el proceso de renegociación se realice el mismo número de veces. Para que sea cíclica solo se necesita agregar la transición y los arcos del final al inicio de la transmisión que indiquen que se realiza una nueva transmisión.

CAPITULO 6

6 Implementación y experimentos del marco de trabajo QoE

A continuación se describen tanto la implementación como la serie de experimentos realizados durante el desarrollo de este proyecto de tesis, además del método de evaluación con el que serán calificados y los resultados obtenidos.

6.1 Implementación y escenario de experimentación

La plataforma basada en agentes bajo el marco de trabajo de QoE de [21] se tomo como base para la implementación de esta tesis. La plataforma corre sobre una red habilitada con QoS. Dentro de la plataforma un agente *arbitrator* es el encargado del proceso de gestión de QoS. Esto incluye procesar las solicitudes de retroalimentación del usuario, monitoreo de la red, verificación de las tablas de referencia y cálculo de la MEU para la selección del mecanismo QoS y establecer su configuración. Otro agente, llamado agente *application*, recibe las solicitudes del usuario y se las envía al *arbitrator*. Cuando ocurre una solicitud de reducción, se disminuye la QoS y por lo tanto se liberan recursos que quedan disponibles para otros usuarios. Dentro de este marco de trabajo el usuario puede hacer las solicitudes que desee. La tabla de referencia de QoS provee las métricas de desempeño objetivas de la red. Dado que en el punto de inicio del proceso de gestión de QoS no existe retroalimentación del usuario los parámetros de configuración inicial de la red son tomados de la tabla de referencia de QoS. Bajo estas condiciones se consideran como óptimos los valores para el mapeo de QoS a QoE.

¿Por qué Jade Java? La arquitectura que se ha descrito previamente requiere de los mecanismos adecuados para permitir la migración y ejecución de los agentes. Además de que estos deben ser propiamente ejecutados sin importar el hardware y el software de la plataforma del nodo de red. Por tales motivos se ha seleccionado a Jade [59] el cual está hecho en Java [60] para desarrollar la plataforma. Este ofrece todos los mecanismos básicos para la comunicación y la sincronización entre procesos y garantiza la portabilidad del código sobre múltiples arquitecturas y sistemas operativos de manera eficiente y segura. Se provee de todos los mecanismos necesarios para crear, correr, suspender, despertar, desactivar y activar agentes, además de permitirles comunicarse y migrar a través de la red.

La red con los protocolos de QoS habilitados está compuesta por una serie de computadoras Linux corriendo Fedora Core 5 con el software de ruteo Zebra (Quagga). Se utiliza el protocolo BGP (*Border Gateway Protocol*) como protocolo de ruteo. Los ruteadores se sincronizan mediante el protocolo SNTP (*Simple Network Time Protocol*). Los mecanismos de QoS son provistos por Linux. La red está configurada sobre Ethernet 10 BaseT. La red de investigación experimental está dividida en tres sistemas autónomos. Uno de ellos hace la función de ruteador *core* y los otros dos realizan la función de ruteadores *edge*. Del *edge1* penden el transmisor de video y el emisor de ruido. El ruido se produce con el generador de tráfico MGEN [61]. El transmisor de video cuenta con el kernel de tiempo real KURT [62] el cual permite una granularidad más fina, en el orden de decenas de microsegundos, en cuanto al tiempo de calendarización de procesos, logrando con esto una autentica transmisión de video en tiempo real. Del *edge2* pende el receptor de video que es una PC convencional. La siguiente figura muestra el escenario experimental.

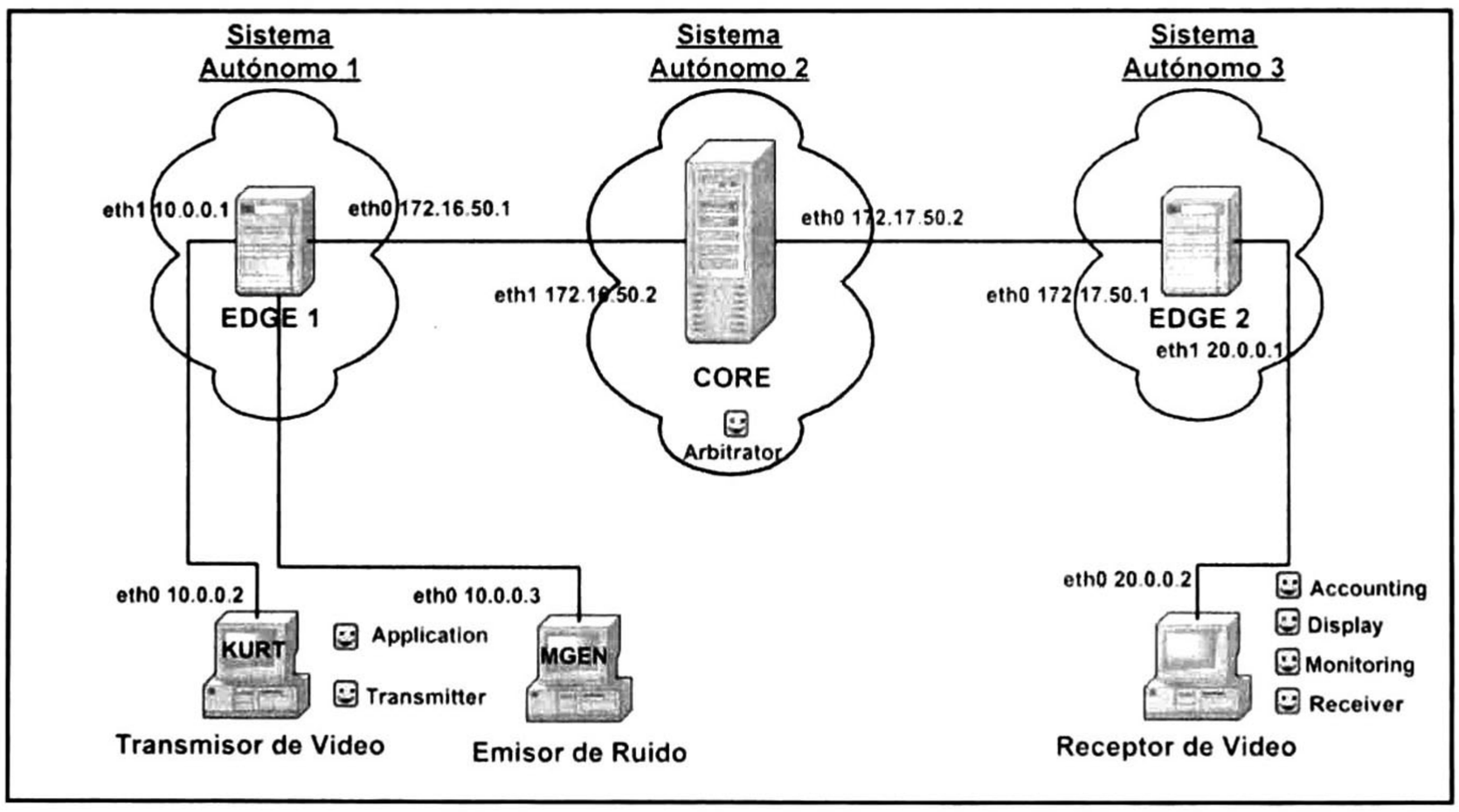


Figura 6.1 Escenario de experimentación del marco de trabajo QoE.

