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

Introduction

The Transmission Control Protocol (TCP) was designed with a focus on wired networks, and therefore misinterprets congestion the cause of packet loss (PL) in wireless media when in reality is caused by factors of the communication channel, such as packet collisions, interference and noise (Xylomenos, Polyzos, Mahonen and Saaranen, 2001). To control congestion in the network, TCP uses an acknowledgment (ACK), which is used by the receiver to indicate that the segment was taken without errors and to specify the next expected segment. If the transmitter does not detect any ACK for a preset time, the packets are forwarded or the connection is canceled (Postel, 1981). TCP can use four mechanisms for the control of the congestion window (CWND of the English Congestion Window): Slow Start, Congestion Avoidance, Fast Retransmit or Fast Recovery. These mechanisms seek to achieve high performance in the network and avoid a collapse due to congestion. Slow Start is used to exponentially increase the transmission rate (BR of English Bit Rate) in the medium until a loss of segments is detected.

From this point, the Congestion Avoidance algorithm acts, with which the transmission rate increases linearly and decreases when detecting duplicate ACKs. Fast Retrasmit and Fast Recovery allow you to resend segments immediately after receiving three duplicate ACKs. It should be noted that the passage from one algorithm to another occurs when missing packets are detected (Stevens, 1997). In networks with a low bit error rate (VER of the English Bit Error Rate), these algorithms achieve this objective; however, in wireless networks there are

erroneous executions of the four algorithms and low transmission rates, as indicated in Tian, Xu y Ansari (2005).

The TCP protocol has different versions that seek to optimize the CWND by means of an analysis of the ACK notification rate (Olmedo, 2008), two of the most popular versions are TCP Reno and TCP Westwood. TCP Reno halves the CWND after receiving three duplicate ACKs; In contrast, TCP Westwood selects a Slow Start threshold and a CWND that are consistent with the effective connection rate at the time of congestion (Zanella, Procissi, Gerla and Sanadidi, 2001). To reinforce the effort of CWND control in wireless networks, such as those using Wi-Fi technology, different mechanisms have been developed, for example, by using explicit loss notifications, adding explicit bits to the TCP segment as feedback to the source for determine the nature of the loss, which improves up to 30% throughput values ($\bar{\theta}$) compared to TCP-NewReno, for simulated web traffic under high error probabilities (Buchholcz, Ziegler and Do, 2005).

This proposed protocol, defined as TCP-ELN, however, has the disadvantage of not reporting the defective packet immediately after detecting it. TCP-ELN has to wait for the successful reception of a package to notify all previous losses as additional information inserted in the generated ACK. Another problem in wireless networks is the retransmission time (RTO of the English Retransmission Time Out) which is calculated based on the round trip time (RTT of the English Round Trip Time), which is variant in wireless media and can reach to cause false RTO (Chakraborty and Nandi, 2014).

This dependence on RTT can be seen in closed environments, where, together with hidden terminal problems and interferences between basic service sets, des$\bar{}$imbalances occur. As a solution to this scenario, control of the containment windows of the MAC layer has been proposed through the use of an analytical model of nonlinear equations. With this, a fair distribution of $\bar{\theta}$and a confidence interval of 99.98% at a rate of two Mbps were achieved (Hung and Bensaou, 2011). Likewise, based on the same RTT dependency problem, TCP friendly flow control protocol (TFRC) suffers from PL and high transmission times when applied to unstable links or long distance links.

As a solution, an improved calculation of RTT and RTO values has been proposed, with which the TFRC protocol is given the ability to differentiate the cause of PL. Through simulations, the improved TFRC protocol reaches 22% greater $\bar{\theta}$compared to the standard TFRC protocol (N. Reddy, Reddy and Padmavathamma, 2017).

According to the above, it can be said that there is a need to modify the TCP protocol to apply a simple and immediate retransmission of the damaged package without reducing the CWND. For this reason, the purpose of this paper is to design, implement and evaluate a new TCP protocol for wireless environments. To fulfill this objective, a negative ACK confirmation segment (NACK of the English Negative Acknowledgment) was added within the TCP protocol header. These negative notifications will be responsible for informing the transmitter that a packet was received with errors, instead of just discarding it. With NACK notifications, the TCP protocol is prevented from confusing PL due to the instability of the PL channel caused by congestion, which avoids an unnecessary reduction of the CWND.

For this, first of all, a mathematical modeling of the behavior of the proposed protocol was started; then modifications were made to the kernel (source code) of Linux to introduce it to the operation of the Operating System, and finally it was evaluated in an emulated scenario in order to obtain precise values of performance under controlled conditions.

