Mansoura Engineering Journal

Volume 33 | Issue 1 Article 4

12-1-2020

Improving TCP Performance over Mobile Ad Hoc Networks with an Adaptive Backoff and Response Approach.

Tamer Ghanem
Instructor., Information Technology Department., Faculty of Computers and Information., Menoufia University., Shebin El-Kom., Egypt., amersg@hotmail.com

Wail Elkilani
Assistant Professor., Information Technology Department., Faculty of Computers and Information., Menuofia Unjversity., Shebin El-Kom., Egypt., welkilani@gawno.com

Mohiy Hadhood
Vicc dean of Faculty of Computers and Information., Menoufia University., Shebin El-Kom., Egypt., mmhadhoud@yahoo.com

Recommended Citation
Ghanem, Tamer; Elkilani, Wail; and Hadhood, Mohiy (2020) "Improving TCP Performance over Mobile Ad Hoc Networks with an Adaptive Backoff and Response Approach.," *Mansoura Engineering Journal*: Vol. 33 : Iss. 1 , Article 4.
Available at: https://doi.org/10.21608/bfemu.2020.126808

Mansoura Engineering Journal, (MEJ), Vol. 33, No. 1, March 2008. E. 19

Improving TCP Performance over Mobile Ad Hoc Networks with an Adaptive Backoff and Response approach

تحسين أداء بروتوكول التحكم فى الإرسال على الشبكات المحمولة المبعثرة بضبط مؤقت إعادة الإرسال مع تحسين الإستجابة لعودة الإتصال

Tamer F. Ghanem* Wail S. Elkilani** Mohiy M. Hadhood***

* Demonstrator, Information Technology Dept, Faculty of Computers and Information, Menoufia University, Shebin El-Kom, Egypt, tamerfg@hotmail.com
** Lecturer, Information Technology Dept, Faculty of Computers and Information, Menoufia University, Shebin El-Kom, Egypt, *welkilani@gawab.com*
*** Vice dean of Faculty of Computers and Information, Menoufia University, Shebin El-Kom, Egypt, mmhadhoud@yahoo.com

ملخص البحث

فى الشبكات الاسلكية المحمولة المبعثرة يتكرر تغير و انقطاع مسارات الإتصال. و لذلك فإن أداء بروتوكول التحكم فى الإرسال (TCP) على هذه الشبكات يكون سيئ حيث أنه يعتبر ان اى فقد فى البيانات هو نتيجة لوجود تزاحم فى هذه الشبكات. إن كثيرا من البحث الجارى لتحسين أداء هذا البروتوكول على هذه الشبكات يحتاج إلى استقبال معلومات من الشبكة أو من الطبقات السفلى منه. فى هذا البحث نتعرض لأسلوب جديد لتحسين أداء هذا البروتوكول. هذا الإسلوب يعتمد على بدء الإرسال من جديد فى أقرب وقت بعد بناء مسار جديد الى الهدف بعد تعرضه للإنقطاع. ولتنفيذ هذا الإسلوب فإننا نقترح نظام لضبط قيمة مؤقت إعادة الإرسال فى هذا البروتوكول. و قد أوضحت النتائج ان هناك تحسن فى الأداء بمقدار ١٧% و ذلك بدون الحاجة الى معلومات من الشبكة أو من الطبقات السفلى.

ABSTRACT

In a Mobile Ad Hoc Network (*MANET*), temporary link failures and route changes happen frequently. With the assumption that all packet losses are due to congestion, TCP performs poorly in such environment. Most of research performed for improving TCP performance over *MANET* requires feedback from the network or the lower layer. Moreover, several attempts have been proposed for a layered TCP improvement. Yet, their percentage enhancements are not satisfactory. In this paper, we explore a new approach to improve TCP performance. This approach depends on beginning transmission as soon as route reestablishment is done after route failure. To do this we propose an adaptive backoff strategy with decreasing congestion window and slow start threshold values when we receive acknowledgement from the receiver. The simulation results showed that this approach had achieved an average performance improvement of 17%. Moreover, the proposed technique does not require feedback from the network or the lower layers.

