ISSN 0798 1015

REVISTA ESPACIOS

HOME Revista ESPACIOS ÍNDICES A LOS AUTORES

Vol. 39 (Nº 19) Año 2018 • Pág. 12

Simulación de envío de paquetes bidireccionales aplicando el protocolo HTTP y TCP/RENO

Simulation of sending bi-directional packets using the HTTP and TCP / RENO protocol

Liliana ENCISO 1; Darwin GUAJALA 2; Jordy SARANGO 3; Pablo Alejandro QUEZADA-Sarmiento 4

Recibido: 09/01/2018 • Aprobado: 05/02/2018

Contenido

RESUMEN:
Este artículo muestra el uso del protocolo HTTP (Protocolo de Transferencia de Hipertextos) con el protocolo de transmisión TCP (Protocolo de Control de Transmisión) en su variante RENO, el cual ha modificado la retransmisión rápida para adicionar el Fast Recovery, lo cual previene que se vacíe el medio de transmisión, evitando la necesidad de ser llenado con el algoritmo Slow-Start. La implementación muestra el funcionamiento de ambos protocolos bajo el lenguaje de programación Python; beneficiando la creación tanto de aplicaciones web como aplicaciones móviles.
Palabras-Clave: Control de congestión TCP, performance TCP, HTTP, TCP Reno

ABSTRACT:
This article shows the use of the HTTP Protocol (Protocol for the Transfer of Hypertexts) with the transmission protocol TCP (Transmission Control Protocol) in its RENO variant, which has modified the fast retransmission to add the Fast Recovery, which prevents the transmission medium is emptied, avoiding the need to be filled with the Slow-Start algorithm. The implementation shows the operation of both protocols under the Python programming language; benefiting the creation of both web applications and mobile applications.
Keywords: TCP congestion control, performance TCP, HTTP, TCP Reno

1. Introducción

Conforme avanza la tecnología hay más personas que se conectan a Internet, este hecho conlleva a que su tráfico sea impresionante y gigante al mismo tiempo, propiciando la congestión de las redes, las cuales han estado en una evolución constante con el pasar del tiempo y la evolución de las tecnologías, este y otros más es uno de los principales motivos de que el protocolo de transmisión TCP (Postel, 1981),(Aguilar et al, 2016) tenga la necesidad de ir evolucionando para brindar confiabilidad y rapidez en una red de computadoras (Rigotti, 2008), ofreciendo un servicio de transferencia de datos confiable y

orientado a la conexión(Quispe & Galan, 2013), (Enciso et al, 2016), lo cual lo hace apto para aplicaciones que requieren transferencias de datos de cierto volumen y sin errores. Desde el punto de vista de la red, se posee mecanismos de control de congestión, que impiden que las aplicaciones saturen a la red con información, y por lo tanto contribuye a evitar su congestión. Estos mecanismos, ausentes en la especificación original, se han ido agregando a TCP(Pedroso & Fonseca, 2016), debido a la creciente necesidad de controlar el tráfico en la red, motivada por el crecimiento de la misma (Kurosse et al, 2010). Este mismo ha tenido una evolución dependiendo de sus necesidades por lo que podemos mencionar sus variantes como son: RENO, TAHOE, SACK y VEGAS (García, Ramírez, & Rátiva, 2013a). Además de tener una relación con varios tipos de protocolos a nivel de la capa de aplicación como son FTP, HTTP, DNS, SMTP, SMMP y TELNET, el protocolo HTTP es un protocolo sin estado que trabaja a nivel de la capa de aplicación [2], sin guardar ningún tipo de información sobre las conexiones establecidas, es por eso que son muy usadas los cookies (información enviada por un servidor y almacenada en el navegador del usuario), este protocolo trabaja con el esquema petición-respuesta entre los dos agentes que establecen la comunicación, en este caso cliente-servidor, el cliente envía realiza una petición al servidor (Tian, Xu, & Ansari, 2005), mientras que el servidor le envía un mensaje de respuesta, de esta manera los dos establecen una comunicación. La finalidad de este trabajo es desarrollar programa que realice la simulación el protocolo TCP variante Reno y trabaje conjunto el protocolo HTTP que es de la capa de aplicación, esta simulación deberá mostrar el funcionamiento de TCP (Biradar, Sarkar, & Puttamadappa, 2010) desde establecer una conexión hasta transmitir los datos, y tener un control de congestión de la red, y poder aplicar el protocolo HTTP con el cliente y servidor en el esquema petición-respuesta. Estos protocolos se pueden usar en aplicaciones de escritorio, portales web, y también aplicaciones móviles.

2. Metodología

2.1 Máquinas de estados finitos

La capa del modelo TCP tienen protocolos principales, en la capa de aplicación los cuales son: HTTP, FTP, SSH, TELNET entre otros y en la capa de transporte son TCP y UDP. Se propone el uso de HTTP como protocolo de la capa de aplicación y TCP con variación RENO para el transporte de datos, donde la función de RENO es el control de la congestión. A continuación, se muestra las máquinas de estados finitos usadas para el proyecto:

Figura 1
Cliente en Capa de Aplicación (HTTP).

