Introducción a las Redes de Computadoras

Obligatorio 2 – 2011

Facultad de Ingeniería
Instituto de Computación
Departamento de Arquitectura de Sistemas

Nota previa - IMPORTANTE

Se debe cumplir íntegramente el "Reglamento del Instituto de Computación ante Instancias de No Individualidad en los Laboratorios", disponible en `http://www.fing.edu.uy/inco/pm/uploads/Ense%f1anza/NoIndividualidad.pdf`

En particular está prohibido utilizar documentación de otros grupos, de otros años, de cualquier índole, o hacer público código a través de cualquier medio (news, correo, papeles sobre la mesa, etc.).

Forma de entrega

Una clara, concisa y descriptiva documentación es clave para la evaluación de los trabajos entregados.

La entrega del obligatorio consiste en un único archivo `obligatorio2.tar.gz` que deberá a su vez contener los siguientes archivos:

- Un documento llamado `obligatorio2.pdf` donde se documente todo lo solicitado en el presente obligatorio. El mismo deberá incluír las desiciones de diseño, máquinas de estado y pseudocódigo, y finalmente la documentación de la implementación.

- Los archivos correspondientes a la implementación de la aplicación propuesta (implementando el pseudocódigo)

- Makefile para la compilación de éste (siempre que sea más complejo que g++ proxyserver.cc)

La entrega se realizará a través del sitio Web del curso, en la URL `http://www.fing.edu.uy/inco/cursos/redescomp/entrega.html`

Fecha de entrega

Los trabajos deberán ser entregados antes del domingo 24 de abril a las 23:30 horas. No se aceptará ningún trabajo pasada la citada fecha y hora. En particular, no se aceptarán trabajos enviados por e-mail a los docentes del curso, ni entregados en medios magnéticos en el instituto.

El sistema de entregas soporta múltiples entregas por grupo, llevando un histórico de las mismas. Se recomienda realizar una entrega vacía con tiempo, a los efectos de verificar que su sistema le permite entregar correctamente.

Observaciones

Todas las ejecuciones deberán ser realizadas en las máquinas virtuales distribuidas como parte del material del curso.

Objetivo del Trabajo

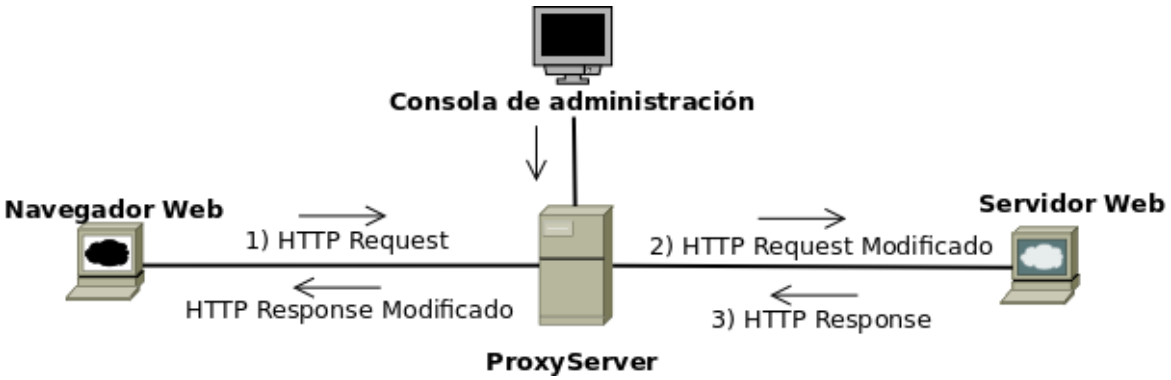
Aplicar los conceptos teóricos de capa de aplicación, mediante el desarrollo de una aplicación que funciona en red.

Familiarizarse con el uso de las API de *sockets* y *threads*, fundamentales en el desarrollo de aplicaciones para redes de computadoras.

Consolidar los conocimientos acerca del manejo de protocolos, mediente el estudio particular del protocolo HTTP (versiones 1.0 [1] y 1.1 [2])

Descripción general del problema

Se desea implementar una aplicación (que funciona al mismo tiempo como cliente y servidor) para permitir controlar el tráfico web mediante ciertas políticas definidas por los administradores de la red. De esta manera, la aplicación actuará a manera de proxy HTTP, y adicionalmente analizará los datos involucrados en la petición/respuesta para aplicar las políticas definidas y filtrar el tráfico que se haya decidido denegar.



La solución utilizará un protocolo de transporte confiable, orientado a conexión (TCP) e implementará un subconjunto del protocolo HTTP/1.0. Así mismo, el servidor deberá permitir la conexión de los administradores de red, para que puedan programar las políticas y filtros dinámicamente.

Descripción del servidor proxy

El servidor proxy deberá atender en dos puertos diferentes, uno para las conexiones administrativas y el otro al que se conectarán los navegadores para hacer los pedidos HTTP. Para ambos casos, el servidor será una aplicación *multithreaded* por cada conexión que acepta; ésto significa que cada vez que un navegador o un administrador se conecta al servidor, se le asignará un hilo de ejecución independiente. Las estructuras compartidas por los threads deberán estar correctamente mutuo-excluídas.

