Índice

Descripción de protocolos y aplicaciones

Aplicación:

La aplicación se conecta al servidor a través del protocolo SCTP.
La conexión de la aplicación no es bloqueante. En caso de no poder establecerse la conexión, un timeout de 5 segundos cierra la aplicación.

Los comandos que recibe la aplicación son los siguientes:

- Ip y puerto: (obligatorio, siempre primer parámetro)
  - x.x.x.x:port (soporta únicamente ipv4)
- Métricas: (opcionales, pueden estar desordenados)
  - -m1 bytes transferred
  - -m2 currently serving
  - -m3 max served concurrently
  - -m4 requests served
- Configuraciones: (opcionales, pueden estar desordenados)
  - -c0 media/type filters reset for specified media type
  - -c1 media/type toupper filter applied to specified media type
  - -c2 media/type echo filter applied to specified media type
  - -c3 media/type duplicate filter applied to specified media type
  - -c3 media/type leet filter applied to specified media type

Si se corre la aplicación se debe pedir una métrica o una configuración obligatoriamente.
Si se pide una configuración se debe especificar obligatoriamente el media type a continuación.

Una vez abierta la aplicación se debe especificar una contraseña de 8 caracteres.

Protocolo:

Se maneja pasando strings.

Request:

1. Los primeros 8 bytes corresponden a la contraseña
2. El próximo byte especifica el tipo de request:
    - ‘0’ se está pidiendo una métrica
    - ‘1’ se está pidiendo una configuración
3. El siguiente byte indica el tipo de métrica/configuración
    - Métrica:
        - ‘1’ bytes transferred
        - ‘2’ currently serving
        - ‘3’ max served concurrently
        - ‘4’ requests served
    - Configuración:
        - ‘0’ filters reset for specified media type
        - ‘1’ toupper filter applied to specified media type
        - ‘2’ echo filter applied to specified media type
        - ‘3’ duplicate filter applied to specified media type
        - ‘4’ leet filter applied to specified media type
4. Si se especificó un tipo de configuración, a continuación se pasa el media type, con longitud variable y terminado de un espacio
5. Se vuelve al punto 2.

Response:

1. El primer byte especifica el tipo de response:
    - ‘0’ métrica
    - ‘1’ configuración
    - ‘2’ error de métrica
    - ‘3’ contraseña incorrecta
2. Si es una métrica se pasa el tipo de métrica y a continuación 64 bytes con los datos de la métrica
3. Si es una configuración se pasa el tipo de configuración
4. Si es una error de métrica se pasa el tipo de métrica que fallo
5. Si es contraseña incorrecta no se envia nada más y se termina el paquete
6. Si no es contraseña incorrecta se vuelve al punto 1.

Problemas encontrados

Primero comenzamos haciendo threads y nuestra intención era que por cada host tengamos un thread corriendo para atender todas sus requests. El problema era que los browsers no persistían la conexión necesariamente y se terminaba haciendo un thread por cada request, lo cual no era eficiente. Para solucionar esto decidimos usar el selector provisto por la cátedra.

Otro problema encontrado fue con el POST o PUT grande. Al pensar al server solo lo hicimos para que una vez que se mande toda la request, se comienza a leer la response, pero este no era el caso. Para solucionar esto….

Asimismo, nos costó mucho lograr entender el código provisto por la cátedra y crear los buffers para ir pasando la información. Comenzamos no utilizando copy pero luego comprendimos que lo debíamos usar y con un mismo buffer y los punteros de read y write correspondientes.

Para la aplicación, de vez en cuando la función sctp_recv() nos tiraba el error *Invalid Argument*. Es por esto que decidimo usar en vez de sctp_recv() y sctp_send(), recv() y send().

Limitaciones

Algunas de las limitaciones encontradas son:

- Si se reciben dos media-types en un mismo header content-type, las transformaciones correspondientes a estos media-types no se ejecutarán.
- No se transforma el contenido que está comprimido.
- No se puede aplicar una transformación a todos los media-types.
- Solo se ejecuta una transformación por media-type.
- Las transformaciones son fijas, es decir, el usuario no puede configurar nuevas transformaciones.
- Los parámetros utilizados en la aplicación no son del todo claros.
- El máximo de conexiones es aproximadamente 500 debido a que el selector sólo acepta 1024 file descriptors y hay algunos que se utilizan para transformaciones.

