AVAILABLE NETWORK BANDWIDTH SCHEMA TO IMPROVE PERFORMANCE IN TCP PROTOCOLS

Marcos Talau and Emilio Carlos Gomes Wille

Federal University of Technology - Paraná (UTFPR)
Av. Sete de Setembro 3165, 80230-901, Curitiba (PR), Brazil
talau@users.sourceforge.net, ewille@utfpr.edu.br

ABSTRACT

The TCP congestion control mechanism in standard implementations presents several problems, for example, large queue lengths in network routers and packet losses, a misleading reduce of the transmission rate when there are link failures, among others. This paper proposes a schema to congestion control in TCP protocols, called NGWA, witch is based on the network bandwidth. The NGWA provides information considering the available bandwidth of the network infrastructure to the endpoints of the TCP connection. Hence, it helps in choosing a better transmission rate for TCP improving its performance. Simulation results show superior performance of the proposed scheme when compared to those obtained by TCP New Reno and standard TCP. A physical implementation in the Linux kernel was performed to prove the correct operation of the proposal.

KEYWORDS

TCP, Congestion Control, Network Bandwidth, Linux.

1. INTRODUCTION

The Transmission Control Protocol (TCP) is the dominant transport protocol on the Internet. It supports a wide range of applications such as WEB, e-mail and, recently, emerging applications such as peer-to-peer. The TCP is based on a sliding window protocol and provides end-to-end, reliable, congestion controlled connections over the network [1].

Nowadays, given its importance, there is great interest in the analysis of its performance and on the overcoming of its limitations. Among the TCP problems we can mention: large queue lengths in network routers and packet losses, a misleading reduce of the transmission rate when there are link failures (e.g., wireless transmission problem), the network congestion recognition based on packet losses, and bandwidth unfair sharing [2,3,4,5,6].

The default TCP implementation uses the window of the received TCP segment and the congestion window (*cwnd*) to assign the burst of data to be transmitted. The congestion window tries to predict the network bandwidth. It is done simply by the exponential increase of the window until the occurrence of a loss, then the window is reduced and fewer bytes are transmitted. If the loss was not caused by congestion (a very common event in wireless networks), the transmission rate will reduce unnecessary [3,7,8]. The literature shows that if the TCP transmitter has information about the network bandwidth it could, quickly and properly, enforce a better congestion control mechanism [2,9].

DOI : 10.5121/ijcnc.2013.5504

This paper proposes a new congestion control scheme for TCP, called *New General Window Advertising* (NGWA). The NGWA provides information considering the available bandwidth of the network infrastructure to the endpoints of the TCP connection. Hence, NGWA helps in choosing a better transmission rate for TCP improving its performance. It is similar to GWA proposed in [10], but presents a different functioning principle. To evaluate the performance of our proposed scheme, simulation results (obtained using the NS-3 package [11]) are presented and analysed. Due to the good simulation results, an implementation in the Linux kernel was performed to physically prove the operation of the proposal.

The remainder of this paper is structured as follows. Section 2 presents the architecture and operation of the NGWA proposal. In Section 3 a set of simulation results is discussed. Section 4 describes the Linux implementation, as well as, analyzes numerical results. Finally, concluding remarks and suggestions are presented in Section 5.

2. THE NEW GENERAL WINDOW ADVERTISING

The NGWA is a new congestion control scheme for TCP, it provides to the TCP end points the amount of bandwidth available in the network infrastructure. The NGWA work as follows: the total number of bytes available in a router queue is recorded in a variable (W_{ngwa}), transiting in the network layer. This amount is stored in the *options* field of the IPv4 header (for compatibility with the IP protocol). The update of the variable is performed by the nodes in the packets route. Then, each node performs as follows: if $W_{nwga}^{node} < W_{nwga}^{packet}$ then $W_{nwga}^{node} = W_{nwga}^{packet}$.

Figure 1 illustrates the process. When a packet arrives the receiver, it is processed normally and the variable W_{ngwa} is extracted from the IP header and deposited into the memory. During the creation of an ACK segment to be sent to the transmitter, W_{ngwa} is extracted from memory (and may eventually undergo some type of processing) and it is inserted in the window field of the TCP segment. With this mechanism the TCP receiver knows the maximum quantity of bytes that the network supports at that instant. When the ACK segment reaches the TCP sender, it is advised of the available network bandwidth.

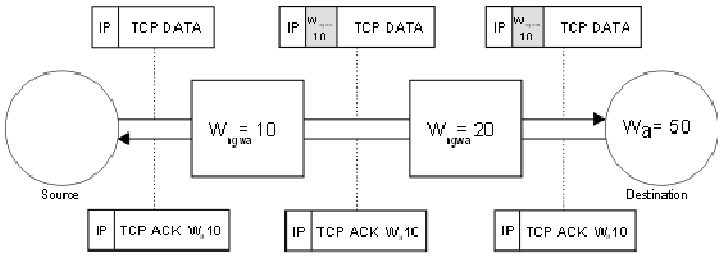

Fig. 1. Simple example of the proposal.

Finally, the above stated mechanism has been divided into eight modes (Table 1) considering the following combination of options: *congestion window* (CW), *smoothing equation* (SE) and *division by number of flows* (FD). Each option can be active or not.

