TCP Adaptive RTO to improve TCP performance in mobile ad hoc networks

Haifa Touati
CRISTAL Laboratory
National School of Computer Science
Haifa.touati@cristal.rnu.tn

Ilhem Lengliz
CRISTAL Laboratory
Prince Salman Research Center
Prince Sultan University
Ilengliz@pscw.psu.edu.sa

Farouk Kamoun
CRISTAL Laboratory
National School of Computer Science
Farouk.Kamoun@ensi.rnu.tn

Abstract

In a mobile ad hoc network, temporary link failures and route changes occur frequently. With the assumption that all packet losses are due to congestion, TCP performs poorly in such an environment. This paper proposes a new mechanism called TCP Adaptive RTO (TCP AR) that improves TCP performance in mobile ad hoc networks. TCP AR distinguishes routes failures from network congestion and adapts the RTO's value to network conditions. For this purpose it relies on the estimation of the network's throughput. Using the NS-2 simulator we compared TCP AR throughput to those of TCP New Reno and TCP Westwood using different scenarios which vary in nodes' mobility and traffic load. The simulation results discussed in this paper show that TCP AR can achieve up to a 152% improvement in network throughput.

Key words*: TCP, Ad hoc network, mobility, RTO, simulation, throughput.*

1. Introduction and related work

The Transmission Control Protocol (TCP), which has evolved over time into a number of versions, from TCP Tahoe to the currently widely used TCP NewReno, provides end-to-end, reliable, congestion controlled connections over the Internet [1, 2].

Since TCP is the standard transport protocol for the Internet, it is expected to be deployed over MANETs to allow seamless integration with the Internet. However, earlier research suggested that TCP performs poorly over cellular (single-hop) wireless networks [3]. This is because the packet loss induced by the wireless link failure is erroneously interpreted as congestion-induced, which triggers an inappropriate response by the TCP mechanism. This is further exemplified in MANETs, since link failures occur more frequently due to node mobility, and give rise to packet loss.

Many researches have since focused on mechanisms to enhance TCP performance in wireless environments. Example of such schemes includes TCP Westwood (TCPW).

TCP Westwood has been initially designed in [4]. This protocol relies on a simple modification of the TCP source protocol behaviour for a faster recovery. This is performed by setting both a slow start threshold and a congestion window values 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 losses and random losses.

However, in presence of random losses, TCP Reno overreacts and reduces the window by half. Whereas, after a packet loss, TCPW resumes with the previous window as long as the bottleneck is not yet saturated (i.e., there is no buffer overflow) [5, 6].

The TCPW ABSE (Adaptive Bandwidth Share Estimation) protocol has been proposed in [7]. It palliates TCP efficiency degradation in packet loss

environment. TCPW+, described and studied in [8], is intended to improve TCPW performance in the case of Internet transmissions. TCP Westwood+ is a slightly modified version of the bandwidth estimation algorithm proposed in [9] to cope with ACK compression effect.

Nevertheless, we noted via a series of simulations that TCP New Reno, TCPW ABSE and TCP Westwood+ throughput drop significantly in presence of continuous nodes' mobility.

To address this issue, we propose a new protocol: TCP AR (Adaptive RTO) protocol to enhance TCP performance in mobile environments by adapting RTO's (Retransmission Time-Out) value to network conditions, while preserving both the TCP New Reno principle and TCPW ABSE throughput estimation.

The remaining of this paper is organized as follows. In Section 2, we recall the TCPW ABSE principles. Section 3 gives an illustrative analysis of TCP's throughput degradation causes. Section 4 is devoted to the description of the new TCP AR protocol. In Section 5, we discuss the simulation results when TCP AR is applied in an ad hoc network. Finally, Section 6 gives a conclusion and draws some perspectives to this work.

2. TCPW ABSE Protocol

In this section we intend to summarize the operation of the TCPW ABSE protocol, since the new TCP AR we are proposing rely in its major functioning on the TCPW ABSE protocol.