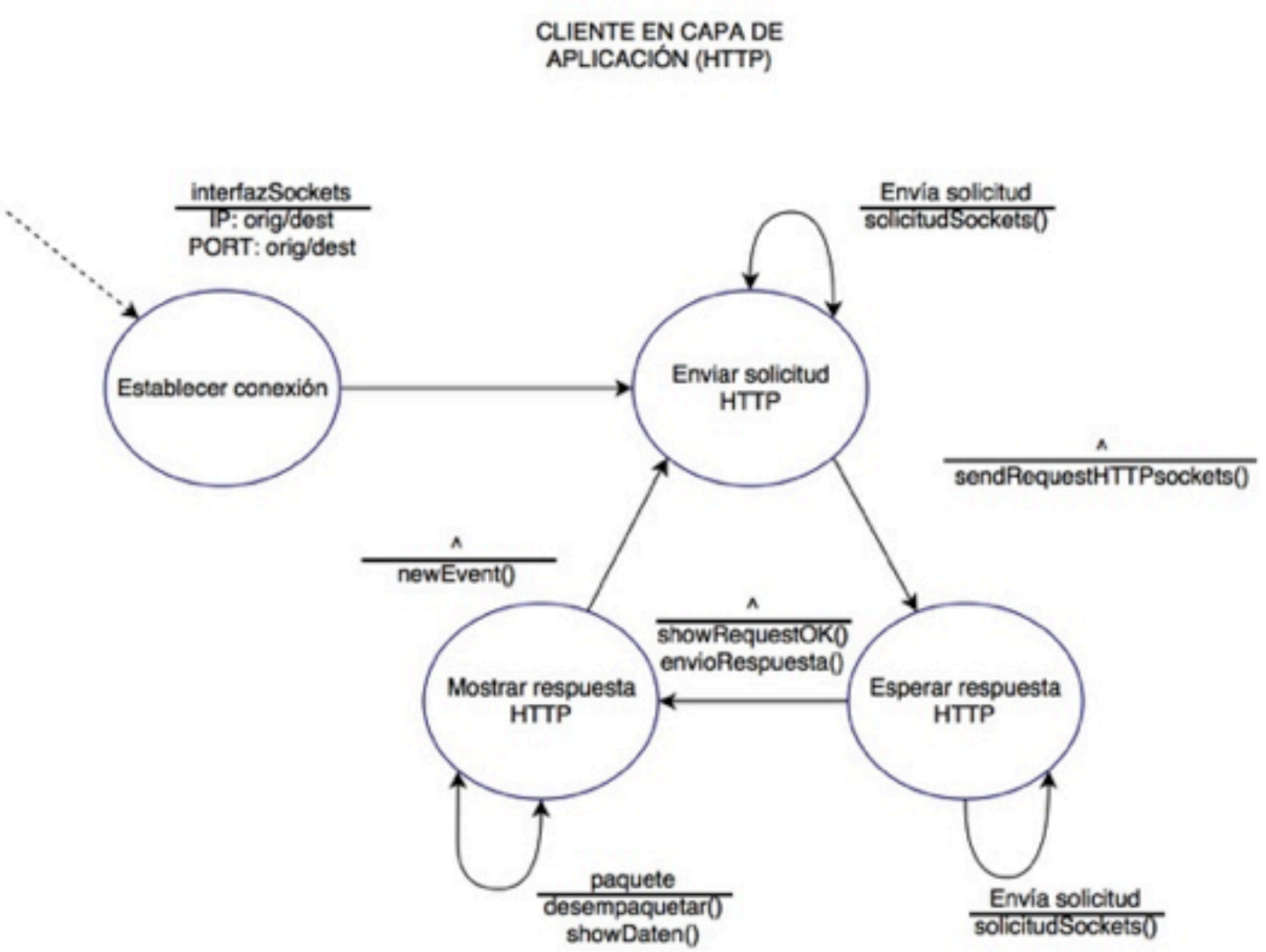


Figura 2
Servidor en Capa de Aplicación (HTTP).

