Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Departamento de Ingeniería Eléctrica

Sección de Computación

Sistema de Distribución de Tráfico sobre Redes Locales (SDTRL)

Tesis que presenta

Oscar Gonzalo Landa Rosales

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Ingeniería Eléctrica

Director de la Tesis

Dr. Arturo Díaz Pérez

México, D.F. **Diciembre 2004**

Resumen

La tesis presenta el desarrollo de un prototipo denominado Sistema de Distribución de Tráfico sobre Redes Locales (SDTRL), que interconecta dos o más redes de cómputo locales a una red común de trabajo de una organización. La tarea principal del SDTRL es distribuir el tráfico de la red común de trabajo hacia las redes externas, con base en la identificación de los flujos de datos que circulan en su interior. Asimismo, controla el manejo de ancho de banda y prioridad de atención, con la finalidad de poder administrar los recursos de las diversas redes externas a favor de la organización.

El SDTRL está implantado en Linux y es un sistema capaz de efectuar el enrutamiento sobre la base de políticas (EBP), y facilita aún más el control de tráfico tanto de entrada como de salida entre las redes externas y la privada. Este sistema tiene la cualidad de ocultar o enmascarar los conceptos técnicos, e intenta reducir las actividades de configuración con el propósito de facilitar su empleo.

Abstract

This thesis describes the development of a prototype called SDTRL, which connects two or more computer networks to a common working network of an organization. The main task of the SDTRL is to distribute the common working network traffic to the external networks according the identification of the data flows that travel internally. The SDTRL also controls the bandwidth management and attention's priority, whit the objective of managing the resources of the several external networks to the organization's benefit.

The SDTRL has been developed in Linux, it is a system with the capacity of policy-based routing (PBR) and make easier the traffic control in the input and the output between the external and private networks. This system has the quality to hide the technical concepts to the user and it reduces the configuration tasks to make easy its use.

Agradecimientos

Índice general

Resumen III

Abstract V

Agradecimientos VII

1. Introducción 1

2. Conceptos generales sobre redes 7
2.1. Conceptos básicos . . . 7
2.1.1. Modelo de referencia OSI . . . 7
2.1.2. TCP/IP . . . 8
2.1.3. Enrutamiento de TCP/IP . . . 10
2.1.4. Direccionamiento privado, CIDR y NAT . . . 12
2.2. Protocolos de enrutamiento . . . 12
2.2.1. Acerca de protocolos de enrutamiento . . . 13
2.2.2. Clasificación de los IGP´s . . . 14
2.3. Enrutamiento en base a políticas . . . 15
2.4. Calidad de servicio . . . 16
2.4.1. Arquitecturas con calidad de servicio . . . 17
2.4.2. Mecanismos de CdS . . . 20
2.4.3. Herramientas de CdS . . . 21
2.5. Resumen . . . 23

3. Distribución y control de tráfico sobre redes de cómputo 25
3.1. Redes de cómputo con CdS . . . 25
3.1.1. Administración de recursos de una RAL . . . 27
3.2. Distribución y control de tráfico por hardware . . . 28
3.2.1. Ruteadores . . . 29
3.3. Distribución y control de tráfico por software . . . 30
3.3.1. Solaris con Solaris Bandwidth Manager . . . 30
3.3.2. FloodGate-1 . . . 31
3.3.3. BSD con ALTQ . . . 31
3.3.4. Linux con Iproute2 . . . 32
3.3.5. Comparaciones entre el SBM, FG-1, ALTQ e IPROUTE2 . . . 33
3.4. Distribución de tráfico en Linux . . . 34
3.4.1. Sistema de balanceo de carga . . . 35
3.5. Control de tráfico en Linux . . . 35
3.5.1. Disciplinas de colas . . . 36
3.5.2. Filtros para clasificar paquetes . . . 39

3.5.3. La arquitectura de Linux para la CdS . . . 39
3.5.4. Dispositivo de intermedio de encolado (DIE) . . . 41
3.5.5. IPTABLES . . . 42
3.6. Resumen . . . 43

4. Sistema de distribución de tráfico entre redes Locales **45**
4.1. Solución del caso de estudio . . . 45
4.1.1. Escenarios de trabajo del SDTRL . . . 46
4.1.2. Restricciones del SDTRL . . . 48
4.1.3. Funciones del SDTRL . . . 49
4.2. Marco de referencia conceptual para el SDTRL . . . 49
4.3. Distribución de tráfico del SDTRL . . . 51
4.4. Control de Trafico del SDTRL . . . 52
4.5. Funcionamiento General del SDTRL . . . 55
4.6. El SDTRL en Linux . . . 57
4.6.1. La base de tecnológica . . . 57
4.6.2. Modelo de configuración base para el SDTRL . . . 57
4.7. SDTRL implantado en Linux . . . 59
4.8. Resumen . . . 60

5. Diseño e implementación del SDTRL **63**
5.1. Arquitectura del sistema . . . 63
5.2. Sistema de Inicialización del SDTRL . . . 64
5.2.1. Procedimientos base . . . 65
5.2.2. Procedimientos intermedios de configuración . . . 66
5.2.3. Acceso dividido para los nodos . . . 67
5.2.4. Construcción de reglas de SNAT . . . 69
5.2.5. Balanceo de carga . . . 71
5.3. Sistema de asignación de recursos particulares . . . 74
5.4. Sistema de asignación de recursos masivo . . . 77
5.5. Sistema de asignación de recursos de respuesta . . . 80
5.6. Resumen . . . 82

6. La instrucción sdtrc **83**
6.1. Descripción de operaciones para el sistema . . . 83
6.2. Escenario utilizado . . . 83
6.3. La instrucción **sdtrc** . . . 86
6.4. Modo de operación del sdtrc . . . 90
6.4.1. Configuración inicial del SDTDL . . . 91
6.4.2. Tráfico particular . . . 94
6.4.3. Tráfico masivo . . . 98
6.4.4. Tráfico de respuesta . . . 100
6.5. Resumen . . . 101

7. Conclusiones **103**

A. Descripción del filtro u32 **107**

Bibliografía **111**

Índice de figuras

1.1. Un sistema troncal redundante. 2
1.2. Integración de dos proveedores de servicios de red. 2
1.3. Topología del caso de estudio. Donde un sistema intermedio tal, se encarga de encaminar y controlar las diversas clase de tráfico, en base a las exigencias de la organización. 4

2.1. Proceso de encapsulamiento en el modelo de referencia OSI. 8
2.2. Interconexión de redes con sistemas abiertos. 9
2.3. Clases de direcciones de IPv4, que describen un comportamiento jerárquico. 10
2.4. Procesos de entrada ha un nodo externo de Servicios Diferenciados 19

3.1. Descripción general de un sistema de distribución y control de tráfico. . . 28
3.2. Lista de reglas de la BDPE, que describe prioridad, el selector y la tabla a usar. 35
3.3. Cada interfaz de red asociada a Linux tiene una disciplina de cola, que por, omisión es la disciplina de cola Pfifo_fast. 37
3.4. Arquitectura de Linux para el manejo de paquetes. 40
3.5. Procedimiento de encolamiento y desencolado de paquetes dentro de una Qdisc. 41
3.6. Flujo de paquetes cuando se activa los DIE's en el núcleo de Linux. . . . 41

4.1. Los usuarios de la organización utilizan los servicios de distintas redes externas. 46
4.2. Utilización de varios PSI's para una organización. 47
4.3. Arquitectura recomendable de trabajo en cuestiones de seguridad. 48
4.4. Acceso dividido. Sale o entra una comunicación por el mismo lugar que entró o salió respectivamente. 52
4.5. Los tráficos X, Y y Z pertenecen a una clase de tráfico particular, y se encaminan a distintos nodos. 54
4.6. Módulos generales del SDTRL. 55
4.7. Propuesta de la arquitectura General del SDTRL en la plataforma Linux. 58
4.8. Diagrama de interrelación de paquetes de software y módulos de programas para implementar el SDTRL. 60

5.1. Diagrama de bloques del SDTRL. 64
5.2. Diagrama de flujos del sistema de inicialización. 65
5.3. Diagrama de flujo de los procesos intermedios de configuración. 67
5.4. Modelo jerárquico del SDTRL. 70
5.5. Diagrama que muestra el recorrido que exhibe los paquetes ipv4 a través de sistema, según la arquitectura Netfilter. 75

5.6. Diseño del sistema de enrutamiento en base a políticas. 76
5.7. Fragmentación de ancho de banda de una interfaz de red con clases de trafico excluyentes. 78
5.8. Trayectorias de los paquetes dentro Qdisc de la interfaz de red para el tráfico masivo. 80
5.9. Diagrama de flujo de paquetes proveniente de las redes externas en dirección a las red local. 81

6.1. Topología de la red para pruebas y desarrollo del SDTRL. 85
6.2. Ingreso de los nodos de red al SDTRL. 91
6.3. Inicialización del SDTRL con dos nodos de red externos. 92
6.4. Tablas de enrutamiento creadas en la inicialización. 93
6.5. Inicialización del SDTRL con un nodo compuesto de dos conexiones de red. 94
6.6. Estructuras de las tablas de ruteo en presencia de un nodo de red compuesto. 95
6.7. Ejemplificación de la forma de aplicar políticas de enrutamiento. 96
6.8. Ejemplificación del uso de las políticas de enrutamiento con un nodo compuesto. 97
6.9. Reglas de tráfico particular que combinan distribución y control de tráfico. 98
6.10. Ejemplificación del uso de las reglas tráfico masivo para la interfaz red eth2. 99
6.11. Control de tráfico de entrada en la red privada de uso común. 100

Índice de cuadros

2.1. Métricas más utilizadas para los protocolos de enrutamiento. 13
2.2. Comparativas entre la tecnología vector distancia y el estado del enlace. . 15
2.3. Tabla comparativa entre las dos principales arquitecturas de QoS. 20

3.1. Puntos comparativos entre los diferentes sistemas de administración. . . . 33
3.2. Colas disponibles en Linux. 36

4.1. Puntos de distinción entre clases de tráficos. 51

6.1. Actividades generales del SDTRL. 84

A.1. Selectores específicos. 108

Capítulo 1

Introducción

La creciente demanda de servicios de telecomunicaciones con características especiales, así como la constante integración que ha tenido el mercado mundial a nivel de servicios, ha llevado a las redes de telecomunicaciones a abrigar la necesidad de interconectarse de formas diversas y sofisticadas. Muchas de las instituciones educativas y científicas así como organizaciones empresariales tienen la necesidad de utilizar diferentes tipos de redes de cómputo.

Tradicionalmente, una organización contaba con un solo proveedor de servicios de red de cómputo. Hoy en día esto no es así, hay varios factores por los cuales es necesario el uso de dos o más puntos de conexión a redes para una sola organización. Esto puede darse por varias razones. La primera y más usada es la de un sistema de redundancia a una red de cómputo, garantizando la disponibilidad de la operación. La segunda razón es para proporcionar servicios diferenciados a los usuarios de una organización. Por ejemplo, mientras un enlace se puede ser usado para proporcionar servicios generales, un enlace adicional puede ser utilizado para servicios de misión crítica.

Lo anterior evidencia la necesidad de priorizar las comunicaciones que se genera dentro de la organización a fin de poder distribuir el tráfico de acuerdo a su importancia y utilidad. Por otra parte, el poder tener varios puntos de interconexión a una red permite, entre otras cosas, hacer un uso eficiente de los canales de comunicación mediante un balanceo de carga.

Un escenario donde la distribución de tráfico es punto crucial, se presenta en el sistema troncal redundante. La figura 1.1 ilustra como los ruteadores garantizan la confiabilidad de la operación, disponibilidad y mantenimiento en días críticos de la red. Un buen diseño de red es tal que, si el sistema troncal primario falla, un sistema troncal secundario esté disponible como un inmediato y automático respaldo.

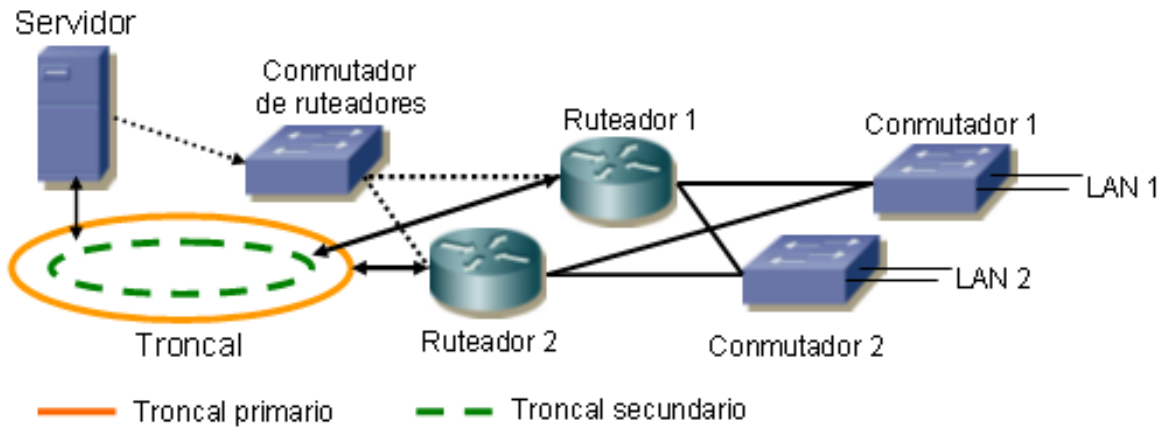

Figura 1.1: Un sistema troncal redundante.

Una vicisitud de la forma anterior de distribuir el tráfico, aparece cuando se tiene a dos proveedores de servicios de red (posiblemente de tecnologías diferentes), los cuales se interconectarán con una tercera red, que buscará hacer uso de estos dos servicios, muy probablemente para propósitos diferentes (ver Figura 1.2).

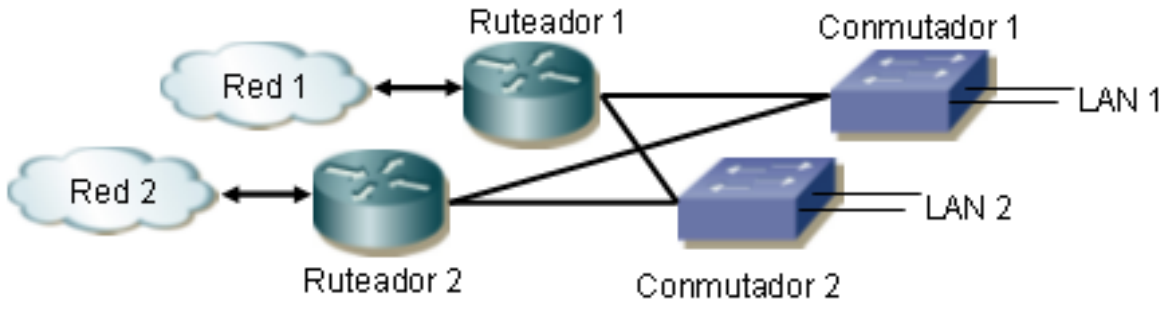

Figura 1.2: Integración de dos proveedores de servicios de red.

El problema fundamental para la utilización de dos o más canales de comunicación es tener la capacidad de distribuir el tráfico de acuerdo a criterios diversos. La forma actual de realizar la distribución de tráfico sobre las diferentes redes de cómputo existentes es utilizando los llamados sistemas intermedios (ruteadores y puertas), los cuales se encargan de identificar, controlar y distribuir el tráfico a través de dichas redes.

Los sistemas intermedios realizan dos funciones básicas:

1. Crear y mantener tablas de ruteo para cada capa de protocolo de red. De esta manera, el ruteador extrae de la capa de red la dirección destino y realiza una decisión de envió basado sobre el contenido de la especificación del protocolo en la tabla de ruteo.

2. Un sistema intermedio permite seleccionar la mejor ruta, basándose sobre diversos factores, en vez de simplemente la dirección destino. Estos factores pueden incluir la cuenta de saltos, velocidad de la línea, costo de transmisión, retraso y condiciones de tráfico.

La tecnología de ruteo es eficiente para los trabajos de distribución descritos en las Figuras 1.1 y 1.2, salvo que las políticas usuales están basadas fundamentalmente en la dirección destino del paquete a transmitir. Sin embargo, es necesario tener la posibilidad de establecer criterios diferentes para la distribución de tráfico. Es decir, sería muy conveniente poder distinguir el tráfico de una aplicación de usuario y, con base en alguna política, enviarlo por un canal de comunicación adecuado.

Actualmente existen ruteadores ya sea en base a hardware o software que incorporan técnicas de aceleración de tráfico para evitar el problema de sobrecarga de trabajo. Normalmente, estas nuevas características que adquiere el ruteador, dan pie a pensar en sistemas que brindan Calidad de Servicio. El desarrollo de los ruteadores y de otros sistemas intermedios se está enfocando hacia la capacidad de garantizar ciertas características de calidad en la transmisión de información. El objetivo es evitar la congestión de determinados nodos de la red, previniendo así que se vean afectados algunos flujos de paquetes específicos que requieran un especial ancho de banda o retardo. Para solucionar este problema existen dos tendencias [1]:

- Sobredimensionar adecuadamente la red de cómputo, lo que implica aumentar cuando resulte necesario los equipos de conmutación así como el ancho de banda disponible en los canales.

- Gestionar de forma eficiente los recursos disponibles, de tal modo que se reparta de manera desigual entre los diferentes flujos de tráfico, con base en la conveniencia de uso. Esto requiere que los mecanismos de calidad de servicio sean capaces de distinguir los flujos de tráfico y asignarles recursos en la red.

El problema bajo estudio en este trabajo de tesis considera la necesidad de utilizar dos o más redes de cómputo dentro de un lugar de trabajo. El planteamiento se muestra en la Figura 1.3.

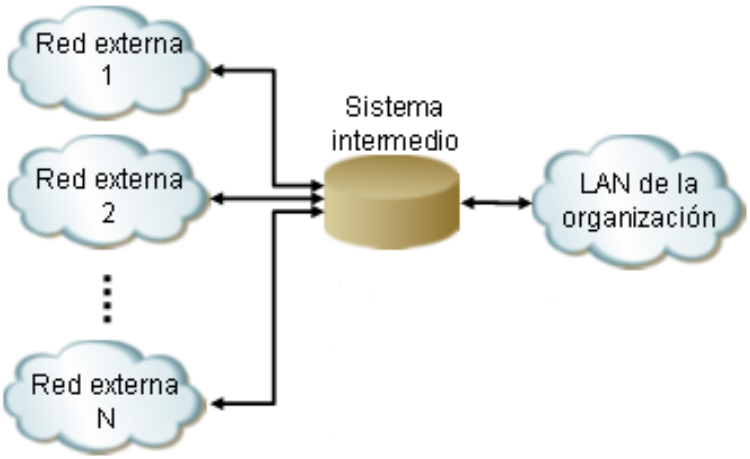

Figura 1.3: Topología del caso de estudio. Donde un sistema intermedio tal, se encarga de encaminar y controlar las diversas clase de tráfico, en base a las exigencias de la organización.

Los puntos a considerar para la solución del problema son los siguientes:

- La interconexión de dos o más redes de cómputo, para poder proporcionar servicios a una red común de trabajo.
- La distribución del tráfico dentro del conjunto de redes debe ser en base a los requerimientos de la organización y de los usuarios. Es decir, la capacidad de poder reservar o asignar recursos para diferentes clases de tráfico, y el manejo de prioridades entre ellas.

Los requerimientos de la organización y de los usuarios se desprenden de la diversidad de trabajos que se realizan en la actualidad. Se debe de considerar que los servicios que proporciona una red de cómputo no pueden ser proporcionales para cada usuario que labora en una organización. Es decir, si consideramos que no todas las personas tienen la misma responsabilidad, entonces no todas tienen la misma necesidad de recursos y privilegios en una red. De aquí que surge la necesidad de poder distinguir los servicios de una red para cada uno de los usuarios, respetando los interese de la organización.

De lo anterior se concluye, que se tiene como propósito controlar y administrar los recursos de entrada y salida de una red, cuando se tiene dos o más redes de cómputo a su disposición.

El objetivo general de este trabajo de tesis es desarrollar un mecanismo que pueda identificar, controlar y distribuir el tráfico de paquetes de dos o más redes de computadoras permitiendo el acceso a los servicios de éstas en una sola red común de trabajo, manteniendo transparencia para el usuario en su uso.

De manera específica se busca cumplir con los siguientes objetivos particulares:

- Construir un prototipo que permita distribuir tráfico entre diferentes redes de cómputo.
- Tener la capacidad de encaminar el tráfico en base a ciertas características del encabezado de los paquetes a nivel 3 y 4 del modelo OSI.
- Tener la capacidad de administrar adecuadamente el medio de comunicación en base a dirección fuente y destinos, tipo de protocolo, y puertos fuente y destino.
- Integración de los diferentes grupos de trabajo que utilizan cada una de las redes de cómputo de una manera transparente. Además, tener una administración y control de manera simple de la red.

Para alcanzar los objetivos propuestos se ha desarrollado un "Sistema de Distribución de Tráfico sobre Redes Locales"(SDTRL) cuya construcción ha considerado el desarrollo de la siguiente metodología:

Modelado del tráfico. En primer lugar es necesario modelar el tráfico del sistema para poder integrar distintos tipos de tráfico (llamadas clases), identificando sus propias necesidades y políticas sobre una red cómputo con recursos evidentemente finitos dentro de la organización.

Identificación del tráfico. Para poder priorizar cierto tipo de tráfico sobre otro se requiere un proceso previo de identificación. Los mecanismos habituales para realizar esta identificación son: por dirección IP origen o destino y tipo de protocolo de transporte (TCP o UDP); o por número de puerto TCP/UDP origen o destino usado por la aplicación en cuestión y entre otras. Básicamente la identificación se realiza con los datos

de encabezado de nivel 3 y nivel 4 del modelo OSI.

Marcado de paquetes. Para permitir que la red pueda manejar de forma diferenciada los distintos tipos de tráfico identificados, es necesario que éstos sean marcados con una determinada etiqueta. Este proceso debe producirse en los extremos de la red, representados por los sistemas de ruteo de la LAN.

Asignación de recursos. La asignación de recursos se limita a la reserva de espacio dentro de los búferes internos de los equipos, salida preferencial hacia alguna red de alta prioridad, y al uso de ciertos algoritmos de priorización para optimizar la transmisión de los paquetes.

Muchos elementos de comunicaciones soportan mecanismos de encolado con prioridad estricta, de tal manera que siempre que exista tráfico en el buffer de mayor prioridad, se transmitirá en primer lugar. Otros equipos soportan mecanismos más eficientes de encolado como WFQ [2], que asigna una determinada cantidad del buffer a cada tipo de tráfico. De esta manera, todas las aplicaciones pueden recibir los recursos mínimos que necesitan.

Gestión de la congestión. Si bien los mecanismos de encolado garantizan un ancho de banda mínimo a cada tipo de tráfico. Además de ello, es indispensable tener algún mecanismo que evite situaciones de colapso y descarte de paquetes, por lo que la gestión de la congestión se encarga de prever estas situaciones.

Este documento está organizado en 7 capítulos: El primero presenta una introducción, objetivos y la organización del documento. El capítulo 2 hace una remembranza de los temas involucrados en esta tesis; como son el enrutamiento y calidad de servicio en redes de cómputo. El capítulo 3 alude los sistemas de distribución de tráfico actuales basados en hardware y software. El capítulo 4 describe los requerimientos, necesidades y perspectivas de la propuesta del SDTRL. El capítulo 5 hace una reseña de la forma de cómo fue implementado el SDTRL. El capítulo 6 habla de la instrucción **sdtrc** y de las pruebas del prototipo. El último capítulo presenta las conclusiones de la tesis.

Capítulo 2

Conceptos generales sobre redes

Para solucionar el problema de una comunicación mucho más fácil entre los usuarios de una red, se agrega entre el hardware de comunicación y las aplicaciones un módulo de red encargado de manejar de manera automática los problemas y detalles de comunicación. Por ello, los diseñadores han decidido dividir el problema de la comunicación en partes. Cada parte que participa en la comunicación acuerdan un conjunto de reglas, formatos de los mensajes y las acciones adecuadas que servirán para el intercambio de información, y esto es conocido como protocolos de red o protocolos de comunicación.

2.1. Conceptos básicos

Como primera instancia, se describe los conceptos más sobresalientes sobre la transmisión de datos y comunicación entre sistemas diversos, además de mencionar algunas normas estándares internacionales.

2.1.1. Modelo de referencia OSI

El *modelo de referencia de interconexión de sistemas abiertos*, mejor conocido como OSI, proporciona un conjunto de estándares que aseguraron una mayor compatibilidad e interoperabilidad entre los distintos tipos de tecnología de red.

La Organización Internacional de Normas (ISO) dividió el modelo de referencia OSI en siete capas. Cada capa tiene una interfaz bien definida con las capas que están inmediatamente arriba y debajo de ella, definiendo servicios a la capa adyacente superior. La descripción esquemática de las diversas capas que componen este modelo es como sigue:

- La capa física tiene la misión principal de transmitir bits por un canal de comunicación.
- En la capa de enlace de datos convierte el medio de transmisión en un medio libre de errores de cualquier tipo.
- La capa de red proporciona conectividad y selección de la mejor ruta, para la comunicación entre máquinas, ubicadas en redes geográficamente distintas.
- La capa de transporte especifica los detalles para mantener y terminar adecuadamente los circuitos virtuales, proporcionando un servicio confiable.
- La capa de sesión provee el medio necesario para que las entidades, organicen y sincronicen su diálogo, y procedan al intercambio de datos.
- La capa de presentación garantiza que la información pueda ser entendida y utilizada por las capas de aplicación de los diferentes sistemas terminales.
- La capa de aplicación está relacionada con las funciones de más alto nivel, proporcionando soporte a las aplicaciones o actividades del sistema.

Una consecuencia implícita dentro del modelo de capas es el encapsulamiento datos, una técnica donde la información a enviar se coloca en el área de datos de un paquete o cuadro (ver Figura 2.1).

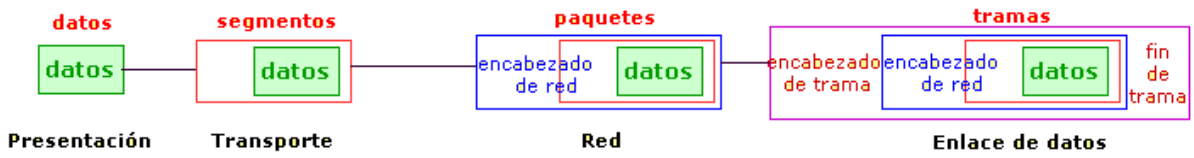

Figura 2.1: Proceso de encapsulamiento en el modelo de referencia OSI.

2.1.2. TCP/IP

La pila de protocolos TCP/IP, también llamado modelo de capas de interred, permite que se comuniquen cualquier par de computadoras, a pesar de las diferencias de hardware. El software del protocolo TCP/IP funciona bien y maneja interredes grandes. Además de usarse en muchas interredes privadas, también se emplea en la Internet global.

El modelo de referencia OSI y TCP/IP tienen puntos en común. Ambos son de una arquitectura de capas, salvo que TCP/IP sólo tiene cuatro, y se basan en el concepto de protocolos independientes. La Figura 2.2 muestra un intento de establecer una correspondencia entre las diferentes capas de las dos arquitecturas, pero se deben tomar en cuenta las discrepancias básicas de las mismas, explicadas en [3].

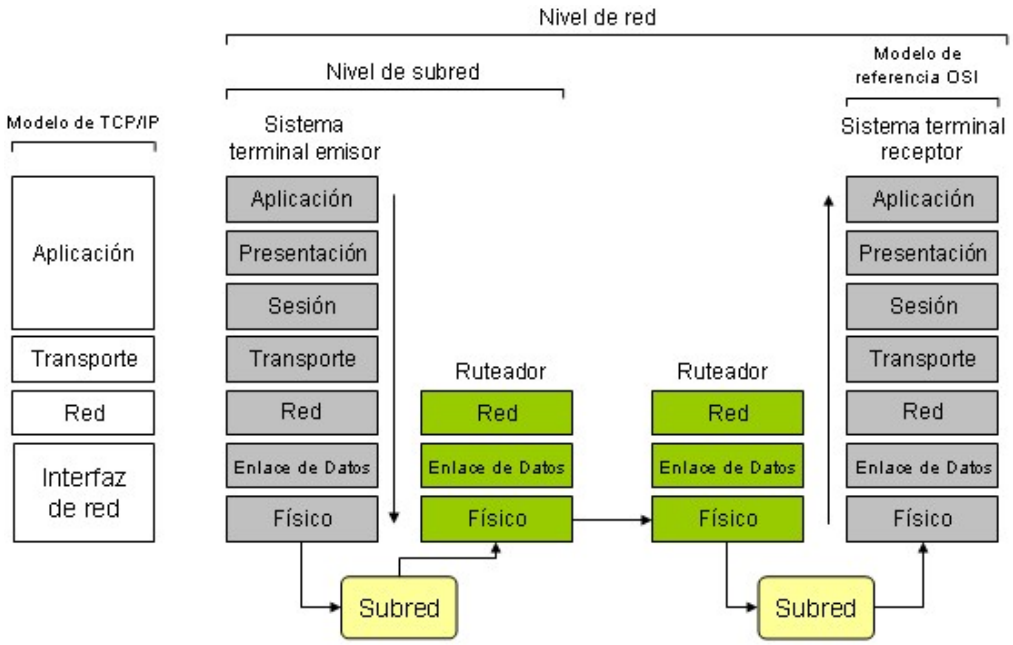

Figura 2.2: Interconexión de redes con sistemas abiertos.

Una de las capas de mayor importancia en TCP/IP es la capa de red, encargada de llevar los paquetes desde el origen hasta el destino. Dotando a cada sistema terminal de un servicio de red extremo a extremo que abarca toda la interred. Para realizar estas tareas, la capa de red se ocupa de la determinación de la mejor ruta y la conmutación de paquetes a través de diferentes redes.

El Protocolo Internet (IP) es sólo uno de los protocolos en la capa de red asociados a TCP/IP, que aparece como el elemento integrador de sistemas, capaz de hacer converger todas las necesidades de comunicación en una misma infraestructura. Esto se debe principalmente a que las redes IP son abiertas, flexibles, robustas y estandarizadas.

Para dar la apariencia a una interred de un sistema único y transparente, todos los sistemas terminales deben usar un esquema de direccionamiento uniforme. Dentro la pila de protocolos TCP/IP, el direccionamiento lo especifica el protocolo de Internet. La norma IP versión 4 (IPv4) especifica que cada sistema terminal recibe un número único de 32 bits conocido como dirección IP o de internet.

Para hacer posible la orientación de los paquetes en TCP/IP, por lo menos cada sistema terminal necesita una dirección IP, que identifica tanto un sistema terminal concreto como la red a la que éste pertenece, ya que el sistema de direcciones IP es jerárquico (ver Figura 2.3).

Figura 2.3: Clases de direcciones de IPv4, que describen un comportamiento jerárquico.

Existen dos direcciones de red asociadas a un sistema terminal conectado a una interred. En términos de TCP/IP, estas son la dirección del punto de conexión a la red (dirección física - MAC) y la dirección de IP. La dirección física es diferente en cada tipo de red o subred, en tanto que la dirección de IP es un identificador único para toda la interred.

2.1.3. Enrutamiento de TCP/IP

A partir de una dirección IP, un sistema terminal puede determinar si los datos deben ser enviados a otro miembro de la red, o a través de un sistema intermedio (también conocido como ruteador o puerta de enlace) en trayectoria a una tercera red. Los mecanismos que permiten encaminar los paquetes dentro de una red local lo realizan forzosamente con la dirección física del sistema terminal destino.