2.1. Congestion Window Option

This feature applies only to TCP sources. When active, a TCP implementation with congestion control is used by the transmitter; in this work we used the TCP New Reno.

Table 1. NGWA modes.

	congestion window	smoothing equation	flow division
1	×	–	–
2	×	×	–
3	–	–	–
4	–	×	–
5	×	–	×
6	×	×	×
7	–	–	×
8	–	×	×

If the feature is not active, naturally standard TCP (RFC 793) [12] will be used. Equation 1 defines the amount of data W_t that TCP is allowed to transmit at any given time.

$$W_t = \begin{cases} \min[W_a, W_c] - W_u, & \text{CW active.} \\ W_a - W_u, & \text{CW inactive.} \end{cases} \quad (1)$$

where W_a is the receiver advertised window (*rcvwnd*) which is set by the receiver in the header of acknowledgment (ACK) segments; W_c is the congestion window (*cnwd*) that is computed by the transmitter following a congestion control algorithm; and W_u is the amount of outstanding data, i.e., the data already sent but not yet acknowledged.

It is important to make clear that the variable W_{ngwa} is processed by the receiver side, and indirectly applied to the transmitter by means of W_a, which is obtained from the receiver. This parameter is generated by the receiver by using Equation 2.

$$W_a = \begin{cases} \min[W_b, W_{\max}, W_{ngwa}], & \text{NGWA on.} \\ \min[W_b, W_{\max}], & \text{NGWA off.} \end{cases} \quad (2)$$

where W_{max} corresponds to the maximum window size, and W_b indicates the amount of space available in the TCP receiver memory.

2.2. Division by Number of Flows Option

By default the NGWA stores in the W_{ngwa} variable the total amount of bytes available in the queue of the intermediate nodes. This behavior can induce injustice because a flow can consume all the resources of the node. The flow division (FD) option realizes the sharing of free bytes in the queue node by the amount of active flows (connections). The detection of TCP flows is done by recording the source/destination of IP addresses and ports. When this option is active, a process is executed to create and keep track of active connections transiting the specific node (variable n_f).

The W_{ngwa} value is then calculated by Equation (3), where B_r is the total space available in the queue of the router. The factor 0.98 is used to provide space in the queue for acknowledgments (ACK).

$$W_{ngwa} = \frac{0.98 \times B_r}{n_f} \tag{3}$$

The flow division feature is ideal when routing is static, if it is dynamic (where packets can travel different paths) the amount W_{ngwa} will not be correctly estimated. This may cause some unfairness, but will not significantly change the performance of the system.

2.3. Smoothing Equation Option

The TCP window generally varies substantially in short time intervals [13]. Aiming to avoid abrupt changes in the transmission window, a smoothing equation (4) is used by TCP receiver, where $0 < \alpha < 1$. When the smoothing equation (SE) option is active, W_{ngwa} in Equation (2) is replaced by $\overline{W}_{ngwa}$.

$$\overline{W}_{ngwa} = (1-\alpha) \times \overline{W}_{ngwa} + \alpha \times W_{ngwa} \tag{4}$$

3. SIMULATION RESULTS

The proposal was implemented in the network simulator NS-3. A natural choice would be the NS-2, but it does not offer a basic TCP implementation to support the method. Simulations were performed to evaluate the performance of the proposal. We performed experiments in order to estimate a set of metrics: *transmission rate*, *network fairness*, *transmission window behavior*, and *packet loss rate*. For the experiments we considered a topology, widely considered in literature [9,14,15,16,17,18], witch is depicted in Figure 2. In the simulations, each node on the left side has established a TCP connection with a node on the right side.

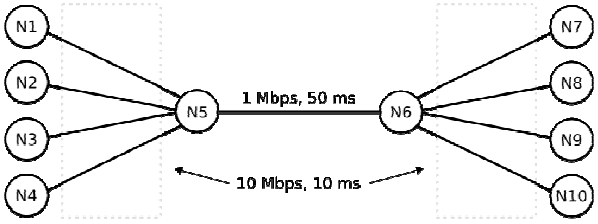

Fig. 2. Network topology for experiments.

As a benchmark to the proposed implementation we used the TCP New Reno and the standard TCP (RFC 793) [12]. The following values were considered: W_{max} = 64 kbps; TCP segment size: 1000 Bytes, queue: *drop tail*; router queue capacity: 97 kBytes; constant $\alpha = 0.3$; simulation time: 10 minutes. Traffic was generated using the model *OnOffApplication*, present in NS-3, witch was configured to continuously transmit data until the end of the simulation at a rate of 500 kbps.

3.1. Transmission Window Behavior

In order to analyze the window behavior, the number of active connections was modified, being a function of the time. At the beginning a connection has been established, followed by a new connection at each 15s interval. In the time interval from 45s to 120s four connections were active. Next, only two connections were active from 121s until the end of the simulation. Hence, it is possible to analyze variations in the transmission window considering the start/end of connections (Figure 3).

