Redes de Computadoras

Capa de Aplicación

Obligatorio 1 – 2012

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 (foros, 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 laboratorio consiste en un único archivo **lab1.tar.gz** que deberá a su vez contener los siguientes archivos:

- **lab1.pdf**, donde se documenta todo lo solicitado en el presente obligatorio. El mismo deberá incluir las decisiones de diseño, máquinas de estado y pseudocódigo, y finalmente la documentación de la implementación
- Los archivos correspondientes a la implementación de la aplicación propuesta.
- Scripts para la construcción del ejecutable (makefile).

El archivo **lab1.tar.gz** deberá ser generado utilizando la herramienta *GNU tar* y compresión *gzip*. Otros formatos (bz2, rar, zip, cab, jar, etc.) no son válidos y serán rechazados, con la consecuente pérdida del curso para todos los integrantes del grupo.

La entrega se realizará a través de un recurso habilitado para tal fin en la plataforma EVA, que será oportunamente avisado por dicho medio.

Fecha de entrega

Los trabajos deberán ser entregados antes del **domingo 16 de setiembre 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.

El laboratorio se realizará en la máquina virtual (VM) **BackTrack3** disponible en las computadoras de las salas linux, en el directorio `/home/redes`. Puede trabajar con esta VM en cualquier entorno, pero recuerde que la defensa de la tarea se se realizará en las salas de FING.

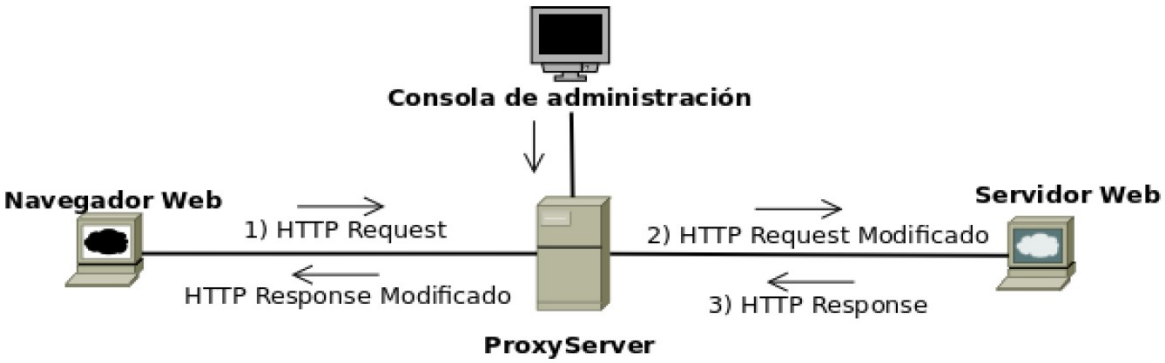
Objetivo del trabajo

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 en C/C++, fundamentales en el desarrollo de aplicaciones para redes de computadoras.

Consolidar los conocimientos acerca del manejo de protocolos, mediante 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, permitirá el *caching* de contenido de los tipos predefinidos y bloqueará contenido o archivos de tamaños superiores a un máximo configurable.



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.
No se solicita la implementación completa de los diferentes métodos soportados, sino, solamente los métodos GET y POST definidos en HTTP/1.0. Todo otro método que pueda recibir será considerado inválido, e informado de tal forma al navegador.

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. Se usará como agente de usuario del protocolo HTTP un navegador estándar.

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: esperando 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 es recibido, será analizado y procesado de acuerdo a:
   a) Si es un método GET y está denegado administrativamente, retornar únicamente el mensaje de error (con el código adecuado, por ejemplo, un tipo de imágen prohibido o un archivo demasiado grande). Los únicos filtros a implementar serán aquellos relativos al tamaño de los objetos transferidos.
   b) Si es un método GET y no se lo tiene en memoria, se deberá realizar la conexión al servidor correspondiente, obtener el objeto, entregárselo al cliente y si su tamaño es menor al tamaño máximo de objeto cacheable, almacenarlo en memoria.
   c) Si es un método POST y está denegado administrativamente, retornar únicamente el mensaje de error (con el código adecuado).
   d) Si no es uno de los metodos definidos en HTTP/1.0, retornar únicamente el mensaje de error (con el código adecuado).
3. Se deberá mantener un máximo de objetos en memoria, y los más viejos serán eliminados en caso de peticiones nuevas (política “Least Recently Used” - LRU).
4. Se deberán eliminar objetos expirados de la cache y no se entregarán a los usuarios.
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:

`show run`	muestra los valores actuales y datos estadísticos como memoria total utilizada, cantidad de objetos, cantidad de requests atendidos y cantidad de objetos cacheados entregados.
`purge`	borra todos los objetos cacheados en memoria.
`set max_object_size XXX`	determina el mayor tamaño de objeto transferible por este proxy, expresado en KB (por defecto, su valor será de 10MB)
`set max_cached_object_size YYY`	determina el mayor tamaño de objeto cacheable por este proxy, expresado en KB (por defecto, su valor será de 100KB)
`set max_object_count ZZZ`	determina la cantidad máxima de objetos a almacenar en RAM (por defecto, su valor será de 200 objetos)
`quit`	cierra la sesión de administración.

Herramientas

El lenguaje de programación será C/C++, pero solamente se podrá hacer uso de bibliotecas estándar que implementan tipos básicos como string, vector, list, etc. Las bibliotecas que use deberán estar acordadas con su docente de monitoreo.

El obligatorio se desarrollará en un entorno GNU/Linux, para lo cual se dispone de la 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.

Muchos servidores web (también el servidor web de Fac. de Ingeniería) requieren del atributo *user-agent* definido en el request. Recomendamos tener esto presente durante las pruebas si encuentra comportamientos inesperados.

Dado que no se implementará la totalidad del protocolo HTTP/1.0, se suministrará una URL para pruebas de este obligatorio. De todas formas, luego de implementado, se alienta a los estudiantes a ver los límites de su desarrollo en Internet.

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: Agosto 2012.