Al iniciar la ejecución del servidor, la misma podrá ser usando parámetros por defecto, o los especificados por el usuario. Por defecto, la IP será la `127.0.0.1`, el puerto de datos el `5555` y el puerto de administración el `6666` (este último no será parametrizable). A continuación se muestra un ejemplo de invocación:

```
lab2$ ./proxyserver 127.0.0.1 9876
DEBUG:      Iniciando servidor...

DEBUG:      Posibles invocaciones del servidor:
DEBUG:            ./server
DEBUG:            ./server IP
DEBUG:            ./server IP PUERTO
DEBUG:      IP: 127.0.0.1     PUERTO: 9876
DEBUG:      ./proxyserver: eperando conexiones...
```

Manejo de peticiones HTTP

Para la implementación del servidor proxy, se tendrán como referencia las RFCs 1945 y 2616, que definen HTTP/1.0 y HTTP/1.1 respectivamente. En esos documentos están especificados los encabezados y sus significados, los delimitadores para cada encabezado y el formato de un pedido o respuesta.

El funcionamiento esperado en cada nueva conexión con un pedido HTTP por parte del navegador es:

1. Es atendida en un hilo de ejecución independiente.
2. El pedido HTTP recibido, será analizado y modificado[1] si es necesario (los encabezados), antes de enviar esa petición al servidor web de destino.
    1. Si es un método GET y está denegado administrativamente, retornar únicamente el mensaje de error (con el código adecuado).
    2. Si es un método POST y está denegado administrativamente, retornar únicamente el mensaje de error (con el código adecuado).
    3. Si no es uno de los metodos definidos en HTTP/1.0, retornar únicamente el mensaje de error (con el código adecuado).
    4. Si en la URL aparece alguna de las palabras definidas con el comando `addDUW`, retornar únicamente el mensaje de error (con el código adecuado).
3. En el caso que ninguna de las políticas y filtros sean aplicables, se establece conexión con el servidor web (cuya identificación se debe obtener de los encabezados HTTP) y se envía el pedido modificado.
4. Se espera la respuesta, y se analiza si existe dentro del contenido alguna de las palabras que se quiere denegar. En caso que existan, esas palabras son eliminadas del contenido y se envía lo restante como respuesta a la petición del navegador Web.
5. Una vez que ya fueron obtenidos todos los datos, y fueron enviados (con modificaciones) al navegador, se cierra la conexión.

Interfaz de administración del servidor proxy

La interfaz de administración será basada en texto (comandos) y se accederá a través de el comando `telnet`. Desde una sesión de administración, se detallan a continuación los comandos válidos para el proxy, y su significado:

`addDUW PALABRA` – Agrega una palabra a la lista de palabras prohibidas en la URL. Por ejemplo, ejecutar el comando `addDUW fing` tendrá el efecto que si un navegador solicita la URL `http://www.fing.edu.uy/`, el servidor proxy retornará un mensaje indicando que esa página está denegada (enviando también el código HTTP adecuado).

[1] Los valores de algunos encabezados de las peticiones HTTP cambian cuando se tiene conexión directa al servidor a lo que se ve cuando se conectan a través de un proxy. Se recomienda hacer dos pedidos (uno con proxy y otro sin proxy), capturarlos con un sniffer y ver esa direrencia referida a las URLs y el upstream host.

`addDW PALABRA` – Agrega una palabra a la lista de palabras prohibidas dentro del contenido del recurso. Por ejemplo, ejecutar el comando `addDW guerra` tendrá el efecto de eliminar esa palabra del contenido de la página.

`listDUW` – Lista las palabras que están configuradas para filtrar URLs.

`listDW` – Lista las palabras que están configuradas para no ser mostradas dentro del contenido.

`deleteDUW PALABRA` – Elimina una palabra de las que están configuradas para filtrar URLs

`deleteDW PALABRA` – Elimina una palabra de las que están configuradas para no ser mostradas dentro del contenido.

`denyPOST` – Hace que todas las peticiones que usen el método POST sean denegadas administrativamente.

`allowPOST` – En caso de encontrarse inhabilitado el uso del método POST (por ejemplo por haber ejecutado el comando `denyPOST`), vuelve a habilitar su uso.

`denyGET` – Idem `denyPOST` pero para el método GET.

`allowGET` – Idem `allowPOST` pero para el método GET.

`quit` – cierra la sesión de administración.

Herramientas

El lenguaje de programación será C/C++, pero solamente se podrá hacer uso de librerías estándar que implementan tipos básicos como `string`, `vector`, `list`, etc.

El obligatorio se desarrollará en un entorno GNU/Linux, para lo cual se dispone de máquina virtual. Este entorno tiene las herramientas necesarias ya instaladas, tales como:

- Navegador Web Mozilla
- telnet
- Wireshark [3]

NOTAS:

Se recomienda el uso de la herramienta `Wireshark`, durante el desarrollo y *debug* de la solución. Es una herramienta fundamental al momento de estudiar el tráfico que genera el `telnet` (para descubrir delimitadores) al igual que para el HTTP (encabezados y delimitadores).

Si bien el proxy es HTTP/1.0, nada impide que podamos pasar (solamente para los métodos que serán permitidos) tráfico señalizado como HTTP/1.1. En este sentido, está permitido que igualmente la solución maneje las conexiones como se especifica en HTTP/1.0, es decir, un pedido/respuesta por conexión, y luego cerrarla.

La modificación de cualquiera de de los encabezados HTTP está permitida, siempre que sea necesaria para la resolución del obligatorio.

Referencias y Bibliografía Recomendada

[1] Berners-Lee, T.; Fielding, R. & Frystyk, RFC 1945, H. Hypertext Transfer Protocol – HTTP/1.0 *IETF,* **1996**

[2] Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P. & Berners-Lee, T., RFC 2616, Hypertext Transfer Protocol – HTTP/1.1 *IETF,* **1999**

[3] Analizador de Tráfico Wireshark. En línea: `http://www.wireshark.org/`. Última visita: Marzo 2011.