Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

# Hacia un nuevo enfoque de TCP para un aumento del throughput en redes inalámbricas

## *Towards a new approach of TCP for increasing throughput in wireless networks*

## *Rumo a uma nova abordagem TCP para maior rendimento em redes sem fio*

**Román Alcides Lara Cueva**
Universidad de las Fuerzas Armadas, ESPE, Ecuador
ralara@espe.edu.ec
http://orcid.org/0000-0001-8848-9928

**Diego Xavier Martínez Hidalgo**
Universidad de las Fuerzas Armadas, ESPE, Ecuador
dxmartinez@espe.edu.ec
https://orcid.org/0000-0001-9723-6885

## Resumen

Uno de los protocolos más importantes para el correcto funcionamiento de las redes de Internet es el protocolo TCP, el cual asegura la comunicación entre transmisor y emisor mediante un control de la tasa de transmisión en función de la congestión. Sin embargo, con la nueva tendencia de las comunicaciones inalámbricas, TCP enfrenta un nuevo desafío para el cual no estaba originalmente diseñado. Siendo el PL debido al medio de transmisión motivo de la ineficiencia de TCP en medios inalámbricos, se han desarrollado propuestas como el empleo de notificaciones de pérdida explícita y mejores gestiones de la ventana de congestión con el fin de adaptar TCP a tal medio de transmisión. Dichas propuestas evidencian un mejor desempeño, aunque en su gran mayoría se limitan a evaluar su funcionamiento en programas de simulación como Network Simulator o similares. En este contexto, el objetivo del presente trabajo es implementar un protocolo TCP adaptado a escenarios inalámbricos en el sistema operativo Linux, considerando un acuse de recibo negativo NACK, el cual ocupa un pequeño segmento del protocolo TCP. TCP-NACK es capaz de diferenciar entre pérdidas por congestión

Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

y pérdidas por el medio de transmisión, y produce una mejora en *throughput* de 182 % bajo un escenario emulado en comparación con TCP Reno.

**Palabras clave:** CWND, emulación, Linux, NACK, TCP inalámbrico.

## Abstract

One of the most important protocols for the proper functionality of Internet networks is the Transmission Control Protocol TCP, which ensures communication between transmitter and sender by controlling the transmission data rate based on congestion. However, with the new trend of wireless communications, TCP faces a new challenge for which it was not originally designed. As the packet loss due to the transmission medium causes TCP inefficiency in wireless media, several proposals have been developed such as the use of explicit loss notifications and better management of the congestion window in order to adapt TCP to such medium. These proposals shows a better performance, however, the vast majority are limited to evaluating their performance in simulation programs such as Network Simulator or similar. In this context, the objective of this paper is to deploy a TCP protocol adapted to wireless scenarios in the Linux operating system, considering a Negative Acknowledgment NACK, which occupies a small segment of the TCP protocol. TCP-NACK is able to differentiate between losses due to congestion and losses by the transmission media, and produces a Throughput improvement by 182% under an emulated scenario when compared with TCP Reno.

**Keywords:** CWND, emulation, Linux, NACK, TCP wireless

## Resumo

Um dos protocolos mais importantes para o bom funcionamento das redes de Internet é um protocolo TCP, o que garante a comunicação entre o transmissor eo emissor, controlando a taxa de transmissão dependendo do congestionamento. No entanto, com a nova tendência das comunicações sem fio, o TCP enfrenta um novo desafio para o qual não foi originalmente projetado. Sendo o PL devido ao meio de transmissão por causa da ineficiência do TCP nos meios de comunicação sem fio, eles desenvolveram propostas, tais como o uso de notificações de forma explícita e melhores esforços de janela de congestionamento, a fim de adaptar o TCP para tal modo de perda de transmissão. Estas propostas mostram um melhor desempenho, mas principalmente limitado para avaliar o seu desempenho em programas de simulação, tais como

Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

Network Simulator ou similar. Neste contexto, o objetivo deste trabalho é a implementação de um protocolo TCP adaptado para configurações sem fio no sistema operacional Linux, considerando uma confirmação negativa NACK, que ocupa um pequeno segmento do protocolo TCP. TCP-NACK é capaz de diferenciar entre as perdas e as perdas de congestionamento a partir do meio de transmissão, e produz uma melhoria na taxa de transferência sob 182% em relação a uma fase de TCP Reno emulado.

**Palavras-chave:** CWND, emulação, Linux, NACK, TCP sem fio.

**Fecha Recepción:** Septiembre 2017 **Fecha Aceptación:** Diciembre 2017

## Introducción

El protocolo de control de transmisión (TCP del inglés *Transmission Control Protocol*) fue diseñado con un enfoque hacia las redes cableadas, y por ello malinterpreta por congestión la causa de la pérdida de paquetes (PL del inglés *Packet Loss*) en medios inalámbricos cuando en realidad es causada por factores propios del canal de comunicaciones, como las colisiones de paquetes, las interferencias y el ruido (Xylomenos, Polyzos, Mahonen y Saaranen, 2001). Para el control de la congestión en la red, TCP emplea un acuse de recibo (ACK del inglés *Acknowledgement*), el cual es utilizado por el receptor para indicar que se tomó el segmento sin errores y para especificar el siguiente segmento esperado.

