INSTITUTO TECNOLÓGICO de BUENOS AIRES - ITBA

SEGUNDO CUATRIMESTRE 2013

PROTOCOLOS DE COMUNICACIÓN

TRABAJO PRÁCTICO ESPECIAL

Alumnos:

Fontanella De Santis, Teresa

Legajo 52.455

Pagnoni, Agustín

Legajo 52.118

Pomerantz, Alan

Legajo 51.233

Responsables Académicos:

VALLÉS, Santiago Raúl

SOTUYO DODERO, Juan Martín

CODAGNONE, Juan Francisco

ÍNDICE

PROTOCOLOS DE COMUNICACIÓN
Instituto Tecnológico de Buenos Aires - ITBA
2º Cuatrimestre
2013

Alumnos:
FONTANELLA DE SANTIS, Teresa (Leg. 52455)
PAGNONI, Agustín (Leg. 52118)
POMERANTZ, Alan (Leg. 51233)

OBJETIVO

El objetivo del presente trabajo consiste en implementar un servidor proxy para el protocolo HTTP versión 1.1 (Hypertext Transfer Protocol) (RFC2616) que pueda ser usado por User agents como Mozilla Firefox, Internet Explorer y Google Chrome para navegar por Internet.

1. DESCRIPCIÓN Y ANÁLISIS DE LOS RFC UTILIZADOS

Los estándares RFC que serán tomados en cuenta para este trabajo son:

1) RFC 2616 ("Hypertext Transfer Protocol – HTTP/1.1"):

   Este documento define el protocolo HTTP versión 1.1, el cual será utilizado por el proxy para poder hablar tanto con los clientes (User Agents), como con los servidores. Si bien todo el documento es importante (porque definen las distintas funcionalidades que pueden utilizarse, y que, por eso, el programa tiene que saber cómo se aplican), se enfatizará en:

   - Conexiones persistentes (sección 8): Es decir, conexiones que duren más de un request. Se consideraron necesarias para poder mantener la conexión entre el proxy y el servidor, y que éste pueda enviarle las respuestas al primero, porque éste es transparente (el servidor envía las respuestas al cliente sin saber que éstas pasan por un proxy). Además, reusan conexiones TCP, lo cual mejora a la performance y la eficiencia.

   - "Status codes" (sección 10): Para poder reportar los eventuales fallos que se produzcan en las conexiones (el hostname no es válido, por ejemplo).

   - "Media types" (sección 3.7): Para poder detectar los distintos tipos de datos que se transfieren y efectuar la transformación a formato 133t (donde corresponda).

   - Estructura de los mensajes (secciones 4 – 6) y métodos definidos (sección 9): Para poder parsear correctamente los mensajes del cliente para el servidor y viceversa.

2) RFC 2045 (" Multiporpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies"):

   Ya que HTTP puede manejar el protocolo MIME para especificar el formato del cuerpo de un mensaje, el proxy también debe conocerlo (especialmente para la transformación del texto plano).

Alumnos:
FONTANELLA DE SANTIS, Teresa (Leg. 52455)
PAGNONI, Agustín (Leg. 52118)
POMERANTZ, Alan (Leg. 51233)

3) RFC 3117 (“On the Design of Application Protocols”):

Si bien no es un estándar, menciona aspectos muy importantes a tener en cuenta a la hora de definir un protocolo. Conceptos como “framing”, ”encoding”, “reporte de errores”, “asyncronism”, “escalabilidad”, “eficiencia”, “simplicidad”, “extensibilidad”, ”principio de la robustez de Postel”, se tendrán en cuenta para el armado del protocolo para los puntos 1.11 y 1.12 del enunciado del Trabajo Práctico Especial.

2. DESCRIPCIÓN DEL PROTOCOLO PROPUESTO

En las secciones 1.11 y 1.12 del enunciado, se pide la definición de un protocolo que permita realizar monitoreo remoto de la actividad del proxy y configurarlo. El mismo se llama PDC v1.0 y se lo describe a continuación:

Características:

- Se basa en la arquitectura cliente/servidor.
- Request/response.
- “Stateless” (sin estados): cada request es independiente del anterior.
- Orientado a texto.
- Estructura general: cada mensaje (sea request o response) consta de una primera línea, una segunda línea, una serie de headers seguida por un caracter de nueva línea (“\n”) y, de ser necesario, datos.
- El cliente se conecta al servidor por medio del puerto 9090.

