Protocolos de Comunicación

TPE

Grupo 2

Diseñado por ahora

Dentro de las características básicas:

1. Proxy transparente no bloqueante que envía datos de un buffer a otro
2. Resolución de nombre (no bloqueante) utilizando pthreads.
3. Logging en formato Apache / nginx
4. Parser [ver más adelante] que interpreta los request y response.
5. Cálculo de métricas
6. Comienzo de protocolo SCTP

¿Qué nos falta?

1. Integrar el parser de request y response para que pueda conectarse a un origin server (actualmente se conecta a uno hardcodeado por nosotros)
2. Agregar pipes (ya comenzado, tenemos un problema. Comentar)
3. Integrar con protocolo SCTP

Estructura básica de 1 cliente.

En este caso se contempla que es un usuario normal. Después hay que hacer modificaciones para que tambien pueda ser un client del tipo administrador (para conexiones SCTP)

```
typedef struct client {
  int fd_client;

  struct buffer *request;
  struct buffer *response;
  State state;

  int fd_originserver;
  struct sockaddr_in addres;

  int did_resolve;
  char *server_ip;
  int server_port;

  struct client *prev;
  struct client *next;
} client;
```

```c
//NUEVO CLIENTE
if (FD_ISSET(master_socket, &readfds)) {
    errno = 0;
    if ((new_socket = accept(master_socket, (struct sockaddr *)&address, (socklen_t*)&addrlen)) < 0) {
        perror("accept");
    }

    client *c = add_client(cl, new_socket);
    // placeholder(c);
    pthread_t thread;

    if (pthread_create(&thread, NULL, placeholder, c)) {
        perror("thread");
        exit(EXIT_FAILURE);
    }
}

//CHEQUEO LAS ACTIVIDADES DE LOS CLIENTES
resolve(cl, readfds, writefds, address, addrlen);
```

```c
struct addrinfo hints, *infoptr;
void* placeholder(void *args) {

    client *c = (client*) args;

    hints.ai_family = AF_INET; // AF_INET means IPv4 only addresses
    int result = getaddrinfo("sharlock.net", NULL, &hints, &infoptr);
    c->server_port = 80;

    if (result) {
        fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(result));
        c->did_resolve = -1;
        return NULL;
    }

    struct addrinfo *p;

    for(p = infoptr; p != NULL; p = p->ai_next) {
        getnameinfo(p->ai_addr, p->ai_addrlen, c->server_ip, 256, NULL, 0, NI_NUMERICHOST);
    }
    // freeaddrinfo(infoptr);

    c->did_resolve = 1;
    return NULL;
}
```

Ej.: 127.0.0.1 - - [05/Feb/2012:17:11:55 +0000] "GET / HTTP/1.1" 200 140

```
FILE* init_log_path(char* path);
FILE* init_log();
void close_log(FILE* fptr);
void write_log(FILE* fptr, char* log);
char* print_time();
char* format_log(char* ip, char* verb, char* context, int response_code);
```

Configuraciones

Protocolo de Transporte: SCTP

- Multihoming: Proxy escuchando en más de una interfaz
- Notificaciones: Por cada conexión se reciben notificaciones con información relevante (implementado el evento ASSOC_CHANGE, que indica la conexión y desconexión de un cliente, con ID asociado).

Protocolo de Configuración

Comandos y Respuestas ASCII, case-sensitive, de una palabra con un argumento o ninguno, de una línea terminados en CRLF.

Largos máximos de comandos y argumentos a determinar.

Manejo de estados (ON_HOLD, AUTH, NOT_AUTH) y contraseña inicial para autenticar cliente.

Contraseña correcta: estado autorizado, se puede configurar y pedir métricas.

Contraseña incorrecta: desconexión.

Comandos del Protocolo

Estado inicial: ON_HOLD
Cliente envía contraseña.
Si es correcta -> estado: AUTH
Si es incorrecta o el comando inválido: estado -> NOT_AUTH para próxima desconexión.