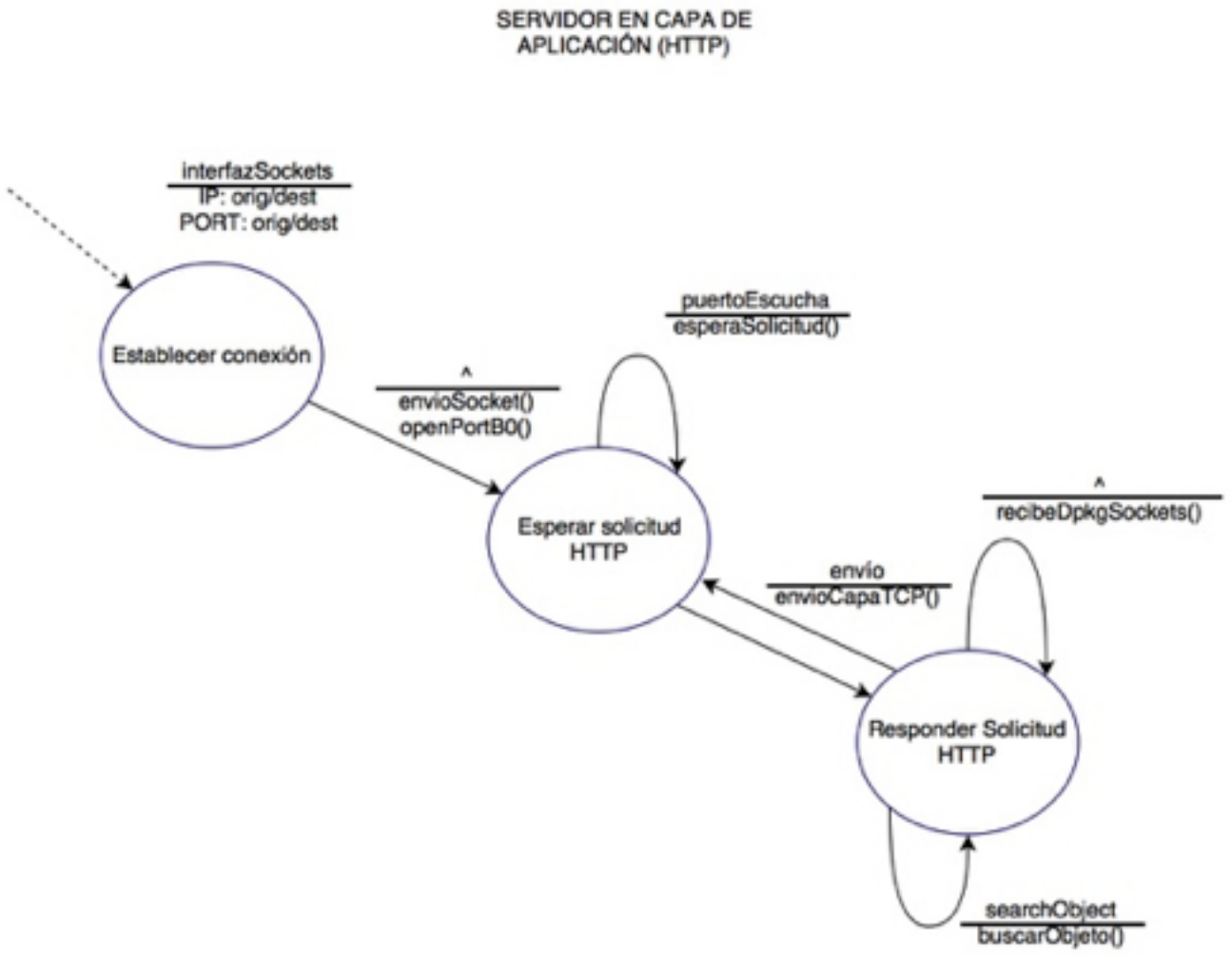


Figura 3
Cliente en Capa de Transporte (TCP/Reno).

