MEASUREMENTS OF TCPW ABSE FAIRNESS AND FRIENDLINESS

Ilhem Lengliz, Haifa Touati, Fehmi Sanàa, Farouk Kamoun
RAMSIS/CRISTAL Laboratory, National School of Computer Science
Complexe Universitaire La Manouba, La Manouba 2010, Tunisia

Medy Yahia Sanadidi
UCLA, Computer Science Department
4732 Boelter Hall, Los Angeles, CA 90095-1596

Keywords: TCP Westwood ABSE, Internet, congestion control, experimentation.

Abstract: TCP Westwood (TCPW), is a TCP protocol with a sender-side modification of the window congestion control scheme. This protocol is intended to act in packet lossy environments. It relies on a continuous estimation, by the traffic source, of the connection packet rate based on the ACK reception rate. And this in order to compute the congestion window and the slow start threshold settings after a congestion episode. Given that it has been yet established through experimental and simulation studies that TCPW exhibits significant improvements in throughput performance over Reno in various environments, we are focusing in this paper on TCPW performance measurements with respect to throughput, fairness and friendliness towards TCP New Reno in a wired LAN and in the Internet. Which constitutes a proceeding of a set of measurements achieved on TCPW in similar environment. In this paper we present the results of some experimentations carried out in the CRSITAL Laboratory with a FreeBSD TCPW ABSE protocol implementation.

1 INTRODUCTION

TCP Westwood has been initially designed in (Casetti, 2000). This new protocol relies on a simple modification of the TCP source protocol behavior for a faster recovery. This latter mechanism consists in choosing both a slow start threshold and a congestion window that result from the effective connection while congestion is experienced. Hence, TCPW attempts to make a more "informed" decision, in contrast with TCP Reno, which automatically halves the congestion window after three duplicate ACKs.

Like TCP Reno, TCPW cannot distinguish between buffer overflow loss and random loss. However, in presence of random loss, TCP Reno overreacts and reduces the window by half. Whereas, TCPW after packet loss and retransmission timeout, resumes with the previous window as long as the bottleneck is not yet saturated (i.e., there is no buffer overflow).

To prevent the unnecessary window reduction of TCP Reno in case of random packet loss, and more precisely the loss caused when there is wireless links, several schemes have been proposed (Balakrishnan, 1997). All of these schemes require the cooperation of intermediate routers/proxies. However, TCPW does not require any specific intervention/support from intermediate routers, thus preserving the original "end to end design" principle (Gerla, 2001).

One of the TCPW refinements is proposed in (Wang, 2002), and named TCPW RE (Rate Estimation) to address TCP Reno Friendliness. The other is TCPW+, described and studied in (Ferorelli, 2002), it is intended to improve TCPW performance in the case of Internet transmissions. The TCPW ABSE protocol we used in our work to carry out the experimentations has been proposed in (Casetti, 2002). It palliates TCP efficiency degradation in packet loss environment.

Lengliz I., Touati H., Sanàa F., Kamoun F. and Yahia Sanadidi M. (2004).
MEASUREMENTS OF TCPW ABSE FAIRNESS AND FRIENDLINESS.
In *Proceedings of the First International Conference on E-Business and Telecommunication Networks*, pages 175-182
DOI: 10.5220/0001398801750182

In this paper we present the results of preliminary experimental measurements on TCPW performances when used in a LAN or in the Internet.
The remaining of this paper is organized as follows. In Section 2, we describe the TCPW ABSE protocol. In Section 3, we present the measurement results when TCPW is applied in a wired LAN. Section 4 gives a synthesis of TCPW performance measurements in the Internet. Finally section 5 gives the conclusion of the paper and draws some perspectives for this work.

2 TCPW ABSE PROTOCOL

TCPW ABSE (Adaptive Bandwidth Share Estimation), initially proposed in (Wang, 2002), is a sender-only modification of TCP NewReno. The TCP sender Adaptively determines a Bandwidth Share Estimate. This estimate is based on information carried in the ACKs, and on the rate at which the ACKs are received. After a packet loss indication, which could be due to either congestion or link errors, the sender uses the estimated bandwidth to properly set the congestion window and the slow start threshold. Further details regarding bandwidth estimation are provided in the following sections. For now, let us assume that a sender has determined the connection bandwidth estimate as mentioned above, and let us describe how the estimate is used to properly set *cwin* and *ssthresh* after a packet loss indication.
In TCPW, congestion window dynamics during slow start and congestion avoidance are unchanged; that is, they increase exponentially and linearly, respectively, as in current TCP NewReno.
A packet loss is indicated by :
(a)the reception of 3 DUPACKs,
or
(b) a coarse timeout expiration.
In case (a), TCPW sets *cwin* and *ssthresh* as follows :

```
if (3 DUPACKs are received)
ssthresh = (ABSE*RTTmin)/seg_size;
if (cwin > ssthresh) /*congestion avoid.*/
cwin = ssthresh;
endif
endif
```