El protocolo de resolución de direcciones (ARP) permite a un sistema terminal encontrar la dirección física de otro miembro de la misma red, con sólo proporcionar la dirección IP de su objetivo. La manera de lograrlo es difundiendo un mensaje, denominado petición ARP, a todas las demás máquinas de su red, para preguntar, por la dirección física pertenece a una dirección IP. La respuesta surge únicamente en manos de la máquina con dicha IP, de forma que envía su dirección física.

La dirección física obtenida se guarda luego en una tabla de relación, dirección lógica

- dirección física, denominadas tablas ARP o caché ARP. De esta manera, los próximos envíos al mismo destinatario no será ya necesario realizar nuevas peticiones ARP, pues su dirección física es conocida. Por ello, estas tablas son fundamentales para la labor y el rendimiento óptimo de una red.

Cada ruteador posee una tabla de enrutamiento, en la que figuran los diferentes ruteadores conectados a su lado, y una tabla de máscaras[1] y de direcciones de red. Las tablas de enrutamiento no son fijas, el mismo ruteador puede ir modificándolas dinámicamente de acuerdo con el estado de la red y de los ruteadores adyacentes. Este es el motivo por el cual, los paquetes pertenecientes a una misma conexión pueden ser dirigidos por caminos diferentes.

La forma de trabajar de un ruteador consiste en extraer, del paquete que recibe, la dirección IP del sistema terminal destino, y realiza la operación AND lógica entre ésta IP y las diferentes máscaras de red que posee. Ello definirá, si el paquete tiene como destino alguna de las redes que conecta el ruteador. En caso contrario, el ruteador mirará en sus tablas de ruteo internas para ver a dónde debe mandar el paquete; destino que generalmente será otro ruteador. Se produce así una serie de saltos, donde el paquete es enviado a sucesivos ruteadores, hasta llegar a uno, en el que se comunica con la red destino del paquete. Finalmente dicho ruteador, será encargado de entregar el paquete al sistema terminal apropiado.

El concepto de subred fue introducido para desacoplar los ruteadores asociados a un solo sitio de la función de enrutamiento global en la interred. Es decir, permite la división de una red en varias partes (segmentos de red) para uso interno, pero aun actúa como una sola red ante el mundo exterior. Para lograr esto, en lugar de que cada red de área local (LAN) asociada a un sitio tenga su propio identificador de red, sólo se asignará un identificador de red al sitio completo.

Las subredes no son visibles fuera de la red, por lo que la asignación de una nueva subred no requiere la modificación de las tablas de ruteo externas. Al igual que, la presencia de un número posiblemente grande de subredes y ruteadores asociados resultará transparente para el sistema intermedio de la interred.

[1] La máscara de red tiene la propiedad de que cuando se realiza una operación AND lógica con la dirección IP de un sistema terminal se obtiene la dirección propia de la red, donde se localiza el sistema.

2.1.4. Direccionamiento privado, CIDR y NAT

Aunque el número de direcciones posibles de IPv4 parece suficiente (2^{32}), éstas han sido agotadas prácticamente en la actualidad según el NIC (Centro de Información de Redes). Para suplir esta carencia, así como para solventar otros problemas inherentes a su propia naturaleza, se implementaron varias técnicas.

Desde el punto de vista de su accesibilidad podemos clasificar las direcciones IP en públicas y privadas. Las direcciones IP públicas son aquellas que son visibles por todos los sistemas terminales conectados a Internet. Las direcciones IP privadas son visibles únicamente por los sistemas terminales de su propia red o de otra red privada interconectada por medio de ruteadores.

Frecuentemente, para las organizaciones, las redes de clase A son demasiado grandes, y una red clase C es demasiado pequeña, entonces la opción por eliminación es la clase B. En realidad una red clase B es demasiado grande para la mayoría de las organizaciones. La solución que da el enrutamiento interdominios sin clases (CIDR) a este problema consiste en repartir las redes clase C restantes, en bloques de redes contiguas de tamaño variable a una organización de acuerdo a sus necesidades. Asimismo, el mundo se dividió en cuatro zonas, con el propósito de comprimir todo una zona en una solo entrada en la tabla de ruteo [4].

Otra de las técnicas para superar la carencia de direcciones IP en una organización es conocida como traducción de direcciones de red (NAT). Lo que hace básicamente está técnica es alterar las cabeceras de los paquetes IP (principalmente las direcciones de red) y mantener un registro de entrada y salida de los paquetes modificados, para poder alterar los paquetes respuesta de igual forma. NAT trabaja a nivel de la capa de red, lo cual es transparente el procedimiento de alterar la dirección fuente (SNAT: NAT por origen) o destino (DNAT: NAT por destino) de un paquete. Las aplicaciones de NAT son muchísimas, aunque actualmente, las aplicaciones más conocidas son lo que se conoce como enmascaramiento y balanceo de carga [5].

2.2. Protocolos de enrutamiento

La principal dificultad del enrutamiento radica en cómo los sistema intermedios, y los sistemas terminales de la interred consiguen y mantienen la ruta óptima hacia el sistema terminal destino.

Tabla 2.1: Métricas más utilizadas para los protocolos de enrutamiento.

Métrica	Descrición
Fiabilidad	La proporción de errores de cada enlace de red.
Número de saltos	El número de ruteadores que debe atravesar un paquete para alcanzar un destino.
Ancho de banda	La capacidad máxima de transporte de información en un enlace.
Retraso	El tiempo necesario para trasladar un paquete de extremo a extremo en la red.
Carga	El grado de uso de un enlace.
Coste	El valor arbitrario, generalmente basado en el ancho de banda, coste en dólares u otra medida, que asigna un administrador de red.

La optimización de la ruta, que habitualmente es una función de los sistemas intermedios, se logra utilizando la tabla de ruteo creada por los protocolos de enrutamiento. Estos protocolos permiten que se mantenga el contenido de cada tabla en forma dinámica y distribuida. Además, ellos desencadenan que los ruteadores interconectados creen un mapa interno de los demás ruteadores de la interred.

La habilidad que tiene el protocolo de enrutamiento de seleccionar la mejor ruta depende de la métrica (Tabla 2.1) y peso que se use, para realizar la operación de optimización con base en un procedimiento específico.

2.2.1. Acerca de protocolos de enrutamiento

Una red independiente que forma parte de una interred, con frecuencia se denomina sistema autónomo (SA). Cada sistema autónomo tiene sus propios algoritmos de enrutamiento y autoridad gestora. Usualmente, una puerta de enlace permite el ingresar a un SA, logrando ocultar la estructura interna de la red del resto de la interred.

Si la red tiene más de una puerta de enlace y estas pueden hablar entre sí, sé consideran vecinos interiores. Asimismo, las puertas de enlace que pertenecen a sistemas autónomos diferentes, utilizadas para conectar un sistema autónomo a la red central, son conocidas como puertas de enlace exteriores.

Dentro de una red, la negociación de la transferencia de información de enrutamiento entre las puestas de enlace internas se realiza con el Protocolo de Puerta de enlace Interior (IGP), el cual puede ser [2], [6]: el Protocolo de Información de Enrutamiento

(RIP), el Protocolo de Enrutamiento de Puerta de enlace Interior (IGRP), el Protocolo de Enrutamiento de Puerta de enlace Interior Mejorado (EIGRP), el protocolo de abrir primero la ruta libre más corta (OSPF) o el menos común el protocolo HELLO.

El Protocolo de Puerta de enlace Exterior (EGP) es utilizado para el intercambio de información de encaminamiento entre puertas de enlace exteriores. Impidiendo que pase demasiada información (que no sea útil a otras puertas de enlace) a través de las redes. EGP se basa en el sondeo periódico de solicitudes seguidas por respuestas, para monitorear la accesibilidad de los vecinos y para preguntar si hay solicitudes de actualización. De esta forma, una puerta de enlace exterior que usa EGP pasa información a sus vecinos EGP's pero no anuncia la información de accesibilidad de estas fuera del SA.

La información completa de enrutamiento se organiza de forma jerárquica. Los sistemas terminales mantienen suficiente información para reenviar datagramas a uno o más puertas de enlace conectadas a la misma red. Por su parte, las puertas de enlace envían datagramas a otras puertas de enlace internas dentro del mismo sistema autónomo. En la parte más alta de la jerarquía están las puertas de enlace externas, que poseen la información para reenviar datagramas a puertas de enlace internas de su propio SA o a otras puertas de enlace externas.

Como parte de la descripción de los protocolos de enrutamiento, se hace mención a continuación de manera esbozada de los tipos clases de IGP's. En tanto que, los EGP's son tópicos que salen del alcance de esta tesis.

2.2.2. Clasificación de los IGP´s

Los protocolos de enrutamiento IGP's pueden clasificarse de la siguiente forma: (1) vector distancia, (2) estado de enlace e (3) híbrido. La Tabla 2.2 establece comparativas puntuales entre las dos formas principales de determinar la mejor ruta a un destino.

Además de los protocolos de estado del enlace y de los protocolos de vector distancia también se dispone de los protocolos de enrutamiento híbridos. Estos protocolos emplean la métrica de los protocolos de vector distancia, sin embargo utilizan bases de datos para las actualizaciones de los cambios de topología, al igual que los protocolos de estado del enlace. Es decir, combinan la facilidad de uso de los protocolos de ruteo de vector de distancia tradicional con las capacidades de ruteo rápido de los protocolos de estado del enlace. Un ejemplo de protocolo híbrido sería EIGRP [7].

El especificar que un protocolo es mejor que otro, sería muy pretencioso. Pues cada

Tabla 2.2: Comparativas entre la tecnología vector distancia y el estado del enlace.

Vector distancia	Estado del enlace
Protocolos con esta tecnología: RIP, IGRP.	Protocolos con esta tecnología: OSPF.
Vista de la topología de la red desde la perspectiva del vecino.	Consigue una vista común de toda la topología de la red.
Pasa copias de la tabla de enrutamiento a los ruteadores vecinos.	Pasa las actualizaciones de enrutamiento de estado del enlace a los otros ruteadores.
Añade vectores de distancias de ruteador a ruteador.	Calcula la ruta más corta hasta otros ruteador.
Frecuentes actualizaciones periódicas.	Actualizaciones activadas por eventos.
Presenta una convergencia lenta.	Presenta una convergencia rápida.
No siempre se obtiene la mejor ruta posible.	Tiene un buen sistema de optimización de rutas, en base a grafos.

uno de ellos tiene un ámbito en donde se desenvuelve de manera adecuada. Las formas de seleccionar un protocolo se distinguen en las métricas que emplea para seleccionar rutas, y en la forma de que se almacenan y se intercambian las actualizaciones de la tabla de enrutamiento.

2.3. Enrutamiento en base a políticas

La distribución de tráfico es un concepto que permite dar un trato diferenciado a distintos tipos de tráficos en base a características específicas de las cabeceras de los paquetes. Los mecanismos de distribución están más allá del tradicional encaminamiento por dirección destino, permiten la identificación, clasificación y orientación de los paquetes de una forma más abierta. Entre los mecanismos que se encuentran son: el enrutamiento a base de políticas (EBP) [6] , las listas de control de acceso (LCA) y reconocimiento a base de red de aplicación (NBAR). Se explorará más sobre el tema de EBP por que es el mecanismo que se emplea en esta tesis.

El EBP o RFC 1104 potencializa la forma de distribuir el tráfico mediante la determinación de políticas[2], proporcionando un mecanismo de identificación de paquetes de modo que ciertas clases del tráfico reciban un servicio diferenciado preferente. Además, en conjunto con una técnica de encolamiento, proporcionan un instrumento sumamente poderoso, simple y flexible para administrar la red [8].

[2]Las políticas en su forma más fácil, son las reglas que describen las acciones que se tienen que llevar a cabo cuando una condición específica ocurre.

El EBP es usado para dirigir paquetes a caminos (interfaces de salida) que dependen más de la conveniencia e interés de la organización. La distribución basada en este criterio se desarrolla a través de una central de conmutación (ruteador). Esto provee un mecanismo más flexible para encaminar a los paquetes [9]. Por supuesto, el verdadero encargado de dirigir los paquetes a su destino es el protocolo enrutado (IP) con la ayuda de los protocolos de enrutamiento.

En cuanto a las ventajas de utilizar el EBP, éste nos permite, que la distribución del tráfico no sea solo en base a la dirección IP destino, si no en base a otras características de los valores del encabezado de los paquetes. Tales como la dirección IP, puerto TCP, protocolo, tamaño del paquete, etc.

La clasificación de los paquetes en el EBP se realiza definiendo una clase para cada tráfico identificado. De acuerdo con la clasificación se aplican distintas políticas, para asegurar una salida para cada clase, establecido por los criterios del administrador de la red.

2.4. Calidad de servicio

La calidad de servicio en una red de cómputo tiene varias definiciones, lograr que todos se pongan de acuerdo en lo que constituye un buen servicio es un ejercicio nada trivial. Inicialmente, el concepto de calidad de servicio era simplemente la confianza, en el sentido, de que no se perdieran datos en una comunicación, y un ejemplo claro de ello es TCP/IP. Las redes TCP/IP fueron diseñadas para el transporte óptimo del tráfico, por lo que la calidad de servicio requerida en las mismas se basa únicamente en la integridad de los datos. Por esto, el servicio que brinda IPv4 es del tipo mejor esfuerzo[3] y sin requerimientos de tiempo real.

Muchas aplicaciones con requerimientos de tiempo real necesitan que la red de garantías de calidad de servicio (CdS), para su correcto funcionamiento. Estas garantías, se basan en que ciertos parámetros estén acotados. Las métricas que habitualmente se usan son: la latencia, la variación del retardo, el ancho de banda y la pérdida de datos.

La latencia, a veces llamada retraso de propagación, es el tiempo que tarda una trama o paquete en viajar desde la estación de origen hasta su destino final en la red. Las diferencias de latencias entre cada paquete, se llama variabilidad (jitter). Es un

[3]Cada paquete es enviado independientemente, no garantiza que los paquetes lleguen a su destino, ni que lleguen en el orden en que fueron enviados, tampoco no implementa corrección de errores ni control de congestión.

término que se refiere al nivel de variación de retado que introduce una red. Una red con variación cero tarda exactamente lo mismo en transferir cada paquete de información. Mientras que una red con variación de retardo alta tarda mucho más tiempo en entregar algunos paquetes, que en entregar otros. Las cualidades de retardo reducido y retardos uniformes son las dos condiciones que persigue la calidad de servicio.

2.4.1. Arquitecturas con calidad de servicio

Los mecanismos de señalización para CdS permiten a los distintos tipos de flujos de paquetes establecer y liberar reservas de recursos (típicamente ancho de banda) en redes IP, con la intención de proporcionar un determinado servicio. Estos mecanismos permiten establecer la negociación entre las aplicaciones y la red con el fin de obtener un canal virtual de señalización.

En las redes IP hay dos principales arquitecturas, definidas por el IETF, para soportar la calidad de servicio [10]: IntServ y DiffServ.

Servicios integrados (IntServ)

La arquitectura de Servicios Integrados (IntServ) o RFC 1633 se ocupa de administrar el ancho de banda durante la congestión de la red. Permite ampliar la arquitectura IP existente para soportar sesiones en tiempo real, manteniendo el servicio de mejor esfuerzo existente. La idea de IntServ es pretender implementar calidad de servicio realizando la clasificación del tráfico, asignación de prioridades y una reserva de recursos mediante un protocolo de señalización.

IntServ utiliza para administrar la congestión un conjunto de mecanismos de control de tráfico subyacentes (según [2]):

- Un control de admisión, ejecutado por un protocolo de reserva, es capaz de poder comprobar si es viable la petición y determinar si se dispone de recursos para ofrecer la CdS a un flujo[4]. IntServ no define ningún método de control a utilizar pero normalmente se relaciona con el RSVP (RFC2205) [2], [11].

- El mecanismo de encaminamiento, en cargado de proporcionar la información del siguiente ruteador para cada dirección destino.

[4]En términos de IntServ, el flujo es entendido como una corriente de paquetes con la dirección origen y destino, puerto origen y destino, iguales.

- El clasificador de paquetes, el cual analiza los campos de direcciones y puertos para determinar el flujo al que pertenece el paquete.
- El algoritmo de encolado gestiona la transmisión de los paquetes por un enlace de salida dentro de un ruteador.
- Una norma de descarte, que proporciona un mecanismo uniforme que indica las condiciones bajo las cuales se descartan los paquetes.

IntServ ofrecen tres tipos de servicios: garantizados, de carga controlada y del mejor esfuerzo. Los dos primeros emulan a los circuitos dedicados, garantizando los parámetros de la especificación del tráfico del emisor. El tercero suministra mejor esfuerzo que el mejor esfuerzo de IP.

La forma de garantizar estos servicios se centra en el protocolo RSVP, ya que hace posible la asignación de recursos (ancho de banda, fluctuación de fase, ráfaga máxima, etc.) a través de una red IP. RSVP representa un protocolo de señalización el cual esta dirigido a sistemas terminales y especialmente al sistema terminal receptor. La razón es que cuando un emisor transmite un mensaje llamado Path (Trayecto) para solicitar recursos a los sistemas terminales receptores. La ruta que deben seguir estos mensajes es la misma que siguen los datos de usuario; determinada por el protocolo de enrutamiento, de lo contrario para nada serviría RSVP. Cuando el receptor (o receptores) recibe el mensaje Path, esté envía el mensaje Resv (reserva) como respuesta a los mensajes PATH, y solicitan a la red (a los ruteadores RSVP) las correspondientes reservas de recursos para soportar la comunicación con cierta CdS, fluyendo hasta la fuente datos del usuario emisor. Las reservas de recursos no son permanentes y deben ser refrescadas periódicamente con mensajes PATH y RESV. A todo ello, el mensaje Resv es realmente el que realiza la solicitud de recursos a los dispositivos intermedios entre extremo a extremo.

Servicios diferenciados (DiffServ)

Los servicios diferenciados (DiffServ) o RFC 3289, es una forma de ofrecer diferentes servicios a flujos de tráfico distintos. Es una manera sencilla y tosca de clasificar los servicios de las aplicaciones. Sin embargo, es una solución escalable, más apropiada para grandes entornos como Internet. Se basa en marcar los paquetes IP y la red (los ruteadores) los tratará en base a esa marca. Define y utiliza diferentes tipos de ruteadores. Esta diferenciación depende de si se trata de un nodo interior o un nodo frontera.

El marcado del tráfico lo realizan los ruteadores de frontera, aunque también los sistemas terminales pueden realizarlo. La versión IPv6 contempla este marcado de paquetes, mediante el campo DS (campo de servio diferenciado) de la cabecera IP. IPv4 permite marcar paquetes, a través del byte ToS (Tipo de Servicio) y en tal caso se utiliza éste como byte DS [2].

El funcionamiento de DiffServ se basa en clasificar los datos a la entrada de la red con relación a un determinado servicio, después se aplica el proceso de preparación del tráfico (ver Figura 2.4). La clasificación a la entrada en la red está basada en el análisis de uno o varios campos de la cabecera del paquete. Después el paquete se marca, como perteneciente a una determinada clase de servicio. Los enrutamientos centrales sólo examinan el campo donde se marcó el paquete y le dan el tratamiento correspondiente a esa clase de servicio. Finalmente, antes de salir de la red se suprime la marca.

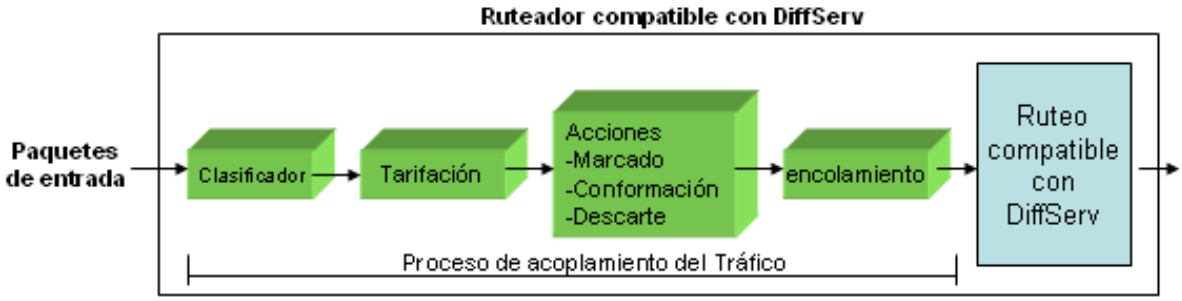

Figura 2.4: Procesos de entrada ha un nodo externo de Servicios Diferenciados

Se han definido dos tipos de servicio de DiffServ con garantía de CdS: Reenvío Rápido (RR) y Reenvío Asegurado (RA). El primero equivale a una línea arrendada virtual, por lo que se garantiza cierto ancho de banda y reducida demora de cola. RA permite que los paquetes se etiqueten con 12 posibilidades, descritos por el campo DS, pues tiene 4 clases con 3 procedimientos en cada clase que determinan como descartar tráfico. Cuando hay congestión en un ruteador los paquetes con mayor precedencia son descartados primero. Además, la utilización de RA asume la existencia de un acuerdo entre el usuario y la red, en el nivel de servicio (NdS). El NdS define el perfil del tráfico (ancho de banda, retardo, jitter y tasa de pérdidas) y la política (tiempo de disponibilidad, penalizaciones, etc.). Una vez se establece el NdS, se espera que el tráfico enviado por el cliente sea conformado y marcado en la entrada en la red de acuerdo con lo acordado en el NdS y cualquier tráfico no conforme no tendrá calidad de servicio.

Tabla 2.3: Tabla comparativa entre las dos principales arquitecturas de QoS.

Servicios Integrados (IntServ)	Servicios Diferenciados (DiffServ)
Son aplicables en redes pequeñas.	Presenta un buen desempeño tanto en redes pequeñas como grandes.
Funciona en el nivel 4 del modelo OSI.	Trabaja en el nivel 3 del modelo OSI, el cual, lo hace transparente para el usuario.
Deja que los usuarios puedan realizar explícitamente peticiones de CdS.	Se establece un acuerdo previo entre el usurario y red, para el manejo de CdS.
Permite solicitudes de calidad de servicio con gran granularidad.	Tiene solo 12 posibilidades de servicios.
Necesitan periódicamente refrendar el tipo de servicio.	Los tipos de servicio son permanentes.
Utilizan un protocolo de reserva para designar recursos.	Los recursos son asignados en el ruteador de frontera.
Emulan conmutación de circuitos.	Los nodos internos procesan los paquetes de acuerdo al campo DS.
Posee un mecanismo más complejo y exigente.	Tiene una forma sencilla de clasificar y priorizar el tráfico.

El comparativo entre IntServ y DiffServ

Si bien, para garantizar la calidad de servicio en una red IP se puede utilizar ya sea IntServ o DiffServ. Se hace evidente entonces, la necesidad de tener encuentra los atributos y debilidades de cada uno de estos protocolos (ver la Tabla 2.3), con la intención de emplearlos en los contextos correctos.

La descripción de estos dos protocolos de CdS podría hacer pensar que son excluyentes, pero no es así, de hecho se complementan. En la práctica, es muy frecuente encontrar muchas posibles combinaciones entre estas dos arquitecturas y más aún, pueden combinarse con otras tecnologías para dar soporte a la CdS extremo a extremo.

Aparte de estos dos protocolos, también existe la conmutación de etiquetas multiprotocolo (MPLS) que es similar a DiffServ en algunos aspectos, dentro de sus múltiples funcionalidades, realiza la ingeniería de tráfico, el control del ancho de banda y la priorización de aplicaciones [11], [12].

2.4.2. Mecanismos de CdS

Uno de los problemas más frecuentes que se presenta en las interredes es la diferencia entre la velocidad con que un sistema de origen transmite paquetes y la velocidad con

la que destino puede aceptarlos. La solución común para esto, es introducir un control de flujo para inspeccionar la velocidad de transmisor de modo que no envíe a mayor velocidad que la que puede manejar el receptor.

De igual modo, el sistema de control de congestión se ocupa de una situación similar. Esto sucede cuando los paquetes llegan a un nodo de la red con mayor rapidez que aquélla con que pueden procesar y reenviar. El nodo se congestionará, debido a que el buffer crece y aumenta el retardo efectivo. El control de congestión en realidad es un mecanismo de gestión de recursos y de tráfico, para evitar y prevenir situaciones límite (desbordamiento de búferes o ancho de banda insuficiente) que puedan provocar que se colapse la red.

Si la red se congestiona, el sistema de control de flujo regularía el paso de paquetes hacia la red, ayudando a controlar la congestión. Sin embargo, no la evita del todo. Por tanto, para este problema existen dos soluciones, llamadas pasiva y activa, que permiten el control y prevención.

Una solución pasiva se considera en un diseño adecuado de la red. Existen múltiples variables con las que el diseñador puede manejar a la hora de planear la red, estas variables influirán en el comportamiento frente a la congestión. La solución trata de lograr su meta usando políticas adecuadas en varios niveles del modelo OSI, como lo señala.

Las soluciones activas controlan la congestión de manera dinámica, pues se ejecutan solo cuando se detecta el problema. Estas soluciones tienen tres fases [13]: el monitoreo de parámetros, reacción y ajuste del sistema.

2.4.3. Herramientas de CdS

Los distintos tipos de herramientas que se disponen para asegurar una CdS dentro de una red IP, previenen o manejan una congestión, distribuyen el tráfico e incrementan la eficiencia de la red.

Colas de control de congestión

El tratamiento de la congestión se fundamenta en el manejo de las colas en buffer mediante diferentes técnicas. Actualmente hay cuatro algoritmos genéricos utilizados en los sistemas intermedios [14]:

Cola FIFO. Los paquetes se reenvían a la salida en el orden en qué arribaron. Este es el mecanismo de CdS por omisión en las redes IP. Es válido solo en redes con mínima congestión.
Cola con Prioridad. Este mecanismo de control de congestión se basa en la prioridad de tráfico de varios niveles. En este caso, se puede dirigir el tráfico en N colas de prioridades descendentes. El trafico de la cola de mayor prioridad es atendida antes que el tráfico de las colas de menor prioridad.
Cola Personalizada. Este mecanismo se basa en garantizar el ancho de banda mediante una cola de espera programada. El operador reserva un espacio de buffer y configura una asignación temporal a cada tipo de servicio. Esta cola puede impedir la inanición potencial de las colas de baja prioridad.
Cola ecuánime ponderada (WFQ). Este mecanismo asigna una ponderación a cada flujo de forma que determina el orden de tránsito en la cola de paquetes. La ponderación se realiza mediante discriminadores disponibles en TCP/IP (dirección de origen y destino y tipo de protocolo en IP, número de Socket -puerto de TCP/UDP-) y por el ToS en el protocolo IP.

Control de tráfico

Una manera secundaria de controlar la congestión es utilizar un control de flujo. Dos de estas herramientas son el Formador de Tráfico (o mejor conocido en inglés como "Traffic Shaper") y la Pronta Detección Aleatoria Ponderada.

Traffic Shaping. Es el término genérico que se aplica para dar una amplia variedad de técnicas diseñadas para cumplir políticas de asignación de prioridades junto a la razón de transmisión de datos, sobre un sistema terminal de la red. Suelen utilizarse para controlar el flujo del tráfico en una interfaz de red, fijando una velocidad de transmisión entre el operador y el usuario. El control de flujo se logra asignando al tráfico una velocidad binaria particular, que consiste en enviar al tráfico a una cola de espera. De este modo, cualquier tráfico por encima del ancho de banda establecido es encolado para su salida posterior. Esto eliminara cuellos de botella en topologías con desajustes de velocidad de transferencia de datos [14], [15].

Pronta Detección Aleatoria Ponderada. Trabaja monitoreando la carga de tráfico en algunas partes de las redes y descarta paquetes en forma aleatoria si la congestión au-

menta. Está diseñada para aplicaciones TCP debido a la posibilidad de retransmisión. La pérdida de datos en la red obliga ha TCP a un control de flujo, que reduzca e incremente el tamaño de la ventana de forma paulatina. No realizar un descarte sistemático de paquetes, ya que los mecanismos de retransmisión en TCP/IP nos llevarían a situaciones de mayor congestión. Además, favorece las clases de mayor prioridad en situaciones de carga en la red.

2.5. Resumen

En este capítulo se destacan principalmente tres tópicos, el ruteo, la distribución de tráfico y la calidad de servicio.

En un ruteador, el ruteo permite evaluar las trayectorias disponibles hacia un destino, seleccionando la mejor ruta en todos los casos, y establece el manejo de un paquete. Asimismo, el ruteador selecciona qué ruta debe utilizar (distribución de tráfico), buscando en la tabla de enrutamiento IP, para enviar los paquetes desde la red origen a la red destino. Por supuesto, este tipo de distribución de tráfico es permanente, ya que utiliza siempre la dirección IP destino para orientar un paquete. Finalmente, la CdS se define como la capacidad de la red para proporcionar un mejor servicio al tráfico seleccionado en base a políticas y funciones de administración para el control de tráfico.

Capítulo 3

Distribución y control de tráfico sobre redes de cómputo

En este capítulo se mencionan los diferentes sistemas y mecanismos que realizan la distribución y control de tráfico dentro de una red de cómputo. Para lograr un servicio diferenciado entre clases de tráficos, los distintos fabricantes y diseñadores de hardware como de software admiten varias técnicas que propicien esta funcionalidad. Algunas de estas técnicas son estándares, mientras que otras lo serán en un futuro. Se completa el capítulo abundando específicamente en las cualidades que pose el núcleo de Linux para la distribución y control de tráfico.

3.1. Redes de cómputo con CdS

En una red de conmutación de paquetes se comparten las conexiones entre varios nodos, optimizando el uso de la línea, ya que se asegura que no hay ningún tiempo muerto en la conexión. Aunque se incrementa el nivel de eficiencia en la red, introduce varios problemas. Por ejemplo, cuando intenta transmitir datos en tiempo real.

Una de los casos que hace difícil transportar datos en tiempo real a través de una red de paquetes, se debe a que son atendidos en forma FIFO. Junto a esto también está el caso cuando un nodo puede verse desbordado y obligado a descartar paquetes. Debido a estas situaciones, la red no garantiza que los paquetes vayan a llegar con un retardo mínimo o con una variabilidad uniforme. Para ello se desarrollaron varias arquitecturas (IntServ, DiffServ y MPLS) para proporcionar la funcionalidad de CdS.

Otro problema importante, habla de la necesidad de la interconexión de una red de conmutación de paquetes con las redes de área local (RAL). La arquitectura RAL más común es Ethernet, basada en CSMA/CD, que es un método de acceso en un medio

compartido donde solo se permite que haya una estación transmitiendo al mismo tiempo sobre una base de igualdad. Por lo que la CdS entre los usuarios finales no es la adecuada, en los segmentos Ethernet, aun pesar de que las redes de conmutación de paquetes son posibles implementar la CdS extremo a extremo. Sin embargo, se han propuesto diversas soluciones a éste problema, entre las que se encuadran:

1. Ethernet full-duplex permite una transmisión y recepción simultáneas, que requieren el uso de dos pares de cables y una conexión conmutada entre cada nodo. Tiene el inconveniente de que todos los elementos (adaptadores y concentradores) deben ser full-duplex.

2. IsoEthernet (Isochronous Ethernet) es una variación de la Ethernet para soportar tráfico en tiempo real. Una propuesta de la IEEE (802.9) con la idea de soportar en el mismo cable de par trenzado no blindado la transmisión de datos de una RAL con varios canales isócronos tipo B de la Red Digital de Servicios Integrados (RDSI). Esta solución no se ha sido tenido en cuenta, debido a que requiere nuevos adaptadores de red y conmutadores [16].