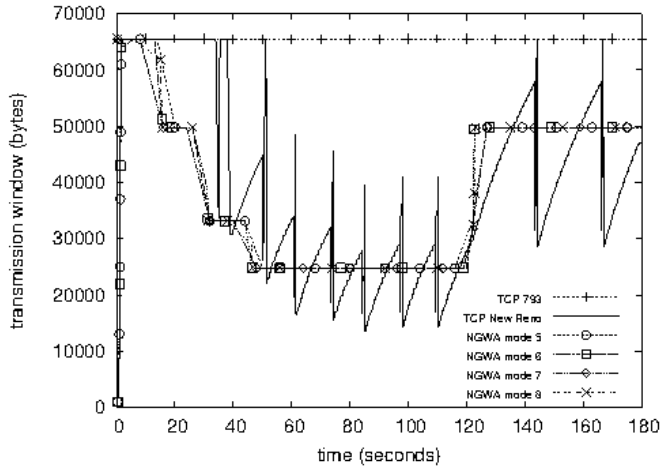

Fig. 3. Transmission Window Behavior. The graph shows the flow that remained active from the beginning to the end of the simulation.

The results of Figure 3 are analyzed in three intervals: (1) 0-45s, (2) 60s-120s, and (3) 121s-180s. In the first interval it can be seen that the transmission window of TCP New Reno started at a low value, increasing constantly to a peak and remaining there until about 30s, then the rate reduces. Modes 5 and 6 of NGWA followed the initial behavior of New Reno, while modes 7 and 8 started with a high value. After that, both modes presented similar behavior. At around 15s the window decreased reaching 50000 bytes, remaining constant until the establishment of a new connection. Then, for each new connection there was a drop in the transmission rate followed by a stability phase. In the second interval New Reno kept its oscillatory behavior, while NGWA persisted with a stable window. In the last interval only two flows remained active. In this case, TCP New Reno presented high variation on its window size, while NGWA rapidly adjusted the window size, keeping it stable until the end of the simulation.

Note that, in general, NGWA achieved a stable transmission rate with good fairness between flows. TCP New Reno presented unfairness and high oscillation.

3.2. Transmission Rate and Network Fairness

In this experiment we compare the transmission rate (in bytes/s) for each flow. With the result of this analysis is possible to verify the network fairness. We considered the standard TCP, TCP New Reno, and all modes of NGWA. Figures 4 and 5 shows the transmission rate for all flows. Each node on the left side established a TCP connection with a node of the right side. We obtained 95% confidence interval by using the *batch means* approach (with 30 batches), a standard technique to evaluate simulation results [19].

Analyzing the results it is clear that the standard TCP was, as expected, the most unfair - the flows 2 and 4 consumed the whole network bandwidth, leaving flows 1 and 3 with a low transmission. TCP New Reno maintained their flows with a good justice. The modes 1 and 3 of NGWA were unfair. On the other hand, the modes 5, 6, 7 and 8 obtained a fair behavior, overcoming the TCP New Reno performance. The NGWA smoothing option presented a more interesting effect in modes 2 and 4, reducing the injustice presented in modes 1 and 3.

