AGENTE ADMINISTRADOR DE QoS SOBRE REDES IPv6: APROXIMACIÓN INTSERV Y DIFFSERV EN LOS NODOS DE ACCESO

Ing. GUSTAVO ADOLFO RAMÍREZ ESPINOSA

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
MAESTRÍA EN INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C.-MAYO 2013

AGENTE ADMINISTRADOR DE QoS SOBRE REDES IPv6: APROXIMACIÓN INTSERV Y DIFFSERV EN LOS NODOS DE ACCESO

Ing. GUSTAVO ADOLFO RAMÍREZ ESPINOSA

TRABAJO DE GRADO

DIRECTOR:

Ing. LUIS CARLOS TRUJILLO ARBOLEDA, *M.Sc.*

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA ELECTRÓNICA

BOGOTÁ D.C. - MAYO 2013

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD INGENIERÍA
MAESTRÍA EN INGENIERÍA ELECTRÓNICA

Rector Magnífico: Padre Joaquín Emilio Sánchez García S.J.

Decano Académico Facultad de Ingeniería: Ing. Jorge Sánchez Téllez M.Sc.

Decano Del Medio Universitario: Padre Sergio Bernal Restrepo S.J.

Director de la Maestría en Ingeniería Electrónica: Ing. Cesar Leonardo Niño. Ph.D.

NOTA DE ACEPTACIÓN

_____

_____

_____

_____
Director del proyecto

_____
Jurado

_____
Jurado

MAYO 2013

NOTA DE ADVERTENCIA

"La Universidad no se hace responsable de los conceptos emitidos por algunos de sus alumnos en los proyectos de grado. Solo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vea en ello el anhelo de buscar la verdad y la justicia."

Artículo 23 de la Resolución No. 13, del 6 de julio de 1946, por la cual se reglamenta lo concerniente a Tesis y Exámenes de Grado en la Pontificia Universidad Javeriana.

A mis padres, mi familia,
y mi novia que amo,
por todo el apoyo
recibido y motivaciones.

Tabla de Contenido

LISTA DE ILUSTRACIONES

LISTA DE TABLAS

LISTA DE GRÁFICAS

LISTA DE ANEXOS

1 INTRODUCCIÓN

La expansión de las redes de datos basadas en IP en la última década ha impulsado la implementación de nuevas aplicaciones que a su vez han incrementado el volumen de transferencia de datos [6]. Sin embargo las redes de transporte intra e inter dominio no han aumentado su ancho de banda a la misma velocidad [1].

La creciente digitalización de servicios multimedia sobre redes IP como voz, video y nuevas aplicaciones como telemedicina, control industrial, etc., servicios condicionados a ciertas restricciones, generalmente asociadas a su sensibilidad, al tiempo y a errores durante la transmisión. A través de la redes, los servicios experimentan congestión o retardos, diferencia en el tiempo de arribo entre paquetes (*jitter*), pérdida de paquetes o errores en la secuencia que afectan el comportamiento del servicio [2]. Tales condiciones deben ser respetadas para que la calidad de experiencia del usuario (QoE) sea positiva y se cumplan las restricciones impuestas por la sensibilidad al tiempo de cada servicio.

Para cumplir dichas especificaciones de sensibilidad la red debe ajustar ciertos parámetros o políticas de admisión de paquetes, así mismo, se deben tener los mismos parámetros o políticas en el manejo de la transmisión para cumplir los valores de throughput[1] y restricciones establecidas dependiendo del tipo de servicio. El conjunto de características cuantitativas y cualitativas de un sistema distribuido multimedia que son necesarias para lograr la funcionalidad requerida de una aplicación se denomina calidad de servicio (QoS) [12].

Para establecer QoS es necesario diferenciar el tráfico sensible al tiempo (tráfico transaccional) del no transaccional, este último es indiferente al reenvío de paquetes, retardos o arribo fuera de secuencia, por ejemplo un servicio de descarga de archivos. Para la diferenciación de servicios, se han establecido diferentes mecanismos a lo largo de cada una de las capas del modelo de referencia OSI [1], pero en mayor medida en los protocolos de la capa de red. Por ejemplo algunas aproximaciones para lograr QoS por parte de los protocolos de red [10] se basan

[1] *Throughput* se define como la cantidad de datos que son procesados por una maquina o sistema en un periodo particular de tiempo. (Definición de throughput tomada de Cambridge Business English Dictionary © Cambridge University Press)

en los campos Type of Service para la cabecera IPv4 o Traffic Class para la cabecera IPv6 [11]. Ambos tienen como objetivo proporcionar ciertos bits que permiten diferenciar el tipo de servicio que es encapsulado en el paquete IP [10] [12]. Otros protocolos de capas inferiores como MPLS (Multiprotocol Label Switching) asignan una etiqueta a cada tipo de tráfico para que este sea enviado por diferentes rutas según la etiqueta que se le ha asignado [2]. Independientemente de la forma de clasificación del tráfico, el objetivo de QoS es asignar prioridades de procesamiento y transmisión al tráfico de carácter transaccional, buscando garantizar los parámetros de retardo, *jitter* y errores en secuencia [1]. La actual disruptiva en un modelo de QoS es la presente migración mundial hacia el protocolo IPv6 el cual da mayores opciones en su cabecera para el manejo de diferenciación de paquetes y flujos de paquetes. IPv6 brinda además una gran área para la investigación y desarrollo de protocolos de QoS[3].

Este trabajo de grado pretende abordar el aprovisionamiento de QoS, una problemática actual y objeto de investigación, en la que actualmente no existe un proceso o algoritmo definitivo que de solución [6]. Se busca también una solución basada en el planteamiento de un modelo que aproveche las características del protocolo IPv6, protocolo de gran importancia en la evolución y crecimiento de los servicios sobre Internet y soporte de nuevos servicios. Sin embargo, IPv6 no provee QoS per se, aunque el protocolo provee diferentes campos [7] [11]. Esta propuesta tiene la intención de abordar la problemática de QoS buscando una integración óptima de los campos que provee IPv6.

2 MARCO TEÓRICO

El protocolo IPv6 surge como una tecnología disruptiva frente a la necesidad de una actualización del protocolo IPv4. Sobre IPv4 esta soportado el actual Internet. Sin embargo, IPv4 fue creado en los inicios de los 70's para interconectar redes gubernamentales y académicas. Inicialmente este protocolo no fue pensado para la masificación y popularidad con que hoy cuenta, tampoco previó la numerosa cantidad de servicios multimedia y la inmensa cantidad de dispositivos de consumo masivo que tienen la capacidad de conectarse a Internet, tales como: cámaras (vigilancia, web, etc), televisores, teléfonos móviles, etc. IPv4 ha estado vigente por más de 30 años, periodo en el cual ha sufrido varias modificaciones debido a una asignación arbitraria de las direcciones de red [7], en la cual el 60% de la capacidad de direccionamiento teórica (4.3 mil millones) fue asignado solo a los Estados Unidos, a la mala gestión a partir de clases de direccionamiento de red que conllevó a la eliminación de estas con el mecanismo CIDR *(Classless InterDomain Routing*), y a la introducción del mecanismo de traslación de direcciones NAT[1] (*Network Address Translation*).

Todas estas modificaciones y actualizaciones buscaron contrarrestar el problema de la capacidad de direccionamiento, en mayor medida el uso de NAT. Sin embargo, el uso de NAT también trajo consigo desventajas en la operación, en la seguridad *end-to-end* y en esquemas de mapeo de direccionamiento, pos su mayor complejidad [7]. En materia de QoS, el protocolo IPv4 solo provee el campo *Type of Service* dentro de su estructura de cabecera como se ve en la figura 2-1.

[1] RFC 1631, actualizado por el RFC 3022

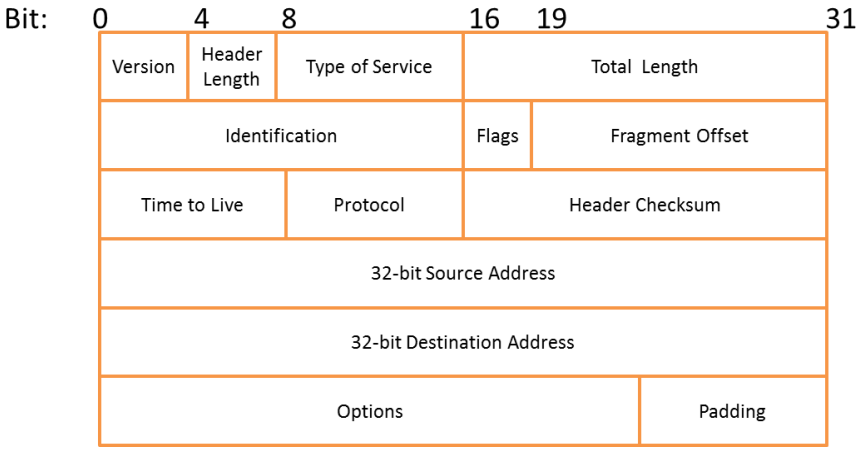


Figura 2-1. Cabecera IPv4. [2]

El campo *Type of Service* es un campo de 8-bits en el que se definen parámetros como fiabilidad, prioridad, retardo y rendimiento. La codificación de este campo ha sido cambiada a lo largo de la vida de IPv4 en respuesta a las necesidades de compatibilidad con otros protocolos, también a necesidades de codificación en la que dicha codificación estuvo vigente. Inicialmente en el RFC 791 los tres primeros bits fueron designados para indicar la precedencia, los siguientes tres se designaron como *Type of Service* (TOS), y los últimos dos se reservaron para uso futuro (Figura 2-2a). Posteriormente se incluyeron los bits 6 y 7 en el campo TOS pero no se definió el uso para estos dos bits (Figura 2-2b). En el RFC 1349 se designó nuevamente el bit 6 como "*Minimize Monetary Cost*" y el bit 7 como "*Must Be Zero*" (Figura 2-2c).

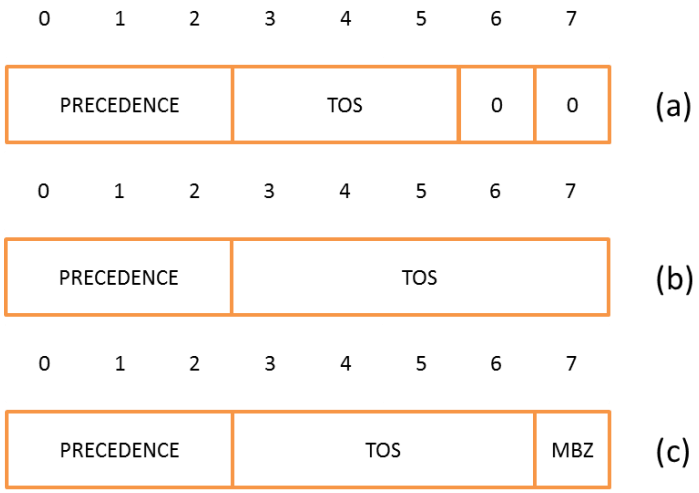


Figura 2-2 Campo Type of Service definido en (a) RFC 791, (b) RFC 1122 y (c) RFC 1349[3]

[2] Autoría Propia

El RFC 1349 y 1455 fueron remplazados por la definición de los campos correspondientes a *Differentiated Services* (DiffServ) o *DS field* en el RFC 2474. En este RFC se definieron en los primeros 6 bits la codificación DSCP (*Differentiated Services Codepoint*) y los últimos 2 bits fueron definidos como "*currently unused*" (CU) como se puede ver en la figura 2-3a. Fue entonces hasta el RFC 2780 donde estos dos últimos bits se designaron como "*Explicit Congestion Notification*" ECN figura 2-3b. Ambos campos serán profundizados en la siguiente sección.

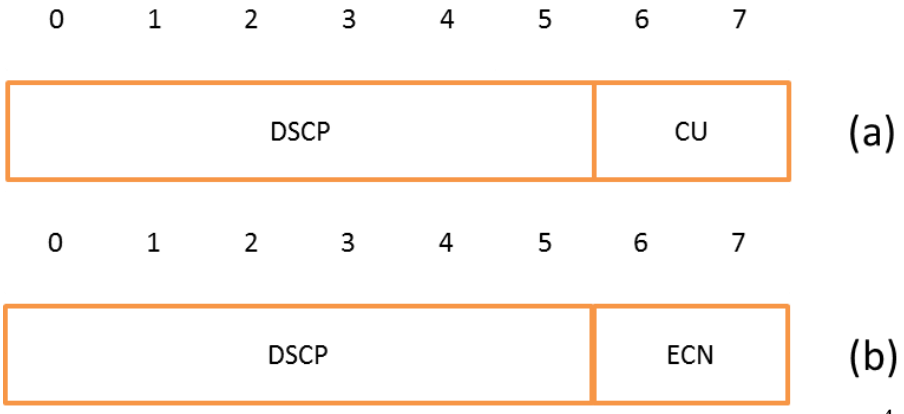


Figura 2-3 Campo Type of Service definido en (a) RFC 2474, (b) RFC 1122[4].

Aunque IPv4 presenta opciones para el manejo de QoS el campo anteriormente expuesto no es suficiente. IPv4 ha sufrido cambios y "parches" que le han actualizado o introducido nuevas características pero el principal problema es el agotamiento de direcciones IPv4. Los países asiáticos principalmente presentan una gran parte de la población mundial y requieren así un gran número de direcciones IPv4, aún más con la alta demanda que presenta su población actualmente. Razón que ha generado una demanda hacia un nuevo protocolo de mayor capacidad.

Dadas estas dificultades el IETF (*Internet Engineering Task Force*) a principios de los 90's planteó la necesidad de diseñar un protocolo que sucediera a IPv4.

[3] Autoría Propia
[4] Autoría propia.

2.1 Protocolos de Internet de Próxima Generación

Diferentes propuestas surgieron para contrarrestar las debilidades de IPv4, dentro de las principales se encuentran *CNAT*, *IP Encaps*, *Nimrod*, y *Simple CLNP*. Luego les siguieron propuestas como the P Internet Protocol (PIP), Simple Internet Protocol (SIP)[5], y TP/IX. En 1992 luego de una reunión del IETF, CLNP evolucionó a *TCP and UDP Bigger Addresses* (TUBA)[6], *IP Encaps* a *IP Address Encapsulation* (IPAE), este último junto a PIP y SIP formaron el *Simple Internet Protocol Plus* (SIPP)[7]. Por último el TP/IX convergió al protocolo *Common Architecture for the Internet*

En 1993 el IETF creó el área para la investigación de las diferentes propuestas y también con el fin de desarrollar la recomendación RFC 1752 "*The recommendation for the IP Next Generation Protocol*". El principal objetivo de esta recomendación era diseñar un protocolo que heredara la madurez del protocolo IPv4, estuviera acorde a las nuevas necesidades y tuviese la misma expectativa de vida que IPv4.

No fue sino hasta noviembre 17 de 1994 cuando el *Internet Engineering Steering Group* publicó un borrador sobre el protocolo IPv6 y en 1995 el RFC 1883 "*Internet Protocol, Version 6 (IPv6) Specification*". Posteriormente varios borradores fueron consolidados y en 1998 se publicaron los protocolos principales de IPv6, incluyendo la actualización RFC 2460 que dejo obsoleto el RFC 1883 [7].

2.2 IPv6

El protocolo IPv6 soluciona todas las limitaciones de IPv4 en materia de escalabilidad y flexibilidad. El objetivo del diseño en IPv6 era crear un protocolo que pudiera manejar el gran crecimiento de Internet y suplir los requerimientos de servicios, movilidad y seguridad.

Las principales mejoras de IPv6 [7] se pueden resumir en:

Extensión del espacio de direccionamiento: IPv4 tenía un espacio de direccionamiento de 32 bits el cual en IPv6 es incrementado a 128 bits, espacio suficiente para asignar millones de direcciones IPv6 a cada habitante del planeta hoy en día. Adicionalmente permite una distribución de direcciones en el mundo más organizada, logrando un uso eficiente del espacio de direccionamiento y un enrutamiento más eficiente.

[5] No confundir con el protocolo SIP "Session Initation Protocol" RFC 3261.
[6] RFC 1347, 1526 y 1561.
[7] RFC 1710.

Auto configurable: En IPv6 un dispositivo que desee conectarse a una red puede utilizar su propia dirección de capa 2 y tomar un prefijo para auto asignarse una única dirección IPv6. En IPv4 era necesaria la utilización del protocolo DHCP para asignar la dirección IPv4 a cada dispositivo.

Simplificación de la cabecera: IPv6 fija la longitud de la cabecera a 40 bytes, elimina algunos campos de IPv4 e introduce otros, ver figura 2-4. 32 bytes están destinados a la información de direcciones de origen y destino, los restantes 8 bytes están destinados a información general de la cabecera. IPv6 al tener una longitud de cabecera fija hace más eficiente el procesamiento.

Soporte mejorado a opciones y extensiones: Las opciones en IPv4 eran incluidas en la cabecera, IPv6 redirecciona estas opciones a extensiones de cabecera adjuntas a la cabecera fija, y son utilizadas solo cuando es necesario para no comprometer la velocidad de procesamiento. La especificación base describe un conjunto de cabeceras de extensión, cabeceras para enrutamiento, movilidad, calidad de servicio y seguridad. El uso de extensión de cabecera permite agregarle nuevas características al protocolo IPv6 para futuras necesidades. La cabecera y las cabeceras de extensión se explican con más detalle a continuación.

2.2.1 Cabecera IPv6

La cabecera IPv6 definida en el RFC 2460 consta de una longitud de 40 bytes, a diferencia de IPv4, IPv6 elimina algunos campos como *Header Length* al mantener siempre una cabecera fija. Los campos *Identification*, *Flags* y *Fragment Offset,*encargados de manejar la fragmentación en IPv4, también fueron retirados. IPv6 reconoce el máximo MTU para la transmisión aunque si un host desea fragmentar debe utilizar la cabecera de extensión prevista con la restricción de que solo las fuentes de tráfico pueden hacerlo, diferente a lo que sucede en IPv4 con el cual cualquier enrutador o elemento intermedio puede realizar la fragmentación. Finalmente el campo *Header Checksum* fue también suprimido para mantener la velocidad en el procesamiento y derivar la responsabilidad de integridad a capas superiores. Otros campos como *Type of Services*, *Time to Live* y *Protocol Type* fueron renombrados, como se aprecia en la figura 2-4. Los campos de la cabecera IPv6 se especifican a continuación:

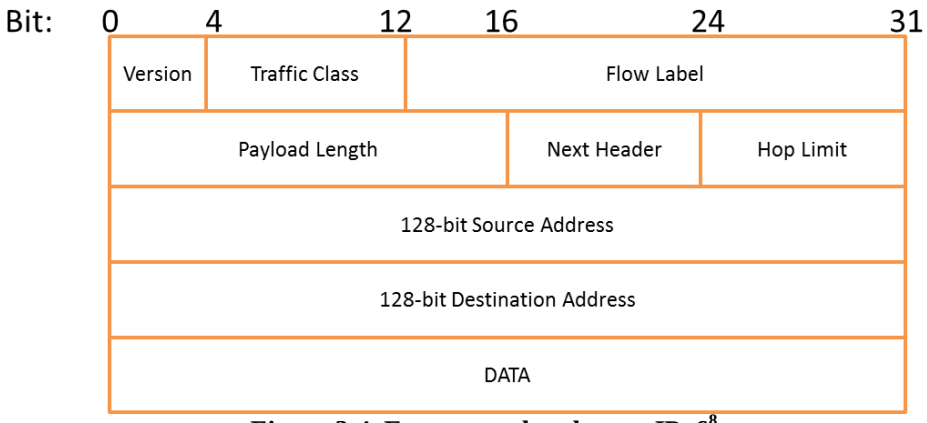


Figura 2-4. Estructura de cabecera IPv6[8].

Version (4-bits): Este campo indica la versión del protocolo IP, para este caso 6.

Traffic Class (8-bits): Este campo remplazó el campo *Type of Service* de la versión 4. Su función principal es facilitar el manejo de tráfico sensible al tiempo y darle un tratamiento especial frente a tráfico no tan sensible, clasificándolo en diferentes clases o prioridades según la codificación DSCP (Differentiated Services Code Point, RFC 2474) y manejo de congestiones que puedan presentarse (bits ECN).

Flow Label (20-bits): Este campo distingue paquetes que requieren un mismo tratamiento y pertenecen a un tipo de tráfico, agrupándo este conjunto de paquetes como un flujo. Un enrutador puede hacer seguimiento a un determinado flujo de paquetes y proporcionarle un manejo selectivo en el enrutamiento al poder distinguirlo de otros flujos. Los paquetes son del mismo flujo si y solo si tienen el mismo valor de *Flow Label* y la misma dirección origen y destino.

Payload Length (16-bits): Este campo especifica la longitud de la carga útil encapsulada por IPv6, incluyendo las cabeceras de extensión más no la cabera principal. Al ser un campo de 16 bits permite un máximo de 64 Kbytes de carga útil. Sin embargo, mediante cabeceras de extensión (*Jumbogram Extension Header*) puede soportar paquetes de mayor carga. (RFC 2675).

Next Header (8-bits): Es el campo *Protocol Type* de IPv4 renombrado debido a la nueva organización de IPv6. Este campo indica el protocolo siguiente ó la cabecera de extensión correspondiente.

[8] Autoría Propia

Hop Limit (8-bits): Este campo es análogo al Time To Live de IPv4, siendo la principal diferencia que ya no es una medida de tiempo sino únicamente el número de saltos por la que puede pasar un paquete.

Source Address (128-bits): Este campo contiene la dirección IPv6 del host que origina el paquete.

Destination Address (128-bits): Este campo contiene la dirección IPv6 del host a quien va dirigido el paquete , el cual puede ser el destino final, o la dirección del próximo salto o a quien aplique una cabecera de extensión.

2.2.2 Cabeceras de extensión

La longitud fija de IPv6 garantiza un manejo eficiente de la cabecera, sin embargo son necesarias otras opciones. Para este fin fueron definidas las cabeceras de extensión, que inicialmente el RFC 2460 establece las primeras cabeceras de extensión:

- *Hop-by-Hop Options Header*
- *Routing Header*
- *Fragment Header*
- *Destination Options Header*
- *Authentication Header*
- *Encrypted Security Payload Header*

Nuevas cabeceras se han definido para diferentes necesidades, por ejemplo, la cabecera de extensión para Movilidad (RFC 3775). Las cabeceras de extensión son analizadas generalmente en el host que indica la dirección de destino con el fin de hacer eficiente el enrutamiento. Las cabeceras de extensión no son consideradas en la ejecución para este proyecto. Este proyecto se concentra en QoS sobre la cabecera principal.

2.3 QoS en IPv6

En el mundo IP los paquetes son manejados de la misma manera, los enrutadores redireccionan paquetes que reciben de diferentes fuentes en diferentes interfaces y son procesados según el orden de llegada; a esto se le denomina "mejor esfuerzo". El desempeño de las redes según esta forma de entrega conlleva ciertos fenómenos no deseados en los cuales se encuentra: la posibilidad de que los paquetes sean entregados en orden o en desorden; la posibilidad de que los paquetes sean entregados suavemente o en ráfagas; la posibilidad de que sean o no entregados. Estos fenómenos [13] se pueden presentar debido a las siguientes causas:

- Transitorios de Congestión
- Ausencia de diferenciación de paquetes
- Independencia en las políticas de priorización de tráfico en cada enrutador.

Los protocolos de QoS tienen como objetivo determinar una diferenciación de extremo a extremo y una asignación de recursos según las necesidades del tráfico, de tal manera que estos se mantengan constantes a lo largo de toda la ruta y durante el tiempo de la comunicación.

La clasificación del tráfico resulta una tarea esencial más no trivial, pues el volumen de tráfico presente en una red como Internet es increíblemente alto. Un procesamiento detallado que clasifique cada paquete demanda un alto procesamiento[4] [8] [7]. En redes *backbone* que soportan el alto volumen de tráfico, el procesamiento a cada paquete y la asociación de una decisión de enrutamiento para cada paquete, generaría congestiones y retardos aún mayores de los que se desea solucionar.

El aprovisionamiento de QoS es un proceso de ingeniería de tráfico limitado, utiliza criterios de decisión basados en algoritmos. Una alta complejidad en los algoritmos y criterios de decisión pueden significar retardos nocivos que deben limitarse al máximo [11] [9]. Las soluciones deben ser altamente eficientes sobre todo en el *backbone* de una red donde se presenta el mayor número de tráfico [7] [4]. En la actualidad se encuentra diferentes aproximaciones para proveer QoS los cuales se pueden condensar en dos grandes grupos: *IntServ (Integrated Services)* y *DiffServ (Differentiated Services)* [5] [3] [2]. Independientemente del tipo de aproximación que se implemente en una solución de QoS, cada solución cuenta con problemas de adaptabilidad a la variación de la demanda de tráfico y problemas de escalabilidad en donde se compromete el rendimiento. QoS es aún materia de investigación [8] y el criterio de selección de cada solución es determinada por su escalabilidad y nivel de granularidad en la clasificación del tráfico [2].

2.3.1 Integrated Services

Integrated Services fue definido en 1994 en el RFC 1633 "*Integrated Services in the Internet Architecture: an Overview*" en el cual bajo la experiencia obtenida en pruebas con *multicast*, se buscó combatir los retardos variables de la gestión de colas y perdidas por congestión. El objetivo de esta aproximación, es emular un circuito que tenga similares características a un circuito perteneciente a una red de conmutación de circuitos asegurando ancho de banda o recursos según la categoría del tráfico. Estos recursos de red como ancho de banda, y rutas seguras son

necesarias para poder dar un tratamiento uniforme al tráfico por lo que es necesario un proceso de establecimiento de estos recursos, para tal fin, la aplicación solicita ciertos recursos de calidad de servicios a todos los nodos de red a lo largo de un camino definido hasta el nodo final. IntServ para lograr la funcionalidad descrita se soporta en protocolos de señalización sobre redes IP para proveer QoS [9]. El protocolo de señalización de QoS más utilizado es el protocolo RSVP (Resource Reservation Protocol, RFC 2205). RSVP proporciona una negociación, establecimiento de conexión y reserva de recursos que buscan garantizar QoS [5] [10].