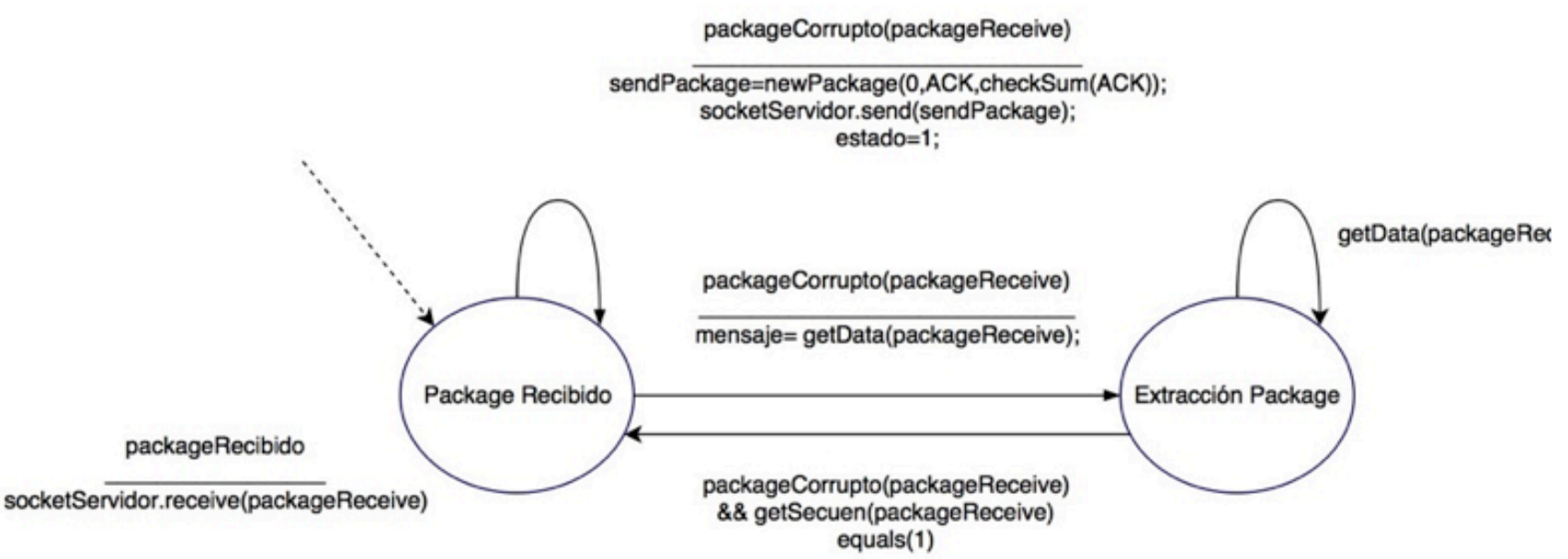
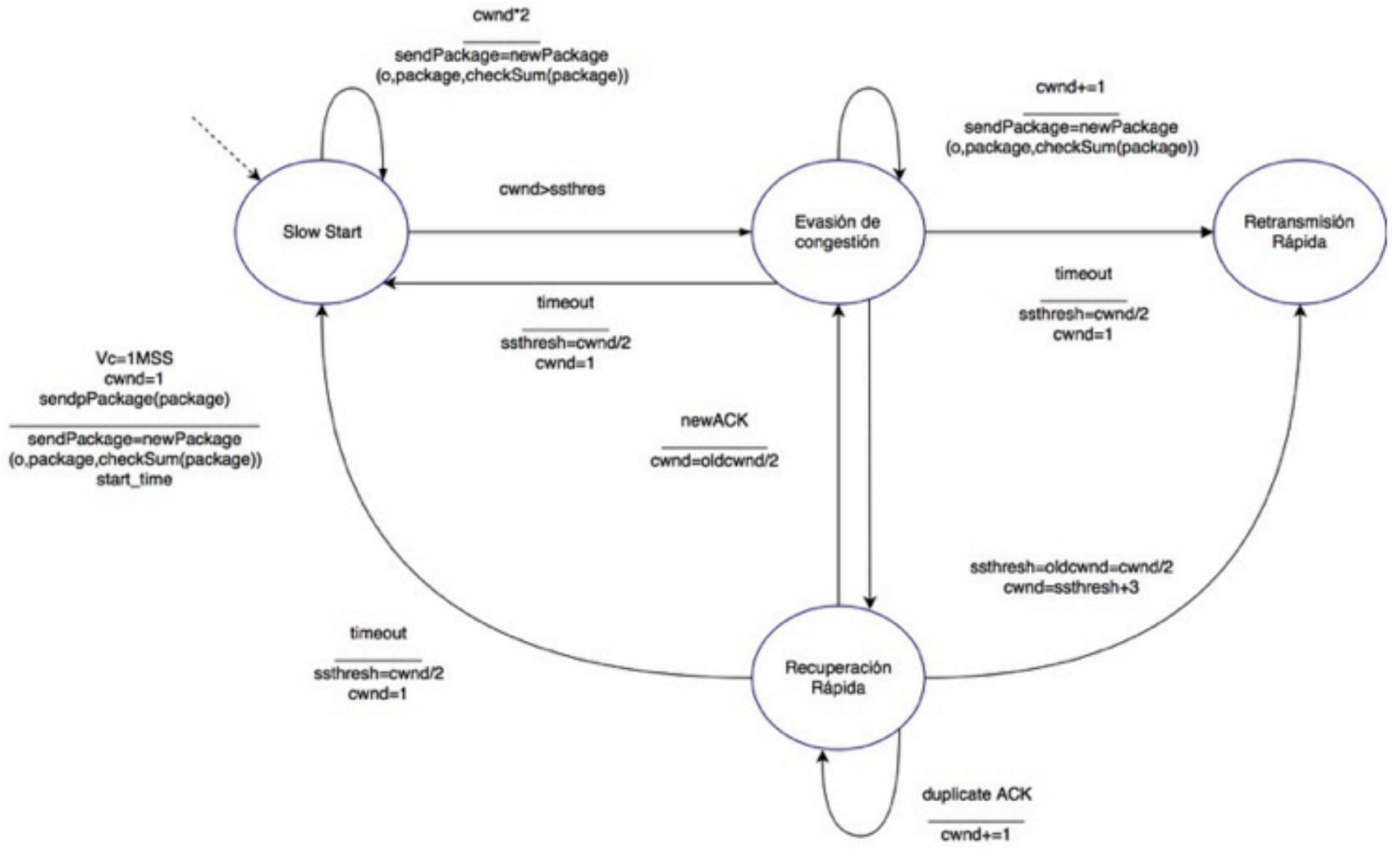


Figura 4
Servidor en Capa de Transporte (TCP/Reno).