3. Rether (Real Time Ethernet) es una técnica que está basada en el paso de un testigo similar a la de Token Ring y permite dotar de CdS a Ethernet [17]. Rether puede comportarse de manera dual, ya se como una estación normal de Ethernet, o en el modo Rether cuando sea el caso de transmitir sin colisiones en base a demanda. Esta técnica añade complejidad y retardos al sistema.

4. Dos de los conmutadores más utilizados para resolver el problema de CdS en una RAL son: el conmutador que utiliza el estándar IEEE 802.1p, que es una extensión del 802.1D del funcionamiento de los puentes, y él que utiliza tecnología PACE de 3Com [17].

Todo lo anterior se abrevia diciendo, que para garantizar la CdS extremo a extremo se necesita de la participación conjunta de tres elementos generales [14], [18]: las aplicaciones de tiempo real, el acceso a las RAL's y el acceso a las interredes.

Sin embargo, las aplicaciones actuales de red no son de tiempo real, pero sí exigen un ancho de banda garantizado, tiempos de respuesta mínimos, además de que algunas se pueden considerar de misión crítica. Estas aplicaciones necesitan de redes que cumplan ciertas exigencias, para que no sean tratadas de igual forma, como sucede en las redes tradicionales.

Una actividad derivada del acceso a las RAL's, de la CdS de extremo a extremo, es empleada por las aplicaciones actuales, que necesitan de un sistema no tan estricto como uno de tiempo real, pero que permita efectuar un control de tráfico en las fronteras de las redes. Asimismo, la existencia probable de varias conexiones de redes externas que posee una RAL, hace notar la necesidad de la distribución del tráfico con base en criterios más allá de los acostumbrados. A todo esto se le conoce como un sistema de administración de recursos de red, que se describirá a continuación.

3.1.1. Administración de recursos de una RAL

Dentro de una LAN que posee un sistema de administración de recursos de red es posible priorizar el flujo de tráfico, reservar el ancho de banda para diferentes aplicaciones, activar la seguridad sobre las aplicaciones y los usuarios que ingresan a la red, y sincronizar las necesidades de la organización con un comportamiento deseado de red. A esto, se suma de forma independiente la capacidad de distribuir el tráfico con base en características de los paquetes, cuando se tiene dos o más posibles alternativas de salida (conexiones de red).

La manera de administrar recursos de una red depende del tipo de topología, protocolos, mecanismos de planificación, control de tráfico y control de admisión, que se emplean. Se describe a continuación cómo se realiza de forma general el control de tráfico en una red. Así mismo, también se detalla la forma de cómo un paquete se le asigna una de varias interfaces de red para su reenvío. La Figura 3.1 muestra la estructura interna de un punto de frontera que interconecta una RAL con redes externas, donde se asocia y contrasta las actividades de distribución y control de tráfico [19].

La Figura 3.1 señala como el tráfico proveniente de la RAL tiene como primera instancia la entrada al bloque de distribución de tráfico. En ella se realizan generalmente dos acciones: la identificación del tráfico y la ejecución de las políticas de enrutamiento.

El proceso de identificación del tráfico da la posibilidad de poder ordenar todo los paquetes en clase. La forma de poder identificar un paquete con respeto a otra, va desde, la comparación entre los encabezados de los paquetes, hasta el análisis del contenido de información de los datos.

La ejecución de las políticas de enrutamiento no es más que el emparejamiento de reglas con los tráficos ya clasificados, que toman como acción determinar la red salida que debe tomar un paquete.

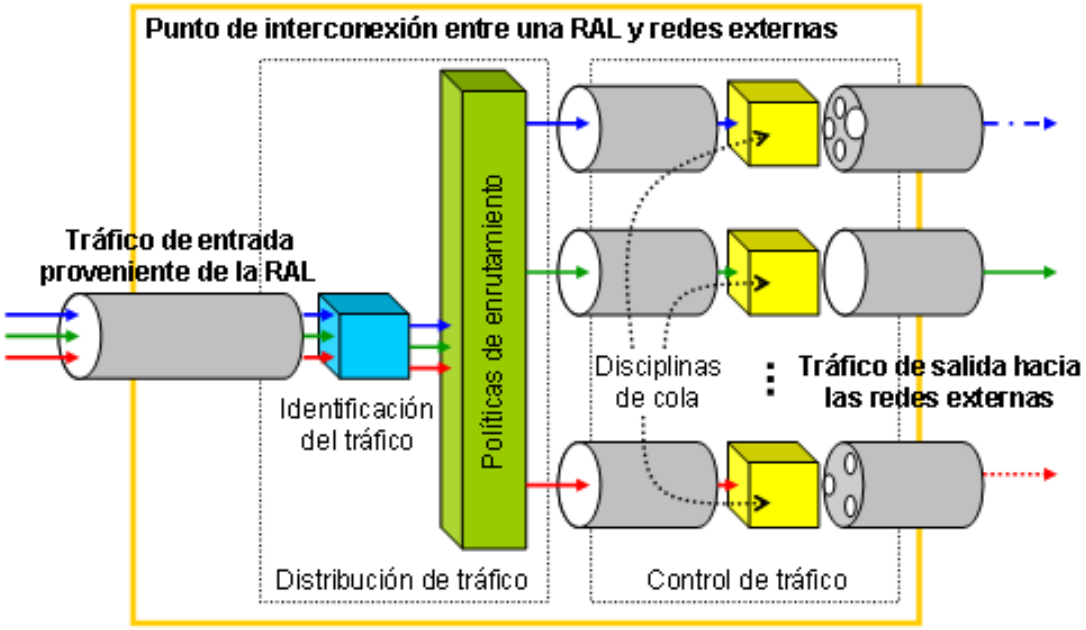

Figura 3.1: Descripción general de un sistema de distribución y control de tráfico.

El bloque de control de tráfico tiene lugar en las colas de los dispositivos de red de salida. Para el control de tráfico, e implícitamente la gestión de ancho de banda, se usan dos conceptos: filtros y colas. Los filtros ponen los paquetes en las colas y las colas deciden qué paquetes mandar antes que otros, con la ayuda de las disciplinas de colas. La disciplina de colas (Qdisc) es el algoritmo que gestiona el proceso de encolar y desencolar paquetes en un dispositivo de red.

En las siguientes secciones se describen herramientas en la que no todos son propiamente sistemas de administración de recursos de red, pero si tienen muchas de las operaciones que ellos realizan. Específicamente se harán mención de aquellos sistemas, ya sea por hardware o software, que proveen mecanismos de distribución de tráfico y calidad de servicio.

3.2. Distribución y control de tráfico por hardware

En la actualidad los dispositivos que realizan la distribución de tráfico para las diferentes redes de comunicación de mayor complejidad y disponibilidad casi absoluta, son con aquellos equipos de hardware. Estos tienen un buen desempeño en sus tareas, pues son cada vez más especializados, además que actualmente son capaces de ofrecer calidad de servicio.

3.2.1. Ruteadores

Los ruteadores son los responsable directos de la distribución del tráfico entre redes de cómputo. Ello hace que estos dispositivos tengan la necesidad de cumplir un cierto conjunto de reglas o estándares para proporcionar un servicio confiable. Se puede decir que el ruteador tiene dos enfoques de distribución de tráfico en base a los conceptos de ruteo clásico y ruteo de tiempos críticos.

El ruteo clásico se encarga principalmente de determinar una ruta, mediante la evaluación de las diferentes rutas disponibles al destino, y decidir el camino que toma un paquete en base a la consulta de la tabla de ruteo.

El ruteo de tiempos críticos no es más que una anexión de mecanismos al ruteo clásico. Por ejemplo, cuando un ruteador presenta una situación de sobrecarga de trabajo, éste añade un retardo significativo al tráfico de mayor prioridad. Siendo esto, el motivo por el que se incorporan varias técnicas para acelerar el flujo de paquetes. Entre las que destacan:

- La utilización de diversas colas para poder diferenciar tráficos.
- La compresión de la cabecera de TCP, tal como se realiza con la cabecera del Protocolo de Transporte de Tiempo Real (RTP) [20].
- El marcado o la definición de clases de tráfico para indicar el tipo de servicio preferido de enrutamiento.
- Algoritmos que reservan de manera dinámica el ancho de banda de la red.

Cabe destacar que los ruteadores basados en circuitos integrados específicos de aplicación (ASIC) [21], permiten tener operaciones de consulta de tablas de ruteo relativamente rápidas, haciendo ver a las técnicas de aceleración de tráfico innecesarias. La ventaja indudable de utilizar estas técnicas radica en el poder distribuir a los paquetes en función a clases de tráficos que normalmente no manejan las tablas de ruteo, sin mencionar que también se logra el control de las clases.

En el ámbito de la calidad de servicio, los ruteadores permiten construir redes en base al modelo de Servicios Integrados o el modelo de Servicios Diferenciados. Al igual que, también proporcionan la funcionalidad de Reconocimiento de Aplicaciones en Red para clasificar el tráfico en función de la aplicación, Protocolo de Reserva de Recursos que permite garantizar el ancho de banda para una aplicación, y gestión de normas y políticas para CdS.

3.3. Distribución y control de tráfico por software

Es claro, que los dispositivos de distribución de tráfico basados en hardware realizan el trabajo de manera eficiente, pero también, es claro que su costo es elevado. Por ello, los sistemas intermedios por software son una alternativa, ya que están basados en un equipo de cómputo dedicado, con un sistema operativo de multiprogramación, una tarjeta multipuerto, software con capacidad multiprotocolo y software de administración de red. Estos trabajan de manera conjunta con los conmutadores, los puentes e incluso con ruteadores.

Los sistemas de administración de recursos de red son totalmente independientes a la labor de ruteo y distribución de tráfico, ya que estos últimos dependen completamente de las características que posee el sistema operativo para el manejo de redes de cómputo. En cuanto a los sistemas de administración de recursos, existen una gran variedad de productos de software, entre los más divulgados se encuentran: el Solaris Bandwidth Manager y FloodGate-1. Además, existen herramientas que facilitan el control de tráfico en un sistema operativo, tales como: ALTQ e IPROUTE2.

3.3.1. Solaris con Solaris Bandwidth Manager

El sistema operativo Solaris versión 9 es una plataforma basada en UNIX, pesada para procesadores SPARC y para procesadores x86. Específicamente en el terreno de enrutamiento, este sistema operativo provee un sistema de ruteo y de distribución de tráfico tradicional. Sin embargo, Solaris permite utilizar simultáneamente dos o más caminos en operaciones de E/S (multipathing) hacia dispositivos de almacenamiento y hacia las redes de cómputo, mediante el uso de Sun StorEdge Traffic Manager y el IP Multipathing. Por lo tanto, si fallase un adaptador de la red el sistema cambia todos los accesos de red automáticamente al adaptador alterno.

Lo que nos hace referir a este sistema operativo es la capacidad de gestión de la calidad de servicio tanto para el sistema como para la red de cómputo. Solaris provee una herramienta de administración de red llamada Solaris Bandwidth Manager (SBM).

El Solaris Bandwidth Manager es un software que controla y asigna ancho de banda, a aplicaciones y sistemas que hacen uso de una o más líneas compartidas para conectarse a una o más redes. De este modo, los administradores de red pueden realizar de la forma más acertada posible sus tareas de gestión y control de tráfico. Este sistema de administración resuelve tareas tan variadas de CdS a partir de: control de políticas de la organización, mecanismos de clasificación de tráfico, manejo de todo el tráfico IP, priorización de tráfico de entrada y de salida para reducir congestión, proveer diferentes

clases de servicios, monitoreo de los flujos de paquetes y adopta arquitecturas de CdS.

El clasificador de paquetes que emplea SBM utiliza un conjunto de reglas para poder filtrar y adjudicar cada paquete a una clase. La clase de un paquete es definida por uno o varios de los factores siguientes: dirección IP destino y fuente, puerto fuente y destino de TCP o UDP, protocolos TCP o UDP, URL o grupos de URL's y valor del campo ToS.

3.3.2. FloodGate-1

El FloodGate-1 es literalmente un sistema de administración de recursos de red enfocado a solucionar la gestión de ancho de banda, basado en políticas, para redes privadas virtuales, WANs privadas y conexiones a Internet. Éste no contempla características de distribución de tráfico, pues lo deja en manos del sistema operativo donde reside (que pueden ser: Windows Server, Solaris, Red Hat, Nokia IPSO y Check Point SecurePlatform).

FloodGate-1 es un administrador de red centralizado y de alta disponibilidad. Basado en la tecnología patentada por la empresa Check Point para inspección de tráfico, llamada Stateful Inspection, que realizar una captura detallada de la información de todo el tráfico en la red, la cual es almacenada y actualizada dinámicamente.

El funcionamiento de FloodGate-1 realiza los siguientes pasos: inspecciona, clasifica, encola y programa el tráfico antes de ser entregado al canal. La clasificación interpreta la información y la ordena de acuerdo a más de 150 servicios y aplicaciones. Luego el tráfico es programado para su envío basado en las políticas de CdS. El mecanismo de desenconamiento emplea un algoritmo WFQ para realizar un control preciso de asignación, de acuerdo al ancho de banda.

FloodGate-1 permite clasificar tráfico en un amplio juego de criterios, tales como: dirección IP destino y fuente, puerto fuente y destino de TCP o UDP, servicios de Internet y aplicaciones, usuarios, URL o grupos de URL's y por tiempo.

3.3.3. BSD con ALTQ

El BSD es un sistema operativo de tipo UNIX, basado en 4.4BSD, multiusuario, disponible para múltiples plataformas. Este sistema permite filtrar trafico TCP/IP, traducir direcciones de red (NAT) y proporciona mecanismos de CdS.

La manera como BSD realiza el enrutamiento, no es más que la forma estándar sustentada por las tablas de enrutamiento y la dirección IP destino. Las opciones del núcleo

en BSD pueden ser configuradas para trabajar como un sistema de ruteo avanzado [22], activando características para acelerar el trabajo, y aumentar la eficiencia del mismo.

Normalmente, se usan las herramientas `pf` (filtrar paquetes) y `altq` (encolamiento alternativo) para las implementaciones de cortafuegos y calidad de servicio. La función de `pf` es bloquear o permitir el paso a los paquetes de datos de forma selectiva, según van llegando a una interfaz de red. Por su parte, la herramienta `altq` permite realizar funciones de CdS, que incluyen funciones de asignación de ancho de banda, manejo de prioridad de paquetes y monitoreo del tráfico de una manera más sencilla.

Por definición, BSD usa un planificador tipo FIFO en las colas de los dispositivos de red, sin embargo, también tiene soporte para dos planificadores adicionales, las Colas Basadas en Clases y las Colas Basadas en Prioridades. Para evitar la congestión, BSD utiliza la Pronta Detección Aleatoria que es un algoritmo que asegura de que la cola no se llene.

3.3.4. Linux con Iproute2

Linux es un sistema operativo multiusuarios, multitareas, tipo UNIX, la cual, se ejecuta en una amplia variedad de plataformas. El trabajo con redes es un área de funcionalidad clave para Linux. A partir de la versión 2.4 del núcleo Linux, está disponible un socket llamado NETLINK que permite implementar en espacio usuario código de gestión de paquetes y tráfico IP [23].

Linux es un sistema operativo en rápida evolución, y presenta características muy aceptables en el panorama de sistemas de ruteo dedicados y para ofrecer CdS. En particular, lo destacable es la distribución de datos cuando existen conexiones simultáneas con redes de diversa naturaleza.

El núcleo de Linux nos permite hacer cosas como túneles IP, utilizar múltiples tablas de ruteo, reservar de ancho de banda, multidifusión, Proxy ARP, control de tráfico, entre otras cosas. El acceso a los usuarios a estas características avanzadas de red es proporcionado por el paquete de software Iproute2, que contiene las herramientas `ip` y `tc`.

La primera herramienta del paquete Iproute2 es `ip`, la cual se encarga de gestionar las tablas ARP, de hacer uso de las múltiples tablas de ruteo para una distribución avanzada del tráfico, la creación de túneles IP y del monitoreo de las rutas. La segunda herramienta llamada `tc` (traffic control) es la que permite implementar toda la gestión de tráfico en cuanto control y manipulación del ancho de banda y prioridad.

Tabla 3.1: Puntos comparativos entre los diferentes sistemas de administración.

Comparativas	SBM	FloodGate-1	ALTQ	IPROUTE2
Sistema operativo	Solaris	SO más conocidos	BSD	Linux
¿Es un sistema de administración de recursos?	Si	Si	No	No
¿Es un software comercial?	Si	Si	No	No
Sistema de identificación de tráfico	Muy bueno	Muy bueno	bueno	bueno
¿Es un sistema centralizado?	No	Si	No	No
Contempla la distribución de Tráfico entre redes	No	No	No	Si
Software adicional en las estaciones terminales	No	Si	No	No
Tipo de interfase con el usuario	Gráfica	Gráfica	Comando	Comando

3.3.5. Comparaciones entre el SBM, FG-1, ALTQ e IPROUTE2

Las herramientas que procuran la administración de recursos de la red mencionadas anteriormente tienen puntos coyunturales que originan una discusión mostrada en la Tabla 3.1.

El SBM y FloodGate-1 son productos comerciales, que entre sus ventajas más importantes es la identificación de flujos de tráfico sofisticados. El SBM puede identificar tipos de clases en base grupos de URL's. Por su parte, FloodGate-1 identifica servicios de Internet y aplicaciones, al igual que asigna anchos de banda a usuarios sin importar la estación terminal en donde estén ubicados. Estos dos sistemas pueden implantar las arquitecturas de CdS, además que su configuración y manejo se realiza en una interfase grafica.

En cuanto a las herramientas altq e IPROUTE2 son productos de uso libre y tiene un potencial muy bueno para la distribución y el control de tráfico, pues cuentan con una gran cantidad de mecanismos y técnicas muy variadas.

Las plataformas BSD y Linux son una alternativa, segura y funcional que proporciona enrutamiento múltiple, tareas de cortafuego y asignación de ancho de banda. Destaca el hecho de que, el enrutamiento múltiple de BSD no maneja la filosofía de múltiples tablas de enrutamiento como en Linux. Además, para la asignación de ancho de banda la herramienta `altq` esta muy limitado en comparación de la herramienta `tc` de Linux.

Resulta que el modulo de red de Linux (Net-4) es uno de los mejores, entre los sistemas operativos que permite la distribución de tráfico y la CdS [5]. En las siguientes secciones se describen las características avanzadas de enrutamiento y control de tráfico en Linux.

3.4. Distribución de tráfico en Linux

El núcleo de Linux nos permite trabajar con múltiples tablas de ruteo. Habitualmente, los ruteadores tienen por cada sistema, una tabla de enrutamiento única en la que almacena información sobre el mejor camino que pueden seguir los paquetes para llegar a su destino. Gracias al paquete Iproute2 con su herramienta `ip` es posible hacer uso de varias tablas de enrutamiento a la vez, forma en la que se puede decidir qué tabla es utilizada por un determinado paquete, según las características de su encabezado [24].

La base de datos de políticas de enrutamiento (BDPE) se encarga de seleccionar la ruta apropiada de un paquete verificando un conjunto de reglas. En el que cada regla se define las características de una clase de tráfico y su relación con una tabla de ruteo. Además, estas poseen una prioridad, que indica el orden en el que se examinan de forma secuencial (entre 0 a 32767).

Las reglas de BDPE se emparejan con los paquetes únicamente por la dirección IP de fuente y destino, el campo ToS, interfaz de entrada del tráfico y por el valor marcado del paquete (`fwmark`). Por su parte, las tablas de ruteo tienen un funcionamiento de forma normal, pues siguen distribuyen a los paquetes por su dirección IP destino.

Cada regla de BDPE esta compuesta de un seleccionador y una acción, como se muestra en la Figura 3.2. El núcleo de Linux examina el seleccionador de cada regla, en caso que el seleccionador coincide (empareje) con un paquete, la acción se realizará validando la tabla de ruteo relacionada con la regla. El objeto de la acción consiste en encontrar una ruta en la tabla de ruteo que haga juego con el paquete, en función de su dirección destino. Si la acción devuelve éxito, entonces la salida de la regla proporcionará una ruta válida, y la consulta de BDPE es terminada. Si no es así, se procede a seguir examinando las reglas hasta que se encuentre una ruta apropiada para el paquete o se terminen las reglas.

Linux define inicialmente tres tablas de ruteo, llamadas `local`, `main`, y la de omisión (identificada por el número 253); cada una con una prioridad distinta y se aplican a todos los tráficos. La tabla `main` tiene el cometido de realizar el enrutamiento del sistema cuando no se ha definido ninguna regla de BDPE. La tabla de ruteo `local` es mantenida

```
[root@centauri root]# ip rule add from 192.168.140.1 table Salida_con_firewall
[root@centauri root]# ip rule ls
0:      from all lookup local
32764:  from 192.168.140.1 lookup Salida_con_firewall
32765:  from 148.247.102.43 lookup Salida_sin_firewall
32766:  from all lookup main
32767:  from all lookup 253
[root@centauri root]#
```

Figura 3.2: Lista de reglas de la BDPE, que describe prioridad, el selector y la tabla a usar.

por el núcleo, y es de uso exclusivo del sistema. Todas las demás tablas, incluyendo la de omisión, son gestionadas por el software de red o él administrador, para definir tablas de ruteo alternativas para los diversos tráficos.

3.4.1. Sistema de balanceo de carga

El balanceo de carga en Linux efectúa una distribución equitativa del tráfico de salida a los puntos de enlace, gracias a la incorporación de "equal cost multi path"(RFC 2992). Sin embargo, para tener un mejor desempeño es necesario incorporar el parche provisto por Julian Anastasov [19]. Las innovaciones de este parche radican en un proceso de balanceo dinámico, más confiable y equitativo.

El funcionamiento del balanceo de carga describe, que en lugar tomar una sola salida de red, es posible configurar al núcleo de Linux para proporcionar varias salidas hacia un mismo lugar, para que tengan un funcionamiento conjunto. El objetivo principal es sumar sus anchos de banda. Algo muy importante que hay que tener en mente, el balanceo de carga significa tener la posibilidad de tomar otros caminos alternativos, pero no significa que, una conexión se divida entre el número de caminos disponibles. Es decir, toda la comunicación de una conexión se hará por el camino elegido inicialmente, aun a pesar que este sea el único camino congestionado.

3.5. Control de tráfico en Linux

Linux tiene un sistema sofisticado para el manejo de ancho de banda y prioridad, llamado Control de Trafico. Destaca, el hecho de que casi la totalidad de los mecanismos que existen en Linux para el control tráfico estan basados en la manipulación de colas. Por esta razón, antes de explicar la arquitectura que provee Linux para la Calidad de Servicio, es necesario tener en cuenta las disciplinas de colas (Qdisc) y sus filtros, que pueden emplea tanto para el tráfico de entrada como el de salida [23], [24], [25].

Tabla 3.2: Colas disponibles en Linux.

Disciplina de colas con clase	Disciplina de colas sin clase
Pfifo fast	Qdisc PRIO
Token Bucket Filter	Class Based Queueing (CBQ)
Stochastic Fairness Queueing	Hierarchical Token Bucket (HTB)

3.5.1. Disciplinas de colas

Existen dos tipos de disciplinas de colas en Linux, con clase y sin clase. La Tabla 3.2 expone las disciplinas de colas con las que cuenta el núcleo 2.4.20 de Linux, e indica a que tipo pertenecen.

Disciplina de colas sin clases: Es aquella disciplina de colas que no admite una subdivisión interna que pueda ser configurada por el usuario. Además, cuando éste acepte datos solo se limitará a reordenarlos, retrasarlos, o descartarlos. Esta disciplina es normalmente usada para ajustar el tráfico de una interfaz red entera.

Pfifo_fast
La disciplina de cola Pfifo_fast está formada por tres bandas (bandas 0, 1 y 2), como se muestra en la Figura 3.3. Dentro de cada banda los paquetes son procesados siguiendo una política FIFO. Sin embargo, ningún paquete de la banda 1 es enviado mientras existan paquetes por enviar en la banda 0, y lo mismo ocurre para las bandas 2. Es decir, existe una prioridad definida entre dichas bandas, siendo la banda 0 la más prioritaria y la banda 2 la menor. Para determinar cuales paquetes van a cada banda se utiliza el campo ToS de la cabecera IP del mismo.

Token Bucket Filter
El Token Bucket Filter es una disciplina de cola sencilla que se limita a dejar pasar paquetes que lleguen a una tasa que no exceda una impuesta administrativamente, pero con la posibilidad de permitir ráfagas cortas que excedan dicha tasa. Este tipo de disciplina de colas es la que debe escoger en el caso de que la única necesidad sea limitar el ancho de banda de un determinado interfaz.

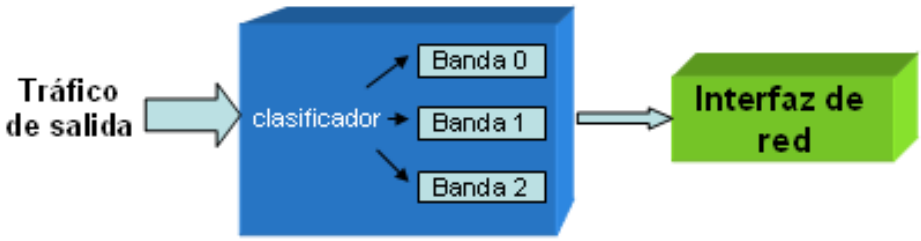

Figura 3.3: Cada interfaz de red asociada a Linux tiene una disciplina de cola, que por, omisión es la disciplina de cola Pfifo_fast.

Stochastic Fairness Queueing (SFQ)
Este tipo de disciplina de cola intenta distribuir el ancho de banda de un determinado interfaz de red de la forma más justa posible. Para ello, está disciplina implementa una política de Round Robin entre todos y cada uno de los flujos de comunicación establecidos en la interfaz. El tráfico se divide en un número bastante grande de colas FIFO por cada una de las conversaciones, dando a cada una la oportunidad de enviar sus paquetes por turnos. De esta forma, lo que se consigue es que ninguna conversación impida al resto poder enviar parte de su información.

Disciplina de colas con clases: Esta disciplina tiene n-cantidad de clases internas. Donde, cada clase contiene una nueva disciplina de cola, que puede ser con, o sin, clases. Las disciplinas de colas con clases necesitan determinar a qué clase envían cada paquete que llega. Esto lo hacen utilizando un clasificador, que a su vez hace uso de los filtros, los cuales comprueban una serie de condiciones que deben cumplir los paquetes para que sea enviado a una determinada clase.

Qdisc PRIO
La Qdisc PRIO es casi una copia de la cola Pfifo_fast. Sin embargo, la gran diferencia radica en dos factores:

1. Las tres clases asociadas a la disciplina PRIO tienen por omisión una disciplina con política FIFO. Estas pueden ser cambiadas por la disciplina de cola que se quiera.

2. Está disciplina puede hacer uso de filtros. De forma, que es posible hacer una clasificación todo lo compleja que se desee para repartir los paquetes entre las clases.

Class Based Queueing (CBQ)
La disciplina CBQ permite dividir el ancho de banda de una interfaz de red entre diversas clases. El ancho de banda se asigna de acuerdo a su nivel de prioridad, aplicando el algoritmo Round Robin por pesos entre las clases, de manera que siempre se comienza con la de menor prioridad. Su enfoque jerárquico permite configuraciones variadas de estructuras de clases, garantizando la posibilidad de dividir y asignar de diversas formas el ancho de banda.

Un punto importante en el CBQ es la obtención del ancho de banda, que se basa en asegurar de que el enlace está ocioso sólo el tiempo necesario para regular el ancho de banda de acuerdo a lo configurado. Ello, se ha comprobado que no siempre se consiguen buenas aproximaciones, e incluso a veces los resultados son totalmente equívocos [26]. No obstante, para la mayoría de las situaciones, este algoritmo trabaja de forma adecuada cumpliendo perfectamente con las necesidades.

Hierarchical Token Bucket (HTB)
HTB tiene una funcionalidad igual a la de CBQ, aunque su implementación es completamente distinta y su configuración mucho más sencilla. Debido, ha que no reconoce cálculos de tiempo ocioso para el ajuste de velocidad del tráfico. En su lugar, es un Token Bucket Filter con clase, la cual, usa pocos parámetros.

El problema de HTB radica en que apenas en la versión 2.4.20 forma parte del núcleo estándar de Linux, y las aplicaciones no la consideran todavía, por ello es necesario parcharlas y recompilarlas para utilizar HTB. En el [27] se describe el funcionamiento y maneras de uso para diferentes situaciones.

Puntos destacables de las Qdisc
Las diferentes Qdisc mencionadas anteriormente no tienen puntos de comparación, salvo CBQ y HTB, pues cada una tiene una función especifica enfocada a solucionar un problema. Entre las disciplinas de cola sin clase se destaca la Stochastic Fairness Queueing. Esta disciplina sólo tiene sentido en aquellas interfaces de red o en colas de clase que normalmente estén saturadas, y en las que no se quiera que una determinada comunicación bloque a otras.

En este trabajo de tesis se utilizan disciplinas de colas con clases con el algoritmo HTB en vez de CBQ, ya que este último es complicado de utilizar. Esto obedece básicamente al hecho de que, CBQ tiene una forma dificultosa de regular el ancho de banda. Sin embargo, no es por una mala implementación, sino sólo que el algoritmo CBQ no

es tan preciso y realmente no se ajusta del todo a la manera de trabajar de Linux. En tanto que, HTB tiene mejor comportamiento y desempeño, y es más fácil de usar.

3.5.2. Filtros para clasificar paquetes

Hay varios tipos de filtros distintos para ser usados en una disciplina de cola con clase en Linux [24], estos son: `u32`, `fwmark`, `route`, `rsvp` y `tcindex`. Entre los que destacan `u32` y `fwmark`, por su uso y utilidad.

U32

El `u32` proporciona una versatilidad enorme al permitir muchos criterios de selección entre los paquetes. Este es el más avanzado filtro disponible en la implantación actual para la Qdisc, lo cual hace que sea el más ampliamente utilizado. Por definición, este tipo de filtro permite distinguir en función de cualquier conjunto de bits, tanto de la cabecera del paquete IP, como de la cabecera del segmento de datos. Sin embargo, esté es bastante confuso y complicado de manejar, por lo que se suelen utilizar formas más directas para el filtrado.

Fwmark

Este filtro utiliza el marcado de un paquete para seleccionar tráfico. La marcación hecha por un sistema de cortafuegos sólo es valida dentro del sistema Linux, cuando son transmitidos los paquetes al exterior, ésta se pierde. Si se utiliza Netfilter como arquitectura [28], se puede marcar un determinado grupo de paquetes mediante iptables con un valor numérico. Después, se utilizar el filtro Fwmark para identificar dicha marca y colocar los paquetes en la cola asignada dentro de la Qdisc.

3.5.3. La arquitectura de Linux para la CdS

La manera de poder entender bien la relación que guardan los mecanismos de clasificación, filtrado y de encolamiento en el funcionamiento del núcleo de Linux, se describe de forma explicita en el diagrama de la arquitectura del sistema de red (ver Figura 3.4). El tráfico proveniente de la red es recibido primeramente por la disciplina de colas (Qdisc) de entrada, donde es posible aplicar filtros a los paquetes para decidir descartarlos o no. Esto ocurre en una etapa muy temprana, por tanto, es un buen lugar para descartar tráfico sin consumir mucho tiempo de CPU [24]. Si se le permite continuar la trayectoria a los paquetes, pueden estar destinados a una aplicación local. En cuyo caso

entran en la pila IP para ser procesados, y ser enviados a un programa en espacio de usuario. Los paquetes también pueden ser reenviados sin pasar por una aplicación, en tal caso se destinan en dirección a la salida. Ahí son clasificados, dirigidos y agregados a una de las varias Qdisc existentes, que pertenecen a las interfaces de red. A esto se le llama encolamiento.

