Implementación de Calidad de Servicio en Sistemas Linux

Ana Carolina Alonso de Armiño
Rodolfo del Castillo
Departamento de Ciencias de la Computación,
Universidad Nacional del Comahue.
Buenos Aires 1400, Neuquen, Argentina
Email: aalonso@uncoma.edu.ar

Resumen

La implementación de Calidad de Servicio en Sistemas Linux permite administrar el uso del ancho de banda de una red manipulando el tráfico que circula por ella. "Calidad de Servicio" se refiere al servicio de transferencia de datos en el que pueden diferenciarse los distintos tipos de tráfico que son enviados por la red. De esta manera, se los puede tratar de modo diferenciado según sus requisitos. En este paper será presentada la forma en que pueden ser configurados los sistemas Linux para conseguir routers con capacidades Qos, es decir, routers que puedan brindar distintos servicios de calidad según los requisitos del tráfico que pasa a través de él.
Nuestro objetivo final es comprender los mecanismos y elementos que constituyen la arquitectura de Servicios Diferenciados, para luego poder elaborar guías que nos indiquen qué configuración puede ser más útil frente a un entorno y situación dados.

PALABRAS CLAVES: Calidad de Servicio, Servicios Diferenciados, Disciplinas de Colas.

1. Introducción

Tradicionalmente los proveedores del servicio de red proveían a todos los clientes el mismo nivel de servicio, el cual alcanzaba para el tráfico moderado que consistía en su mayoría en la transferencia de archivos de texto plano, relativamente pequeños sin imágenes, sonidos, videos, etc. Hoy, el incremento en el uso de Internet y el contenido multimedial ha comprometido la performance de las aplicaciones críticas, a la vez que aparecieron nuevas aplicaciones con grandes demandas de calidad, como por ejemplo televisión, radio, etc. Por esta razón fue necesario que los proveedores ofrezcan niveles alternativos de servicio para cubrir las necesidades de cada cliente.

Diff-Serv o *Servicios Diferenciados* [15] es un mecanismo que ofrece distintos niveles de servicio al tráfico de aplicaciones de clientes singulares, proveyendo la calidad de servicio (QoS) y performance adecuada a cada uno. Los clientes requieren un nivel de performance paquete por paquete, marcando en cada uno el campo DS [14] del header IP del mismo. Este valor especifica el PHB (per-hop Behaviour) con el que los routers manejan el tráfico resultando así en diferente tratamiento en cada punto para un mismo flujo. Con este esquema muchos flujos de tráfico pueden ser agrupados en uno o en un número pequeño de PHBs, facilitando el procesamiento y almacenamiento asociado con la clasificación de los paquetes. Inicialmente el proveedor y el cliente elaboran una acuerdo (policing profile) que describe la tasa a la cual podrá ser entregado el tráfico en cada nivel de servicio.

La implementación de Servicios Diferenciados es la alternativa adecuada cuando se desea mejorar la performance de una red o cuando se requiere brindar servicios de calidad a nuestros clientes. Con el objetivo de comprender los mecanismos y elementos que constituyen la arquitectura de Servicios Diferenciados vamos a comparar algunas posibles configuraciones de los routers. Elaboraremos también el conjunto de pruebas que nos permitirán evaluar su funcionamiento.

La organización de este trabajo es como sigue: en la sección 2 serán presentados los pasos que deben seguirse para efectuar la configuración de un router Linux con capacidades QoS y explicaremos el trabajo que estamos desarrollando. En la sección 3 se presentan nuestras conclusiones y se mencionan los trabajos de investigación futuros.

2. Configuración de Calidad de Servicio

La configuración de QoS comienza con la instalación de las máquinas Linux cuyo rol será de routers con capacidades QoS [5,6,7]. Esto se inicia con la instalación del sistema operativo y la compilación del kernel para que soporte las disciplinas de colas que se desean configurar [1]. QoS está soportado desde la versión 2.4 del kernel Linux [2]. En nuestra investigación estamos trabajando con Suse Linux 8.2 en la máquina que actúa como router.