2.2. Descripción de la solución inicial

La implementación de un simulador del protocolo TCP de la capa de transporte con el protocolo de HTTP de la capa de aplicaciones puede ser implementada correctamente, pero se tiene que tomar en cuenta cual es el control de congestión, como ya sabemos ese es un problema muy conocido hoy en día, el uso de TCP con variante reno mejora mucho en aspecto de velocidad y entrega confiable de paquetes, esta simulación puede ser realizada con Python (Jin, Hahn, & Croll, 2016) conformando una red local con la arquitectura servidor-cliente, esto ayudará a implementar el segundo protocolo HTTP con el esquema petición-respuesta, los cuales deberán establecer una conexión previamente antes de empezar a transmitir datos, luego de esto podremos ver el funcionamiento de reno aplicando los 4 algoritmos que son inicio lento, evasión de congestión, retransmisión rápida, recuperación rápida y de esta manera brindar una solución al problema de congestión de paquetes en una red.

2.3. Desarrollo y codificación host servidor

En la figura 5, se muestra parte del código implementado en el servidor, sobre la conexión, el puerto con el que se comunicará con los clientes, así mismo contiene las variables del contenido del protocolo TCP, como son: umbral, MSS, TTL (línea 1-11) además la parte de la confirmación de conexión que se está estableciendo con el cliente (línea 32).

Figura 5
Código del Servidor.

```
1  def variablesGLO(host="192.168.1.2",port
       =80,limite=800,ttl=10,mss=1):
2    global HOST
3    global PORT
4    global listen_socket
5    global clientes
6    global umbral
7    global MSS
8    global TTL
9    global msgsizeT
10   global umbralmax
11   global umbralT
12   umbralT=[]
13   umbralmax=int(limite)+100
14   msgsizeT=[]
15   MSS=mss
16   umbral=limite
17   TTL=ttl
18   HOST = host
19   PORT = port
20   clientes=[]
21   listen_socket = socket.socket(socket.
         AF_INET, socket.SOCK_STREAM)
22 class bcolors:
23       HEADER = '\033[95m'
24       OKBLUE = '\033[94m'
25       OKGREEN = '\033[92m'
26       WARNING = '\033[93m'
27       FAIL = '\033[91m'
28       FIN = '\033[0m'
29 def pasos(estado1="X",estado2="X",estado3
       ="X",estado4="X"):
30   os.system('clear')
31   print (bcolors.FAIL+"1.- Levantar
         Servicio"+bcolors.FIN,estado1)
32   print (bcolors.FAIL+"2.- Establecer
         Conexion"+bcolors.FIN,estado2)
33   print (bcolors.FAIL+"3.-
         Desfragmentacion de Datos"+bcolors.
         FIN,estado3)
34   print (bcolors.FAIL+"4.- Envio de Datos
         "+bcolors.FIN,estado4)
35   print (bcolors.FAIL+"5.- Confirmacion
         de Datos Enviados"+bcolors.FIN)
36 def menu():
37   print (bcolors.HEADER+"\tOPCIONES DE
```

```
        SERVIDOR"+bcolors.FIN)
38      print (bcolors.OKGREEN+"1.- Levantar
            servidor"+bcolors.FIN)
39      print (bcolors.OKGREEN+"2.- Levantar
            servidor (Opciones avanzadas)"+
            bcolors.FIN)
40      print (bcolors.OKGREEN+"3.- Ver
            varibles"+bcolors.FIN)
41      print (bcolors.WARNING+"Detener (CTRL+C
            )"+bcolors.FIN)
42      res=raw_input("Inserta un numero >> ")
43      return res
```

2.4. Interfaces servidor y cliente

Primero se ejecutará el archivo Servidor.py, en la consola de Linux, tal como muestra en la figura 6, a continuación nos abrirá varias opciones que nos ofrece el servidor, se escogerá la opción de levantar servidor, el mismo que tendrá los valores por defecto del servidor.

Figura 6
Interfaz del Servidor.

Una vez que el servidor este levantado, se ejecutara el archivo cliente.py, en la consola de Linux, tal como muestra la figura 7, en donde nos saldrá las opciones de conectarse a servidor y ver variables de nuestro cliente.

Figura 7
Interfaz del Cliente.

---

3. Resultados

Entorno Servidor

Una vez escogida la opción de levantar servicios, nos mostrará los valores por defecto del servidor, de esta manera el servidor se levantará con éxito. Además, nos muestra que el mismo se está comunicando por el puerto 80 esperando a la conexión de algún cliente. Tal como se puede observar en la figura 8.

Figura 8
Conexión Servidor.