Posibles extensiones

Algunas extensiones posibles pueden ser:

- Hacer muchas transformaciones por cada media-type.
- Que el usuario pueda configurar sus transformaciones.
- Soportar todos los métodos http.
- Hacer una aplicación interactiva para la configuración y monitoreo.
- Pasar hostname en vez de ip en el client.
- Que la contraseña cuando se ingresa al cliente no se escriba en la terminal.
- Tener buffer que almacena al host variable.
- Hay un leak de memoria en caso de que se mate al server cuando el thread que resuelve la dirección del host está corriendo. Como se usa pthread_detach() y este sólo libera los recursos una vez que termina su ejecución. Si esta no termina, sus recursos no se liberaran. Se deberían liberar estos recursos.
- Incluir más métricas

Conclusiones

Consideramos que es un trabajo práctico complejo pero finalmente logramos solucionar los problemas encontrados. Requiere de mucho tiempo y dedicación pero estamos orgullosos del trabajo final. Es por esto que también consideramos que tendríamos que haber empezado antes.

Las presentaciones fueron útiles para saber si lo que habíamos pensado estaba bien y también para realizar los cambios necesarios en caso contrario. Además, ver a los otros grupos y sus correcciones nos permitió considerar factores que previamente no habíamos tenido en cuenta. Sin embargo, como nosotros no teníamos mucho planeado en ese momento, hubiese sido más productivo si este no fuese el caso.

Hay varias mejoras que se pueden hacer pero creemos que teniendo en cuenta el tiempo provisto es lo mejor que podemos hacer. Algo que deberíamos haber hecho fueron los ejercicios introductorios ya que podría haber ayudado ya tener ciertas cosas resueltas.

Ejemplos de prueba

Los ejemplos de transformaciones y métricas se encuentran en la sección Ejemplos de configuración y monitoreo.

Archivo estático:

```
parallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$ curl -i -s http://localhost:9090/1.is
o -x localhost:8090|head -n7
HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 14:39:46 GMT
Last-Modified: Mon, 11 Jun 2018 00:32:50 GMT
Accept-Ranges: bytes
Vary: Accept-Encoding, User-Agent
Content-Length: 2018508800
```

```
parallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$ curl -i -s http://localhost:9090/leet
.txt -x localhost:8090|head -n7
HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 14:40:31 GMT
Last-Modified: Sat, 10 Jun 2017 22:45:23 GMT
Content-Type: text/plain
Accept-Ranges: bytes
Vary: Accept-Encoding, User-Agent
Content-Length: 165000
```

Archivos chunked:

```
parallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$ curl -i -s http://localhost:9090/api/
chunked/leet -x localhost:8090|head
HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 14:42:15 GMT
Content-Type: text/plain
Vary: Accept-Encoding, User-Agent
Transfer-Encoding: chunked

aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
```

Chunked para payloads mayores a 4096:

```
parallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$ curl -i -s -X GET http://localhost:90
90/api/other -x localhost:8090
HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 14:44:00 GMT
Content-Type: text/plain
Vary: Accept-Encoding, User-Agent
Content-Length: 10

hola mundoparallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$
```

Método PUT:

```
parallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$ curl --data-binary 'hola' -X PUT http
://localhost:9090/api/other -x localhost:8090 -i
HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 14:50:05 GMT
Content-Type: application/octet-stream
Content-Length: 4

holaparallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$
```

Dos requests en la misma línea y tratar de conectarse al proxy:

```
parallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$ curl -i http://localhost:8090/api/oth
er/bu http://localhost:9090/api/other/bu -x localhost:8090
HTTP/1.1 409 Conflict
Content-Length: 0

HTTP/1.1 204 No Content
Date: Tue, 12 Jun 2018 14:53:37 GMT
```

Método POST grande:

```
parallels@parallels-vm:~/Downloads/http-latest/webapp$ time curl -s --data-binary @1.iso -X POST http://localhost:9090/api/other -x l
ocalhost:8090| sha256sum
524bd959dae09ad6fc8e0476ea478700d89f82ec5795d0b1a7b873613f3f26ac  -

real    0m19.838s
user    0m12.424s
sys     0m4.096s
parallels@parallels-vm:~/Downloads/http-latest/webapp$ time curl -s --data-binary @1.iso -X POST http://localhost:9090/api/other| sha
256sum
524bd959dae09ad6fc8e0476ea478700d89f82ec5795d0b1a7b873613f3f26ac  -

real    0m17.649s
user    0m11.692s
sys     0m4.916s
```

Método GET condicional:

```
parallels@parallels-vm:~/Downloads/http-latest/webapp$ curl -I http://localhost:9090/leet.txt -x localhost:8090
HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 15:00:11 GMT
Last-Modified: Sat, 10 Jun 2017 22:45:23 GMT
Content-Type: text/plain
Accept-Ranges: bytes
Content-Length: 165000
```