6.1.1 Métodos de evaluación subjetiva de calidad de Imagen

Varias metodologías para la evaluación subjetiva de calidad de imagen han sido desarrolladas y validadas. Entre ellas se encuentran DSIS (*Double Stimulus Impairment Scale*) [63], en donde a un grupo de sujetos se les pide calificar la reducción de calidad de una imagen recién procesada con respecto a la imagen original la cual no ha reducido su calidad. DSCQS (*Double Stimulus Continuous Quality Scale*) [63], en ella no se sabe con qué orden se presentan las imágenes por lo que los sujetos

no saben si es una imagen ya procesada o no. Los sujetos entonces proceden a dar una calificación en la escala del 1 al 100 en cuyos intervalos de 20 puntos tienen diferente significado. 1 -20 = malo, 21 – 40 = pobre, 41 – 60 = adecuado, 61 – 80 = bueno y 81 – 100 = excelente. Las imágenes se muestran a los sujetos por aproximadamente diez segundos y el promedio de las calificaciones de los sujetos resulta en el MOS (*Mean Opinion Score*), la cual es una métrica de la calidad de imagen y video. Se necesitan al menos de 20 a 25 visores no expertos para dar una métrica MOS confiable.

Estos métodos se utilizan normalmente en la evaluación de calidad de imágenes estáticas. Para la evaluación de video es preferido SSCQE (*Single Stimulus Continuous Quality Evaluation*) [63] en donde la calidad de la imagen se evalúa conforme varía el tiempo. En este método se le pide al grupo de sujetos evaluar continuamente la calidad de imagen de un conjunto de escenas de video. El criterio de evaluación es el mismo que en DSCQS. Dado que las secuencias de video pueden ser largas, se segmentan en partes de diez segundos, y se calcula una MOS para cada segmento del video.

6.2 Descripción del experimento

Objetivo: Medir la Calidad de Experiencia del usuario final usando la plataforma Basada en Agentes. Los parámetros que controlan la calidad son manejados por un agente arbitrador el cual utiliza algunas tablas de referencia, el cálculo de la MEU (*Maximum Expected Utility*) y la retroalimentación del usuario principalmente.

Descripción General: Se le pidió a un grupo de veinte personas que participaran como receptores durante una transmisión de video. A cada usuario se le explico que estaría utilizando el servicio de QoE y que la interacción con este es a través de un simple menú. En el menú se le permite al usuario aumentar o reducir la calidad de la transmisión y muestra la tarifa asociada al servicio. También cuenta con una barra de desplazamiento que sirve para calificar la transmisión percibida conforme al SSCQE. La barra no cuenta con alguna etiqueta o escala salvo los signos + y – a los extremos de esta indicando que mientras más se acerque a estos signos la calificación otorgada será mayor o menor según sea el caso. De esta manera se emula una escala continua de calificaciones y se evita que el usuario otorgue alguna calificación influenciado por las etiquetas o escalas que en ella pudieran aparecer. Con estas calificaciones se obtuvo una métrica basada en el MOS para medir de cierta manera la Calidad de

Experiencia que el usuario percibió. A cada usuario se le dio una oportunidad para practicar y familiarizarse con el menú QoE, figura 6.2(a), y el procedimiento que involucra.

Figura 6.2 QoE Menú (a), Secuencia de video experimental (b), Secuencia de práctica (c).

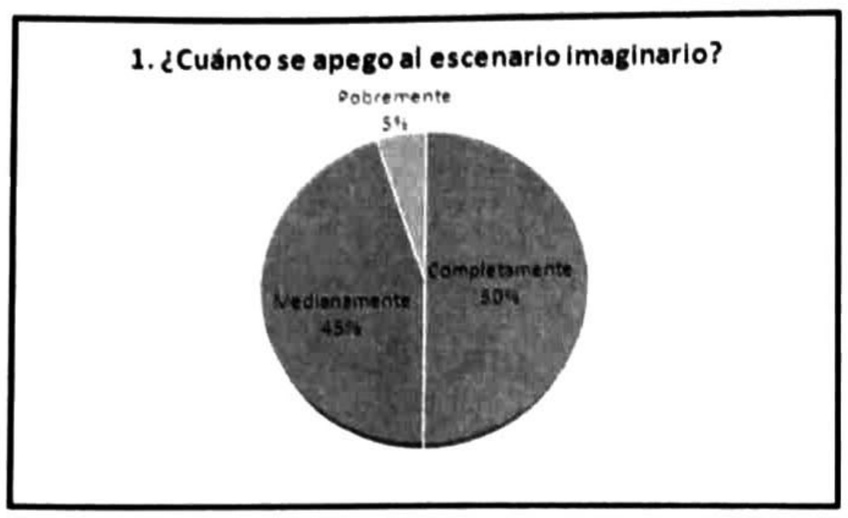
Se utilizaron dos secuencias de video para los experimentos las cuales se muestran en la figura 6.2 (b) y (c). Las secuencias son de tipo CIF, codificadas a 30 cuadros por segundo y cicladas de manera apropiada para lograr secuencias de 4 minutos de duración.

Durante las sesiones se les pidió a los usuarios imaginarse en un *escenario* que les diera el contexto para establecer una relación entre la secuencia de video que ve y la experiencia que percibe. La mitad de los usuarios imaginaron un escenario de negocios en el cual la transmisión involucro un contrato de por medio, mientras que la otra mitad imagino un escenario recreativo en el cual la transmisión simplemente implico diversión.

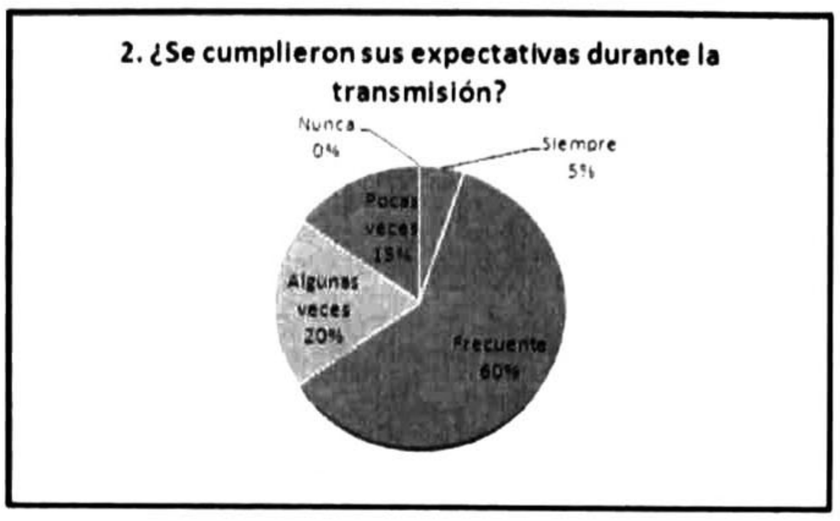
Después de todo esto se les pidió contestar a un pequeño cuestionario. Este tiene el objetivo de analizar factores como el desempeño de la plataforma, interés y contenido del video, tarifas y costo, factores en la decisión del aumento/reducción de calidad, y posibles aplicaciones del servicio.