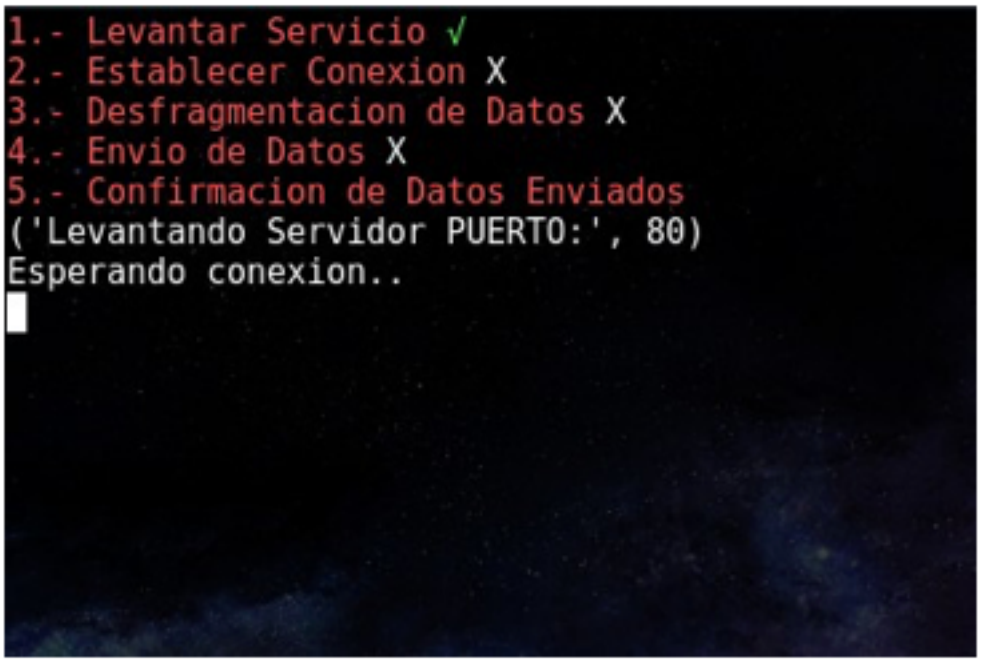
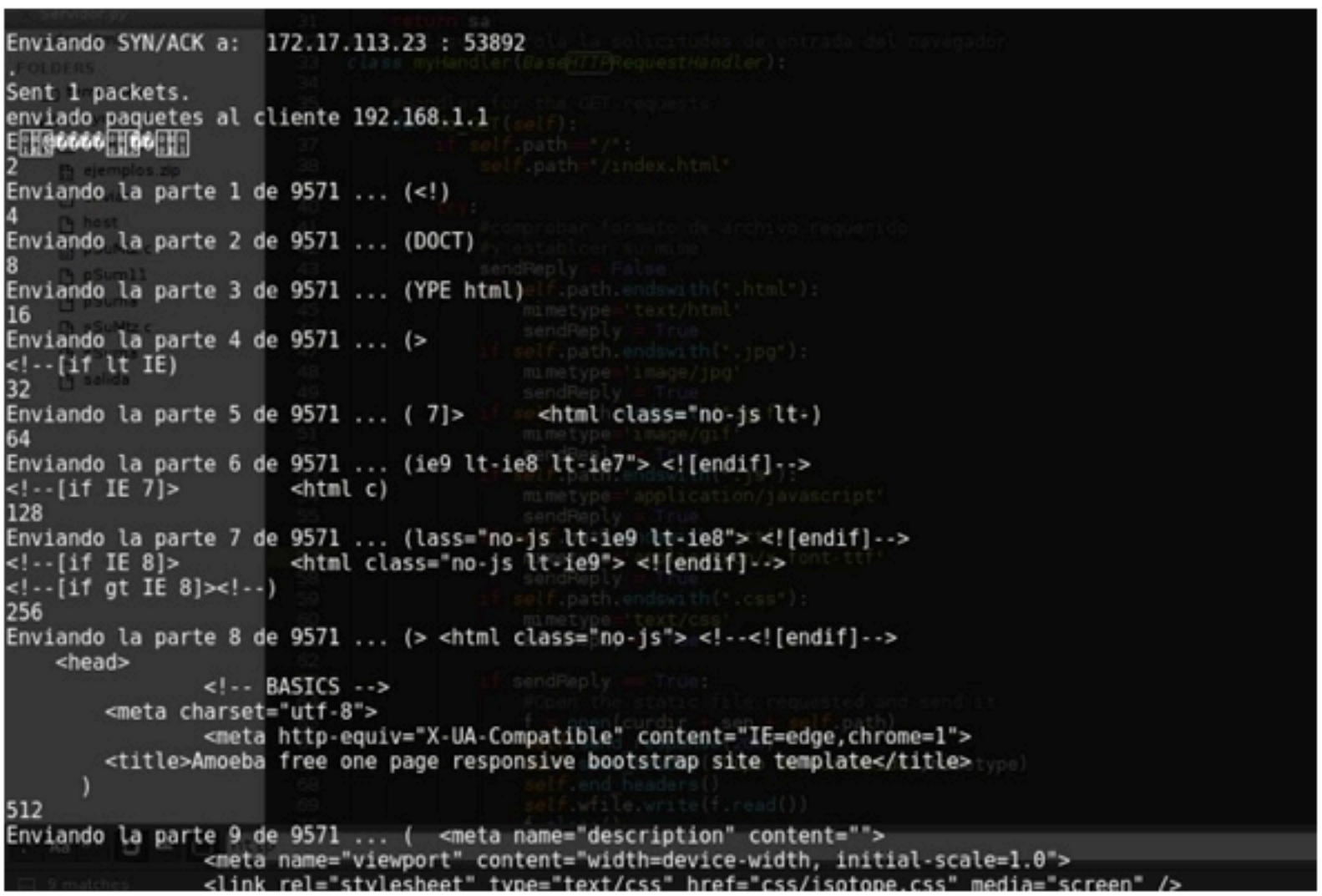