In summary, the rest of this document is organized as follows. Section II presents the original mathematical proposal that gave way to the implementation. Section III discusses the modified TCP Reno protocol design within the Linux Kernel. Section IV presents the development of the test scenario and the results obtained. Finally, section V discusses future work.

Original mathematical analysis

Initially, the proposed protocol was modeled in Olmedo's research (2008), where the stages of Slow Start and Congestion Avoidance were considered from the TCP versions Reno and Westwood. The evolution of the CWND was established as a Markow process, which allowed obtaining an equation of $\bar{\theta}$ based on the error probability of the TCP segment.

The effect of the probability of loss of segments due to the wireless medium was also added, adapting the modeling to an increase of the CWND until the space exhaustion in the receiver buffer, that is, establishing a retransmission of received packets with errors without the need to reduce the CWND. The equation corresponding to the modeling of the proposed protocol is as follows:

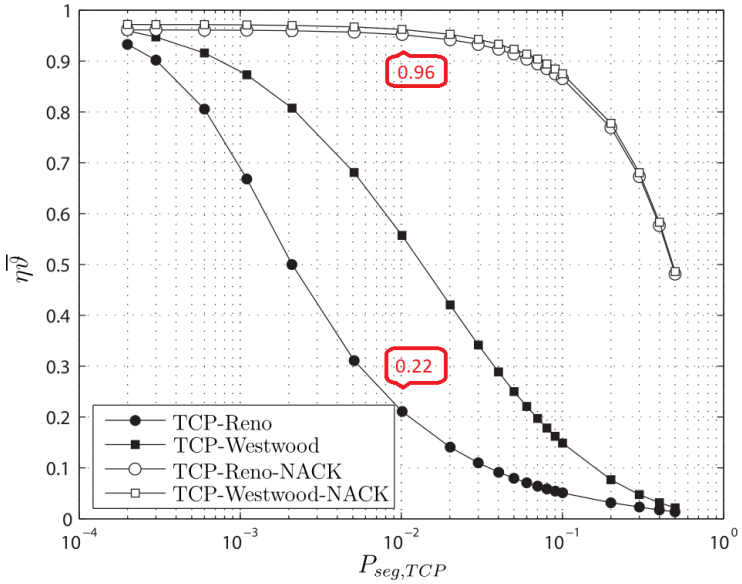
$$\bar{\vartheta} = \sum_{i=2}^{C} \left(\pi_{(W_o=i)} \sum_{n=2}^{n_{of}(W_o=i)} \frac{(n-1)\left(1-P_{seg,TCP}\right)}{\Delta t(W_o=i,n)} P_r\{n|W_o=i\} N_{TCP} \right), [bits/s] \qquad (1)$$

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

Where C corresponds to the burst number in which the maximum window size is reached without considering the receiver buffer, W_o is the initial CWND, n_{of} is the index of the lost packet when the receiver buffer is saturated, N_{TCP} is the number of bits per TCP segment, $\Delta t(W_o, n)$ is the time of arrival of the umpteenth ACK, π_{W_o} are the values of the stability vector of the Markov process, $P_r\{n|W_o\}$ is the probability that a segment has been lost due to having an initial window of congestion, y $P_{seg,TCP}$ corresponds to the probability of loss of a TCP segment due exclusively to the problems of the wireless medium.

Also, from equation 1 a graphical comparison between the $\bar{\theta}$ normalized ($\eta\bar{\theta}$) of the TCP-Reno protocol, TCP-Westwoody its modified versions (TCP-Reno-NACK and TCP-Westwood-NACK) depending on $P_{seg,TCP}$. Figure 1 shows the generated curves for each protocol.

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



Fuente: Elaboración propia

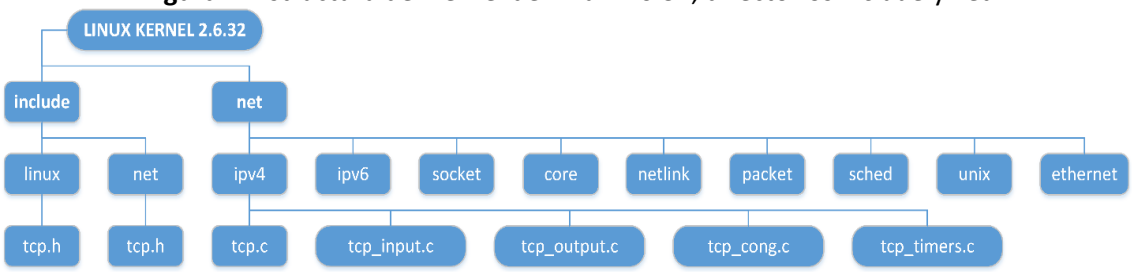
Both improved versions of the TCP protocol have similar behavior and better performance compared to their generic versions. Specifically, TCP-Reno-NACK (hereinafter TCP-NACK) has a gain of (0.96-0.22) · 100 / 0.22 = 336% (3.36 times greater) compared to TCP-Reno in $P_{seg,TCP} = 10^{-2}$ (hereinafter generic TCP).