6.3 Análisis y Resultados

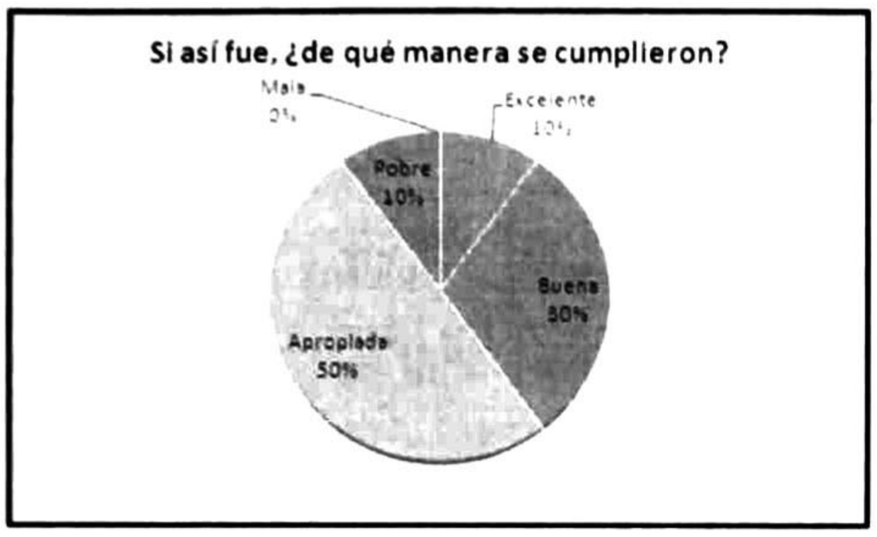
En la primera pregunta, figura 6.3 (a), se cuestiona al usuario que tanto se apego este al escenario imaginario. Esto permite saber la habilidad que mostro el usuario para asumir el rol del escenario imaginario y da un indicador de significancia para la respuesta a la pregunta 3. El 50% de los usuarios se aplicaron *completamente* al escenario imaginario y el 45% se aplicaron *medianamente*. Solo uno de los usuarios no se aplico del todo al escenario asignado.