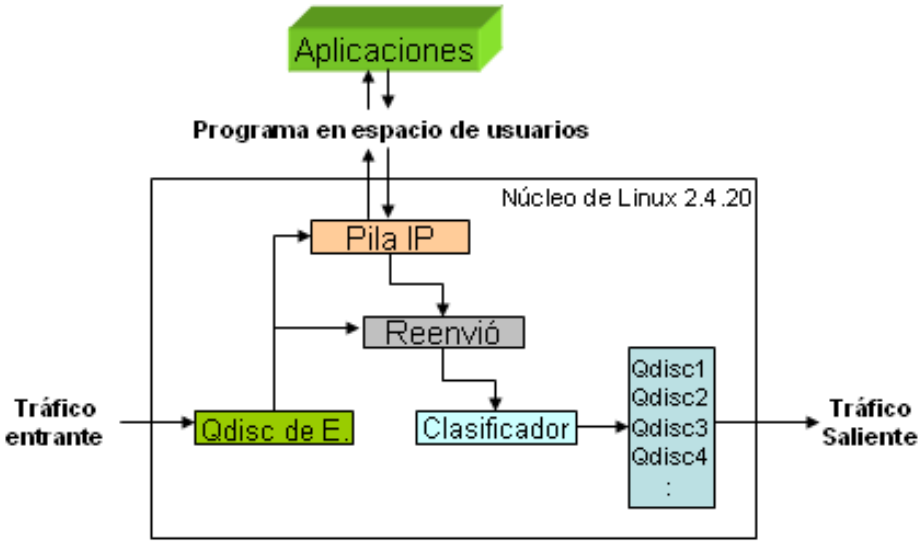

Figura 3.4: Arquitectura de Linux para el manejo de paquetes.

Ya dentro de la Qdisc, particularmente hablando de una Qdisc con clase, los paquetes sufren un nuevo proceso de clasificación para ser repartidos dentro de la estructura de clases, de la cola de algún dispositivo de red (ver Figura 3.5). Los filtros son llamados desde dentro de una Qdisc, para determinar a que clase dentro de la estructura se envían los paquetes. En caso de que existieran subclases dentro de la clase, las clases pueden probar de nuevo los filtros para indagar si se reenvían los paquetes hacia alguna de sus subclases. Esto se repite, hasta que finalmente se encole.

En la Figura 3.5 también ilustra el proceso de desencolamiento de un paquete. El proceso es ascendente, y va a través de una clase hija a su clase padre hasta llegar a la Qdisc raíz. De está forma, cuando el núcleo efectúa una petición de desencolamiento para enviar paquetes al dispositivo de red, la Qdisc raíz es la encargada de remitirle los paquetes. El mecanismo de selección de un paquete para ser enviado se basa de acuerdo ha parámetros de ponderación de cada Qdisc (principalmente por prioridad).

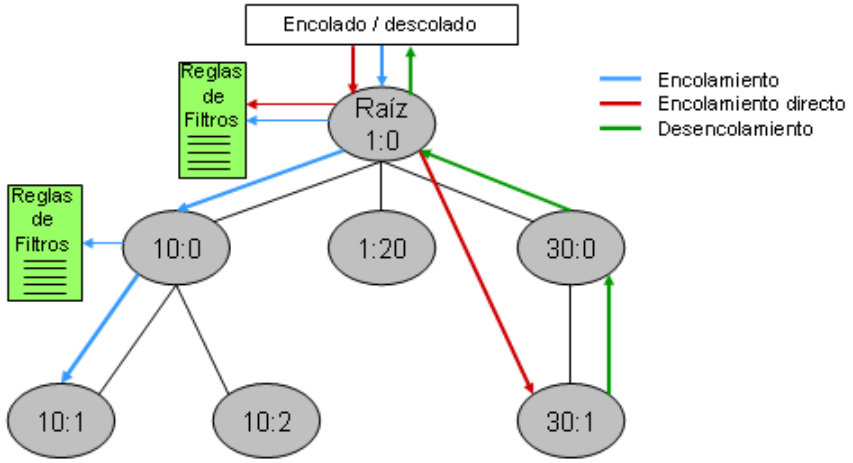

Figura 3.5: Procedimiento de encolamiento y desencolado de paquetes dentro de una Qdisc.

3.5.4. Dispositivo de intermedio de encolado (DIE)

El dispositivo intermedio de encolado (DIE) permite realizar el control de tráfico mucho antes de poder asignar la salida de los paquetes hacía un interfaz de red. Se logra, gracias al hecho de que dentro de Linux el DIE esta antes y después del sistema de ruteo, permitiendo hacer un verdadero control de tráfico de entrada (ver Figura 3.6). En realidad, DIE no hace nada y la única labor que cumple, es dejar pasar todo el tráfico según las disposiciones en anchos de banda o prioridades configuradas a la disciplina de cola que tiene asociada [29].

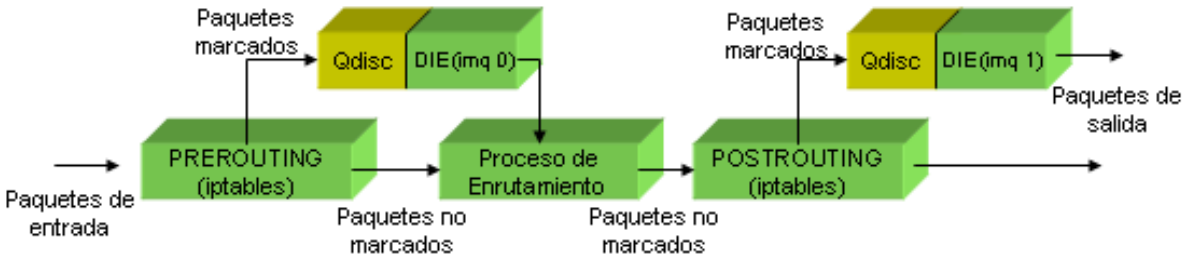

Figura 3.6: Flujo de paquetes cuando se activa los DIE's en el núcleo de Linux.

Los paquetes que entran o salen del sistema Linux, que hallan sido marcados por algún objetivo particular con la herramienta iptables, podrían ser dirigidos a alguno de los DIE's (Linux define el IMQ0 e IMQ1). Para que lo anterior sea posible, es necesario declarar inicialmente reglas de iptables que indiquen que clases de tráficos podrán tener acceso a los IMQ's.

A manera de resumen, las Qdisc's de los DIE's se privilegian de estar en un lugar clave dentro del sistema, logrando establecer dos nuevos enfoques a los ya tradicionales: definir políticas de control de tráfico globales, y poder tratar a los dispositivos de red de salida como clases de tráfico.

3.5.5. IPTABLES

Iptables es una herramienta capas de configurar, mantener e inspeccionar las reglas de cortafuegos IP del núcleo de Linux. La arquitectura de red llamado Netfilter [28], que utiliza `iptables`, emplea tablas con cadenas de reglas de decisión, colocadas en puntos estratégicos del sistema. Estos puntos de revisión de paquetes o también llamados enganches se encuentran dentro de la pila de protocolos de red. Linux proporciona cinco cadenas predefinidas (INPUT, FORWARD, PREROUTING, POSTROUTING y OUTPUT), aplicadas en los puntos de enganche dentro del sistema.

Netfilter integra tres posibilidades en el manejo de los paquetes, cada una de esas posibilidades, corresponde con una tabla. Estas son:

- **filter**: En la tabla filter, se llevan a cabo la tarea principal de `iptables`, el filtrado de paquetes. Está tabla contempla las cadenas INPUT, FORWARD y OUTPUT. La primera hace referencia a los paquetes entrantes cuyo destino es el propio cortafuego. La segunda decidir que hacer con los paquetes que llegan al cortafuegos y tienen como destino otro sistema terminal. Por última, la cadena OUTPUT se utiliza para filtrar paquetes generados en el propio cortafuego con destinos externos.

- **nat**: La tabla nat se utiliza para configurar el protocolo de Traducción de Direcciones de Red (NAT). Esta tabla tiene tres cadenas sobre las que podemos añadir reglas. La cadena PREROUTING se utiliza para alterar los paquetes tan pronto llegan al sistema (DNAT o NAT del destino). La cadena OUTPUT se utiliza para alterar los paquetes generados localmente, antes del enrutamiento. Finalmente tenemos la cadena POSTROUTING que altera a los paquetes que saldrán del sistema (SNAT o NAT en el origen).

- **Mangle**: La tabla mangle permite modificar o manipular los paquetes en cualquiera de los puntos de la pila de protocolo IP, a excepción del NAT. Una capacidad especial que tiene la plataforma Netfilter, es poder marcar un paquete IP con un

valor numérico y mantener un registro de ello. También consta de tres cadenas, PREROUTING, POSTROUTING y OUTPUT.

Si bien, la herramienta `iptables` no es propiamente un sistema de distribución y mucho menos uno de control de tráfico, ofrece características que pueden ser ventajosas en conjunto con otras herramientas para desempeñar un trabajo más versátil y útil.

3.6. Resumen

Lo que se mostró en este capítulo, son distintos sistemas que logran la administración de una red. Particularmente, Linux es uno de los mejores sistemas operativos que permite una distribución avanzada, y permite aplicar técnicas de CdS. No todo es perfecto en Linux, de hecho, su manejo y configuración para estas tareas es muy complicada para trabajos grandes, aun así, es una un instrumento confiable y eficaz.

Capítulo 4

Sistema de distribución de tráfico entre redes Locales

En este capítulo se describe la solución al caso de estudio, aplicado a Linux, que puede identificar, controlar y distribuir el tráfico de paquetes de dos o más redes de cómputo, permitiendo el acceso a los servicios de estas redes en una sola red común de trabajo. Inicialmente, se describen las necesidades y beneficios que se esperan lograr con éste. Posteriormente, se realiza el análisis y el esquema de funcionamiento del sistema. Se concluye, describiendo el modelo de configuración propuesto para Linux, el cual se utilizará para la implementación del prototipo desarrollado en esta tesis.

4.1. Solución del caso de estudio

En esta sección se habla del contexto donde se coloca y emplea el sistema propuesto que da solución al caso de estudio, con la intención de delimitar el espacio de trabajo.

La mayoría de las organizaciones actuales no necesiten aplicaciones de tiempo real, sin embargo hay la necesidad de distribuir el tráfico de una forma más individualizada y de garantizar CdS, para poder gestionar adecuadamente los recursos limitados de las diferentes conexiones a redes externas con que estas cuentan. La solución propuesta al caso de estudio, llamada Sistema de Distribución de Tráfico sobre Redes Locales(SDTRL), esta enfocada principalmente a la labor de distribución de paquetes y de forma secundaria al control de tráfico, para la administración de recursos en el punto de frontera de una red, tal como se muestra en la Figura 4.1.

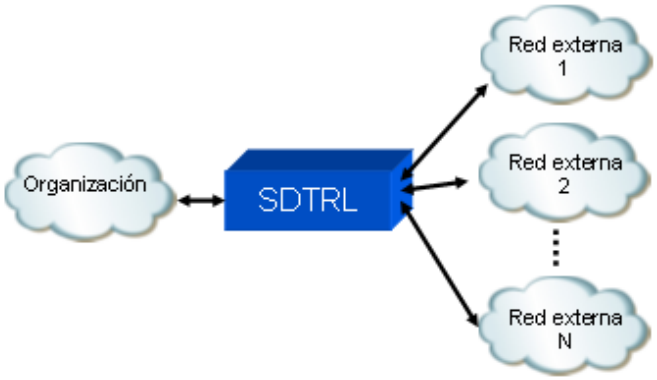

Figura 4.1: Los usuarios de la organización utilizan los servicios de distintas redes externas.

4.1.1. Escenarios de trabajo del SDTRL

En este apartado, se describen los escenarios de trabajo que pueden ser implementados y solucionados por medio del SDTRL. Antes de comenzar con la descripción, es conveniente señalar las redes participantes en el entorno de operación, mencionadas en el capítulo uno. La red común de trabajo de una organización, que hace uso de una o más conexiones de redes externas, es una red de área local (RAL) privada. Así mismo, se plantean dos casos de interconexión entre redes de cómputo, los cuales delimitan el nivel homogeneidad con respecto a las redes externas que se emplean:

- **Primer caso**: Se tiene dos o más puntos de conexión a redes externas diferentes (posiblemente de tecnologías distintas), las cuales, los usuarios de la organizan necesitan utilizar.
- **Segundo caso**: Se tiene dos o más puntos de conexión a redes externas iguales (posiblemente de proveedores diferentes), donde quizá tengan propósitos distintos dentro de la organización.

La interconexión entre la red privada y las redes externas lleva de forma implícita la labor de distribución de tráfico. Por esta razón, se describen a continuación los dos casos anteriores como dos escenarios de trabajo en donde se desenvuelve el SDTRL.

El primer escenario de interconexión de redes es como se muestra en la Figura 4.1. El SDTRL es capaz de poder distribuir el tráfico de forma bidireccional dentro de la red común de trabajo con las diferentes redes externas. Cuando la distribución de tráfico esta basada en un direccionamiento por IP destino, el sistema de enrutamiento tradicional del SDTRL lo realizan sin mayor problema. Sin embargo, cuando las políticas

de distribución de tráfico se basen en otras características de los paquetes; en donde los ruteadores ya no son muy eficientes para ello. El SDTRL toma otras medidas para lograr la adecuada distribución del tráfico.

El segundo escenario, se establece en un lugar de trabajo donde se tiene la necesidad de interconectar y usar redes de igual tecnología, como se ejemplifica en la Figura 4.2. Un primer enfoque para el SDTRL consiste en balancear equitativamente el flujo de tráfico de la organización entre las diferentes conexiones de red. Un enfoque alterno se presenta, cuando es necesario identificar el tráfico que es enviado hacia Internet. Esto es con la finalidad de controlar las clases de tráfico que vayan hacia una tarea de misión crítica y enviarlas a un proveedor de servicios de Internet (PSI) que es utilizado para este fin. Las clases de tráfico restantes serán enviadas a otros PSI's de acuerdo a la conveniencia de la organización. Este esquema se puede extender a cualquier tipo de red.

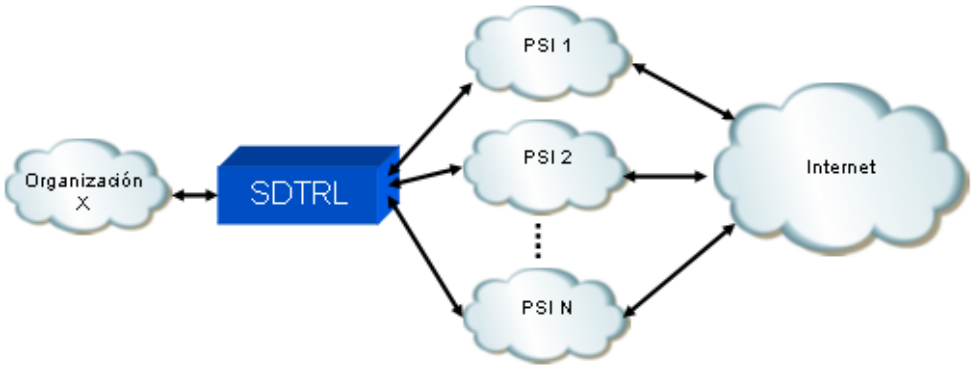

Figura 4.2: Utilización de varios PSI's para una organización.

En los dos escenarios anteriores, el control de tráfico en el SDTRL se establecer por medio del control de asignación de ancho de banda y la planificación de prioridades entre paquetes, con respecto a los requerimientos genérales de la organización y para las necesidades particulares de cada uno de los usuarios. El propósito es administrar y controlar los recursos provistos por las redes externas hacia dentro de la red de la organización. Para esto, el SDTRL aplica un conjunto de varias técnicas de calidad de servicio, tanto para el flujo de paquetes de salida como el de entrada, entre las redes externas y la red común de trabajo.

4.1.2. Restricciones del SDTRL

El SDTRL es un sistema intermedio, capaz de encaminar el flujo del tráfico hacia las diferentes redes interconectadas. El sistema cuenta con una sola dirección de red de cada una de las redes conectada a ella. Por tanto, si la red de la organización es una red privada, que necesita hacer uso de los servicios que proveen las redes externas, entonces el SDTRL necesita establecer un proceso de enmascaramiento (NAT) entre las redes externas y la red privada.

El SDTRL no maneja algún protocolo de calidad de servicio, como puede ser el ServDiff o ServInt, para propiciar las condiciones necesarias para el manejo del tráfico en tiempo real (revisar la Sección 2.4.1). Únicamente, utiliza herramientas de CdS (ver Sección 2.4.3) que permiten administrar y controlar los recursos de una red.

El SDTRL no agrega funciones de cortafuego de tipo Filtrado o de tipo Proxy, por la razón de evitar sobrecarga de trabajo al sistema. Pero esto no indica que no las pueda tener, de hecho, es recomendable tener un cortafuego tipo Stateful según [30]. Para ser más preciso, el sistema no podrá filtrar, rechazar o negar servicios a los paquetes que entran y salen de este. Solo se limita a distribuir el tráfico en función de las reglas de políticas de enrutamiento.

Las cuestiones de seguridad se dejan en manos de la misma organización o en casos especiales a los propios proveedores de servicios de red. Sea cual sea el caso, la configuración recomendable debe tener un cortafuego entre SDTRL y cada punto de conexión de las redes externas, como se ilustra en la Figura 4.3.

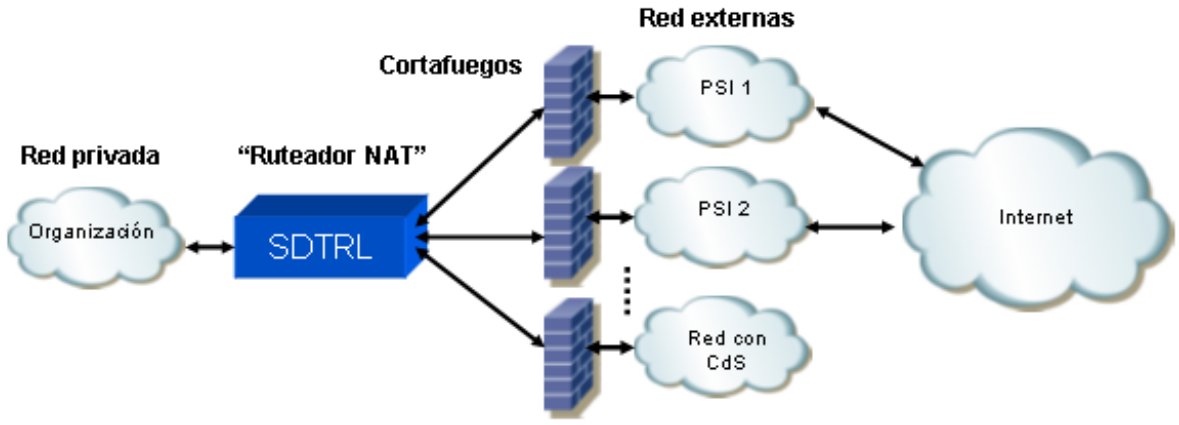

Figura 4.3: Arquitectura recomendable de trabajo en cuestiones de seguridad.

4.1.3. Funciones del SDTRL

El SDTRL puede realiza las siguientes funciones generales:

- Identificación del tráfico por diversas características del encabezado de los paquetes en los niveles 3 y 4 del modelo OSI para establecer clases de tráfico.
- Establece políticas de enrutamiento por destino a las clases de tráfico cuando dos o más redes lógicas son diferentes.
- Establece enrutamiento en base a políticas (EBP) para clases de tráfico, cuando las redes lógicas tienen propósitos diferentes.
- Establece mecanismos de balanceo de carga cuando las redes lógicas tienen propósitos similares.
- Establece criterios de calidad de servicio, tales como asignación de ancho de banda y priorización de tráfico, cuando las redes lógicas tienen propósitos diferentes o cuando los servicios que ofrecen no son igualmente accesibles a todos los nodos de la red local.

4.2. Marco de referencia conceptual para el SDTRL

Dado que la red común de trabajo de la organización solo hace uso de los servicios de las redes externas y no proporciona servicios hacia el exterior de la misma, el SDTRL vislumbra tres tipos de tráfico: *el tráfico masivo*, *el tráfico particular* y *el tráfico de respuesta*.

El *tráfico masivo* es aquel flujo de paquetes provenientes del uso generalizado de una aplicación dentro de la organización en dirección hacia las redes externas. Ejemplos típicos de este tipo de tráfico son flujos http, telnet, ssh, o cualquier tráfico con una especial importancia a nivel organizacional. El interés de definir este tipo de tráfico es especificar reglas generales a partir de las políticas internas de la organización. Es decir, cada regla refleja la definición de un tráfico masivo que tiene establecido una prioridad de atención y asignación de ancho de banda de acuerdo a los mejores intereses. El tráfico masivo se precisa en cada uno de los nodos[1] de las redes externas, ya sea igual o de manera distinta.

[1]El término de nodo hace referencia a una conexión de una red externa, o a la integración de dos o más conexiones a redes externas que tienen como propósito balancear la carga entre ellas.

El *tráfico particular* se emplea para identificar los flujos de paquetes, ya sea de forma individual o conjunta, de los sistemas terminales o de las aplicaciones que estos últimos utilizan, pertenecientes a la red de la organización en dirección a las redes externas. El definir este tipo de tráfico facilita la individualización de forma precisa de **la ruta de salida**, **la asignación de ancho de banda** y **la prioridad** de ciertos de flujos de paquetes específicos. Para ello, el tráfico particular se define inmediatamente después de haber entrado al SDTRL, esto ocurre solo con los paquetes de entrada provienen de la red privada de la organización. Asimismo, el tráfico particular tiene la peculiaridad de pertenecer también a una clasificación de tráfico masivo, debido que a su vez, se le asigna otro ancho de banda y prioridad dependiendo de la salida que esté tome.

El *tráfico de respuesta* es el flujo de paquetes de contestación, originado por cierta petición hecha por alguna estación terminal de la red común de trabajo. Gracias a su identificación es posible controlar y priorizar los paquetes de respuesta de las solicitudes de servicio a las redes externas.

El SDTRL maneja un conjunto de reglas que permiten manifestar la existencia de cada uno de estos tipos de tráficos. Las *reglas de tráfico masivo* detallan el control de tráfico, para establecer las políticas internas de la organización en cada una de las conexiones externas. Las *reglas de tráfico particular* permiten que las aplicaciones o las estaciones terminales de la red común de trabajo puedan tener rutas especificas, asignaciones de anchos de banda y prioridades distintas a las *reglas de tráfico masivo*, ya que a estas ultimas les resulta imposible identificar el origen de los paquetes dentro de la red. Las *reglas de tráfico de respuesta*, especifican las formas de control de los flujos de paquetes enfocados hacia el interior de la red común de trabajo, emanados de las redes externas. El administrador del SDTRL es el único conciente de las necesidades de la organización y de los usuarios, y por ende es responsable de la definición de tales reglas. Para el caso de los usuarios de la red, todo lo anterior les es transparente, ya que no tienen noción de la diferenciación del tráfico y solo perciben los resultados de las aplicaciones de dichas reglas.

La capacidad del SDTRL de identificar los diferentes flujos paquetes de datos, permite la creación de *clases de tráfico*, las cuales están conformadas por paquetes con características similares en sus encabezados. Cada clase de tráfico se diferencia entre ellas por una serie de posibles combinaciones propias (ver Tabla 4.1) establecidas por una regla.

Tabla 4.1: Puntos de distinción entre clases de tráficos.

Opciones de emparejamiento entre los paquetes	Nivel en OSI
Interfaz de entrada del flujo Interfaz de salida del flujo	Nivel 2
Dirección IP destino Dirección IP fuente Numero Protocolo Tipo de servio Tipo de mensaje ICMP	Nivel 3
Número de puerto fuente de TCP y UDP Número de puerto destino de TCP y UDP Banderas de TCP (URG, ACK, PSH, RST, SYN, FIN) Opciones particulares de los paquetes TCP	Nivel 4

4.3. Distribución de tráfico del SDTRL

El SDTRL es un sistema para distribuir el tráfico hacia alguno de los nodos de las redes externas con base en características específicas de los paquetes IP's, descritas en las *reglas de tráfico particular*. El encaminamiento del SDTRL se realiza en función de los intereses de cada organización y de las necesidades de los usuarios. Esta manera de encaminar puede parecer ser subjetiva y posiblemente poco precisa, si fuera únicamente en función de estas perspectivas. Pero no es así, la organización sabe exactamente que conexiones posee; distingue las redes de menos retardo, de más ancho de banda, como también saben cual es la menos confiable, y de acuerdo con este conocimiento se crean las políticas de enrutamiento. Además, el tráfico ya dentro de las redes externas tiene un comportamiento fijado por los protocolos de enrutamiento provistos por los ruteadores internos.

El acceso dividido es una característica fundamental en el SDTRL, pues las redes externas con las que se conecta son entidades independientes. Por esta razón, se hace forzoso que las peticiones al SDTRL providentes de algún lugar de las redes externas, sus respuestas deban ser dirigidas a los mismos nodos por donde entraron las peticiones, a esto se le llama acceso dividido. Pasa igual para las comunicaciones que salen o entran, sus respuestas deben ser enviadas al mismo lugar de entrada o salida, como se ejemplifica en la Figura 4.4.

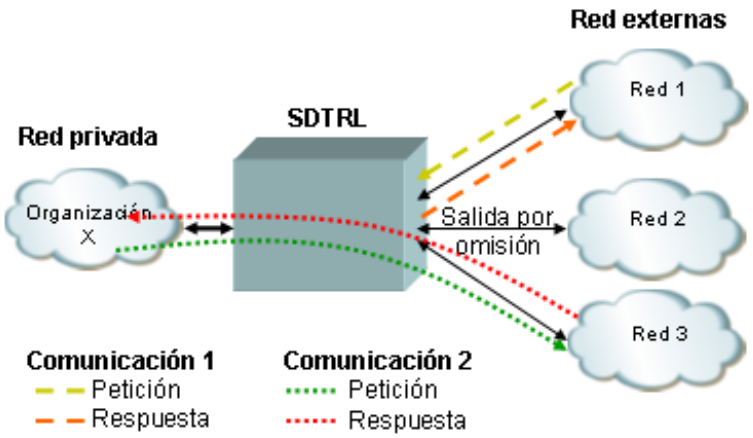

Figura 4.4: Acceso dividido. Sale o entra una comunicación por el mismo lugar que entró o salió respectivamente.

4.4. Control de Trafico del SDTRL

Las *políticas internas* en el contexto del SDTRL detallan la manera de repartir los recursos asignados por las redes externas a la red común de trabajo. Estos acuerdos pueden ser diferentes en cada uno de los enlaces externos, dependiendo de la forma de uso que requiera la organización.

También es necesario asignar recursos de manera individual por cada sistema terminal o grupos de sistemas terminales de acuerdo al trabajo que realicen, ya sea por niveles de responsabilidad que manejen u otros factores. De ahí que, se hace indispensable definir las *reglas particulares*, independientes de las *políticas internas*, que se encarguen de resolver estás tareas.

El SDTRL aplica el control de tráfico a los tres tipos de tráfico que circularán dentro de él. A continuación se describe la manera de operación de estos.

Tráfico masivo

La asignación de recursos para un tráfico masivo se maneja de la siguiente forma:

- Las reglas de asignación de recursos de tráfico masivo para un nodo se establecen en cuanto a uso y utilidad dentro de la organización.

- La asignación de ancho de banda para un tráfico masivo pude ser de forma estática o dinámica.

La asignación estática permite asignar un ancho de banda a un tráfico específico de forma única y excluyente de los demás.

La asignación dinámica permite que los tráficos que circulan hagan uso del ancho de banda disponible. Es decir, si se detectara un tráfico masivo que tiene reservado un ancho de banda, este se asignara automáticamente provocando que los otros tráficos liberen ancho de banda para darle cabida al tráfico nuevo.

La asignación estática se parece mucho a la repartición de un pastel, a cada invitado le corresponde una rebanada. Esta es la razón, de que el ancho de banda este limitado a un número definido de tráficos. En contraste, la asignación dinámica es más parecida al paso de una caravana de ambulancias dentro de una carretera llena. Los automóviles de un carril ceden su espacio para dejar pasar a las ambulancias, y cuando éstas hayan pasado, ellos retomaran su lugar.

- El tráfico masivo que tenga el menor nivel de prioridad, es el tráfico que se atenderá primero, y así de manera creciente hasta el nivel más alta.
- El SDTRL concede un ancho de banda a un tráfico masivo en base a un nivel de prioridad. La asignación dinámica se controla por los niveles de prioridad que tienen las clases de tráfico que existen en este momento.
- El tráfico masivo no determinado (el no definido dentro de una regla) se atenderá como el tráfico masivo por omisión, el cual tendrá un ancho de banda limitado base y se considerara el de menor prioridad. Este siempre existirá en cada nodo de red.

Tráfico particular

Ahora examinaremos un caso de interés en relación al tráfico particular y masivo, que mostrará la existencia de clases y subclases de tráfico en el SDTRL (ver Figura 4.5).

Para esto, suponemos que existe una clase de tráfico masivo llamada **XYZ**, la cual engloba a tres clases de tráfico particular identificadas como **X**, **Y** y **Z**. Los nodos de red **a**, **b** y **c** (mostrados en la Figura 4.5) tienen reservados recursos iguales para la clase de tráfico masivo **XYZ**. Asimismo, se presume que un tráfico llamado **Ax** que pertenece a la clase **X**, tiene un ancho de banda y prioridad definida por una regla de tráfico particular. De igual forma, en la clase **Y**, existe un tráfico **By** con otras asignaciones de

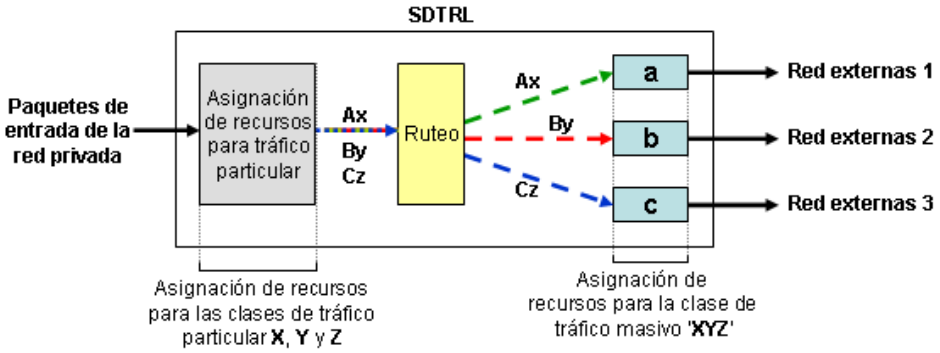

Figura 4.5: Los tráficos X, Y y Z pertenecen a una clase de tráfico particular, y se encaminan a distintos nodos.

recursos distintas a la anterior y lo mismo pasa para la clase **Z** la cual tiene un tráfico llamado **Cz**.

Se sabe que, la clase **X** tiene la misma asignación de ancho de banda que la clase **XYZ**, la clase **Y** tiene mayor ancho de banda asignado que **X**, y la clase **Z** tiene menos ancho de banda asignado que alguna de las clases anteriores. Entonces, el trafico **Ax** no tendrá ninguna reducción de ancho de banda al pasar por la clase **XYZ** del nodo **a**. El trafico **Cz** tampoco presentara algún problema en cuanto a su ancho de banda al pasar por el nodo **c**, pero es evidente que el nodo **c** podría manejar una mayor velocidad de transmisión. Para el caso del tráfico **By**, el flujo paquetes esta limitado por la clase **XYZ** que empezara a descartar paquetes por tener asignado un ancho de banda menor. Lo anterior, describe un sistema jerárquico de clases en el SDTRL, ya que las clases de tráfico particular siempre actúan como subclases de las clases de tráfico masivo.