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

Implementation in Linux

To implement the TCP-NACK protocol, the Linux kernel was modified in version 2.6.32 obtained from the Ubuntu distribution repositories, as it is an open access and open source system (Herbert, 2004). Specifically, the include and net directories were modified, which are distributed within the Kernel, as shown in Figure 2.

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



Fuente: Elaboración propia

A. *Sockets* TCP

In Linux, the different communication protocols are implemented through sockets, which work as a common interface between the user and the different file systems and devices. There are three data structures for the management of sockets. The first is called socket buffer, which stores the general information of the package. The second one is called socket, and it records the open connections. The last one is called sock, and it keeps the state of open connections (Herbert, 2004).

B. Sending data through TCP

TCP is a protocol that ensures reliable transmission by controlling the flow of data. Within the Linux Kernel there are three main functions involved in the process of data transmission and flow control through TCP:

1) ***tcp_sendmsg*:** Copy the data in the user's space to the Linux kernel space, where they are assigned to the socket buffers and divided into smaller segments.
2) tcp_send_skb: It organizes the data in the socket buffers to the transmission queue and decides if the transmission can be carried out or not.
3) tcp_transmit_skb: Build the TCP header and send the segments to the network layer.

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

C. Data reception by TCP

The received packets must be transferred from the network layer to the transport layer through the following functions for processing the TCP header and the data contained in the segment:

1) ***tcp_v4_rcv*:** Verify the integrity of the package, check that it is destined to that computer, process the checksum of the transport layer and remove the IP header.
2) tcp_v4_do_rcv: Verify that the received segment contains a complete header and check the current status of the TCP connection.
3) tcp_rcv_established: If the current state of the TCP connection is defined as established, it processes the received segments and copies the data to the user's space. If the segments do not show errors, a fast process called Fast Path is executed; Otherwise, Slow Path is executed.

D. Modification of the TCP header

To create the flag corresponding to the NACK notification, a bit of the reserved field was taken from the TCP segment header, as shown in figure 3. With this modification, if the transmitter receives an acknowledgment packet with the NACK bit equal to 1, means that the receiver received a defective packet and requests its retransmission without a reduction of the CWND. On the other hand, if the NACK bit is equal to 0 in the acknowledgment, it corresponds to a normal ACK notification, which causes the transmitter to send packets in the standard manner.

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)
o número de acuse de recibo negativo (NACK)
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

The code that defines the TCP header is located in the /include/Linux/tcp.h file. To make the NACK flag, a new attribute called nack has been created inside that file, which reduces the res1 attribute to three bits, which corresponds to the reserved TCP field.

To define the position of the flags within the TCP header, the structure tcp_flag_world should be modified with the format TCP_FLAG_FLAGNAME = __cpu_to_be32 (0xXXXXXXXX), where each X corresponds to a possible position of the flag. Therefore, to assign the position of the NACK flag, the code line TCP_FLAG_NACK = __cpu_to_be32 (0x01000000) was added within that structure.

Finally, it is necessary to define within the structure tcp_skb_cb the value that the NACK flag will take when it is assigned to a segment. For this reason, the line of code #define TCPCB_FLAG_NACK 0x100 was created within that structure. It is also necessary to extend the type of the flags attribute from u8 (8 bits) to u16 (16 bits).

E. Sending NACK notifications

Due to the similarity between the NACK and ACK notifications, the implementation of the function for sending NACK was based on the function tcp_send_ack, which is located in the file /net/ipv4/tcp_output.c.

E.1 tcp_send_nack function

This function, which is declared inside the /include/net/tcp.h file, sends NACK notifications in four steps. First, the status of the TCP connection is checked: if it is verified that the connection was rebooted, the NACK notification is not sent and the function ends. Otherwise, the skb_reserve function is invoked, which creates a buffer socket with a memory space corresponding to the maximum size of the TCP header. Subsequently, by means of the function tcp_init_nondata_skb, the positive state for the NACK flag and the sequence number to be retransmitted are sent to the control buffer. Finally, with the function tcp_transmit_skb, the segment is transmitted to the network layer.