Si el transmisor no detecta ningún ACK durante un tiempo preestablecido, se reenvían los paquetes o se cancela la conexión (Postel, 1981). TCP puede utilizar cuatro mecanismos para el control de la ventana de congestión (CWND del inglés *Congestion Window*): *Slow Start, Congestion Avoidance*, *Fast Retransmit* o *Fast Recovery*. Estos mecanismos buscan alcanzar un alto rendimiento en la red y evitar un colapso debido a la congestión. *Slow Start* es empleado para aumentar, de manera exponencial, la tasa de transmisión (BR del inglés *Bit Rate*) en el medio hasta que se detecte una pérdida de segmentos.

A partir de ese punto, actúa el algoritmo *Congestion Avoidance*, con el cual la tasa de trasmisión aumenta linealmente y se reduce al momento de detectar ACK duplicados. *Fast Retrasmit* y *Fast Recovery* permiten reenviar segmentos inmediatamente después de recibir tres ACK duplicados. Cabe indicar que el paso de un algoritmo a otro ocurre cuando se detectan

paquetes perdidos (Stevens, 1997). En redes con baja tasa de error de bit (VER del inglés *Bit Error Rate*), estos algoritmos logran dicho objetivo; sin embargo, en redes inalámbricas se producen ejecuciones erróneas de los cuatro algoritmos y bajas tasas de transmisión, tal como se indica en Tian, Xu y Ansari (2005).

El protocolo TCP posee diferentes versiones que buscan optimizar la CWND por medio de un análisis de la tasa de notificaciones ACK (Olmedo, 2008), dos de las versiones más populares son TCP Reno y TCP Westwood. TCP Reno reduce a la mitad la CWND después de recibir tres ACK duplicados; en cambio, TCP Westwood selecciona un umbral de *Slow Start* y una CWND que sean consistentes con la tasa de conexión efectiva en el momento que ocurre la congestión (Zanella, Procissi, Gerla y Sanadidi, 2001). Para reforzar el esfuerzo del control de la CWND en redes inalámbricas, como las que utilizan tecnología wifi, se han desarrollado diferentes mecanismos, por ejemplo, mediante el empleo de notificaciones de pérdida explícita, añadiendo bits explícitos al segmento TCP como retroalimentación a la fuente para determinar la naturaleza de la pérdida, lo cual mejora en hasta 30 % valores de *throughput* ($\bar{\theta}$) comparado con TCP-NewReno, para un tráfico web simulado bajo altas probabilidades de error (Buchholcz, Ziegler y Do, 2005).

Este protocolo propuesto, definido como TCP-ELN, presenta, no obstante, la desventaja de no realizar la notificación del paquete defectuoso justo después de detectarlo. TCP-ELN tiene que esperar a la recepción exitosa de un paquete para notificar todas las pérdidas anteriores como una información adicional insertada en el ACK generado. Otro problema en las redes inalámbricas es el tiempo de retransmisión (RTO del inglés *Retransmission Time Out*) que es calculado con base en el tiempo de ida y vuelta (RTT del inglés *Round Trip Time*), el cual es variante en medios inalámbricos y puede llegar a provocar RTO falsos (Chakraborty y Nandi, 2014).

Esta dependencia al RTT se puede apreciar en ambientes cerrados, donde, en conjunto a problemas por terminales ocultos y a las interferencias entre sets de servicio básico, ocurren desbalances de $\bar{\theta}$. Como solución a este escenario ha sido propuesto un control de las ventanas de contención de la capa MAC mediante el uso de un modelo analítico de ecuaciones no lineales. Con ello se alcanzó, mediante simulación, una distribución justa del $\bar{\theta}$ y un intervalo de confianza de 99.98 % a una tasa de dos Mbps (Hung y Bensaou, 2011). Igualmente, con base en el mismo problema de dependencia al RTT, el protocolo TCP amigable con el control de flujo (TFRC del inglés *TCP Friendly Rate Control Protocol*) sufre de PL y tiempos altos de transmisión cuando es aplicado a enlaces inestables o de larga distancia.

Como solución ha sido propuesto un cálculo mejorado de los valores de RTT y RTO, con el cual se brinda al protocolo TFRC la capacidad de diferenciar la causa de PL. Mediante simulaciones, el protocolo TFRC mejorado alcanza 22 % mayor $\bar{\theta}$ en comparación con el protocolo TFRC estándar (N. Reddy, Reddy y Padmavathamma, 2017).