Las aproximaciones basadas en *IntServ*, tienen como objetivo garantizar QoS para cada sesión de una aplicación individual [9] controlando el retardo. La reserva de recursos a su vez permite predecir un retardo y garantizar un retado máximo, para tal fin se definió el concepto de flujo. Un flujo está definido como el conjunto de paquetes que tienen en común una misma dirección de origen y destino y un mismo puerto de origen y destino (en el caso de IPv4), o misma etiqueta identificadora de flujo (caso IPv6). Para los paquetes que cumplen estas características, es válido asumir [13] que hacen parte de una transmisión perteneciente a una misma aplicación. Estos flujos pueden ser definidos de dos maneras [9] por especificación de tráfico y especificación por solicitud del servicio. La primera es una especificación de un patrón de tráfico que espera recibir un flujo, mientras que la segunda es una solicitud específica de QoS para un flujo perteneciente a un servicio.

La aproximación IntServ demanda que cada uno de los enrutadores presentes en una red realice las siguientes funciones:

Control de Admisión de Paquetes: El control de admisión es un algoritmo que le permite al host o enrutador determinar si es posible verificar, aceptar o denegar los recursos solicitados para un determinado flujo.

Control de Políticas: Según el usuario que solicita los recursos, el control de políticas administra y determina si tiene los permisos y si pueden ser asignados a este usuario o si excede los recursos solicitados. De exceder los recursos asignados el control de políticas decide las acciones a tomar, las cuales pueden variar, desde el descarte de los paquetes que demandan o superan la capacidad del recurso asignado, bajar la prioridad del flujo, o marcar los paquetes que excedan dicha capacidad.

Clasificación de Paquetes: Un paquete admitido debe ser identificado dentro de su flujo correspondiente, y clasificado en una clase de QoS designada para ese flujo, donde todos los paquetes pertenecientes a la misma clase y flujo experimentan el mismo tratamiento. Dicha clasificación se realiza gracias al contenido en la cabecera.

Algoritmo de atención de colas: El algoritmo de atención determina el redireccionamiento o envío de paquetes a una serie de colas y temporizadores. El algoritmo se asegura de aplicar el nivel de QoS solicitado a cada flujo identificado por el clasificador, permite que paquetes con mayor prioridad pertenecientes a un flujo sean enviados primero que paquetes de otros flujos, también decide el tratamiento a los paquetes que sobrepasen la capacidad de la cola, este proceso generalmente se lleva a cabo en la capa de enlace de datos.

En general *IntServ* requiere que cada nodo analice la información de cada flujo de dato, determine el tipo de flujo, y por último aplique los criterios de direccionamiento y QoS para cada tipo de flujo [1]. La principal ventaja de esta aproximación es la granularidad de asignación de recursos, flexible en adicionar o liberar recursos de una conexión existente. Sin embargo su gran deficiencia es la escalabilidad [2] [5]. Las principales deficiencias son la sobrecarga enorme de almacenamiento y procesamiento de los nodos que crece proporcionalmente a los flujos, razón por la cual considerando la complejidad de Internet en materia de flujos, los requerimientos de procesamiento en cada nodo son altos, ya que cada nodo debe implementar RSVP[9], control de admisión, clasificación y planificación de paquetes afectando el desempeño de enrutadores que manejen grandes volúmenes de paquetes como lo son los enrutadores de *backbone*[3].

2.3.1.1 IntServ sobre IPv6

Desde la primera definición del protocolo IPv6 (RFC 1809) se especificó el campo *Flow Label*, pero hasta el RFC 2460 se definió su longitud a 20bits. El RFC 2460 especifica que este campo puede ser usado por una fuente para marcar paquetes que requerían un manejo especial por los enrutadores, por manejo especial se refiera a un manejo determinado por un protocolo de señalización de QoS o por alguna cabecera de extensión asociada al manejo de QoS E.j. *Hop-by-Hop Header*. También define un flujo en IPv6 como una secuencia de paquetes de una misma fuente a un mismo destino (unicast o multicast) y con la misma etiqueta *Flow Label*, si un paquete no pertenece a un flujo este debe tener en este campo todos los bits en cero. La etiqueta

[9] Existen más protocolos de señalización para QoS, para mayor referencia consultar RFC 4094 "*Analysis of Existing Quality-of-Service Signaling Protocols*".

de *Flow Label* debe ser asignada por la fuente del tráfico, y equivale a un número pseudoaleatorio en combinación con la dirección de origen.

Durante el enrutamiento cada enrutador analiza cada flujo asociándolo a una política o requerimiento de QoS, la información de este campo al estar en la cabecera principal, la cual no es cifrada por protocolos de seguridad como IPsec. Sin embargo, para un enrutador el mantener la información de cada flujo hace poco escalables las soluciones y reservas de recursos dentro de redes extensas como lo es Internet.

Flow Label no ha sido ampliamente usado en soluciones de QoS. En un principio fue utilizado a manera de experimento, y solo hasta los últimos años se ha empezado a estandarizar su uso como lo es el RFC 3697, y actualmente definido por el RFC 6437 de noviembre de 2011.

2.3.2 Differentiated Services

La segunda aproximación es la diferenciación de servicios (DiffServ), esta aproximación pierde un poco de granularidad frente a IntServ, ya que clasifica el tráfico por clases de mayor a menor prioridad utilizando un código identificador para cada tipo de flujo. Estas categorías son identificadas por un *comportamiento por salto o PHB* (per-hop bevavior) con el cual cada nodo asigna una prioridad o recursos de ancho de banda según el tipo de servicio y sus necesidades de QoS [4] [6].

El clasificador puede diferenciar servicios basándose en la información de las cabeceras de cada paquete. Existen dos tipos de clasificadores, Clasificadores que se basan en un campo como el ToS o Traffic Class, (*Behavior Aggregate Classifier*) y los clasificadores que se basan en más de un campo de la cabecera IP incluso capas superiores (*Multi Field Classifier*). Un clasificador distingue entre clases pero no distingue flujos entre una misma clase, un enrutador del tipo *backbone* no podría manejar flujo por flujo al ser muy numeroso el número de flujos que enruta. Por lo tanto, de implementarse una solución basada en *IntServ* generaría mayores retardos dentro de un *backbone*.

2.3.2.1 DiffServ sobre IPv6. El campo DSCP

En IPv4, los 6 primeros bytes del campo *Type of Service* (ToS) y *Traffic Class* de IPv6 son algunos ejemplos de clasificación por tipo de tráfico que permiten la diferenciación de servicios gracias a la codificación de servicios DSCP (*Definition of the Differentiated Services Code Point*, RFC 2474) [4] [2]. La codificación DSCP asocia a cada paquete un PHB: El campo DSCP consta

de los primeros 6 bits, permitiendo 64 posibilidades distintas de códigos DSCP. Sin embargo se divide este grupo de códigos en los especificados en la tabla 2-1.

Grupo	Rango de Código	Uso
1	xxxxx0	Uso estándar
2	xxxx11	Uso local o experimental
3	xxxx01	Uso local o experimental; Potencial uso estándar en el futuro.

Tabla 2-1 Definición de los códigos dentro del campo DSCP.

El primer grupo es de uso estándar el cual permite 32 códigos determinados por la estandarización definida por la IETF. El grupo 2 abarca 16 códigos que han sido reservados para uso experimental o local. Por último el grupo 3, también tiene uso local y experimental, sin embargo de ser necesarios más códigos que los que soporta el grupo 1, el grupo 3 puede ser utilizado como extensión.

El rango de códigos posibles en el campo DSCP no definen un PHB en específico, a un código DSCP puede ser asignado cualquier PHB, lo que hace que el número de PHB sea ilimitado. Los PHB son mapeados en los códigos del campo DSCP. Los PHB más usados y mayormente aceptados son descritos a continuación.

- **Default PHB.** Este campo es recomendado codificarlo con todos los bits del campo DSCP en 0, y es asignado a paquetes que no necesitan ningún tratamiento especial. Los paquetes marcados con este PHB son enviados tan pronto estén disponibles los recursos para su transmisión o procesamiento.

- **Expedited Forwarding (EF).** Este PHB implica que los paquetes deben ser tratados con baja latencia, baja perdida y bajo *jitter*. Para tal fin el tráfico perteneciente a este PHB tiene prioridad en el encolamiento sobre otro tipo o clase de tráfico. Para que no exista congestión debido a altos volúmenes de paquetes EF, son necesarias políticas de admisión estrictas que limiten la cantidad de tráfico que puede ser codificado con este PHB. El IETF define estos PHB en el RFC 3246.

- **Assured Forwarding (AF).** Los PHB de este grupo permiten controlar la entrega de paquetes en donde se determinan unas cuotas de uso justo para cada clase. De presentarse congestión los paquetes tienen una probabilidad de ser descartados, según la prioridad del código DSCP. Los PHB también son divididos en cuatro clases con diferentes prioridades, cada clase tiene códigos que determinan la probabilidad de ser descartados en una congestión. El IETF define estos PHB en los RFC 3260 y 2597.

- **Class Selector (CS).** Anteriormente se utilizaba el campo *Precedence* en el campo *Type of Service* de IPv4 para determinar prioridad. Estos PHB son utilizados para mantener compatibilidad hacia dispositivos antiguos.

- **Voice Admit (VA).** En el RFC 5865 se define este PHB y básicamente tiene el mismo principio de funcionamiento que EF aplicado a tráfico de llamadas de voz (VoIP).

El IETF también define unas reglas para la asignación de codigos a los PHB en el RFC 3140, en resumen un PHB especifica el tratamiento por cada enrutador dentro de un dominio DiffServ (DS). Se denomina un dominio DS a un grupo de enrutadores contiguos los cuales trabajan con una política de servicio común implementada en cada enrutador. Todos los enrutadores tienen reglas de redireccionamiento basadas en los valores de DS de los paquetes, los cuales, son comparados con su correspondiente valor de PHB. En caso de no poseer ningún tratamiento de QoS el campo DSCP será igual a 0. Los límites de la región la determinan los enrutadores o host de frontera, estos son los encargados de clasificar los paquetes entrantes en una determinada clase y se aseguran que estén correctamente etiquetados usando PHB por todo el dominio. Un dominio DS generalmente consiste en una red o un conjunto de redes que utilizan los mismos códigos PHB. Un domino también se consideran como una unidad administrativa (E.j. Un ISP[10]) [7].

Ahora bien se denomina una región DS como un conjunto de dominios de DS que aseguran las rutas de servicios por todas las redes y dominios que abarca. Cada dominio dentro de una región puede contar con una definición y mapeo igual o diferente de PHB, de ser diferente es necesario el uso de condicionadores de tráfico hacer las respectivas traducciones de los mapeos y definiciones de PHB.

[10] Internet Service Provider

DiffServ, es una de las técnicas utilizadas para proveer QoS sobre redes MPLS. Las redes MPLS según el tipo de tráfico asocian una etiqueta y por ende un camino dentro de la red; adicionalmente asigna recursos de ancho de banda según las políticas de administración de recursos y QoS. La principal desventaja de las soluciones basadas en DiffServ es la falta de homogeneidad en la implementación entre ISPs. Cada ISP utiliza mapeos de PHB distintos reduciendo o eliminando las prestaciones de QoS interdominio. La falta de concertación entre ISPs en un código único y clasificación única de paquetes para QoS, tanto en MPLS como en IPv4 o IPv6 resulta ser el principal obstáculo de las soluciones basadas en DiffServ [4]. Aunque DiffServ pierde granularidad, gracias a clasificar los tipos de servicios de forma general, su escalabilidad aumenta, y los requerimientos en el procesamiento disminuyen convirtiéndola en una solución deseable dentro de las redes que presenten altos volúmenes de tráfico como las redes *backbone*.

2.3.2.2 DiffServ sobre IPv6. El campo ECN

El campo ECN o Notificación Explicita de Congestión permite identificar en que segmentos de red existen congestiones, permitiendo nuevas funcionalidades como un enrutamiento inteligente basado en estas características.

El campo ECN como se describió anteriormente está definido en el RFC 3168. ECN utiliza una codificación simple a través de 4 diferentes códigos que se explican a continuación.

Código	Funcionalidad
00	Indica que un nodo no está utilizando o no tiene habilitado el uso de ECN.
01/10	Estos dos códigos tienen el mismo tratamiento indicando que el ECN está habilitado entre las dos nodos (receptor-transmisor). El código 01 es llamado ECT(0) y el 10 ECT(1).
11	Indicador de congestión, un enrutador envía este código para notificar que está presentando congestión.

Tabla 2-2 Codificación de ECN definido en el RFC 3168.

El uso de este campo permite no solo el desarrollo de protocolos de enrutamiento dinámicos más inteligentes, sino que, un enrutador antes de que llegue a congestión y empiece a descartar o perder paquetes, notifica la congestión para que disminuya la pérdida de paquetes.

2.4 Algoritmos de gestión y atención de colas.

Antes de definir los algortimos, es necesario identificar los principales valores de desempeño en un algortimo de gestión de QoS, entre estos están: Retardo (*Delay*) y Variación del retardo (Jitter). El primero define el tiempo que un paquete utiliza en llegar de un punto de referencia hacia un destino específico. El *jitter* definido por el RFC 3393 hace referencia a que tan variable es el retardo de un conjunto de paquetes con un origen y destino iguales. Para el cálculo de este valor se utiliza la fórmula de la ecuación 2-1, definida por el RFC 3393.

$$Jitter_i = Jitter_{i-1} + \frac{(|\Delta t| - Jitter_{i-1})}{16} \qquad \text{(Ec 2-1)}$$

Donde $|\Delta t|$es la diferencia de retardo entre paquetes, y el Jitter ($Jitter_i$) depende del valor anterior $Jitter_{i-1}$.

2.4.1 Weighted Fair Queueing (WFQ)

Los algoritmos de tipo Fair Queueing (figura 2-5) son diseñados para obtener un reparto equitativo o justo de los recursos de red de forma que ninguna cola tenga prioridad en la atención sobre otra. Los paquetes son divididos y organizados en cada cola según el esquema de admisión y clasificación que se especifique. El método de atención se realiza a través del método Round Robin (un paquete a la vez por cada cola), sin embargo un algoritmo Fair Queueing puro solo puede reservar la misma cantidad de ancho de banda para cada cola y está pensado para que los paquetes sean del mismo tamaño, situación que muy rara vez es cierta.

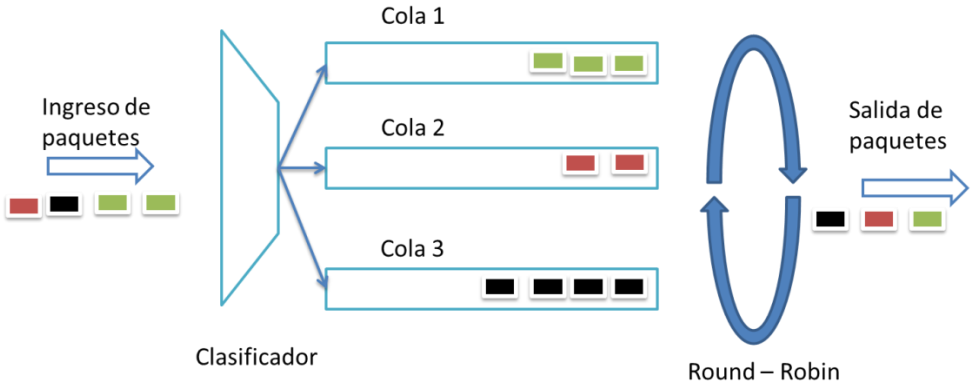


Figura 2-5 Algoritmo de atención de cola Fair Queueing – Round Robin[11]

El algoritmo WFQ es una modificación que busca mejorar el desempeño buscando diferenciar los servicios en términos de ancho de banda a través de pesos garantizando una tasa de transferencia definida por la ecuación 2-2, donde N es el número de colas o flujos donde a cada cola se le asigna un peso ω_i y c la capacidad de transmisión del enlace. Cada cola recibirá mínimo una tasa de transferencia R_i.

$$R_i = \left(\frac{\omega_i}{\sum_{n=1}^{N} \omega_n}\right) c \qquad \text{(Ec 2-2)}$$

El retardo D_i en este algoritmo está determinado en función de la tasa de transferencia R_i y la longitud de la cola B_i según la ecuación 2-3.

$$D_i \leq \frac{B_i}{R_i} \qquad \text{(Ec 2-3)}$$

La desventaja de este esquema es la complejidad de proceso al aumentar el tiempo que utiliza según el número de flujos que procesa

2.4.2 Deficit Round Robin (DRR).

El algoritmo WFQ presenta dificultades como la alta complejidad y el consumo de procesamiento (*O(logN)*). Deficit Round Robin (DRR) surge ya que en algunas aplicaciones, la distribución del ancho de banda tiene más importancia que mantener un retardo bajo sin tener un costo en el procesamiento demasiado alto (*O(1)*)[13].

DRR está compuesto por tres factores claves: un quantum Q_i, que representa la cantidad de bytes asignados a una cola *i* para la transmisión dentro de un esquema Round Robin; el contador

[11] Autoría propia.

de déficit C_i^{def}, el cual lleva el registro de la diferencia entre el número de bytes que pueden ser transmitidos en una cola *i* y el número de bytes que actualmente han sido transmitido de la cola *i* en cada ciclo; finalmente, el tamaño del buffer B_i para la cola *i*, indica el tamaño en bytes de la sumatoria de los paquetes encolados.

En cada ciclo durante la secuencia *round robin* el algoritmo en cada cola *i* de transmisión **no vacía**, asigna la cantidad de bytes (créditos) determinada por $Q_i + C_i^{def}$ y transmite todos los paquetes cuyo sumatoria de tamaño cumplan la desigualdad de la ecuación 2-4. Si el buffer B_i es mayor que $Q_i + C_i^{def}$, se transmiten solo los paquetes que no superen la desigualdad (B_i'), los bytes restantes son asignados a C_i^{def} (Ec 2-5), en la siguiente ronda para la misma cola *i* la nueva cantidad de bytes de transmisión es de terminada por la ecuación 2-6.

$$B_i \leq Q_i + C_i^{def} \qquad \text{(Ec. 2-4)}$$

$$C'^{def}_i = \left(Q_i + C_i^{def}\right) - B'_i \qquad \text{(Ec. 2-5)}$$

$$Cantidad\ para\ transmision = Q_i + C'^{def}_i \qquad \text{(Ec. 2-6)}$$

Reemplazando 2-5 en 2-6.

$$Cantidad\ para\ transmision = Q_i + \left(\left(Q_i + C_i^{def}\right) - B'_i\right) \qquad \text{(Ec. 2-7)}$$

Si después de la transmisión la cola *i* queda vacía el contador de déficit C_i^{def} toma el valor de cero. En la figura 2-6 se puede observar un ejemplo del funcionamiento de DRR.

Si el quantum Q_i es diferente para cada cola permite determinar diferentes porcentajes de ancho de banda de la salida a cada cola, teniendo un comportamiento similar a WFQ sacrificando tiempos de retardos por un procesamiento menos complejo.

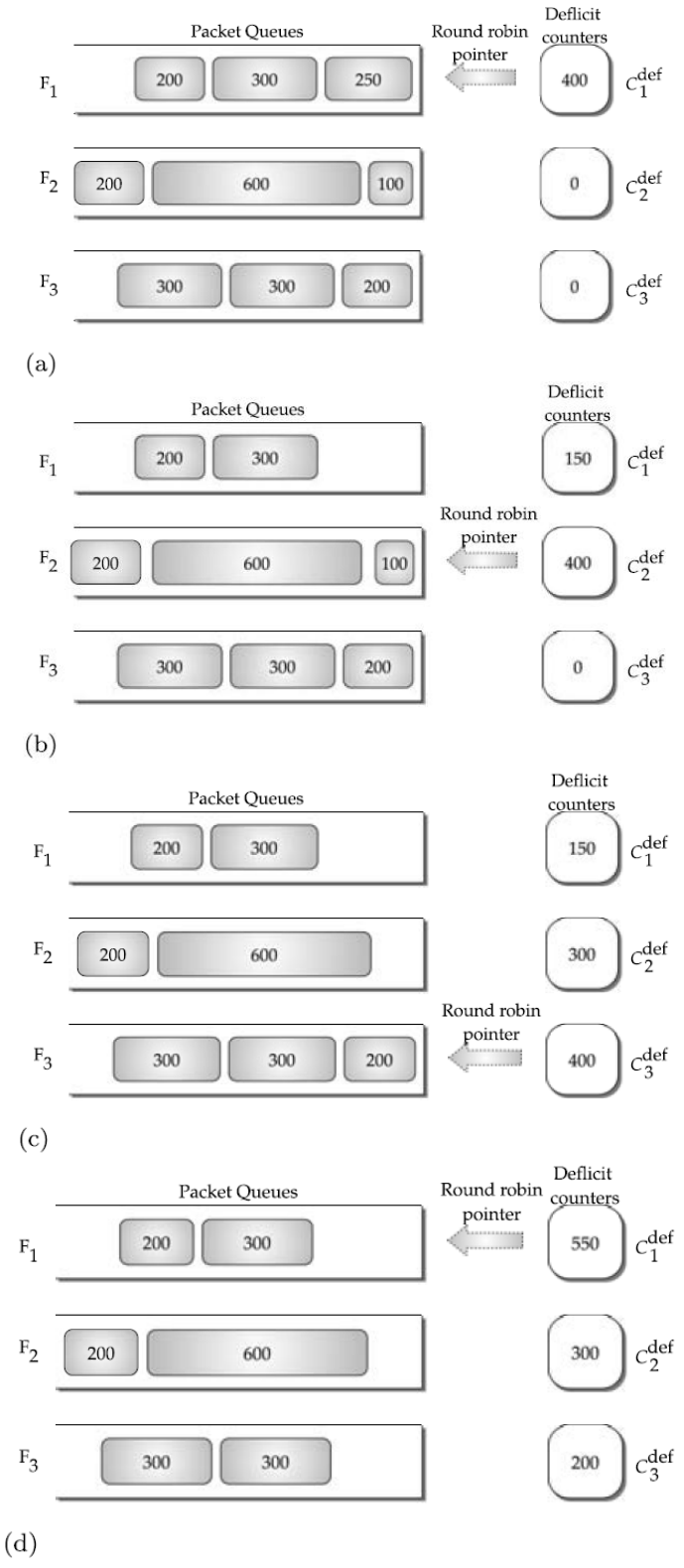


Figura 2-6 Ejemplo algoritmo Deficit Round Robin[12]

[12] Imagen tomada de [13]

En la figura 2-6(a) se asigna el quantum a la primera cola F1 y se transfieren los paquetes que no superen el valor del quantum asignado, hasta que el quantum llegue a cero o como en este caso queda un déficit al no haber la suficiente cantidad para enviar otro paquete. El valor de déficit es almacenado. En la figura 2-6 (b) y 2-6(c) se repite el mismo procedimiento de la 2-6(a) para las colas F2 y F3. Finalmente en la figura 2-6(d) al volver a la cola F1 se suma el valor de quantum al déficit de la ronda anterior y se transiten los paquetes que sigan cumpliendo la condición de la ecuación 2-7.

2.4.3 Weighted Random Early Detection (WRED)

Random Early Detection (RED) es uno de los algoritmos utilizados para evitar la congestión en Internet descartando un paquete TCP que permite reducir el efecto *burst* presente en las transmisiones TCP. Al ser descartado TCP reduce la transmisión para que el buffer de un nodo no se desborde.

RED determina la congestión mediante un estimador que calcula la media de ocupación de una cola cada vez que un paquete ingresa a dicha cola. Se designan dos umbrales Q_{min} y Q_{max} . Estos valores son comparados con la estimación de la media de ocupación Q_{avg} , el cual es calculado mediante un filtro paso bajas con factor de ponderación wq que evita los efectos burst de TCP (Q_{inst}) según la ecuación 2-8, donde $0 \leq wq \leq 1$.

$$Q_{avg} = (1 - wq)Q_{avg} + (wq * Q_{inst}) \qquad \text{(Ec. 2-8)}$$

Si el paquete arriba a una cola vacia el Q_{avg} es calculado según las ecuaciones 2-9 y 2-10.

$$Q_{avg} = (1 - wq)^m Q_{avg} \qquad \text{(Ec 2-9)}$$

$$m = \frac{tiempo\ actual - tiempo\ de\ arribo\ del\ ultimo\ paquete}{0.001} \qquad \text{(Ec. 2-10)}$$

Una vez calculado el Q_{avg} pueden presentarse tres casos

- **Q_{avg} es menor que Q_{min}:** Este caso se considera el estado de operación normal, ningún paquete es descartado y se encola.
- **Q_{avg} es mayor que Q_{min} y menor que Q_{max}:** En este caso los paquetes pueden ser descartados según la probabilidad de descarte P_d que se determina en función de la ocupación media de la cola y la probabilidad máxima de descarte P_{max} (definida por

el usuario) como se expone en la ecuación 2-11. A medida que Q_{avg} aumenta la probabilidad de descarte aumenta

$$P_d = P_{max} * \frac{Q_{avg} - Q_{min}}{Q_{max} - Q_{min}} \qquad \text{(Ec. 2-11)}$$

- Q_{avg} **es mayor que** Q_{max}**:** En este estado el paquete es siempre descartado.

El comportamiento de estos tres estados se observa en la figura 2-7.