Modification in tcp_transmit_skb

If the segment to be transmitted has the new flag TCP_FLAG_NACK raised, then the field "negative acknowledgment number" must contain the value of the sequence number of the

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

detected TCP segment with errors. Finally, the function tcp_send_nack is invoked inside the function tcp_rcv_established just after detecting an error.

E.2 Receiving NACK error notifications

Within the treatment of the received segments there are two possible paths: Fast Path and Slow Path. All the segments with a raised NACK flag use Slow Path, where first the integrity of the TCP segment is verified by means of a checksum or checksum to consequently verify the state of the NACK flag. If positive, the function tcp_retransmit_skb is invoked for an immediate retransmission of the requested segment, which is determined by the function *tcp_write_queue_head*.

Experimentation and results

The test scenario consists of two computers loaded with the modified Linux Kernel, connected by a wireless link emulated by the Netem software, which is installed on a third intermediate computer with two network interfaces configured in bridge mode, as described indicated in Figure 4. The emulation of the channel is due to the need to maintain a fixed value of error rate for multiple injections of traffic.

Figura 4. Topología de la prueba de emulación

Fuente: Elaboración propia

Netem is a utility available in the Linux Kernel that allows to emulate a link by specifying parameters of bandwidth, delay, losses and traffic control, using statistical probability (Hemminger, 2005). The emulated wireless link contains the parameters described in table 1.

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

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

Additionally, the Iperf and Tcpprobe tools, available in the Linux repositories, were used in each terminal equipment. These tools allow you to obtain CWND values and $\bar{\theta}$.

A. Iperf configuration

Iperf allows a client-server TCP connection and measures its data transfer rate. Table 2 details the commands necessary for its configuration by the terminal, both for the transmitting equipment and for the receiver.

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. Tcpprobe configuration

Tcpprobe is a Linux module that allows you to extract parameters such as the CWND and the thresholds of Slow Start and Congestion Avoidance. It is usually executed in the issuing terminal together with the Iperf tool and then stores the results in the data.out file. Code 1 shows its use as a whole.

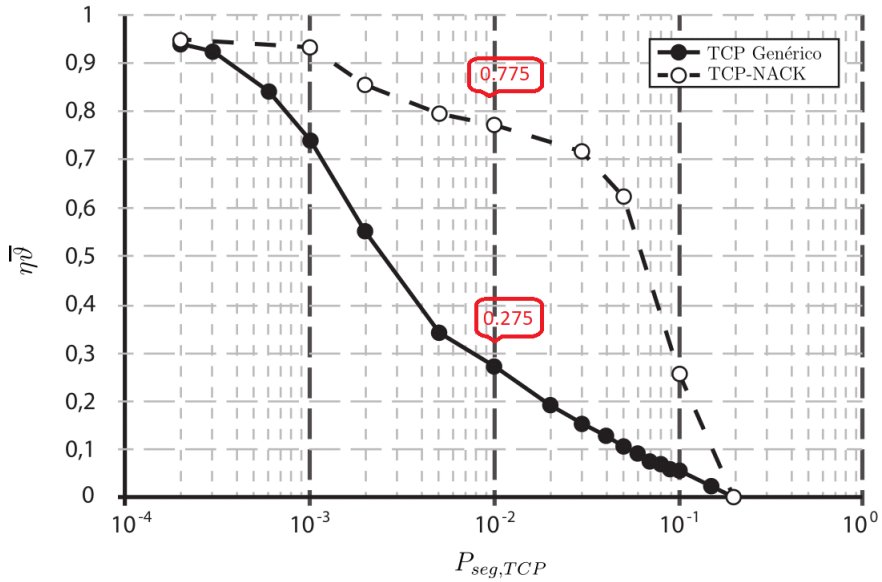
Código 1. Líneas de código Tcpprobe e Iperf para obtener la CWND

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

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

With all these tools a comparison was obtained between the $\eta\bar{\theta}$ of the generic TCP and TCP-NACK depending on $P_{seg,TCP}$. Figure 5 shows the resulting curves for each emulated protocol.

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



Fuente: Elaboración propia

Although the emulation did not show an increase as high as the theoretical results suggest (figure 1), it was possible to reach an improvement of $(0.775 - 0.275) \cdot 100/0.275 = 182$ % (1.82 veces mayor) to $P_{seg,TCP} = 10^{-2}$.