In case a packet loss is indicated by a timeout expiration, *cwin* and *ssthresh* are set as follows:

```
if (coarse timeout expires)
cwin = 1;
ssthresh = (ABSE * RTTmin)/seg_size;
if (ssthresh < 2)
ssthresh = 2;
endif
```

The rationale of the algorithm above is that after a timeout, *cwin* and the *ssthresh* are set equal to 1 and ABSE, respectively. Thus, the basic Reno behavior is still captured, while a reasonably speedy recovery is ensured by setting *ssthresh* to the value of the ABSE.

2.1 Adaptive Bandwidth Share Estimation

The ABSE algorithm adapts to the congestion level in performing its bandwidth sampling, and employs a filter that adapts to the round trip time and to the rate of network conditions change. The bandwidth share estimation is computed using a time varying coefficient, exponentially-weighted moving average (EWMA) filter, which has both adaptive gain and adaptive sampling. Let t_k be the time instant at which the k_{th} ACK is received at the sender. Let s_k be the bandwidth share sample, and $\hat{s}_k$ the filtered estimate of the bandwidth share at time t_k.. Let α_k be the time-varying coefficient at t_k. The ABSE filter is then given by :

$$\hat{s}_k = \alpha_k \hat{s}_{k-1} + (1-\alpha_k)\left(\frac{s_k + s_{k-1}}{2}\right) \quad (1)$$

where

$$\alpha_k = \frac{2\tau_k - \Delta t_k}{2\tau_k + \Delta t_k}$$

and τ_k a filter parameter which determines the filter gain, and varies over time adapting to path conditions. In the filter formula above, the bandwidth sample at time *k* is :

$$s_k = \frac{\sum_{t_j > t_k - T_k} d_j}{T_k} \quad (2)$$

where d_j is the number of bytes that have been reported delivered by the j_{th} ACK, and T_k is an interval over which the bandwidth sample is calculated.

2.1.1 ABSE adaptive sampling

ABSE provides an adaptive sampling scheme, in which the time interval T_k associated with the k_{th} received ACK is appropriately chosen, depending on the network congestion level. To determine the network congestion level, a simple throughput filter is proposed to estimate the recent throughput achieved. By comparing this estimate with the instantaneous sending rate obtained from *cwin*, the path congestion level is determined.
When the k_{th} ACK arrives, a sample of throughput during the previous RTT is calculated as :

$$\hat{Th}_k = \frac{\sum_{t_j > t_k - RTT} d_j}{RTT}$$

where d_j is the amount of data reported by ACK j. In (Wang, 2002), the value (ε = 0.6) has been employed for the constant-gain filter to calculate the recent throughput as :

$$\hat{Th}_k = \varepsilon \hat{Th}_{k-1} + (1-\varepsilon) Th_k \quad (3)$$

When $\hat{Th}_k * RTT$ is larger than the current *cwin* value, indicating a path without congestion, T_k is set to T_{min}. Otherwise, T_k is set to :

$$T_k = RTT * \frac{cwin - (\hat{Th}_k * RTT_{\min})}{cwin} \quad (4)$$

2.1.2 Filter Gain Adaptation

When τ_k is larger, α_k will be larger and the filter tends to be more stable and less agile. After a certain point, α_k basically stays unchanged as the value of τ_k increases. The parameter τ_k adapts to network conditions to dampen estimates when the network exhibits very unstable behavior, and reacts quickly to persistent changes. A stability detection filter can be used to dynamically change the value of τ_k. The network instability U is measured with a time-constant EWMA filter (Wang, 2002)(Casetti, 2002) :