Al igual que el tráfico masivo, el tráfico particular no considerado debe ser atendido por una regla de tráfico particular de omisión, el cual, se le reserve un ancho de banda y prioridad, que para este caso es ajustable.

Tráfico de respuesta

Este tipo de tráfico solo se limita a controlar el tráfico de respuesta, en la perspectiva de la red privada. Una de las limitaciones que posee el tráfico de respuesta es el de no poder identificar los tráficos bien conocidos, como en el caso del tráfico masivo. Esto debido a que los clientes que solicitan servicios, hacen peticiones a través de los números de puertos bien conocidos a las aplicaciones servidoras. Las respuestas de estas peticiones se realizan a través de un número de puerto que es asignado de manera dinámica por el sistema operativo del cliente.

Finalmente, el manejo de prioridades es independiente al tipo de tráfico (masivo, particular o de respuesta), solo determina el orden de atención de las diversas clases dentro de cada bloque de asignación de recursos.

4.5. Funcionamiento General del SDTRL

El tráfico que circula internamente en la red privada, no tiene interés para el SDTRL. En cambio, cuando algún sistema terminal dentro de la red requiere de servicios provistos por las redes externas, el SDTRL interviene realizando la distribución de los paquetes, con la peculiaridad de que no fijar una ruta especifica al destino como normalmente lo hace un ruteador, pues únicamente especifica la salida por algún nodo.

En la Figura 4.6a muestra la topología adecuada para el SDTRL. Este se conecta a nodos externos, en donde cada nodo puede estar acoplado a una o más conexiones de redes externas, para interconectarse con la red privada de la organización, y en ella conectar n-cantidad de sistemas terminales.

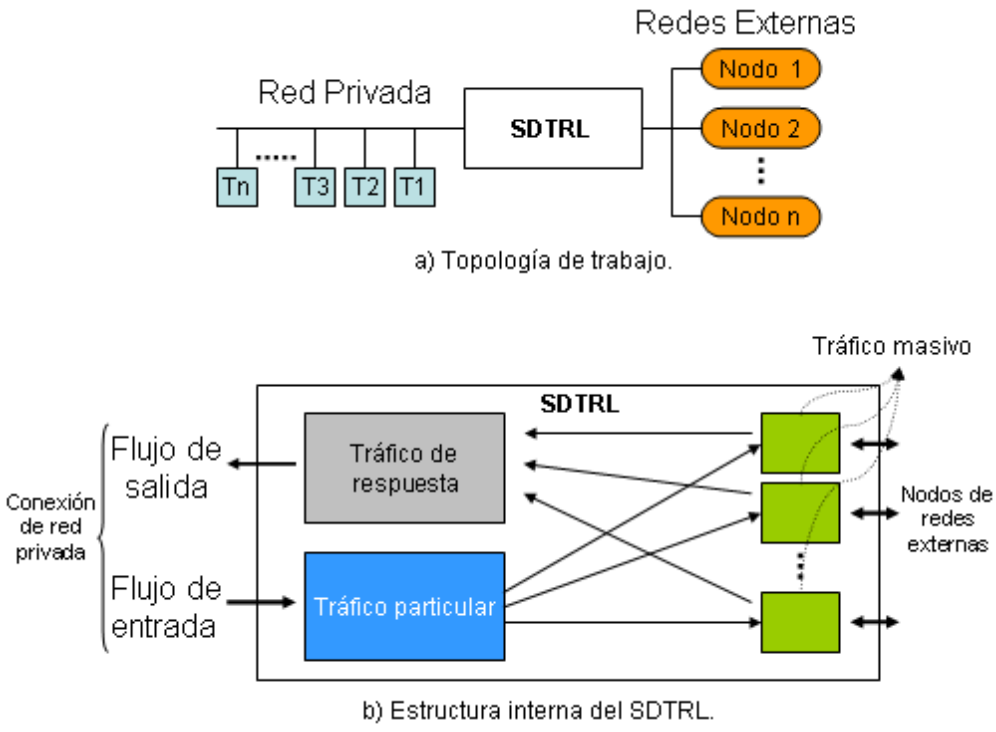

Figura 4.6: Módulos generales del SDTRL.

El SDTRL contempla tres módulos principales de acción que manejan de manera bidireccional el tráfico entre la red privada y las redes externas, como se indica en la

Figura 4.6b. El tráfico que se dirige a las redes externas proveniente de la red privada, circula dentro del SDTRL pasando por el módulo de tráfico particular y algún módulo de tráfico masivo. Para los flujos de paquetes en dirección opuesta, el modulo de tráfico de respuesta es el responsable de controlar el acceso a la red de la organización.

El módulo de tráfico particular se encarga de identificar el tráfico, asignándole una dirección de salida y un ancho de banda. La salida a la que deben encauzarse las diversas clases de tráfico particular depende directamente de las reglas del EBP descritas explícitamente por el administrador.

En el modulo de tráfico masivo, se determina realmente el ancho de banda de salida de las distintas clases de tráfico, hacia las redes externas. Sin embargo, puede existir una clase de tráfico bien conocido que contenga dos o más subclases que posean anchos de banda distintos debido a que éstas provienen de diferentes clases de tráfico particular, donde se les manejo de manera distinta. Esto indica que una clase de tráfico masivo limite "el máximo ancho de banda permitido", para las diversas subclases de tráfico pertenecientes a ella.

El módulo de tráfico de respuesta servirá de apoyo para poder controlar el flujo de entrada a la red privada proveniente de alguno de los nodos de las redes externas, en ella las clases manejaran un ancho de banda y prioridad que permite controlar el acceso.

Si bien, el SDTRL contempla los posibles flujos de datos de entrada como de salida a la red privada de la organización, de igual forma permite controlar el ancho de banda y manejar prioridades entre los paquetes. Esto da lugar al SDTRL de ofrecer un conjunto de servicios. Para ello, primero se debe definir la ubicación del sistema con respecto a las dos arquitecturas de CdS, para dejar claro el campo de acción del sistema.

El administrador del ancho de banda es un componente básico del modelo IntServ, pero requiere de un protocolo de reservación de recursos. Por otro lado el modelo DiffServ clasifica los paquetes de acuerdo a una prioridad señalada en el encabezado de los mismos. El SDTRL sigue un modelo informal de gestión para ruteadores DiffServ, es decir, marca los paquetes para que sean tratados de forma preferencial exclusivamente dentro del mismo ruteador. Los paquetes que salen del sistema hacia una red son tratados de manera tradicional, ya que simplemente actúa como un distribuidor, con llaves de paso para flujos de paquetes específicos.

Por la parte de los servicios del SDTRL, estos se aplican en los tres módulos de tráfico. En cada uno de estos módulos se define un único servicio para los diferentes flujos de paquetes. Este servicio provee al tráfico la capacidad de asignación mínima de ancho de banda garantizada y una prioridad de atención con respecto a todo los tráficos

del modulo donde se encuentre. El tráfico no clasificado de cada módulo tendrá el servicio con una asignación de ancho de banda definida y una prioridad mínima. Sin embargo cada modulo tiene un enfoque y tarea distinta.

4.6. El SDTRL en Linux

El sistema planteado anteriormente da una solución al caso de estudio, independiente de la plataforma en el que se desarrolle. Sin embargo, el SDTRL es llevado al sistema operativo Linux, pues éste dispone de muchos mecanismos y características avanzadas que facilitan la distribución y el control de tráfico. Por ello, se explica en esta sección la forma de configurar al núcleo del sistema para que cumpla con todo lo descrito por el SDTRL. Finalmente, el modelo de configuración que se desprende de esta sección, es utilizado para llevar a cabo la implementación del sistema prototipo.

4.6.1. La base de tecnológica

La propuesta del SDTRL esta basada en la plataforma Linux con el núcleo 2.4.20 con las siguientes características:

- Se hace uso de la herramienta Iptables versión 1.2.6a
- Se hace uso de la herramienta Iproute2 versión iproute2-ss010824
- Se requiere parchar el núcleo 2.4.20 de Linux para agregar la funcionalidad del dispositivo intermedio de encolamiento (DIE). También se agregan las bibliotecas a la herramienta Iptables, para poder hacer uso del DIE.
- Se agrega una nueva versión del programa tc del paquete Iproute2 al sistema Linux, la cual percibe y permite hacer usa de la disciplina de cola HTB (Hierarchical Token Bucket).

4.6.2. Modelo de configuración base para el SDTRL

La configuración del sistema Linux, que se propone, para solucionar el caso de estudio tiene como principal filosofía la idea de "divide y vencerás". Esto es corroborado, cuando una sola disciplina de cola, de una interfaz de red, tiene la tarea de manipular el tráfico masivo y el tráfico particular, teniendo como resultado una estructura muy compleja,

difícil de controlar. Por esa razón, la configuración divide la responsabilidad del tráfico particular, masivo y de respuesta en diferentes interfases de red dentro del sistema.

Una situación de profundo interés dentro la SDTRL, es la cuestión de la identificación inicial de todas las interfases de red, y las redes con las que están conectadas, para especificar los nodos de las redes externas y el nodo de la red privada. El SDTRL en Linux efectúa el enmascaramiento para ligar los servicios provistos por las redes externas a la red privada. Además, al inicio del sistema se establece el acceso dividido para cada una de las interfases de red, para evitar problemas de conexión.

El modelo de configuración para Linux que utiliza el SDTRL, describe la trayectoria que siguen los flujos de paquetes y el tratamiento que éstos reciben dentro de la arquitectura del núcleo 2.4.20, como se muestra en la Figura 4.7.

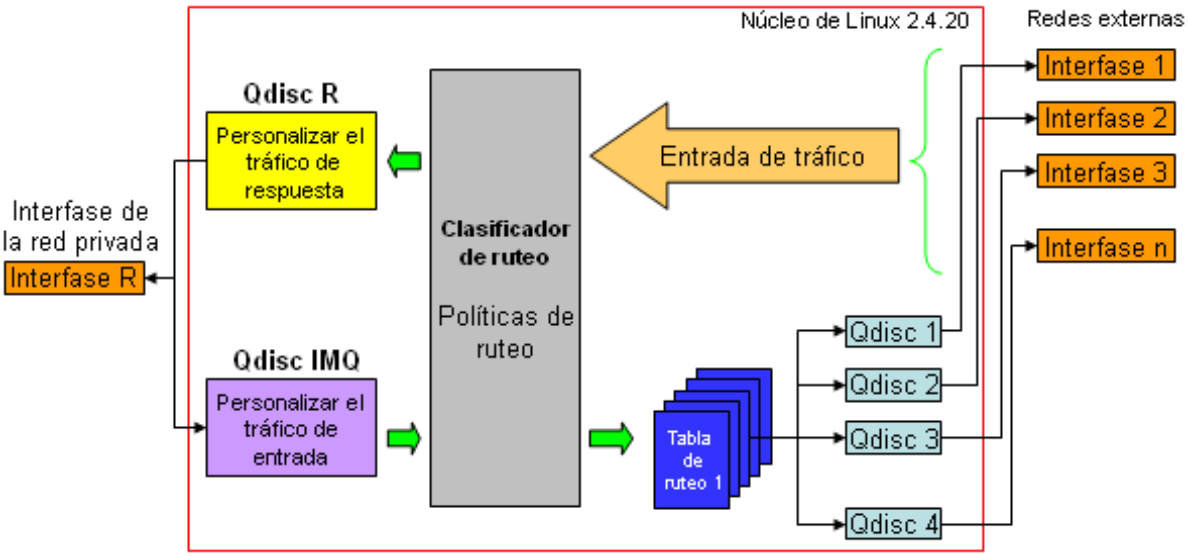

Figura 4.7: Propuesta de la arquitectura General del SDTRL en la plataforma Linux.

El tráfico de entrada al sistema Linux proveniente de la red privada de la organización, se dirige inicialmente a la cola del dispositivo intermedio de encolamiento (DIE). La labor principal del DIE es fijar un ancho de banda y nivel de prioridad para las diversas clases de tráfico particular; bloque identificado con la etiqueta "Qdisc del DIE". En seguida, el flujo de paquetes es mandado a la tabla de enrutamiento la cual decide a que interfase se envía. Para lograr la distribución de tráfico avanzado se hace uso de las múltiples tablas de enrutamiento, que provee el sistema. Las reglas de políticas de enrutamiento se encargan de que cada clase de tráfico particular definida, haga uso de una de varias tabla de enrutamiento disponibles. Tanto, el control y distribución de

tráfico hasta ahora mencionados conforman el módulo de tráfico particular.

Después del proceso de enrutamiento, inicia el módulo de tráfico masivo, donde los paquetes son enviados a la cola del dispositivo de red perteneciente a las redes externas. La disciplina de cola de éste dispositivo se encarga de identificar las diversas clases de tráfico que asignan un ancho de banda y prioridad.

Por otra parte, el tráfico de entrada al SDTRL proveniente en esta ocasión de las redes externas no tiene puntos de revisión hasta la tabla de ruteo que le corresponde. Si el tráfico tiene como destino la red privada de la organización, entonces éste es enviado directamente a la cola de la interfase de dicha red; como se muestra en el bloque identificado con la etiqueta "Qdisc de R"en la Figura 4.7. En este bloque se realizan las tareas de asignación de recursos para las clases de tráficos de respuesta.

4.7. SDTRL implantado en Linux

El modelo de configuración, descrito atrás, puede ser implementado con ayuda las herramientas `iptables`, `ip` y `tc` en Linux. Mediante dichas herramientas se puede resolver el caso de estudio, pues es una alternativa precisa y flexible, pero compleja para configurar, y difícil de coordinar para sistemas de redes grandes.

El SDTRL pretende ser un sistema de uso fácil sin la complejidad técnica, y con reglas concretas que realizan el enrutamiento en base a políticas (EBP) y las tareas de asignación de recursos. Una de las maneras de implantar el prototipo SDTRL, en la plataforma Linux, es utilizando en conjunto los paquetes iptables/Netfilter e iproute2, tal se muestra en la Figura 4.8. En dicha figura, el bloque de SDTRL actúa como un mecanismo de configuración de las herramientas iptables, ip y tc, creando con ello una máscara que ocultado gran parte de los detalles técnicos. Donde, las órdenes de un conjunto de reglas propias determinan el comportamiento del sistema. También existe el módulo de integración, que permita relacionar y controlar las acciones que realizan las herramientas implicadas, con el fin de que tengan un desempeño armónico.

Dentro de Linux, la manera de proporcionar políticas de enrutamiento es aplicando la base de datos de políticas de ruteo (BDPE), que selecciona la ruta apropiada ejecutando un conjunto de reglas que apuntan a tablas de enrutamiento. Estas reglas pueden seleccionar cierto tipo de tráfico, explícitamente por la dirección fuente del paquete, la dirección destino, por el campo ToS, interfaz de entrada, y valores de marca (`fwmark`). Aunque hay que destacar que la manera de cómo Linux fijar una ruta para una clase de tráfico es mejor que la acción que realiza el software de gestión de tráfico de Cisco [2],

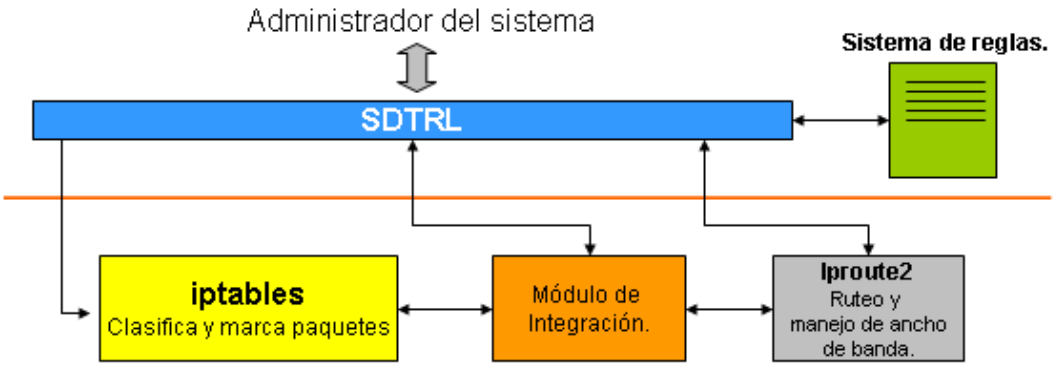

Figura 4.8: Diagrama de interrelación de paquetes de software y módulos de programas para implementar el SDTRL.

donde solo selecciona la siguiente dirección de salto y el dispositivo de salida.

Como ya se mencionó, el módulo de tráfico particular (referido a la Sección 4.3) hace uso del enrutamiento en base a políticas, además también otorga recursos a las diversas clases que de ella se desprenden. Es evidente que la BDPE de Linux no logra esto del todo.

Para implementar el EBP en Linux es necesaria la utilización conjunta de las herramientas `iptables` e `ip`. Donde iptables permite hacer una selección mucho más precisa de las clases de paquetes (mediante el marcando un paquete) que la que nos permite manejar `ip`. Por su parte, `ip` es la encargada de decretar el trayecto de las clases hacía un nodo externo, en función del seleccionador `fwmark`.

Finalmente, para el control de tráfico del SDTRL. En el DIE se utiliza la herramienta `tc` para asignar a las clases de tráfico particular el ancho de banda y prioridad. De igual forma, los modulo de tráfico masivo y de respuesta pueden ser configurados con la herramienta `tc`, en las interfases de las redes externas y de la red privada respectivamente.

4.8. Resumen

El SDTRL tiene como primer objetivo resolver el caso de estudio, la capacidad de controlar el ancho de banda y manejar prioridades entres los paquetes es un valor agregado al sistema. Sin embargo, el SDTRL pretende ser una herramienta útil y simple, que intenta facilitar, controlar y reducir las mayores cantidades de errores, al momento de la configuración.

Más allá de las políticas de enrutamiento. El SDTRL esta orientado a poder distribuir el tráfico en función de factores que dependen de las necesidades de la organización y

de los propios usuarios. Asimismo, con la definición de los tres tipos de tráfico, se logra tener el pleno dominio de la velocidad de transmisión de los flujos de paquetes, tanto de entrada como el de salida a la red común de trabajo.

Debido a que el sistema Linux tiene características avanzadas tanto para la distribución y el control de tráfico. El establecimiento del SDTRL en Linux tiene como especial interés facilitar la configuración del sistema. La razón es que Linux dispone de mecanismos divididos para establecer el EBP, también dispone varias disciplinas de cola y diversos filtros, sin mencionar que las herramientas de software que se utilizan tienen huecos de validación, que dificultan de alguna manera la implantación del modelo configuración, descrito en la Sección 4.4.2.

Capítulo 5

Diseño e implementación del SDTRL

En este capítulo se presentan los subsistemas que conforman el SDTRL, como unidades independientes que ofrecen servicios entre sí. Se definirán las estructuras que son usadas en el diseño, así como los algoritmos utilizados en la implementación. Para describir la manera de cómo se diseño e implementó el SDTRL, se parte del hecho de conocer las tecnologías involucradas y de seguir el modelo de referencia descrito en el Capítulo 5.

El SDTRL en función de Linux, es un sistema capaz de poder implementar el enrutamiento en base a políticas (EBP), y facilitar aún más el control de tráfico tanto de entrada como el de salida entre las redes externas y la privada. Este sistema se construye en la plataforma Linux 2.4.20, con la ayuda de `iptables`, `ip` y `tc`, y se plantea como un sistema estructurado desarrollado con el lenguaje de programación C.

5.1. Arquitectura del sistema

En el diagrama de la Figura 5.1 se muestran los subsistemas que conforma SDTRL, cada bloque es un sistema independiente, que se relaciona con otro; representado mediante líneas con puntas de flechas señalado el intercambio de servicios.

El SDTRL se divide en cuatro subsistemas principales. El sistema de inicialización, encargado de poner a los otros sistemas en las condiciones propicias para que éstos cumplan su cometido y más aún es el responsable de poner al sistemas operativo Linux en el contexto necesario para que los otros sistemas puedan trabajar. El sistema de distribución y control de tráfico particular esta conformado a su vez de dos subsistemas, uno encargado de manejar y establecer la ruta de las diversas clases de tráfico en función de las políticas de ruteo, seguida del subsistema encargado de asignar el ancho de banda

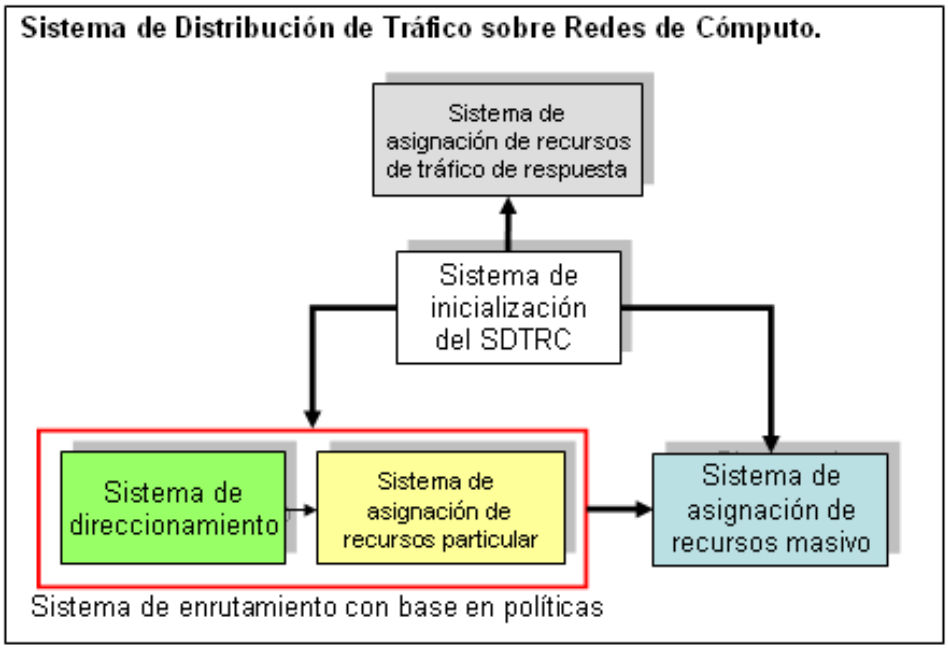

Figura 5.1: Diagrama de bloques del SDTRL.

y determinar cual clase debe ser atendida primero.

Los sistemas de asignación de recursos masivo y de respuesta realizan tareas similares, que determinan realmente la velocidad y orden de salida del SDTRL de las diferentes clases de tráfico, salvo que la diferencia radica en la dirección que toman los flujos de paquetes. El primero dirige el flujo hacia las redes externas y el segundo sistema hacia la red privada, controlando así, la salida y entrada respectivamente desde el punto de vista de la red de la organización.

5.2. Sistema de Inicialización del SDTRL

Antes que nada, el sistema operativo Linux debe ponerse en las condiciones necesarias para desempeñar las labores que persigue el SDTDL. Para ellos, primero se realiza, por parte del administrador, la identificación de los diferentes nodos de red (descritos en la Sección 4.2) y como éstos se conforman. Esta actividad puede ser un proceso relativo o puntual, dependiendo de la información que se tenga de las conexiones de red y de la utilidad que proveen a la organización.

El sistema de inicialización del SDTRL esta dividido en dos bloques principales. El primer bloque precisa la parte de configuración del sistema que es independiente de los datos que el administrador aporta. El segundo depende directamente de la información

que origina el administrador, como puede ser las configuraciones de las redes de cómputo, hasta la definición de los diferentes nodos de red. La Figura 5.2 muestra el diagrama de flujo de los diferentes procedimientos que realiza el sistema de inicialización.

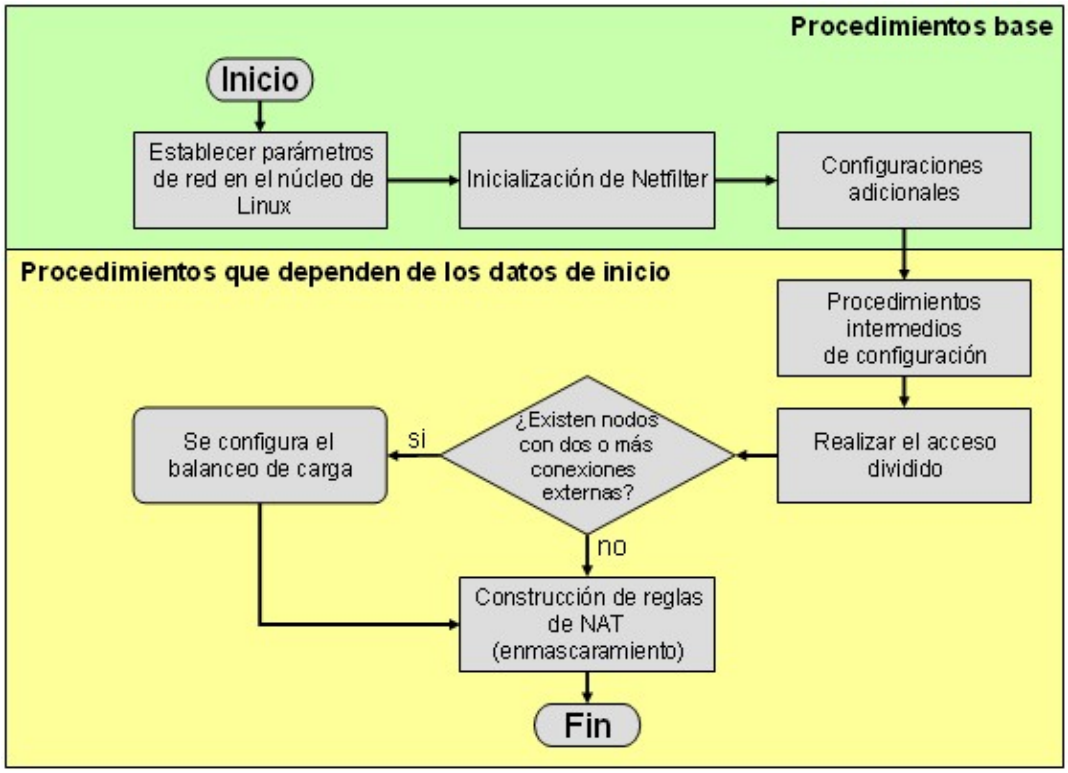

Figura 5.2: Diagrama de flujos del sistema de inicialización.

El sistema de inicialización se conforma de siete procesos. Los primeros tres son estáticos y preparatorios para los otros procedimientos. Los cuatro restantes toman información del sistema y de la ingresada por el administrador, para desempeñar sus funciones. El proceso de balanceo de carga puede no presentarse, si se da el caso de que no existan nodos compuestos (conformados por dos o más conexiones de red).

5.2.1. Procedimientos base

El bloque de procedimientos base mostrado en la Figura 5.2 esta compuesto de tres procesos, que tienen como función realizar la configuración base del sistema operativo.

El primero proceso establece los parámetros de red en el núcleo de Linux, como son: activar la opción de "FORWARD"que permite reenviar paquetes, activar la opción de "SYNC COOKIES"que protege ante posibles ataques DOS de bomba SYNC, y activar

la opción de protección de mensajes ICMP que también de protege de posibles ataques DOS de bomba ICMP. La opción de "FORWARD"es crucial, ya que el SDTRL tiene que reenviar el tráfico a más de una interfaz de red. Estos parámetros de red no son los únicos que requiere la organización, pero son los de mayor importancia, aunque el SDTRL permite agregar más opciones.

El segundo proceso inicializa el modulo Netfilter. Éste elimina las reglas de todas las cadenas de cada tabla (filter, nat y mangle), como todas las cadenas vacías, dejando solo las cadenas por defecto. Al final, precisa las políticas de omisión para el filtrado de tipo ACCEPT [28].

El tercer proceso realiza configuraciones adicionales, como el establecimiento de un cortafuego sencillo tipo stateful que analiza cada paquete de salida (solicitud), al igual que los paquetes de entrada (respuesta). Si ambos no mantienen una coherencia, los descarta. Es decir, descarta automáticamente todo aquello que no es razonable en un contexto determinado a fin de evitar ataques desde el exterior. Otra de las configuraciones que se realiza, es activar el dispositivo intermedio de encolamiento (DIE), que permite establecer las reglas del EBP.

Estas y otras configuraciones se encuentran en un archivo script ubicado en */etc/sdtrc /iniciaSDTRC.sh*, que se ejecuta como primera actividad del SDTRL. El script permite que el administrador pueda agregar nuevas y propias configuraciones de acuerdo a las necesidades que presente.

5.2.2. Procedimientos intermedios de configuración

Este bloque tiene un conjunto de actividades que realizan tareas tan variadas, pero fundamentales en el quehacer del sistema. La Figura 5.3 describe los procedimientos involucrados, que prepara al sistema para los procesos de acceso dividido y balanceo de carga.

El procedimiento de limpiar la tabla de ruteo principal se origina, debido a que el sistema Linux al inicializarse establece una tabla de ruteo principal, que no necesariamente es la adecuada para el SDTRL. Además, en este procedimiento se realiza la activación y desactivación de las diferentes interfaces de red a las que hace y no hace referencia el SDTRL.

El procedimiento para iniciar la tabla de reglas de las políticas de ruteo considera la posibilidad de reiniciar el SDTRL, sin necesidad de volver a reiniciar el sistema operativo. Por ello, es necesario eliminar cada una de las reglas hechas en secciones anteriores del

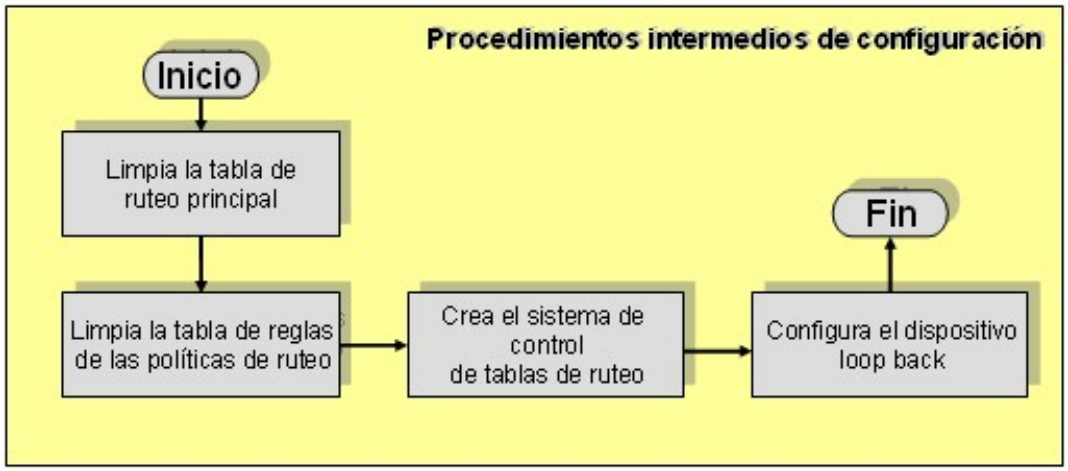

Figura 5.3: Diagrama de flujo de los procesos intermedios de configuración.

sistema y dejar solo las reglas de omisión.

La creación del sistema de control de tablas de ruteo, responde a la necesidad de tener un mecanismo que permita indicarle al SDTRL que tabla de ruteo esta disponible. Además, provee la manera de identificar una tabla de ruteo por un nombre convencional, y no como lo hace el núcleo del sistema (por un valor numérico). Este sistema de control se encarga de llevar el registro de las tablas, y de iniciar cada una de ellas.

Para llevar el registro de las tablas de ruteo, se debe tener en mente que éstas ya existen, y solo se hace uso de ellas. El problema radica en la asignación automática de una tabla de ruteo que no esta siendo empleada en algún momento. Para lograr esto se ejecuta el Algoritmo 5.1.