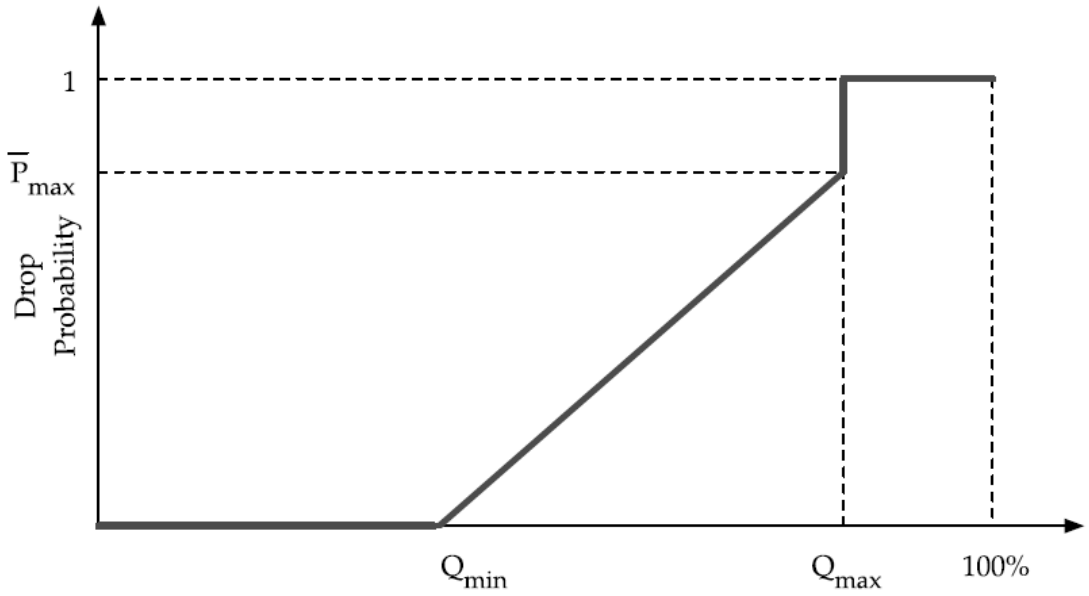


Figura 2-7 Curva de relación entre Qavg, Qmin, Qmax y Pd.[13]

Weighted Random Early Detection (WRED)establece diferentes tipos de probabilidad de descarte para cada tipo de tráfico, es decir, si un paquete arriba a una cola puede ser sometido a diferentes probabilidades de descarte según al perfil que pertenezca. Cada perfil posee una curva de relación que cuenta con sus propios Q_{avg}, Q_{min}, Q_{max} y P_{max}, como se ve en la figura 2-8

[13] Imagen tomada de [13]

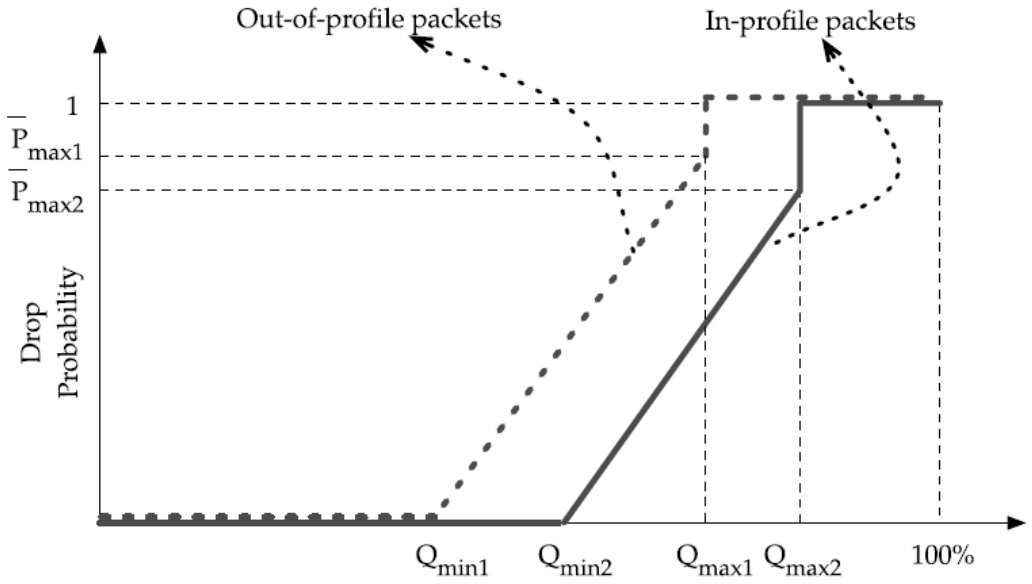


Figura 2-8 Ejemplo de curva de relación para 2 diferentes perfiles[14]

Los perfiles son útiles para definir el comportamiento de flujos de información pues hay flujos sensibles al descarte los cuales se les desea una baja probabilidad de descarte durante una congestión. un ejemplo de aplicación de encolamiento aplicando WRED se puede ver en la figura 2-9.

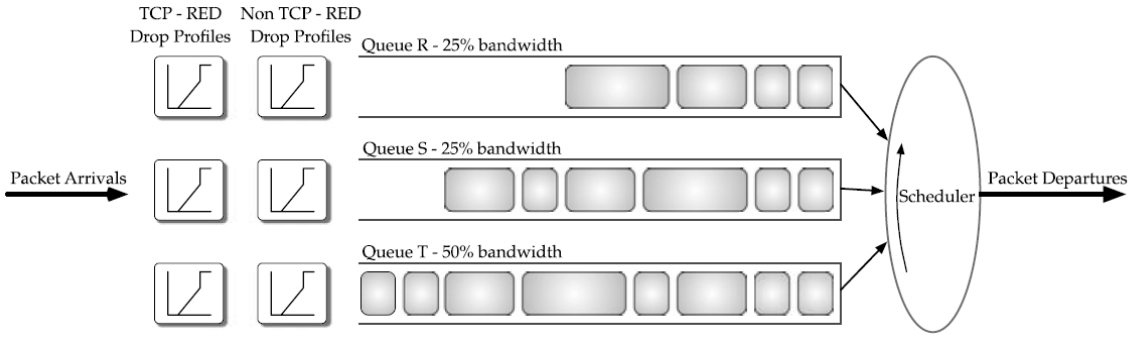


Figura 2-9Ejemplo de encolamiento aplicando WRED.[15]

[14] Imagen tomada de [13]
[15] Imagen tomada de [13]

3 ESPECIFICACIONES

El agente desarrollado en este proyecto aprovecha las funcionalidades de las cabeceras IPv6 descritas anteriormente. El agente administrador de QoS consiste en un proceso adicional a los procesos de enrutamiento normales en un enrutador o nodo. Los procesos de enrutamiento no serán evaluados pues no sufren modificación alguna.

3.1 Funcionalidad del Administrador de QoS

La principal función del agente es realizar la diferenciación del tráfico según los códigos DSCP y Flow Label presentes en IPv6, realizar los procesos de admisión de paquetes según las disponibilidades de ancho de banda y disponibilidad de encolamiento, controlar flujos según disponibilidad de ancho de banda y existencia de congestión, finalmente definir la prioridad de encolado de cada paquete.

Los procesos de *IntServ* demandan demasiados recursos de procesamiento y tiempo en ser procesados. Su implementación dentro del backbone de un ISP de enrutamiento o conmutación resultaría contraproducente debido al gran volumen de flujos de datos.

3.1.1 Propuesta de ubicación en la red del Agente de QoS

El procesamiento por flujo implica un alto consumo computacional. Se debe buscar un punto donde el control y clasificado se realice sobre los puntos de red que presenten los menores flujos o demandas de tráfico. A su vez dentro de una red deben también hacerse el control y clasificado de todos los flujos para que la solución de QoS tenga sentido. El problema demanda una solución distribuida e hibrida que logre abarcar los puntos que presenten menores flujos de datos y puedan abarcar la totalidad de los flujos presentes en una red. Una solución hibrida implica manejar las aproximaciones IntServ y DiffServ dentro de un nodo. En este sentido se aplica IntServ en los puntos hacia el usuario clasificando los flujos y aplicándoles las políticas de QoS. Por otra parte, hacia el *backbone* de red aplicar DiffServ para no sobrecargar los nodos dentro de este.

Los puntos de red que cumplen estas características son los nodos de borde dentro de la red de acceso del operador o ISP. Sin embargo, los nodos de borde de cara a los usuarios finales son los que presentan menor tráfico y números de flujo. En la figura 3-1 se observa la arquitectura de red de un ISP y los nodos de borde próximos al usuario.

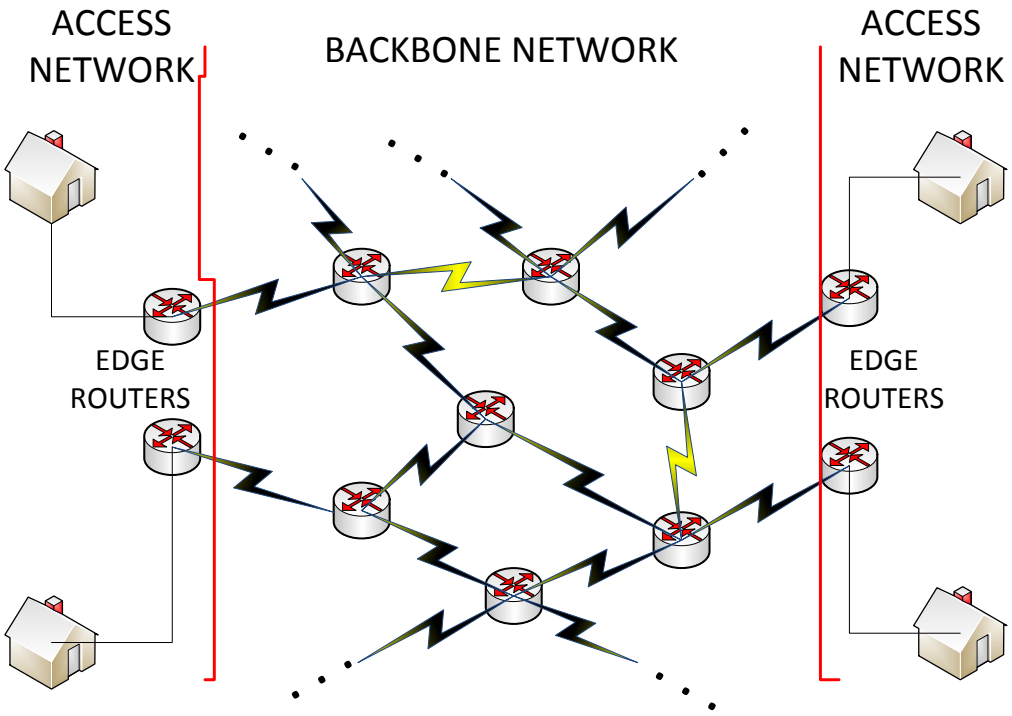


Figura3-1. Arquitectura de red ISP.[1]

Los nodos de borde en la red de acceso (edge routers) al tener la menor cantidad de flujo de toda la red, pueden realizar todo el proceso de admisión y control de los flujos de tráfico aplicando aproximaciones de IntServ al hacer análisis por flujo a través del agente administrador de Qos. El agente administrador de QoS, al realizar todo el proceso de control y admisión, asignará recursos de tal forma que el backbone sea utilizado de la forma más eficiente.

El algoritmo del agente administrador de QoS realiza todo el análisis de una solución de QoS con la salvedad que es únicamente utilizado para la clasificación y definición de flujos que ingresan al backbone. El backbone de red aplicará solamente una solución de diferenciación por clases (DiffServ) debido a su alto volumen de tráfico, sin embargo, su función es monitorear la ocupación de sus buffers y aplicar algoritmos de control de congestión.

3.1.2 Códigod DSCP para manejo de Differentiated Services

Para la aplicación de DiffServ en el backbone se utiliza la codificación DSCP según los PHB definidos en el RFC 2597 y RFC 3260 los cuales se listan en la tabla 3-1.

[1] Autoría Propia.

Probabilidad de Descarte	Clase 1 (LOWEST)	Clase 2 (LOW)	Clase 3 (MEDIUM)	Clase 4 (HIGH)
Baja	AF11 (DSCP 10)	AF21 (DSCP 18)	AF31 (DSCP 26)	AF41 (DSCP 34)
Media	AF12 (DSCP 12)	AF22 (DSCP 20)	AF32 (DSCP 28)	AF42 (DSCP 36)
Alta	AF13 (DSCP 14)	AF23 (DSCP 22)	AF33 (DSCP 30)	AF43 (DSCP 38)

Tabla 3-1 Tabla Codigos DSCP AF.

El algoritmo de control de congestión implementado dentro del modelo del enrutador de *backbone*, se basa en el campo *Explicit Congestion Notification* (ECN) de IPv6. Al presentar congestión el enrutador de *backbone* retornará un mensaje control con la más alta prioridad hacia los nodos de borde que le estén generando el tráfico [7]. En este mensaje el valor del campo ECN estará en valor 11 (binario) indicando la congestión. Los nodos de borde al recibir mensajes de congestión ajustan las políticas de clasificación y admisión de flujos de forma que se garanticen los servicios definidos con la más alta prioridad.

3.2 Diagramas de Estado y De Flujo

Cada flujo es generado por cada cliente que opera bajo el protocolo IPv6, el cual define en su campo Flow Label un identificador pseudoaleatorio para cada tipo de flujos y un código de tipo de flujo en el campo Traffic Class. El algoritmo de clasificación de QoS se describe en el diagrama de estados de la figura 3-2.

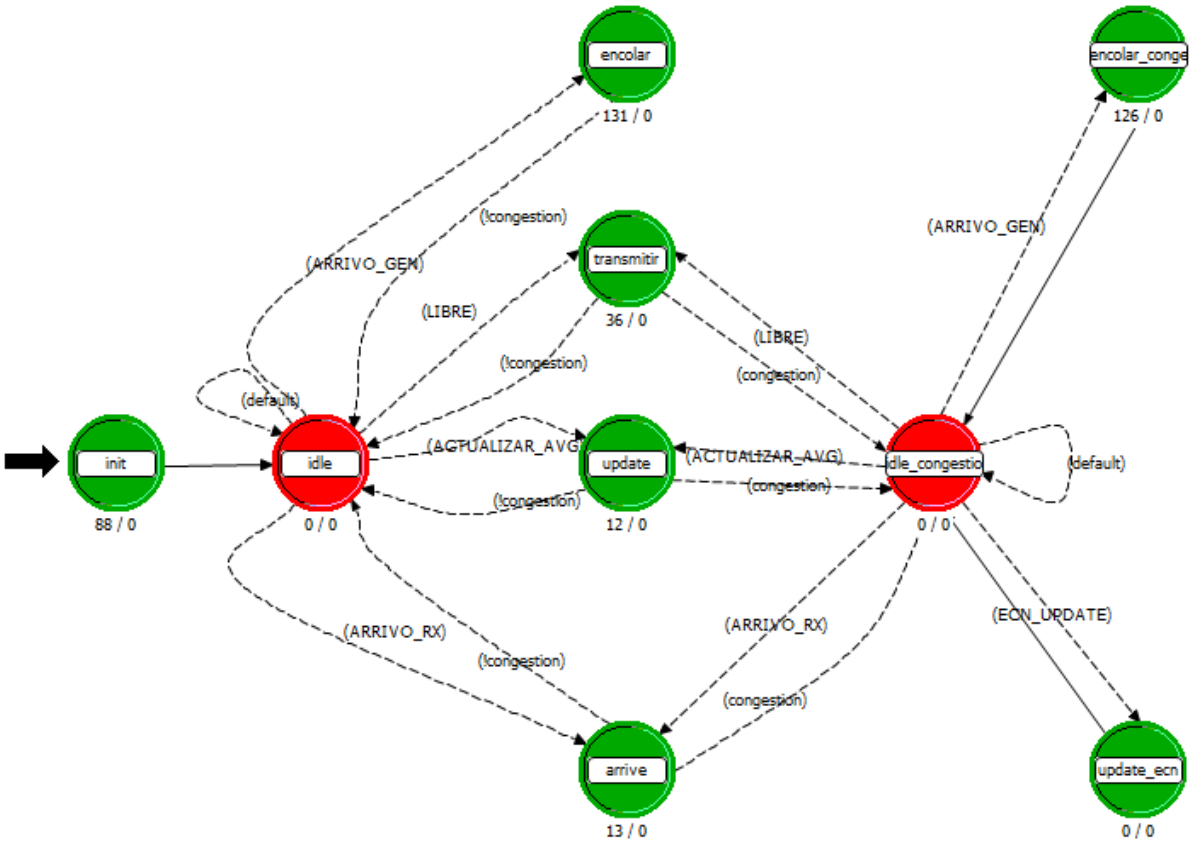


Figura 3-2. Diagrama de estados Agente administrador de QoS.

En un estado inicial (init) el algoritmo inicializa todas sus variables y recursos de memoria necesarios para la ejecución. Posteriormente pasa al estado de espera (idle) en el cual puede pasar a los siguientes estados:

Estado de arribo de paquete (encolar): En este estado se realiza el proceso de admisión de paquete según las disponibilidades de ancho de banda y disponibilidad de encolamiento y control de flujos, los flujos tiene un estado de tipo ***soft state*** según el RFC 6437, por lo cual el estado de los flujos activos son almacenados en una base de datos, la permanencia en la base de datos dependerá de temporizadores. Cada paquete perteneciente al mismo flujo reinicia el contador, al no recibir paquetes el nodo descartará el flujo de la base de datos. El diagrama de flujo de este estado se expone en la figura 3-3.

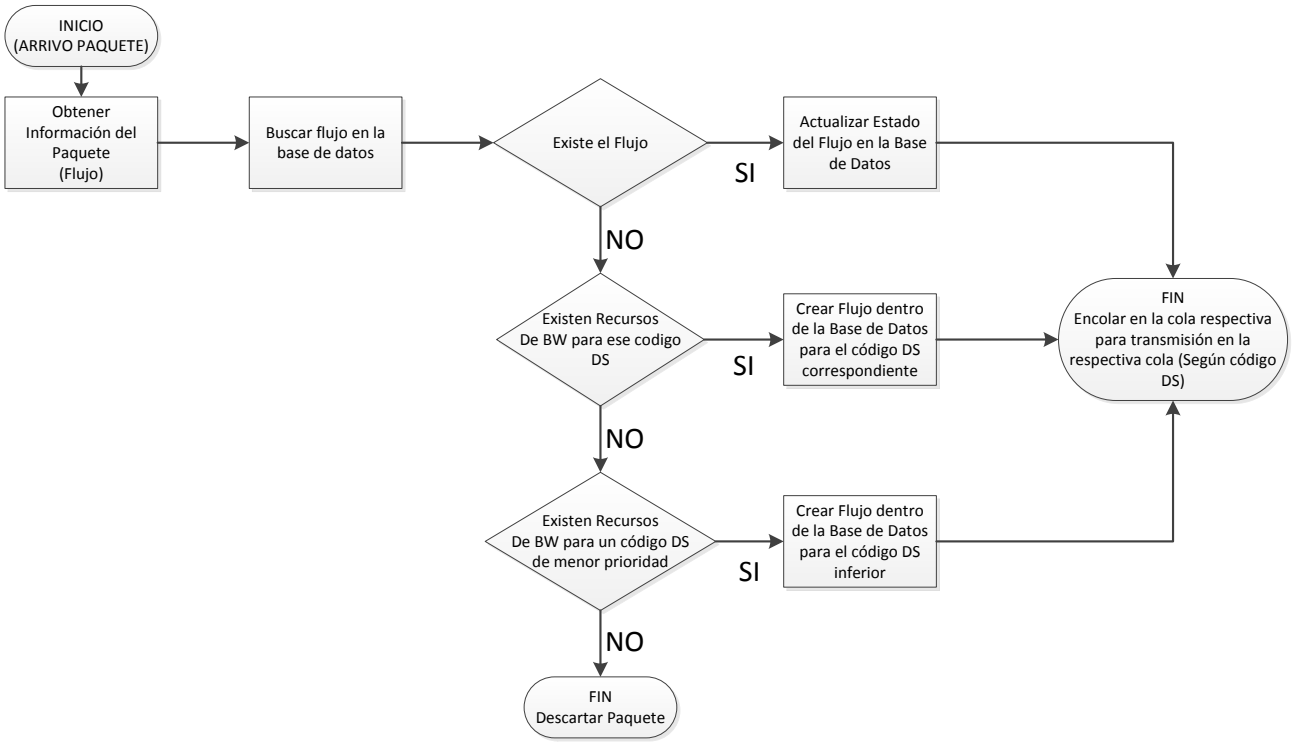


Figura 3-3 Diagrama de flujo del estado *encolar,* Agente administrador de QoS.

Estado de transmisión de paquetes (transmitir): En este estado los paquetes encolados son transmitidos hacia el siguiente nodo de red, cumpliendo con el esquema de atención de colas. Para este caso se implementó el algoritmo *Deficit Rount Robin* – DRR, con pesos quantums variables.[13] [14].

Estado actualización de estadísticas (update): En este estado se actualizan las estadísticas de tasas de transferencias de los flujos pertenecientes a los códigos DSCP de la misma prioridad. Adicionalmente revisa si algún flujo ya no está activo (el tiempo en que no recibe un paquete de dicho flujo es superior al timeout del flujo) y de no estarlo procede a eliminar su estado dentro de la base de datos.

Estado análisis de paquetes provenientes de la red (arrive): Al arribar un paquete desde el lado de la red backbone con destino a los usuarios finales, este estado analiza la cabecera IPv6 y consulta el estado del campo ECN, de existir congestión, envía el paquete hacia el destino y posteriormente cambia el estado a congestión, lo que obliga a pasar al estado de espera en congestión (idle_congestion) y realiza una reorganización de flujos. Si el estado ya está en congestión, se reinicia el timeout para salida de congestión.

La reorganización de flujos actúa sobre la base de datos, inicialmente la base de datos agrupa en grupos de códigos DSCP AF los flujos de un mismo usuario, una vez se presenta congestión, a cada usuario reduce de prioridad el ultimo flujo activo perteneciente a la clase más baja de las clases 2, 3 o 4, hacia la clase inmediatamente anterior. Ver figura 3-4.

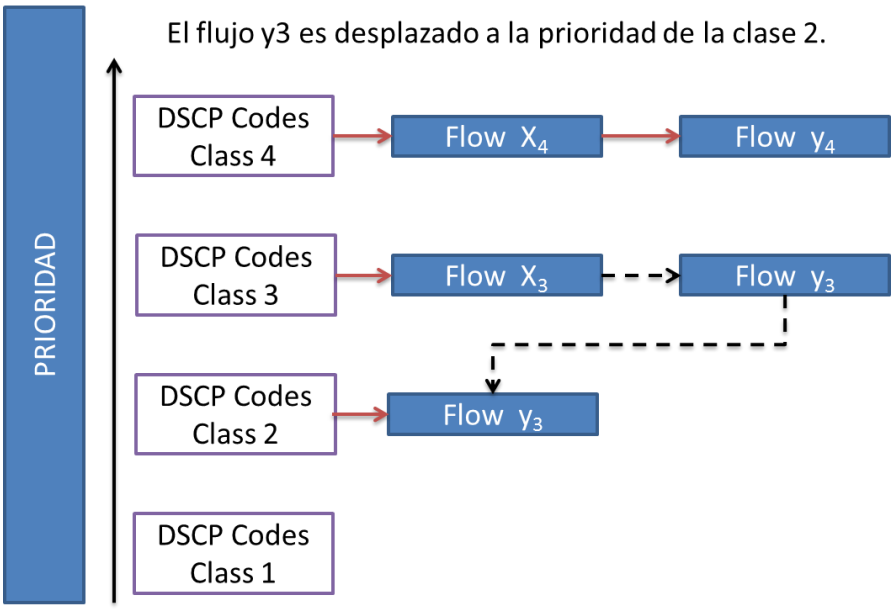


Figura 3-4 Ejemplo de reorganización de flujos dentro de la base de datos para un usuario.[2]

Estado de arribo de paquete en congestión (encolar_congestion): En este estado se realiza el mismo procedimiento que el estado encolar, con la diferencia que de no existir el flujo el paquete es inmediatamente descartado. En el estado de congestión no se permite la admisión de nuevos flujos.

Estado de actualización ECN (update_ecn): En este estado se revisa que aún no se haya cumplido el *timeout* de salida congestión para poder regresar al estado ***idle*** y volver a permitir nuevos flujo. De no cumplirse el *timeout* de salida de congestión, cada tiempo T_ECN (definido por el usuario) se realizaría una nueva reorganización de flujo en la base de datos.

La implementación del agente descrito anteriormente fue implementado en un entorno de simulación de eventos discretos, mediante la construcción de un modelo del agente administrador de QoS se evaluaron los parámetros de retardo, pérdida de paquetes y jitter; necesarios para la verificación de desempeño del agente diseñado. Posterior a la implementación y simulación del agente administrador de QoS, se procederá a comparar su desempeño frente un mismo escenario

[2] Autoría Propia

utilizando el protocolo MPLS, comparando variables de rendimiento como Jitter, retardo y porcentaje de paquetes perdidos frente a congestiones.

3.3 Restricciones del Modelo.

Las tasas de transmisión en los enlaces de conexión del nodo portador del agente administrador de QoS son *full-dúplex* simétricos. El presente modelo no incluye dentro de su alcance consideraciones de enlaces con tasas asimétricas.

Dentro del modelo se consideran usuarios con tasas de transmisión iguales en los enlaces hacia el nodo de borde que contiene el agente.

Se parte de la base que el nodo tiene ya asignado un conjunto de direcciones IPv6 ya definidas a cada usuario.

Todo el procesamiento es analizado en capa de red del modelo de OSI (capa 3), no se consideran algoritmos ni comportamiento de capas inferiores o superiores para las decisiones del agente, por lo cual se asume que no afecta el desempeño de dichas capas.

4 DESARROLLO DEL AGENTE ADMINISTRADOR DE QoS

Descrito el marco conceptual y las especificaciones del agente administrador de QoS el desarrollo e implementación del proyecto se describe a continuación.

4.1 Codificación de campos DSCP, ECN y Flow Label de IPv6.

Según las definiciones anteriormente descritas en la sección 2 del presente documento, los campos de la cabecera IPv6 que evaluará el agente administrador de QoS son el *Traffic Class* compuestos por el *DSPC Code* y el *ECN*; y el campo *Flow Label*.

Al existir diferentes definiciones PHB para DSCP, se analizaron las propuestas realizadas por la IETF, dentro de las cuales las más conocidas son: *Expedited Forwarding (EF), Assured Forwarding (AF), Class Selector (CS)* y *Voice Admit (VA)*. La red de tránsito de los paquetes, junto al agente administrador de QoS para realizar un buen esquema de diferenciación de flujos debe utilizar la codificación PHB que le brinde mayor diferenciación en los flujos por lo que se evalúo la pertinencia de cada codificación:

Expedited Forwarding (EF): Define baja latencia y *jitter*, sin embargo solo permite prioridad con un solo código, no tiene niveles intermedios de prioridad.

Class Selector (CS): Su codificación es pensada en la precedencia utilizada en equipos antiguos, los cuales no soportan el protocolo IPv6, razón por la cual es la que menos viabilidad de uso presenta.

Voice Admit (VA): Está pensada específicamente para tráfico de voz, lo que implica la misma desventaja expuesta en Expedited Forwarding (EF).

Assured Forwarding (AF): Define diferentes prioridades (4 Clases) para el manejo de paquetes, adicionalmente específica 3 probabilidades de descarte de paquetes. Esta diferenciación permite la diferenciación de más tipos de flujos, así como la posibilidad de manejar diferentes tasas de descarte en colas que resultan útiles para implementación de algoritmos de gestión de colas como WRED (Weighted Random Early Detection). Dadas estas características se escoge esta esta codificación para la diferenciación DSCP.