$$U_k = \beta U_{k-1} + (1-\beta)|s_k - s_{k-1}| \quad (5)$$

In (Ferorelli, 2002), s_k is the k_{th} sample, and β is the gain of this filter, which is set to be 0.6 in (Wang, 2002). When the network exhibits high instability, the consecutive observations diverge from each other, as a result, U_k increases. Under this condition, increasing the value of τ_k makes the ABSE filter defined by eq. (1) more stable. When a TCP connection is operating normally, the interval between the consecutive acknowledgements are likely to vary between the smallest the bottleneck capacity allows, and one RTT. Therefore, τ_k should be larger than one RTT, thus $\tau_{min} = RTT$. In (Wang, 2002) τ_k is set to be :

$$\tau_k = RTT + N * RTT \frac{U_k}{U_{\max}} \quad (6)$$

The value of RTT can be obtained from the smoothed RTT estimated in TCP (Wang, 2002).

2.2 Accounting for delayed and cumulative ACKs in bandwidth measurement

This issue has been addressed in (Wang, 2002). DUPACKs should count towards the bandwidth estimation since their arrival indicates a successfully received segment, albeit in the wrong order. As a consequence, a cumulative ACK should only count as one segment's worth of data since duplicate ACKs ought to have been already taken into account. Further complications result from *delayed ACKs*. The standard TCP implementation provides for an ACK being sent back once every other in-sequence segment received, or if a 200 ms timeout expires after the reception of a single segment (Casetti, 2002). The combination of delayed and cumulative ACKs can potentially disrupt the bandwidth estimation process. Therefore, in (Casetti, 2002) two important aspects of the bandwidth estimation process are stressed :

(a) the source must keep track of the DUPACKs number it has received before new data is acknowledged;

(b) the source should be able to detect delayed ACKs and act accordingly.

The approach chosen to take care of these two issues can be found in the `AckedCount` procedure, detailed below, showing the set of actions to be undertaken upon the reception of an ACK, for a correct determination of the number of packets that should be accounted for by the bandwidth estimation procedure, indicated by the variable `acked` in the pseudocode. The key variable is `accounted`, which keeps track of the received DUPACKs.

When an ACK is received, the number of segments it acknowledges is first determined (`cumul_ack`). If it is equal to 0, then the received ACK is clearly a DUPACK and counts as 1 segment towards the BWE; the DUPACK count is also updated. If `cumul_ack` is larger than 1, the received ACK is either a delayed ACK or a cumulative ACK following a retransmission event; in that case, the number of ACKed segments is to be checked against the number of segments already accounted for (`accounted_for`). If the received ACK acknowledges fewer or the same number of segments than expected, it means that the "missing" segments were already accounted for when DUPACKs were received, and they should not be counted twice. If the received ACK acknowledges more segments than expected, it means that although part of them were already accounted for by way of DUPACKs, the rest are cumulatively acknowledged by the currentACK; therefore, the currentACK should only count as the cumulatively acknowledged segments. It should be noted that the last condition correctly estimates the delayed ACKs.

```
cumul_ack = 2 and accounted_for = 0):
PROCEDURE AckedCount
cumul_ack=current_ack_seqno-
last_ack_seqno;
if (cumul_ack = 0)
last_ack_seqno = current_ack_seqno;
acked = cumul_ack;
return(acked);
```

```
END PROCEDURE
accounted_for = accounted_for + 1;
cumul_ack = 1;
endif
if (cumul_ack > 1)
if (accounted_for >= cumul_ack)
accounted_for=accounted_for- cumul_ack;
cumul_ack = 1;
else if (accounted_for < cumul_ack)
cumul_ack = cumul_ack - accounted_for;
accounted_for = 0;
endif
endif
```

In the next two sections we are presenting a performance evaluation on TCPW in a wired environment, by mean of experimental measurements. The first section deals with measurements on an Ethernet LAN at CRISTAL Laboratory in the ENSI (Ecole Nationale des Sciences de l'Informatique). The second section is concerned with an outdoor experimentations. On one hand between the ENSI and the CCK (Centre de Calcul ElKhawarizmi) and on the other hand on a connection between the ENSI and UCLA (UCLA Computer Science Department). We focus usually on the fairness and friendliness of TCPW ABSE in comparison to TCP New Reno, since the latter is shown to be fair.

3 TCPW ABSE in a lan

3.1 The LAN environment

As illustrated by the figure 1, this configuration consists of two traffic sources, the first is a TCPW ABSE one, whereas the second is a TCP New Reno one.