Según lo expuesto anteriormente, se puede afirmar que existe la necesidad de modificar el protocolo TCP para aplicar una retransmisión simple e inmediata del paquete dañado sin la reducción de la CWND. Por ese motivo, el propósito del presente trabajo consiste en diseñar, implementar y evaluar un nuevo protocolo TCP para ambientes inalámbricos. Para cumplir con este objetivo, se agregó un segmento de confirmación ACK negativo (NACK del inglés *Negative Acknowledgement*) dentro de la cabecera del protocolo TCP. Estas notificaciones negativas se encargarán de informar al transmisor que un paquete fue recibido con errores, en lugar de solo descartarlo. Con las notificaciones NACK se impide que el protocolo TCP confunda PL debido a la inestabilidad del canal del PL causado por la congestión, lo cual evita una reducción innecesaria de la CWND.

Para ello, en primer lugar se partió de un modelamiento matemático del comportamiento del protocolo propuesto; después se realizaron modificaciones al Kernel (código fuente) de Linux para introducirlo al funcionamiento del Sistema Operativo, y finalmente se evaluó en un escenario emulado con el fin de obtener valores precisos de desempeño bajo condiciones controladas.

En síntesis, el resto del presente documento se encuentra organizado de la siguiente manera. La sección II presenta la propuesta matemática original que dio paso a la implementación. En la sección III se expone el diseño del protocolo TCP Reno modificado dentro del Kernel de Linux. En la sección IV se presenta el desarrollo del escenario de pruebas y los resultados obtenidos. Por último, en la sección V se exponen los trabajos futuros.

## Análisis matemático original

Inicialmente, el protocolo propuesto fue modelado en la investigación de Olmedo (2008), donde se consideraron las etapas de *Slow Start* y *Congestion Avoidance* a partir de las versiones TCP Reno y Westwood. Se estableció la evolución de la CWND como un proceso de Markow, lo cual permitió obtener una ecuación del $\bar{\theta}$ en función de la probabilidad de error del segmento TCP.

Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

Se añadió también el efecto de la probabilidad de pérdida de segmentos debido al medio inalámbrico, adecuando la modelización a un incremento de la CWND hasta el agotamiento de espacio en el *buffer* del receptor, es decir, estableciendo una retransmisión de paquetes recibidos con errores sin la necesidad de reducir la CWND. La ecuación correspondiente a la modelización del protocolo propuesto es la siguiente:

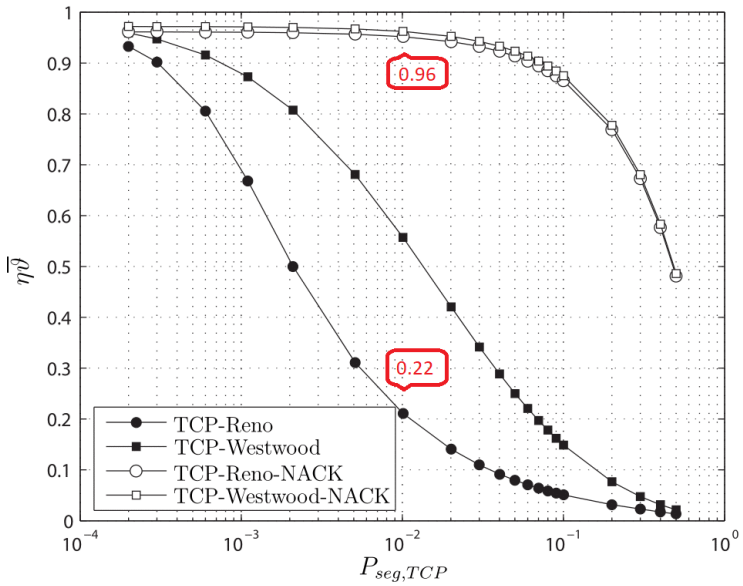
$$\bar{\vartheta} = \sum_{i=2}^{C} \left( \pi_{(W_o=i)} \sum_{n=2}^{n_{of}(W_o=i)} \frac{(n-1)\left(1-P_{seg,TCP}\right)}{\Delta t(W_o=i,n)} P_r\{n|W_o=i\} N_{TCP} \right), [bits/s] \qquad (1)$$

Donde $C$ corresponde al número de ráfaga en el cual se alcanza el tamaño de ventana máxima sin considerar el *buffer* del receptor, $W_o$ es la CWND inicial, $n_{of}$ es el índice del paquete perdido cuando se satura el *buffer* del receptor, $N_{TCP}$ es el número de bits por segmento TCP, $\Delta t(W_o, n)$ es el tiempo de llegada del enésimo ACK, $\pi_{W_o}$ son los valores del vector de estabilidad del proceso de Markov, $P_r\{n|W_o\}$ es la probabilidad de que un segmento se haya perdido debido a que se tenga una ventana inicial de congestión, y $P_{seg,TCP}$ corresponde a la probabilidad de pérdida de un segmento TCP debido exclusivamente a los problemas del medio inalámbrico.