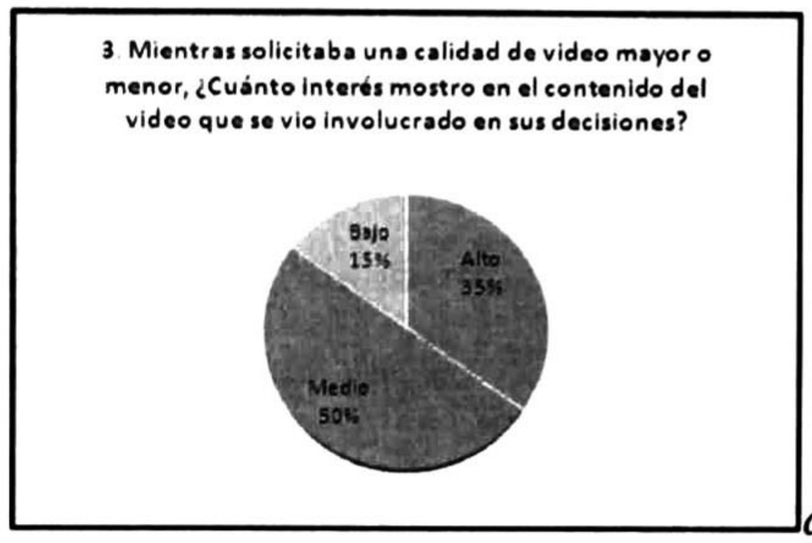
a



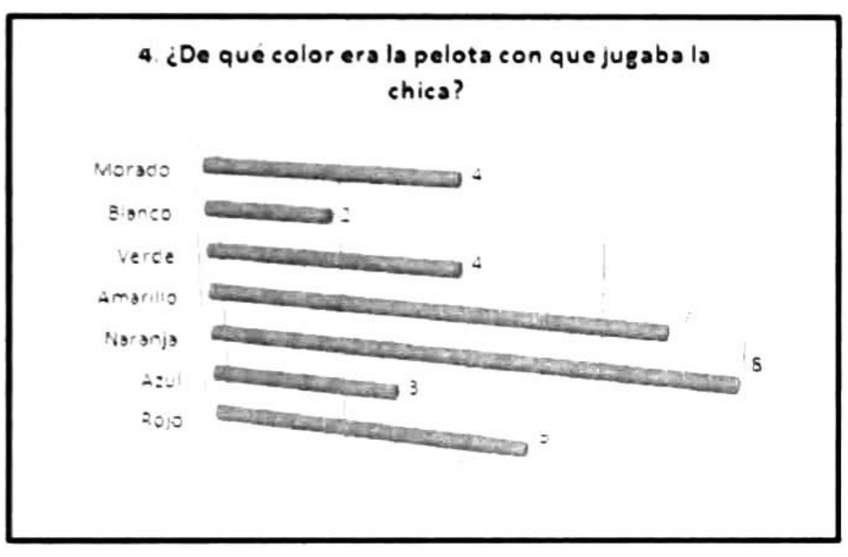
b



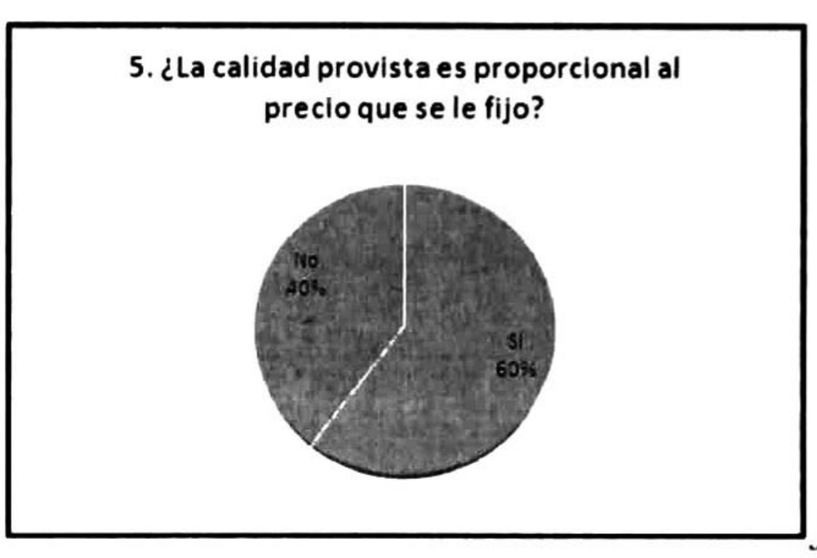
c



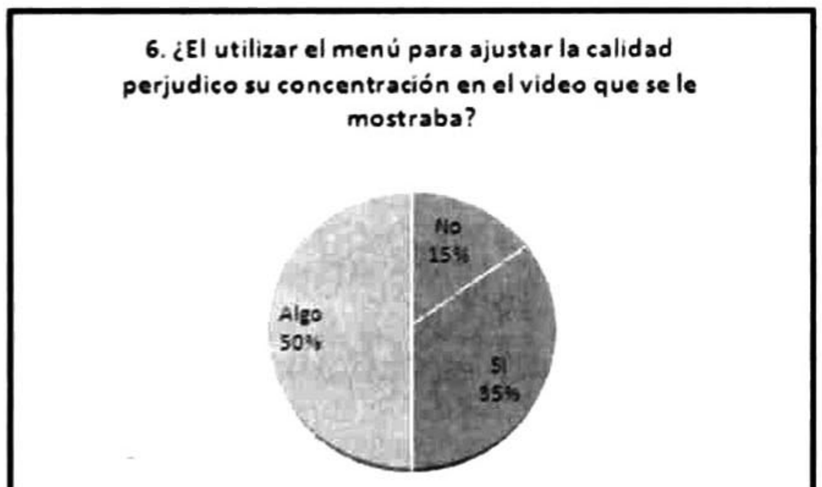
d



e

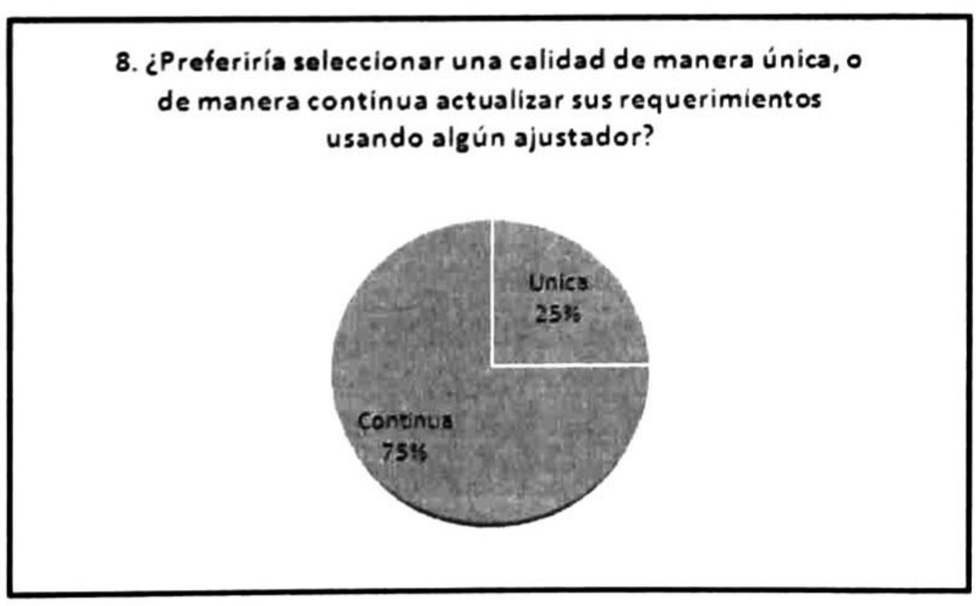


f

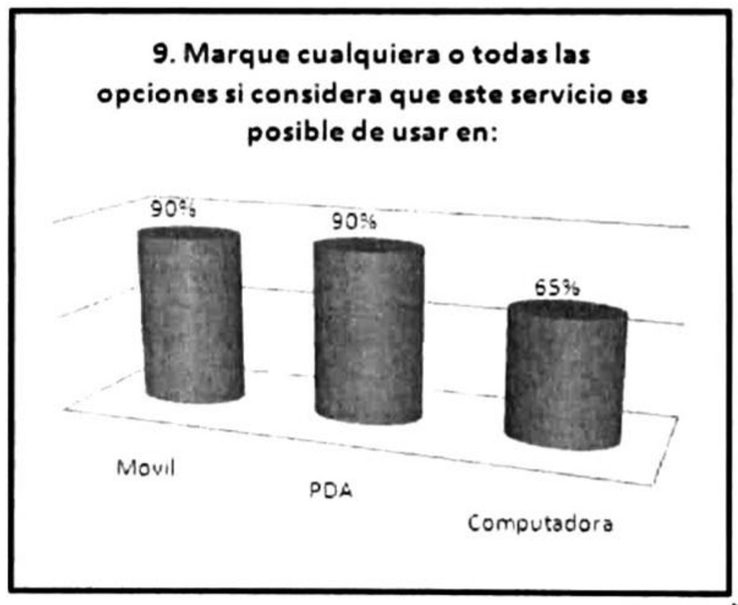


g

h



i



j

Figura 6.3 Graficas representativas de las respuestas obtenidas en el cuestionario.

La pregunta numero 2, figura 6.3 (b), cuestiona de manera genérica las expectativas del usuario con respecto a la transmisión y muestra una distribución de la satisfacción del usuario. El 60% de los usuarios vieron sus expectativas llenas de manera frecuente, el 20% algunas veces, el 15% pocas veces, y solo uno de los usuarios se vio satisfecho en todo momento. Ninguno de ellos expreso que nunca se cumplieron sus expectativas. Esto muestra una tendencia con mayor nivel de satisfacción aprobatorio. Complementando esta pregunta, se cuestiono al usuario de qué manera se vieron cumplidas sus expectativas cuando estas lo hicieron, figura 6.3 (c). A esto el 50% manifestó que fue de manera apropiada, 30% de manera buena y 10% de manera excelente. El 10% restante indico que sus expectativas se cumplieron de manera pobre. Cabe destacar que estos usuarios forman parte de aquellos que vieron sus expectativas pocas veces cumplidas. Ninguno replico que se hubiera cumplido de mala manera.

La tercera pregunta, figura 6.3 (d), cuestiona la influencia que tuvo el interés mostrado por la transmisión sobre las decisiones acerca de la calidad. El 35% mostro un interés alto mientras que el 50% mostro un interés medio. El resto no mostro mucho interés en la transmisión. Esto indica que la mayoría de los encuestados estuvieron atentos e interpretaron de manera confiable su rol dentro del escenario imaginario.

La pregunta numero 4, figura 6.3 (e), se enfoca en la observación del usuario al preguntar el color de la bola con la que jugaba la chica de la secuencia de video. Solo 3 contestaron correctamente los dos colores de la bola que son naranja y morado. Otros se acercaron mucho a los colores contestando amarillo, morado, naranja, azul, y rojo que son colores muy cercanos a los originales. Esta pregunta puede servir para corroborar el interés que el usuario mostro en realidad, pregunta número 3.

La pregunta numero 5, figura 6.3 (f), cuestiona a los usuarios si la calidad de la transmisión provista es proporcional al precio que se fija, a lo cual el 60% estuvieron de acuerdo. Los usuarios se mostraron interesados en probar la relación entre calidad y precio según lo percibido.

Con la pregunta 6, figura 6.3 (g), se trata de evaluar el efecto que causa utilizar el menú QoE durante la transmisión a la concentración del usuario. Solo el 15% manifestó no sufrir algún impacto al utilizarlo, el 50% considero que afecto un poco y el 35% señalo que sí causa una distracción utilizar el

menú. Cabe señalar que la mayoría de los usuarios que manifestaron alguna distracción propusieron la utilización del teclado (flechas del cursor) para controlar el aumento y la reducción de la calidad.

La pregunta 7, figura 6.3 (h), trata de posicionar los factores que afectaron en la toma de decisión de calidad del usuario, que son: contenido, precio y claridad imagen. Los tres obtuvieron resultados semejantes aunque el de mayor preferencia fue claridad de imagen, seguido por contenido interesante y precio.

En la pregunta numero 8, figura 6.3 (i), se cuestiona la preferencia del usuario por establecer una calidad de manera fija al principio de la transmisión o si este prefiere de manera continua ir actualizando sus requerimientos de calidad durante la transmisión. El 75% de los usuarios manifestaron su preferencia por actualizar de manera continua la calidad mientras que el resto solo prefiere una calidad fija.