TCPW ABSE is a sender-only modification of TCP NewReno. The TCP sender adaptively determines a Bandwidth Share Estimate (ABSE). The estimate is based on information in the ACKs, and 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 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:

Pseudo code 1 : TCPW after 3 DUPACKs

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

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

Pseudo code 2 : TCPW after timeout

```
if (coarse timeout expires) then
  cwin = 1;
  ssthresh = (ABSE * RTTmin) / seg_size;
  if (ssthresh < 2) then
    ssthresh = 2;
  endif
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 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 change of network conditions. 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:

$$\hat{s}_k = \alpha_k . \hat{s}_{k-1} + (1 - \alpha_k) s_k \quad (1)$$

where

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

and t_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:

$$th_k = \frac{\sum_{t_j > t_k - RTT} d_j}{RTT} \quad (3)$$

where d_j is the amount of data reported by ACK *j*. In [6], the value (ε = 0.6) has been employed for the constant-gain filter to calculate the recent throughput as:

$$\hat{Th}_k = e\hat{Th}_{k-1} + (1-e)Th_k \quad (4)$$

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:

$$T_k = RTT * \frac{cwin - (\hat{Th}_k * RTT_{\min})}{cwin} \quad (5)$$

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 react 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 [6, 7]:

$$U_k = bU_{k-1} + (1-b)|s_k - s_{k-1}| \quad (6)$$

In (6), s_k is the k_{th} sample, and β is the gain of this filter, which is set to be 0.6 in [6]. 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 (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 [6] τ_k is set to be:

$$t_k = RTT + N * RTT \frac{U_k}{U_{\max}} \quad (7)$$

The value of RTT can be obtained from the smoothed RTT estimated in TCP [6].

3. Degradation of TCP's throughput: illustrative analysis

In this section we investigate the severe TCP throughput degradation phenomena caused by mobility induced behaviors in ad hoc environment. To understand the causes of this behavior we analyze traces we have got from simulations. These traces are collected from simulations of a network model consisting in 50 nodes moving continuously in a

300x300m area according to the random way point model[10]. The nodes' maximum speed is set to 20m/s. Each node uses a wireless channel model with transmission range of 70m. Simulation time is set to 1000 seconds. From t=10 sec to the end of the simulation, node 1 is sending FTP traffic to node 2. A condensed version of the simulation packed trace is shown in table 1.

Table 1 TCP network throughput degradation

Event	Time	Node	TCP Sequence N°	RTO value
s	**39,468**	_1_	1231	
s	41,566	_1_	1231	2,098
s	43,566	_1_	1231	2
s	47,566	_1_	1231	4
s	55,566	_1_	1231	8
s	71,566	_1_	1231	16
s	103,566	_1_	1231	32
s	167,566	_1_	1231	64
s	231,566	_1_	1231	64
s	295,566	_1_	1231	64
s	359,566	_1_	1231	64
s	423,566	_1_	1231	64
s	**423,746**	_1_	1246	0,18

This table lists only "s" events which denote that a packet was sent by node 1. in the last column, we report the RTO's values.

From t=39.468 sec, node 1 is trying to send the packet number 1231. It retransmits it at t=41,566 s, then at t = 43,566 s, etc. and at each retransmission the RTO is doubled. After 11 consecutive timeouts node 1 succeeds to transmit the packet at t=423.566 sec, RTO's value achieves 64 sec which causes TCP transmission to be blocked at packet 1231 for 384.1 sec. The total RTO inactivity time in this test reaches 860 seconds, so packet transmission is suspended for up to 86% of overall simulation time.

In fact, continuous nodes' mobility induced link breakage between the sender and the receiver; which in turn causes packet losses. But TCP cannot distinguish between packet losses due to route failure and packet losses due to congestion, so TCP agent triggers the exponential backoff algorithm, which consists in doubling the RTO value whenever the timeout expires up to a limit of 64 sec that refers to the maximum allowed timeout. After triggering its timer for 64 sec, even if the link is recovered, TCP will stay over one minute frozen. Subsequently this large idle time degrades the TCP throughput.

The same behavior is observed with TCPW ABSE and TCPW+. In fact neither TCPW ABSE nor TCPW+ did solve this throughput degradation in such a mobile environment since they are not designed for this purpose.

Hence with the new proposal of the TCP AR protocol, we intend to give a starting point to address this issue.

4. TCP AR Protocol

From the simulations' traces analysis presented previously, we concluded that consecutive timeouts is a key factor that affects TCP performance in a wireless environment. This is because the RTO mechanism was designed for wired networks, where a TCP agent assumes that losses are due to congestion. If a timeout occurs, TCP concludes that the network is experiencing a severe congestion and it doubles the RTO (calculated from the RTT and its variance) to give a sufficient time to the network to recover.

To palliate this insufficiency we propose the TCP Adaptive RTO (TCP AR), a new approach that combines TCPW ABSE throughput estimation method with a new RTO backoff mechanism. TCP AR adapts the RTO value to network conditions preventing the RTO's exponential backoff when losses are not due to congestion but to link failure.

The key idea of this protocol is that if a packet loss is detected by a timeout while the network is not experiencing a congestion state, it is not necessary to trigger the TCP RTO backoff mechanism. In order to distinguish between timeouts caused by congestion and those caused by ad hoc environments characteristics, TCP AR estimates network throughput. To do so, it deploys the throughput filter already proposed in TCP Westwood ABSE [cf. section 2, equations (3) and (4)].

And then when a timeout occurs, TCP AR uses this estimation to detect if the network is congested or

no (so to double or no the RTO value). The path congestion level is then determined by comparing this estimate to the instantaneous sending rate obtained from *cwin*: if $\hat{th}_k * RTT$ is larger than the current *cwin* value, this indicates a path without congestion.

Pseudo code 3 : TCP AR