El último procedimiento en este bloque es relacionado con loop-back, el cual es un dispositivo que proporciona el núcleo del sistema para conectarse a la misma máquina sin pasar por la tarjeta de red. Este dispositivo no es indispensable tenerlo activado, pero es recomendable. La razón de activarlo, y declararlo en la tabla de ruteo principal, es debido a que fue eliminada en el proceso de limpiar la tabla de ruteo principal.

5.2.3. Acceso dividido para los nodos

El proceso de realizar el acceso dividido a cada una de las redes externas conectadas al SDTRL, se debe a que las direcciones de origen de los paquetes de entrada puede que no pertenezcan a las direcciones de red a las que esta conectado el SDTRL, y la salida de la respuesta deba ser diferente a la salida de omisión.

Algoritmo 5.1 Asignación de una tabla de ruteo disponible

(**Entrada** = Nombre de la tabla de ruteo)

1. **Si** no existe el nombre dentro del registro de tablas de ruteo utilizadas entonces:
   - *a*) Se determina que tabla de ruteo no ha sido utilizada todavía, dentro del rango de 1 a 252.
   - *b*) Se toma la tabla ruteo disponible con el identificador menor.
   - *c*) Inicia dicha tabla de ruteo, eliminando las rutas que pueda tener, y se marca como tabla no disponible.
2. **en caso contrario**
   - *a*) Se concluye el proceso.
3. Agrega el nombre e identificador en el registro de tablas de ruteo utilizadas.

En una configuración básica de red, la tabla de ruteo principal presenta el problema de enrutamiento, cuando dos o más conexiones de red permiten que un paquete llegue a un mismo destino (por ejemplo dos proveedores de Internet). Cuando se realiza una petición probablemente no haya inconvenientes, excepto si se desea enviar un flujo de paquetes hacia una conexión particular que no sea la salida por omisión. El problema sucede cuando una petición ingresa al sistema y la respuesta no sale por el lugar por donde entró.

Para resolver este problema en Linux, cada conexión de red utiliza una tabla de ruteo en el cual se especifica:

- Una ruta hacia el ruteador de la red externa.
- Una ruta por omisión para todo el tráfico a través del ruteador de la red externa.

Lo anterior hace ver a cada una de las tablas de ruteo empleadas, como si existiera únicamente un proveedor de servicios de red. Después se configura la tabla de ruteo principal con todas las rutas de las redes externas involucras, la cual especifican hacia donde mandar los paquetes en función de la dirección destino, junto con una salida por omisión.

Algoritmo 5.2 Procedimiento general para el acceso dividido

1. Se definen las tablas de ruteo de cada interfaz de red externa (Algoritmo 5.1).
2. Se define la tabla de ruteo principal con todas las rutas hacia las redes externas registradas en el SDTRL. La ruta por omisión corresponde al último nodo de red referido.
3. **Para** cada una de las interfaces de externas, se realiza:
   a) Se optime el valor de prioridad máximo disponible entre 1 y 200, dentro de las políticas de ruteo.
   b) Se declara la regla de política de enrutamiento, teniendo como condición la dirección origen de la interfaz de red y la prioridad obtenida anteriormente. Apuntando a la tabla de ruteo que le corresponde a la interfaz de red.
4. La regla de política de enrutamiento que hace referencia a la tabla ruteo principal tiene la prioridad más baja (32766).

Inmediatamente después, se crean las reglas de políticas de enrutamiento para cada una de las redes externas, las cuales apuntarán a las tablas de ruteo que le corresponden, creadas anteriormente. Cada regla esta en función de la dirección origen, y no la dirección destino como normalmente se selecciona una ruta. Estas reglas son de mayor prioridad que la regla que apunta a la tabla ruteo principal. Entonces, al momento de hacer una elección de una ruta de un paquete, se revisan primero las reglas de políticas de enrutamiento que hacen referencia a las redes externas y al final a la de la tabla de ruteo principal. Todo lo anterior se describe en el Algoritmo 5.2.

El acceso dividido no se realiza al nodo de la red local, dado que ésta experimenta un enmascaramiento, garantizando que los paquetes poseen la dirección de origen de algún miembro a la red. Esto permite que funcione de manera jerárquica como se muestra en la Figura 5.4.

5.2.4. Construcción de reglas de SNAT

Existen dos maneras de realizar el enmascaramiento en Linux con la ayuda de la herramienta iptables: dinámica y explícita o directa. La manera dinámica se utiliza cuando no se conoce con exactitud la dirección IP asignada por el proveedor de servicios de red en

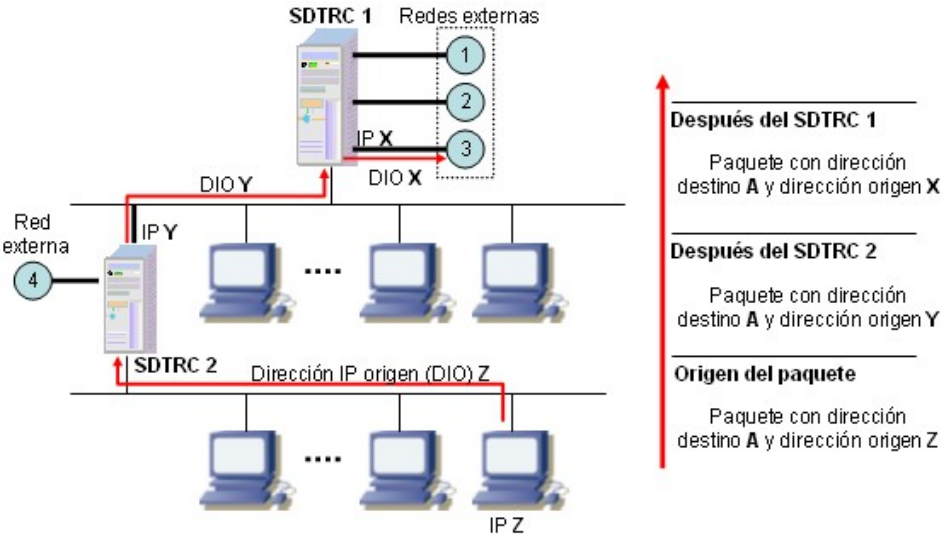

Figura 5.4: Modelo jerárquico del SDTRL.

la máquina cliente en donde se realiza el enmascaramiento. La manera explícita o directa es aquella que conoce inicialmente todos los datos de configuración necesarios y tiene un comportamiento invariable. Sin embargo, de las dos formas se realiza la traducción de direcciones por dirección fuente (SNAT). Excepto que, la primera realiza trabajos adicionales para averiguar la dirección IP que debe tomar para el enmascaramiento. Por ello, el SDTRL utiliza la segunda opción.

En las declaraciones de las reglas que efectúa el SDTRL a Linux para el enmascaramiento, se toma como base la información ingresada por el administrador al inicio de la configuración. Dicha información esta guardada en el archivo *linksSDTRC.cfg*, que contiene las interfaces de red con que cuenta el sistema y el tipo al que pertenecen (tipo externo e interno).

La regla de iptables que ejecuta el SDTRL para lograr el enmascaramiento tiene la siguiente forma:

`iptables` -t nat -A POSTROUTING -s **IPO** -o **DRE** -j SNAT –to **IPE**

Donde:

- **IPO** Las IP´s permitidas, de la red común de trabajo, para el SNAT.
- **DRE** El identificador de la interfaz de la red externa por donde sale el tráfico.

- **IPE** La IP de la interfaz de la red externa de salida.

Esta regla se repite dependiendo del número de conexiones de red externas que posee el sistema. El SDTRL utiliza siempre la misma IPO para todas las reglas, ya que hace referencia a todas las direcciones de la red común de trabajo. En tanto que, **DRE** y **IPE** siempre pertenecen a una misma conexión de red externa.

El trato que recibe el tráfico de paquetes dentro del proceso de enmascaramiento se describe de la siguiente manera. Después de que el SDTRL haya fijado una ruta, el paquete será enviado al sistema NAT, donde se emparejara con alguna de las reglas de la cadena POSTROUTING de la tabla NAT de iptables. Esto sucede solo para el primer paquete. Luego entonces, el proceso es mucho más fácil, ya no importa si se realizó el enmascaramiento de manera dinámica (MASQUERADE) o explícita (SNAT), pues en este momento toda la información necesaria es ya conocida. Simplemente, la dirección IP origen del paquete será sustituida por la dirección IP local origen de la interfaz de salida, y el paquete seguirá el mismo camino de sus predecesores. Este mismo procedimiento es repetido para todos los paquetes adicionales, porque la información de la conexión es guardada durante un cierto tiempo.

Finalmente, para el retorno de los paquetes no hay ningún problema, pues, el núcleo Linux mantiene un registro que reconoce estos paquetes y sabe como tratarlos para enviarlos por el camino adecuado.

5.2.5. Balanceo de carga

Para lograr características más avanzadas sobre el balanceo de carga es recomendable recompilar el núcleo con el parche de Julian Anastov [19], el cual, tiene un mejor desempeño al ya definido. Christoph Simon especifica en [30] los beneficios que este ofrece:

> *Se puede hacer uso de la opción "proto static"para la definición de rutas dentro de las tablas de ruteo, permitiendo que una conexión sobreviva aun sí el dispositivo deja de funcionar. Si una línea falla, todos los atributos del dispositivo permanecen, cuando esta vuelva a funcionar, el sistema no tendrá ningún problema en volverla a utilizar. También, esta opción permite trabajar como un sistema de redundancia, si alguna ruta deja de funcionar, automáticamente es descartada por el núcleo, en un periodo de 60 segundos (por defecto) y vuelve a intentar obtener una nueva ruta.*

> *El núcleo de Linux no tiene forma de descubrir el estado que guarda una interfaz de red. La única manera de lograr esto, es intentando utilizar la interfaz. Esto resuelve el problema cuando una interfaz deja de trabajar, pero cuando vuelve a estar activa, el núcleo ya no intenta utilizarla. La manera de poder indicarle al núcleo que dicha interfaz ya esta activa nuevamente es mediante una prueba directa a dicha salida. Esto se conoce como "detección de puerta de enlace de entrada muerta"(Dead Gateway Detection) según Julian Anastov.*

Para el SDTRL, establecer un nodo compuesto con dos o más conexiones a redes externas, es una tarea que se complementa con el del acceso dividido. Es decir, las declaraciones de las reglas y tablas generadas por el acceso dividido, son complementadas con las tablas de ruteo que contienen las rutas de múltiples caminos, junto con otra tabla de uso exclusivo del SDTRL. Esto se describe mejor en el Algoritmo 5.3.

Cuando llega un paquete al SDTRL, de la red local privada en dirección a las redes externas, el sistema de ruteo decide qué camino ha de tomar mediante el algoritmo de políticas de enrutamiento. Suponiendo que este paquete es enviado a una ruta de múltiples caminos, e inicia una nueva conexión de comunicación, que implica que no tiene registro alguno dentro del caché del sistema. El núcleo de Linux decide por cual camino mandar al paquete, de los múltiples que dispone el nodo para establece el balanceo de carga.

Al igual que en el caso del enmascaramiento, el balanceo de carga describe que las acciones iniciales solo ocurren una solo vez, y esto pasa, cuando se establece una conexión. Esta variación resuelve el problema de consistencia en la salida, es decir, asegura que los paquetes de una comunicación se encaminen por la misma interfaz de red.

Lo anterior se describe de la siguiente manera. Cuando se manda más flujo de paquetes de una conexión ya establecida, sucede que en el sistema de enrutamiento del SDTRL, ya se conoce la dirección IP origen de los paquetes. Esto es gracias al parche de Julian Anastov que revisa mucho antes si hay información en el sistema NAT de dicha conexión. Con ello se logra que los paquetes ya no pasen por la tabla de ruteo que contiene la ruta de múltiples caminos, pues no se podría asegurar en conseguir la misma salida que antes. Para garantizar que el flujo de una conexión tenga salida por un solo lugar, los paquetes consecuentes se emparejan con las reglas de enrutamiento con base en dirección IP origen, apuntando a la salida inicialmente definida.

Algoritmo 5.3 Establecimiento de los nodo que realizan el balanceo de carga

(**Entrada** = Número de nodos existentes, con las interfaces que los conforman.)
ID_TABLA: Tabla asignada para el nodo compuesto
GWE_X: Puerta de enlace de la red externa X que integra al nodo compuesto
IFE_X: Interfaz de red externa X que pertenece al nodo compuesto

1. **Si** existen nodos compuestos en el SDTRL, declarados por el administrador entonces:
   - *a*) Se declara la tabla llamada usefulOnlyForSDTRC, que contiene cada una de las rutas de las redes externas que pertenecen al sistema, y es la segunda tabla que es revisada por todo el tráfico.
   - *b*) **Mientras** existan nodos compuestos para declarar su tabla de ruteo, hacer:
     1) La asignación de una tabla de ruteo disponible para el nodo compuesto. Algoritmo 5.1
     2) Se ejecuta la instrucción que define la ruta de múltiples caminos en la tabla asignada en la acción anterior, como se especifica a continuación:

        `ip` route add default table **ID_TABLA** proto static
        nexthop via **GWE_1** dev **IFE_1**
        nexthop via **GWE_2** dev **IFE_2**
        :
        nexthop via **GWE_N** dev **IFE_N**

2. Se realiza el acceso divido, definido en el Algoritmo 5.2

Algoritmo 5.4 Detección de puerta de enlace muerta

GW_X: La dirección IP de la puerta de enlace de la red externa **X**

1. **Repetir** las siguientes acciones periódicamente en intervalos de 60 segundos.
   - *a*) **Mientras**, se realizan pruebas de estado a las puertas de enlace de las redes externas que pertenecen a los nodos compuestos.
     1) Se ejecuta el comando ping para determinar si es posible alcanzar a un sistema intermedio **GW_X** destino.

        ping -w 3 -c 1 **GW_X** > /dev/null 2>&1
     2) **Si** el GW_X está disponible y funcionando, entonces:
        - *a'* **GW_X** responderá a al ping y se obtiene una respuesta, junto con su dirección MAC.
     3) en caso contrario
        - *a'* El SDTRL no obtiene la dirección MAC, e indicará que no está activa la puerta de enlace.

Para lograr que el SDTRL sea capaz de volver ha utilizar una conexión de red que se haya recuperado de un error o simplemente la interfaz vuelve a funcionar. Es necesario tener un mecanismo que le indique al sistema que dicha interfaz de red esta de nuevo en uso. Para ello se toma la idea presentada en [30]. El demonio que utiliza el SDTRL esta basado en el Algoritmo 5.4, se crea y ejecuta automáticamente al momento que este se inicializa. Su función principal es averiguar periódicamente el estado (activo o muerto) que guardan las puertas de enlace de las redes externas pertenecientes a las conexiones de red que integran los nodos de múltiples caminos.

5.3. Sistema de asignación de recursos particulares

El sistema de distribución y asignación de ancho de banda para el tráfico particular están estrechamente relacionados. Por esta razón, las reglas de enrutamiento con base en políticas (EBP), del SDTRL, permiten asignar recursos a ese mismo tráfico particular.

Para el SDTRL, la Figura 5.5 muestra la llegada de un paquete, proveniente de la red local, que es sometido a un control de sanidad y enviado posteriormente al gancho

NF_IP_PRE_ROUTING (PRE). En este punto, la tabla mangle de Iptables empareja al paquete con alguna regla, provocando que éste sea marcado con algún valor numérico. Existe la posibilidad de ingresar a la Qdisc del DIE a través de esté gancho [29]. El SDTRL utiliza el DIE para el control de tráfico particular de entrada.

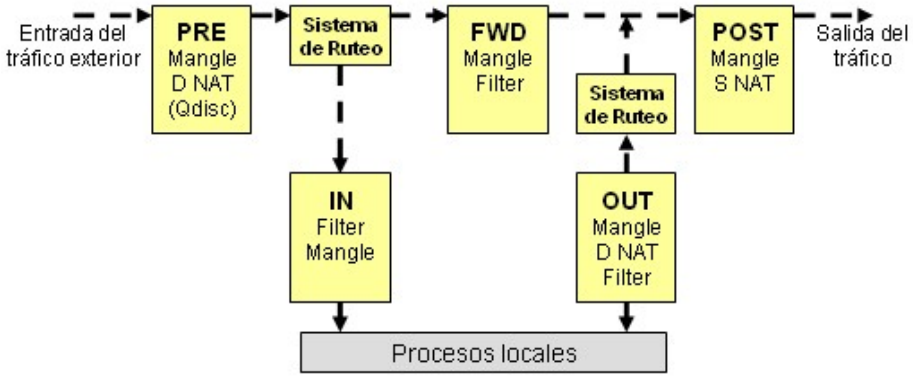

Figura 5.5: Diagrama que muestra el recorrido que exhibe los paquetes ipv4 a través de sistema, según la arquitectura Netfilter.

Después del proceso anterior, el paquete ingresa al sistema de ruteo donde se decide si va destinado a una interfaz de red o a un proceso local. La labor se realiza por medio de las reglas de políticas de enrutamiento, las cuales son revisadas sucesivamente hasta que una de estas reglas se empareje con el paquete y decida a qué tabla de ruteo se envía, y así finalmente dirigirlo al lugar que le corresponde.

La decisión del sistema de ruteo toma uno varios caminos. Si el paquete esta dirigido a una interfaz de red, se pasará al gancho NF_IP_FORWARD (FWD de la Figura 5.5), y en seguida desfila hacia el gancho NF_IP_POST_ROUTING (POST). Pero si, el paquete es dirigido a un proceso local se pasará al gancho NF_IP_LOCAL_IN (IN).

El diseño del sistema de asignación de recursos particulares se describe en la Figura 5.6*a*. Ésta muestra que el sistema se conforma de tres actividades principales. La primera es el marcado de paquetes que tiene como firme propósito poder relacionar una clase de tráfico con una clase de la Qdisc del DIE y una regla de las políticas de ruteo. La siguiente actividad es el proceso de control de tráfico que se realiza en la disciplina de cola del DIE. La tercera actividad la ejecuta el sistema de enrutamiento.

Una cuestión muy importante en el SDTRL es la unión de los sistemas de distribución y de asignación de recursos del tráfico particular, donde el vínculo de integración es el marcado de paquetes. El inciso *b* la Figura 5.6 hace ver la necesidad de tener un procedimiento de administración central de marcación, que determina y proporciona el

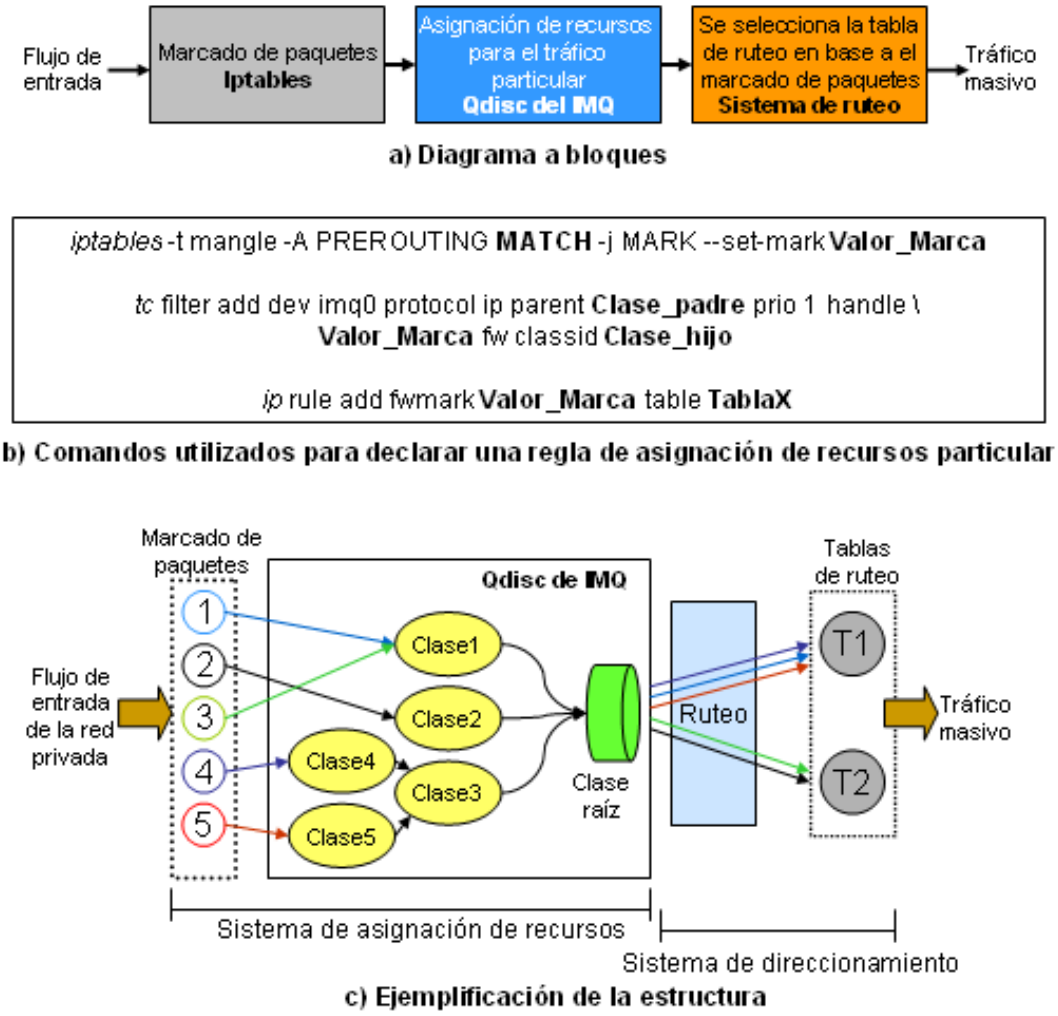

Figura 5.6: Diseño del sistema de enrutamiento en base a políticas.

valor numérico (**Valor_Marca**) disponible para ser usado por los sistemas sin temor a propiciar una colisión. Es decir, una clase de tráfico marcada (definida por el **MATCH**) podrá tener una asignación de recursos (dados por **Clase_hijo**) y ser encaminado a una tabla de ruteo específica (**TablaX**), esto de manera exclusiva, cuando se tiene asignado un único valor numérico para dicha clase.

La descripción funcional de la Figura 5.6 inciso *c* establece que el flujo de paquetes entrante al SDTRL proveniente de la red local de uso común, pasa por la etapa de definición de clases de tráfico particular, mediante el proceso de marcado de paquetes. Cada clase de tráfico se marca con un valor único. Inmediatamente después, las clases de tráfico se dirigen a la cola del dispositivo intermedio de encolamiento (DIE). Allí,

cada clase de tráfico particular es asociada a una clase de la Qdisc del DIE. Puede darse el caso que dos o más clases de tráfico particular pertenezcan a una misma clase de la Qdisc. El principal objetivo de pasar por el DIE, es lograr asignar un ancho de banda mínimo garantizado para cada clase de tráfico particular y manejar prioridades entre las diversas clases. Finalmente, las clases de tráfico se mandan a diferentes tablas de ruteo, dependiendo del valor marcado de los paquetes y el emparejamiento con las políticas de enrutamiento, en donde son encaminadas hacia una interfaz de red.

De lo anterior se concluyen un par de cosas. El tráfico particular se define y comporta gracias a las reglas de enrutamiento en base a políticas (EBP), donde, estas reglas tiene un secuencial de prioridad por orden de definición. Por ejemplo, una primera regla fue definida muy específicamente para una clase de tráfico. Posteriormente se define otra regla en donde establece una especificación más general que abarca la clase de tráfico anterior y otras. El SDTRL tomará como regla para describir el comportamiento de una clase, la última que fue definida y que empareje con ella.

5.4. Sistema de asignación de recursos masivo

El sistema de asignación de recursos masivo esta enfocado a controlar el tráfico de cada una de las salidas de las redes externas. La forma de conseguirlo al igual que en el caso del tráfico particular, es utilizando la Qdisc HTB, pero en esté caso para cada una de las interfaces de red.

Las clases de tráfico que maneja el sistema de asignación de recursos masivo son independientes, ya que, se considera que no hay flujos de paquetes que circulan es este bloque que pertenezcan a una clase y a otra al mismo tiempo. Simplemente, un flujo de paquetes corresponde única y exclusivamente a una clase de tráfico. Por ello, el sistema de asignación de recursos masivo no debería tener en cuenta el orden de ingreso de las reglas, por el simple hecho de ser excluyente el tráfico. Por otra parte, se realiza la revisión de las reglas de manera secuencial, donde la primera que empareje, es la que se toma. De este modo, el administrador deberá tener cuidado de no definir reglas que hagan referencia una clase de tráfico ya definida.

El SDTRL contempla dos acciones que se deben de considerar a la hora de definir el control de tráfico:

a) La primera acción consiste en definir la estructura de clases, las cuales tienen asignado un ancho de banda dentro de la interfaz de red. El SDTRL requiere de la existencia de

clases de tráfico masivo que divide y reduce el ancho de banda de la interfaz de red, como se muestran en los ejemplos de los incisos *a* y *b* de la Figura 5.7. Para lograr esto, cada una de las disciplinas de cola con clase de cada interfaz de red del SDTRL, utiliza una estructura de clases de 3 niveles (Figura 5.7 inciso *c*).

La manera de implementar lo anterior, es utilizando una clase raíz que define el ancho de banda máximo que puede tener una interfaz de red. De dicha clase, se desprenden subclases, donde se reparten el ancho de banda como si fuera un pastel de cumpleaños, salvo que las rebanadas pueden ser de tamaños diferentes de acuerdo a su necesidad. El tercer nivel especifica las Qdisc's de salida que pueden tener cada una de las subclases. El SDTRL provee la utilización de dos Qdisc's de salida, la FIFO y la Stochastic Fairness Queueing (SFQ), las cuales serán asignadas dependiendo de la prioridad de las subclases. Con la sustitución de FIFO por SFQ se obtiene equidad del medio cuando la salida realmente este llena. Si una subclase tiene una prioridad menor a dos se deja la FIFO, en caso contrario se le asignará la SFQ.

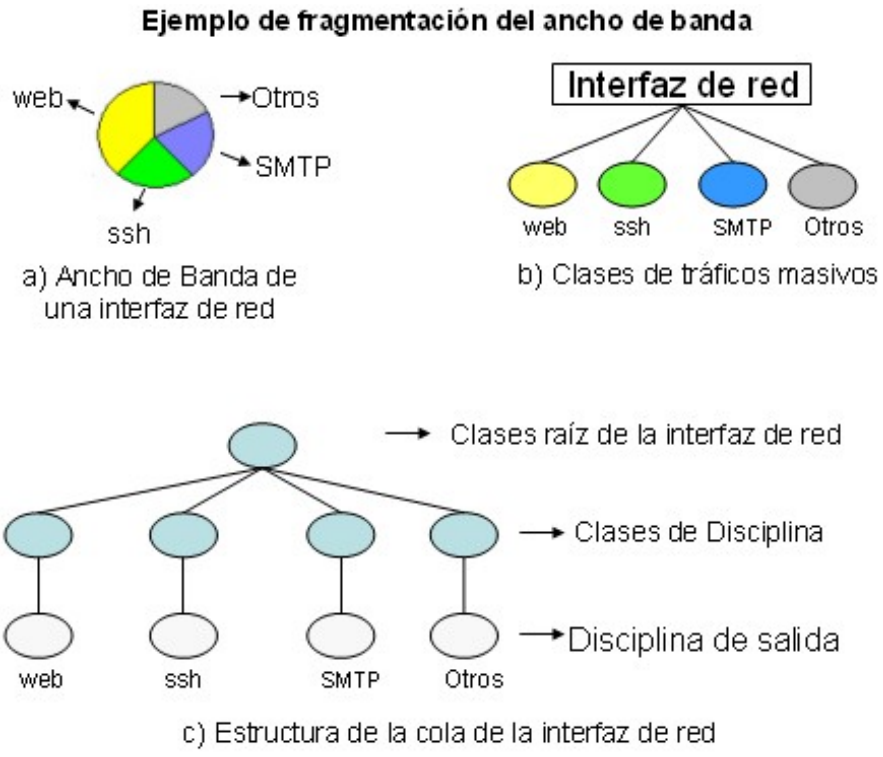

Figura 5.7: Fragmentación de ancho de banda de una interfaz de red con clases de trafico excluyentes.

b) La segunda y última acción consiste en asociar una clase de tráfico a una de las clases dentro de la estructura de la Qdisc, de la interfaz de red externa. Esta labor se realiza explícitamente por el administrador. No obstante, se tiene como objetivo reducir la sintaxis técnica provista por la herramienta tc. Para ello, el SDTRL necesita tres argumentos que permiten desempeñar esta acción, ya que simplemente ejecuta la siguiente instrucción:

tc filter add dev **IFE** protocol ip parent 1:0 prio 1 u32 **MATCH** flowid **ID_CLASE**

Donde:

- **IFE** La interfaz de la red externa por donde sale el tráfico.
- **MATCH** Las características de los paquetes de una clase de tráfico masivo.
- **ID_CLASE** El identificador de la clase dentro de la estructura de la Qdisc de **IFE**.

El contexto funcional de este sistema se describe cuando el flujo de paquetes proveniente del sistema de enrutamiento en dirección a las interfaces de red externas, pasa por un paso intermedio; la cola de la interfaz de red. La Qdisc se encarga de dividir el flujo de paquetes en nuevas clases tráfico, que no tienen relación alguna con las clases de tráfico particular.

El procedimiento de filar y desenfilar paquetes dentro de una Qdisc con clase, necesita realizar una clasificación para determina que hacer con un paquete. El encolamiento hace uso de los filtros asociados a la Qdisc, que devuelve una decisión, utilizada para encolar el paquete en una de las clases. En el sistema de asignación de recursos masivo, la Qdisc HTB cuenta con una clase raíz y con n-cantidad de subclases, donde se encolan los paquetes en las disciplinas de colas de salida asociadas a las subclases, como se muestra en la Figura 5.8. La clase raíz es la única que realiza el procedimiento de filtrado con la ayuda de U32. El número de subclases va depender de la suma de anchos de banda mínimo garantizados que pose cada subclase, donde la adición debe no superar el ancho de banda máximo de la interfaz de red o por lo definido por la clase raíz.

El procedimiento de desencolado de los paquetes, consiste en esperar una petición del núcleo del sistema, para extraer los paquetes de la cola y enviarlos al dispositivo de red. La Qdisc principal de la interfaz de red recibe una petición de desencolado, que se pasa a las clase raíz y este a su vez lo pasa a las subclases. Si se tuviera el caso, de tener una estructura diferente, con mayores niveles de clases como es la situación de la Qdisc

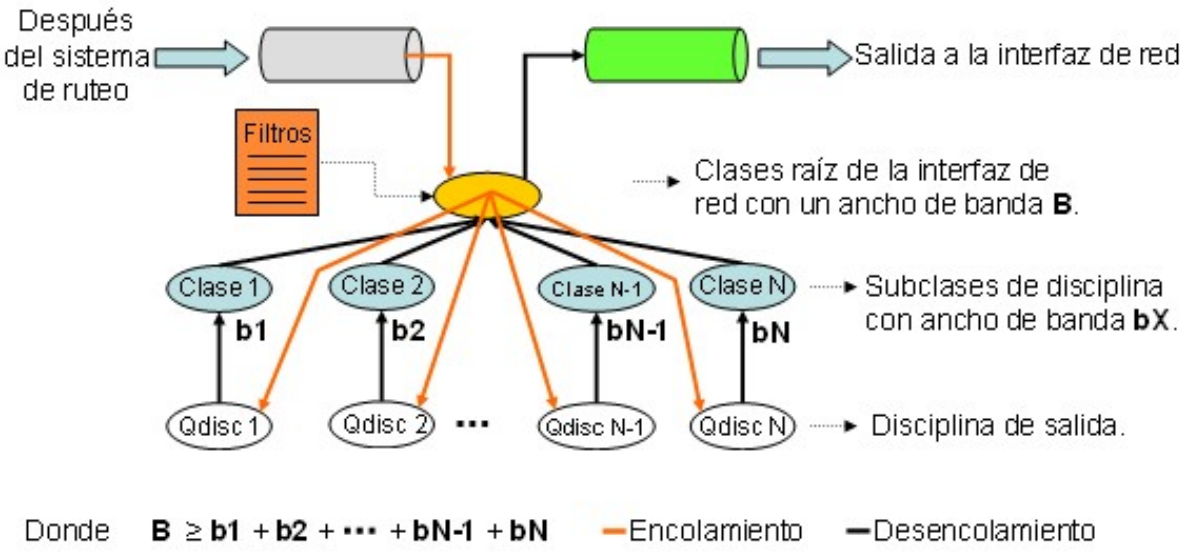

Figura 5.8: Trayectorias de los paquetes dentro Qdisc de la interfaz de red para el tráfico masivo.

del DIE para el tráfico particular. Las clases solicitarían paquetes a sus descendientes, éstas a su vez a sus hijos y así sucesivamente. Entonces, el núcleo requiere recorrer toda la estructura, para desencolar un paquete.