Asimismo, de la ecuación 1 se obtuvo una comparación gráfica entre el $\bar{\theta}$ normalizado ($\eta\bar{\theta}$) del protocolo TCP-Reno, TCP-Westwoody sus versiones modificadas (TCP-Reno-NACK y TCP-Westwood-NACK) en función de $P_{seg,TCP}$. En la figura 1 se muestran las curvas generadas para cada protocolo.

Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

**Figura 1.** $\eta\bar{\theta}$ teórico en función de $P_{seg,TCP}$



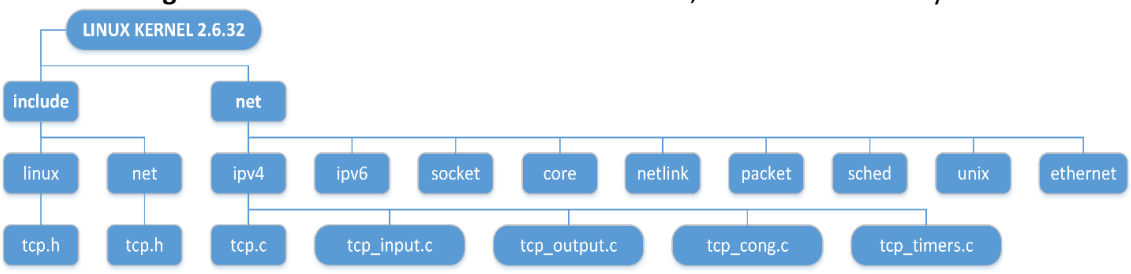

Fuente: Elaboración propia

Ambas versiones mejoradas del protocolo TCP tienen un comportamiento similar y un mejor desempeño en comparación con sus versiones genéricas. Específicamente, TCP-Reno-NACK (en adelante TCP-NACK) tiene una ganancia de $(0.96 - 0.22) \cdot 100/0.22 = 336\ \%$ (3.36 veces mayor) en comparación con TCP-Reno en $P_{seg,TCP} = 10^{-2}$ (en adelante TCP genérico).

## Implementación en Linux

Para implementar el protocolo TCP-NACK se modificó el Kernel de Linux en su versión 2.6.32 obtenido de los repositorios de la distribución Ubuntu, por ser un sistema de libre acceso y de código abierto (Herbert, 2004). Específicamente, se modificaron los directorios *include* y *net*, los cuales se encuentran distribuidos dentro del Kernel, como se muestra en la figura 2.

**Figura 2.** Estructura del Kernel de Linux 2.6.32, directorios *include* y *net*




Fuente: Elaboración propia

Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

### A. *Sockets* TCP

En Linux, los diferentes protocolos de comunicación son implementados a través de *sockets*, los cuales funcionan como una interfaz común entre el usuario y los diferentes sistemas de archivos y dispositivos. Existen tres estructuras de datos para el manejo de *sockets*. La primera se denomina *socket buffer*, la cual almacena la información general del paquete. La segunda se llama *socket*, y registra las conexiones abiertas. La última se denomina *sock*, y mantiene el estado de las conexiones abiertas (Herbert, 2004).

### B. Envío de datos mediante TCP

TCP es un protocolo que asegura la transmisión confiable efectuando un control del flujo de datos. Dentro del Kernel de Linux existen tres funciones principales que intervienen en el proceso de envío de datos y control de flujo mediante TCP:

1) ***tcp_sendmsg*:** Copia los datos en el espacio del usuario hacia el espacio del Kernel de Linux, en donde son asignados a los *socket buffers* y divididos en segmentos más pequeños.
2) ***tcp_send_skb*:** Organiza los datos en el *socket buffers* a la cola de transmisión y decide si la transmisión puede llevarse a cabo o no.
3) ***tcp_transmit_skb*:** Construye el encabezado TCP y envía los segmentos hacia la capa de red.

### C. Recepción de datos mediante TCP

Los paquetes recibidos deben ser trasladados desde la capa de red hacia la capa de transporte mediante las siguientes funciones para el procesamiento del encabezado TCP y de los datos contenidos en el segmento:

1) ***tcp_v4_rcv*:** Verifica la integridad del paquete, comprueba que se encuentre destinado hacia ese computador, procesa el *checksum* de la capa de transporte y remueve el encabezado IP.
2) ***tcp_v4_do_rcv*:** Verifica que el segmento recibido contenga un encabezado completo y comprueba el estado actual de la conexión TCP.
3) ***tcp_rcv_established*:** Si el estado actual de la conexión TCP se define como *established*, procesa los segmentos recibidos y copia los datos hacia el espacio del usuario. Si los segmentos no presentan errores, se ejecuta un proceso rápido denominado *Fast Path*; caso contrario se ejecuta *Slow Path*.

Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

### D. Modificación del encabezado TCP

Para crear la bandera correspondiente a la notificación NACK se tomó del encabezado del segmento TCP un bit del campo reservado, tal como se muestra en la figura 3. Con esta modificación, si el transmisor recibe un paquete de acuse de recibo con el bit de NACK igual a 1, significa que el receptor recibió un paquete defectuoso y solicita su retransmisión sin una reducción de la CWND. En cambio, si en el acuse de recibo el bit de NACK es igual a 0, este corresponde a una notificación ACK normal, lo cual hace que el transmisor envíe paquetes de la manera estándar.