Figura 9
Desfragmentación de página web.



Una vez que se establece la conexión, el servidor comenzará a enviar la página que se encuentra en el mismo, y comenzara a dividirla en paquetes para poder ser enviada, para que el cliente pueda recibir todos estos paquetes. Tal como se muestra en la figura 9.

Figura 10
Arreglo variante reno.

```
python Servidor.py                                  root@by-default: ~/Documentos/ProyectosPersonales/proyecto
    <script src="js/bootstrap.min.js"></script>
    <script src="https://maps.google.com/maps/api/js?sensor=true"></script>
    <script src="js/jquery.isotope.min.js"></script>
    <script src="js/jquery.nicescroll.min.js"></script>
    <script src="js/fancybox/jquery.fancybox.pack.js"></script>
    <script src="js/skrollr.min.js"></script>
    <script src="js/jquery.scrollTo-1.4.3.1-min.js"></script>
    <script src="js/jquery.localscroll-1.2.7-min.js"></script>
    <script src="js/stellar.js"></script>
    <script src="js/jquery.appear.js"></script>
  <script src="js/main.js"></script>
  <script src="contactform/contactform.js"></script>

    </body>
</html>)
Se han terminado de enviar los datos
[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 512, 1024, 512, 1024, 512, 1024, 512, 1024, 512, 1024, 512, 1024, 51
4, 512, 1024, 512]
umbralkes [512, 512, 512, 512, 512, 512, 512, 512, 512, 512, 512, 512]
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET / HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /css/isotope.css HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /js/fancybox/jquery.fancybox.css HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /css/bootstrap.css HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /css/bootstrap-theme.css HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /css/style.css HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /skin/default.css HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /js/modernizr-2.6.2-respond-1.1.0.min.js HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /js/jquery.js HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /js/jquery.easing.1.3.js HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /js/bootstrap.min.js HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /js/jquery.isotope.min.js HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /js/jquery.nicescroll.min.js HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /js/fancybox/jquery.fancybox.pack.js HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /css/font-awesome.css HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /css/overwrite.css HTTP/1.1" 200 -
192.168.1.1 - - [31/Jan/2017 14:20:18] "GET /css/animate.css HTTP/1.1" 200 -
```

Una vez que el servidor envió todo el contenido de la página, nos muestra un arreglo de todos los paquetes enviados en forma exponencial esto es debido a la variante del protocolo TCP que usamos, el variante reno, que envía desde 2 a 4 incrementando exponencialmente, al llegar al umbral comenzará a bajar para enviar paquetes desde la mitad del umbral en la que se encuentra. Tal como muestra en la figura 10.

Entorno Cliente

Figura 11
Solicitud de datos de conexión con el servidor.

```
Sent 1 packets.
Esperando la llegada de los paquetes...
^Cdaguajala@daguajala:~/Escritorio/Proyecto Redes Tcp$ sudo python Cliente.py
```


```
OPCIONES DE CLIENTE
1.- Conectarse a servidor
2.- Ver varibles
Detener (CTRL+C)
Inserta un numero valor >> 1
Selecione interfaz a usar
0 : lo
1 : eth0
2 : wlan0
Selecione>> 1
Su IP es:  192.168.1.1
Ingrese puerto origen>> 4
Ingrese puerto de destino>> 80
Ingrese host de destino>> 192.168.1.2
Conexion establecida
Envio de SYN
SYN/ACK enviado
```

Al momento de conectarnos con el servidor nos solicita el tipo de conexión que tenemos, en este caso una conexión por red, luego se colocará nuestro puerto de origen de nuestro cliente, posteriormente se colocará el puerto de nuestro servidor que será el puerto 80 HTTP,

y por último el host de destino para comunicarnos con nuestro servidor. Tal como muestra en la figura 11.

Figura 12
Desfragmentación de página web.

Después de que la conexión fue exitosa, establecerá una conexión y comenzara con el envío de paquetes, el servidor enviara los paquetes de la página web y enviara por segmentos (fragmentos de la página web divididos), los mismos que será recibido por el cliente para así mostrar el contenido de la página (Ver figura 12).

Resultado Final

Figura 13
Página web alojada en el servidor.