```
if RTO expires then
    if   th_k * RTT > cwin  then
         (no congestion)
         keep RTO's value fixed
    else
         RTO = RTO*2
    Endif
Endif
```

Our solution inherits from ABSE only the estimation process, all other mechanisms (especially cwnd and ssthresh adjustments) are kept as in the standard New Reno version. Compared to TCPW ABSE, TCP AR uses only one estimator instead of three in ABSE, hence decreasing the complexity and reducing the computational overhead induced by the estimation process.

5. Performance Evaluation

In this section we compare TCP AR performance to that of New Reno, W ABSE and TCPW+ under various network conditions including cross traffic and mobility effects. In these experiments we compare:

1. TCP throughput versus nodes mobility speed
2. TCP throughput versus background CBR(Constant Bit Rate) load

All simulations presented in this paper were run using the LBL Network Simulator NS-2 [11] with the CMU wireless extensions. Channel propagation model is the Two Ray Ground reflection model, which is the standard propagation model used in TCP evaluation over MANETs[12]. The IEEE 802.11 DCF protocol is used at the MAC layer, and the link bandwidth is set to 2 Mbps (the NS-2 default setting).

5.1. Simulation model

Since grid topologies are more representative of adhoc configurations [13], we consider in our experiments the grid shown in figure 1.

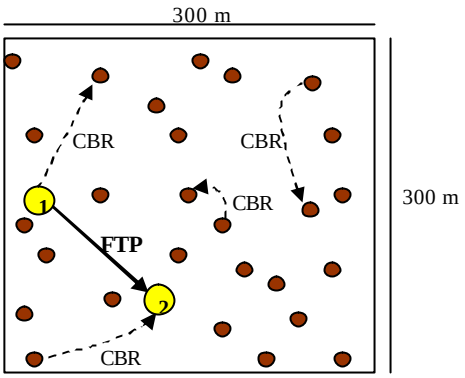


Figure 1 Simulation topology

As reported in figure 1, the network model consists in 50 nodes placed uniformly randomly throughout a 300x300m space. All nodes use a wireless channel model with a transmission range of 70m. The choice of these settings aims simultaneously at avoiding network partitioning and increasing the average number of hops. The nodes move according to the random way point model. Simulations are run for 1000 seconds. Note that these parameters have been widely used in previous TCP MANET investigations [12, 13 and 14]. All of our simulations results are based on the average value of 60 scenarios (movement patterns).

We implemented TCPW ABSE and TCP AR on NS2. To simulate TCPW+, we used the TCPW+ module yet implemented in NS-2[15]. Then, we measured the throughput of TCP New Reno, TCPW ABSE, TCPW+ and TCP AR for different scenarios.

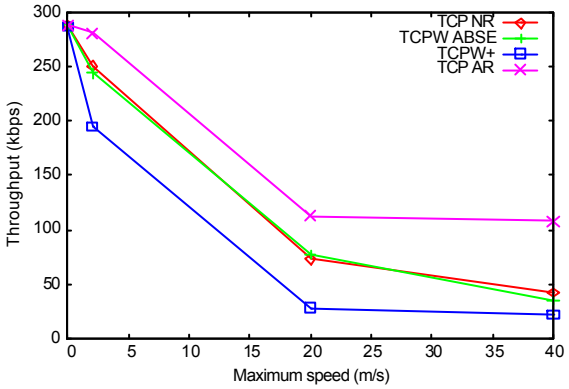
For the current work, we used the proactive routing OLSR protocol (Optimized Link State Routing Protocol) [16]. In a future work we will resume this study using the reactive routing protocol: DSR (Dynamic Source Routing)[17] and AODV (Ad hoc On-Demand Distance Vector)[18].

5.2. TCP throughput versus nodes mobility speed

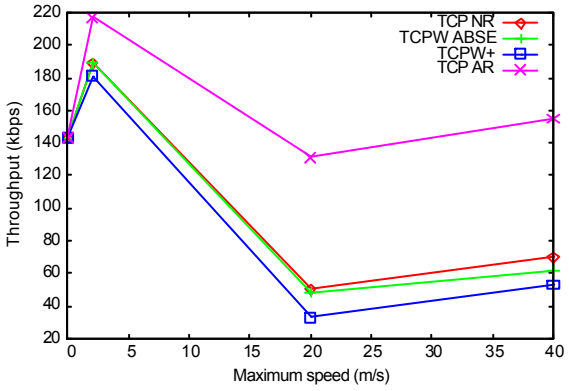
Table 3 Simulations' scenarios

	Scenario1	**Scenario2**
Number of nodes	50	30
Node density	1/1800	1/3000
Pause Time (s)	0	0
Max speed (m/s)	0, 2, 20, 40	0, 2, 20, 40