**Figura 3.** Estructura de la cabecera TCP-NACK

<table>
<tr><td colspan="5">Puerto de origen</td><td colspan="5">Puerto de destino</td></tr>
<tr><td colspan="10">Número de secuencia</td></tr>
<tr><td colspan="10">número de acuse de recibo (ACK)<br>o número de acuse de recibo negativo (NACK)<br>dependiendo de la bandera activada</td></tr>
<tr><td>Longitud de cabecera</td><td>Reservado</td><td>NACK</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Tamaño de ventana</td></tr>
<tr><td colspan="5">Suma de verificación (CRC)</td><td colspan="5">Puntero urgente</td></tr>
<tr><td colspan="10">Opciones</td></tr>
<tr><td colspan="10">Datos</td></tr>
</table>

Fuente: Elaboración propia

El código que define el encabezado TCP se encuentra localizado en el archivo */include/Linux/tcp.h*. Para hacer la bandera NACK se ha creado un nuevo atributo llamado *nack* dentro de dicho archivo, lo cual reduce a tres bits el atributo *res1*, que corresponde al campo TCP reservado.

Para definir la posición de las banderas dentro del encabezado TCP, se debe modificar la estructura *tcp_flag_world* con el formato *TCP_FLAG_FLAGNAME = __cpu_to_be32 (0xXXXXXXXX)*, en donde cada *X* corresponde a una posible posición de la bandera. Por lo tanto, para asignar la posición de la bandera NACK, se añadió la línea de código *TCP_FLAG_NACK = __cpu_to_be32 (0x01000000)* dentro de dicha estructura.

Por último, es necesario definir dentro de la estructura *tcp_skb_cb* el valor que tomará la bandera NACK cuando es asignada a un segmento. Por ese motivo, se creó la línea de código *#define TCPCB_FLAG_NACK 0x100* dentro de dicha estructura. También es necesario ampliar el tipo del atributo *flags* de *u8* (8 bits) a *u16* (16 bits).

Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

### E. Envío de notificaciones NACK

Debido a la similitud entre las notificaciones NACK y ACK, la implementación de la función para el envío de NACK se basó en la función *tcp_send_ack*, la cual se localiza en el archivo */net/ipv4/tcp_output.c*.

### E.1 Función *tcp_send_nack*

Esta función, que se declara dentro del archivo */include/net/tcp.h*, realiza el envío de notificaciones NACK en cuatro pasos. En primer lugar, se comprueba el estado de la conexión TCP: si se verifica que la conexión fue reiniciada, no se envía la notificación NACK y la función finaliza. Caso contrario, se invoca la función *skb_reserve*, la cual crea un *socket buffer* con un espacio de memoria correspondiente al tamaño máximo de la cabecera TCP. Posteriormente, mediante la función *tcp_init_nondata_skb*, se envían al *buffer* de control el estado positivo para la bandera NACK y el número de secuencia a ser retransmitido. Finalmente, con la función *tcp_transmit_skb*, se transmite el segmento a la capa de red.

### E.2 Modificación en *tcp_transmit_skb*

Si el segmento a ser transmitido tiene levantada la nueva bandera *TCP_FLAG_NACK*, entonces el campo "número de acuse de recibo negativo" debe contener el valor del número de secuencia del segmento TCP detectado con errores. Finalmente, la función *tcp_send_nack* es invocada dentro de la función *tcp_rcv_established* justo después de detectar un error.

### E.3 Recepción de notificaciones de error NACK

Dentro del tratamiento de los segmentos recibidos existen dos posibles caminos: *Fast Path* y *Slow Path*. Todos los segmentos con bandera NACK levantada emplean *Slow Path*, en donde primero se verifica la integridad del segmento TCP mediante una suma de comprobación o *checksum* para consecuentemente verificar el estado de la bandera NACK. De resultar positivo, se invoca la función *tcp_retransmit_skb* para una retransmisión inmediata del segmento solicitado, el cual es determinado por la función *tcp_write_queue_head*.

Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

**Experimentación y resultados**

El escenario de pruebas consiste de dos computadores cargados con el Kernel modificado de Linux, conectados por un enlace inalámbrico emulado mediante el *software* Netem, el cual se encuentra instalado en un tercer computador intermedio con dos interfaces de red configuradas en modo puente, tal como se indica en la figura 4. La emulación del canal se debe a la necesidad de mantener un valor fijo de tasa de error para las múltiples inyecciones de tráfico.

**Figura 4.** Topología de la prueba de emulación


Fuente: Elaboración propia

Netem es una utilidad disponible en el Kernel de Linux que permite emular un enlace al especificar parámetros de ancho de banda, retardo, pérdidas y control de tráfico, utilizando probabilidad estadística (Hemminger, 2005). El enlace inalámbrico emulado contiene los parámetros descritos en la tabla 1.