Ya una vez realizado todo el proceso, el cliente se dirigirá a un navegador colocara la dirección IP de la página web, y se nos presentará la página contenida que fue enviada por el servidor (ver Figura 13), en caso de que el servidor se detenga, la página web colapsará.

---

4. Conclusiones

El uso de la variante Reno de TCP es una buena opción para establecerlo frente a las pérdidas de segmentos, ya sea por sus mejoras realizadas de Thoe, en el control de

congestión se puede aplicar o modificar diferentes variantes de TCP Según sea su necesidad. Con el uso del protocolo TCP, las aplicaciones se pueden comunicar de una manera más segura, gracias al sistema de acuse de recibido que nos da este protocolo. Es importante destacar que una función del protocolo TCP es controlar la velocidad de los datos usando adecuadamente su capacidad de emitir mensajes de tamaño variable. Con la implementación de esta simulación se pudo comprobar el las ventajas de usabilidad que tiene estos protocolos, ayudándonos a comprender el nivel de interacción de los dos protocolos juntos puede beneficiar a la creación de aplicaciones web, así como aplicaciones móviles. Además visualizar la interacción en el envío de paquetes, y como este hace el recibimiento de los mismos, entre los clientes y el servidor.

Referencias bibliográficas

Libro

Kurose, J. F., Ross, K. W., Hierro, C. M., y Pablo, Á. P. D. M., & Marrone, L. (2010). Redes de computadoras: un enfoque descendente. Addison Wesley.

Artículo

Aguilar Salazar, D., Lara Cueva, R., Leon Perez, R. P., & Santos Logrono, G. (2016). "Performance Analysis and Optimization of TCP Protocol New Algorithm using Dedicated Hardware," IEEE Lat. Am. Trans., vol. 14, no. 6, pp. 2940–2946, Jun. 2016.

Balakrishnan, H., Padmanabhan, V.N., Seshan, S., & Katz, R. (1997). "A comparision of Mechanisms of Improving TCP Performance Over Wireless Links," IEEE\ACM Trans. Netw., no. 6, pp. 1–14, 1997.

Biradar, S., Sarkar, S. K., y Puttamadappa, C. (2010). A comparison of the tcp variantsperformance over different routing protocols on mobile ad hoc net- works. International Journal on Computer Science and Engineering, 1(2), 340–344.

Enciso, L., Quezada, P., Fernandez, J., Figueroa, B., & Espinoza, V. (2016). Analysis of performance of the routing protocols ad hoc using random waypoint mobility model applied to an urban environment. WEBIST 2016 - Proceedings of the 12th International Conference on Web Information Systems and Technologies, 1, pp. 208-213.

García, N. Y. G., Ramírez, A. R. C., & Rátiva, J. A. S. (2013). "Comparación entre los tipos de TCP RENO, SACK Y VEGAS desde el punto de vista del control de flujo," Redes Ing., vol. 3, no. 2, pp. 61–76, 2013.

Jin, Q., Hahn, J., & Croll, G. (2016). "BIBFRAME Transformation for Enhanced Discovery," Libr. Resour. Tech. Serv., vol. 60, no. 4, p. 223, Oct. 2016.

Pedroso, C.M., & Fonseca, K. (2016). "Um Método para Melhorar o Desempenho de Servidores Web que Apresentam Perfil de Tráfego Altamente Variável."

Postel, J. (1981). "Transmission Control Protocol," Defense Advanced Research Projects Agency Information Processing Techniques Office, 1981. [Online]. Available: https://www.ietf.org/rfc/rfc793.txt. [Accessed: 20-agosto-2017].

Quispe L.E., & Galan, L.M. (2013). Assessment of Throughput Performance under NS2 in Mobile Ad Hoc Networks (MANETs)," 2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks, Madrid, 2013, pp. 338-343. doi: 10.1109/CICSYN.2013.80

Rigotti, G. (2008). "JTCP: una implementación de TCP orientada a la evaluación de técnicas de control de congestión."

Tian, Y., Kai, X., & Nirwan, A. (2005). "TCP in Wireless Enviroments: Problems and Solutions," 2005.

---

1. Doctora en Informática; Master en Ciencias de la Ingeniería; Máster en Ciencias y Tecnologías de la Computación; Ingeniería en Sistemas; Docente de la Universidad Técnica Particular de Loja, Investigadora del Grupo Innovación

Tecnológica Educativa, lenciso@utpl.edu.ec

2. Profesional en Formación en Sistemas Informáticos y Computación; Universidad Técnica Particular de Loja, daguajala@utpl.edu.ec

3. Profesional en Formación en Sistemas Informáticos y Computación; Universidad Técnica Particular de Loja, jrsarango@utpl.edu.ec

4. Ingeniero en Informática y Multimedia, Máster en Ciencias y Tecnologías de la Computación, Docente de la Universidad Internacional del Ecuador Escuela de Informática y Multimedia, paquezadasa@uide.edu.ec

---

Revista ESPACIOS. ISSN 0798 1015
Vol. 39 (Nº 19) Año 2018

[Índice]