These results compare the throughput of TCP New Reno, TCP Westwood ABSE, TCPW+ and TCP AR while varying the maximum nodes' speed. Maximum speed values are chosen to reflect mobility ranging from walking to vehicular speeds: 0 m/s, which simulate a static network, 2m/s (pedestrian), 20m/s and 40m/s (vehicular). We used a pause time of 0 second, so that each node is in continuous motion during the simulation. We repeated these tests for 50 and 30 nodes to vary nodes' density in the network. Results are reported in figure2.



(a) Network with 50 mobile nodes (scenario 1)



(b) Network with 30 mobile nodes (scenario 2)

Figure 2 A comparison of TCP AR, TCPW ABSE, TCPW+ and TCP New Reno versus nodes' speed

TCP AR shows significant improvements of throughput over TCP NR, TCPW ABSE and TCPW+. For example when maximum nodes' speed is set to 40m/s, TCP AR throughput gain reaches 152% over New Reno, 205% over ABSE and 376% over Westwood+. We can see that the throughput gain increases as the maximum speed increases. It is because when nodes move rapidly link breakage will be repaired quickly so when TCP AR retransmits the lost packet there is more chance that the route is reestablished. TCP New Reno, TCPW ABSE and Westwood+ from their side will attempt to retransmit this packet much more lately.

In a static network, i.e. nodes speed set to 0 m/s, TCP AR gives the same throughput as New Reno, ABSE and Westwood+. So our proposal doesn't degrade TCP throughput when nodes don't move.

Finally, we remark that TCP Westwood ABSE gives a comparable throughput as New Reno, and sometimes it gives lower throughput. Unlike, in a wired and in a mixed wireless-wired environment, where TCPW ABSE improves TCP throughput [19, 20], in a mobile ad hoc network, TCPW ABSE performances degrade. In fact, TCPW ABSE induces a lot of computational overhead to estimate the bandwidth. This estimation will be used only when a packet loss is detected by the reception of three duplicate ACKs. But in this environment, and from traces analysis, we notice that packet losses are often detected by timeout. So, TCPW will not use the bandwidth estimation and behaves like New Reno. Tables 4 and 5 report a synthesis of TCP AR throughput gain over New Reno, ABSE and TCPW+.

Table 4 Synthesis of TCP AR gain in scenario 1

speed	50 nodes		
	AR vs NR	AR vs ABSE	AR vs W+
0 m/s	0,05%	0,06%	0,25%
2 m/s	11,86%	15,27%	44,15%
20 m/s	54,67%	46,57%	293,01%
40 m/s	152,75%	205,40%	376,85%

Table 5 Synthesis of TCP AR gain in scenario 2

Speed	30 nodes		
	AR vs NR	AR vs ABSE	AR vs W+
0	0,32%	-0,08%	0,11%
2	14,71%	14,89%	19,77%
20	161,41%	172,29%	292,99%
40	122,21%	147,53%	191,68%

3.3. TCP throughput versus CBR load

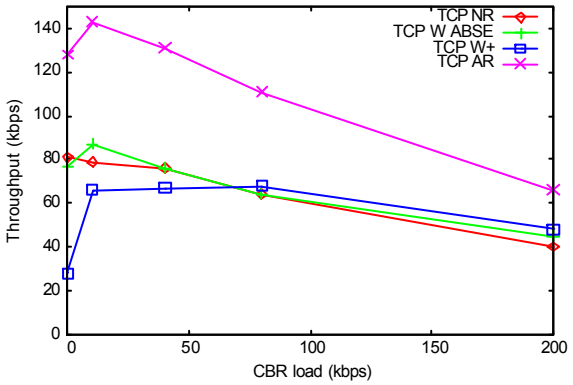
Table 5 Simulations' scenarios

	Scenario3	Scenario4
Node density	1/1800	1/1800
Pause Time (s)	0	200
CBR load (kbps)	0, 10, 40, 80, 200	0,10, 40, 80,200

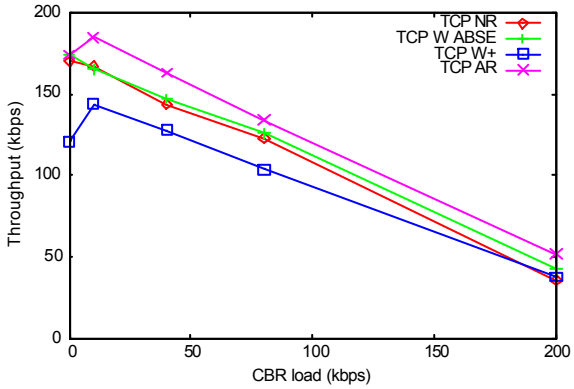
These simulations compare the throughput of TCP New Reno, TCPW ABSE, TCPW+ and TCP AR while varying the volume of background traffic:
- without CBR traffic,
- 10 CBR connections offering a total load of 10 kbps
- 10 CBR connections offering a total load of 40 kbps
- 10 CBR connections offering a total load of 80 kbps
- and finally, with 10 CBR connections offering a total load of 200 kbps