Para comenzar debemos situarnos en el entorno en el que vamos a trabajar, analizar el tráfico que viaja por nuestra red y evaluar que y como lo deseamos manejar y priorizar. Para satisfacer los requisitos de los clientes debemos asegurarnos de que el tráfico es tratado en forma uniforme en toda la red, es decir, que debe existir una configuración consistente en todos los routers que la integran para que el tratamiento que reciban los paquetes sea el mismo en cada hop que atraviesa. Asimismo deberíamos manejar y administrar adecuadamente el tráfico externo que atraviesa nuestra red con el objetivo de que no interfiera con el tráfico interno de nuestros clientes. En nuestro análisis generamos el tráfico de la red utilizando herramientas destinadas a esto.

Las métricas que caracterizarán la calidad del servicio end-to-end son la disponibilidad de la prestación, el retraso y la pérdida de paquetes, la variación del retraso (jitter) y la productividad. Se deben tener presentes son los *SLAs* [3,4], *Acuerdos del Nivel de Servicio* que el proveedor le brindará al cliente para comprobar que se ha ofrecido el servicio deseado. Este es un acuerdo que ambos negociarán y que deberá reflejar las necesidades del cliente y el compromiso del proveedor de satisfacerlas. Una vez que se ha establecido el/los SLAs se deben especificar cuales son los requerimientos del tráfico que se generará en términos de configuraciones. Es decir, debemos ver la forma en que el tráfico será identificado (IP origen, IP destino, puerto origen, puerto destino, etc) y que característica tendrá, por ejemplo disponibilidad de una parte fija del ancho de banda, garantía de baja latencia, de bajo retardo, de baja variabilidad del retardo, etc.

Estas configuraciones se colocan en el router por medio de las *Disciplinas de Colas* o *qdisc*. Las Disciplinas de colas permiten cambiar la forma en que enviamos los datos.

Una qdisc con clases [9] puede contener varias clases que son internas a la qdisc. A cada una de ellas se le puede añadir a su vez otras clases. De esta manera cada una de ellas puede tener como padre a una qdisc u otra clase. Una clase terminal es aquella que no tiene hijas y tiene una qdisc adjunta, la cual es responsable de enviar los datos.

Cada qdisc con clases tiene que saber a que clase enviar los paquetes cuando llegan, esto es hecho usando *clasificadores* o *filtros*. Los filtros contienen una serie de condiciones que deben coincidir para finalmente determinar a que clase pertenece un paquete recibido.

Para aclarar los conceptos de qdisc, clases y filtros veremos un ejemplo. La figura 1 ejemplifica una jerarquía de clases:

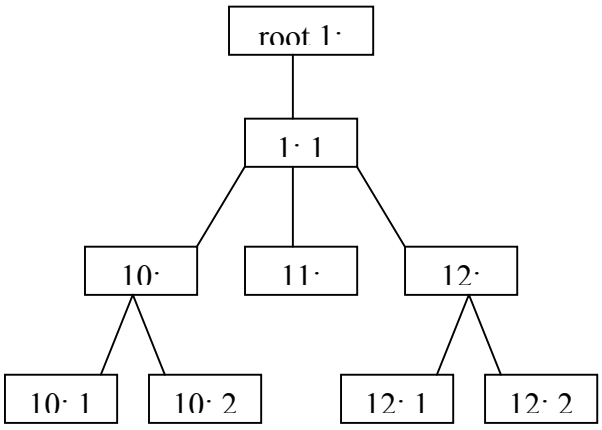

Figura 1: Ejemplo de una jerarquía de clases.

La qdisc es identificada como 1 y la jerarquía de clases comienza en 1:1. Un paquete podría ser clasificado por los filtros según la siguiente secuencia:

1: -> 1:1 -> 12: -> 12:2

es decir que el paquete es clasificado primero como de la clase 1:1, en esa clase se vuelven a consultar los filtros y se decide que el paquete pertence a la clase 12:0, finalmente una nueva consulta a los filtros determinará que el paquete será clasificado como de la clase 12:2. También es posible hacer una clasificación más directa si existiera un filtro que clasifique al paquete como de la clase 12:2 directamente:

1: -> 12:2

Con estos tres elementos (qdisc, clases y filtros) se configura la forma en que el tráfico será reenviado por la red, es decir, que el tráfico podrá ser reacomodado para que responda a las características deseadas. Esto se logra gracias a que en los filtros se puede identificar a que clase pertenece un flujo y en la clase se determina que características de calidad se le otorgarán.