Anidamiento de body de dos requests:

```
parallels@parallels-vm:~/Downloads/http-latest/webapp$ curl -s http://www.mdzol.com/robots.txt http://mirrors.ibiblio.org/robots.txt
-x localhost:8090
User-agent: *
Allow: /
Disallow: /tools/hit.php
Disallow: /lib/1x1.gif
Disallow: /seccion/list/ajax/*
Disallow: /1/seccion/list/ajax.vnc
Disallow: /ranking/*
Disallow: /buscar/*
Disallow: /1/buscar/*
Disallow: /nota/shares/*
Sitemap: https://www.mdzol.com/files/mdzsitemap/sitemap.xml.gz
User-agent: *
Crawl-delay: 2
Disallow:
```

Server no resoluble:

```
parallels@parallels-vm:~/Downloads/http-latest/webapp$ curl -i http://bar/ -x localhost:8090
HTTP/1.1 502 Bad Gateway
Content-Length: 0
```

```
parallels@parallels-vm:~/Downloads/http-latest/webapp$ curl -I http://localhost:9090/api/other http://localhost:9090/api/other -x loc
alhost:8090
HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 14:59:03 GMT
Content-Type: text/plain
Content-Length: 10

HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 14:59:03 GMT
Content-Type: text/plain
Content-Length: 10
```

Soporte método DELETE:

```
parallels@parallels-vm:~/Downloads/http-latest/webapp$ curl -i -X DELETE http://localhost:9090/api/other -x localhost:8090
HTTP/1.1 204 No Content
Date: Tue, 12 Jun 2018 15:03:53 GMT
```

Soporte método PUT grande:

```
parallels@parallels-vm:~/Downloads/http-latest/webapp$ time curl -s --data-binary @1.iso -X PUT http://localhost:9090/api/other -x lo
calhost:8090 | sha256sum
^C

real    0m1.009s
user    0m0.008s
sys     0m0.592s
parallels@parallels-vm:~/Downloads/http-latest/webapp$ time curl -s --data-binary @1.iso -X PUT http://localhost:9090/api/other -x lo
calhost:8090 | sha256sum
524bd959dae09ad6fc8e0476ea478700d89f82ec5795d0b1a7b873613f3f26ac  -

real    0m22.857s
user    0m12.132s
sys     0m4.100s
```

Guía de instalación

Clonar el repositorio:

>> git clone https://MartinaArco@bitbucket.org/itba/pc-2018-07.git

Ingresar al mismo (~/pc-2018-07). Ejecutar:

>> make

Luego correr en una terminal:

>> ./server

Para ejecutar al cliente, dirigirse al mismo directorio en otra terminal y correr:

>> ./client

Instrucciones para la configuración

Para poder configurar los puertos e interfaces del servidor, tanto en tcp como sctp:

-l	ip a escuchar en tcp (si no se asigna se escucha en todas)
-p	puerto a escuchar en tcp (default 8090)
-L	ip a escuchar en sctp (si no se asigna se escucha en todas)
-P	puerto a escuchar en sctp (default 9095)

Si estos parámetros no se pasan, se utiliza el default:
ip de tcp: escuchar en todos
puerto de tcp: 8090
ip de sctp: escuchar en todos
puerto de sctp: 9095

Ejemplos de configuración y monitoreo

```
parallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$ ./client 127.0.0.1:9095 -m1 -m2 -m3 -
m4

Insert the password: tpprotos

KB TRANSFERRED:                         388             (KB)

CURRENTLY SERVING:                      1

MAX SERVED CONCURRENTLY:                1

REQUESTS SERVED:                        4
```

En las siguientes secuencias se pueden ver ejemplos de configuraciones para transformaciones.

Secuencia 1

Se hace un curl a un html:

```
parallels@parallels-vm:~$ curl -x localhost:8090 google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
```

Se activa transformación de tipo 4 (ver página 2):

```
parallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$ ./client 127.0.0.1:9095 -c1 text/plai
n -c4 text/html

Insert the password: tpprotos

Configuration 1 applied
Configuration 4 applied
```

Ahora el html viene transformado:

```
parallels@parallels-vm:~$ curl -x localhost:8090 google.com
<HTML><HEAD><m3t4 http-3qu1v="c0nt3nt-typ3" c0nt3nt="t3xt/html;ch4rs3t=utf-8">
<TITLE>301 M0v3d</TITLE></HEAD><BODY>
<H1>301 M0v3d</H1>
Th3 d0cum3nt h4s m0v3d
<A HREF="http://www.g00gl3.c0m/">h3r3</A>.
</BODY></HTML>
```