The size of the CWND was also obtained according to the BER value over time, with which the $\bar{\theta}$ obtained as the area under the curve generated by the CWND. Considering a BR of 5 Mbps, under BER values of 0.5%, 1% and 5%, the results shown in Table 3 were achieved.

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

For each BER value, an improvement in the transmission speed of TCP-NACK can be observed in comparison with the results of generic TCP, the largest being at a BER value of 5%. The TCP-NACK enhancement increases linearly with the BER value due to the number of

packets that can be notified with NACK and, therefore, retransmitted without the reduction of the CWND.

Conclusions and future work

Because the TCP protocol was developed to work in wired networks, its operation is not suitable for wireless media, where damaged packages - whether due to interference, obstacles or fading - are discarded and mistakenly considered as a consequence of congestion. . This assumption leads to the unnecessary reduction of the CWND, which consequently decreases the performance of the network.

The proposed TCP-NACK protocol - which was modeled, implemented and emulated - solves this problem by incorporating a NACK error notification into the operation of the generic TCP protocol. This notification will indicate to the transmitter the arrival of a defective package and will proceed with an immediate retransmission of said package without the reduction of the CWND.

The results obtained show a better performance of the TCP-NACK protocol compared to its generic counterpart, under the different error probabilities and injection rates used. Although the experimental results are lower than the theoretical results (expected increase of 336%), an increase of θ¯of 182% was achieved in comparison with the generic TCP protocol at P_ (sec, TCP) = 〖10 〗 ^ (- 2). Additionally, with the different BER values, an improvement of 25% was reached, on average, which demonstrates the best performance of TCP-NACK under said emulated parameters.

In comparison with the TCP-ELN protocol, which is the most similar to the proposed TCP-NACK protocol, to a $P_{seg,TCP} = 10^{-2}$, TCP-ELN exceeds the generic TCP-Reno protocol by 119%, as indicated in Buchholcz, Ziegler and Do (2005); however, TCP-NACK exceeds the same protocol, under the same condition of loss of segments, by 182%, as shown in figure 5. This difference can be caused by the immediate retransmission of TCP-NACK and by the use of a single bit for the notification instead of the various bits that TCP-ELN needs from the optional field within the TCP header.

As future work, we are interested not only in expanding the evaluation of the performance of TCP-NACK considering a greater number of BER values, but also analyzing its performance with other quality of service metrics, such as delay, jitter and lost packets. Likewise, we are interested in presenting an improved version of TCP-NACK, where for each received NACK notification the CWND is increased and the retransmission of all the packets from the notified

in error is performed. Our hypothesis is that with this adaptation the TCP-NACK protocol would have the ability to anticipate defective packets not yet notified by NACK or duplicate ACK, which would increase the $\bar{\theta}$ resulting.

Finally, due to the frequent releases of new versions of the Linux Kernel, we are very interested in developing an executable that allows us to apply the TCP-NACK operation in the source code of the system. With this we would avoid the process of modifying line by line the Linux Kernel and we would adapt the protocol for any available version.

Acknowledgment

The authors wish to express their gratitude to the University of the Armed Forces, ESPE, for the financial support in the development of this work through the project 2010-PIT-008.

References

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 ROMÁN ALCIVES LARA CUEVA APOYO APOYO
Metodología	DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL ROMÁN ALCIVES LARA CUEVA APOYO APOYO
Software	DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL ROMÁN ALCIVES LARA CUEVA APOYO APOYO
Validación	DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL ROMÁN ALCIVES LARA CUEVA APOYO APOYO
Análisis Formal	DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL ROMÁN ALCIVES LARA CUEVA APOYO APOYO
Investigación	DIEGO XAVIER MARTÍNEZ HIDALGO
Recursos	DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL ROMÁN ALCIVES LARA CUEVA APOYO APOYO UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE APOYO
Curación de datos	DIEGO XAVIER MARTÍNEZ HIDALGO
Escritura - Preparación del borrador original	DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL ROMÁN ALCIVES LARA CUEVA APOYO APOYO
Escritura - Revisión y edición	DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL ROMÁN ALCIVES LARA CUEVA APOYO APOYO
Visualización	DIEGO XAVIER MARTÍNEZ HIDALGO
Supervisión	DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL ROMÁN ALCIVES LARA CUEVA APOYO APOYO
Administración de Proyectos	DIEGO XAVIER MARTÍNEZ HIDALGO PRINCIPAL 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.