C: PASS
Argumento: *algunaContraseña*
Respuestas: OK Authorized
ERROR Not Authorized

C: QUIT
Sin argumento
Respuesta: OK Conection Closed

Comandos del Protocolo

Estado: AUTH
Se puede configurar y/o pedir métricas, de a un comando por vez hasta finalizar con comando QUIT.

C: PPORT
Argumento: *númeroPuerto* (número decimal)
Respuestas: OK Proxy Port Changed
ERROR Proxy Port Not Changed

C: CPORT
Argumento: númeroPuerto (número decimal)
Respuestas: OK Configuration Port Changed
ERROR Configuration Port Not Changed

Comandos del Protocolo (métricas)

C: CCONEC
Sin argumentos
Respuestas: OK *númeroConexiones* (decimal)
ERROR

C: HACC
Sin argumentos
Respuestas: OK *númeroAccesos* (decimal)
ERROR

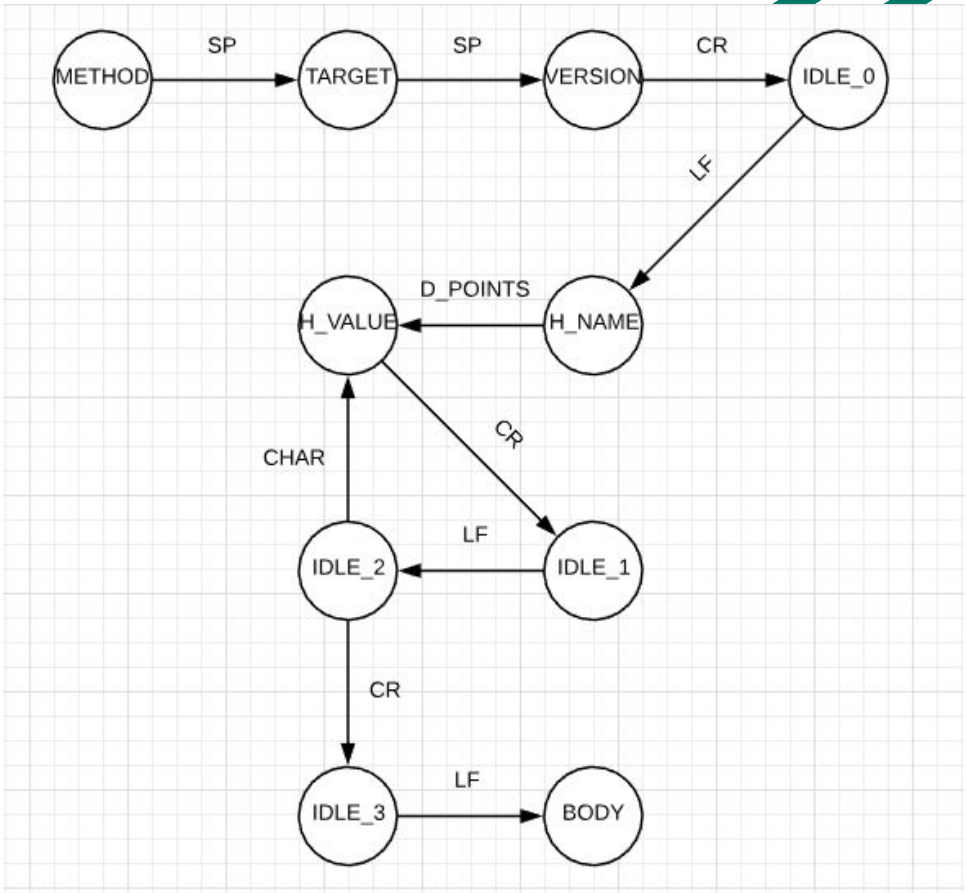
C: CBYTES
Sin argumentos
Respuestas: OK *númeroBytes* (decimal)
ERROR

C: SBYTES
Sin argumentos
Respuestas: OK *númeroBytes* (decimal)
ERROR

Parser

- State Driven
- Continuos parse
- Permissive

Parser : Request



Parser : Response