**Tabla 1.** Parámetros para la emulación del enlace inalámbrico

| Parámetro | Valor |
|---|---|
| BR | 5 Mbits/s |
| delay | 70 ms |
| Probabilidad de error | 0.5 %, 1 % y 3 % |

Fuente: Elaboración propia

Adicionalmente, en cada equipo terminal se emplearon las herramientas Iperf y Tcpprobe, disponibles en los repositorios de Linux. Estas herramientas permiten obtener valores de CWND y $\bar{\theta}$.

## A. Configuración Iperf

Iperf permite realizar una conexión TCP cliente-servidor y medir su tasa de transferencia de datos. En la tabla 2 se detallan los comandos necesarios para su configuración por el terminal, tanto para el equipo transmisor como para el receptor.

Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

**Tabla 2.** Comandos de configuración para iperf

| Terminal TCP | Línea de comando |
|---|---|
| Transmisor | iperf –c IPdirection –t time |
| Receptor | iperf–s |

Fuente: Elaboración propia

## B. Configuración Tcpprobe

Tcpprobe es un módulo de Linux que permite extraer parámetros como la CWND y los umbrales de *Slow Start* y *Congestion Avoidance*. Suele ejecutarse en el terminal emisor junto con la herramienta Iperf para después almacenar los resultados en el archivo *data.out*. En el código 1 se muestra su uso en conjunto.

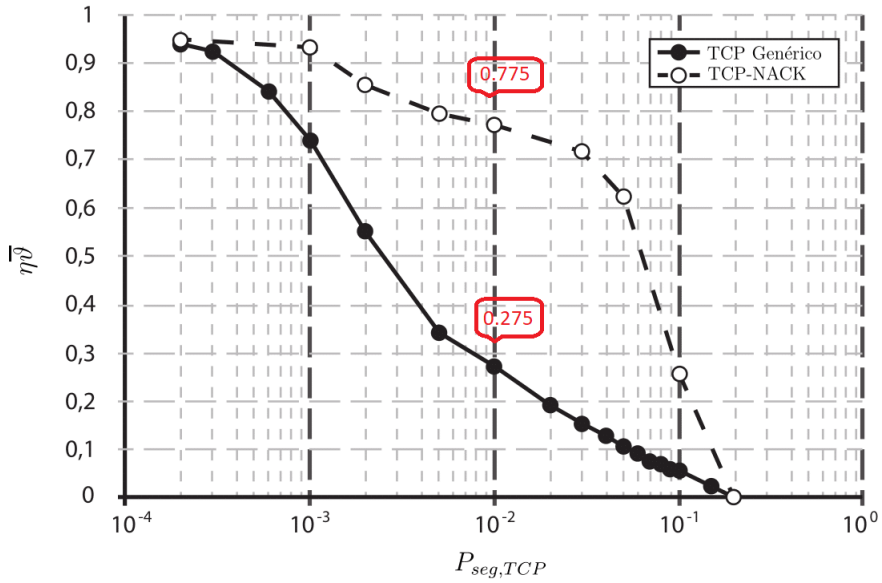
**Código 1.** Líneas de código Tcpprobe e Iperf para obtener la CWND

```
#! /bin/bash
modprobetcp_probe port=5001
cat /proc/net/tcpprobe>/tmp/data.out&
pid=$!
iperf -c 192.168.1.2
kill $pid
```

Con todas estas herramientas se obtuvo una comparación entre el $\eta\bar{\theta}$ del TCP genérico y TCP-NACK en función de $P_{seg,TCP}$. La figura 5 muestra las curvas resultantes para cada protocolo emulado.

Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

**Figura 5.** $\eta\bar{\theta}$ emulado en función de $P_{seg,TCP}$




Fuente: Elaboración propia

A pesar de que en la emulación no se presentó un incremento tan alto como los resultados teóricos sugieren (figura 1), se logró alcanzar una mejora de $(0.775 - 0.275) \cdot 100/0.275 = 182$ % (1.82 veces mayor) a un $P_{seg,TCP} = 10^{-2}$.

También se obtuvo el tamaño de la CWND en función del valor de BER a lo largo del tiempo, con el cual se calculó el $\bar{\theta}$ obtenido como el área bajo la curva generada por la CWND. Considerando un BR de cinco Mbps, bajo valores de BER de 0.5 %, 1 % y 5 %, se consiguieron los resultados mostrados en la tabla 3.