La pregunta 9, figura 6.3 (j), cuestiona a los usuarios en donde creen que este tipo de servicio pueda ser implantado. A lo que la mayoría respondieron que se podría utilizar en dispositivos móviles así como en PDAs, y solo algunos de ellos respondió que también podría utilizarse en computadoras personales.

La pregunta 10 solo intenta obtener del usuario que es lo que entendió sobre Calidad de Experiencia una vez que finalizo el experimento. La mayoría expreso algo como "*la satisfacción que el usuario experimenta en cuanto la calidad del video que percibe, en relación a lo que paga*".

Durante el experimento, a través de la evaluación de los usuarios se obtuvieron los valores con los cuales se califico la calidad de la transmisión conforme sus expectativas. El 60% produjo una calificación de *adecuado*, el 30% produjo una calificación de *bueno* y solo el 10% produjo una calificación de *pobre*. En ninguno de los experimentos se obtuvo una calificación de *excelente* o *malo*. Promediando todas las calificaciones se obtiene un valor de MOS de **55.60** el cual se encuentra dentro del rango de *adecuado*, por lo cual se puede concluir que la calidad en la transmisión que percibió el usuario conforme a sus expectativas, es decir la QoE, fue la *adecuada*. Cabe señalar que el valor de MOS obtenido se encuentra muy cercano al correspondiente a *bueno*.

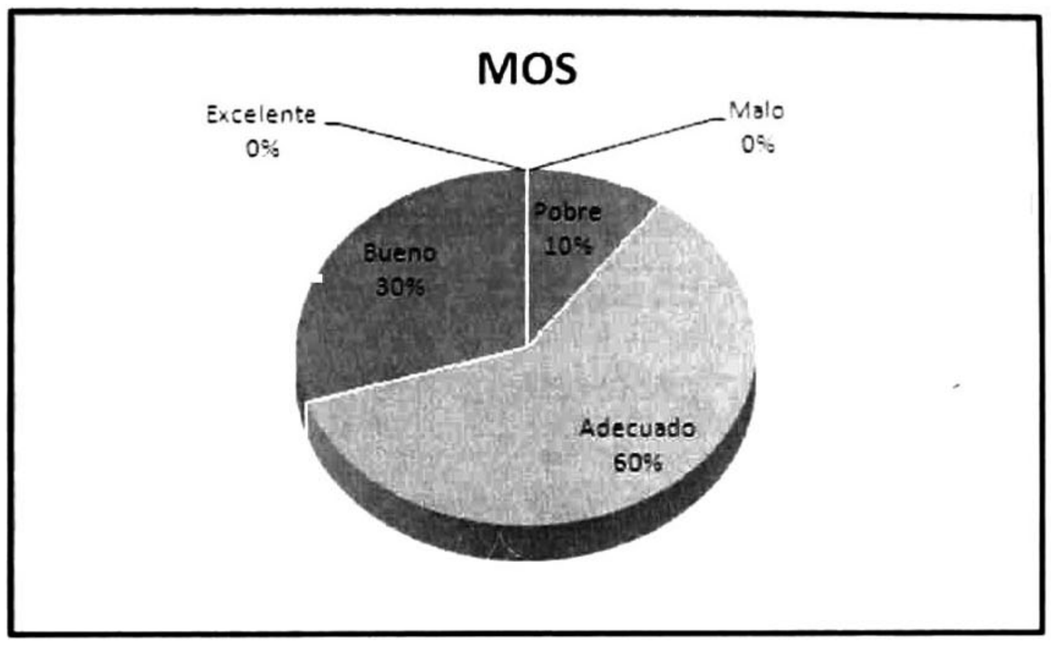


Figura 6.4 MOS obtenido en el SSCQE.

Los requerimientos de QoE varían de un usuario a otro incluso bajo el mismo escenario. Por tal motivo se cree que facultando al usuario para seleccionar la calidad acorde a sus expectativas, la QoE puede mejorar en relación al individuo. El usuario solo paga de acuerdo a sus requerimientos y los recursos que estos involucran en la red. Cuando el usuario reduce la calidad se liberan recursos para que estén disponibles para otros usuarios. Tanto el proveedor del servicio como el usuario se benefician con este enfoque.

Sujeto	Escenario	**cfactor**	Valor de **cfactor**	**Wdelay**	**Adelay**	**Wjitter**	**Ajitter**	**Wloss**	**Aloss**	**NQoS**	**NQoS** Norma-lizada
1	Recreación	Mejora / Reduce = 66/87	0.7600	2.33E-05	1.0	2.33E-05	1.0	4765.7762	0.0005	1.2697	0.5000
2	Negocio	Mejora / Reduce = 30/6	5.0000	7.95E-25	1.0	7.95E-25	1.0	1.40E+23	0.0717	5.01E+21	0.5000
3	Negocio	Mejora / Reduce = 141/64	2.2031	1.04E-120	1.0	1.04E-120	1.0	1.07E+119	0.1103	5.90E+117	0.5000
4	Recreación	Reduce / Mejora = 86/171	0.5029	1.57E-141	1.0	1.57E-141	1.0	7.07E+139	0.0161	5.70E+137	0.5000
5	Recreación	Mejora / Reduce = 42/1	42.000	2.85E-50	1.0	2.85E-50	1.0	3.90E+48	0.3587	7.00E+47	0.5000
6	Recreación	Reduce / Mejora = 59/86	0.6860	6.29E-92	1.0	6.29E-92	1.0	1.77E+90	0.0377	3.34E+88	0.5000
7	Negocio	Mejora / Reduce = 94/105	0.8952	2.34E-76	1.0	2.34E-76	1.0	4.75E+74	0.0623	1.48E+73	0.5000
8	Negocio	Mejora / Reduce = 14/15	0.9333	0.0175	1.0	0.0175	1.0	6.3211	0 0844	0.2843	0.2843
9	Negocio	Mejora / Reduce = 29/8	3.6250	5.51E-19	1.0	5.51E-19	1.0	2.02E+17	0.1434	1.45E+16	0.5000
10	Recreación	Mejora / Reduce =	2.0666	1.00E-48	1.0	1.00E-48	1.0	1.11E+47	0.0398	2.21E+45	0.5000

		62/30									
11	Negocio	Mejora / Reduce = 10/1	10.000	9.19E-08	1.0	9.19E-08	1.0	1209600	0.4783	289295	0.5000
12	Recreación	Mejora / Reduce = 63/13	4.8461	1.03E-52	1.0	1.03E-52	1.0	1.08E+51	0.2391	1.29E+50	0.5000
13	Negocio	Mejora / Reduce = 47/10	4.7000	3.40E-21	1.0	3.40E-21	1.0	3.27E+19	0.1793	2.93E+18	0.5000
14	Recreación	Reduce / Mejora = 76/131	0.5801	4.63E-161	1.0	4.63E-161	1.0	2.40E+159	0.0333	4.00E+157	0.5000
15	Negocio	Mejora / Reduce = 105/17	6.1764	3.30E-117	1.0	3.30E-117	1.0	3.37E+115	0.0573	9.67E+113	0.5000
16	Recreación	Mejora / Reduce = 37/26	1.4230	2.06E-13	1.0	2.06E-13	1.0	5.40E+11	0.0551	1.4897	0.5000
17	Negocio	Reduce / Mejora = 39/65	0.6000	7.08E-14	1.0	7.08E-14	1.0	1.57E+12	0.0010	7.98E+08	0.5000
18	Recreación	Mejora / Reduce = 83/1	83.000	6.08E-123	1.0	6.08E-123	1.0	1.83E+121	0.1024	9.36E+119	0.5000
19	Negocio	Reduce / Mejora = 31/23	1.3478	0.055	1.0	0.055	1.0	2.018	0.0434	0.0989	0.0989
20	Recreación	Mejora / Reduce = 59/64	0.9218	4.29E-11	1.0	4.29E-11	1.0	2.59E+09	0.1304	1.69E+08	0.5000
		Promedio	**8.6133**	**0.0036**	**1.0**	**0.0036**	**1.0**	**1.1990E+158**	**0.1122**	**2.0007E+156**	**0.4692**

Tabla 6.1 Mapeo de QoS a QoE.