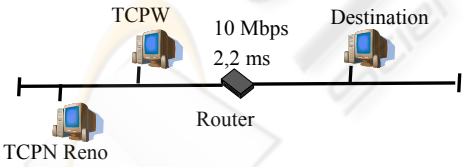

Figure 1: The testbed layout

They share a connection to a same destination, passing through a router. Hence this connection is the bottleneck. The RTT is of 2.2 ms and the data is generated via ftp.

3.2 TCPW Rate

From figure 2, it appears readily that TCPW and TCP New Reno connections share the available bandwidth in a similar manner. In fact, the measured values for each of these protocols rate have the same behavior regarding the number of parallel ftp sessions.

We consider in particular the point 1, where the measurement is made for only one connection. If we compare the TCW rate to that of TCP New Reno, we can observe that New Reno gives a best rate than Westwood. Hence, TCPW doesn't offer, in this case, any gain in the rate compared to TCP New Reno. This is due to the fact that the LAN doesn't induce any random losses because of congestion states.

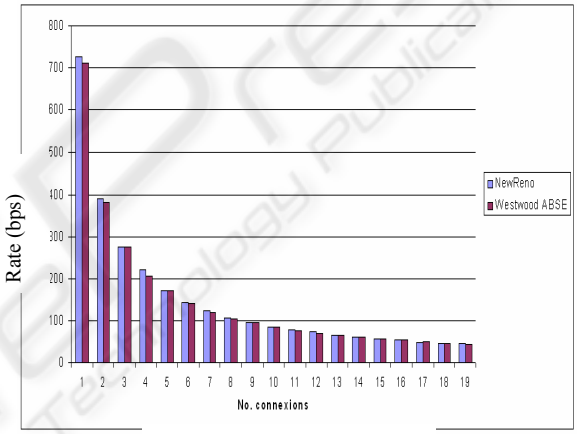

Figure 2: Rate per connection versus number of connection

Besides, the propagation delay is very small, about 2.2 ms. In such a situation, it is preferable not to use TCPW, because it isn't a massive packet lossy environment and the congestion control mechanism isn't invoked usually. This is very insufficient to get a considerable gain in the rate.

3.3 TCPW Fairness

TCP New Reno insures fairness in bandwidth share by nature. In fact, the congestion mechanism accelerates the convergence to the fair share, using the dividing of the congestion window. TCPW ABSE offers a fairness index comparable to those obtained with TCP New Reno. This index is obtained using the following equation :

$$fairness\quad index = \frac{\left(\sum_{i=1}^{n} b_i\right)^2}{N\sum_{i=1}^{n} b_i^{\,2}}$$

b_i : rate of the flow i
n : the total number of active connections
The figure 3 exhibits the variation of this index versus the number of competing connections and this using the test bed configuration of figure 1. We can note that the results performed by TCPW are close to those of TCP New Reno. This consolidates the results yet established in (Wang, 2002). In the first test, only one source is active at once, TCPW ABSE or TCPW. But, a source is generating multiple ftp sessions toward the destination. The number of competing connections ranges from 2 to 19. Each measurement is performed twenty once. Note that each of these TCP protocols has been tested separately. For example, the measure corresponding to the label 10 on the x axis stands for the mean rate of a TCPW or a TCP new Reno connection in the case of 10 ftp sessions transmitting simultaneously.

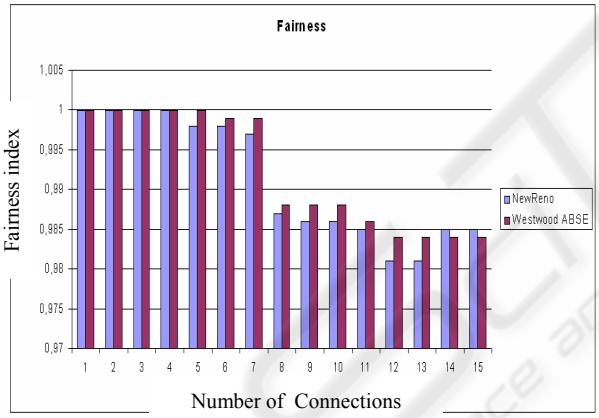

Figure 3: Fairness index versus number of competing connections

3.4 TCPW ABSE friendliness towards TCP New Reno

The figure 4 gives the mean rate achieved by each of the protocols when the number of connections changes (in this case we have 5 TCP New Reno connections against 5 TCPW ABSE connections). Besides, the friendliness towards TCPNReno has been a goal to reach since the BE version of TCPW.

