Trabajo Práctico
Programación de Sockets de Internet con Python

Bibliografía (ampliada)

El tutorial de Python. http://python.org.ar/wiki/Tutorial
Python Standard Library. Fredrik Lundh. O'Reilly & Associates. http://effbot.org/zone/librarybook-ora.htm
Documentación: https://docs.python.org/2/howto/sockets.html

Objetivo: Experimentar la programación de sockets utilizando el lenguaje Python para el manejo de protocolos de la pila TCP/IP de nivel 4. Definir protocolos de aplicación propios.

Parte 1
Fecha de Entrega: **19-10-2015**

1. Utilizando el módulo socket escriba un programa servidor que implemente el protocolo echo con SOCK_STREAM y otro con SOCK_DGRAM. También implemente los respectivos clientes para consultar dicho servicio. Realice pruebas con cliente y servidor en su sistema local (localhost) y luego en uno remoto. En todos los casos realice la captura y explique el resultado.
   Modifique el programa anterior para soportar múltiples clientes simultáneos. ¿El comportamiento del servidor basado en SOCK_DGRAM es igual cuando se pasa a SOCK_STREAM?

2. Implemente el protocolo DayTime (http://www.ietf.org/rfc/rfc867.txt), cliente y servidor, con UDP. Escriba un protocolo de aplicación propio que le permita especificar la fecha y hora de un determinado huso horario y en función de eso responda. El servidor recibe como parámetro la configuración regional del lugar donde opera y debe sincronizarse con algún servidor de hora.
   Agregue al inicio de su programa la especificación de su protocolo (la estructura de PDU debe ser de longitud fija).

3. Escriba un programa que implemente un cliente http utilizando el módulo socket (similar a wget). Agregue opciones para que el programa use un proxy y guarde en un archivo de log todos los header HTTP recibidos. Además, considere evaluar el header de redirección.

4. Se requiere una aplicación para calcular el RTT en la red (consulte http://www.ietf.org/rfc/rfc2681.txt). La misma debe permitir el ingreso de una lista de direcciones IP y un intervalo de tiempo (el mismo para todas) contra las cuales medir el RTT. Evalue los resultados con SOCK_STREAM y con SOCK_DGRAM.

5. Escriba un programa cliente y uno servidor que permitan ejecutar comandos Unix en una máquina remota (por ejemplo: pwd, ls, mv). Ejemplo:

   ```
   #~$ python cliente.py
   >>> pwd
   >>> /home/usuario/Taller_II/
   ```

   Sus programas debe tener una instancia de autenticación en la cual el nombre de usuario y la clave se encripten antes de enviarse por la red. ¿Cómo se puede resolver este problema? Revise cómo lo hace ssh. Incluya una opción para almacenar en un archivo indicado el log de toda la sesión.

6. Desarrolle un servidor http básico que implemente las respuestas para los códigos HTTP de estado 200 y 404. El servidor retornará el código de estado correspondiente de acuerdo a si existe o no el objeto solicitado. Si el código es 404, debe retornar una página HTML pre-diseñada que contiene el mensaje explicando la situación. Realice pruebas con diferentes navegadores.

7. Desarrollo un servidor proxy http básico que permita además realizar cache de objetos de una página Web (imágenes, flash, applets, etc.).

8. Escriba un programa servidor http que de igual forma a lo visto en el práctico anterior que permita realizar un balanceo de carga. Este load balancer debe interactuar con el servidor http del ejercicio 7.

9. Implemente un servicio de chat punto-a-punto. Su programación debe permitir vincular a 2 usuarios y que puedan “dialogar”. ¿Qué requisitos tiene el servicio? ¿Cómo se pueden implementar?

10. Escribir un programa servidor que retorne la hora en UTC y un cliente que ajuste su hora utilizando el Algoritmo de Cristian.

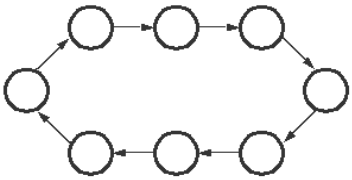
Parte II

Entrega: **Al menos uno** de los siguientes ejercicios deben ser entregados para el **19-10-2015** el resto pueden ser entregados hasta 20 días antes de rendir el final.