Los otros dos campos ECN y Flow Label son utilizados según la definición del RFC 3168 para ECN en materia de codificación y el RFC 6437 para *Flow Label*, el cual especifica una

etiqueta de longitud de 20 bits con un valor pseudoaleatorio, la condición de flujo como paquetes con el mismo *Flow Label, Source Addres* y *Destination Address* y la gestión por parte de los nodos (administración de estados de flujos activos).

4.2 Entorno de simulación

El agente administrador de QoS propuesto dentro en el presente trabajo es evaluado mediante sistemas de simulación. Dentro de las diferentes métodos de simulación se encuentran simulaciones de tiempo continuo, simuladores de por eventos discretos (DES). El primero tiene continuidad en el tiempo y generalmente utiliza aproximaciones y soluciones analíticas que no siempre son muy precisas [15]. La simulación basada en eventos discretos es el método de simulación típicamente utilizado para el estudio de desempeño de una red o protocolos de red, puede crear modelos en extremo detallados, descripción paquete a paquete para que todos los efectos que se evalúen. DES se considera preciso para proveer análisis de manejo de colas en redes, desde un algoritmo simple a un algoritmo complejo[15]. El criterio anteriormente descrito es el criterio para el método de evaluación del agente administrador de QoS propuesto.

Entre los principales simuladores que utilizan DES se encuentran: NS-2, OMNeT++ y OPNET. Este último sistema permite la implementación de nuevos modelos y protocolos a través de lenguajes como C y C++. Además incluye una librería con modelos de diferentes dispositivos de red entre estos dispositivos que utilizan MPLS ya validados, razón por la cual resulta ideal para el cumplimiento del último objetivo del actual proyecto, por la tanto OPNET es el simulador escogido para realizar el estudio del desempeño del agente administrador de QoS.

4.2.1 Entorno de simulación de OPNET Modeler.

El software de simulación plantea tres dominios de desarrollo para la evaluación de sistemas de redes de telecomunicaciones [17]. En el dominio de red se plantean las topologías de red que se van a evaluar, se definen redes, subredes, tipos de nodos, enlaces incluso un contexto de ubicación gráfica, el trabajo sobre este dominio se realiza a través del editor de proyecto. (Ver figura 4-1).

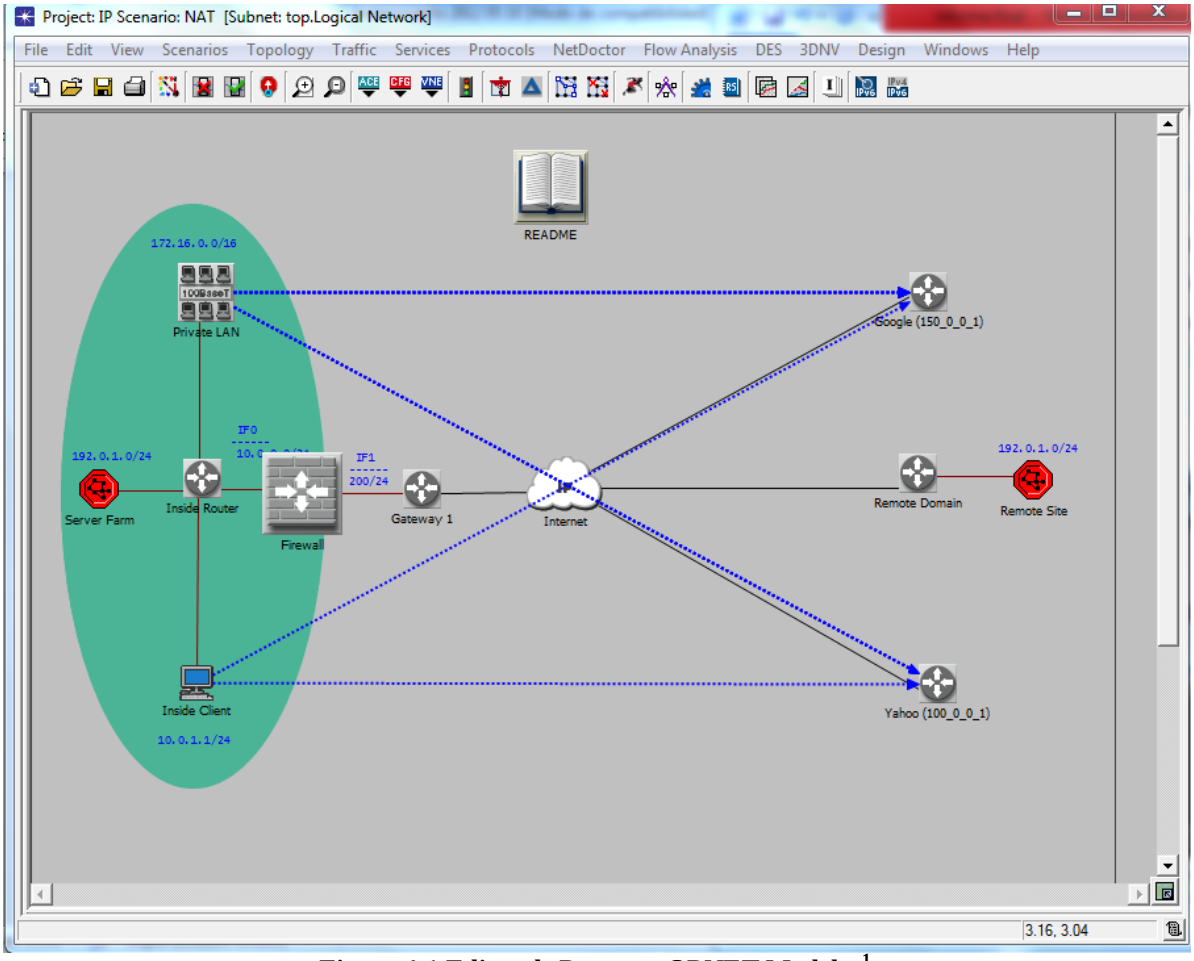


Figura 4-1 Editor de Proyecto OPNET Modeler[1].

En el dominio de nodo se define la arquitectura interna de elementos funcionales o procesos y flujos de datos o estadísticas entre cada elemento funcional, el desarrollo sobre este dominio se realiza sobre el editor de nodo (ver figura 4-2).

El último dominio de desarrollo es el de proceso a través del editor de proceso, en el cual se define el comportamiento de un elemento funcional o proceso, como lo pueden ser protocolos, algoritmos, aplicaciones, etc. Este comportamiento es definido utilizando máquinas de estados finitos y lenguaje de alto nivel (C, C++ y Proto C) [15]. (Ver figura 4-3).

OPNET incluye otros tipos de editores que complementan el desarrollo de topologías de red, como editores de enlace, editores de formato de paquetes, editores de funciones de distribución de probabilidad entre otros que no serán especificados en el presente documento y pueden ser consultados en [17].

[1] Imagen tomada de [17] escenario de ejemplo de red IPv4.

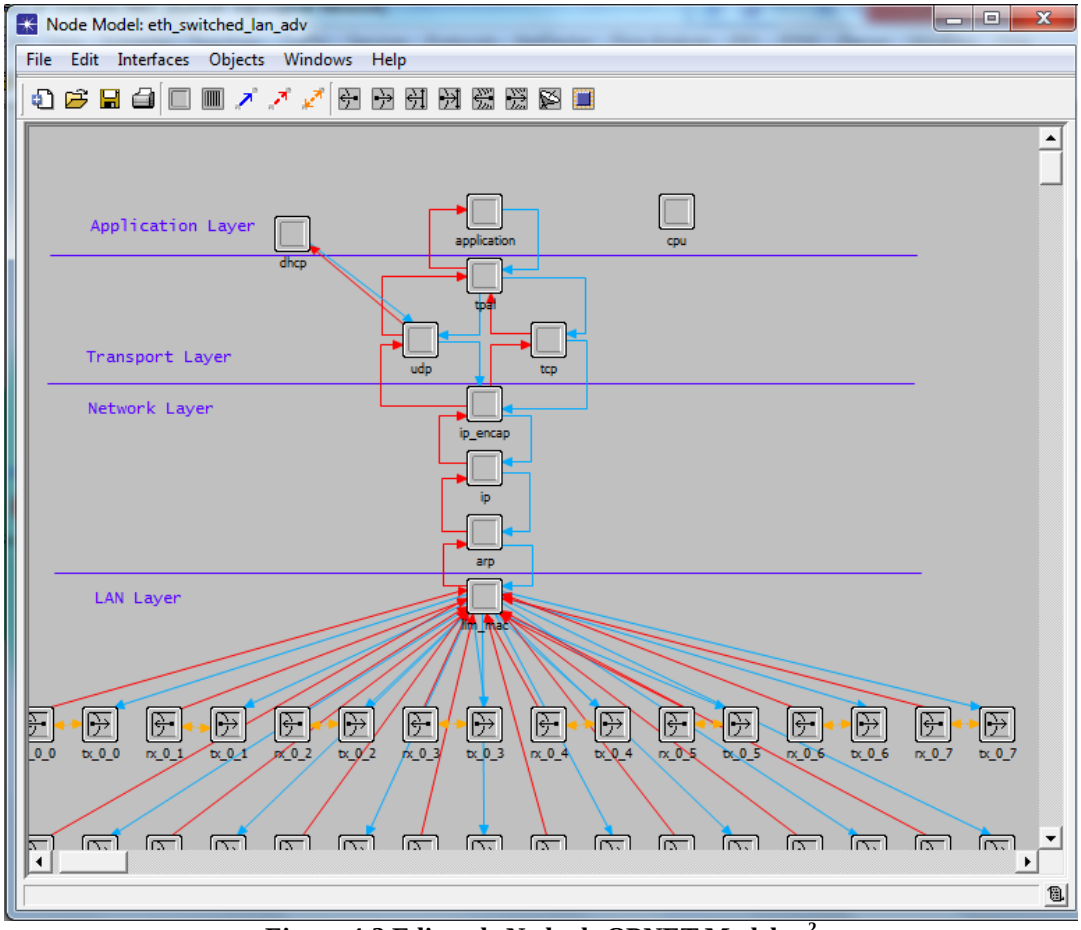


Figura 4-2 Editor de Nodo de OPNET Modeler.[2]

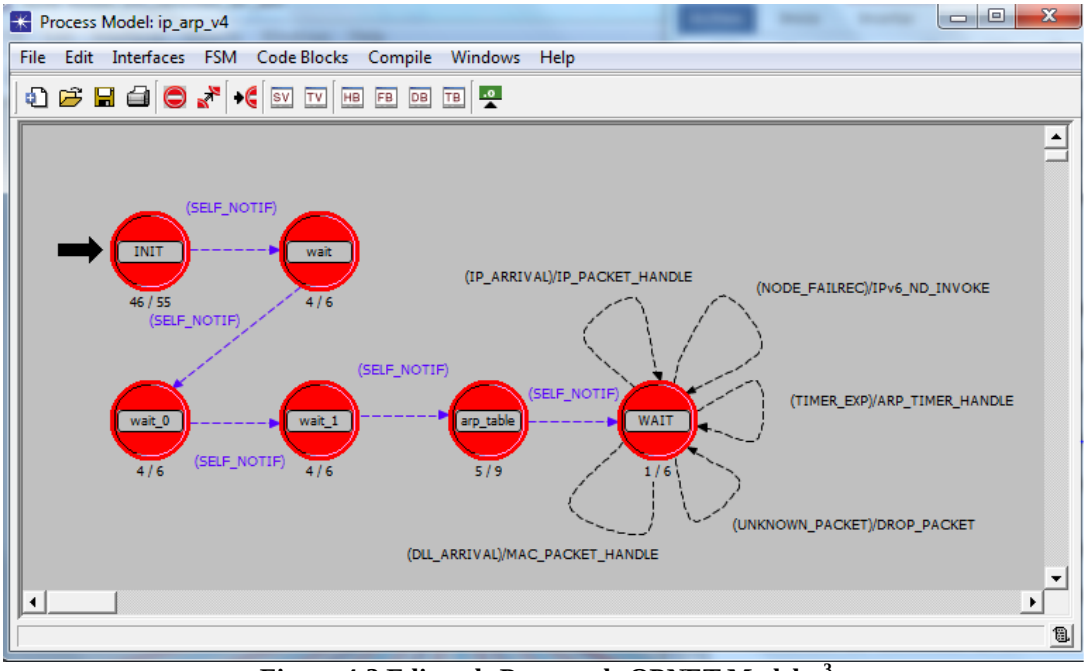


Figura 4-3 Editor de Proceso de OPNET Modeler[3]

[2] Imagen tomada de [17] modelo de nodo de una red LAN.
[3] Imagen tomada de [17] modelo de proceso del protocolo ARP

4.3 Metodología de desarrollo.

El proceso de diseño de cada componente de los modelos de nodo y el Agente administrador de QoS sigue la metodología de diseño planteada en la figura 4-4.

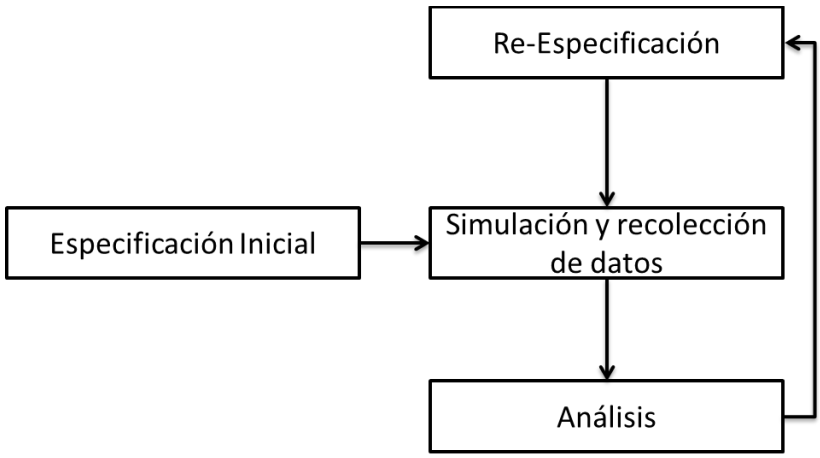


Figura 4-4 Metodología de diseño de nodos en OPNET Modeler

En tal metodología cada proceso y función es evaluada y analizado su desempeño basado en la especificación inicial. Si la conclusión del análisis del modelo no corresponde con la especificación se re específica y se implementa nuevamente repitiendo el ciclo. De ser exitoso el análisis basado en la simulación se da como validado el modelo de proceso dentro de un modelo de nodo.

4.4 Algoritmo de control de ajuste de políticas de aprovisionamiento de recursos de QoS.

El agente administrador de QoS basa su funcionamiento en el algoritmo de control de políticas y recursos de QoS. El algoritmo administra recursos según las políticas de aprovisionamiento con el fin de descongestionar el *backbone* y mantener mejor QoS a los flujos de más alta prioridad que son generados por cada usuario. Este sistema de control que es el núcleo fundamental del agente administrador de QoS pretende realizar una mejora en casos de congestión de la red *backbone*.

4.4.1 Variables de entrada del sistema.

El sistema de control de políticas de aprovisionamientos de recursos de QoS actúa según las variables de entradas al sistema de control de políticas, cada una de las entradas las originan diferentes actores determinando el comportamiento del sistema, a continuación se definen cada una de las entradas y su respectivo actor generador.

4.4.1.1 Entradas de configuración inicial.

Las entradas de configuración inicial son las variables que definen las características, capacidades y recursos del agente administrador de QoS, dentro de las cuales se definen:

- Capacidad promedio de las colas presentes en el nodo, expresada en bits.
- Tasa de transmisión del enlace de salida (full-duplex) hacia la red *backbone*, expresada en bits por segundo.
- Tasa de transmisión del enlace de salida (full-duplex).
- Parámetros para configuración del algoritmo de gestión de colas
- Parámetros para configuración del algoritmo de atención de colas.

El actor generador de estas entradas es el administrador de red, que durante la etapa de puesta en marcha del nodo define tales variables. Este proceso es simulado en el proceso durante el estado init donde son tomadas estas variables y cargadas al agente dentro de sus variables de estado.

4.4.1.2 Paquetes e información de cabecera.

El sistema de control de ajuste de acuerdo a la información obtenida de los paquetes que recibe y las variables de estado propias, todos los paquetes de entrada son bajo el formato de cabecera IPv6, el sistema obtiene las variables de entradas correspondientes para la identificación de flujos y aplicación de políticas según el actor que los genere que pueden ser:

1. Paquetes recibidos por el enlace conectado hacia la red *backbone*.
2. Paquetes recibidos por los enlaces conectados hacia los usuarios finales.

Los paquetes recibidos por el enlace conectado hacia la red *backbone* tienen como destino los usuarios finales, estos paquetes contienen la misma información de los flujos de salida, su principal diferencia es que incluye en el campo ECN información del estado de congestión en el *backbone*, esta última es la variable que analiza y cambia de un estado de operación normal a un estado de congestión. Por lo tanto el actor generador de esta variable son los nodos de la red *backbone* que presentan congestión.

Los paquetes recibos por los enlaces conectados hacia los usuarios finales contienen la información que caracteriza un flujo y su prioridad según codificación DSCP AF disponibles en

las cabeceras de IPv6 mencionadas en el numeral 4.1. Adicionalmente el sistema obtiene tamaño del paquete (cabecera IPv6 junto a su carga de datos), con el fin de determinar las estadísticas de tasas de transferencia unidireccional que se obtienen por cada prioridad.

4.4.2 Variables de estado

Las variables de estado contiene la información de los estados de los flujos y estado del sistema, determinadas en función de las variables de entrada. Las variables de sistema del sistema de control están compuestas por:

4.4.2.1 Tasa de transmisión cada clase AF.

La tasa de transmisión límite es la variable principal en el proceso de admisión y control de la tasa de transmisión de todos los flujos clasificados en la misma clase AF, para cada clase existe una variable de tasa de transmisión límite. Estas variables llevan registro de la tasa actual de transmisión en cada instante durante la ejecución del agente administrador de QoS y determina la admisión y control de paquetes de nuevos flujos que soliciten ingreso al sistema a través del estado encolar que será descrito más adelante.

4.4.2.2 Variables de estados de flujos activos

Para tener control de que flujos se encuentran activos sobre un nodo el sistema recopila información de las variables de entrada obtenidas de las cabeceras y organiza en una estructura de datos. Al existir diferentes variables como *IP Source Address*, *DSCP Code*, *IP Destination Address y Flow Label* se propone el siguiente esquema de organización: Como variable raíz en un proceso de búsqueda se escoge la variable *IP Source Address*, pues el agente al ser propuesto en un nodo de borde el número de direcciones es limitado según la capacidad de usuarios que tenga el nodo (el cual es mucho menor que el rango de direcciones que se puede encontrar en *IP Destination Address*). Posteriormente para poder tener separadas las diferentes prioridades o clases AF se hace dentro de cada *IP Source Address* separación por clases AF, ya dentro de cada clase agregamos el estado de cada flujo según su Flow Label, en la figura 4-5 se expone la estructura de almacenamiento de la estructura de datos propuesta para el manejo de estados de flujos. La definición de la estructura de la figura 4-1 se define en el código de la gráfica 4-1:

```
typedef struct{
   int destino; //Direccion IPv6 de Destino del flujo
   int flow_label;//Flow label IPv6 del flujo
   struct Flujo *next;// Apuntador a otra estructura de tipo flujo
   double last_time;//Ultimo tiempo de arribo de paquete de este flujo
   double init_time;//Tiempo de inicio de toma de estadística
   double datarate; //Tasa de transferencia del flujo.
   } Flujo;

typedef struct{
   Boolean flag_corrido; /*Indicador de que un flujo ha sido reducido de
   Prioridad*/
   Boolean flag_presente; /*Indicador de que un flujo se encuentra en este código
   proveniente de una prioridad más alta*/
   Flujo *inicio; //Apuntador al primer flujo perteneciente a este código.
   Flujo *ultimo; //Apuntador al último flujo perteneciente a este código.
   } DS_Codes;


typedef struct{
   int origen; //Direccion IPv6 de origen
   DS_Codes codigos[4]; //Apuntadores a estructuras DS_Codes
   double promedio_codigos[5]; /*Tasa de transferencia promedio en cada código AF*/
   struct IP_Database *next; /*Apuntador a la siguiente estructura de tipo IP_Database*/
   ColaNodo primero[5]; /*Estructuras de soporte a estadísticas del vector promedio
código*/
   ColaNodo ultim[5]; /*Estructuras de soporte a estadísticas del vector promedio código*/
   int salida; /*Interfaz de salida del paquete IPv6*/
   } IP_Database;
```

Gráfica 4-1 Código fuente de la estructura de datos propuesta para almacenamiento de estado de flujos IPv6.

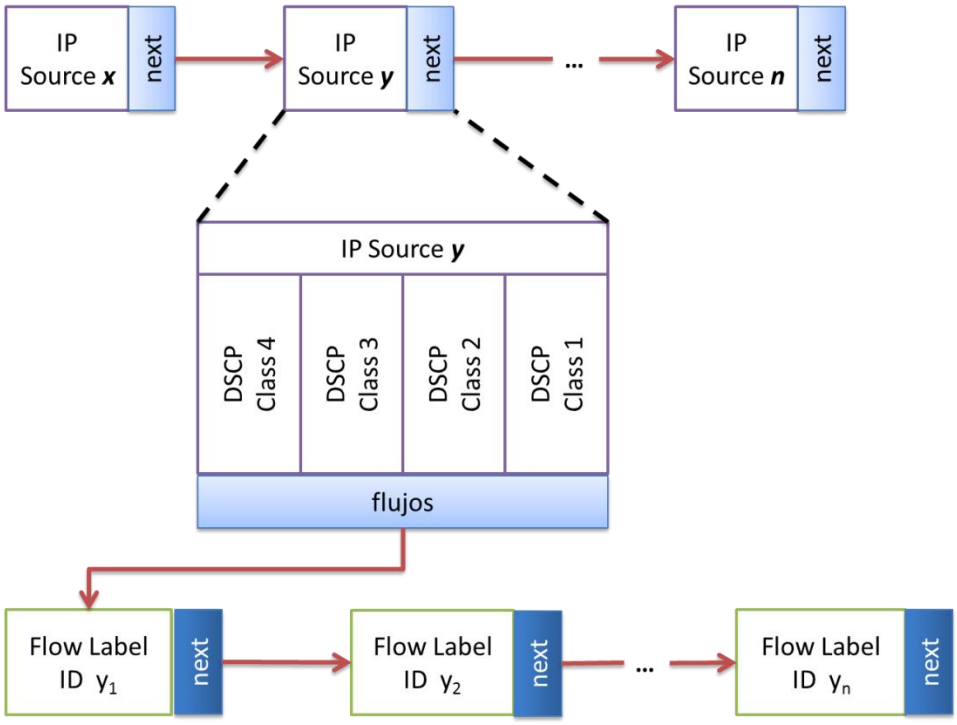


Figura 4-5. Estructura de datos propuesta para almacenamiento de estado de flujos IPv6.[4]

[4] Autoría Propia.

Los campos definidos como ***next*** y ***flujos*** de la figura 4-5 hacen referencia a los apuntadores hacia otras estructuras de datos según la gráfica.

Es importante señalar que para que un flujo coincida no solo debe cumplir con los 3 grupos anteriormente especificados sino también debe cumplir con la dirección del campo *IP Destination Address.*

4.4.3 Máquina de estados y funciones del sistema de control de recursos.

El sistema de control apoya su funcionamiento en procesos de ejecución basado en una máquina de estados que a su vez están soportados en funciones que ejecutan tareas específicas. El diagrama de estados del proceso de control diseñado se puede ver en la figura 4-6, esta máquina de estados basa su funcionamiento en los siguientes eventos:

- Arribo de paquetes por las interfaces de usuario (ARRIVAL_FROM_USERS)
- Evento de actualización de estadísticas y variables de estado (STATISTIC_EVENT)
- Arribo de interfaces por interfaz hacia el backbone (ARRIVE_FROM_BACKBONE)
- Eventos de tiempos cumplidos en estado de congestión (TIMEOUT)
- Congestión resuelta en el backbone (CONGESTION_AVOIDED)

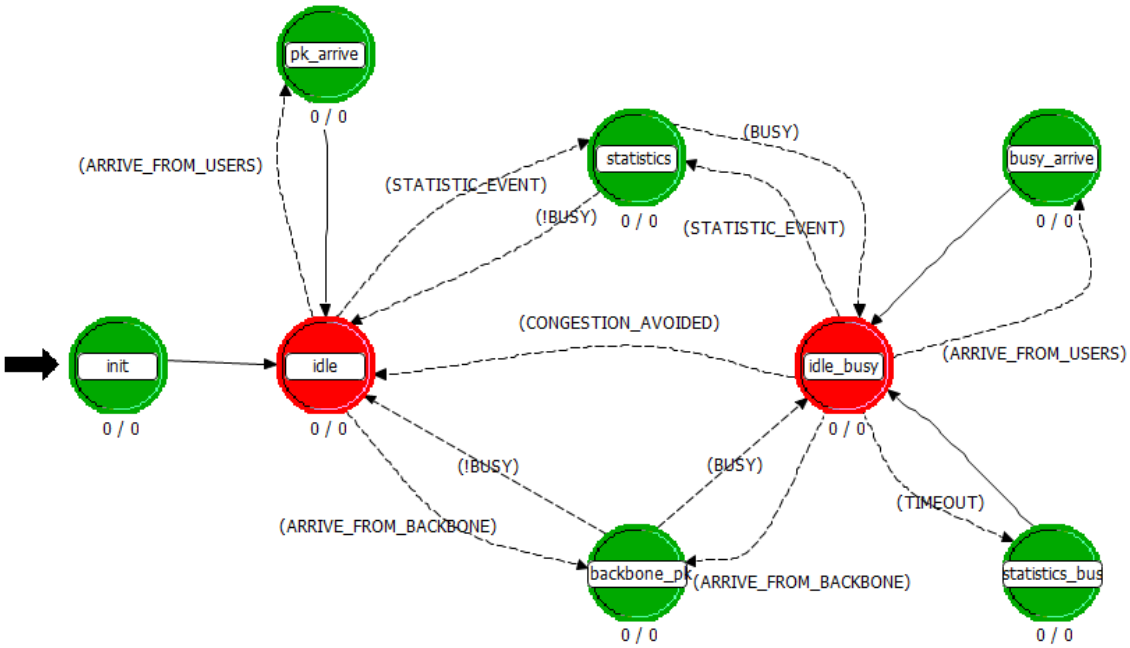


Figura 4-6 Maquina de estados del 4.3 del sistema de control de ajuste de políticas de aprovisionamiento de recursos de QoS.