Request:

Sintaxis:

[OPERATION] [PARAM] PDC/ [VERSION]
[HEADERS]
\n
[DATA]
\n

OPERATION= (“GET”|”ADD”|”DEL”)
HEADERS= lista de headers de la forma: [HEADER_NAME]: [VALUE]\n
HEADER_NAME = (”Authentication”)
Si OPERATION=”GET” => PARAM = (“ACCESSES” |”TXBYTES” |”HYSTOGRAM”)
Si OPERATION=”ADD” o OPERATION=”DEL” => PARAM = (“FILTER”)

Alumnos:
FONTANELLA DE SANTIS, Teresa (Leg. 52455)
PAGNONI, Agustín (Leg. 52118)
POMERANTZ, Alan (Leg. 51233)

Análisis

La primera línea indica la operación a realizar, un parámetro acorde a la misma, y el protocolo y versión con la que se está hablando (PDC y la versión correspondiente, que en este caso es 1.0).

Se definen las siguientes operaciones:

- GET

  Se obtiene información del servidor. No requiere autenticación. Los posibles valores de PARAM son:

  - ACCESSES: Para obtener la cantidad de accesos al servidor.
  - TXBYTES: Para obtener la cantidad de bytes transferidos por el servidor.
  - HYSTOGRAM: Para obtener una serie de datos con las que se puede construir el histograma de status codes.

- ADD

  Agrega reglas de configuración al servidor. Se necesita autenticación (expresada en el header "Authentication"). El único valor posible de PARAM es FILTER, que sirve para agregar filtros al servidor.

- DEL

  Borra reglas de configuración al servidor. Al igual que "ADD", necesita autenticación y el único valor posible de PARAM es FILTER, que sirve para agregar filtros al servidor.

El header puede ser:

- Authentication: Sirve para autenticar a un usuario determinado. Es de la forma: [username] ; [password]. Es optativo para la operación GET, pero es requerido para ADD y DEL. No tiene un valor default.

El contenido de la sección DATA, depende de OPERATION:

- OPERATION= "GET"

  La sección DATA debe estar vacía.

- OPERATION= "ADD"

  La sección DATA debe contener una o más líneas de la siguiente forma:

  [VALIDATION] [URL]

Alumnos:
FONTANELLA DE SANTIS, Teresa (Leg. 52455)
PAGNONI, Agustín (Leg. 52118)
POMERANTZ, Alan (Leg. 51233)

donde VALIDATION=("YES"|"NO") significa si la url es aceptada o no y URL es el sitio web a filtrar. La url a filtrar no debe de existir en los filtros del servidor.

- OPERATION= "DEL"

  La sección DATA debe contener una o más líneas de la siguiente forma:

  [URL]

  donde URL es el sitio web que tiene asignado el filtro.

Response

Sintaxis

PDC/ [VERSION] [STATUS CODE] [TEXT CODE]
[HEADERS]
\n
[DATA]
\n

HEADERS= lista de headers de la forma: [HEADER_NAME]: [VALUE] \n
HEADER_NAME = ("Encoding" |"Content-Type")

Análisis

La primera línea indica el protocolo y versión con la que se está hablando (que serán los mismos que en el request), y el status code y su correspondiente texto (text code) de la operación realizada.
A continuación, se definen los status code con su respectivo texto (su convención está basada en la usada en HTTP):

- Exitoso 2.x.x

  Los status code que comiencen con "2" indican que la operación se realizó con éxito. Sus posibles variantes son:

  - 200 OK: Se utiliza cuando se envían datos (operación GET).
  - 204 NO CONTENT: Se utiliza cuando DATA está vacío (operaciones ADD y DEL).

- Errores del cliente 4.x.x

  Los status code que empiecen con un "4" señalan que la operación no pudo realizarse porque el request está mal formado. Corresponde al cliente arreglar esos errores.

  - 400 BAD REQUEST: Error de sintaxis en el request.

- 401 UNAUTHORIZED: La operación solicitada requiere autenticarse y falta el header "Authentication".
- 404 NOT FOUND: PARAM no es válido para la OPERATION dada.
- 408 REQUEST TIMEOUT: El cliente no realizó un request en el tiempo en que el servidor lo estaba esperando.
- 420 CORRUPTED DATA: Los datos enviados están corruptos o no son los adecuados para el tipo de operación que se quiere realizar.

- Errores del servidor 5.x.x

  Los status code que comiencen con un "5" hacen referencia a errores que se produjeron en el servidor y que evitaron que la operación pudiera llevarse a cabo.
  - 500 INTERNAL SERVER ERROR: Se produjo un error no esperado en el servidor que impidió ejecutar la operación.
  - 501 NOT IMPLEMENTED: La operación solicitada no se implementó aún para ningún tipo de PARAM.
  - 503 SERVICE UNAVAILABLE: El servidor no está disponible. Se usa para cuando se realiza mantenimiento o hay una sobrecarga de requests.