The background load is generated by 10 CBR connections. The CBR packet sizes were fixed at 512 bytes. We run the tests for two different pause time :
1. 0 second (scenario 3): all nodes are in continuous motion during the simulation; this reproduces an environment with high mobility;
2. 200 seconds (scenario 4): to test if the TCP AR protocol can usually maintain a good performance in a network with low mobility and in presence of background traffic.

Results are plotted on Figure 3.



(a) Pause Time = 0 sec (scenario 3)



(b) Pause Time = 200 sec (scenario 4)

Figure 3 Comparison of TCP AR, ABSE, Westwood+ and New Reno throughput versus CBR load

Figure 3(a) shows that both in a congested and a non congested network, TCP AR achieves a significant gain in the throughput compared to TCP NR, TCPW ABSE and TCPW+ when nodes are in continuous motion since consecutive timeouts are more likely to be due to routes failures than due to congestion. So while New Reno, TCPW ABSE, and TCPW+ are blocked due to their large RTO, TCP AR connection is re-established faster and data transmission is resumed.

Results plotted in figure 3 (b) show that in the worst case (without CBR traffic) TCP AR offers comparable throughput to those of New Reno and Westwood. Elsewhere, it outperforms them.

We can see that the throughput gain gets higher when the pause time is set to 0 s (nodes in continuous motion) than when it is set to 200s. In fact, as nodes' mobility increases, as the routes' failure probability gets higher. Table 6 and 7 synthesize TCP AR throughput gain over TCP NR, TCPW ABSE and TCPW+ for various CBR load.

Table 6 Synthesis of TCP AR gain in scenario 3

CBR Load (kbps)	Pause Time = 0 second		
	AR vs NR	AR vs ABSE	AR vs W+
0	57,85%	65,07%	342,63%
10	81,42%	64,24%	115,19%
40	59,31%	72,16%	95,87%
80	72,57%	72,85%	61,66%
200	65,28%	47,88%	36,02%

Table 7 Synthesis of TCP AR gain in scenario 4

CBR Load (kbps)	PauseTime = 200 seconds		
	AR vs NR	AR vs ABSE	AR vs W+
0	2,57%	0,07%	44,22%
10	10,54%	11,81%	28,84%
40	13,13%	11,16%	27,02%
80	9,02%	6,85%	28,75%
200	42,00%	21,07%	34,42%

6. Conclusions and further work

TCP New Reno performs poorly in mobile ad hoc networks caused by frequent route changes. In this

paper we propose a new scheme called TCP AR. With this mechanism, a TCP sender can determine if a retransmission timeout is due to network congestion or temporary route loss by comparing the instantaneous sending rate to the throughput estimation. Hence, TCP AR doubles the RTO's value only if the timeout is due to congestion, otherwise the RTO's value is frozen.

Simulations' results show that TCP AR achieves a better performance when compared with TCP NR, TCPW ABSE and TCPW+ in terms of efficiency (expressed via the network throughput). It provides up to 152% throughput gain with respect to New Reno, and shows more outstanding improvements in performance as node's mobility increases. TCP AR is a pure end to end approach and not dependent on lower layers.

In a future work, we intend to carry out more simulations to investigate other performance metrics such fairness and friendliness of TCP AR toward TCP NR, to study the effects of reactive routing protocol on the performance of our solution and to implement the TCP AR protocol.