**Tabla 3.** Resultados del enlace emulado

| BER (%) | TCP genérico | TCP-NACK | |
|---|---|---|---|
| | $\bar{\theta}$ (Mbits/s) | $\bar{\theta}$ (Mbits/s) | Porcentaje de mejora (%) $(\bar{\theta}_{NACK} - \bar{\theta}_{Gen}) \cdot 100/\bar{\theta}_{Gen}$ |
| 0.5 | 1.53 | 1.79 | 17 |
| 1 | 1.27 | 1.46 | 15 |
| 5 | 0.52 | 0.74 | 42 |

Fuente: Elaboración propia

Para cada valor de BER se puede observar una mejora en la velocidad de transmisión de TCP-NACK en comparación con los resultados de TCP genérico, siendo la más grande a un valor de BER de 5 %. La mejora de TCP-NACK aumenta linealmente con el valor de BER debido a la cantidad de paquetes que pueden ser notificados con NACK y, por ende, retransmitidos sin la reducción de la CWND.

Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

## Conclusiones y trabajos futuros

Debido a que el protocolo TCP fue elaborado para trabajar en redes cableadas, su funcionamiento no es el adecuado para medios inalámbricos, en donde los paquetes que sufren daños —ya sea por interferencias, obstáculos o desvanecimientos— son descartados y erróneamente considerados como consecuencia de congestión. Esta suposición conlleva a la reducción innecesaria de la CWND, que disminuye consecuentemente el rendimiento de la red.

El protocolo propuesto TCP-NACK —el cual fue modelado, implementado y emulado— resuelve este problema mediante la incorporación de una notificación de error NACK al funcionamiento del protocolo TCP genérico. Esta notificación indicará al transmisor la llegada de un paquete defectuoso y procederá con una retransmisión inmediata de dicho paquete sin la reducción de la CWND.

Los resultados obtenidos demuestran un mejor rendimiento del protocolo TCP-NACK frente a su contraparte genérica, bajo las diferentes probabilidades de error y tasas de inyección empleadas. A pesar de que los resultados experimentales son menores a los resultados teóricos (aumento esperado de 336 %), se logró alcanzar un incremento del $\bar{\theta}$ de 182 % en comparación con el protocolo TCP genérico a un $P_{seg,TCP} = 10^{-2}$. Adicionalmente, con los diferentes valores de BER se alcanzó, en promedio, una mejora de 25 %, con lo que se demuestra el mejor desempeño de TCP-NACK bajo dichos parámetros emulados.

En comparación con el protocolo TCP-ELN, el cual es el más similar al protocolo propuesto TCP-NACK, a un $P_{seg,TCP} = 10^{-2}$, TCP-ELN supera al protocolo TCP-Reno genérico en 119 %, tal como se indica en Buchholcz, Ziegler y Do (2005); sin embargo, TCP-NACK supera al mismo protocolo, bajo la misma condición de pérdida de segmentos, en 182 %, como se observa en la figura 5. Esta diferencia puede ser causada por la inmediata retransmisión de TCP-NACK y por el uso de un solo bit para la notificación en lugar de los varios bits que TCP-ELN necesita del campo opcional dentro del encabezado TCP.

Como trabajos futuros, estamos interesados no solo en ampliar la evaluación del desempeño de TCP-NACK considerando un mayor número de valores de BER, sino también analizar su rendimiento con otras métricas de calidad de servicio, como son *delay*, *jitter* y paquetes perdidos. Asimismo, nos interesa presentar una versión mejorada de TCP-NACK, en donde por cada notificación NACK recibida se aumente la CWND y se realice la retransmisión de todos los paquetes a partir del notificado con error. Nuestra hipótesis es que con esta

adaptación el protocolo TCP-NACK tendría la capacidad de adelantarse a paquetes defectuosos aún no notificados por NACK o ACK duplicados, lo cual aumentaría el $\bar{\theta}$ resultante.

Finalmente, debido a los frecuentes lanzamientos de nuevas versiones del Kernel de Linux, estamos muy interesados en desarrollar un ejecutable que nos permita aplicar el funcionamiento de TCP-NACK en el código fuente del sistema. Con esto evitaríamos el proceso de modificar línea por línea el Kernel de Linux y adaptaríamos el protocolo para cualquier versión disponible.

**Agradecimientos**

Los autores desean expresar su agradecimiento a la Universidad de las Fuerzas Armadas, ESPE, por el apoyo económico en el desarrollo de este trabajo a través del proyecto 2010-PIT-008.

## Referencias

Buchholcz, G., Ziegler, T. and Do, T. (2005). TCP-ELN: on the protocol aspects and performance of explicit loss notification for TCP over wireless networks. *First International Conference on Wireless Internet*, 172–179. doi:10.1109/WICON.2005.31.

Chakraborty, S. and Nandi, S. (2014). Evaluating transport protocol performance over a wireless mesh backbone. *Performance Evaluation*, *79*, 198–215. doi:10.1016/j.peva.2014.07.013.

Hemminger, S. (2005). Network Emulation with NetEm. *Proceedings of the 6th Australian National Linux Conference (LCA 2005)*, 18–23. Retrieved from https://goo.gl/3qchck.

Herbert, T. F. (2004). *The Linux TCP/IP Stack: Networking for Embedded Systems (Networking Series)*. Rockland, MA, USA: Charles River Media, Inc.