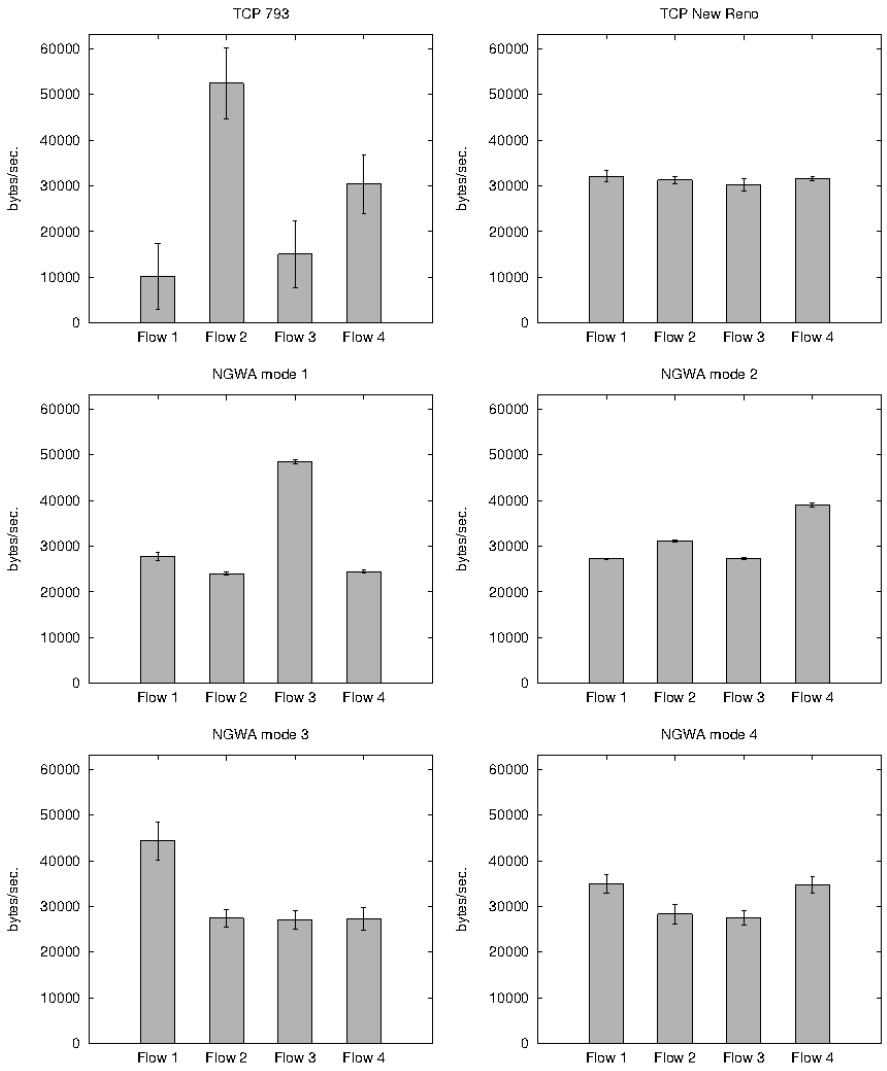

Fig. 4. Total transmission rate for four flows of standard TCP, TCP New Reno, and NGWA modes 1-4.

3.3. Packet Loss Rate

The number of sent, but not received, packets is an important factor when analyzing the quality of the congestion control method, because lost packets, using the network, may increase congestion. Even as *timeouts*, the receiving of *triple dupacks* is considered a sign of network congestion. The total amount of *triple dupacks* obtained along the transmission is shown in Figure 6.

The number of *triple dupacks*, for the standard TCP and New Reno, were the highest in the simulation. NGWA presented different performances depending on the selected mode. The worst case was with the mode 3, reaching 5279 received *triple dupacks*. Being followed by modes 4

(2833), and 1 (30). The smoothing option again showed a good effect for the modes 2 and 4, with superior result when compared with modes 1 and 3. Modes 2/5/6/7/8 do not appear on the graph because received no *triple dupacks*.

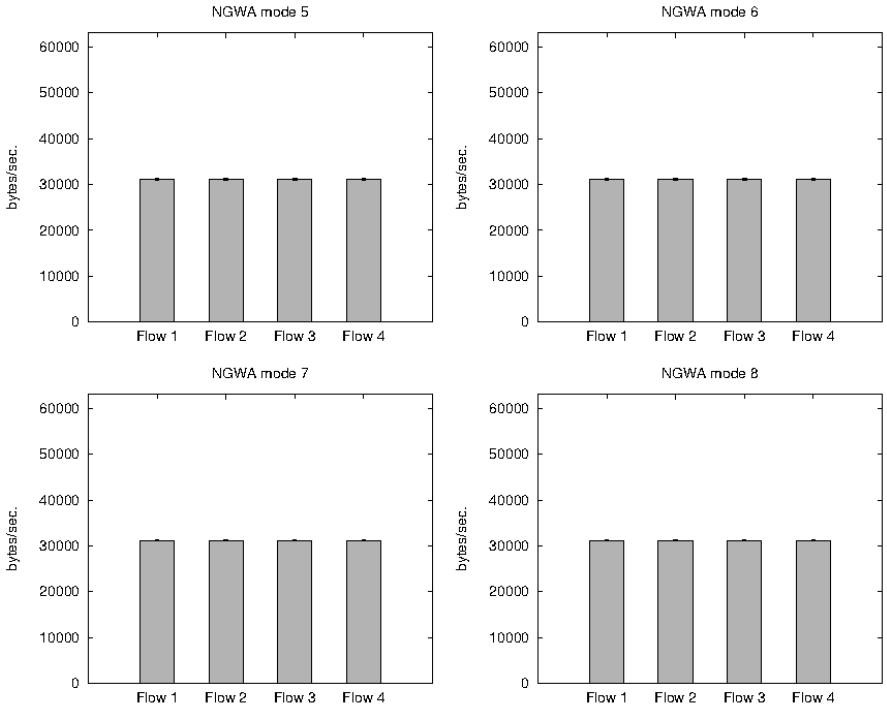

Fig. 5. Total transmission rate for four flows of NGWA modes 5-8.

We also present the average number of lost packets (in 30 rounds) on Table 2. The standard TCP produced the highest average amount of losses (4001), followed by: NGWA mode 3 (1063), mode 4 (869), New Reno (328), and NGWA mode 1 (7). The modes 2/5/6/7/8 do not produced any losses.

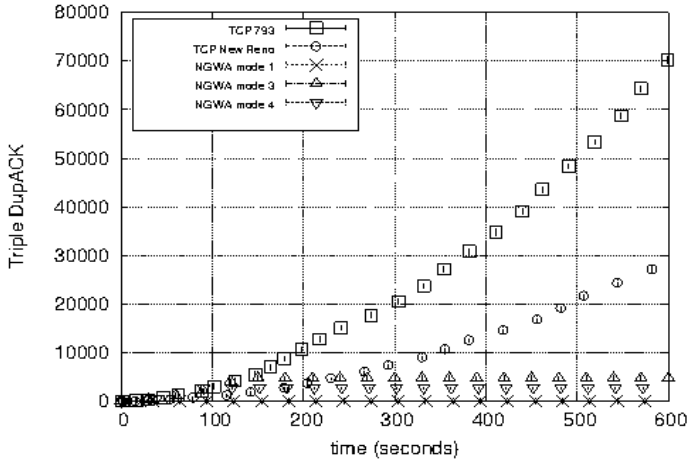

Fig. 6. Total amount of *triple dupacks*.