Dichos eventos son generados por las funciones y procesos de cada estado en función de las entradas y variables de estado del sistema, cada estado con sus funciones de soporte cuya implementación pueden verse en el anexo A.1 son descritas a continuación.

4.4.3.1 Estados de inicialización y espera.

Se hace referencia al estado ***init*** el cual carga todas las condiciones iniciales de las variables de configuración inicial, una vez inicializado el sistema el sistema pasa al estado de espera ***idle*** el cual mantiene el sistema estático a la espera de eventos de arribo de paquetes (ARRIVAL_FROM_USERS y ARRIVE_FROM_BACKBONE), y eventos para actualización de estadísticas y variables de estado (STATISTIC_EVENT). De presentarse una notificación de congestión, el sistema pasa a un estado de espera ***idle_busy*** el cual interactúa con estados que determinan el control de recursos en congestión, el paso a otros estados se da a través de los eventos de arribo de paquetes (ARRIVAL_FROM_USERS y ARRIVE_FROM_BACKBONE), eventos para actualización de estadísticas y variables de estado (STATISTIC_EVENT) y eventos de tiempos cumplidos en estado de congestión (TIMEOUT). Una vez se determina que no hay congestión en el backbone es generado el evento CONGESTION_AVOIDED, permitiendo volver a un estado de espera de estado normal de sistema ***idle.***

Para la generación del evento, el sistema toma un periodo de tiempo T_{WTR} que puede ser definido por el usuario, tiempo en el cual el sistema espera que el backbone se estabilice, evitando así notificaciones ECN generadas por ruido en transiciones del backbone.

4.4.3.2 Estado arribo de paquete y aplicación de políticas de admisión.

Este estado hace referencia al estado ***pk_arrive*** de la figura 4-6. La funcionalidad de este proceso se ilustra en el diagrama de flujo de la figura 3-3 de la sección anterior, cada proceso de ese diagrama de flujo es soportado por las siguientes funciones

Obtener información del paquete (flujo): Proceso de obtención de la información de cabecera IPv6 (Dirección IPv6 de destino y origen, código DSCP, y Flow Label) y obtención de tamaño total del paquete. La función definida para tal fin es llamada:

```
static Packet_Info obtener_info_pkt (Packet *pkte);
```

Esta función recibe un paquete como parámetro y retorna una estructura de datos con la información del paquete, la cual es definida como:

```
typedef struct{
  int flowid;
  int dscode;
  int ip_dest;
  int ip_origen;
  double total_size;
  } Packet_Info;
```

Buscar flujo en la base de datos: Este proceso se basa en las funciones de búsqueda en la base de datos definidas como:

```
static int flujo_existente (Packet_Info *informacion, Flujo *info_flujo);

static Boolean existe (Packet_Info *informacion, DS_Codes *nodo, Flujo *pointer);

static Data_IP buscar_IP (int direccion);
```

Tales funciones realizan la búsqueda por todas las estructuras de la base de datos para determinar la existencia del flujo dentro de los flujos activos

Evaluación de ingreso de Flujo: Esta última parte se encarga de ejecutar el resto del proceso del diagrama de flujo de la figura 3-3, la cual está compuesta por la evaluación de la instrucción a ejecutar según el resultado del proceso anterior, que de ser positiva la búsqueda se retorna el apuntador a la estructura solicitada. De ser negativo el proceso de búsqueda el flujo es agregado si y solo si en un instante ***t*** un flujo ***pk(t)*** cumple la desigualdad de las ecuaciones 4-1 y 4-2:

$$pk(t) + r_i(t) = y_i(t) < limiteAF_i \quad \text{(Ec. 4-1)}$$

$$y_i(t) + \sum_{j \neq i}^{AF} r_j(t) < BW \quad \text{(Ec. 4-2)}$$

Donde $r_i(t)$ es la tasa de transferencia promedio en cada clase i de la codificación AF, $y_i(t)$ es la tasa de transferencia promedio de clase i incluyendo el nuevo flujo, $limiteAF_i$ es el límite de configuración para esa dirección IPv6, finalmente BW es la tasa de transmisión máxima del enlace del usuario.

De cumplirse únicamente la desigualdad de la ecuación 4-2 se procede a evaluar la ecuación con una clase de menor prioridad como se muestra en la ecuación 4-3 siendo 4 la prioridad más alta y 1 la más baja:

$$pk(t) + r_j(t) = y_j(t) < limiteAF_j \qquad \text{(Ec. 4-3)}$$

$$donde\ j < i$$

De no cumplirse las restricciones dadas por las ecuaciones anteriormente descritas el paquete y por ende el nuevo flujo no es admitido y por ende descartado. La evaluación de ingreso de flujo es realizada por la función:

```
static Boolean agregar_flujo (Packet_Info *datos, Flujo *info_flujo, int *colads);
```

4.4.3.3 Estado de actualización de estadísticas.

Al ser simulación de eventos discretos, las estadísticas deben actualizarse constantemente para que los datos sean válidos. Por tal motivo se genera interrupciones que generan el evento STATISTIC_EVENT que actualiza las estadísticas de tasas de transferencia de cada código AF.

Este estado adicionalmente depura la estructura de datos que soporta los datos de flujos inactivos eliminándolos de esta liberando memoria, al ser la depuración un proceso dispendioso de computo se ejecuta cada cierto periodo de tiempo.

La función de depuración esta definida como:

```
static void depurar (void);
```

4.4.3.4 Estado arribo de paquete desde el backbone.

El evento para pasar al sistema a este estado se da cuando un paquete arriba desde el backbone (ARRIVE_FROM_BACKBONE) donde es analizado el campo ECN y transmitido a otros procesos del nodo.

Según el valor del campo ECN, de ser 0b11 se establece el estado del sistema a congestión en el backbone y pasando al estado de espera en congestión ***idle_busy***, durante el estado de congestión este estado está monitoreando todos los campos ECN de los paquetes que arriban desde el backbone para establecer si la congestión ha desaparecido y activa un temporizador T_{WTR} , que monitorea el tiempo que ha pasado desde que se recibió el ultimo ECN, cada vez que arriba un paquete con ECN igual a 0b11, este temporizador es reiniciado.

4.4.3.5 Estado arribo de paquete y aplicación de políticas de admisión en congestión.

El estado definido como ***busy_arrive*** tiene la misma funcionalidad que el estado ***pk_arrive*** con la diferencia que al estar en congestion no admite ningún flujo que no este activo dentro de la base de datos de flujos activos.

4.4.3.6 Estado de actualización de temporizadores en congestión.

Este estado es ejecutado por el sistema cuando se recibe un evento de tipo TIMEOUT el cual ese generado cuando el sistema pasa a ***idle_busy*** que tiene como fin realizar la reorganización de flujos especificada en la sección 3 figura 3-4, con motivo de reducir congestión y dar prioridad a los flujos de más alta prioridad mediante las funciones:

```
static void reorganizar_IP (Data_IP x);
static void reorganizar_db (void)
```

4.5 Modelo Final Agente Administrador de QoS.

Al no ser considerados criterios de enrutamiento ni encapsulación, el agente administrador de QoS implementa un algoritmo de atención de colas DRR con quantums variables y esquemas de gestión de colas WRED. La máquina de estados del agente incluyendo DRR y WRED se especifica en la figura 4-7.

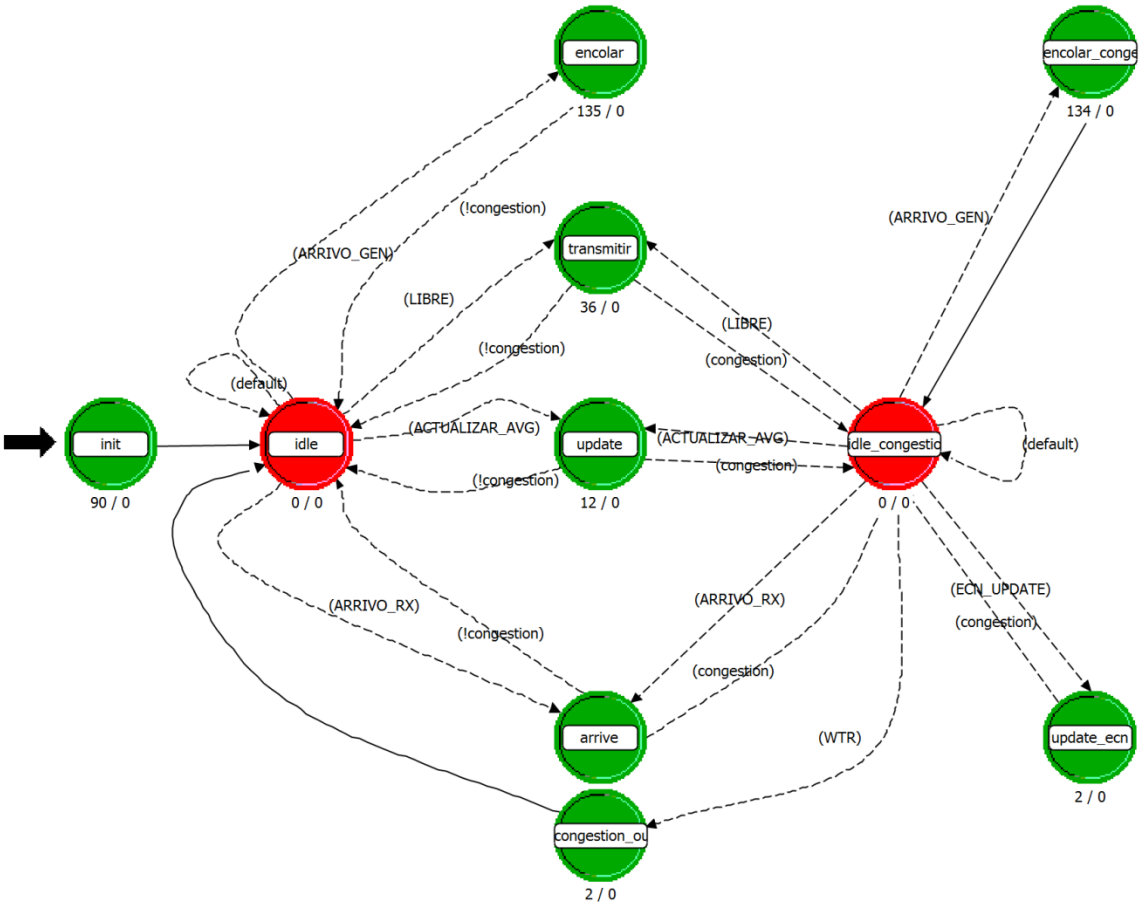


Figura 4-7 Maquina de estados final Agente Administrador de QoS

La equivalencia frente a la máquina de estados definida en la figura 4-7 se lista en la siguiente tabla:

Estados		Eventos	
Sistema de control de recursos	**Agente Administrador de QoS**	**Sistema de control de recursos**	**Agente Administrador de QoS**
idle	Idle	ARRIVE_FROM_USERS	ARRIVO_GEN
idle_busy	idle congestion	STATISTIC_EVENT	ACTUALIZAR_AVG
backbone_pk	Arrive	BUSY	congestion
pk_arrive	Encolar	ARRIVE_FROM_BACKBONE	ARRIVO_RX
statistics_busy	update_ecn	TIMEOUT	ECN_UPDATE
busy_arrive	encolar_congestion	CONGESTION_AVOIDED	WTR
statistics	Update		

Tabla 4-1 Equivalencias entre el sistema de control de recursos y el Agente administrador de QoS

El estado ***encolar*** implementa el algoritmo WRED para la gestión de colas y poder encolar según la probabilidad de descarte especificada por la decodificación AF.

El nuevo estado ***transmitir*** libera y transmite de las colas, dentro del estado esta implementado el algoritmo DRR con quantums variables para garantizar las tasas de transmisión a cada cola. El paso hacia este estado está dado por el el evento LIBRE, el cual se genera cada vez que el canal se encuentra disponible y la cola de transmisión no está vacía. Finalmente ***congestion_out*** únicamente deshabilita los temporizadores de congestión.

4.6 Modelos Complementarios.

La evaluación del agente administrador de QoS necesita de otros modelos que generen paquetes IPv6. Los modelos actuales de OPNET no utilizan una implementación de cabecera IPv6 formal, por lo que fue necesario implementar los modelos que permitieran la evaluación del agente, los cuales trabajan con codificación DSCP AF. Los nodos implementados se enuncian a continuación:

4.6.1 Generador de Tráfico IPv6.

Para generar los flujos de tráfico IPv6 se implementa un generador el cual puede producir *N* flujos de tráfico. El modelo de nodo se puede observar en la figura 4-8.

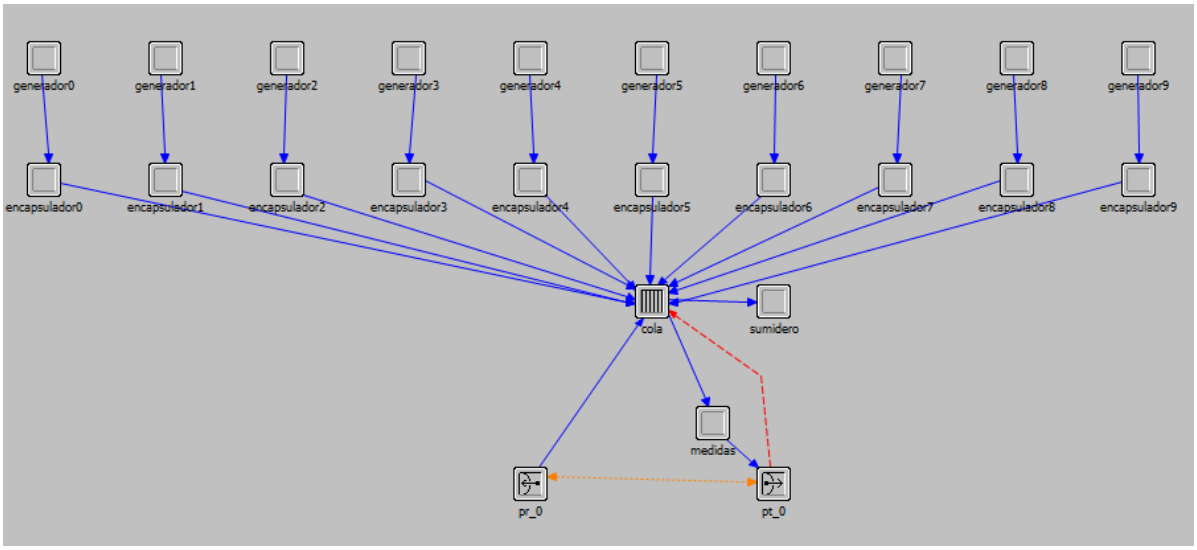


Figura 4-8 Modelo de Nodo del Generado de Tráfico.

Cada flujo es generado por el proceso *generador* al cual le puede ser definido el tiempo entre arribo y el tamaño entre paquetes de tipo constante o según una de las funciones de densidad de probabilidad contenidas en OPNET. Posteriormente el paquete generado es recibido por el proceso *encapsulador* el cual encapsula este paquete dentro de una cabecera IPv6. A este proceso le puede ser definido los campos, *Source Address*, *Destination Address*, *DS Code* y *Flow Label* de IPv6.

Todos los flujos creados convergen en el proceso *cola*, el cual implementa un esquema WRED y DRR para la transmisión hacia los módulos de transmisión y recepción pt_0 y pr_0 respectivamente, al proceso Cola le pueden ser definidos todos los parámetros de WRED y DRR. El proceso *medidas* es un módulo que obtiene estadísticas de la salida del generador antes de ser transmitidas, determina retardos, *throughput* y *jitter* a cada clase AF en la salida del módulo.

4.6.2 Modelo de Enrutador de Borde

Para la realización de la comparación frente a un enrutador que no utilice el modulo del agente administrador de QoS, se implementó el siguiente modelo de nodo (figura 4-9).

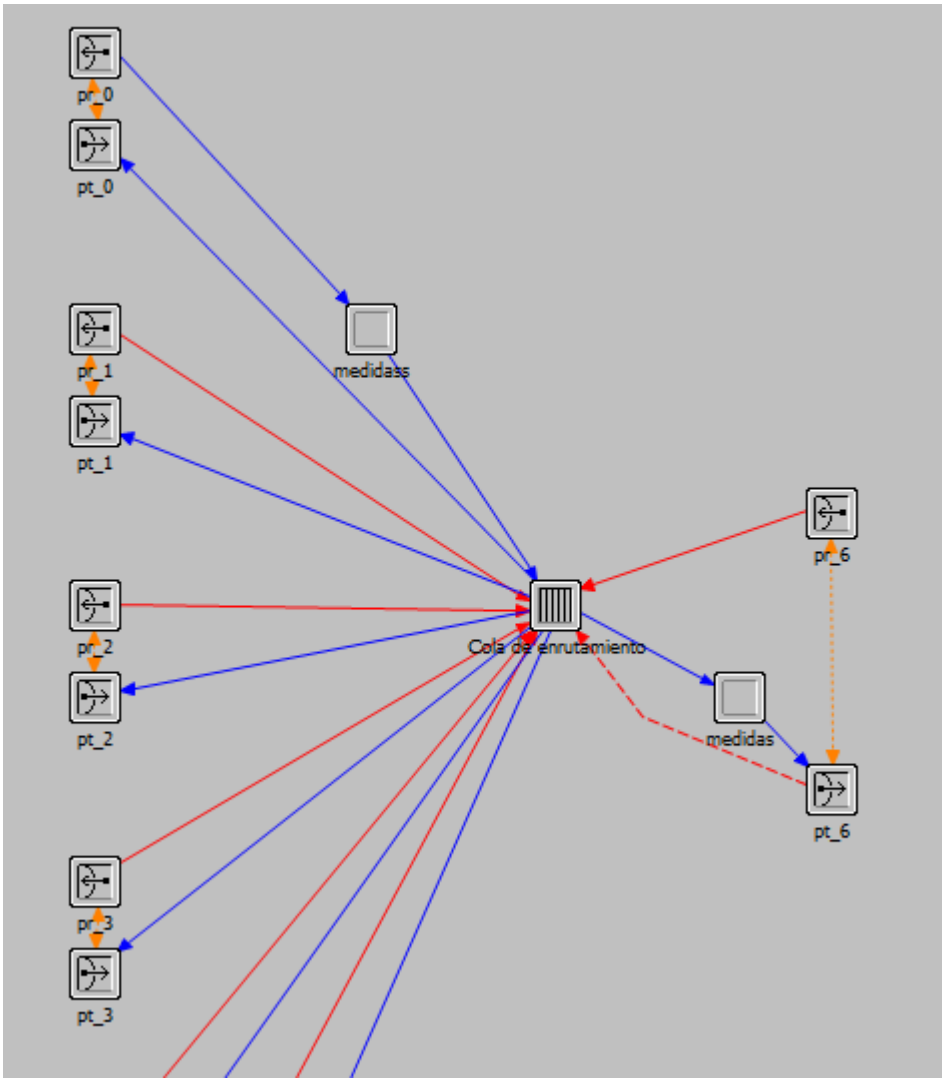


Figura 4-9 Modelo de nodo de enrutador de borde básico.

Este modelo recibe *N* entradas hacia una única salida a través del proceso *Cola de enrutamiento* el cual implementa los algoritmos WRED y DRR. Los módulos *medidas* tienen la función de obtener medidas antes y después del proceso de encolamiento.

4.6.3 Modelo de Enrutador de Borde con Agente Administrador de QoS

Este nodo tiene el mismo modelo de nodo de la figura 4-6 solo que el proceso que implemente es el definido en el numeral **4.4**.

4.6.4 Modelo de Enrutador de Backbone

El enrutador de backbone tiene la capacidad de N entradas e implementa un modelo de cola q_0 el cual también tiene implementado WRED y DRR con quantums variables. De aumentar el Q_{min} de las colas para las prioridades de clase 4, clase 3 y clase 2 este genera un paquete con el campo ECN en valor 0b11 indicando el inicio de congestión.

La evaluación del agente y su desempeño frente al backbone no tiene en cuenta el enrutamiento, por lo que este modelo utiliza enrutamiento estático. Cada nodo transmisor y recepción también poseen una cola de tipo FIFO (First In First Out). El modelo de nodo se ve a continuación en la figura 4-10.

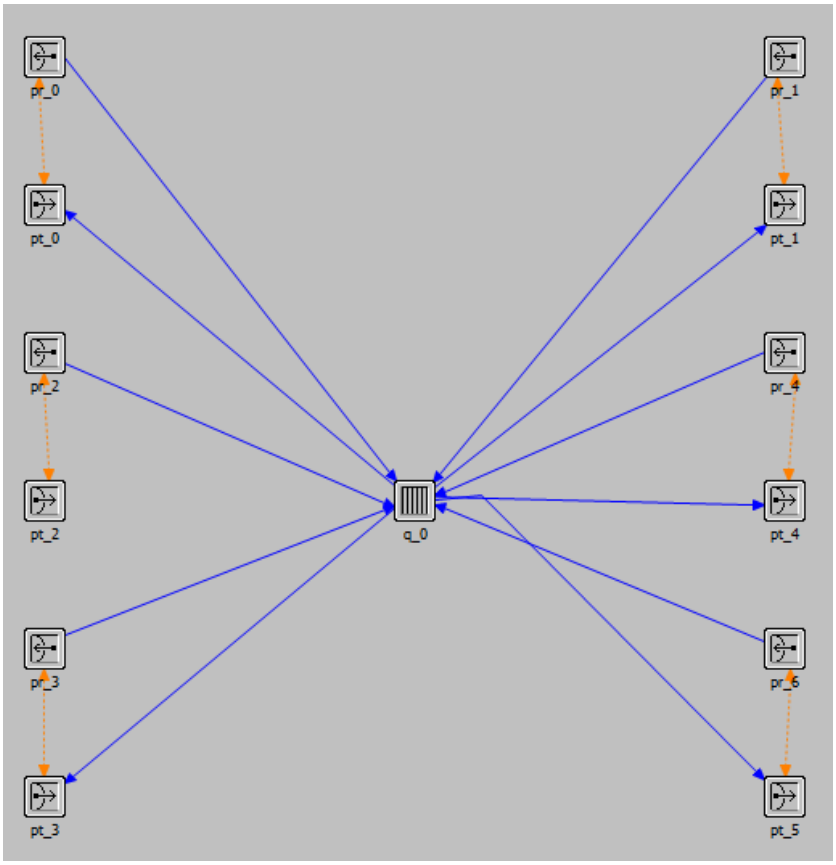


Figura 4-10 Modelo de nodo de enrutador de backbone

4.6.5 Modelo de Enrutador con MPLS

El modelo de nodo de este enrutador es equivalente al mostrado en la figura 4-6, la diferencia es que el proceso ejecutado en q_0 no utiliza el enrutamiento estático. Al estar simulando a nivel de capa 3, el enrutamiento de etiqueta que se realiza en MPLS no puede ser ejecutado. La solución es utilizar el *Interface Control Information* (ICI), este objeto generador por OPNET permite asociar información adicional asociada a un paquete sin que esta aumente el tamaño real del paquete.

Este modelo también es utilizado como enrutador de bordo, donde se le agrega a q_0 la funcionalidad de marcado MPLS de paquetes a través de ICI.

4.6.6 Modelo de Nodo de Resultados

Este último nodo implementado (figura 4-11), tiene como objetivo actuar como nodo de destino final, su principal función (proceso p_0) es obtener medidas de desempeño por cada una de las clases (delay, jitter y throughput) y para un solo flujo.

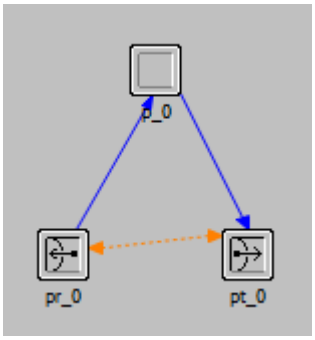


Figura 4-11 Modelo de Nodo de Resultados

5 ESCENARIOS DE PRUEBA.

Para la evaluación del agente y su desempeño frente a un esquema de enrutamiento de mejor esfuerzo y MPLS se establecen los escenarios de prueba que evalúan los nodos implementados.

Para todos los escenarios se asigna la siguiente distribución de clases y túneles. La evaluación se realizara siempre sobre la misma distribución para obtener una comparación de retardo sobre las mismas condiciones. Los algoritmos de WRED y DRR son indiferentes a los parámetros que les sean definidos, por lo que los valores especificados no afectan el comportamiento de estos y son tomados a manera de referencia.

Clase AF	Porcentaje de Ancho de Banda
Clase 4	30%
Clase 3	25%
Clase 2	25%
Clase 1	20%

Tabla 5-1 Distribución de Ancho de Banda en DRR

Los parámetros de WRED especificados para todos los escenarios son

Clase AF	Qmin	Qmax	Pmax
Low Drop	0.9	0.6	0.3
Medium Drop	0.8	0.5	0.6
High	0.7	0.4	0.9

Tabla 5-2 Parámetros de WRED

En todos los dispositivos se trabajará con tamaño nominal de buffer de 200 kbits (promedio)

5.1 Prueba de Desempeño Generador de Tráfico.

Para la prueba del generador de tráfico se establece un nodo generador y un nodo de resultados como se muestra en la figura 5-1. Para evaluar el máximo retardo generado por la cola se utiliza el método especificado en [19] generando la máxima tasa de transferencia para cada cola. El ancho de banda de salida del generador se especifica en 2 Mbits/s. Valor promedio de enlaces DSL. El escenario de prueba para el generador se observa en la figura 5-1

Figura 5-1 Escenario de prueba del generador de tráfico.

Sobre esta prueba es revisada la validez del modelo en la generación y salida de paquetes del mismo. El generador al tener diferentes flujos de entrada combina los flujos mediante un algoritmo WRED y algoritmo de atención DRR cumpliendo los mismos parámetros de las tablas 5-1 y 5-2.

5.2 Prueba de Desempeño Enrutador de Borde

El escenario de prueba utiliza el mismo método utilizado por el escenario anterior, tiene como fin evaluar el retardo introducido por el enrutador de borde a cada clase y a un flujo especifico, el escenario se observa en el la figura 5-2.