Para cada usuario también se registro el historial de solicitudes el cual se analiza junto con algunos parámetros de QoS tal y como se explica en [21], para obtener un mapeo de QoS a QoE. De estos resultados se puede observar lo siguiente:

- Los valores de Ajitter y Adelay permanecen en 1.0. Esto se debe a que las mediciones hechas por el agente monitor promedian un retraso y un jitter el cual cumple con los niveles de referencia de la ITU-T tomados de la tabla de referencia QoS. Esto no ocurre en los valores de Aloss debido a que sí se registran violaciones de la métrica.
- Los valores Wdelay y Wjitter siempre terminan con los mismos valores de manera similar como lo hacen Ajitter y Adelay. La razón es que al no ser Ajitter y Adelay cualquiera de estas la métrica más baja debido a que no se violan los valores de referencia como se menciono en el punto anterior, Wdelay y Wjitter siempre se modifican con la misma proporción, lo cual no pasa con Wloss.
- La mayoría de los usuarios obtuvo el máximo valor de NQoS después de ser normalizada. También todos ellos rebasaron por mucho los valores de la métrica previa a la normalización y esto se debe al valor del cfactor. El valor de cfactor incrementa bastante si la tendencia a

aumentar o reducir la calidad es grande, es decir, se reciben de manera continua una gran cantidad de solicitudes del mismo tipo sin recibir alguna del tipo opuesto. Es también por esto que se observan cantidades extremas en los valores de las métricas subjetivas. En estos casos lo que se puede inferir es que el usuario probo con la plataforma cuales eran los límites de calidad que este servicio le puede ofrecer. También se puede inferir que el usuario se posiciono rápidamente en la calidad y el precio deseados sin necesidad de solicitar cambios posteriores, por lo que no se registraron solicitudes del tipo opuesto. El valor máximo de NQoS indica que los mecanismos de QoS proveen al usuario la QoE deseada.

- El caso de los usuarios 8 y 19 difiere del resto. Ellos muestran una cantidad de solicitudes de mejorar y reducir semejantes durante toda la transmisión además de que no se cumplió con la métrica de referencia de pérdida de paquetes. Esto propicia que los valores de las métricas subjetivas sean pequeños y no se tenga certeza de la QoE del usuario. Por lo tanto se puede presumir que estos usuarios al no haber llegado al máximo de NQoS, la QoE que experimentaron no fue del todo satisfactoria.
- La variable cantidad de solicitudes de mejorar y reducir por parte de los usuarios incluso bajo el mismo escenario, demuestran la subjetividad de este marco de teórico. Este mapeo de QoS a QoE se muestra útil para entender la relación entre las métricas objetivas y subjetivas durante los experimentos. De esta manera se puede obtener un panorama de la red y los requerimientos de QoE durante una transmisión multimedia.

6.4 Conclusiones

Plataforma QoE	MOS	Valor de cfactor	Wdelay	Adelay	Wjitter	Ajitter	Wloss	Aloss	NQoS	NQoS Normalizada
Agentes reflexivos	N/A	1.8513	2.7308	1.0	2.7308	1.0	35.1835	0.1945	18.2067	0.3639
Agentes y utilidades	55.5996	8.6133	0.0036	1.0	0.0036	1.0	1.1990E+158	0.1122	2.0007E+156	0.4692

Tabla 6.2 Comparativa de resultados.

Comparando los resultados obtenidos con los de [21], tabla 6.2, se puede concluir lo siguiente:

- Se observa un incremento considerable en el valor de cfactor. Este fue causado por proveer a los usuarios de un menú QoE más interactivo al incluir botones accionados por clics con el ratón. Esto motivo a que la interacción entre la plataforma y el usuario se disparara, y como

consecuencia hubo tendencias muy marcadas en cuanto a las solicitudes del usuario, mismas que se pueden observar en los valores de dicho parámetro.

- En cuanto a las métricas objetivas: Adelay, Ajitter y Aloss, no se observo cambio alguno en las dos primeras debido a que los valores en sus métricas correspondientes nunca rebasaron el valor de referencia de la tabla de referencia QoS. Sin embargo se obtuvo un valor más pequeño en Aloss, lo cual indica que ocurrió una mayor pérdida de paquetes.
- Por otra parte los valores de las métricas subjetivas se dispararon considerablemente debido a la influencia que el parámetro cfactor produce durante el cálculo de estas métricas, esto a consecuencia de la tendencia de los usuarios con sus solicitudes.
- También como consecuencia del cfactor y las métricas subjetivas, la NQoS se disparo. Sin embargo con los valores ya normalizados se observa un aumento de esta en prácticamente una decima. Lo cual indica que los mecanismos de QoS proveyeron en mayor cantidad la QoE deseada al usuario, presumiblemente por la inclusión de nuevos mecanismos de QoS y la forma racional de gestionar los recursos de la red por parte de los agentes.
- Adicionalmente se incluyo una prueba mediante SSCQE la cual arrojo como resultado un valor de MOS de 55.60. Este corresponde a una calificación de *apropiado* y muestra que la calidad de la transmisión percibida por el usuario fue solamente la *apropiada*. Se estuvo muy cerca de lograr una calificación de *buena*.

Conclusiones y trabajo futuro

En esta tesis se presenta un marco de trabajo para la gestión de QoS tomando en cuenta la percepción del usuario (*QoE*). Se mencionan los conceptos fundamentales de QoS y los mecanismos con los cuales se puede proveer de esta. Entre ellos se encuentran Servicios Diferenciados, Servicios Integrados y Multiprotocolo por Conmutación de Etiquetas. También se mostraron simulaciones implementadas en NS2 de cada mecanismo de QoS. En ellas se pudo corroborar tanto el funcionamiento como las ventajas que los mecanismos de QoS proveen sobre otros tipos de servicio como el de *best-effort*.