Los headers pueden ser:

- Encoding: De significado similar al header homónimo en HTTP, los valores que admite son: "base64" o "quoted-printable".
- Content-Type: Parecido al header de HTTP con el mismo nombre, el único valor que admite es "text/plain" (puede definirse también un charset de la misma manera que en HTTP). Es obligatorio para la operación GET y prohibido para las demás.

El contenido de la sección DATA, depende de la operación definida en el request:

- OPERATION= "GET"

  Depende del tipo de información que se esté pidiendo (dada por PARAM en el request). Si PARAM="HYSTOGRAM", los datos serán pares clave-valor (uno por línea), de la siguiente manera:

  [HTTP_STATUS_CODE] : [NÚMERO]

  donde HTTP_STATUS_CODE se refiere a un status code de HTTP (y no del protocolo PDC); y NÚMERO, a la cantidad de responses que retornaron dicho status code.

Alumnos:
FONTANELLA DE SANTIS, Teresa (Leg. 52455)
PAGNONI, Agustín (Leg. 52118)
POMERANTZ, Alan (Leg. 51233)

En cualquier otro caso, los datos contendrán una línea con el número en cuestión (sea para cantidad de accesos o bytes transferidos).

- OPERATION= ”ADD” o OPERATION= ”DEL”

  La sección DATA debe estar vacía.

Ejemplo 1:

Request

GET HYSTOGRAM PDC/ 1.0
\n
\n

Response

PDC/ 1.0 200 OK
Encoding: base64
Content-Type: text/plain; charset=ISO-8859-4
\n
200 : 100
404 : 1000
\n

Ejemplo 2:

Request

ADD FILTER PDC/ 1.0
\n
NO www.xxx.com
\n

Response

PDC/ 1.0 401 UNAUTHORIZED
\n
\n

Alumnos:
FONTANELLA DE SANTIS, Teresa (Leg. 52455)
PAGNONI, Agustín (Leg. 52118)
POMERANTZ, Alan (Leg. 51233)

3. POSIBLES PROBLEMAS Y DIFICULTADES DETECTADOS

Si bien los RFC son explicativos, hay cuestiones que no detallan y que pueden generan ciertos problemas. Una de ellas es qué tipo de error se retorna cuando se ingresa una url que no se puede resolver.

4. APLICACIONES A UTILIZAR EN DESARROLLO Y TESTING

Para realizar el desarrollo, utilizaremos inicialmente al navegador Google Chrome, configurado de manera particular que ingrese a nuestro proxy, en la etapa en que el proxy no sea transparente.

Cuando lo sea, mediante algunas reglas de firewall convenientes para la red, dicha configuración será innecesaria.

Se creará un ambiente de testing pertinente para poder simular múltiples conexiones y medir la performance y robustez del sistema, a la hora de soportar numerosos enlaces distintos.

5. CASOS DE PRUEBA TENTATIVOS A REALIZAR

Se intentará realizar una simulación de N conexiones simultáneas, con el objetivo de medir la capacidad de manipulación de conexiones que tenga el proxy, dada la arquitectura seleccionada.

Probar que efectivamente el proxy, en caso de no tener ningún filtro, deje pasar requests del tipo GET, POST y HEAD. En caso contrario que envíe el código de error correspondiente. Se probará enviar varios requests con distinta frecuencia al servidor proxy, por ejemplo un Nginx, que sirva un recurso cuyos valores sean conocidos para probar la correctitud del procesamiento de los request/response que atraviesen al proxy. En el caso de accesos, se medirá en base a la cantidad de sitios accedidos diferentes. La cantidad de bytes transferidos se chequea contra los datos del recurso servido por el servidor proxy, contemplando la cantidad de veces que se accedió. Utilizando dos computadoras que lo corran, dada la configuración en la red para que el flujo de request/response atraviese ambos proxies, se harán las mismas pruebas que para un proxy, haciendo prueba del funcionamiento del encadenamiento de proxies.

Alumnos:
FONTANELLA DE SANTIS, Teresa (Leg. 52455)
PAGNONI, Agustín (Leg. 52118)
POMERANTZ, Alan (Leg. 51233)

6. FUENTES CONSULTADAS

- ***Apuntes de la Cátedra.***
- ***Consultas a la Cátedra***.