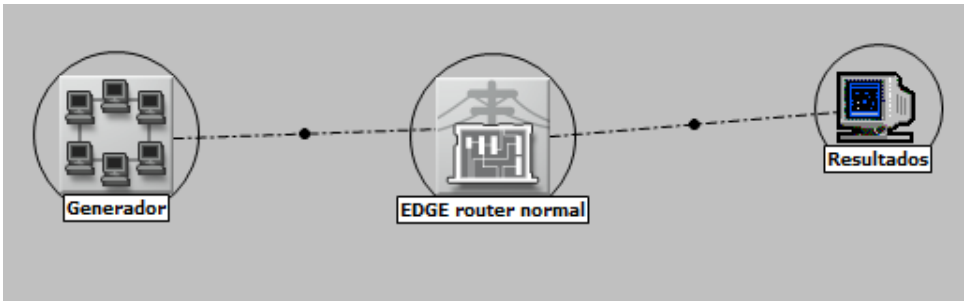


Figura 5-2 Escenario de prueba del desempeño del enrutador de borde

5.3 Prueba de Desempeño del Enrutador de Borde con Agente Administrador de QoS

El escenario es equivalente el escenario anterior, donde también se evalúan el retardo introducido por el enrutador de borde a cada clase y a un flujo, el escenario se observa en el la figura 5-3

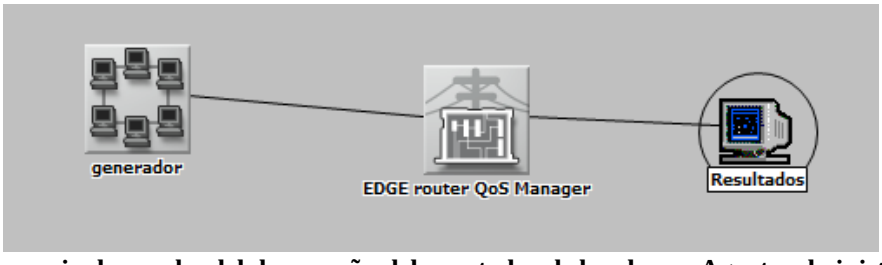


Figura 5-3 Escenario de prueba del desempeño del enrutador de borde con Agente administrador de QoS

5.4 Prueba de Desempeño con Reúso.

En este conjunto de escenarios se incluyen otros nodos generadores de tráfico para un ancho de banda de salida igual a la suma de todos los flujos de clase 4, para el caso propuesto de 4 Mbits/s. En este escenario se evalúan los retardos por clase, jitter y throughput, también se realizan las medidas para un flujo particular incluyendo porcentaje de paquetes perdidos.

En estos escenarios se compara la eficiencia del agente administrador de QoS frente a un enrutador normal. La topología del conjunto de escenarios se puede observar en la figura 5-4.

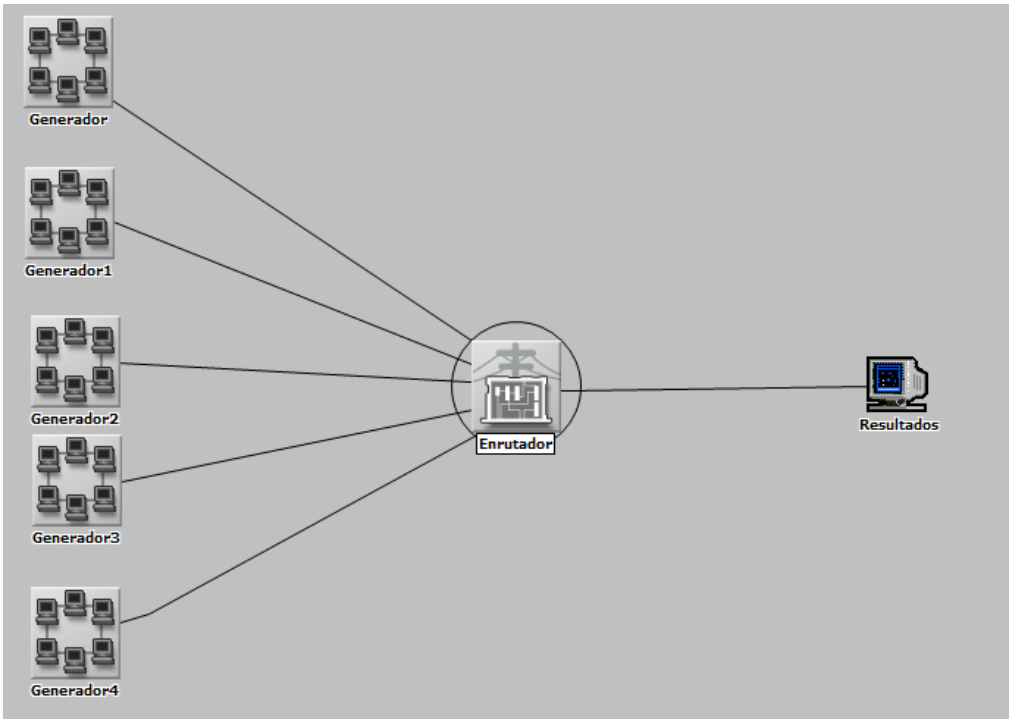


Figura 5-4 Escenario de prueba de desempeño con reúso.

5.5 Prueba Escenario MPLS

Esta prueba (figura 5-5) consiste en generar diferentes flujos de tráfico generando congestión en los enrutadores de backbone que funcionan con protocolo MPLS, se realizan mediciones de retardos por clase, jitter y throughput en congestión para cada clase AF y también se realizan las medidas anteriormente enunciadas para un flujo particular incluyendo porcentaje de paquetes perdidos. La capacidad de cada túnel MPLS corresponde a los mismos valores de la tabla 5-1

5.6 Prueba Escenario Agente Administrador de QoS.

En este último escenario (figura 5-6) bajo las mismas condiciones de tráfico del escenario anterior se evalúa el desempeño de la red operando con control de congestión ECN y el agente administrador de QoS. Cada enrutador del backbone genera un paquete notificador de congestión

si las colas superan el umbral mínimo Q_{min}, si el nodo de borde recibe el paquete con los bits de ECN en 0b11, limita nuevos flujos por parte de los usuarios hasta que la congestión termine.

Las medidas de este escenario son las mismas descritas en el escenario anterior.

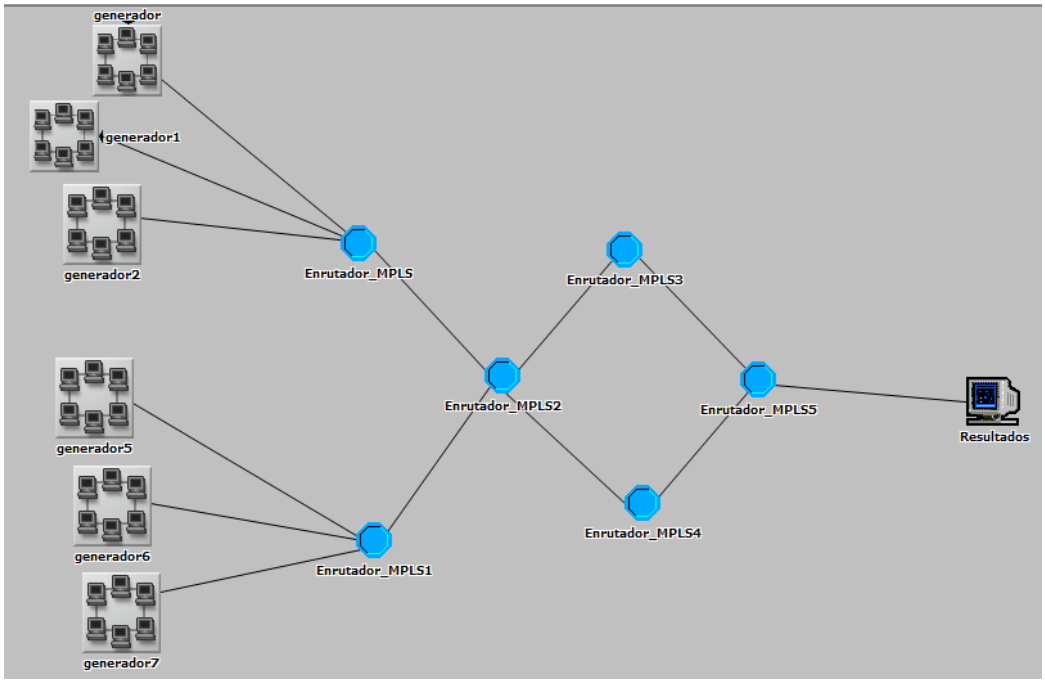


Figura 5-5 Escenario de prueba sobre MPLS

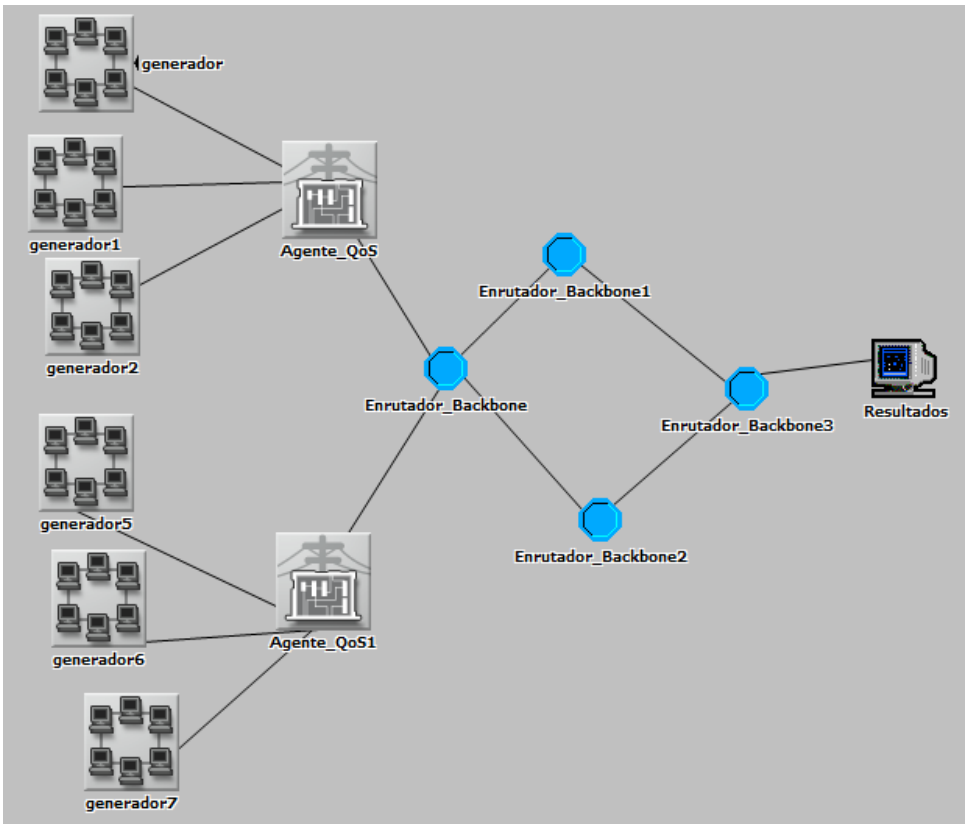


Figura 5-6 Escenario de prueba con Agente administrador de QoS

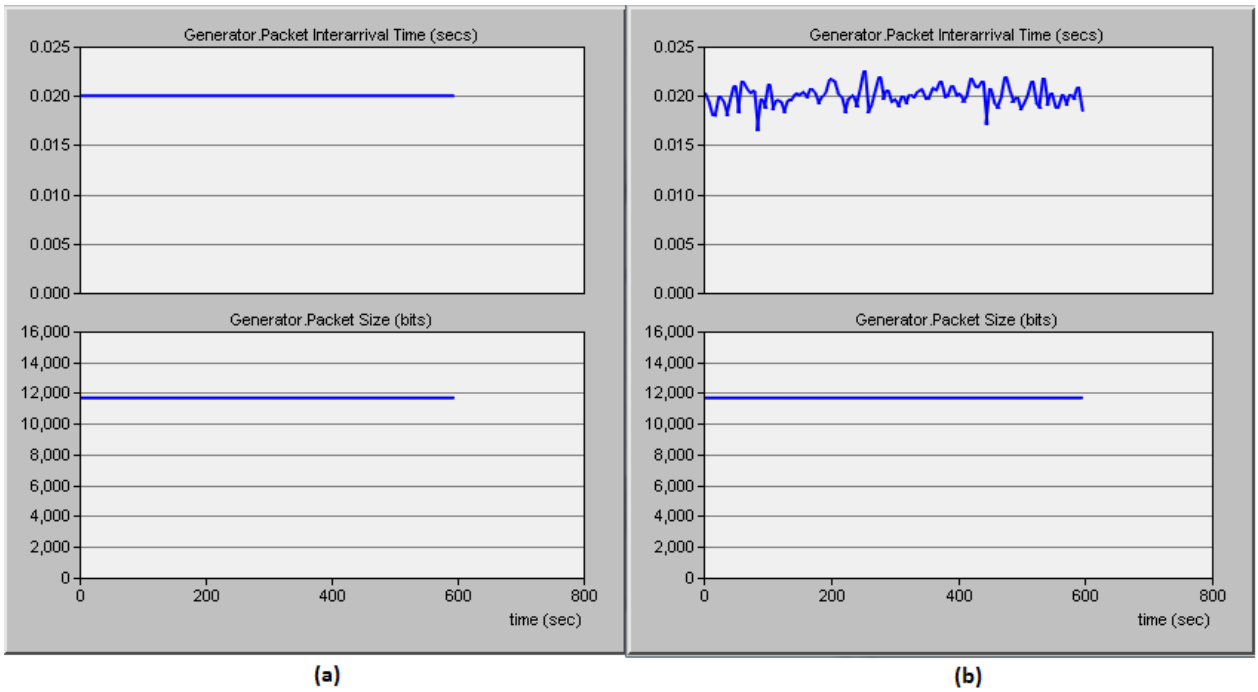
6 RESULTADOS Y ANALISIS

El análisis y evaluación en su primera fase se enfoca en el desempeño de los nodos de red, en cada escenario se realizan 40 simulaciones modificando la semilla de los generadores de números aleatorios. Los resultados expuestos en estas pruebas, hacen referencia a los valores promedios de cada una de las mediciones realizadas en cada simulación.

6.1 Desempeño Generador de Tráfico.

Antes de analizar los resultados se valida el modelo algorítmico del generador. En primera instancia los procesos generadores de paquetes son basados en los procesos de generación estándar ofrecidos por OPNET. El proceso de encapsulación de IPv6 recibe los paquetes y los encapsula en su respectiva cabecera IPv6, el proceso de generación y encapsulado establecen el tiempo de creación del paquete, con el cual se tomara de referencia para cálculos de retardo.

Los tiempos de interarribo para estos procesos tanto para funciones de densidad de probabilidad exponenciales y constantes se observan en la gráfica 6-1

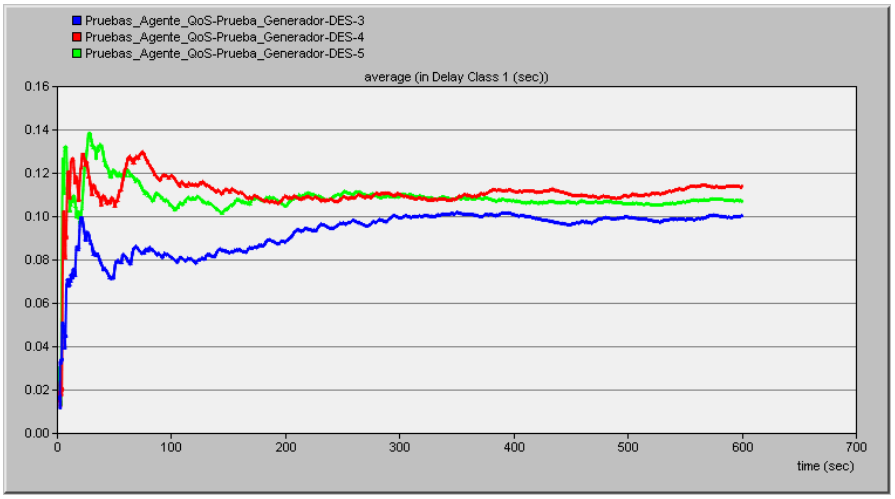


Gráfica 6-1 Tiempos interarribo y tamaño de paquete generado para tiempo constante (a) y tiempo interarribo exponencial (b).

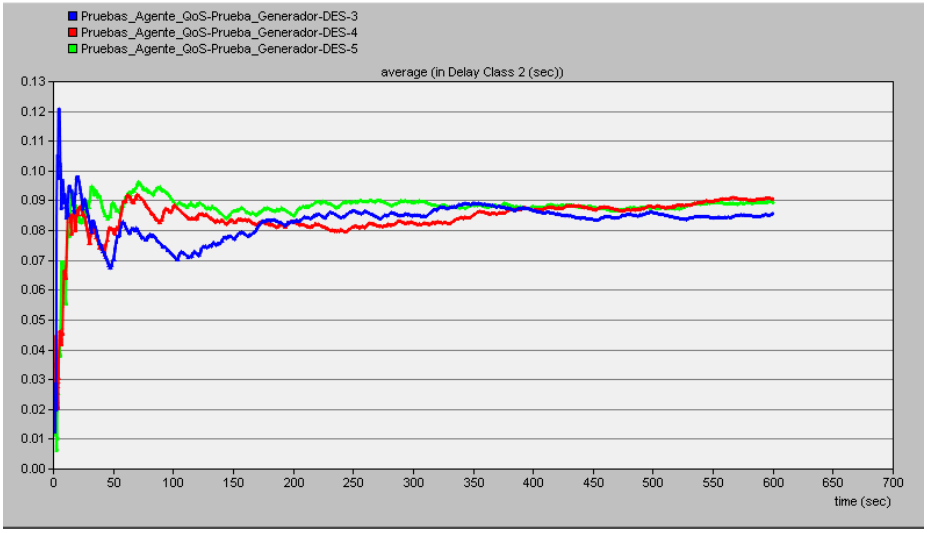
Al calcular la media del tiempo de interarribo en la gráfica 6-1 (b) se obtiene como media 0,02 segundos, de esta manera podemos concluir que el proceso de generación y encapsulamiento

en cabecera IPv6 cumple las características esperadas, al igual como los valores del tamaño de los paquetes que permanecen constantes

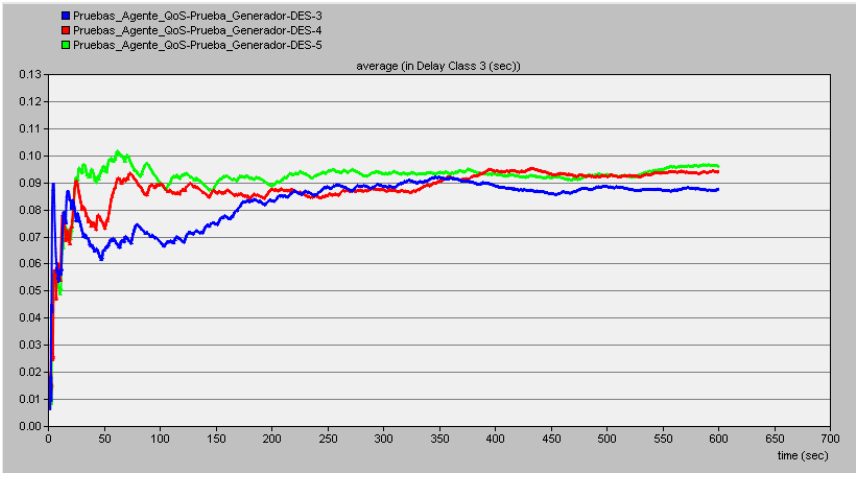
Los resultados de retardo para cada clase del generador fueron obtenidos colocando fuentes de tráfico por clase con tasa promedio de transmisión igual a la máxima capacidad por clase en la cola, las tasas promedio de arribo siguen una distribución de probabilidad exponencial con el fin de emular comportamiento aleatorio en las colas del generador. Todos los paquetes fueron generados con el tamaño máximo 1500 Bytes, equivalente al tamaño máximo transportable en una red Ethernet, los resultados obtenidos se observan en las gráficas 6-2, 6-3, 6-4 y 6-5.



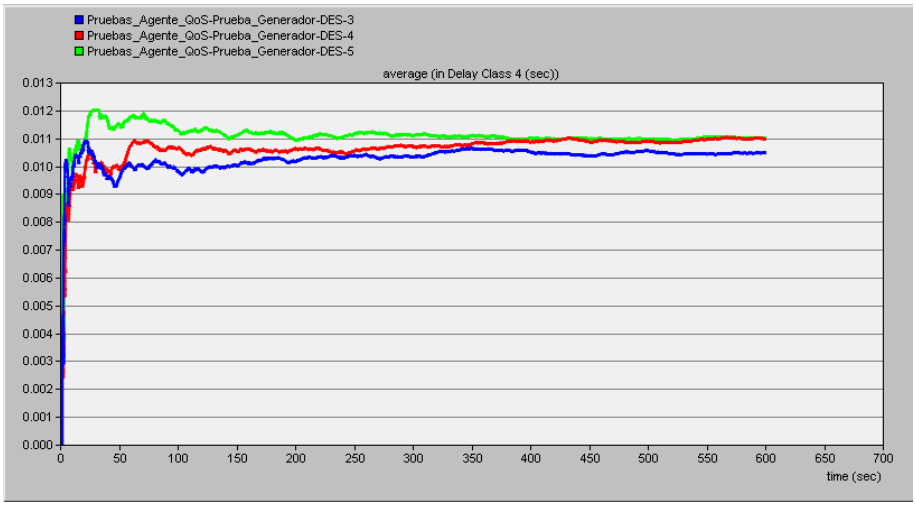
Gráfica 6-2Retardo Promedio Paquetes Clase 1 en Cola del Generador



Gráfica 6-3 Retardo Promedio Paquetes Clase 2 en Cola del Generador



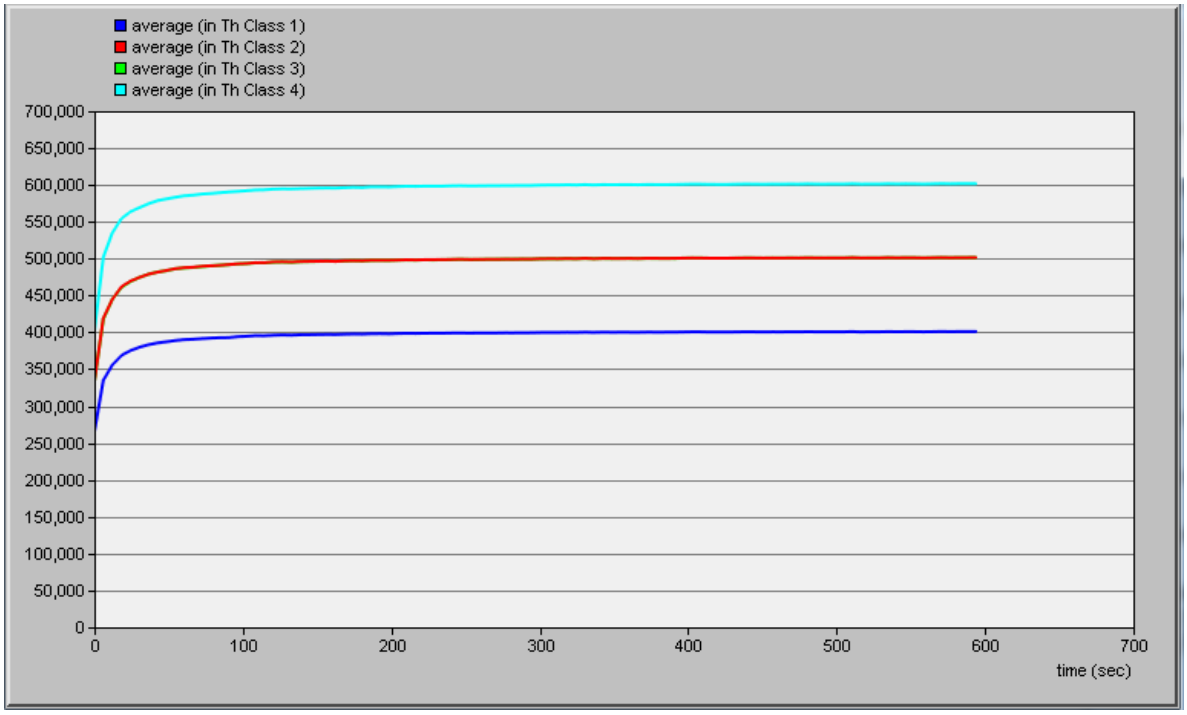
Gráfica 6-4 Retardo Promedio Paquetes Clase 3 en Cola del Generador



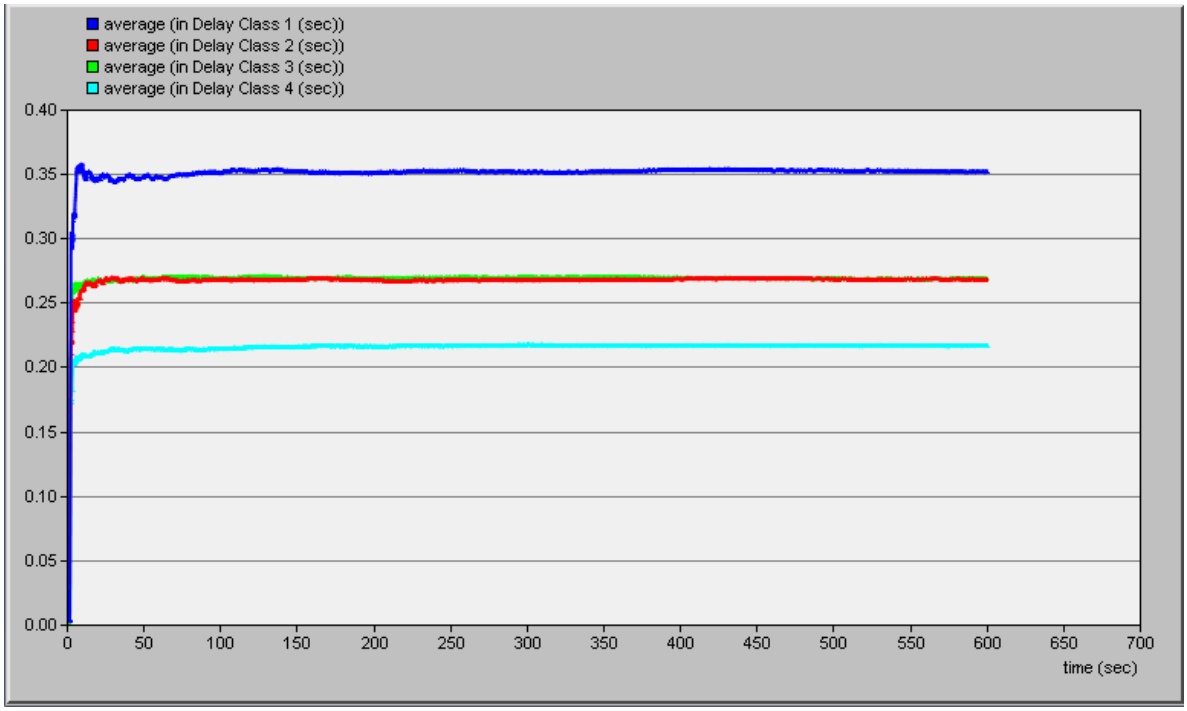
Gráfica 6-5 Retardo Promedio Paquetes Clase 4 en Cola del Generador

Para todas las colas el retardo se mantiene por debajo del límite especificado en la ecuación 2-2. Por ejemplo para el retardo promedio para tráfico con la más alta prioridad (Clase 4) por encolamiento en el generador es de 10.5 ms aproximadamente frente al límite de 3 segundos especificado por la ecuación 2-2.