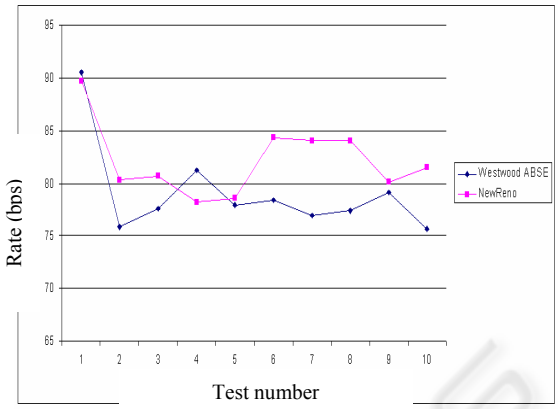

Figure 4: TCPW ABSE/ New Reno rates

Using the same configuration of figure 1, both of the sources are initiated simultaneously. The total number of connections is 10, but for each test we vary the proportion of TCP connections of each protocol (i.e. n TCPNReno vs (10-n) TCPW, n=1..9) in order to study the effect of one protocol on the other.

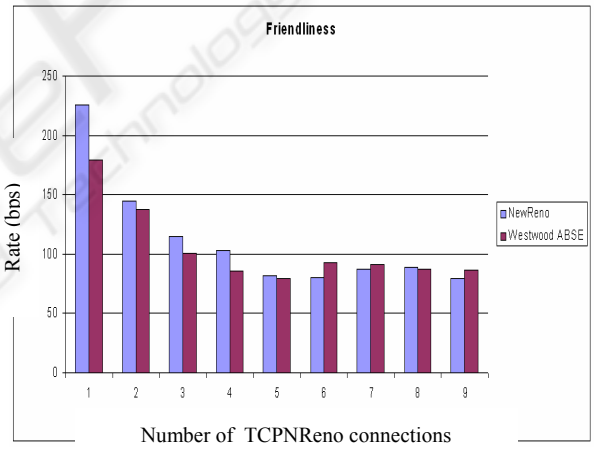

Figure 5: New Reno vs. TCPW ABSE

The results plotted on figure 5 show that TCPW exhibits a friendly behavior towards TCP New Reno. Thus, the new estimation technique used in TCPW improves the friendliness of this protocol towards TCP New Reno. From an other point of view, we can observe the effect of these protocols each on other. This figure shows the rate reached by each of the protocols versus the number of the other competing flows. As an illustration, the measure 1 shows that the connection TCP New Reno reaches 225.53 Ko/s and this with 9 competing TCPW connections, whereas a TCPW ABSE connection reaches only 179.15 Ko/s in the presence of 9 TCP New Reno connections. All of these results

consolidates once again the conclusions yet established by simulations in (Wang, 2002).

4 TCPW ABSE ON THE INTERNET

4.1 ENSI-CCK Connection

4.1.1 Test scenario

We are comparing here the rate performed by TCPW to that of TCP New Reno over an Internet connection between the ENSI and the CCK. The test platform is shown in figure 6. It consists of two stations located in CRISTAL Laboratory at the ENSI and a third station situated in the CCK. TCPW ABSE is installed on the first station : cristal1.ensi.rnu.tn, whereas the second : cristal2.ensi.rnu.tn, supports TCP New Reno. The test scenario consists in starting two ftp sessions from one of the two sources at the ENSI to the destination at the CCK. The files transferred are of 10 Mo. In order to collect the results, we used tcpdump (www.tcpdump.org), then we analyzed them by tcptrace (www.tcptrace.org). We repeat these measurements over two days.