Table 2. Average number of lost packets (in 30 rounds).

Test	Result
RFC 793	4001
New Reno	328
NGWA mode 1	7
NGWA mode 2	0
NGWA mode 3	1063
NGWA mode 4	869
NGWA mode 5	0
NGWA mode 6	0
NGWA mode 7	0
NGWA mode 8	0

4. PHYSICAL IMPLEMENTATION (NGWA IN LINUX KERNEL)

In order to prove the correct operation of the proposal, a physical implementation was done in Linux 2.6.34. This implementation was naturally divided in two groups: Queue and TCP. The first one was developed to be a module, and must be used in the network routers, and the second one was placed into the TCP stack of the kernel core. These groups are detailed below.

4.1 NGWA Queue

The NGWA Queue runs on the network routers, and do not need to be in the TCP endpoints. Incoming packets are processed by the router, and forwarded to the NGWA Queue. The queue was implemented in the network subsystem, into the file *net/sched/sch_ngwa.c*. The queue management module, in the directory *net/sched*, follow a standard interface for functions and structures and the NGWA is called by the function dequeue. First, it is checked the packet type, which is obtained by reading a field of IP header. If the packet isn't TCP, nothing will be done. Next it is checked the state of the DF option. The amount of available bytes in the router queue is determined, and it is checked the existence of W_{ngwa} in the IP header, if it exists, its amount is stored. When the packet doesn't have NGWA mark or when the amount of bytes in the router queue is less than the W_{ngwa} inserted in the packet, it will be inserted/updated the variable W_{ngwa} in the options field of IP header. Further details of this module are presented below.

Division by Flows: The implementation verifies the type of the TCP segment by checking the control bits of the header, performing a proper operation on a double linked list. If the segment is of type SYN and if the connection has not been previously recorded, it is inserted in the list. If the segment is of type FIN or RST the connection is removed from the list. To avoid duplicate records it was used a set of variables to identify each connection, as follows: TCP source and destination ports, source and destination IP addresses, and the protocol identify.

Options Field: The use of the options field is done based on the *ip_options* structure declared in *include/net/inet_sock.h.* Its main variables are: *optlen*, that register the total size (in bytes) of options; *__pad2*, a generic use pointer (that can be used for new options); and *__data*, a vector to store the options' data. The main functions involved in its use are *ip_options_build* and the *ip_options_compile*. The first one is used to write the options in the IP header, the structure *ip_options* is passed as parameter to store the content of variable *__data* in the IP; and *ip_options_compile* checks the options and stores in an *ip_options* structure [20]. We created a

new type of options to be used by the NGWA; the type must have the size of eight bits and not be in use; we chose the value of 33 (0010 0001). The NGWA option is used only in the kernel level, not having user interface.

4.2 NGWA TCP

The code that implements the NGWA TCP was inserted in the TCP stack of Linux, and was divided into *incoming* and *outgoing* TCP.

Incoming TCP: A received TCP segment is processed by the Incoming TCP code. During a TCP connection, all segments pass through the *tcp_rcv_established* function, found at file *net/ipv4/tcp_input.c.* The main codes for the Incoming TCP were inserted in this function. But, before reaching this level, the received segment is compiled in the network layer to adjust the *ip_options* structure. This is done by the *ip_options_compile* routine. Returning to transport layer, at the arrival of a segment, it is verified if the options field are in the IP header, if it is true, it is checked if the variable *opt->__pad2* is present (this confirms that the segment is of NGWA type), and then the octets are read and stored in a new variable *ngwa_ipopt* of the structure *tcp_sock.*

Outgoing TCP: The Outgoing TCP uses the information of Incoming TCP as parameter for calculating the TCP window. All the code for Outgoing TCP is inserted in the file *net/ipv4/tcp_output.c.* The information of NGWA has been previously saved by the Incoming TCP in the variable *ngwa_ipopt* of the *tcp_sock* structure, and now it is used to obtain the new window value; the new window is determined by the function *tcp_select_window*.

In the NGWA approach, when a packet does not contain the options field in the IP header, this will be added. Naturally, with more bytes in the packet, it can exceed the maximum packet size supported by the network. To prevent this from happening, the Outgoing TCP also performs a reduction of the maximum segment size (MSS) by the size of NGWA option.

4.3 Numerical Results

To assess the performance of the Linux implementation we set up a physical environment with computers and network cables. The connection between the devices on this network was carried out directly, as show in Figure 7. All computers have the same configuration: *Core 2 Duo* processor; 1024 MBytes of memory; and network card *RTL-8169* Gigabit Ethernet.

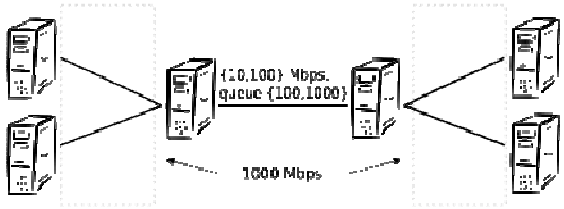

Figure 7. Network topology.