Keywords:

Mobile ad hoc networks, MANET, TCP

Accepted March 31, 2008

1. Introduction

A mobile ad hoc network consists of a collection of peer mobile nodes that are capable of communicating with each other without help from a fixed infrastructure. The inter connections between nodes are capable of hanging on a continual and arbitrary basis. Nodes within radio range communicate directly via wireless links, while far apart nodes use other nodes as relays in a multi-hop routing fashion.

The TCP protocol has been extensively tuned to give good performance in the traditional wired network environment as a transport layer. However, TCP in its present form is not well suited for mobile ad hoc networks (*MANETs*). In addition to all links being wireless, frequent route failures due to mobility can cause serious problems to TCP as well. Route failures can cause packet drops at the intermediate nodes, which will be misinterpreted as congestion loss.

In fact, when a route failure happens for a period of time greater than retransmission timer value, TCP will understand this as congestion which means decreasing both the congestion window (*CWND*) and slow start threshold (*SSThr*). Then it retransmits the first unacknowledged packet and executes backoff by doubling the value of retransmission timer. With multiple successive backoffs, the value of the retransmission timer will be too long. However during the long retransmission period, the route may come back but TCP will not try to retransmit the first unacknowledged packet until the retransmission timer expires. So there is a wasted time that TCP will not use although the route may come back some period ago.

The ideal solution for the route failure problem is to freeze its state as soon as the route breaks and resume as soon as a new route is found [**Error! Reference source not found.**,2,3]. However, this requires instant notification or feedback from the intermediate nodes to all TCP senders, and from the network layer to the transport layer. Such a feedback system can be difficult to implement and expensive to operate.

In this paper, we explore a new adoption technique of TCP to frequent route failures without relying on feedback from the network. It is based on adapting the TCP backoff after the expiration of the retransmission timer, when the sender receives an acknowledgement which covers the retransmitted packet, TCP returns to its state before the expiration of its retransmission timer. Moreover, *CWND* and *SSThr* will be divided by 2 and 4 respectively. We call it adaptive backoff and response approach (ABRA). Our study has shown that this approach can improve TCP performance over mobile ad-hoc networks.

The rest of this paper is organized as follows: In the next section, related work is described. In section 3, we discuss in detail the ABRA. In section 4, the simulation methods are stated. Then we show the simulation results and analysis of the performance in section 5. The conclusion is in section 6.

2. Related Work

In this section we present the various proposals that have been made in the literature to overcome the performance of the TCP for ad hoc networks. The main problem which affects TCP performance over ad hoc networks is frequent route failures which are misunderstood by TCP as network congestion. The proposals for improving TCP performance over ad hoc can be divided into cross layer proposals and layered proposals [4]. Cross layer proposals depends on providing lower layer information to the upper layer. This information helps the upper layer to perform better. Layered proposals rely on adapting TCP layer independently from other layers. Here, we introduce a layered proposal to enhance TCP performance over ad hoc.

In general, cross layered solutions report better performance than layered solutions. But deciding which one of them to use depends on the interest of the advantages provided by each one. Cross layered solutions provide short term gain and it is more complex to implement and design. Layered solutions provide long term solutions and designing protocols in isolation. Most research work in the area of TCP over ad hoc is concentrated on using cross layered approaches [3,5,6,7].

In the following, we present the main layered proposals made in the literature.

Fixed RTO [8]: This technique is implemented at the sender and does not depend on feedback coming from network. It concludes that if two successive timeouts exists, then it is a route failure and not a network congestion. The action taken due to this situation is to retransmit the unacknowledged packet and the RTO is not doubled a second time and remains fixed until the route is reestablished and the retransmitted packet is acknowledged. This is in contrast to standard TCP, in which an exponential backoff algorithm is used.