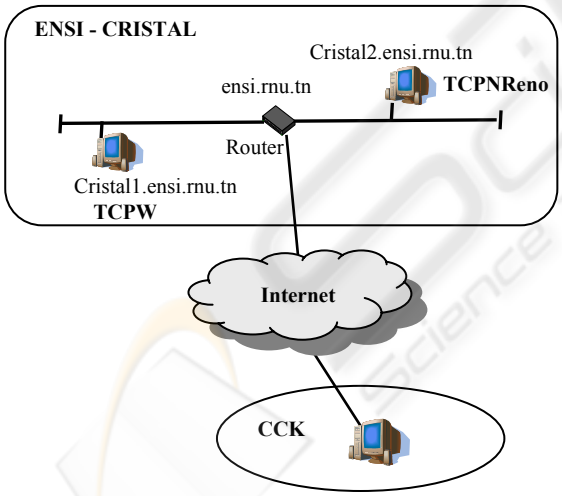

Figure 6: Test platform ENSI-CCK

4.1.2 Results

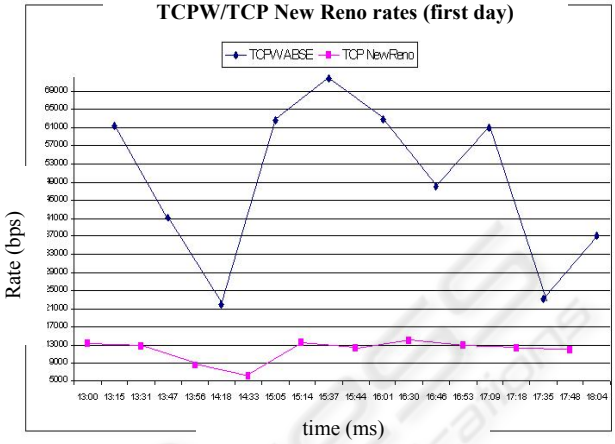

Figure 7: TCPW ABSE/TCP New Reno rates ENSI-CCK (first day)

The measurements are plotted on figures 7 and 8 for the first and the second day respectively. Once again, these results agree with those established for TCPW in (Wang, 2002). As one can see, TCPW ABSE achieves a better rate than TCP New Reno. In fact, during the first day of experimentation, the mean TCPW rate is 490.87 kbps whereas that of TCP NewReno is 119.88 kbps. The second set of measurements shows a TCPW mean rate of 319.04 kbps, while TCP New Reno achieves only a 61.821 kbps rate.

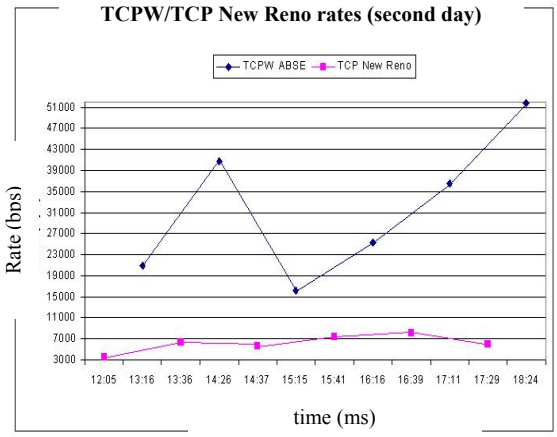

Figure 8: TCPW ABSE/TCP New Reno rates ENSI-CCK (second day)

4.2 ENSI-UCLA Connection

4.2.1 Test scenario

The second experimentation on the TCPW performance in the Internet has been carried on two connections between the CRISTAL Laboratory and the UCLA Computer Science Department. The test platform is presented in figure 9.
In this configuration, the station at the CRISTAL Laboratory : Cristal1.ensi.rnu.tn, stands for the destination, while the two stations in UCLA are the sources. Indeed, the first station : Orion.cs.ucla.edu, implements TCPW ABSE and the second : Mvalla.cs.ucla.edu keeps its default TCP package i.e. TCP New Reno.
We used iperf data transfer to measure and compare the rates of both TCPW and TCP New Reno on this connection. The file is of 4 Mo.
We execute so tcpdump to collect the measures, and tcptrace for the analysis. The connections TCPW and TCP New Reno are activated alternatively. We repeat the test thirty once for each TCP protocol in different periods of the day.