Se dieron a conocer diferentes herramientas ya existentes para la implementación de una red experimental *open source* que cuente con los mecanismos de QoS habilitados. Se partió desde los protocolos de ruteo básico, como BGP el cual puede ser implementado con el software de ruteo Zebra. Se dio a conocer el paquete Iproute2 con el cual se pueden implementar estrategias de ruteo avanzado, desde marcado de paquetes (*DiffServ*) hasta reserva de ancho de banda (*IntServ*). También se mostro MPLS-Linux, una herramienta con la cual se puede implementar ruteo por conmutación de etiquetas (*MPLS*). También se muestran otras soluciones muy interesantes para la construcción de la red las cuales fueron consideradas y analizadas durante la elaboración de este trabajo. Entre ellas está el proyecto Dynamips / Dynagen, el cual emula ruteadores Cisco en una PC convencional, RSVPd, un demonio que implementa IntServ con RSVP, *RSVP-TE daemon for DiffServ over MPLS under Linux*, el cual consiste en DiffServ sobre MPLS utilizando RSVP como protocolo de señalización para llevar a cabo experimentos de Ingeniería de Tráfico y el software propietario Packet Tracer versión 4.1 hecho por Cisco Systems el cual funge como simulador de redes.

Se examinaron varias ontologías de QoS. Se observo que están varían desde propósito, arquitectura como número de capas o semántica, forma de representación abstracta que va desde lógica de primer orden hasta XML, y demás conceptos de QoS. Se identificaron ventajas y desventajas entre ontologías. La mayoría de las ontologías QoS toman en cuenta conceptos como métricas, protocolos y propiedades de la red. Además, estas son más orientadas a aplicación ya que su propósito es dar soporte a servicios web. De las ontologías examinadas, la única ontología que involucra tanto al usuario como a la aplicación y los aspectos de QoS de la red es la ontología QoE. A esta ontología le conciernen ambas

QoE y QoS, sin embargo se cree que puede ser mejorada y complementada, por lo cual se le hizo una extensión a esta. Se considera que la ontología QoE extendida tiene un enfoque más robusto para QoS Management.

Se propone el algoritmo para el proceso de gestión de QoS QoE mediante agentes el cual sugiere una alternativa aun más racional que la de los trabajos analizados. Se utiliza la Teoría de la Decisión la cual está conformada por la Teoría de la Probabilidad y la Teoría de la Utilidad para dotar de este racionalismo. Con la utilidad se pretende satisfacer en lo más posible las necesidades del usuario con la menor cantidad de recursos posibles. Mientras que con las probabilidades se asignaron grados de creencia para decidir que configuración de mecanismo QoS resultara más provechoso dadas las condiciones de la red. Se muestra el diagrama de decisión con el cual se calcula la Máxima Utilidad Esperada que sin duda arrojara la mejor acción a elegir para el *Arbitrator Agent*. También se justifica el control de QoS mediante control por retroalimentación. Se plantea una arquitectura del sistema multiagentes en base a las fases de una sesión multimedia y las funciones de QoS Management que intervienen en esta, además de incorporar características deseables que se observan en otros trabajos analizados.

Se presento el modelo formal del proceso de QoS Management mediante redes de Petri y se analizan sus propiedades. De las propiedades a destacar está Libre de Bloqueo la cual nos indica que el proceso siempre terminara y este no caerá en un estado no deseado. No es acotada ya que puede recibir una cantidad infinita de solicitudes durante la transmisión que hará que el proceso de renegociación se realice el mismo número de veces. Para que sea cíclica solo se necesita agregar la transición y los arcos del final al inicio de la transmisión para que indiquen que se realiza una nueva transmisión.

Finalmente se hizo una comparativa de los resultados obtenidos en los experimentos realizados con la anterior plataforma y esta. Al proveer a los usuarios de un menú QoE más interactivo al incluir botones accionados por clics con el ratón, motivo a que la interacción entre la plataforma y el usuario se disparara, y como consecuencia hubo tendencias muy marcadas en cuanto a las solicitudes del usuario, mismas que se pueden observar en los valores del parámetro cfactor. Según las métricas objetivas no se violaron los valores de referencia establecidos por la ITU-T en cuanto a jitter y retraso. Sin embargo ocurrió una pérdida de paquetes mayor que en los experimentos con la anterior

plataforma. Los valores de las métricas subjetivas se dispararon considerablemente debido a la influencia que el parámetro cfactor produce durante el cálculo de estas métricas, esto a consecuencia de la tendencia de los usuarios con sus solicitudes. Mismo acaso para la NQoS. Con los valores ya normalizados de NQoS se observa un aumento de esta, lo cual refleja que los mecanismos de QoS proveyeron en mayor cantidad la QoE deseada al usuario, presumiblemente por la inclusión de nuevos mecanismos de QoS y la forma racional de gestionar los recursos de la red por parte de los agentes. Adicionalmente se incluyo una prueba mediante SSCQE la cual arrojo como resultado un valor de MOS de 55.60, el cual corresponde a una calificación de *apropiado* y muestra que la calidad de la transmisión percibida por el usuario fue solamente la *apropiada*. Se estuvo muy cerca de obtener una calificación de *buena*.

Como trabajo futuro se consideran cuatro puntos principalmente.

- Probar la operatividad y escalabilidad de la plataforma. Se contempla la construcción de un modulo para el simulador de redes NS2 en el cual se pueda implementar la solución de gestión de QoS - QoE propuesta en esta tesis. De esta manera se podría observar su comportamiento en redes de gran escala. Se debe incluir el integrar los tres mecanismos convencionales de QoS en una misma versión de NS2 ya que el modulo de MPLS y el de RSVP están hechos para diferentes versiones de este.
- Usar RSVP en IntServ. Esta propuesta involucra tomar el código de RSVPd como base para construir una solución compatible con *kernel* versión 2.4 y 2.6 y así poder implementar IntServ con RSVP en la red experimental.
- Incluir aprendizaje en el agente *Arbitrator*. Ya que no se tiene conocimiento a priori del estado de la red y tampoco se está guardando un registro historial para facilitar la toma de decisiones del agente, dotar de aprendizaje a este optimizaría aun más la gestión de los recursos de la red.
- Inclusión de mecanismos activos de QoS. A pesar de que la anterior implementación contaba con un agente para el descarte de paquetes, se desea incluir más de estos mecanismos activos para proporcionar un servicio todavía más variado y tratar de mejorar aun más la QoE del usuario.

Referencias

[1] Nichols, K., Blake, S., Baker, F. and D. Black, "Definition of the Differentiated Services Field (DSField) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[2] Braden, R., Clark, D., Shenker, S., "Integrated Services in the Internet Architecture: an Overview", RFC 1633, July 1994.

[3] Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S., "Resource ReSerVation Protocol (RSVP)", RFC 2205, September 1997.

[4] Rosen, E., Viswanathan, A., Callon, R., "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[5] http://www.isi.edu/nsnam/ns/

[6] Ghinea, G., and Thomas, J.: 'Quality of perception: user quality of service in multimedia presentations', IEEE Trans. Multimed., 2005, 7, pp. 786–789.

[7] M Siller and J Woods, "A Quality of Experience Framework for Audio and Video Transmission", IEE WIAMIS 2003 Proceedings, pp. 288-293, 4th European Workshop on Image Analysis for Multimedia Interactive Services, London, UK, April 2003.

[8] http://www.zebra.org/

[9] http://www.gnu.org/

[10] http://www.gnu.org/licenses/gpl.html

[11] http://www.isi.edu/rsvp/.index.html

[12] http://dsmpls.atlantis.ugent.be/

[13] http://mpls-linux.sourceforge.net/

[14] http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator

[15] http://www.utc.fr/

[16] http://www.dynagen.org/

[17] http://www.cisco.com

[18] http://www.winpcap.org/

[19] "Pathload : A measurement tool for end-to-end available bandwidth", by Manish Jain and Constantinos Dovrolis, published at PAM 2002.