Para la validación de la salida del generador de los algoritmos WRED y DRR se configuran los parámetros de la generación de tráfico de tal forma que las colas entren en congestión, adicionalmente, son configurados el tamaño de los paquetes de tal forma que se pueda evaluar la independencia al throughput en la salida del generador obteniendo:



Gráfica 6-6 Throughput de salida del generador con algoritmo DRR.

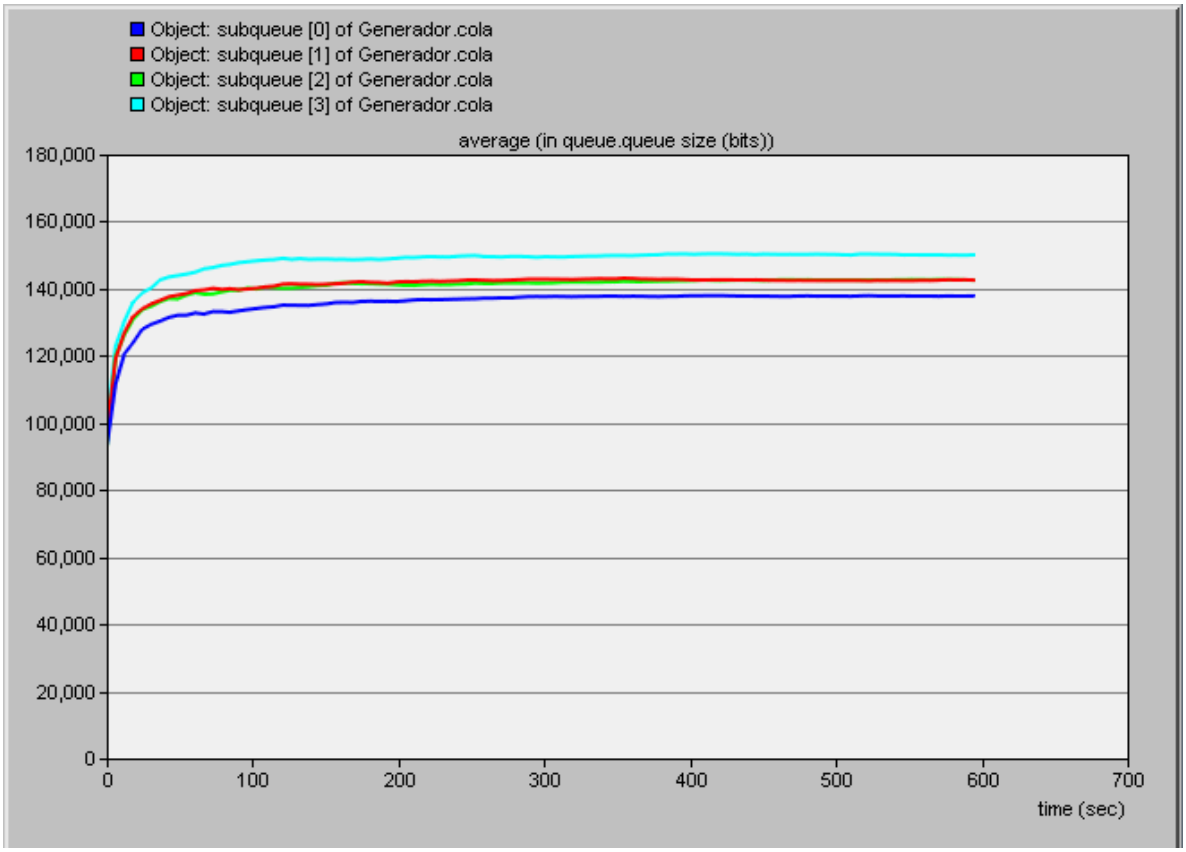


Gráfica 6-7 Retardo promedio de encolamiento en el generador, algoritmo WRED.

Como se observa en las gráfica 6-6 el valor de throughput para cada cola es exactamente el esperado 400kpbs para la clase 1, 500 kbps para la clase 2, 500 kbps para la clase 3 y 600kbps para la clase 4. Frente a la gráfica 6-7 se puede concluir que el retardo se mantiene por debajo de

los límites de otros algoritmos como WFQ (ecuación 2-2) 3 segundos aproximadamente pero con costo computacional menor. Estos resultados validan el comportamiento del algoritmo DRR utilizado por los nodos en los siguientes escenarios.

En la figura 6-8 se exponen los valores de tamaño de cola promedio que durante la simulación mantienen su valor promedio constante, deseable para el algoritmo WRED.



Gráfica 6-8 Tamaño de cola promedio en el generador.

De los resultados de WRED y DRR se puede concluir la validez de las implementaciones, los modelos descritos en otros escenarios contienen la misma implementación por lo que la validación anteriormente expuesta se aplica a los otros modelos.

6.2 Desempeño Enrutador de Borde.

Utilizando el mismo método de evaluación del escenario 6.1 los retardos obtenidos se listan en la tabla 6.1

	Retardo (promedio)	Jitter (promedio)	Throughput (promedio)	Throughput (esperado)
Clase 1	12 ms	24.2 ms	393 kbps	400 kbps
Clase 2	12 ms	18.4 ms	499 kbps	500 kbps
Clase 3	12 ms	18.3 ms	500 kpps	500 kbps
Clase 4	12 ms	16.4 ms	599 kbps	600 kbps

Tabla 6-1 Resultados de desempeño de enrutador de borde.

Los valores de jitter obtenidos se deben al generador de tráfico pues está generando mayor tráfico que el throughput de salida. Realizando mediciones de jitter antes y después del enrutador de borde los valores permanecen constantes, por lo tanto el enrutador de borde frente a sus valores máximos de throughput no afecta el jitter. Al mantener los valores de throughout máximos en la entrada del enrutador de borde el throughput de salida se mantiene constante. El algoritmo DRR mantiene uso justo en la atención de colas y aunque los valores de throughput máximo para cada clase es de 500kbps (configuración DRR en el nodo), la clase 1 al solo tener un máximo de 400kbps el ancho de banda restante es asignado a la clase 4 que posee un throughput de 600 kbps. En cuanto al retardo, el encolamiento y retardo generado por los procesos de encolamiento y procesamiento de paquete es constante para cada clase. El valor de retardo incluye los retardos de transmisión del enlace que es aproximadamente de 6 ms para un paquete de 12000 bits, por lo que los retardos obtenidos equivalen a retardos debidos a la transmisión. De estas medidas podemos validar el comportamiento del enrutador de borde.

6.3 Desempeño del Enrutador de Borde con Agente Administrador de QoS

Utilizando el mismo método de evaluación del numeral 6.1 los retardos obtenidos se listan en la tabla 6.2 donde se obtienen los valores de retardo, jitter y throughput obtenido frente al throughput esperado teóricamente.

	Retardo (promedio)	Jitter (max)	Throughput (promedio)	Throughput (esperado)
Clase 1	12 ms	23.1 ms	398 kbps	400 kbps
Clase 2	12 ms	18.8 ms	498 kbps	500 kbps
Clase 3	12 ms	18.8 ms	498 kpps	500 kbps
Clase 4	12 ms	16.6 ms	598 kbps	600 kbps

Tabla 6-2 Resultados de desempeño de enrutador de borde con agente administrador de QoS.

El agente administrador de QoS mantiene constante los valores de retardo y el throughput, frente a los valores de jitter la diferencia en las clases de mayor prioridad tiene diferencias en milisegundos respecto al modelo de enrutador de borde planteado.

Los resultados obtenidos en los escenarios anteriores validan los modelos planteados y desarrollados para este proyecto, a continuación se realiza la validación de los modelos frente a condiciones de reúso sometiendo a los modelos a manejar throughput mayores al valor nominal.

6.4 Desempeño con Reúso.

6.4.1 Enrutador Normal.

Durante esta prueba se transmiten 2 flujos de alta prioridad (clase 4) y 2 flujos con tasa de arribo con función de densidad de probabilidad exponencial en las demás clases y un flujo de prueba con tasa de transferencia constante. El enrutador clasifica solo por diferenciación de servicios con un ancho de banda constante para cada clase. Se evaluó el comportamiento por clases y el desempeño con cada uno de los flujos utilizando una tasa de utilización de 105% de la capacidad del enlace. Los resultados se enuncian en la tabla 6-3.

	Retardo (promedio)	Jitter (max)	Throughput (esperado)	Throughput (promedio)
Clase 1	92.6 ms	11.2 ms	750 kbps	747.46 kbps
Clase 2	96 ms	10.9 ms	750 kbps	747.72 kbps
Clase 3	90.4 ms	10.9ms	750 kpps	748.56 kbps
Clase 4	144.2 ms	18.1 ms	750 kbps	750.23 kbps

Tabla 6-3 Resultados de desempeño de enrutador de borde con reúso.

Adicionalmente se contabilizo el número de paquetes perdidos por cada flujo. Los cuales se enuncian en la tabla 6-4.

	Flow Label	Tasa de arrivo (promedio)	Tiempos de Inicio	% Paquetes Recibidos.
Clase 1	1111	363.94 kbps	3 seg	87.8 %
	201	386.7 kbps	8 seg	96.13 %
Clase 2	396	371.9 kbps	3 seg	88.9 %
	281	386.9 kbps	8 seg	95.36 %
Clase 3	285	363.3 kbps	3 seg	90.31 %
	262	386.2 kbps	8 seg	96.16 %
Clase 4	143	375 kbps	2 seg	79.21 %
	343	375 kbps	5 seg	99.97 %
Flujo de Prueba	434	375 kbps	10 seg	84.75 %

Tabla 6-4 Resultados por flujos en enrutador de borde con reúso.

Los resultados obtenidos demuestran el efecto de congestión sobre los flujos y la ejecución de un algoritmo de descarte como RED, para este sistema, la congestión y el descarte de paquetes se presenta por igual a cada una de las clases. Los valores de retardo y jitter se mantienen

similares al no existir ninguna diferenciación del tráfico. En la siguiente prueba se aumenta el tráfico a 125% de la capacidad del enlace obteniendo como resultado:

	Retardo (promedio)	Jitter (max)	Throughput (esperado)	Throughput (promedio)
Clase 1	146.1 ms	12.9 ms	750 kbps	747.36 kbps
Clase 2	146.7 ms	12 ms	750 kbps	748.82 kbps
Clase 3	146.2 ms	12 ms	750 kpps	748.36 kbps
Clase 4	162.9 ms	16.3 ms	750 kbps	750.43 kbps

Tabla 6-5 Resultados de desempeño de enrutador de borde con reúso.

	Flow Label	Tasa de arribo (promedio)	Tiempos de Inicio	% Paquetes Recibidos.
Clase 1	1111	408 kbps	3 seg	73.53 %
	201	464.32 kbps	8 seg	83.21 %
Clase 2	396	420.8 kbps	3 seg	74.1 %
	281	465.5 kbps	8 seg	82.07 %
Clase 3	285	426.82 kbps	3 seg	74.84 %
	262	462.62 kbps	8 seg	81.42 %
Clase 4	143	375 kbps	2 seg	22.37 %
	343	375 kbps	5 seg	81.70 %
Flujode Prueba	434	375 kbps	10 seg	97.58 %

Tabla 6-6 Resultados por flujos en enrutador de borde con reúso.

El aumento de tráfico acentúa el nivel de pérdida de paquetes en cada clase incluso en los flujos de prioridad alta quienes pueden llegar a tener una alta pérdida de paquetes. Sin embargo en un enrutamiento con solo diferenciación de servicios, los valores de retardo y de jitter son constantes para cada una de las clases.

La ausencia de un control de admisión congestiona el tráfico por clases aumentando la perdida de paquete en cada cola de prioridad, el control de admisión implimentado mejora las prestaciones como se describe en la siguiente prueba.

6.4.2 Enrutador con agente administrador de QoS.

Realizando las mismas pruebas ahora con un enrutador con administrador de QoS, 105% de la capacidad del enlace y throughput por clase descritos en la tabla 6-7 se obtienen los siguientes resultados:

	Retardo (promedio)	Jitter (max)	Throughput (promedio)	Throughput (esperado)
Clase 1	266 ms	14.1ms	599.2 kbps	600 kbps
Clase 2	146.4 ms	11.1 ms	748.6 kbps	750 kbps
Clase 3	151.7 ms	11.2 ms	749.8 kpps	750 kbps
Clase 4	31.1 ms	9 ms	897.3 kbps	900 kbps

Tabla 6-7 Resultados de desempeño de enrutador de borde con agente administrador de QoS con reúso.

	Flow Label	Tasa de arribo (promedio)	Tiempos de Inicio	% Paquetes Recibidos.
Clase 1	1111	353.06 kbps	3 seg	75.45 %
	201	381.76 kbps	8 seg	81.23 %
Clase 2	396	358.4 kbps	3 seg	90.07 %
	281	387.9 kbps	8 seg	96.51 %
Clase 3	285	366.2 kbps	3 seg	89.47 %
	262	390.76 kbps	8 seg	95.84 %
Clase 4	143	300 kbps	2 seg	100 %
	343	300 kbps	5 seg	100 %
Flujo de Prueba	434	300 kbps	10 seg	100 %

Tabla 6-8 Resultados por flujos en enrutador de borde con reúso.

En esta prueba cada flujo es monitoreado, y es aplicado un control de admisión en la entrada. La aplicación de throughput diferentes para cada clase, genera una desigualdad en los niveles de retardo, de manera más acentuada en las de prioridades más bajas. Este comportamiento es esperado ya que se considera que los flujos de clases 1 o 2 sean de naturaleza elástica, es decir no susceptibles a retardos y puedan tener ancho de banda variable, como lo son las transferencias de archivos o algunos protocolos basados en WEB, adicionalmente, este tipo de tráfico debe estar soportado en protocolos de entrega confiable como TCP para soportar la perdida de paquetes.

En cuanto a la clase de prioridad más alta el agente administrador de QoS mantiene los niveles de retardo y jitter bajos; perdidas de paquetes por desborde de buffer iguales a cero. Este resultado valida la operación del agente de calidad de servicio frente a los criterios de diseño establecidos. Este comportamiento es el deseado para tráfico sensible al tiempo.

Para analizar el comportamiento frente a variaciones se introduce un flujo de prueba, el flujo de prueba al ser generado con código DS en las clases 1 y 3 incrementa un poco la perdida de paquetes de dichos flujos, si el flujo de prueba es generado como flujo de clase 4 a este se le garantiza el mismo comportamiento que los demás flujos de capa 4, retardo bajo, jitter bajo y sin pérdida de paquetes por desbordamiento de buffer.

Haciendo un incremento del tráfico del 20% a 125% se obtuvieron los siguientes resultados:

	Retardo (promedio)	Jitter (max)	Throughput (promedio)	Throughput (esperado)
Clase 1	264.3 ms	14.1ms	599.1 kbps	600 kbps
Clase 2	181.5 ms	11.6 ms	748.6 kbps	750 kbps
Clase 3	179.1 ms	11.4 ms	748.6 kpps	750 kbps
Clase 4	18.2 ms	5.8 ms	898.6 kbps	900 kbps

Tabla 6-9 Resultados de desempeño de enrutador de borde con agente administrador de QoS con reúso.

	Flow Label	Tasa de arrivo (promedio)	Tiempos de Inicio	% Paquetes Recibidos.
Clase 1	1111	414.96 kbps	3 seg	63.3 %
	201	462.02 kbps	8 seg	70.87 %
Clase 2	396	415.86 kbps	3 seg	80.44 %
	281	465.1 kbps	8 seg	88.54 %
Clase 3	285	420.7 kbps	3 seg	80.35 %
	262	460.8 kbps	8 seg	88.56 %
Clase 4	143	375 kbps	2 seg	100 %
	343	375 kbps	5 seg	100 %
Flujo Prueba	434	375 kbps	10 seg	0 %

Tabla 6-10 Resultados por flujos en enrutador de borde con reúso.

Para el agente administrador de QoS, los resultados evidencian mejoras en tiempos de retardo y jitter en la clase 4. Los flujos al tener un mayor tráfico alcanzan rápidamente los límites de usuario designados en el agente administrador de QoS por lo que el uso de ancho de banda es priorizado para el tráfico de alta prioridad. La priorización de dicho tráfico también genera aumento en los retardos de las demás clases lo cual puede llegar a ser contraproducente para aplicaciones con estos niveles de prioridad. En un escenario de congestión el agente rechaza tráfico que le genere congestión como se detallara en los resultados de la siguiente prueba.

En caso de que un flujo supere los niveles permitidos por usuario el agente manejará el tráfico hacia una clase inferior para no degradar el retardo, jitter y pérdida de paquetes de la clase a la que pertenece. Dicho comportamiento se describe en la gráfica 6-9, en la cual se generan los mismos flujos de clase 4 y 3. En la clase 2 se atribuyen todos los flujos a una misma IP origen con el fin de superar el límite de usuario. Los flujos de clase 1 son detenidos y el flujo de prueba es adicionado a la misma IP utilizada para los flujos de clase 3. Al superarse el límite de usuario el flujo de prueba es reclasificado hacia la clase 1.

Gráfica 6-9 Reclasificación de flujo por el Agente administrador de QoS

En conclusión el agente presenta mejoras en la atención a flujos prioritarios, aumenta las tasas de atención y causa descartes masivos en colas de menor prioridad. Sin embargo, este escenario carece o del control de congestión que utiliza el agente de QoS para optimizar sus resultados.

6.5 Desempeño Escenario MPLS.

En este escenario se evaluó el comportamiento de la topología con nodos MPLS. Para esta prueba en el primer nodo de borde (Enrutador_MPLS), se generó el tráfico descrito por la tabla 6-11.

Ref. IP Origen	Flujos Clase 4		Flujos Clase 3		Flujos Clase 2		Flujo Clase 1	
	Throughput	Tiempo de Inicio	Throughput	Tiempo de Inicio	Throughput	Tiempo de Inicio	Throughput	Tiempo de Inicio
431	600 Kbps	1 s	300 Kbps	50 s	300 Kbps	25 s	200 Kbps	125 s
	500 Kbps	100 s	-	-	100 Kbps	75 s	-	-
400	400 Kbps	325 s	300 Kbps	350 s	300 Kbps	375 s	200 Kbps	400 s
435	-	-	-	-	-	-	200 Kbps	400 s

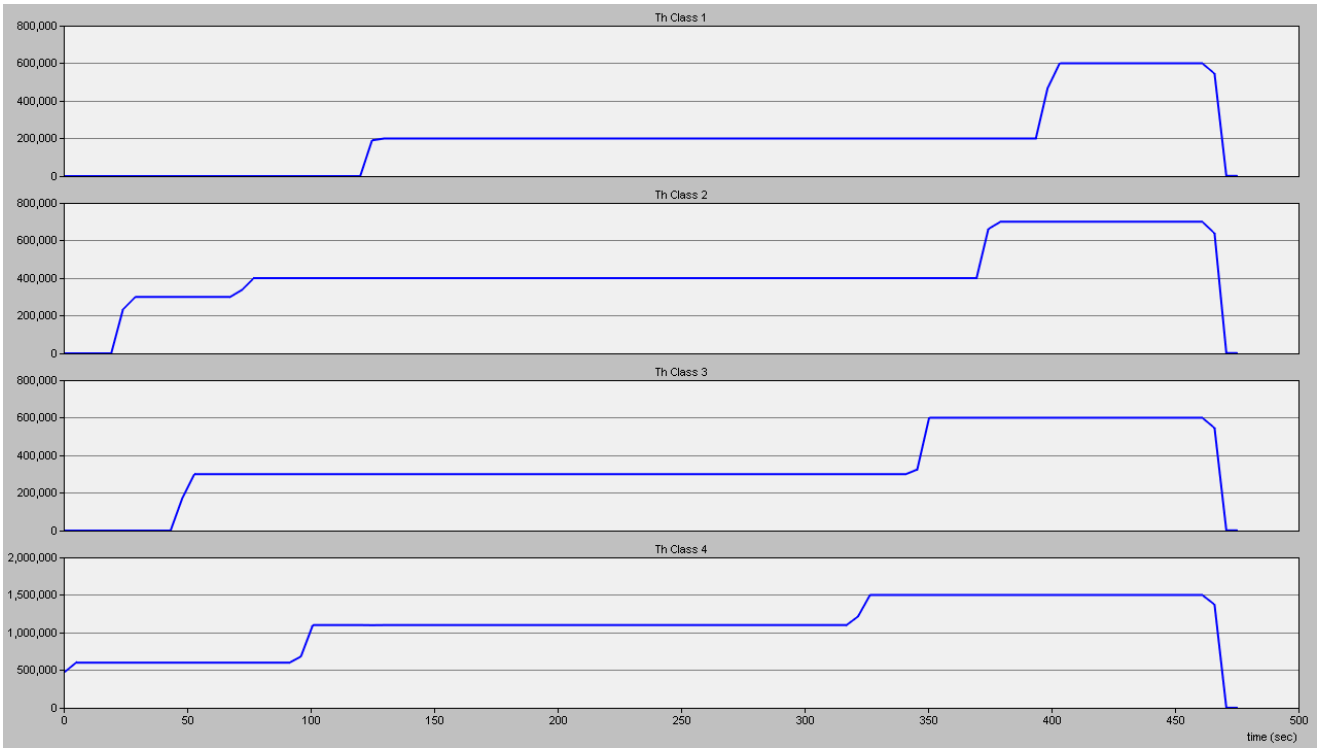
Tabla 6-11 Configuración de tráfico para los generadores en el primer nodo de borde.

En la tabla 6-12 se describe el tráfico para el segundo nodo de borde.

IP Origen	Flujos Clase 4		Flujos Clase 3		Flujos Clase 2		Flujo Clase 1	
	Throughput	Tiempo de Inicio	Throughput	Tiempo de Inicio	Throughput	Tiempo de Inicio	Throughput	Tiempo de Inicio
531	600 Kbps	150 s	300 Kbps	200 s	300 Kbps	175 s	200 Kbps	250 s
	-	-	-	-	100 Kbps	225 s	-	-

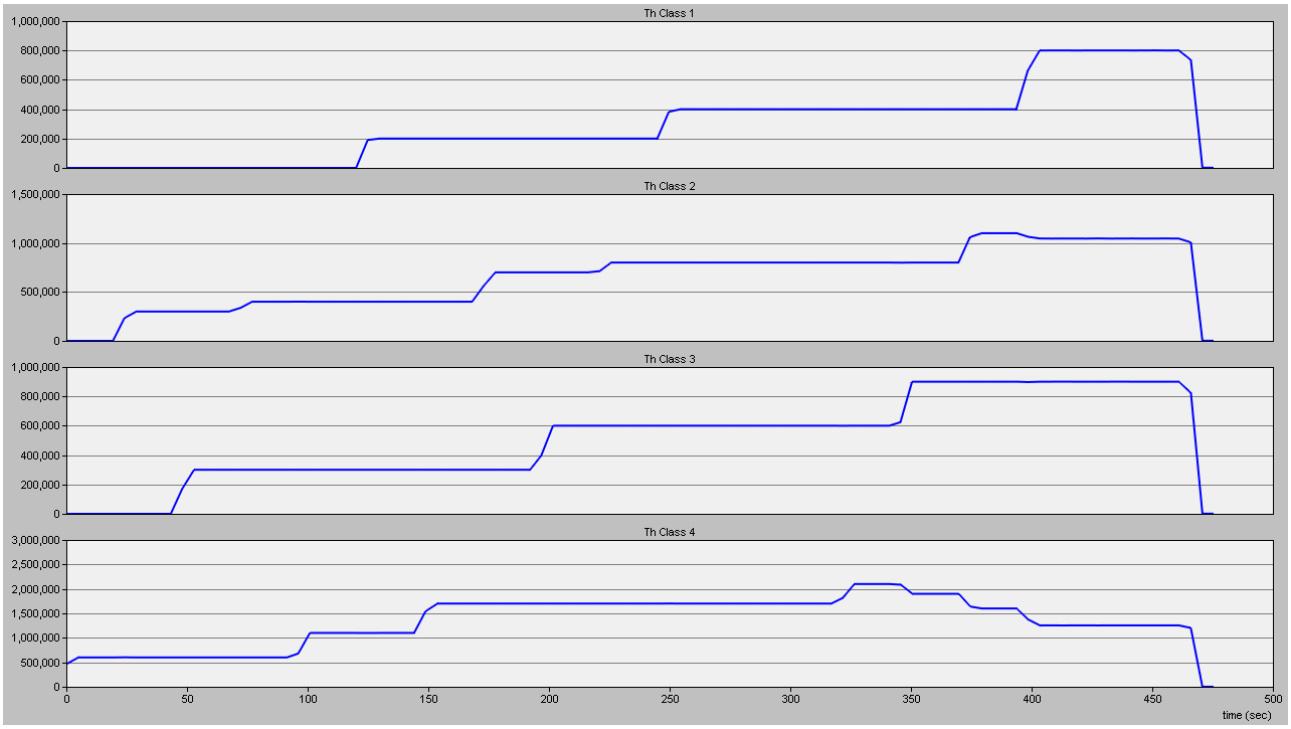
Tabla 6-12 Configuración de tráfico para los generadores en el segundo nodo de borde.

