Universidad de Palermo

Facultad de Ingeniería

Programación en Redes

Trabajo Práctico
Sockets

Revisión: 2.0

Año 2023

Profesor: Ing. Diego Marafetti

Objetivos y definición

El trabajo práctico se basa en la implementación del primer prototipo de un sistema distribuido que permita mostrar contenido al usuario implementado una versión simplificada del protocolo http. Será una aplicación muy simple programada en C que se ejecutará sobre la plataforma Unix. Utilizará el protocolo HTTP v1.1 para aceptar como cliente a un browser (por ejemplo Google Chrome o Firefox). Expondrá una serie de *endpoints* a través de los cuales los clientes podrán acceder a los recursos estáticos y dinámicos aunque en principio solo habrá uno.

Se espera que:

- ✓ el alumno se familiarice con la API de Berkeley Sockets.
- ✓ entiendan la importancia de una norma o protocolo estándar en la comunicación entre procesos.
- ✓ dominen los problemas específicos de este tipo de implementaciones.

Algunas consideraciones extra:

- ✓ El lenguaje de programación que se utilizará será ANSI C.
- ✓ El protocolo que se utilizará será TCP sobre IPv4.
- ✗ No se pueden utilizar bibliotecas externas para manejar sockets. Solo se puede usar la API de Berkeley directamente.

Entregas

El desarrollo del TP estará dividido en 5 iteraciones con una demo de la aplicación al final de cada una.

Iteración 1

Se desarrollará la primera versión del webserver. Este proceso será un programa que se levantará por consola con la posibilidad de recibir ciertos parámetros como el puerto de escucha. El servidor será un proceso multithread, por lo tanto cada request recibido deberá atenderse en un nuevo thread. El mismo deberá finalizar al terminar de atender el pedido, liberando todos los recursos asociados. Se utilizará el soporte de la librería *pthreads*. Los threads serán creados en modo *detached* (ver man pthread_create). En paralelo, también se desarrollará un pequeño proceso cliente cuya única función será la de establecer una conexión TCP al webserver. Enviará un mensaje con el string “PING” y el server responderá con el string “PONG”. Al recibir la respuesta la imprimirá por consola y finalizará adecuadamente.

Iteración 2

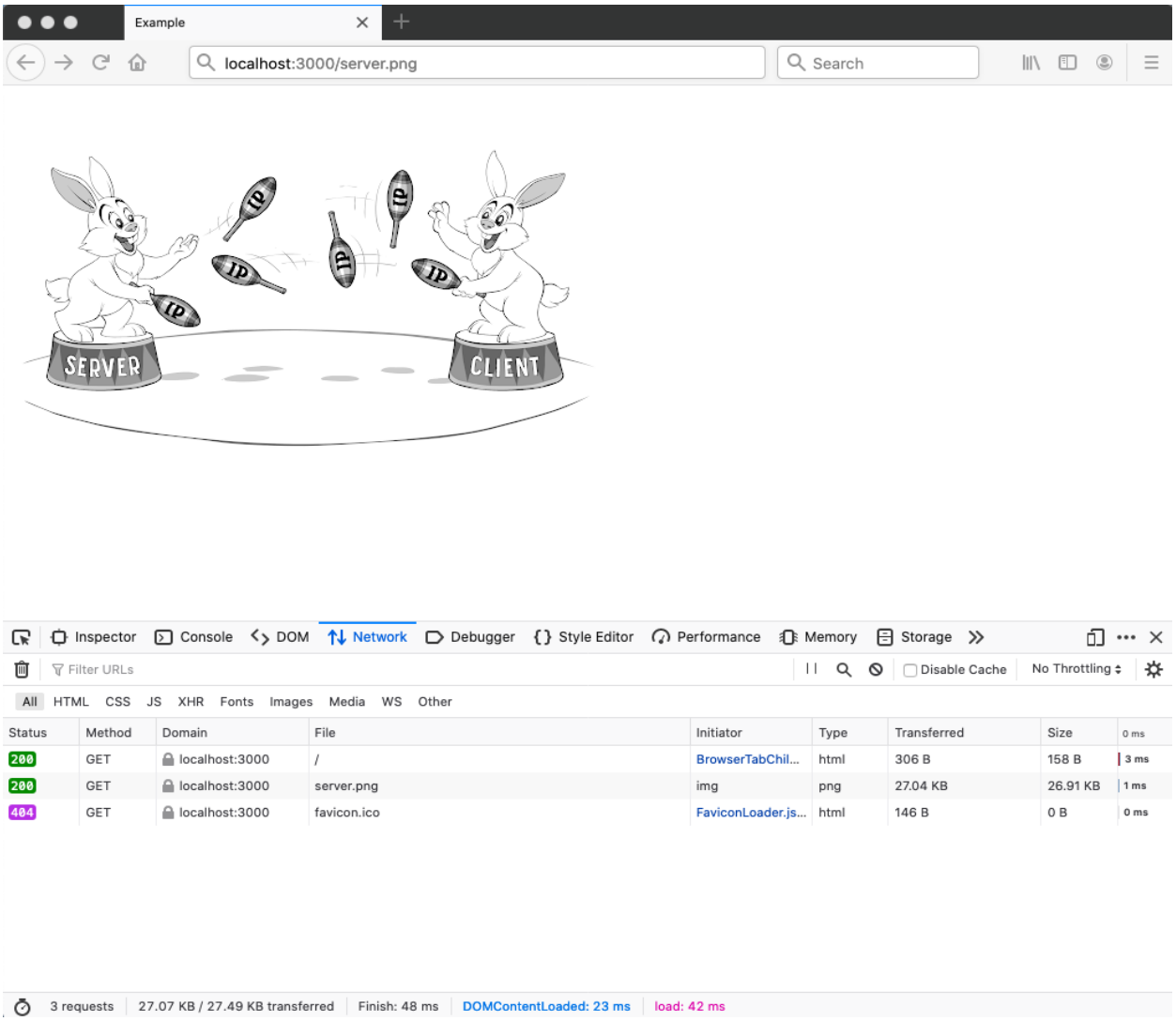
Para esta iteración se modificará el webserver desarrollado en la iteración anterior y se lo adaptará para realizar non-blocking I/O sobre un single-thread. Utilizará la función `select()` ó `poll()` para atender todas las conexiones TCP entrantes. El resto de la funcionalidad quedará igual que antes.

Iteración 3

El proceso webserver levantará un thread el cual creará un socket UDP cuyo único propósito será el de responder al mensaje de *heartbeat*. Al mismo tiempo se programará un pequeño proceso que, periódicamente y de forma indefinida, envíe el mensaje al server para indicar la normal operación del mismo. Si durante cierto período de tiempo o tras varios intentos no se recibió respuesta, entonces se deberá indicar por consola sobre dicho evento.

Iteración 4

El webserver implementará una versión muy reducida del protocolo http. El cliente programado para testear el servidor será dado de baja y a partir de ahora los clientes serán los browsers del mercado. El único endpoint que expone el servidor es `HTTP GET http://direcciónIp:puerto/imagen.jpg`. Un request a dicha url deberá traer el archivo de imagen completo.



Iteración 5

Como último paso, la implementación multithread del webserver deberá incluir un pool de threads cuyo tamaño de worker threads será variable por línea de comando. Cuando se acepte una nueva conexión, se deberá seleccionar el primer thread del pool para atender la tarea. Al finalizar el thread deberá volver al pool sin finalizar. En el caso de no haber thread disponible, la tarea deberá quedar en espera hasta que haya uno disponible nuevamente. La opción de timeout queda a criterio del alumno.