Llamamos *Scheduling* (u ordenamiento) al proceso por el cual una qdisc determina el tratamiento que un paquete debe recibir. Llamamos *Shaping* (o ajuste) al proceso de retrazar los paquete para hacer que el tráfico no supere la tasa máxima configurada. El descarte de paquetes para hacer más lento el tráfico es también llamado *Shaping*.

Policing, es la manipulación que recibe el tráfico de forma de ser retrazado o descartado para lograr que el tráfico respete el ancho de banda configurado.

La tarea de reordenar y la política de manejo del tráfico que sale de nuestro router es la base de la Calidad de Servicio, con esto se puede lograr que el tráfico indeseado no afecte a las aplicaciones que requieran mayor calidad, como aquellas de tiempo real.

En nuestro estudio hemos elegido dos configuraciones para comparar. Una de ellas se basa en la disciplina CBQ y la otra en HTB. CBQ es una disciplina de colas con clases. La forma en que efectúa el ajuste del tráfico es asegurando que el enlace sobre el que se transmiten los datos esté ocioso el tiempo necesario para que cada clase configurada obtenga el ancho de banda especificado. El enfoque jerárquico de HTB se adecua a situaciones donde se tiene una cantidad fija de ancho de banda a dividir para diferentes propósitos, asegurándole a cada uno una cantidad de ancho de banda. HTB funciona en forma similar a CBQ pero no efectúa cálculos de tiempo ocioso. En ambas configuraciones se ha definido la siguiente diferenciación del tráfico:

- Tráfico dirigido al puerto 5000: ancho de banda de 500 Kb que puede compartir con el tráfico dirigido al puerto 5004.

- Tráfico dirigido al puerto 5004: ancho de banda de 300 Kb que puede compartir con el tráfico dirigido al puerto 5000.

- El tráfico restante podrá usar el ancho de banda restante.

Asumimos que tenemos un canal de comunicación de 10 Mb.

Los grupos de pruebas que hemos planteado son los siguientes:

Prueba 1:

Generación de
- Flujo 1: tráfico dirigido al puerto 5000 de 200 Kb
- Flujo 2: tráfico dirigido al puerto 5004 de 350 Kb
- Flujo 3: el tráfico subyacente es dirigido al puerto 5005 y es de 550 Kb.

Hipótesis: como el volumen de tráfico generado en el flujo 2 supera el ancho de banda que tiene asignado pero a su vez puede utilizar el del flujo 1 no deberían ocurrir descartes de paquetes en ninguna de las clases.

Prueba 2:

Generación de
- Flujo 1: tráfico dirigido al puerto 5000 de 400 Kb
- Flujo 2: tráfico dirigido al puerto 5004 de 400 Kb
- Flujo 3: el tráfico subyacente es dirigido al puerto 5005 y es de 800 Kb.

Hipótesis: como el volumen de tráfico generado el flujo 1 no supera el ancho de banda asignado no deberían ocurrir descarte; como el volumen de tráfico del flujo 2 supera los 300 Kb deberían ocurrir descartes.

Prueba 3:

Generación de
- Flujo 1: tráfico dirigido al puerto 5000 de 200 Kb
- Flujo 2: tráfico dirigido al puerto 5004 de 200 Kb

- Flujo 3: el tráfico subyacente es dirigido al puerto 5005 y es de 400 Kb.

Hipótesis: como el volumen de tráfico generado en cada flujo no supera el ancho de banda asignado a ellos no deberían ocurrir descartes.

Prueba 4:

Generación de

- Flujo 1: tráfico dirigido al puerto 5000 de 450 Kb
- Flujo 2: tráfico dirigido al puerto 5004 de 250 Kb
- Flujo 3: el tráfico subyacente es dirigido al puerto 5005 y es de 700 Kb.