7. References

[1] Jacobson V., *Congestion Avoidance and Control*, ACM Computer Communications Review, 18(4): 314 - 329, August 1988.

[2] Jacobson V., *Berkeley TCP evolution from 4.3-Tahoe to 4.3 Reno*, Proceedings of the 18th Internet Engineering Task Force, University of British Colombia, Vancouver, BC, Sept. 1990.

[3] Abdullah-Al-Mamun M., Rahman M.,Tan H., *performance evaluation of TCP over routing protocols for mobile ad hoc networks*, CHINACOM 2006.

[4] Casetti C., Gerla M., Lee S., Mascolo S., Sanadidi M., *TCP with Faster Recovery*, MILCOM 2000.

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

[6] Wang R., Valla M., Sanadidi M.Y., Gerla M., *Efficiency/Friendliness Tradeoffs in TCPWestwood*, Seventh IEEE Symposium on Computers and Communications, 2002.

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

[8] Ferorelli R., Grieco L. A., Mascolo S., Piscitelli G., Camarda P., *Live Internet measurements using Westwood+ TCP Congestion Control*, IEEE Globecom 2002, Taipei, Taiwan, November 18-20, 2002.

[9] Grieco, L. A., and Mascolo, S., *Westwood TCP and easy RED to improve Fairness in High Speed Networks*, Proceedings of IFIP/IEEE Seventh International Workshop on Protocols For High-Speed Networks, PfHSN02, (Berlin, Germany, April, 2002).

[10] Camp T., Boleng J., Davies V., *A Survey of Mobility Models for Ad Hoc Network*, Research. Dept. of Math. and Computer Sciences Colorado School of Mines, Golden, 2002.

[11] Network Simulator Ns2, http://www.isi.edu/nsnam/ns.

[12] Papanastasiou S., Ould-Khaoua M, Mackenzie L. M., *On the evaluation of TCP in MANETs*, Proc. of International Workshop on Wireless Ad-hoc Networks, London, UK, 23 - 26 May 2005.

[13] Chen. L.-J., Sun. T., Yang, G., Sanadidi, M. Gerla, M., *AdHoc Probe: Path Capacity Probing in Ad Hoc Networks*, Proceedings of the First International Conference on Wireless Internet WICON '05, pp 156- 163, Budapest, Hungary, July 10 - 15, 2005

[14] Holland G., Vaidya N., *Analysis of TCP performance over Mobile ad hoc networks*. Proceedings of the fifth annual ACM/IEEE International conference on Mobile computing and networking, pp 219-230, ACM Press 1999.

[15] TCP Westwood+ NS-2 modules, June 2003, http://193.204.59.68/mascolo/tcp%20westwood

[16] Clausen T., Jacquet P., *Optimized Link State Routing Protocol (OLSR), RFC 3626*, IETF MANET Working Group, http://hipercom.inria.fr/olsr/rfc3626.txt, 2003.

[17] D. Johnson D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing*.

[18] C. E. Perkins, E. M. Belding-Royer, and S. Das, "Ad hoc On- Demand Distance Vector (AODV) Routing". *RFC 3561*, July 2003.

[19] Lengliz I., Touati H., Kamoun F., Sanadidi M. Y., *Experimentations towards TCP Westwood application in ad-hoc mobile networks*, Med-Hoc Net'03, Mahdia, Tunisia, June 25-28, 2003.

[20] Lengliz I., Touati H., Kamoun F., Sanadidi M. Y., *Measurements on Fairness and Friendliness of TCP Westwood in Wireless Environments*, Med-Hoc Net'04, Bodrum, Turky, June 25-30, 2004.