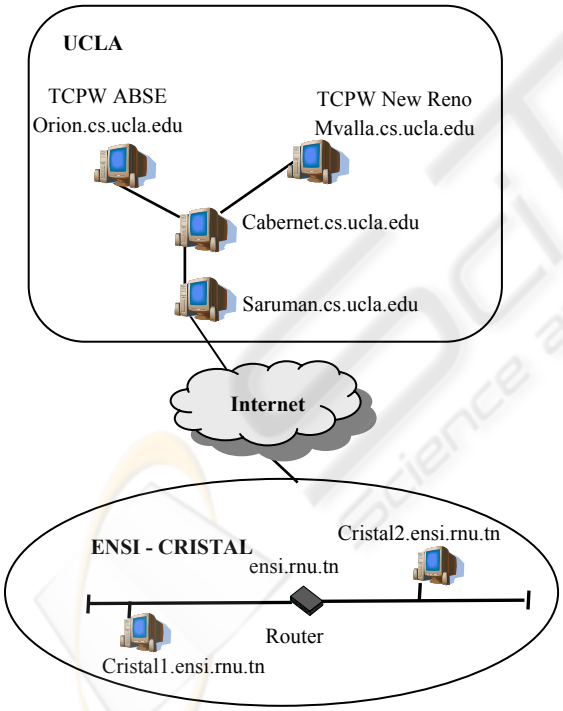

Figure 9: Test platform ENSI-UCLA

4.2.2 Results

During the tests, we noted that the RTT varies on a large scale, so we classed all the measurements into three groups. The results are plotted on figure 10.

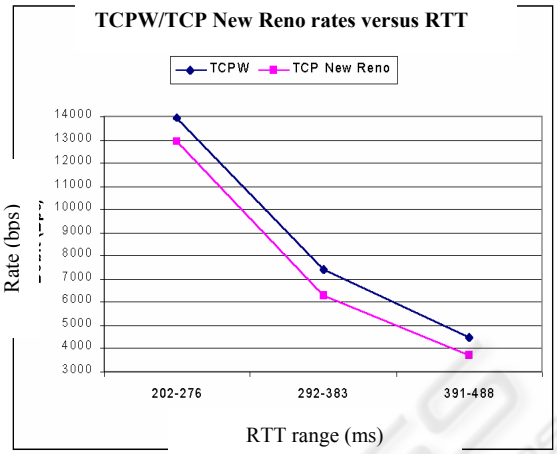

Figure 10: TCPW/TCP New Reno rate versus RTT

Once again, these results matche those yet retrieved in (Wang, 2002), showing that TCPW achieves a better rate than TCP New Reno on Internet connections. We can also observe that greater is the RTT better will be the enhancement given by TCPW. Hence, for an RTT ranging in (202-267 ms) this enhancement is about 7,9%, it is about 17,64 % for an RTT between 292 and 383 ms and it reaches 20,88% for the last range (391-488 ms).

5 CONCLUSION AND FUTURE WORK

We have presented an exhaustive study of TCPW ABSE performance in a LAN environment and in the Internet. In the first case, we have shown that this new protocol doesn't perform better than TCP New Reno as far as the transmission rate is concerned. This is an expected result, because TCPW is designed for lossy networks. Nevertheless, the fairness and friendliness of this protocol have been once again proved. In the Internet, both of the test configurations allowed us to verify the rate enhancement that TCPW achieves compared to TCP New Reno, especially when the RTT is large.
In a future work, we will be concerned with a large simulation tests on TCPW ABSE, so that we can study this protocol on larger RTT scales and loss rates. Besides, we will investigate the enhancement of the bandwidth estimation process.

REFERENCES

Casetti C., Gerla M., Lee S., Mascolo S., Sanadidi M., 2000 “TCP with Faster Recovery,” *MILCOM* 2000.

Balakrishnan H., Padmanabhan V. N., Seshan S., Katz R.H., 1997, “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links,” *IEEE/ACM Transactions on Networking*.

Gerla M., Sanadidi M.Y., Wang R., Zanella A., Casetti C., Mascolo S.,2001, "TCP Westwood: Congestion Window Control Using Bandwidth Estimation", In Proceedings of *IEEE Globecom*, Volume: 3, pp1698-1702.

Wang R., Valla M., Sanadidi M.Y., Gerla M., 2002, "Efficiency/Friendliness Tradeoffs in TCP Westwood", *Seventh IEEE Symposium on Computers and Communications*.

Ferorelli R., Grieco L.A., Mascolo S., Piscitelli G., Camarda P., 2002, “Live Internet measurements using Westwood+ TCP Congestion Control”, *IEEE Globecom*.

Wang R., Valla M., Sanadidi M.Y., Gerla M., 2002, "Adaptive Bandwidth Share Estimation in TCPWestwood", *IEEE Globecom*, Taipei, Taiwan.

Casetti C., Gerla M., Mascolo S., Sanadidi M.Y., Wang R., 2002, "TCP Westwood : End-to-End Bandwidth Estimation for Enhanced Transport over Wireless Links", *Journal of Wireless Networks*, Volume 8, pp 467-479.