Hipótesis: como el volumen de tráfico generado el flujo 1 y 2 no superan el ancho de banda asignado a cada uno de ellos no supera el ancho de banda provisto no deberían ocurrir descartes. Pero, como se ha configurado que entre ambas clases no pueden superar los 600 Kb deberían ocurrir descartes proporcionales al desborde causado en cada uno.

Cada una de las pruebas será efectuada con las dos configuraciones elegidas y con diferentes tamaños de paquetes para analizar el comportamiento de las disciplinas a comparar. Este análisis nos permitirá utilizar la configuración mas adecuada frente a diferentes entornos. Un buen análisis de los patrones de tráfico que viajan por el medio nos mostrará las características de los paquetes. Conociendo además las aplicaciones de los clientes que debemos priorizar será posible hacer una configuración adecuada en la que cada aplicación podrá funcionar correctamente aunque haya tráfico indeseado en la red, ya que éste no consumirá los recursos existentes.

3. Conclusiones y trabajos futuros:

Se ha hecho una breve descripción de las características y elementos que intervienen en la implementación de Calidad de Servicio. Se han explicado los pasos que hay que llevar adelante para hacer una implementación. Se están estudiando y elaborando test para analizar el funcionamiento de las disciplinas de colas (tales como CBQ y HTB) y la arquitectura de servicios diferenciados con el fin de elaborar guías de usos para realizar implementaciones efectivas en un entorno determinado.

4. Referencias:

1. **"Linux - Advanced Networking"**. Saravanan Radhakrishnan. The University of Kansas, Lawrence, KS 66045-2228. 1999-09-30. http://qos.ittc.ukans.edu/howto/howto.html
2. **"The Linux Kernel HOWTO"**. http://www.tldp.org/tldp-redirect.php?url=/HOWTO/Kernel-HOWTO.html
3. **"Service Level Agreements"**. Ron Sprenkels, Aiko Pras. - April 18, 2001. http://ing.ctit.utwente.nl/WU2/
4. **"Differentiated Services on Linux"**. Werner Almesberger, Jamal Hadi Salim, Alexey Kuznetsov. June 1999. http://citeseer.ist.psu.edu/almesberger99differentiated.html
5. **"Linux Network Traffic Control - Implementation Overview"**. Werner Almesberger, EPFL ICA. February 4, 2001. http://cad.csie.ncku.edu.tw/~wnlee/docu/slides/linux_traffic_control.pdf
6. **"Traffic Control - Next Generation. Reference Manual"**. Werner Almesberger. June 19, 2003. http://linux-ip.net/gl/tcng/
7. **"Linux Advanced Routing & Traffic Control HOWTO"**. bert hubert, Gregory Maxwell,

Remco van Mook, Martijn van Oosterhout, Paul B Schroeder, Jasper Spaans. Published v0.9.0 Date: 2002/03/06. http://lartc.org

8. **"Internet Protocol - Quality of Service Page"**. http://qos.ittc.ukans.edu/
9. **"Supporting Differentiated Service Classes: Queue Scheduling Disciplines"**. Chuck Semeria. Juniper Networks. December 2001. http://www.juniper.net/solutions/literature/white_papers/200020.pdf
10. **RFC 1633**, "Integrated Services". Bob Braden, David Clark, Scott Shenker. June 1994.
11. **RFC 1157**: "Simple Network Managerment Protocol". Jeffrey D. Case, Mark Fedor, Martin Lee Schoffstall, James R. Davin. May 1990.
12. **RFC 2212**: "Specification of Guaranteed Quality of Service". Scott Shenker, Craig Partridge, Roch Guerin. September 1997.
13. **RFC 2215**: "General Characterization Parameters for Integrated Services Network Elements". Scott Shenker, John Wroclawski. September 1997.
14. **RFC 2474** - "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers". K. Nichols, S. Blake, F. Baker, D. Black. December 1998.
15. **RFC 2475** - "An Architecture for Differentiated Services". M. Carlson, W. Weiss, S. Blake, Z. Wang, D. Black, and E. Davies. December 1998.
16. **RFC 2386** - "A Framework for QoS-based Routing in the Internet". E.Crawley, R. Nair, B. Rajagopalan, H. Sandick. August 1998.