Hung, K. L. and Bensaou, B. (2011). TCP performance optimization in multi-cell WLANs. *Performance Evaluation*, *68*(9), 806–824. doi:10.1016/j.peva.2011.04.002.

Olmedo, G. (2008). *Controle de congestionamento do protocolo TCP em sistemas de comunicação sem fio CDMA usando estrategia de detecção multiusuario, arranjo de antenas e correção de erro FEC* (tesis de doctorado). Universidade Estadual de Campinas. Facultade de Engenharia Elétrica e de Computação, Brasil. Recuperado de https://goo.gl/ef2DJD.

Postel, J. (1981). *Transmission Control Protocol*. *Rfc 793*. doi:10.17487/rfc0793.

Reddy, N., Reddy, P., y Padmavathamma, M. (2017). Efficient Traffic Engineering Strategies for Improving the Performance of TCP Friendly Rate Control Protocol. *Future Internet*, *9*(74). doi:10.3390/fi9040074.

Stevens, W. R. (1997). TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. doi:10.17487/RFC2001.

Tian, Y., Xu, K. and Ansari, N. (2005). TCP in wireless environments: Problems and solutions. *IEEE Communications Magazine*, *43*(3), S27-S32. doi:10.1109/MCOM.2005.1404595.

Xylomenos, G., Polyzos, G. C., Mahonen, P. and Saaranen, M. (2001). TCP performance issues over wireless links. *IEEE Communications Magazine*, *39*(4), 1-12. doi:10.1109/35.917504.

Zanella, A., Procissi, G., Gerla, M., and Sanadidi, M. Y. (2001). TCP Westwood: analytic model and performance evaluation. *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, *3*, 1703–1707. doi:10.1109/GLOCOM.2001.965870.

Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

| Rol de Contribución | Autor(es) |
|---|---|
| **Conceptualización** | DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL<br>ROMÁN ALCIVES LARA CUEVA APOYO APOYO |
| **Metodología** | DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL<br>ROMÁN ALCIVES LARA CUEVA APOYO APOYO |
| **Software** | DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL<br>ROMÁN ALCIVES LARA CUEVA APOYO APOYO |
| **Validación** | DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL<br>ROMÁN ALCIVES LARA CUEVA APOYO APOYO |
| **Análisis Formal** | DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL<br>ROMÁN ALCIVES LARA CUEVA APOYO APOYO |
| **Investigación** | DIEGO XAVIER MARTÍNEZ HIDALGO |
| **Recursos** | DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL<br>ROMÁN ALCIVES LARA CUEVA APOYO APOYO<br>UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE APOYO |
| **Curación de datos** | DIEGO XAVIER MARTÍNEZ HIDALGO |
| **Escritura - Preparación del borrador original** | DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL<br>ROMÁN ALCIVES LARA CUEVA APOYO APOYO |
| **Escritura - Revisión y edición** | DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL<br>ROMÁN ALCIVES LARA CUEVA APOYO APOYO |
| **Visualización** | DIEGO XAVIER MARTÍNEZ HIDALGO |
| **Supervisión** | DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL<br>ROMÁN ALCIVES LARA CUEVA APOYO APOYO |
| **Administración de Proyectos** | DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL<br>ROMÁN ALCIVES LARA CUEVA APOYO APOYO |
| **Adquisición de fondos** | DIEGO XAVIER MARTÍNEZ HIDALGO |

Reci
Revista Iberoamericana de las Ciencias Computacionales e Informática
ISSN: 2007 - 9915

**Síntesis curricular de los autores**

**Román Lara Cueva**

Ingeniero en Electrónica y Telecomunicaciones por la Escuela Politécnica Nacional (Ecuador, 2001), magíster en Sistemas Inalámbricos y Tecnologías Relacionadas del Politécnico di Torino (Italia, 2005). Maestría y doctorado en Redes de Telecomunicación para Países en Desarrollo por la Universidad Rey Juan Carlos (España, 2010 y 2015, respectivamente). Se unió al Departamento de Ingeniería Eléctrica de la Universidad de las Fuerzas Armadas, ESPE, en 2002 y ha sido profesor titular principal de dicha casa de estudios desde 2005. Ha participado en más de diez proyectos de investigación con fondos públicos, y ha dirigido cinco de ellos. Sus principales intereses de investigación incluyen el procesamiento digital de señales, ciudades inteligentes, sistemas inalámbricos y la teoría de aprendizaje automático.

**Diego Martínez Hidalgo**

Bachiller en Eléctrica y Electrónica por el Colegio Técnico Experimental Salesiano Don Bosco (Ecuador, 2009). Ingeniero en Electrónica y Telecomunicaciones por la Universidad de las Fuerzas Armadas, ESPE (Ecuador, 2017). En el 2018 culminó el segundo nivel para la certificación en redes de computadoras y telecomunicaciones Cisco CCNA2. Sus intereses de investigación incluyen los sistemas de comunicaciones inalámbricas, las redes Ethernet y el diseño de soluciones electrónicas.