Realmente, el procedimiento de desencolar es el que efectúa el control de tráfico. Esto se debe a que las subclases se comunican únicamente con su clase paterna y nunca directamente al hardware de red; ello únicamente lo hace la Qdisc principal. Las clases realizan el desencolado de manera ascendente hacia la clase padre (como se muestra en la Figura 5.8), teniendo un rango máximo de transferencia de paquetes con base en su ancho de banda asignado. Se repite hasta llegar a la clase raíz. La manera de determinar que paquete se descola primero, se define por la prioridad que ostenta cada una de las clases de la estructura. Las prioridades se comparan en función del mismo nivel y del mismo parentesco, para definir el orden de salida.

5.5. Sistema de asignación de recursos de respuesta

La interfaz de red que conecta a la red privada local con el SDTRL presenta un comportamiento dual. Por una parte, puede comportarse como un nodo de red externa, de modo que puede utilizar el sistema de asignación de recursos masivo. Pero ello implicaría no tener la capacidad de hacer una diferenciación precisa y personalizada para asignar

anchos de banda y prioridades de entrada hacia los diferentes sistemas terminales conectados a la red local. Por esta razón, se implementó un sistema independiente y propio, que individualiza recursos para los miembros de la red.

El sistema de asignación de recursos de respuesta controla la Qdisc del dispositivo de red que comunica con la red local de la organización. Su estructura de clases en la Qdisc no esta definida, ya que el administrador lo hace con base en sus requerimientos, al igual que como sucede para el tráfico particular. Una peculiaridad de este sistema es la limitación que presenta cuando se quiere identificar tráficos bien conocidos de respuesta.

El sistema de asignación de recursos de respuesta esta conformado de dos actividades principales: la definición de clases de tráfico y la asignación de recursos. Esto se muestra en el diagrama de la Figura 5.9.

Figura 5.9: Diagrama de flujo de paquetes proveniente de las redes externas en dirección a las red local.

La actividad de definición de clases de tráfico esta referida al procedimiento de marcado de paquetes. Un paquete marcado con un valor específico pertenece a una clase de tráfico, y cada clase se asocia con un valor único de marca. La definición de las clases se realiza con reglas de iptables, en la cadena POSTROUTING de la tabla mangle, ubicada antes de que salgan del SDTRL, y después que se haya tomado la decisión de ruteo.

Por otra parte, la asignación de recursos para el tráfico de respuesta se efectúa en la estructura de clases de la Qdisc de la interfaz de red local, pues asigna el ancho de banda y prioridad a distintos flujos de tráficos. La manera de conectar las clases de la estructura con las clases de tráfico de respuesta es indicada en la Figura 5.9, mediante el marcado

de los paquetes. Cada clase dentro de la estructura de la Qdisc tiene relacionado un valor de marca, asegurado por el filtro `fw`. Se garantiza cada clase de tráfico de respuesta tenga un ancho de banda mínimo y una prioridad dentro del conjunto de clases.

5.6. Resumen

Este capítulo muestra claramente la importancia que tiene la inicialización en el SDTRL, ya que esta acción sirve como cimiento de las demás operaciones y de los atributos del sistema. En tanto que los subsistemas de los tres tipos de tráfico (particular, masivo y de respuesta) realizan la ejecución de algunas sentencias de comandos con las herramientas `iptables`, `ip` y `tc`, que trabajan de manera conjunta y sincronizada, debido a los módulos desarrollados para estos fines.

El SDTRL en Linux pretende ser un sistema capaz de poder implementar el enrutamiento en base a políticas, y facilitar un más el control de tráfico tanto de entrada como el de salida entre las redes externas y la privada. Además, el sistema procura evitar al máximo los conceptos técnicos innecesarios e intenta reducir las actividades de configuración frecuentes. Una cuestión importante en el balanceo de carga, es que la repartición del tráfico entre las conexiones de red no será perfecta, ya que se fundamenta en rutas y las rutas se guardan temporalmente en el caché.

Capítulo 6

La instrucción sdtrc

En este capítulo se delinea la forma de uso del SDTRL descrito en un escenario de prueba con únicamente dos conexiones de red hacia Internet y una red privada con aproximadamente 147 computadoras. El conjunto de pruebas se divide en 4 principales actividades: el tráfico particular, tráfico masivo, tráfico de respuesta y balanceo de carga. Finamente se describe los resultados obtenidos por el sistema con base en funcionalidad, manejo y rendimiento.

6.1. Descripción de operaciones para el sistema

En función de la propuesta del SDTRL referida en el capítulo anterior, el sistema tiene que cumplir un determinado número de actividades cronológicas que debe realizar para desempeñar las metas establecidas. Dicha actividades se describen en la tabla 6.1.

Del conjunto de actividades de la tabla 6.1, muestra de manera evidente que el administrador requiere de ingresar información útil al sistema, ejemplos de ello son las actividad 1, 3 y 4. Existen diversas formas de indicarle a los sistemas nuestras exigencias. La manera como se hace en el SDTRL es mediante un conjunto de comandos y reglas que permitan expresar las necesidades de la organización. Estos se describen en detalle mediante algunos ejemplos en las siguientes secciones.

6.2. Escenario utilizado

La primera versión del prototipo se ha implantado de acuerdo con el esquema de la figura 6.1, que es una simplificación de la red de la figura 4.3. La simplificación consiste en solo utilizar dos conexiones de red externas hacia Internet. Este escenario,

Tabla 6.1: Actividades generales del SDTRL.

Orden	**Actividad**	**Descripción**
1	Definición e identificación de nodos del SDTRL.	El administrador tiene que identificar y definir los nodos de las redes externas y de la red privada para indicar que conexiones de red las conforman.
2	Inicialización del sistema.	Efectúa el inicio del SDTRL realizando los siguientes sub-puntos.
2.1	Cargar datos al sistema.	A parte de la configuración de las interfaces de red de las conexiones externas y de la red privada, también se configuran los parámetros y se cargan los datos necesarios al sistema para auspiciar al SDTRL.
2.2	Acceso dividido.	Para cada una de las conexiones físicas de red se efectué el acceso dividido.
2.3	Balanceo de carga.	Si el sistema detecta nodos con dos o más conexiones físicas de red. El sistema se encarga de configurar, manipular y controlar todo lo relacionado con los nodos que realizan el balanceo de carga.
2.4	Enmascaramiento.	El SDTRL provee el mecanismo necesario para brindar los servicios de las redes externas a la red privada.
3	Cargar la lista reglas de tráfico masivo y de respuesta.	El administrador ingresa las reglas de asignación de recursos para el tráfico masivo y de respuesta con base en las políticas internas de la organización.
4	Cargar la lista reglas de tráfico particular.	El administrador ingresa las reglas de enrutamiento en base a políticas, la cuales están con base en las necesidades de los diferentes usuarios sin afectar las políticas internas de la organización.

aunque sencillo, permite validar la distribución y control de tráfico, y comprobar el funcionamiento y las características de la arquitectura del sistema. También este es el escenario que se da, cuando se quiere transmitir por un nodo que establece un balanceo de carga con las dos conexiones de red externa.

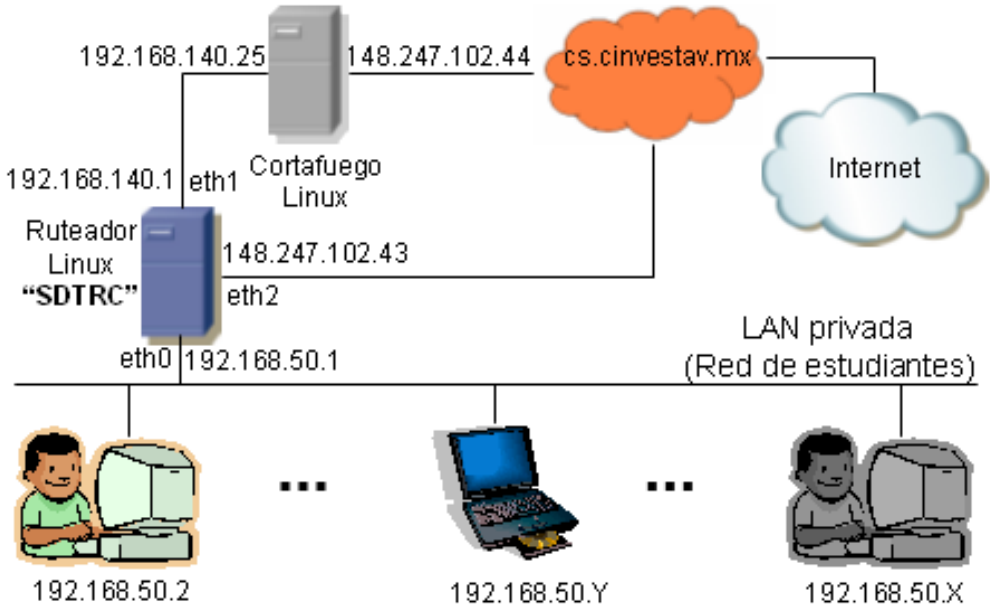

Figura 6.1: Topología de la red para pruebas y desarrollo del SDTRL.

La figura 6.1 indica que el SDTRL esta conectado a tres diferentes redes; por medio de interfaces de red Ethernet de 100 Mbps. La interfaz eth2 se conecta a la subred de la Sección de Computación (148.247.102.0/24) del CINVESTAV. Por su parte, la interfaz eth1 esta conectada directamente al cortafuego, la cual forman la red privada 192.168.140.0/24. La interfaz eth0 se conecta la red privada de los estudiantes que hacen uso de los servicios de la subred de la Sección de Computación y de la Internet. Esta red privada cuenta con cerca de 147 computadoras clientes, conectadas por medio de conmutadores y segmentada en tres redes privadas por sistemas que realizan NAT.

El ruteador Linux etiquetado con "SDTRL"de la Figura 6.1 se conecta a dos proveedores de servicios de red externos y una red privada que hace uso de los servicios de los dos enlaces. Para obtener las dos conexiones de red externas independientes se hace uso de la red privada (conectada a la interfaz eth1) que emula un proveedor de servicios de Internet diferente. Esto debido a que se tiene una salida hacia Internet, por medio del uso de la dirección pública que tiene el cortafuego para dar acceso a la subred 148.247.102.0/24 mediante el uso del enmascaramiento. Además, dicho sistema Linux

cuenta con la base tecnológica indicada en la sección 4.4 y con el prototipo del SDTRL ya compilado e instalado. Entonces, si los sistemas clientes hacen una petición de servicios externos a la red privada (192.168.50.0/24) estos son dirigidos al SDTRL. Ya dentro del sistema, se procede a realizar el trabajo de distribución y control de tráfico.

6.3. La instrucción sdtrc

En este apartado, se presenta la instrucción **sdtrc**, que es la herramienta para crear las reglas del SDTRL. El conjunto de reglas que se diseñó, esta pensado para facilitar el uso del sistema. De hecho, toma características de las tres herramientas involucradas (*iptables*, *ip* y *tc*), con objeto de evitar diseñar un formato distinto y poco conocido para los usuarios. Además, se evita al máximo los detalles técnicos de las tres herramientas implicadas.

El **sdtrc** permite configurar varias objetivos tales como: inicializar el SDTRL, crear tablas de ruteo, establecer trayectorias en la tabla de ruteo, creación de reglas de enrutamiento en base a políticas, reglas de control de tráfico para el tráfico masivo y de respuesta, y establecer la estructura de clases de la Qdisc para el control de tráfico. Cada una de estos objetivos, tiene definidas un conjunto de funciones que también son llamados comandos. Por tal razón, se toma la idea de tener una instrucción multi-objetivos, igual que las herramientas *ip* y *tc*.

sdtrc OBJETIVO COMANDO

Donde OBJETIVO = { **route** | **rule** | **table** | **class** | **nodo** | **start** }

La palabra COMANDO se refiere a las diferentes acciones que realiza cada uno de los objetivos dentro del SDTRL.

El objetivo **nodo** identifica y da nombre a los nodos de red para el SDTRL, ejecutando la siguiente sintaxis.

sdtrc nodo COMANDO

Donde COMANDO = { **add** | **del** } name NOMBRE_NODO dev ID_INTERF type { **ext** | **int** }

o

COMANDO = **ls** (visualiza el estado de los nodos)

La opción COMANDO del objetivo **nodo** tiene la posibilidad de ingresar, eliminar y mostrar los nodos existentes dentro del SDTRL. Ello, presenta tres valores de argumentos, el nombre propio o alias del nodo de red (NOMBRE_NODO), el identificador del dispositivo de red asociado a este nodo (ID_INTERF), y el tipo de nodo, que indica si es un nodo de red externa (**ext**) o el nodo de la red privada local (**int**).

El objetivo **table** inicializa y proporciona un sobrenombre a una de las 252 posibilidades de tablas de ruteo que Linux provee, de las cuales hace uso el SDTRL. Sin olvidar, que también permite dejar de utilizarlas. Este objetivo presenta la siguiente sintaxis.

sdtrc table COMANDO

Donde COMANDO = { **add** | **del** } NOMBRE_NODO

o

COMANDO = **ls** (visualiza las tablas de ruteo activas)

La manera de establecer trayectorias en las tablas de ruteo es tomado directamente de la herramienta ip con el objetivo route [24]. El propósito es utilizar una sola herramienta para transparentar la configuración del SDTRL.

sdtrc route COMANDO

Donde COMANDO es igual al utilizado en "ip route".

La manera de especificar la estructura de clases en las Qdisc's de los dispositivos de red involucrados en el SDTRL para el control tráfico es mediante el objetivo **class**. El COMANDO de este objetivo hace referencia a los tres tipos de tráficos. La sintaxis para referirse al tráfico particular (**tp**) y de respuesta (**tr**) es igual, por el simple hecho de que están basados en el trabajo conjunto de las herramientas *tc* e *iptables*. Lo cual no ocurre en el caso del tráfico masivo (**tm**), donde todo el trabajo de la estructuración de clases lo realiza completamente la herramienta tc. La sintaxis del objetivo **class** es la siguiente.

sdtrc class COMANDO

Para el tráfico particular y de respuesta.

COMANDO para establecer la clase raíz.

COMANDO =

add{ **tp** | **tr** } **root classid** ID_RAIZ **rate** RANGO **default** ID_CLASE

COMANDO para establecer las clases hijos.

COMANDO =

add{ **tp** | **tr** } **parent** ID_PADRE **rate** RANGO [**ceil** RANGO_MAX] [**burst** VALOR_RAFAGA] [**cburst** VALOR_RAFAGA] [**prio** VALOR_PRIO]

COMANDO para eliminar toda la estructura de clases.

COMANDO = **del** { **tp** | **tr** } **root**

Para el tráfico masivo.

COMANDO para establecer la clase raíz.

COMANDO =

add tm dev ID_INTERF **root rate** RANGO **default** ID_CLASE

COMANDO para establecer las clases hijos.

COMANDO =

add tm dev ID_INTERF **rate** RANGO [**ceil** RANGO_MAX] [**burst** VALOR_RAFAGA] [**cburst** VALOR_RAFAGA] [**prio** VALOR_PRIO]

COMANDO para eliminar toda la estructura de clases.

COMANDO = **del tm dev** ID_INTERF **root**

El COMANDO del objetivo **class** hace referencia a varios argumentos relacionados con el control de ancho de banda y prioridad. Donde ID_RAIZ es el valor de identificación numérico de la clase raíz. Por lo general, debe ser un valor positivo pequeño. RANGO es un valor en bits por segundo (bps) o bytes (b), que especifica el ancho de banda de una clase. ID_CLASE simplemente es el identificador numérico que hace referencia a la clase. Este valor debe ser mayor al valor de ID_RAIZ. ID_PADRE es el identificador

de la clase paterna de la cual se desprende la clase. El ID_INTERF es el nombre de la interfaz de red donde se aplica el control de tráfico.

Los argumentos de **ceil**, **burst**, **cburst** y **prio** no son siempre necesarios u obligatorios dentro del COMANDO. El argumento de **ceil** específica el ancho de banda máxima que una clase puede usar. El **ceil** por omisión toma el mismo valor de **rate**, el cual define el ancho de banda de una clase. Los argumentos de **burst** y **cburst** controlan la cantidad de datos que pueden ser enviados a la velocidad máxima en un tiempo muy corto sin tratar de servir otra clase. La particularidad de **cbusrt** es no exceder el rango del valor de **ceil**. Esto se describe mejor en [27]. El argumento de **prio** es un valor que determina el orden de atención de una clase de trafico. El valor de máxima prioridad es el valor cero, su predecesores los valores mayores a esté.

El objetivo **rule** establece las reglas en el SDTRL. Las reglas están conformadas de condiciones y acciones. La sintaxis utilizada para indicar las condiciones de emparejamiento (identificado como MATCH) entre paquetes y reglas es tomada de la herramienta Iptables, para el enrutamiento en base a políticas (EBP) y el sistema de asignación de recursos para el tráfico de respuesta. Para el sistema de asignación de recursos de tráfico masivo se emplea la sintaxis del selector utilizada en el filtro U32 dentro de las disciplinas de colas, identificado como MATCHtm (ver Apéndice A).

Las acciones que toman las reglas del EBP son la distribución de una clase de tráfico particular y la asignación de recursos para el mismo tráfico. Una de estas acciones al menos debe existir dentro de la regla. Como se describe ha continuación.

sdtrc rule COMANDO

Para el tráfico particular.

COMANDO = { **add** | **del** } **tp** MATCH **table** NOMBRE_TABLA

COMANDO = { **add** | **del** } **tp** MATCH **toclass** ID_CLASE

COMANDO = { **add** | **del** } **tp** MATCH **table** NOMBRE_TABLA **toclass** ID_CLASE

Para el tráfico masivo.

COMANDO = **add tm dev** ID_INTERF MATCHtm **toclass** ID_CLASE

COMANDO = **del tm dev** ID_INTERF

Para el tráfico de respuesta.

COMANDO = { **add** | **del** } **tr** MATCH **toclass** ID_CLASE

Visualiza el conjunto de reglas de los tres tipos de tráficos.

COMMAND = **ls tp**

COMMAND = **ls tm dev** ID_INTERF

COMMAND = **ls tr**

Las reglas para el tráfico masivo y de respuesta realizan la acción de dirigir el tráfico hacia las diversas clases, donde estas tiene asignado un ancho de banda mínimo garantizado y una prioridad.

Finalmente, el objetivo **start** se encarga de iniciar o reiniciar el SDTRL con los valores de omisión, además configura al sistema en función a los nodos de red que actualmente tiene registrados.

sdtrc start

6.4. Modo de operación del sdtrc

El sistema se instala en una plataforma que cumpla con todas las características necesarias para su funcionamiento correcto, descritas en la sección 4.4.1. En el caso inicial, cuando el **sdtrc** ha sido apenas compilado e instalado, no presenta ningún comportamiento de reenvío, distribución y control del tráfico, simplemente es una estación de trabajo conectado a las redes.

El trabajo del **sdtrc** se exhibe solo cuando se efectúe al menos el primero de los dos bloques generales: la configuración inicial y la asignación de recursos.

La parte de configuración inicial se presentará en el momento después de su instalación o cuando se tenga que modificar de nuevo su estructura en los nodos de red del sistema.

El segundo bloque de acción consiste en definir los tráficos masivos y de respuesta que circulan por el sistema. Además, establece el EBP para el tráfico particular.

6.4.1. Configuración inicial del SDTDL

La inicialización del STDRL tiene como propósito identificar los nodos de las redes externas y el nodo de la red privada, y con ello realizar la primera configuración básica. La identificación y valorización de las redes con la que cuenta la organización permitirá hacer una buena administración de los recursos más adelante.

El SDTRL identifica básicamente dos tipos de conexiones de red, la red común de trabajo y las redes externas que proveen los servicios. Existen dos posibilidades de ingresar esta información al sistema: por medio del objetivo **nodo** la instrucción **sdtrc** o por la edición del archivo /etc/sdtrc/linksSDTRC.cfg. Sea cual sea el caso, se tomará esta información para poder iniciar o reiniciar el **sdtrc**; dependiendo de la situación.

La Figura 6.2 refiere a la ejemplificación del escenario de la sección 6.2, donde se indica que el sistema inicialmente no contiene ninguna información sobre algún nodo de red. En seguida se ingresan los nodos identificados de acuerdo a la conveniencia de la organización o administrador. El orden en como se ingresan los nodos de red no tiene mayor importancia, salvo que el último en ingresar es el que se tomará como nodo de salida por omisión para el tráfico no clasificado.

```
[root@i2landa root]# sdtrc nodo ls
NOMBRE DEL NODO         INTERFACES DE RED               TIPO DE NODO
[root@i2landa root]# sdtrc nodo add name alumnos dev eth0 type int
[root@i2landa root]# sdtrc nodo add name Salida_sin_firewall dev eth2 type ext
[root@i2landa root]# sdtrc nodo add name Salida_con_firewall dev eth1 type ext
[root@i2landa root]# sdtrc nodo ls
NOMBRE DEL NODO         INTERFACES DE RED               TIPO DE NODO
alumnos                 eth0                            int
Salida_sin_firewall     eth2                            ext
Salida_con_firewall     eth1                            ext
```

Figura 6.2: Ingreso de los nodos de red al SDTRL.

El SDTRL se inicia en el instante en que se ejecuta la instrucción **sdtrc start**. En este momento, el sistema prepara, configura y establece lo necesario para dar cabida a la capacidad de distribución y control de tráfico. Ello se logra mediante el establecimiento de parámetros al sistema, el enmascaramiento y un acceso dividido para cada uno de los nodos externos, entre otras cosas (ver figura 6.3). El proceso de reiniciar el **sdtrc** implica, además de volver a ejecutar de las actividades anteriores, poner al sistema en un estado inicial, sin reglas, sin clases de tráfico o incluso depura cualquier condición que pueda meter alguna anomalía. La excepción a lo anterior es la información ingresada por

el objetivo **nodo**, la cual permanece intacta, debido a que ésta es meramente informativa y primordial para la inicialización.

```
[root@i2landa pruebas_CS]# sdtrc start
        [.] Kernel version : 2.4.20
- Estableciendo parametros del Kernel en /proc/sys/net/ipv4 -
- Iniciando NetFilter -
- Iniciando Firewall stateful -
- Activando la interface IMQ -

        CONFIGURANDO SDTRC
Inicializando la interface                                  <eth2>
Inicializando la interface                                  <eth1>
Inicializando la interface de la red local                  [eth0]
Configurando la interface para la red local.                [eth0]
Configurando la interface para las redes externas.          <eth2>
Configurando la interface para las redes externas.          <eth1>
NAT Configuration                                           [ OK ]
```

Figura 6.3: Inicialización del SDTRL con dos nodos de red externos.

Las primeras cuatro acciones que se ejecutan por el objetivo **start** son invocados por el script /etc/sdtrc/iniciaSDTRC.sh, el cual puede ser modificado para poder agregar nuevas funciones que el administrador considere pertinentes. Inmediatamente después de la ejecución del script, el sistema realiza una serie de acciones que van en función del los tipos de nodos, tales como la inicialización de las interfaces de red, hasta la configuración de NAT.

Para llevar a cabo los objetivos de inicialización, el SDTRL crea una tabla de enrutamiento para cada nodo de red registrado en el sistema. Estas tablas contienen las rutas de la red a la que pertenecen el nodo y una salida por omisión. Lo interesante de esto, es cuando alguna política de enrutamiento tiene como propósito dirigir cierta clase de tráfico hacia un nodo particular, lo único que se tiene que hacer es referenciar a su tabla de enrutamiento. La Figura 6.4 presenta la existencia de tablas de enrutamiento adicionales a la tabla main y las rutas que ellas contienen.

Balanceo de carga

Los nodos externo que se relacionan con dos o más interfaces de red permiten realizar un balanceo de carga para dividir los enlaces de comunicación entre sus diferentes salidas. El balanceo de carga es un caso muy especial para el SDTRL, pues tiene una iniciación distinta comparada con el caso donde todos los nodos tienen relacionado una sola interfaz de red. Asimismo, la forma de ingresar nodos de red al sistema permanece igual, salvo

```
[root@i2landa pruebas_CS]# sdtrc table ls
NOMBRE_TABLA            ID NUMERICO
Salida_sin_firewall     1
Salida_con_firewall     2
main                    254
[root@i2landa pruebas_CS]# sdtrc route ls
148.247.102.0/24 dev eth2  proto kernel  scope link  src 148.247.102.43
192.168.50.0/24 dev eth0  proto kernel  scope link  src 192.168.50.1
192.168.140.0/24 dev eth1  proto kernel  scope link  src 192.168.140.1
default via 192.168.140.25 dev eth1
[root@i2landa pruebas_CS]# sdtrc route ls table Salida_sin_firewall
148.247.102.0/24 dev eth2  scope link  src 148.247.102.43
default via 148.247.102.254 dev eth2
[root@i2landa pruebas_CS]# sdtrc route ls table Salida_con_firewall
192.168.140.0/24 dev eth1  scope link  src 192.168.140.1
default via 192.168.140.25 dev eth1
```

Figura 6.4: Tablas de enrutamiento creadas en la inicialización.

en el caso de querer tener un nodo con dos o más conexiones de red, solo se agregan interfaces a nodos ya existentes, como se ilustra en la figura 6.5. La presencia de al menos un nodo con dos o más interfaces generara la activación del demonio de detección de puertas de enlace muerto, identificado con el nombre de proceso pingDemo.sh el cual es ejecuta en background.

Si el proceso de inicialización del SDTRL detecta nodos que realicen balanceo de carga, provoca que se presenten tablas de enrutamiento adicionales en comparación al primer caso expuesto. Esto debido a que el SDTRL crea una tabla de enrutamiento por cada interfaz de red (que garantizan el acceso dividido) sumadas a las tablas encargadas de realizar el balanceo de carga. En la Figura 6.6 se observa una tabla de enrutamiento llamada usefulOnlyForSDTRC que solo tiene sentido y utilidad para el propio sistema; esto significa que no debe ser alterada por los usuarios.

Por otra parte, las Figuras 6.5 y 6.6 presentan una situación muy peculiar que merece ser mencionada. La existencia de solo dos conexiones de red externas implica que solo puede haber un nodo compuesto de red de tipo externo que balance la carga de tráfico. Al iniciar el SDTRL este nodo no provee una salida por omisión para la tabla de ruteo principal, provocando que el tráfico no clasificado carezca de una opción de salida hacia Internet. La única manera de lograr este cometido es definiendo explícitamente una política de enrutamiento que apunta a alguna de las salidas del sistema por medio tablas de enrutamiento. En la siguiente sección se describirá las diferentes posibilidades de uso de estas políticas.

```
[root@i2landa pruebas_CS]# sdtrc nodo add name alumnos dev eth0 type int
[root@i2landa pruebas_CS]# sdtrc nodo add name Salida dev eth1 type ext
[root@i2landa pruebas_CS]# sdtrc nodo add name Salida dev eth2 type ext
[root@i2landa pruebas_CS]# sdtrc nodo ls
NOMBRE DEL NODO         INTERFACES DE RED               TIPO DE NODO
alumnos                 eth0                            int
Salida                  eth1,eth2                       ext
[root@i2landa pruebas_CS]# sdtrc start
        [.] Kernel version : 2.4.20
- Estableciendo parametros del Kernel en /proc/sys/net/ipv4 -
- Iniciando NetFilter -
- Iniciando Firewall stateful -
- Activando la interface IMQ -

        CONFIGURANDO SDTRC
Inicializando la interface                                  >eth1<
Inicializando la interface                                  >eth2<
Estableciendo El BALANCEO DE CARGA                          [ OK ]
Inicializando la interface de la red local                  [eth0]
Configurando la interface para la red local.                [eth0]
Configurando la interface de Balanceo de carga.             >eth1<
Configurando la interface de Balanceo de carga.             >eth2<
Activating the dead gateway detection                       [ OK ]
NAT Configuration                                           [ OK ]
```

Figura 6.5: Inicialización del SDTRL con un nodo compuesto de dos conexiones de red.

6.4.2. Tráfico particular

El tráfico particular establece reglas que se refieren a flujos específicos relacionados con la red privada, el cual se les da un trato diferenciado. Las reglas de tráfico particular que se crean tienen un doble enfoque que especifica por un lado la asignación de ancho de banda y prioridad, por el otro, la distribución de tráfico hacia alguna salida provistas por nodos externos; este tipo de reglas son llamadas enrutamiento en base a políticas. Estas pueden ser tan específicas o tan generales dependiendo de las necesidades del administrador del sistema, y son las que más frecuentemente son modificadas para poder fijar recursos a los miembros de la red privada.

Distribución de tráfico

Después de inicializar el SDTRL, se procede a ingresar reglas para implantar las políticas de administración de la red. Una de esas reglas son las políticas de enrutamiento que se logran por medio de la definición y ejecución de reglas de tráfico particular que apuntan a tablas de enrutamiento, como se indica en la Figura 6.7. Dichas tablas son realmente

```
[root@i2landa pruebas_CS]# sdtrc table ls
NOMBRE_TABLA            ID NUMERICO
Salida                  1
usefulOnlyForSDTRC      2
Salida-eth1             3
Salida-eth2             4
main                    254
[root@i2landa pruebas_CS]# sdtrc route ls table Salida
default  proto static
        nexthop via 192.168.140.25  dev eth1 weight 1
        nexthop via 148.247.102.254  dev eth2 weight 1
[root@i2landa pruebas_CS]# sdtrc route ls table usefulOnlyForSDTRC
148.247.102.0/24 dev eth2  scope link  src 148.247.102.43
192.168.50.0/24 dev eth0  scope link  src 192.168.50.1
192.168.140.0/24 dev eth1  scope link  src 192.168.140.1
[root@i2landa pruebas_CS]# sdtrc route ls table Salida-eth1
default via 192.168.140.25 dev eth1  proto static  src 192.168.140.1
prohibit default  proto static  metric 1
[root@i2landa pruebas_CS]# sdtrc route ls table Salida-eth2
default via 148.247.102.254 dev eth2  proto static  src 148.247.102.43
prohibit default  proto static  metric 1
[root@i2landa pruebas_CS]# sdtrc route ls
148.247.102.0/24 dev eth2  proto kernel  scope link  src 148.247.102.43
192.168.50.0/24 dev eth0  proto kernel  scope link  src 192.168.50.1
192.168.140.0/24 dev eth1  proto kernel  scope link  src 192.168.140.1
```

Figura 6.6: Estructuras de las tablas de ruteo en presencia de un nodo de red compuesto.

las encargadas de dirigir el flujo de paquetes hacia una salida del sistema.