[20] "End-to-End Available Bandwidth: Measurement methodology, Dynamics, and Relation with TCP Throughput", by Manish Jain and Constantinos Dovrolis, in Proceedings of ACM SIGCOMM

in August 2002.

[21] M. Siller and J. Woods, "Using an Agent Based Platform to Map Quality of Service to Experience in Conventional and Active Networks", Journal IEE Proceedings Communications, Vol. 153, issue 6, pp. 828-840, 2006.

[22] 'PROTEGE: The protégé ontology editor and knowledge acquisition system.' http://protege.stanford.edu/

[23] T. R. Gruber, "A translation approach to portable ontologies", Knowledge Acquisition, Vol.5, No. 2, pp199-220, http://kslweb.stanford.edu/KSL_Abstracts/KSL-92-71.html, 1993.

[24] Zhou, J., "Knowledge Dichotomy and Semantic Knowledge Management", in In the proceedings of the 1st IFIP WG 12.5 working conference on Industrial Applications of Semantic Web, (Jyvaskyla, Finland, 2005), 305 316.

[25] Dobson, G.; Sanchez-Macian, A., "Towards Unified QoS/SLA Ontologies", Services Computing Workshops, 2006. SCW '06. IEEE, Sept. 2006 Page(s):169 – 174

[26] V. Tosic, B. Esfandiari, B. Pagurek, and K. Patel, "On Requirements for Ontologies in Management of Web Services", international Workshop on Web Services, E-Business and the Semantic Web, May 2002.

[27] Dobson, G.; Sanchez-Macian, A., "Towards Unified QoS/SLA Ontologies", Services Computing Workshops, 2006. SCW '06. IEEE, Sept. 2006 Page(s):169 – 174

[28] Jiehan Zhou; Niemela, E., "Toward Semantic QoS Aware Web Services: Issues, Related Studies and Experience", Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on Dec. 2006 Page(s):553 – 557.

[29] Zhou, C., Chia, L., and Lee, B., "DAMLQoS Ontology for Web Services", in proceeding of the International Conference on Web Services 2004(ICWS04), (2004).

[30] FIPA, FIPA Quality of Service Ontology Specification, 2005. www.fipa.org/specs/fipa00094/XC00094.pdf

[31] Tian, M., Gramm, A., Ritter,H., and Schiller,J., Efficient Selection and Monitoring of QoSaware Web services with the WSQoS Framework. in 2004 IEEE/WIC/ACM International Conference on Web Intelligence (WI'04), (Beijing, China, 2004).

[32] Dobson, G., Lock, R., and Sommerville, I., "QoSOnt: a QoS Ontology for Service Centric Systems", in Euromicro SEAA, (2005), 8087.

[33] Maximilien, E.M. and Singh, M.P, "A Framework and Ontology for Dynamic Web Services

Selection", IEEE InternetComputing, 8 (5). 8493.

[34] QoS Management By Mobile Agents in Multimedia Communication. S. S. Manvi and P Venkataram. 2000

[35] Artificial Intelligence- A Modern Approach. 2nd Edition. Stuart J. Russell and Peter Norvig. Prentice Hall. 1995.

[36] An Introduction to Multiagent Systems. Michael Woldridge. 2002

[37] H. Raiffa. *Decision Analysis: Introductory Lectures on Choices under Uncertainty*. Addison Wesley, Reading, MA., 1968.

[38] Game Theory and Decision Theory in Multi-Agent Systems. Simon Parsons, Michael Wooldridge. *Autonomous Agent and Multi-Agent Systems* 2000

[39] J. O. Kephart and A. R. Greenwald. Shopbot economics. International Journal of Autonomous Agents and Multi-Agent Systems, 2000.

[40] W. C. Stirling, M. A. Goodrich, and D. J. Packard. Satisficing equilibria: a non-classical theory of games and decisions. International Journal of Autonomous Agents and Multi-Agent Systems, (this issue), 2000.

[41] J. Lang, L van der Torre, and E. Weydert. Utilitarian desires. International Journal of Autonomous Agents and Multi-Agent Systems, 2000.

[42] An Agent-Based Approach for Quality of Service Negotiation and Management in Distributed Multimedia Systems. Luiz A.G. Oliveira, Paulo C. Oliveira and Eleri Cardozo.

[43] A. Vogel et al.: Distributed Multimedia and QoS: A Survey, IEEE Multimedia, Summer 1995, pp. 10-18.

[44] Management of QoS with Software Agents. Hermann de Meer, Antonio Puliafito, and Orazio Tomarchio

[45] An Approach to Quality of Service Management for Distributed Multimedia Applications. A. Hafid and G.v.Bochmann. 1995.

[46] An agent-based framework for QoS management. Antonio Puliafito, Orazio Tomarchio, Hermann de Meer

[47] A Multi-Agent Architecture for Cooperative Quality of Service Management. Abdelhakim Hafid and Stefan Fischer.

[48] Intelligent Agents for QoS Management. Krunoslav Trzec and Darko Huljenic. AAMAS'02.

[49] An XML-based multi-agent system for the user oriented management of QoS in

telecommunicactions networks. Pasquale De Meo, Jameson Mbale, Giorgio Terracina and Domenico Ursino. Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology (IAT'03). 2003

[50] Programmable Agents for Flexible QoS Management in IP Networks. Hermann De Meer, Aurelio La Corte, Antonio Puliafito, and Orazio Tomarchio. 2000

[51] Adaptive QoS Management Using Layered Multi-Agent System for Distributed Multimedia Applications. Masakatsu Kosuga, Tatsuya Yamazaki, Nagao Ogino, Jun Matsuda. 1999.

[52] Agent Based QoS Management and Pricing. Austin Poulton and Peter Clayton. 1999.

[53] Intelligent Algoritms for QoS Management in Modern Communication Networks. Bin Qiu. 2003.

[54] ITU-T, 'Network performance objectives for IP-based services', Recommendation Y.1540 ed., May 2002. Series Y

[55] Applaying Feedback Control to QoS Management. Giovanna Ferrari.

[56] López, E. "Introducción a las Redes de Petri" Apuntes de curso. Facultad de Ciencias Físico-Matemáticas. UANL, Octubre, 1997.

[57] Dessel J., Esparza J. "Free choice Petri nets" Cambridge University Press 1995.

[58] http://pipe2. sourceforge.net/

[59] Jade: 'The agent development framework', http://jade.tilab.com/

[60] http://java.sun.com/

[61] MGEN: Mgen: The milti-generator toolset. http://mgen.pf.itd.nrl.navy.mil/.

[62] KURT: Kansas university real time linux. http://www.ittc.ku.edu/kurt/.

[63] ITU-R Recommendation BT.500: "Methodology for the subjective assessment of the quality of television pictures." International Telecommunication Union, Geneva, Switzerland, 2002.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Gestión y Control de Mecanismos de Calidad de Servicio en Redes IP

del (la) C.

Edson GALLO CONTRERAS

el día 30 de Mayo de 2008.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Mario Angel Siller González
Pico
Investigador CINVESTAV 2A
CINVESTAV Unidad Guadalajara

CINVESTAV
BIBLIOTECA CENTRAL
SSIT000006319