The tests were performed considering four different network configurations. The central link capacity is 1 Gbps, but we utilized capacities of 10 and 100 Mpbs. To perform this setting we used the *ethtool* software in Linux. The queue was dimensioned to 100 and 1000 packets, according to the test. Using these network settings, the tests were conducted with NGWA, TCP New Reno, and TCP New Reno with RED queue [21]. The following configurations were considered in NGWA: basic, SE, FD, and SE/FD options (modes 1/2/5/6, respectively). The

traffic was generated using the *iperf* software; it has been used in similar studies [22,23]. The *iperf* was configured to use TCP protocol with 64 KBytes of window, transmitting data constantly in one direction (client to server). The experiments were performed with one flow, and with two simultaneous flows. The data was analyzed using the *batch means* method, with 30 batches, to obtain 95% confidence intervals.

One Flow: In this experiment we established a connection between a computer on the left side with one on the right side of the topology, through the software *iperf*. Each experiment ran for two minutes in steady state. In the tests with TCP New Reno we used a FIFO queue, except when RED was explicitly configured. RED parameters were adjusted according to the procedure found in [24]. We considered the packets average size equal to 1700 bytes; then, for a 100 packets size queue, we have the following RED parameters: *buffer limit* = 170 kBytes, *min_th* = 20 kBytes, *max_th* = 84 kBytes. In the test with 1000 packets, the previous values were multiplied by ten. In both cases the *max_p* = 0.02.

Results are shown in Figure 8, which displays the transmission rate (in bytes/s). It can be seen that, in the four combinations, NGWA performance overcomes the others methods. The performance of New Reno and RED improved significantly with the increase of the queue size, but the results of NGWA remained stable. The NGWA with smoothing equation obtained better results than basic NGWA.

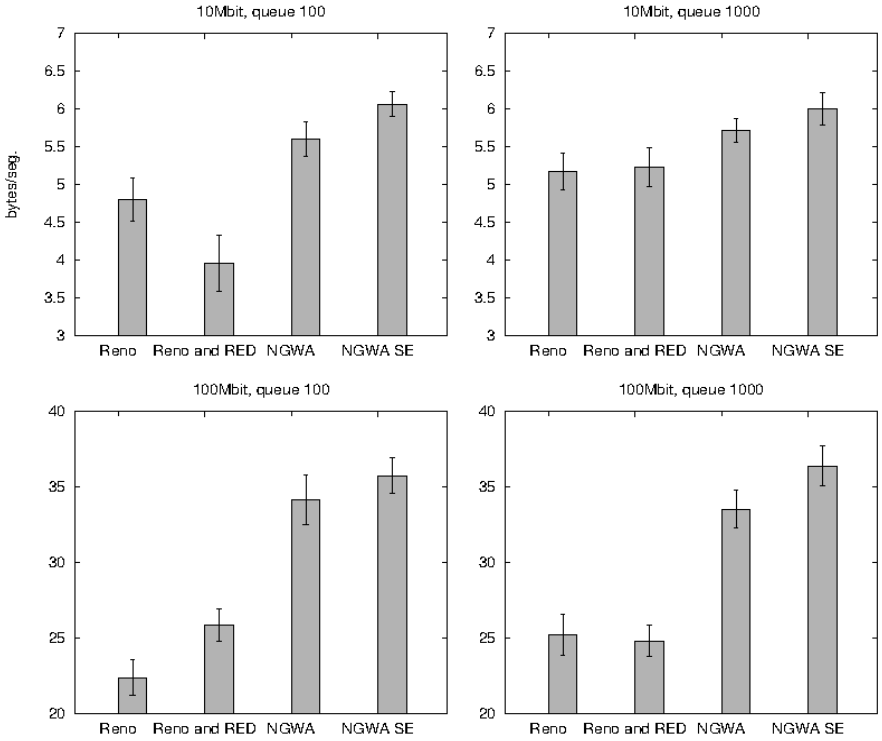

Figure 8. Total transmission rate of one flow in Linux.

Two Flows: In this experiment two new computers were added to the network; this way two computers on the left side transmit to those the right side. The *iperf* ran on the four computers and two sets of data were collected. One set for the first pair (flow 1) and another one for the other two computers (flow 2). The settings of the simulation are the same as in the previous section,

however the main link was set to 100 Mbps. In this second test it is also checked the network fairness.

Figure 9 present the results. The division by flow worked properly (with 100 packets queue), and good results were obtained when using the smoothing NGWA option. In the case of 1000 packets queue, the NGWA fairness decreased, but remained higher than the other methods. The transmission rate of the NGWA, in general, was higher than New Reno and New Reno/RED. Analyzing the performance of TCP New Reno and New Reno/RED, or they had a good transmission rate, or reasonable fairness, but not both.