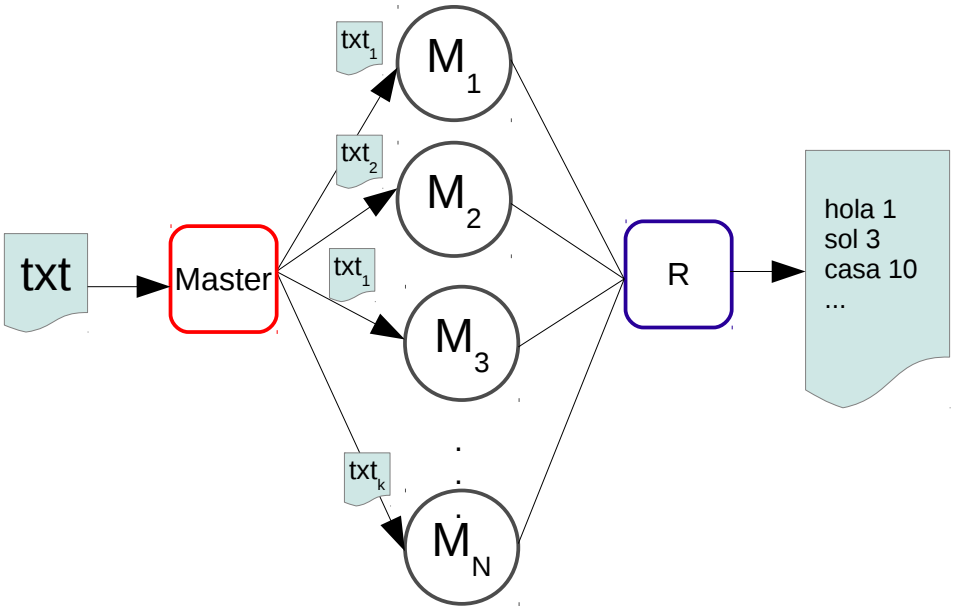
11. En este ejercicio el desafío es implementar fiabilidad, ordenamiento de PDUs y control de congestión sobre UDP. Escriba un cliente y servidor de archivos con las ideas explicadas en esta página http://gafferongames.com/networking-for-game-programmers/reliability-and-flow-control/ y luego haga una comparación con respecto a TCP.

12. Analice los requerimientos de un sistema de archivos distribuidos y escriba un programa que permita mantener sincronizado un directorio y todo su contenido replicado en varios equipos. Se deben contemplar los requerimientos de Transparencia, Replicación y Tolerancia a fallos. La funcionalidad del cliente debe ser provista a través de una terminal y debe poseer los comandos básicos de una terminal UNIX para listar, copiar, mover, etc.

13. Escriba un programa que implemente un anillo lógico como se muestra en la figura. Cada proceso en un host diferente tiene un ID (único) y cada uno puede transmitir cuando le llega el turno (secuencial). Cuando se envía un mensaje se debe incluir a qué ID va dirigido y siempre se reenvía en anillo – por ejemplo – hacia el nodo de la izquierda. El mensaje deber circular hasta llegar nuevamente al nodo que lo originó quien dará lugar a que transmita el siguiente (similar al protocolo Token Ring). Cada nodo lee de un archivo la lista de mensajes que tiene que transmitir. Los programas finalizan su ejecución cuando el nodo con ID = 0 envía un comando de finalización. Usted debe definir el protocolo de aplicación, tanto la estructura de datos como el comportamiento ante cada situación. Trate de escribir su especificación en formato de RFC.



14. Escriba un programa que implemente 3 tipos de nodos diferentes: Nodo master, Nodo de mapeo y Nodo de reducción. El nodo master, será el encargado de recibir un archivo de texto sin formato de al menos 5MB y deberá dividirlo a lo sumo en la mitad de nodos de mapeo que tiene la red y asignar una porción del archivo a cada nodo de mapeo. Una misma porción podrá ser asignada a más de un nodo de mapeo en simultáneo. Cada uno de estos nodos de mapeo deberá construir una estructura *<termino: frecuencia>* para esa porción de documento recibida. Al terminar la tarea el nodo anunciará la finalización de la tarea al nodo de reducción y si este se lo permite le enviará la estructura generada.
    Por último, el nodo de reducción será el encargado de generar una única estructura con todos los índices recibidos.



15. Escribir un programa que permita mantener un índice distribuido de documentos. Cada nodo de la red debe ser capas de encontrar nodos conectados en el mismo segmento de red e intercambiar información acerca de los documentos que tiene disponibles. También se deberá poder emitir un listado que muestre para cada documento en qué nodo esta este disponible.