Tomando de nuevo un caso real, descrito por las figuras 6.5 y 6.6, es posible ejemplificar fácilmente el uso y manera de trabajar de la distribución de tráfico en el SDTRL. Con la existencia de un solo nodo de red externo que realiza balanceo de carga, todo hace indicar que existe una sola posibilidad de encaminar el tráfico. Por si fuera poco, este nodo es el encargado de decidir por que interfaz de red dirigir un enlace de comunicación con base en sus criterios. Las Figuras 6.7 y 6.8 abren la posibilidad de que esto no sea así del todo. Aun a pesar de que solo se tiene registrado un solo nodo de red externo, es posible hacer uso de las interfaces de red con las que él cuenta. Esto se logra aprovechando las tablas de enrutamiento creadas para dar acceso divido a cada conexión. Con estas tablas se tiene la posibilidad de dirigir un flujo de paquetes directamente a una salida de red.

Como se puede apreciar en la Figura 6.8, primero se ingresa una regla que permite dar una salida por omisión a todo el tráfico hacia una interfaz de red particular. Esto podría suceder también con cualquier tipo de tráfico más específico y dirigido a cualquier

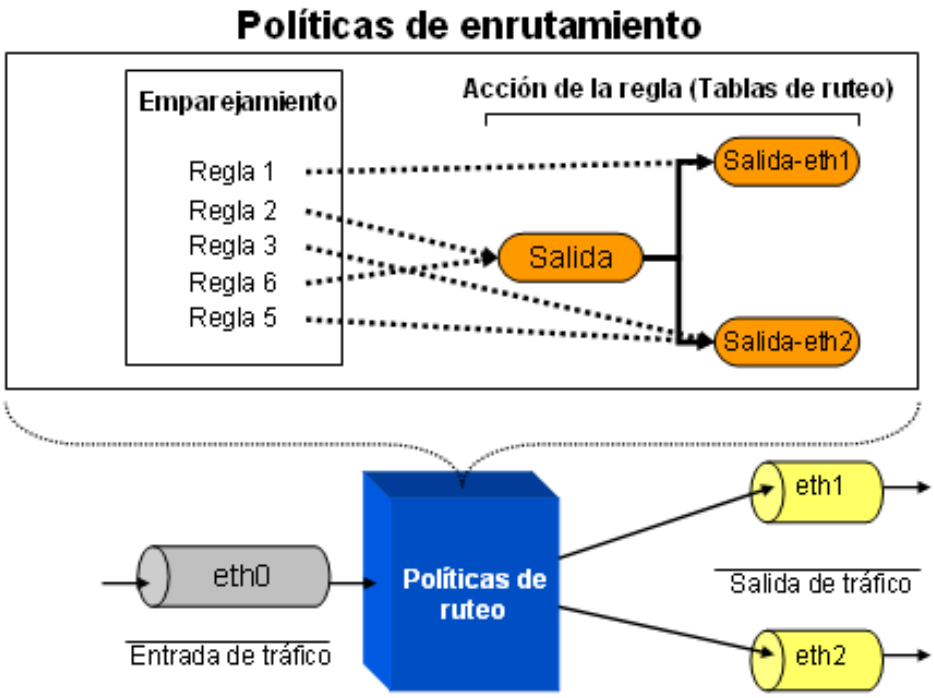

Figura 6.7: Ejemplificación de la forma de aplicar políticas de enrutamiento.

interfaz de red por medio de su tabla de ruteo. Lo interesante de esto, es que igual puede existir reglas que haga uso del nodo compuesto a un a pesar de que otros reglas estén haciendo uso de sus interfaces de red asociadas. En todo caso, el nodo compuesto evalúa el tráfico de cada uno de sus miembros, ya sea que lo aya generado él mismo o no, para determinar por que camino mandar un nuevo enlace de comunicación en función al criterio de equilibrar los niveles de trafico. Es decir, puede existir una interfaz de red miembro de un nodo compuesto completamente saturado por una regla que hace referencia directamente a está. Entonces el nodo compuesto de red decidirá mandar siempre su tráfico a los otros miembros no saturados. Esto sucederá hasta que la interfaz deje de estar saturado, para que vuelva hacer considerada.

Control de tráfico

La asignación de recursos por parte del conjunto de reglas de tráfico particular requiere un paso previo. Este evento es la estructuración de las clases de servicio[1] para subdividir el ancho de banda que dispone cada interfaz de red. Sobra mencionar que la forma

[1]Se define como clase de servicio a la forma de tratar a los paquetes en base un ancho de banda y prioridad.

```
[root:~]# sdtrc rule add tp -i eth0 table Salida-eth1
[root:~]# sdtrc rule add tp -i eth0 -s 192.168.50.71 -p tcp --dport 80 table Salida-eth2
[root:~]# sdtrc rule add tp -i eth0 -s 192.168.50.11 table Salida
[root:~]# sdtrc rule add tp -i eth0 -s 192.168.50.11 -p tcp --dport 22 table Salida-eth1
[root:~]# sdtrc rule add tp -i eth0 -s 192.168.50.101 table Salida
[root:~]# sdtrc rule add tp -i eth0 -s 192.168.50.102 table Salida
[root:~]# sdtrc rule add tp -i eth0 -s 192.168.50.103 table Salida
[root:~]# sdtrc rule add tp -i eth0 -s 192.168.50.104 table Salida
[root:~]# sdtrc rule add tp -i eth0 -s 192.168.50.106 table Salida
[root:~]# sdtrc rule add tp -i eth0 -s 192.168.50.107 table Salida
[root:~]# sdtrc rule add tp -i eth0 -s 192.168.50.110 table Salida
[root:~]# sdtrc rule add tp -i eth0 -s 192.168.50.111 table Salida
[root:~]# sdtrc rule ls tp
PROT OPT SOURCE            DESTINATION       DETAIL         TABLE          CLASS_ID
all  --  anywhere          anywhere          ----           Salida-eth1    ----
tcp  --  sigma1            anywhere          tcp dpt:http   Salida-eth2    ----
all  --  video1            anywhere          ----           Salida         ----
tcp  --  video1            anywhere          tcp dpt:ssh    Salida-eth1    ----
all  --  192.168.50.101    anywhere          ----           Salida         ----
all  --  192.168.50.102    anywhere          ----           Salida         ----
all  --  192.168.50.103    anywhere          ----           Salida         ----
all  --  192.168.50.104    anywhere          ----           Salida         ----
all  --  192.168.50.106    anywhere          ----           Salida         ----
all  --  192.168.50.107    anywhere          ----           Salida         ----
all  --  192.168.50.110    anywhere          ----           Salida         ----
all  --  192.168.50.111    anywhere          ----           Salida         ----
```

Figura 6.8: Ejemplificación del uso de las políticas de enrutamiento con un nodo compuesto.

como organice la estructura de clase será la manera de aprovechar los recursos que la organización dispone. Asimismo, tampoco existe una forma o estrategia rigurosa de estructurar las clases de servicios.

Para evitar la confusión entre las unidades para medir el caudal de información que un sistema entrega; llámese bits por segundo o bytes por segundo. El SDTRL utiliza las siguientes reglas para especificar el ancho de banda:

1kbit equivale a 1024bit/s
1Bps equivale a 1bytes/s
Donde
1mbit = 1024 * 1 kbit
1mBps = 1024 kbps = 1024 *1024 Bps

Tomando como base inicial la configuración del ejemplo descrito por las figuras 6.2, 6.3 y 6.4. En la figura 6.9 podemos apreciar el ingreso de los comandos con el objetivo **class** para el tráfico particular, que tiene como fin estructurar diversas clases de servicio que se emplean en el ajuste y control de flujo de paquetes de entrada. Para lograr

esto, inicialmente se define el máximo ancho de banda que puede transmitir el SDTRL hacia todas las conexiones de red externas. A partir de esta definición, los siguientes comandos se encargaran de construir la estructura de clases repartiéndose el ancho de banda y fijando el nivel de prioridad. Posteriormente, las reglas de tráfico particular hacen uso de las diversas clases de servicios con el simple hecho de hacer referencia a ellas por medio de su identificador, con el fin controlar el torrente de los paquetes de una clase de tráfico particular.

```
[root:~]# sdtrc class del tp root
RTNETLINK answers: Invalid argument
SDTRC 1.0: try "sdtrc {command} help" for more information.
[root:~]# sdtrc class add tp root classid 1 rate 100mbit default 11
[root:~]# sdtrc class add tp parent 1 classid 10 rate 99mbit ceil 100mbit prio 1
[root:~]# sdtrc class add tp parent 1 classid 20 rate 1000Kbit ceil 10mbit prio 5
[root:~]# sdtrc class add tp parent 10 classid 11 rate 25mbit ceil 100mbit prio 3
[root:~]# sdtrc class add tp parent 10 classid 12 rate 75mbit ceil 100mbit prio 2
[root:~]# sdtrc rule add tp -i eth0 table Salida_con_firewall toclass 11
[root:~]# sdtrc rule add tp -i eth0 -s video1 table Salida_sin_firewall toclass 12
[root:~]# sdtrc rule add tp -i eth0 -s video1 -p tcp --dport 80 toclass 11
[root:~]# sdtrc rule add tp -i eth0 -s video3 table Salida_con_firewall toclass 12
[root:~]# sdtrc rule add tp -i eth0 -s video10 toclass 12
```

Figura 6.9: Reglas de tráfico particular que combinan distribución y control de tráfico.

Cabe señalar que en el ejemplo de la figura 6.9 define tres clases de servicios, cada una con un nivel de prioridad distinta (alta, media y baja). Como comentario, las clases que realmente asignar un ancho de banda garantizado son las que se consideran hojas dentro de su estructura, como en esta ocasión son la 11, 12 y 20. Estos servicios pueden ser usados por los miembros de la red privada para establecer diferencias de privilegios. Por ejemplo, toda la red privada de estudiantes tiene asignado un servicio con prioridad media (clase 11) y la salida con cortafuego. No así, el caso de la terminal video1 que tiene asignado un servicio con prioridad alta (clase 12) y la salida sin cortafuego. Al igual que los casos anteriores, existe una regla implícita que permiten a los flujos no clasificados tener asignado una clase de servicio, en este caso la clase 11 es la de omisión.

6.4.3. Tráfico masivo

Primero necesitamos que el administrador de la red identifique los tráficos masivos que transitan en la organización, después fija un ancho de banda y prioridad para cada uno de los tráficos en compromiso a su utilidad y uso. Es importante tener en cuenta las tasas máximas de transmisión de cada conexión, para evitar que la latencia empiece a crecer,

debido a llenados de búferes. Ya conocidos estos datos se procede a establecer las reglas para cada uno de los nodos externos, con objeto de definir las clases existen en cada uno de ellos, como se indica en la figura 6.10. De esta forma, posiblemente cada nodo tendría un número distinto de clases. Al final, el sistema logra tener un funcionamiento con base en reglas de la organización representadas por el tráfico masivo, sin tener una distinción en especial entre los sistemas terminales de la red privada.

```
[root:~]# sdtrc class del tm dev eth2 root
[root:~]# sdtrc class add tm dev eth2 root rate 100mbit default 14
[root:~]# sdtrc class add tm dev eth2 classid 10 rate 33mbit ceil 33mbit prio 0
[root:~]# sdtrc class add tm dev eth2 classid 11 rate 33mbit ceil 100mbit prio 1
[root:~]# sdtrc class add tm dev eth2 classid 12 rate 5mbit ceil 100mbit prio 2
[root:~]# sdtrc class add tm dev eth2 classid 13 rate 4mbit ceil 4mbit prio 2
[root:~]# sdtrc class add tm dev eth2 classid 14 rate 25mbit ceil 100mbit prio 3
[root:~]#
[root:~]# sdtrc rule add tm dev eth2 match ip protocol 1 0xff toclass 10
[root:~]# sdtrc rule add tm dev eth2 match ip tos 0x10 0xff toclass 10
[root:~]# sdtrc rule add tm dev eth2 match ip tos 0x02 0xff toclass 12
[root:~]# sdtrc rule add tm dev eth2 match ip tos 0x04 0xff toclass 12
[root:~]# sdtrc rule add tm dev eth2 match ip tos 0x08 0xff toclass 14
[root:~]# sdtrc rule add tm dev eth2 match ip protocol 0x6 0xff \
>  match tcp src 22 0xffff toclass 11
[root:~]# sdtrc rule add tm dev eth2 match ip protocol 0x6 0xff \
>  match tcp src 80 0xffff toclass 13
[root:~]# sdtrc rule add tm dev eth2 match ip protocol 0x6 0xff \
>  match u8 0x02 0xff at 33 toclass 11
```

Figura 6.10: Ejemplificación del uso de las reglas tráfico masivo para la interfaz red eth2.

Al igual que en el caso del trafico particular, se define primero el estructura de clases, pero en este caso se especifican en un solo nivel y se dividen el ancho de banda de la clase root de manera extrita. Concretamente en el ejemplo de la figura 6.10, las cinco clases de servicio están pensadas para dar beneficio a ciertos tráficos. Como habíamos mencionado en la sección 6.3 las reglas de tráfico masivo tiene una sintaxis distinta, pues están con base al filtro u32. La primera regla en este ejemplo hace referencia al tráfico ICMP, recibiendo un trato de máxima prioridad y un ancho de banda garantizado y acotado. Las siguientes reglas permiten la identificación y control de los datagramas con los bits de Tipo de Servicio (ToS), donde cada uno de éstos es tratado de manera diferente. En seguida, se definen reglas para tráficos bien conocidos (http y ssh) que apuntan a ciertas clases. Finalmente, existe otra regla que hace referencia a los paquetes que comienzan un enlace de comunicación tcp, para dándoles un trato privilegiado.

6.4.4. Tráfico de respuesta

La configuración de tráfico de respuesta se comporta de manera similar a una configuración de asignación de recursos de un nodo de red externa. La discrepancia radica en que el tráfico de respuesta maneja la contestación del tráfico masivo, con la particularidad de que no puede identificar por aplicaciones. Además, también puede hacer referencia a flujos de paquetes específicos pertenecientes a sistemas terminales de la red privada. De hecho el tráfico de respuesta tiene como objetivo limitar flujos bien conocidos y tráficos particulares de una manera mixta. Cabe resaltar, que este tráfico no es tan precisos o especifico como puede ser el caso de los tráficos masivos o particulares, pues para ello sirven. Simplemente el tráfico de respuesta es de apoyo para éstos dos.

El ejemplo de la figura 6.11 considera varios tipos de servicios de entrada a la red privada de los estudiantes (192.168.50.0/24). Se divide el ancho de banda disponible en tres partes (66mbps, 1mbps y 33mbps), uno para el tráfico de respuesta sin restricciones, el otro para subdividirlo en clases de tráfico más específicas y el último es para uso común de todos los tráficos no clasificados.

```
[root:~]# sdtrc class add tr root classid 1 rate 100mbit default 30
[root:~]# sdtrc class add tr parent 1 classid 10 rate 66mbit ceil 100mbit
[root:~]# sdtrc class add tr parent 1 classid 20 rate 1220Kbit ceil 1500kbit
[root:~]# sdtrc class add tr parent 1 classid 30 rate 32780kbit ceil 100mbit
[root:~]# sdtrc class add tr parent 20 classid 21 rate 75kbit ceil 75kbit
[root:~]# sdtrc class add tr parent 20 classid 22 rate 150kbit ceil 150kbit
[root:~]# sdtrc class add tr parent 20 classid 23 rate 1mbit ceil 1500kbit
[root:~]#
[root:~]# sdtrc rule add tr -o eth0 -d video1 -p tcp toclass 10
[root:~]# sdtrc rule add tr -o eth0 -d video1 -s computacion -p tcp toclass 22
[root:~]# sdtrc rule add tr -o eth0 -d video2 -s computacion -p tcp toclass 22
[root:~]# sdtrc rule add tr -o eth0 -d video3 toclass 21
[root:~]# sdtrc rule add tr -o eth0 -d video3 -s computacion -p tcp toclass 22
[root:~]# sdtrc rule add tr -o eth0 -d video4 -s computacion -p tcp toclass 22
[root:~]# sdtrc rule add tr -o eth0 -d sigma2 -s altair toclass 21
[root:~]# sdtrc rule ls tr
```

Figura 6.11: Control de tráfico de entrada en la red privada de uso común.

En el caso de la clase con el identificador 20 esta se subdividir en tres clases. La clase 21 maneja un rango de ancho de banda muy pequeño en comparación con la clase root, lo mismo es para las clases 22 y 23. El objetivo de tener estás clases es controlar la descarga de de información proveniente de Internet; el ancho de banda que se dispone para acceder a esta red es mucho menos a la velocidad de la red privada (entre 3 a 5 mbps). Las clases 21 22 y 23 están pensadas para miembros que usan la red para revisar

correos y examinar alguna que otra página web, evitando así que hagan uso inadecuado de los recursos. También es notorio el hecho que para crear la estructura de clases no se necesito de especificar niveles de prioridad, pero esto no significa que no pueda tener, la idea de esto es dejar a los paquetes que se gobiernen con la filosofía de una cola tradicional.

6.5. Resumen

En principio, la idea de este capítulo es transmitir la forma de cómo puede ser usado el prototipo SDTRL (**sdtrc**). Los ejemplos descritos son una de las muchas maneras de emplear al sistema, que realmente debe estar enfocado a las necesidades de la organización. El SDTRL ciertamente ataca en tres flancos la administración de los recursos de la red, independientemente de sus complejidades técnicas: la distribución de tráfico de acuerdo al beneficio de la organización, el control de flujo paquetes cuando se reciben datos de un sistema terminal remoto (download) y el envió de datos hacia otras estaciones terminales remotas (upload).

Se había mencionado que las reglas de tráfico particular tienen una alta probabilidad de ser cambiadas frecuentemente por el administrador, por el hecho de dar servicio a los clientes. En contraste, las listas de reglas de tráfico masivo y de respuesta son reglas que por lo general se establecen una sola vez o son modificadas muy pocas veces, pues estas refieren a las políticas de la organización.

Dentro del SDTRL, el bloque de tráfico particular y de respuesta permite realizar una estructura de clases mucho más libre que en el caso de la estructura de clases para el tráfico masivo que es de un solo nivel. La explicación a ello es que normalmente la repartición de los recursos para los tráficos bien conocidos se describen al estilo de gráficas de pastel, sin manejar niveles de profundidad. Como nota adicional para las estructuras de clases; para obtener un buen desempeño en el sistema se debe utilizar rangos mayores a 4kbit para garantizar una operación segura en la disciplina de cola HTB.

Este escenario probó una situación, que si no es un error de implementación, es una cuestión provista por la emulación de dos PSI independientes. En caso de utilizar un servidor DNS que esta conectado a la red que provee salida hacia Internet, las peticiones DNS siempre saldrán por la interfaz que conecta hacia está red. En caso de que esta conexión fallara ya no habría manera de obtener más respuestas DNS, aun a pesar de tener disponible la otra conexión. La solución a ello es tener registrado, en los sistemas

terminales clientes, un DNS que sea independiente a alguna de las redes externas.

Capítulo 7

Conclusiones

Los sistemas que brindan la posibilidad de administrar los recursos de una red han demostrados ser cada vez más ineludibles. Gran parte de las aplicaciones actuales requieren tener un trato, que si bien no de una forma rigurosa como puede ser el caso de las aplicaciones de tiempo real, si para garantizar un trato preferencial ajustable entre ellas.

En la presente tesis, se ha planteado una alternativa para solucionar la problemática de la identificación, control y distribución de tráfico de datos entre las diferentes redes interconectadas. En el que, la mayor preocupación es realizar una distribución de tráfico mucho más flexible que la distribución tradicional (por dirección destino), adicionado a su vez mecanismo de calidad de servicio. Se obtiene así, un sistema de administración de recursos de red el cual puede ser colocado en la frontera o en el punto de interconexión de todas las redes involucradas.

Como resultado de este trabajo, se ha desarrollado un prototipo del SDTRL (**sdtrc**) que ha sido implementado en Linux. El prototipo esta completamente ligado a las cualidades que el núcleo del sistema operativo posee en cuanto a la distribución y control de tráfico. La forma de cómo se organiza y configura el sistema para el prototipo es realmente la aportación de este trabajo de tesis.

El prototipo ha sido diseñado para poder cumplir con la mayoría de las necesidades de distribución e identificación de privilegios de diferentes tráficos en una organización. Para ello, el SDTRL contempla estratégicamente la existencia de tres puntos de control para tipos de tráficos específicos, ubicados en puestos coyunturales entre las entradas y salidas del mismo sistema. Estas características son las novedades que presenta el SDTRL, además de que provee el esqueleto base para conseguir un control de red más eficaz.

Los limítes del espectro de soluciones que puede dar el SDTRL a las organizaciones, esta directamente refrendado por su misma arquitectura. Es decir, el problema de no poder solucionar un caso particular, se debe a que el diseño del prototipo no se acopla o adecua a los requerimientos y necesidades de la organización. Por otra parte, el SDTRL no es un sistema de ruteo tradicional, ya que no contiene ningún protocolo de enrutamiento, ni tampoco es un sistema que entrega calidad de servicio extremo a extremo ya que no posee un protocolo de CdS.

El SDTRL no maneja algún protocolo de calidad de servicio, como puede ser el ServDiff o ServInt, para propiciar las condiciones necesarias para el manejo del tráfico en tiempo real. Únicamente, utiliza herramientas de CdS que permiten administrar y controlar los recursos de una red. Por la parte de los servicios del SDTRL, se define un único servicio para los diferentes flujos de paquetes. Este servicio provee al tráfico la capacidad de asignación mínima de ancho de banda garantizada y una prioridad de atención con respecto a todo los tráficos del modulo donde se encuentre.

En comparación con otros sistemas, el SDTRL es una buena alternativa por su simplicidad de uso en cuanto a su configuración, o por su nivel de abstracción que evita hacer referencia a comandos sofisticados. Si bien, la idea principal que maneja el sistema es la distribución de tráfico entre dos o más interfaces, ya sea que haya balanceo de carga o no, para ya ello maneja el concepto de nodos de red que facilitan esta labor. Otras de las ventajas son la incorporación del dispositivo IMQ que permite una mayor flexibilidad y manejo de tráfico de entrada, coadyuvando así la definición de tráfico particular que de manera explicita esta relacionada con el direccionamiento y control de tráfico en una sola instrucción.

Desde luego, también es cierto que el SDTRL tiene limitaciones como pueden ser la dependencia directa con otras herramientas, carece de un sistema de monitoreo de tráfico y mantiene un ambiente de trabajo sobre comandos por mencionar algunas, lo cual no lo hace competitivo en ciertos aspectos con otros sistemas actuales. Sin embargo, esto contribuye para el desarrollo de líneas de investigación futuras las cuales se describen a continuación.

Una de las flaquezas del sistema planteado con respecto a otros similares es la identificación de tráficos más sofisticado y puntual. El SDTRL identifica únicamente los encabezados de los paquetes en los niveles 3 y 4 del modelo OSI. Sin embargo, es innegable la necesidad de poder clasificar los tráficos por otras características tales como por aplicaciones, por URL's o grupos de URL's, por usuarios, entre otras más.

Actualmente, el prototipo tiene como interface con el usuario una instrucción que

posee un conjunto de objetivos y comandos, que permite configurar y operar al SDTRD. Sin embargo, un elemento no crucial pero que sin duda sería de gran ayuda es una interface gráfica que propicie y pueda facilitar aún más la forma de operar al sistema.

Finalmente, el SDTRL ha probando ser un sistema confiable en un escenario real teniendo un desempeño y funcionalidad adecuada. Sin mencionar que también disminuye la probabilidad de errores técnicos y humanos en el momento de sus configuración.

Apéndice A

Descripción del filtro u32

El filtro U32 permite filtrar de acuerdo a cualquier conjunto de bits del paquete IP. En su forma más simple el filtro U32 es una lista de registros. La línea de órdenes del programa tc filter, que se usa para configurar el filtro, consiste en tres partes: especificación del filtro, selector y acción. La mayoría de las reglas de ajuste que se emplean en la especificación del filtro normalmente empiezan con éste preámbulo:

`tc` filter add dev eth0 parent 1:0 protocol ip prio 1 u32 ...

Los selectores se comparan con el paquete IP que se está procesando hasta encontrar la primera coincidencia, y entonces se ejecuta la acción asociada. Se distinguen dos formas de especificar una selección:

Filtrado con base en el encabezado: Ésta forma se conoce como criterio directo y es muy fácil de utilizar, algunos criterios son:

- Dirección IP origen/destino.
- Protocolo utilizado: tcp, udp, icmp, etc.
- Puertos origen/destino.
- Campo ToS.

Filtrado con base en el segmento de datos: Permite filtrar por criterios muy específicos, pero muy complicado de utilizar. En la Tabla A.1, se muestran los tipos de parámetros con su respectiva descripción. Los parámetros se pueden especificar en números decimales, hexadecimales o direcciones IPv4.

Tabla A.1: Selectores específicos.

Selector de Protocolo	Selector de campo	Parámetros	Descripción
match ip	src	*Prefix/32*	packet source address
	dst	*Prefix/32*	packet destination address
	tos	*tos u8*	packet TOS field
	dsfield	*tos u8*	packet TOS field (synonym)
	precedence	*tos u8*	packet TOS field (synonym)
	ihl	*ihl u8*	packet header length
	protocol	*prot u8*	encapsulated (higher layer) protocol number
	nofrag		packet is not fragmented
	firstfrag		packet contains first fragment
	df		Don't Fragment flag is set
	mf		More Fragments flag is set
	sport	*port u16*	higher layer protocol source port
	dport	*port u16*	higher layer protocol destination port
	icmp_type	*type u8*	ICMP message type
	icmp_code	*code u8*	ICMP message code
match udp, tcp	src	*src u16*	source port
	dst	*dst u16*	destination port
match icmp	type	*type u8*	ICMP message type
	code	*code u8*	ICMP message code

Lo anterior se ejemplifica en el siguiente comando:

`tc` filter add dev ppp0 parent 1:0 prio 10 u32 match tcp dst 53 0xffff flowid 1:4

La **sintaxis general** del selector en el filtro u32 es:

match [u32 | u16 | u8] PATRON MASCARA [at DESPLA | nexthdr+DESPLA]

Las palabras clave u32, u16 o u8 especifica la longitud en bits de los patrones. En seguida se describe el PATRON y MASCARA, de la longitud definida por la palabra clave anterior. La clave *at* indica que la coincidencia debe comenzar en un desplazamiento específico. La opción *nexthdr* indica la siguiente cabecera encapsulada en el paquete IP, esto es, la cabecera del protocolo de la capa superior. Para mayor referencia ver [24].

Bibliografía

[1] M. Reisslein, K. Ross, and S. Rajagopal. A framework for guaranteeing statistical QoS. *IEEE/ACM Transactions on Networking (TON)*, 10(1):27 – 42, 2002.

[2] George C. Sackett. *Cisco Router Handbook*. The MacGraw-Hill, 2nd edition, August 1999.

[3] Andrew S. Tanenbaum. *Redes de computadoras*. Prentice Hall, 3rd edition, April 1998.

[4] Marcel Waldvogel, George Varghese, Jon Turner, and Bernhard Plattner. Scalable high speed IP routing lookups. In *Proceedings of SIGCOMM '97*, pages 25–36, September 1997.

[5] O'Reilly. *Linux Network Administrators Guide*. O'Reilly & Associates, 2nd edition, 2000.

[6] Inc Cisco Systems. *Cisco Networking Academy Program: Second-Year Companion Guide*. Pearson Education, 2nd edition, May 2001.

[7] Inc Cisco Systems. Cisco white paper: Enhanced Interior Gateway Routing Protocol (EIGRP), December 2003.

[8] Inc Cisco Systems. Cisco white paper: Policy-Based Routing (PBR), December 1996.

[9] M. Chatzaki and S. Sartzetakis. QoS-policy based routing in public heterogeneous broadband networks. In *In Proceedings of Interworking'98 Conference*, Ottawa, July 1998.

[10] J. Shin, J. Kim, and C. Kuo. Quality-of-Service mapping mechanism for packet video in Differentiated Services Network. *IEEE Transactions on Multimedia*, 3(2):219–231, June 2001.

[11] I. Foster, A. Roy, V. Sander, and L. Winkler. End-to-end quality of service for high-end applications. Technical report, Argonne National Laboratory, 1999.

[12] X. Xiao, A. Hannan, B. Bailey, and L.Ni. Traffic engineering with MPLS in the internet. *IEEE Network Magazine*, pages 28–33, March 2000.

[13] Fred Halsall. *Comunicacion de Datos, Redes de Computadores y Sistemas Abiertos*. Addison Wesley Longman, 4th edition, February 2000.

[14] Inc Cisco Systems. *Internetworking Technologies Handbook*. Cisco Press, 3rd edition, December 2000.

[15] M. Bechler, H. Ritter, and J. Schiller. Traffic shaping in end systems attached to qos-supporting networks. In *In proceedings of the 6th IEEE Symposium on Computers and Communications (ISCC 2001)*, Tunesia, July 2001.

[16] F. Kuo, W. Effelsbeg, and J. Garcia-Luna-Aceves. *Multimedia Communications: Protocols and Applications*. Prentice Hall PTR, 1st edition, September 1998.

[17] C. Venkatramani. *The design, implementation and evaluating of RETHER: Areal-Time Ethernet Protocol*. PhD thesis, University of New York, New York, 1997.

[18] La Dirección de Telemática de México. Propuesta para el grupo de desarrollo de qos en red CUDI. <http://telematica.cicese.mx/internetii/qcudi/qos_cudi.html>, 2000.

[19] Julian Anastasov. Software, patches and doc. Routing and NAT extensions for Linux. <http://www.ssi.bg/~ja/#routes>, 2001.

[20] Schulzrinne, Casner, Frederick, and Jacobson. RTP: A transport protocol for real-time applications. *Internet-Draft ietf-avt-rtp-new-01.txt (work inprogress)*, 1998.

[21] Ranjita Bhagwan and Bill Lin. Fast and scalable priority queue architecture for high-speed network switches. In *INFOCOM (2)*, pages 538–547, 2000.

[22] S. Miltchev, S. Ioannidis, and A. Keromytis. A study of the relative costs of network security protocols. In *proceedings of the USENIX Annual Technical Conference, Freenix Track*, pages 41–48, June 2002.

[23] Bert Hubert. Linux advanced routing & traffic control. In *Proceedings of the Ottawa Linux Symposium*, pages 213–223, 2002.

[24] Bert Hubert, Thomas Graf, and Martijn van Oosterhout. Linux advanced routing & traffic control howto. Technical report, LARTC, 2004.

[25] Alexey Kuznetsov. Ip command reference, iproute2 package. <http://linux-ip.net/gl/ip-cref/>. Technical report, Institute for Nuclear Research., 1999.

[26] KJ Loh and Irwin Gui. Performance of a linux implementation of class based queueing. In *International Conference on Computer Communications and Network*, 1998.

[27] Martin Devera aka devik. HTB linux queuing discipline manual - user guide. <http://luxik.cdi.cz/d̃evik/qos/htb/manual/userg.htm>. Technical report, HTB Home, 2002.

[28] Rusty Russell and Harald Welte. Linux netfilter hacking howto, <http://www.netfilter.org/documentation/howto/netfilter-hacking-howto.html>. Technical report, Netfilter Home, 2002.

[29] IMQ Home. The intermediate queueing device. <http://trash.net/ kaber/imq/>, 2004.

[30] Christoph Simon. Nano-howto Linux kernel patches. <http://www.ssi.bg/ ja/nano.txt>, 2001.