Bajo estas condiciones los valores de *throughput* en el enrutador de borde que recibe el tráfico descritos en la tabla 6-11 (Enrutador_MPLS) se observan en la gráfica 6-10



Grafica 6-10 *Throughput* en el enrutador de borde (Enrutador_MPLS)

Los resultados demuestran que los flujos en este nodo no presentan ninguna modificación durante toda la simulación incluso superándose los limites correspondientes a cada capacidad del túnel. Por ejemplo el túnel de clase 4 tiene capacidad de 1200 Kbps y durante esta prueba se llega a 1500 Kbps. Debido a este efecto, los flujos pueden acumularse en las categorías de mayor prioridad y generar perdida de paquetes como se puede evidenciar en la gráfica 6-11. En dicha gráfica se puede ver que a partir del segundo 350 comienza el efecto de congestión en el enrutador que concentra los flujos (Enrutador_MPLS2), el efecto inmediato es la reducción del throughput en las clases que están por encima del límite definido.



Grafica 6-11 *Throughput* en enrutador de *backbone* (Enrutador_MPLS2)

Ahora los resultados de las mediciones de *throughput, jitter* y *delay* para esta prueba se muestran a continuación en la tabla 6-13. El valor de throughput descrito en este escenario equivale a cuando todos los flujos se encuentran activos, (después del segundo 400 de simulación).

	Retardo (promedio)	Jitter (max)	Throughput (recibido)	Throughput (generado)
Clase 1	54.6 ms	20 ms	800 kbps	800 kbps
Clase 2	66 ms	17.8 ms	1050 kbps	1100 kbps
Clase 3	49.2 ms	15.6 ms	900 kpps	900 kbps
Clase 4	60.3 ms	17 ms	1255 kbps	2100 kbps

Tabla 6-13 Configuración de generadores de tráfico.

Los flujos pertenecientes a las clase 3 y clase 1 conservan sus paquetes, sin embargo se ve un gran decremento en el *throughput* de clase 4, para un mayor detalle la tabla 6-14 describe las características por flujo.

	Flow Label	Tasa de arrivo (promedio)	Tiempos de Inicio	% Paquetes Recibidos.	Retardo	Jitter (Max)
Clase 1	102	200 kbps	125 seg	100 %	56.6 ms	20 ms
	1111	200 kbps	400 seg	100 %	74 ms	18.9 ms
	301	200 kbps	400 seg	100 %	44.3 ms	19.2 ms
	112	200 kpbs	250 seg	100 %	50.8 ms	18.5 ms
Clase 2	163	250 kbps	50 seg	85.31 %	97.4 ms	17.1 ms
	101	100 kbps	75seg	95.94 %	98.8 ms	17 ms
	396	300 kbps	375 seg	97,68 %	217.3 ms	17.8 ms
	173	300 kbps	175 seg	99.97 %	101.34 ms	17.62 ms
	111	100 kbps	225 seg	99.95 %	110.36 ms	17,43 ms
Clase 3	181	300 kbps	25 seg	100 %	52.47 ms	14.94 ms
	285	300 kbps	350 seg	100 %	43.11 ms	15.56 ms
	191	300 kbps	200 seg	100%	42.71 ms	15.18 ms
Clase 4	143	230kbps	1 seg	68.15 %	93 ms	16.08 ms
	162	300 kbps	100 seg	79.84 %	100.26 ms	16.75 ms
	174	320 kbps	325 seg	80.4 %	160.78 ms	16.97 ms
	135	400 kbps	150 seg	80.23%	98.87 ms	16.47 ms

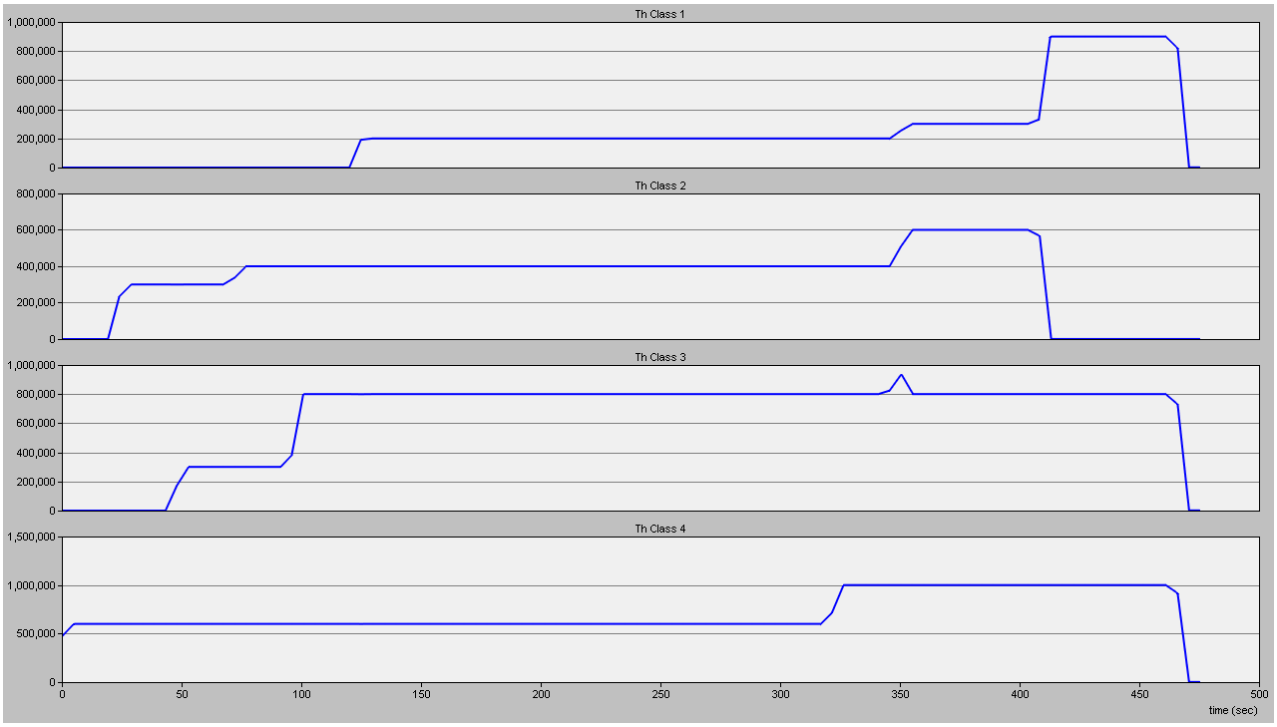
Tabla 6-14 Resultados por flujo en el nodo de resultados.

De lo anterior se concluye que la ausencia de control de flujo en cada una de las clases genera congestión en los enrutadores del *backbone* que tiene como resultado latencias más altas y perdidas de paquetes.

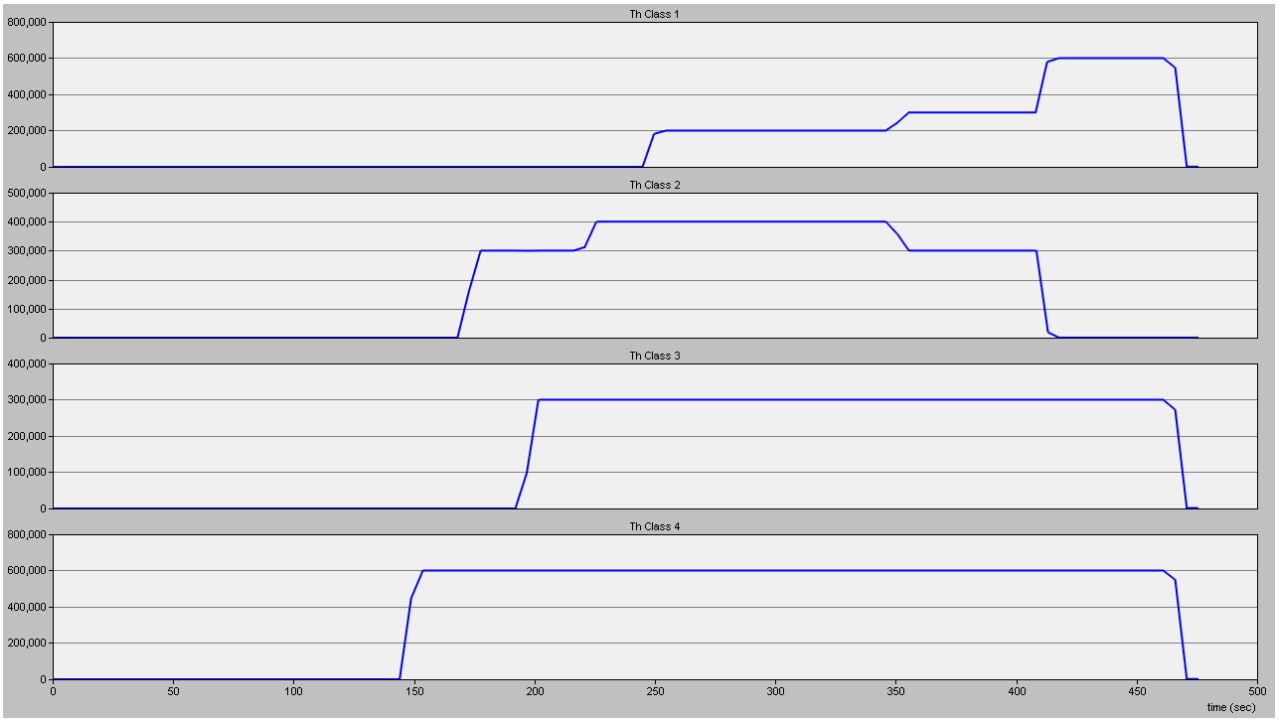
6.6 Desempeño Escenario Agente Administrador de QoS.

El agente administrador de QoS para el mismo escenario de simulación utiliza las mismas características de tráfico descritas en la tabla 6-11 y 6-12. El tráfico generado por el primer generador con IP origen 431 supera los valores máximos para la clase 4, el agente administrador de QoS organiza los flujos de tal manera que el flujo que excede la capacidad es redirigida a la clase 3 como se ve en la gráfica 6-12.

A partir del segundo 325 inicia la transmisión del segundo nodo, antes de ese tiempo el otro nodo de borde (Agente_QoS1) inicio su transmisión (gráfica 6-13). Por lo que el enrutador de *backbone* notifica la congestión. En la gráfica 6-12 se puede observar un pico en la clase 3, en el segundo 350, generado por la entrada a congestión del *backbone* y corresponde al transitorio de reorganización de la base de datos por el agente administrador de Qos. De igual manera en la gráfica 6-13 se puede observar como el *throughput* perteneciente a la clase 2 se traslada al de clase 1 (de menor prioridad).

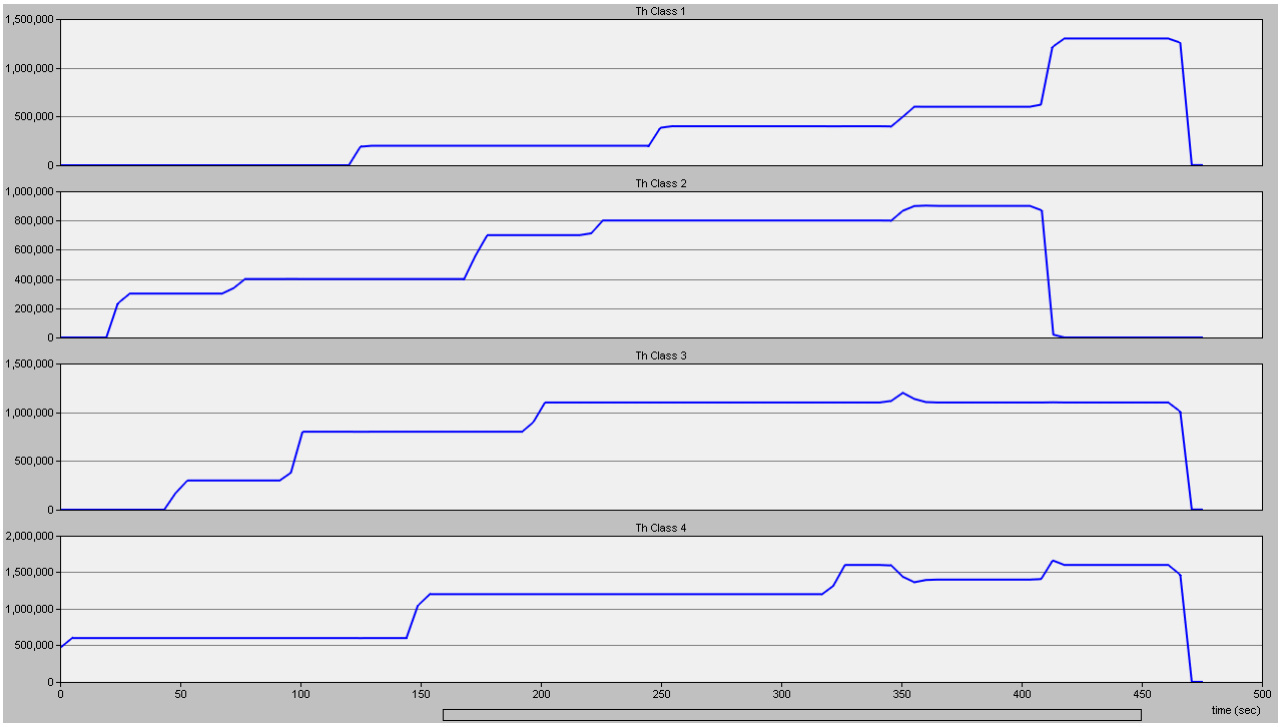


Gráfica 6-12 *Throughput* en el enrutador de borde (Agente_QoS)



Gráfica 6-13 *Throughput* en el enrutador de borde (Agente_QoS1)

Como resultado de la reorganización de los flujos y las nuevas políticas de QoS aplicadas debido a la reorganización, el volumen de tráfico en el enrutador del *backbone* se ve reducido hasta que la notificación de congestión es eliminada. La gráfica 6-14 muestra el comportamiento del *throughput* por cada una de las clases.



Gráfica 6-14 *Throughput* en enrutador de *backbone* (Enrutador_Backbone)

La reducción de flujos garantiza que los enrutadores de *backbone* mantengan fuera de congestión las colas para clases mayores que uno, esto reduce la posibilidad de pérdida de paquetes en los flujos de mayor prioridad.

Bajo los mismos parámetros de tráfico del escenario anterior los resultados obtenidos en cada una de las clases son:

	Retardo (promedio)	Jitter (max)	Throughput (recibido)	Throughput (generado)
Clase 1	146.02 ms	15.53 ms	1300 kbps	800 kbps
Clase 2	48.68 ms	15.95 ms	900 kbps	1100 kbps
Clase 3	47.48 ms	7.7 ms	1100 kpps	900 kbps
Clase 4	66.7 ms	10.7 ms	1600 kbps	2100 kbps

Tabla 6-15 Configuración de generadores de tráfico.

Según los resultados se observa que el agente administrador de QoS mejora ligeramente los resultados relacionados con retardo, *jitter* y *througput*, pues los retardos en la clase 4 se mantienen similares pese al aumento del *throughput*. La reorganización de flujo tiene impacto en el *throughput* de las otras clases incrementando sus valores tal como indica la tabla 6-15.

Para una mayor claridad en el comportamiento en la tabla 6-16 se describen detalladamente las características por flujo.

	Flow Label	Tasa de arrivo (promedio)	Tiempos de Inicio	% Paquetes Recibidos.	Retardo	Jitter (Max)
Clase 1	102	150 kbps	125 seg	95.81 %	79.7 ms	14.4 ms
	1111	-	400 seg	0%	-	-
	301	-	400 seg	0 %	-	-
	112	100 kpbs	250 seg	87.65 %	81.14 ms	14.19 ms
Clase 2	163	250 kbps	50 seg	97.79 %	97.4 ms	15.34 ms
	101	100 kbps	75seg	99.94 %	98.8 ms	14.57 ms
	396	-	375 seg	0 %	-	-
	173	300 kbps	175 seg	99.99 %	87.48 ms	15.57 ms
	111	100 kbps	225 seg	99.95 %	95.91 ms	14.94 ms
Clase 3	181	300 kbps	25 seg	99.99 %	48.11 ms	11 ms
	285	300 kbps	350 seg	99.6 %	148.87 ms	15.95 ms
	191	300 kbps	200 seg	100%	40.63 ms	10.7 ms
Clase 4	143	600 kbps	1 seg	97.81 %	69.55 ms	17.52 ms
	162	500 kbps	100 seg	100 %	46.93 ms	10.87 ms
	174	400 kbps	325 seg	99.32 %	122.42 ms	11.23 ms
	135	600 kbps	150 seg	97.06 %	64.32 ms	17.45 ms

Tabla 6-16 Resultados por flujo en el nodo de resultados.

Los resultados descritos anteriormente describen el comportamiento de cada uno de los flujos, se puede concluir que el porcentaje de perdida resulta menor en los flujos de alta prioridad que son mantenidos en su respectiva prioridad. Los flujos que exceden las capacidades son reducidos a categorías más bajas en las cuales experimentan una mayor congestión, que como resultado se evidencia un aumento en el retardo. Las congestiones experimentadas en el *backbone* implican perdidas de paquetes que en el caso de flujos de alta prioridad son bajas casi despreciables.

Al presentarse la congestión el agente administrador de QoS para no degradar los parámetros de retardo, *jitter* y perdida de paquetes por lo que los flujos que se generan después de la congestión son descartados. Por lo que este comportamiento permite concluir que la notificación de ECN resulta eficiente en la interrupción de tráfico entrante, limitando el número de flujos permitiendo reducir los niveles de descarte de los flujos de mayor prioridad y reduce los flujos de las demás clases a prioridades inferiores. 0

Un aumento del tráfico en el escenario MPLS conlleva directamente a congestión en algunas de las clases AF, aumentando aún más los valores de perdida de paquetes, los valores de perdidas tienden a incrementar a medida que aumenta el trafico sobre la cola congestionada debido a el algoritmo WRED. El agente administrador de QoS trata de evitar la congestión en las clases de

mayor prioridad penalizando flujos de menor prioridad, dicho esquema demostró una eficiencia en la pérdida de paquetes inferior al 3% del total de paquetes, cumpliendo con su funcionamiento a cabalidad. De generarse un aumento en el tráfico el agente administrador de QoS bloquea la recepción de flujos cuando el *backbone* presenta congestión de forma que el tráfico existente no resulte afectado, resultado que en MPLS no es posible, generando mayor degradación del servicio.

En la tabla 6-17 se resumen en comparativa los dos escenarios

	% Paquetes Recibidos (MPLS)	% Paquetes Recibidos (Agente).	Delay (MPLS)	Delay (Agente)	Tiempo de inicio.
Clase 1	100 %	95.81 %	56.6 ms	79.7 ms	125 seg
	100 %	0%	74 ms	-	400 seg
	100 %	0 %	44.3 ms	-	400 seg
	100 %	87.65 %	50.8 ms	81.14 ms	250 seg
Clase 2	85.31 %	97.79 %	97.4 ms	97.4 ms	50 seg
	95.94 %	99.94 %	98.8 ms	98.8 ms	75seg
	97,68 %	0 %	217.3 ms	-	375 seg
	99.97 %	99.99 %	101.34 ms	87.48 ms	175 seg
	99.95 %	99.95 %	110.36 ms	95.91 ms	225 seg
Clase 3	100 %	99.99 %	52.47 ms	48.11 ms	25 seg
	100 %	99.6 %	43.11 ms	148.87 ms	350 seg
	100%	100%	42.71 ms	40.63 ms	200 seg
Clase 4	68.15 %	97.81 %	93 ms	69.55 ms	1 seg
	79.84 %	100 %	100.26 ms	46.93 ms	100 seg
	80.4 %	99.32 %	160.78 ms	122.42 ms	325 seg
	80.23%	97.06 %	98.87 ms	64.32 ms	150 seg

Tabla 6-17 Comparación de parámetros entre escenario bajo MPLS y el Agente administrador de QoS

7 CONCLUSIONES.

El agente administrador de QoS frente a una solución tradicional de MPLS intenta garantizar los niveles de *throughput* de los flujos prioritarios. Para eso, los niveles se basan en la información presente en la cabecera IPv6, gracias a los campos *Flow Label, Traffic Class* y *Explicit Congestion Notification.* Los cuales permiten generar la granularidad necesaria para clasificar los flujos. En mayor medida el campo *Flow Label*, cuyo aporte genera un mayor control sobre el flujo en un nodo.

La organización de flujo es costosa en términos de memoria, una solución en un enrutador de *backbone* resulta contraproducente en términos de QoS pues este recibe una relación de flujos mucho mayor que en los enrutadores de borde; un mayor número de flujos tiene implicaciones en los tiempos de enrutamiento y capacidad de procesamiento. El agente administrador de QoS planteado controla el tráfico en los puntos de acceso a la red (enrutadores de borde), y dejando solo labores de notificación de congestión a los enrutadores de *backbone*. El agente mediante la recepción de una notificación de congestión gracias al campo ECN de IPv6, reorganiza las políticas de admisión de tráfico, como también el nivel de prioridad de cada uno de ellos.

Bajo estas premisas el agente administrador de QoS fue evaluado respecto a MPLS en un escenario de congestión; como resultado se evidenció una notable mejora en valores de j*itter* y retardo para los flujos de más alta prioridad. Se considera un flujo de alta prioridad al tráfico en tiempo real que es sensible al retardo y a la perdida de paquetes. La pérdida de paquetes obtenida con el agente administrador de QoS demuestra una protección del tráfico prioritario frente a problemas de congestión en la red al no ser mayor al 3% de la totalidad de paquetes, sin importar los niveles de tráfico que ingresan en el nodo de borde. El escenario de MPLS si carece de esta funcionalidad y a medida que aumenta el tráfico aumenta la perdida de paquetes.

La mejora de estas características por el agente administrador de QoS tiene como efecto el empeoramiento de las mismas en los flujos de bajo nivel. Sin embargo, los flujos considerados como de baja prioridad son el tráfico poco sensible al tiempo, por ejemplo navegación web, transferencia de archivos etc. Estos flujos basan su funcionamiento en protocolos orientados a conexión como TCP por lo que la pérdida de paquetes activa los mecanismos anti-congestión de este protocolo.

El agente administrador de QoS carece de mecanismos de medida de flujos entrantes, por lo que flujos de alto *throghput* puedan saturar las clases, superando las políticas de administración generando pérdidas de paquetes y alterando los parámetros de retardo y *jitter*. Para mejorar el desempeño del agente administrador de QoS para trabajo futuro se recomienda el desarrollo de un protocolo de aprovisionamiento de QoS entre un host y el enrutador de borde, que permita regular los flujos que ingresan hacia este y permita mayor control al agente.

8 BIBLIOGRAFÍA Y FUENTES DE INFORMACIÓN.

[1] Armitage, G. (2000). *Quality of service in IP Networks: foundations for a multi-service internet* (p. 309). Macmillan Technical Pub. R

[2] Ash, G. R., & Farrel, A. (2008). *Network Quality of Service: Know It All* (p. 330). Elsevier/Morgan Kaufmann Publishers.

[3] Fgee, E. B., Kenney, J. D., Phillips, W. J., Robertson, W., & Sivakumar, S. (2004). Implementing an IPv6 QoS management scheme using flow label & class of service fields. *Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No.04CH37513)* (pp. 1049-1052). IEEE.

[4] Fgee, E.-B., Kenney, J. D., Phillips, W. J., Robertson, W., & Sivakumar, S. (2005). Comparison of QoS Performance between IPv6 QoS Management Model and IntServ and DiffServ QoS Models. *3rd Annual Communication Networks and Services Research Conference (CNSR'05)* (pp. 287-292). IEEE.

[5] Fgee, E.-B., Phillips, W. J., Robertson, W., & Sivakumar, S. C. (2003). Implementing QoS capabilities in IPv6 networks and comparison with MPLS and RSVP. *CCECE 2003 - Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No.03CH37436)* (Vol. 2, pp. 851-854). IEEE.

[6] Fu, D., & Ping, G. (2010). The Discussions on Implementing QoS for IPv6. *International Conference on Multimedia Technology (ICMT)*, pp.1-4.

[7] Hagen, S. (2006). *IPv6 essentials* (Second Edition, p. 418). O'Reilly Media, Inc.

[8] Marsan, M. A. (2005). *Quality of Service in Multiservice IP Networks: Third International Workshop, QoS-IP 2005, Catania, Italy, February 2-4, 2005 : Proceedings (Google eBook)* (p. 656). Springer.

[9] Stergar, J., & Horvat, B. (2000). Int-serv framework architecture and QoS implementation in the IPv6. *International Symposium on Industrial Electronics, 1999. ISIE '99. Proceedings of the IEEE*, Vol. 3, pp. 1214-1217.

[10] Szigeti, T., & Hattingh, C. (2005). *End-To-End QoS Network Design* (p. 734). Cisco Press.

[11] Tang, X., Tang, J., Huang, G.-B., & Siew, C.-K. (2003). QoS provisioning using IPv6 flow label in the internet. *Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint* (Vol. 2, pp. 1253-1257). IEEE.

[12] Joseph, V., & Chapman, B. (2009). *Deploying QoS for Cisco IP and Next-Generation Networks: The Definitive Guide* (p. 495). Morgan Kaufmann.

[13] Medhi.D, & Ramasamy, K. (2007). *Network Routing: Algorithms, Protocols, and Architectures* (p. 823). Morgan Kaufmann.

[14] Cuesta, V., Álvarez, T. (2006). *Algoritmos AQM basados en teoría de control en Opnet*. Universidad de Valladolid. Disponible en línea: http://www.isa.cie.uva.es/~tere/memoria_virginia.pdf (Última consulta Octubre 2012).

[15] Zheng, Lu., & Yang, H. (2012). *Unlocking the Power of OPNET Modeler* (p. 254). Cambridge University Press.

[16] Sethi, A., & Hnatyshin, V. (2009). *The Practical OPNET User Guide for Computer Network Simulator* (p. 527). Chapman and Hall/CRC.

[17] OPNET Modeler Documentation Set Version: 14.5. (2008). OPNET Technologies, Inc.

[18] Gebert, S., Pries, R., Schlosser, D., Heck, K. (2012). *Internet Access Traffic Measurement and Analysis.* University of Würzburg, Institute of Computer Science. Disponible en línea: http://tma2012.ftw.at/TMA/papers/TMA2012paper3.pdf (Última consulta Julio 2012).

[19] Sethi, A., & Hnatyshin, V. (2009). *The Practical OPNET User Guide for Computer Network Simulator* (p. 527). Chapman and Hall/CRC.