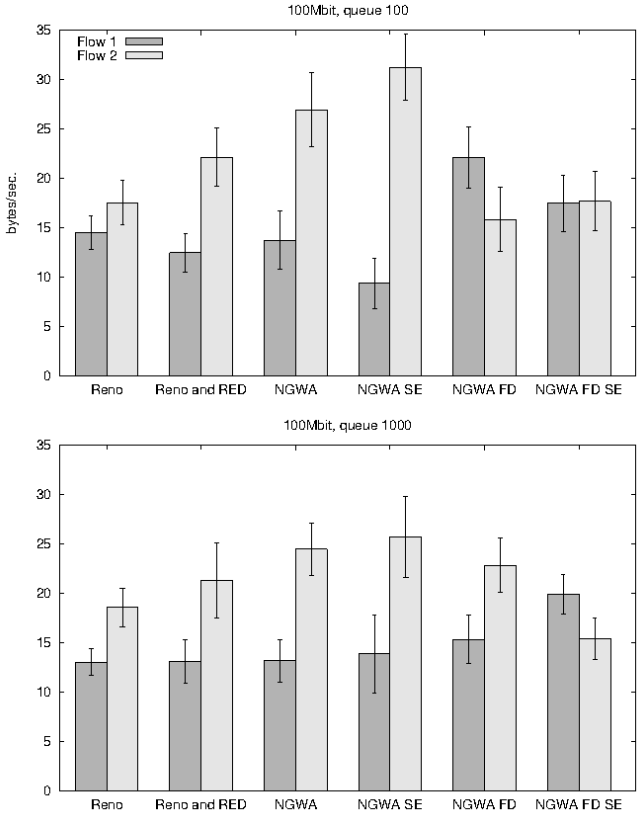

Figure. 9. Total transmission rate of two flows in Linux.

5. DISCUSSION

In this paper we have presented a new scheme, called NGWA, for congestion control of TCP protocols. The NGWA provides information considering the available bandwidth of the network infrastructure to the endpoints of the TCP connection. Hence, it helps in choosing a better transmission rate for TCP improving its performance. The NGWA was subdivided into eight modes, each with a different set of features. It was tested in the network simulator NS-3 and also implemented in the Linux kernel. We performed experiments in order to analyze a set of metrics: *transmission rate*, *network fairness*, *transmission window behavior*, and *packet loss rate.*

Simulation results show superior performance of the proposed scheme when compared to those obtained by TCP New Reno and standard TCP. Compared to TCP New Reno and standard TCP, NGWA achieved best performance in all tests, especially the loss rate, where the modes 5/6/7/8 did not record the receipt of *triple dupacks* and losses. The smoothing option improved the

performance of the modes 1 and 3. The division by flow provided results greatly superior to the others methods. The modes 7 and 8 require, by construction, no congestion control mechanisms.

Even in environments of different nature (NS-3 and Linux), it is acceptable to compare the results in each environment, in order to evaluate the NGWA performance in general. Hence, analyzing the results obtained with NS-3 and in the Linux kernel, it was found that the results were consistent. In the basic NGWA (without SE or FD options) the volume of data transmitted was high, but there was unfairness in the network; the FD option has brought greater justice to the flows, and the smoothing option improved the results.

The studies indicated that the proposed scheme is promising. However, the simulations do not allow us to evaluate all the potential and/or possible shortcomings of NGWA. Hence, improvements in the proposal are possible and some issues deserve further investigation, including: (a) to study the effectiveness of the method when dealing with the accounting of active connections that make use of routers. In a dynamic network, it can occur rerouting processes and/or loss of flows due to failures. In this case, it is suggested that the necessary statistics occur by "time windows" to ensure a better representation of the state of the connections; (b) to perform an analysis of the system performance when the NGWA competes with different schemes for congestion control, such as TCP New Reno, TCP CUBIC, and other types of traffic, such as UDP; (c) to test more complex networks, with a larger number of nodes and links, so that there are different round trip times (RTTs) in routes. It is known that when there are flows that compete for limited resources, connections with higher RTTs will be most penalized, reducing the network fairness. In general, in this case, the source reacts with delay. Hence, one need to check what would be the impact on NGWA performance due to ACK delays; and (d) to use the Linux environment to perform more complex or impossible (simulation) tests, for example, the use of transmission rate of 1 Gbps and beyond.

REFERENCES

[1] J. Kurose and K. Ross, "Computer Networking: A Top-Down Approach." *Addison Wesley Higher Education*, 2010.

[2] S. Jiang, Q. Zuo, and G. Wei, "Decoupling congestion control from TCP for multi-hop wireless networks: semi-TCP", in *CHANTS '09: Proceedings of the 4th ACM workshop on Challenged networks*, (New York, NY, USA), pp. 27-34, ACM, 2009.

[3] S. Vangala and M. Labrador, "The TCP SACK-aware snoop protocol for TCP over wireless networks" , in *Vehicular Technology Conference*, 2003. VTC 2003-Fall. 2003 IEEE 58th, vol. 4, (Orlando, Florida, USA), pp. 2624-2628, IEEE Press, oct. 2003.

[4] E. Weigle and W. Feng, "Dynamic right-sizing: a simulation study", in *Computer Communications and Networks*, 2001. Proceedings. Tenth International Conference on, (St. Thomas, Virgin Islands, USA), pp. 152-158, IEEE Press, 2001.

[5] O. Gnana Prakasi and P. Varalakshmi, "Enhancing performance of TCP in multihop networks", *International Journal of Computer Networks & Communications* (IJCNC), vol. 4, sep. 2012.

[6] S. Utsumi and S. M. S. Zabir, "Utilization-based congestion control", *International Journal of Computer Networks & Communications* (IJCNC), vol. 4, sep. 2012.

[7] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks", *Wireless Networks*, vol. 1, no. 4, pp. 469-481, 1995.