Se desactiva la transformación

```
parallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$ ./client 127.0.0.1:9095 -c1 text/plai
n -c0 text/html

Insert the password: tpprotos

Configuration 1 applied
Configurations reseted
```

El response viene como estaba originalmente:

```
parallels@parallels-vm:~$ curl -x localhost:8090 google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
```

Secuencia 2

Se hace un curl a un text/plain:

```
parallels@parallels-vm:~$ curl -s -i -x localhost:8090 http://localhost:9090/leet.txt
HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 13:05:12 GMT
Last-Modified: Sat, 10 Jun 2017 22:45:23 GMT
Content-Type: text/plain
Accept-Ranges: bytes
Vary: Accept-Encoding, User-Agent
Transfer-Encoding: chunked

4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
```

Se activa la transformación de tipo 4 a text/plain:

```
parallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$ ./client 127.0.0.1:9095 -c4 text/plai
n

Insert the password: tpprotos

Configuration 4 applied
```

Se hace un curl a algo comprimido y se ve que no se realiza la transformación:

```
parallels@parallels-vm:~$ curl --compressed -s -i -x localhost:8090 http://localhost:9090/api/chunked/leet
HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 13:03:28 GMT
Content-Type: text/plain
Vary: Accept-Encoding, User-Agent
Content-Encoding: gzip
Transfer-Encoding: chunked

aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
```

Se hace un curl a algo chunked y se ve que se realiza la transformación:

```
parallels@parallels-vm:~$ curl -s -i -x localhost:8090 http://localhost:9090/api/chunked/leet
HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 13:04:24 GMT
Content-Type: text/plain
Vary: Accept-Encoding, User-Agent
Transfer-Encoding: chunked

4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
4310uc 4bcd3fgh1jklmn0pqrstuvxyz
```

Secuencia 3

Se activa la configuración de tipo 1 (si la contraseña no cumple con longitud se reporta):

```
parallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$ ./client 127.0.0.1:9095 -c1 text/html


Insert the password: tpproto
The number of characters for password is 8
Insert the password: tpprotos

Configuration 1 applied
```

Se hace un curl a un text/html y se ve que se aplicó la transformación:

```
parallels@parallels-vm:~$ curl -s -i -x localhost:8090 google.com
HTTP/1.1 301 Moved Permanently
Location: http://www.google.com/
Content-Type: text/html; charset=UTF-8
Date: Tue, 12 Jun 2018 13:16:05 GMT
Expires: Thu, 12 Jul 2018 13:16:05 GMT
Cache-Control: public, max-age=2592000
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked

<HTML><HEAD><META HTTP-EQUIV="CONTENT-TYPE" CONTENT="TEXT/HTML;CHARSET=UTF-8">
<TITLE>301 MOVED</TITLE></HEAD><BODY>
<H1>301 MOVED</H1>
THE DOCUMENT HAS MOVED
<A HREF="HTTP://WWW.GOOGLE.COM/">HERE</A>.
</BODY></HTML>
```

Se hace un curl a un text/plain y se ve que no se aplicó la transformación ya que no está configurada:

```
parallels@parallels-vm:~$ curl -s -i -x localhost:8090 http://localhost:9090/leet.txt
HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 13:15:10 GMT
Last-Modified: Sat, 10 Jun 2017 22:45:23 GMT
Content-Type: text/plain
Accept-Ranges: bytes
Vary: Accept-Encoding, User-Agent
Content-Length: 165000

aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
aeiouc abcdefghijklmnopqrstuvxyz
```

Secuencia 3

Si se ingresa una contraseña incorrecta, no se conecta y reporta:

```
parallels@parallels-vm:~/Desktop/Parallels Shared Folders/Home/Documents/Protocolos/pc-2018-07$ ./client 127.0.0.1:9095 -c1 text/html

Insert the password: protocol

Incorrect password
```

Diseño del proyecto

Concurrencia

Decidimos que un socket cliente siempre va a corresponder a un solo socket origin. Se sirve una request y se cierra la conexión. Las dos estructuras de datos comparten buffers, con la particularidad de que el buffer de escritura de uno es el buffer de lectura del otro. De este modo cuando uno lee, se registra el interés de escritura en el otro, que escribe lo que hay en el buffer hasta que quede vacío. Así se maneja la concurrencia asegurando que solo escribo cuando puedo escribir y sólo leo cuando puedo leer.

Una vez terminado de parsear el request, se lanza un thread DNS y el selector client se bloquea, este thread al terminar notifica al client, que pasa a instanciar un objeto origin, en estado connecting. Cuando se puede escribir en origin, es decir se establecio la conexion, se continua normalmente.