[8] K.-F. Leung and K. Yeung, "G-Snoop: enhancing TCP performance over wireless networks", in *Computers and Communications*, 2004. Proceedings. ISCC 2004. Ninth International Symposium on, vol. 1, (Alexandria, Egypt), pp. 545-550, IEEE Press, jun. 2004.

[9] G. Hasegawa and M. Murata, "TCP symbiosis: congestion control mechanisms of TCP based on Lotka- Volterra competition model", in Interperf '06: *Proceedings from the 2006 workshop on Interdisciplinary systems approach in performance evaluation and design of computer & communications sytems*, (New York, NY, USA), p. 11, ACM, 2006.

[10] M. Gerla, R. L. Cigno, and W. Weng, "Generalized Window Advertising for TCP Congestion Control", *European Transactions on Telecommunications*, vol. 13, pp. 549-562, dec. 2002.
[11] NS-3, "The Network Simulator", nov. 2011.
[12] J. B. Postel, "Transmission Control Protocol", 1981. RFC 793.
[13] J.-C. Moon and B. G. Lee, "Rate-adaptive snoop: a TCP enhancement scheme over rate-controlled lossy links", *IEEE/ACM Transactions on Networking* (TON), vol. 14, no. 3, pp. 603-615, 2006.
[14] S. Floyd, "TCP and explicit congestion notification", *SIGCOMM Computer Communication Review*, vol. 24, no. 5, pp. 8-23, 1994.
[15] M. Gerla, W. Weng, and R. Cigno, "Bandwidth feedback control of TCP and real time sources in the Internet", in *Global Telecommunications Conference*, 2000. GLOBECOM '00. IEEE, vol. 1, (San Francisco, United States), pp. 561-565, IEEE Press, nov. 2000.
[16] S. H. Low, L. Peterson, and L. Wang, "Understanding TCP vegas: a duality model", in *Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '01, (New York, NY, USA), pp. 226-235, ACM, 2001.
[17] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control", *IEEE/ACM Transactions on Networking* (TON), vol. 8, no. 5, pp. 556-567, 2000.
[18] Y. Yang and S. Lam, "General AIMD congestion control", in *Network Protocols*, 2000. Proceedings. 2000 International Conference on, (Osaka, Japan), pp. 187-198, IEEE Press, 2000.
[19] C. Chien, "Batch size selection for the batch means method", in *Proceedings of the 26th conference on Winter simulation*, WSC '94, (San Diego, CA, USA), pp. 345-352, Society for Computer Simulation International, 1994.
[20] K. Wehrle, F. Pählke, H. Ritter, D. Müller, and M. Bechler, *The Linux Networking Architecture: Design and Implementation of Network Protocols in the Linux Kernel*. Gravenstein Highway North, Sebastopol, CA: Prentice Hall, 2004.
[21] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance", *IEEE/ACM Transactions on Networking* (TON), vol. 1, no. 4, pp. 397-413, 1993.
[22] Y.-T. Li, D. Leith, and R. Shorten, "Experimental Evaluation of TCP Protocols for High-Speed Networks", *IEEE/ACM Transactions on Networking*, vol. 15, pp. 1109-1122, oct. 2007.
[23] D. X. Wei and P. Cao, "NS-2 TCP-Linux: an NS-2 TCP implementation with congestion control algorithms from Linux", in *Proceeding from the 2006 workshop on NS-2: the IP network simulator*, WNS2 '06, (New York, NY, USA), ACM, 2006.
[24] E. C. G. Wille, M. Mellia, E. Leonardi, and M. Ajmone-Marsan, "Design and Analysis of IP Networks with End-to-End QoS Guarantees", in *XXI Brazilian Symposium on Telecommunications*, (Belém- Pará, Brazil), Brazilian Society of Telecommunications, sep. 2004.

Authors

Marcos Talau was born in Dois Vizinhos/PR - Brazil. He received his degree in Information Systems in 2005 from UNISEP (Dois Vizinhos/PR). He received his M.Sc. in Electrical Engineering in May 2012 from Federal University of Technology of Paraná - UTFPR (Curitiba/PR - Brazil). Prof. Talau is with the UTFPR since January 2007. His teaching duties at UTFPR comprise undergraduate course on Computer Science. His research interests are centered upon the area of computer networks and congestion control protocols.

Emilio C. G. Wille was born in Lapa/PR - Brazil. He received his degree in Electronic Engineering in February 1989, and a M.Sc. in Electronic and Telecommunications Engineering in July 1991, both from Federal University of Technology of Paraná - UTFPR (Curitiba/PR - Brazil). He received his Ph.D. degree in Telecommunications Engineering in February 2004 from Politecnico di Torino (Italy). During his stay in Italy he was supported by a CAPES Foundation scholarship from the Ministry of Education of Brazil. Prof. Wille is an Associate Professor at the UTFPR, and since October 1991 he is with the Electronics Department. His teaching duties at UTFPR comprise graduate and undergraduate-level courses on electronic and telecommunication theory. He has co-authored several papers presented in national and international conferences, all of them in the area of telecommunication systems and networks. His research interests are centered upon the application of optimization algorithms for telecommunication networks design and planning, Markov processes, queueing models, and performance analysis of telecommunication systems.