Timeout

El manejo del timeout en caso de falla en conexion o resolucion DNS se hizo modificando el codigo del selector. A la estructura item se le agrego una struct timeval y en cada iteracion se revisa si el item interactua de alguna forma. En ese caso se renueva su timeval. En el caso de que no se use luego de un tiempo determinado, se llama a la funcion que tiene definida como timeout_handler. Esto nos brindo transparencia a la hora de diferenciar comportamiento entre un objeto origin, client o transform.

Transformations

Para las transformaciones se decidio instanciar otros objetos, que compartan buffers con origin y client. De una forma interceptan las escrituras de origin y le dejan en el mismo buffer a client. Client no sabe si se esta haciendo una transformacion. En el caso de que se haga una transformacion, se cambia el header Transfer-Encoding a chunked, para soportar transformaciones que modifiquen tamanios.

Parsers

Para los parsers se decidió utilizar el código provisto por la cátedra (~/socksv5/hello.c). Los parsers consumen de a un byte a la vez ya que se tiene en cuenta que TCP es orientado a conexión. Tanto en request cómo response sólo se parsean los headers, el body directamente se manda al lugar correspondiente.

Request

Para el parseo de requests se utilizaron las siguientes estructuras:

```
struct request_parser {
  struct request *request;
  request_state_t state;
  /*contador auxiliar */
```

```
  uint16_t i;
  /**buffer auxiliar*/
  char* buffer;
};
```

Se utilizó request_parser para guardar el estado del parser a medida que se consumen los caracteres. Allí se guarda una estructura request que se explica más adelante, un estado que es un enum para poder saber en qué sección de la request se encuentra el parser y un contador auxiliar y un buffer auxiliar para que se puedan almacenar strings que se quieran tener en consideración. Por ejemplo, para ver si se llegó al "\r\n\r\n". Este buffer tiene un tamaño de 4K ya que lo que se almacena se sabe que es pequeño.

```
struct request {
   method_t method;
   char * host;
   char* headers;
   int headers_length;
   int content_length;
   uint16_t dest_port;
};
```

Se utilizó la estructura request para almacenar los datos que se desean obtener para luego utilizar. El method es un enum con los métodos aceptados (GET, POST, HEAD, DELETE, PUT) o UNSUPPORTED en caso de que no lo sea. El host es donde se almacena el hostname. Es un buffer de 4K y en caso de sobrepasar esa longitud se retorna un error. Los headers es donde se guarda todo lo que parsea en el request, hasta '\r\n\r\n'. Este tiene un tamaño de 32K. *Content_length* lo que se parsea del header 'Content-Length' y el *dest_port* es el puerto que se especifica en el header 'Host'.

Response

Para la response la estructura es muy similar a request, lo único que varía es la estructura donde se almacenan los datos que se desean utilizar ya que estos son distintos.

```
struct response {
   uint8_t * media_type;
   bool chunked;
   uint8_t * headers;
   int header_length;
   int body_length;
   int status_code;
   bool compressed;
};
```

El *media_type* es donde se almacena el contenido de 'transfer-encoding' y esto es lo que se utiliza para saber si la transformación se aplica o no. En *chunked* y *compressed* se almacena si la response viene chunked o comprimida para saber cómo se aplica la

transformación y cuándo termina la response. *Header_length* es un contador para saber cuándo se llega a '\r\n\r\n'. El *status_code* es el código de la response.

Transform

Para tener un seguimiento de las configuraciones se tiene un TAD con un vector de estructuras:

```
typedef struct {
   char* media_type;
   int activated;
   transformation_type_t type;
} transformation_t;
```

De esta forma se almacena el media type al cual se le aplica la transformación, si está activada o no y el tipo de transformación que es. Este es un enum con las posibilidades:

```
typedef enum transformation_type {
   TOUPPER,
   ECHO,
   DUPLICATE,
   LEET,
} transformation_type_t;
```

Estas están explicadas en la sección Descripción de protocolos y aplicaciones.

Por otro lado, para poder transformar y no perder mucha eficiencia, una vez que se parsean los headers de la response y se sabe que se deben transformar, primero se abre el thread de la aplicación que va a transformar (ubicadas en ~/pc-2018-07/bin). Esta recibe por stdin el body y deja en stdout lo que transformó. Luego, se pasa al copy de transformation donde a medida que la app va resolviendo, se va mandando el body transformado. Por cada bloque de body que se transformó, se genera un chunk y así es como se va a enviar.