TCP Door [9]: It stands for TCP Detection of Out-of-Order and Response. This technique depends on detection of out-of-order (OOO) events of the received packets. The detection of OOO is accomplished at the sender or the receiver. If the detection is done at the receiver, it should notify the sender with this event. Once the TCP sender knows about an OOO event, there are two actions to be taken, first, it temporarily disables congestion control for a specific time period (*T1*), second, recovers during congestion avoidance. In the second action, if during the past time period (*T2*) the TCP sender has suffered from "congestion symptoms" (such as gross timeout or three duplicate ACKs) and entered the congestion avoidance state (such as halving its window size), it should recover immediately to the state before such congestion avoidance action was invoked.

In Fixed RTO proposal, the belief that two consecutive timeouts are the exclusive results of route failures need more analysis, especially in cases of congestion. Also In TCP Door, the belief that OOO events are the exclusive results of route failure deserves much more analysis. Actually, multipath routing protocols may produce OOO events that are not related to route failures [4].

We believe that proposing an effective TCP enhancement technique based on layered approach will empower system design to make use of our solution.

3. Adaptive Backoff and Response Approach

In this section, we discuss in details the proposed adaptive backoff and response approach (ABRA). For TCP, in the event of a retransmit timeout, TCP retransmits the oldest unacknowledged packet and doubles the retransmit timeout interval (*RTO*) [10]. This process is repeated until an ACK for the retransmitted packet has been received. So retransmit timeout interval may be very long and the route may be reestablished some time ago, this leads to a wasted time. This wasted time can be used to send packets since the route to the receiver exists. We try to make use of this wasted time by making the retransmit timeout interval depends on smoothed round trip time (*SRTT*).

The ABRA depends on saving the values of congestion window, slow start threshold and smoothed round trip time as last_cwnd, last_ssthr and last_srtt respectively when the retransmission timer expires. Then instead of multiplying *RTO* interval by two each time, we multiply it by a value called $backoff_{new}$ between one and two depending on the last_srtt. The $backoff_{new}$ and the new RTO value (RTO_{new}) are computed as follows:

$$backoff_{new} = 1 + \frac{last\_srtt - \min\_srtt}{\max\_srtt - \min\_srtt}$$

$$RTO_{new} = backoff_{new} * RTO_{current}$$

Where $RTO_{current}$ is the current RTO value, *min_srtt* is the minimum smoothed round trip time seen so far, and *max_srtt* is the maximum smoothed round trip time seen so far as shown in figure 1. We initialize *min_srtt* and *max_srtt* with the values 0.1 and 0.6 seconds respectively. Choosing initialization values of these two variables were done experimentally since they give the best results.

The reader may ask why this form was chosen. In fact, we have experimented many forms. In the initial phase of the experiments, fixed values between one and two are given to $backoff_{new}$, then we develop our idea to use the previous mentioned formula but with adoption depending on round trip time (rtt). Finally, we have used smoothed round trip time (srtt) instead of rtt. We have noticed that the previous mentioned form with an adoption depending on srtt can improve the performance of TCP over ad hoc networks.

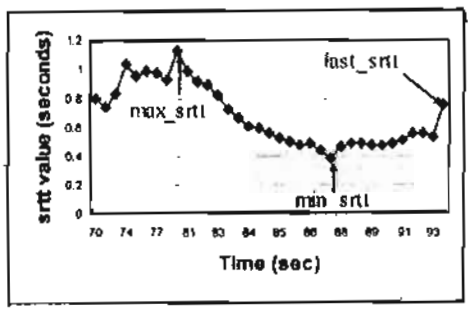

***Figure 1** : The meaning of min_srtt, max_srtt and last_srtt*

When the sender receives an acknowledgement, TCP returns to its

transmission state before the expiration of its retransmission timer. Also, congestion window (*CWND*) and slow start threshold (*SSThr*) will be divided by 2 and 4 respectively to avoid network sudden congestion. It is be noted that the values, by which CWND and SSThr are divided, are set by trial and error since they give best results during simulation experiments.

4. Simulation Methods

To emulate the proposed algorithm, a simulation study was done using NS2 network simulator [11] (version NS2.29). It included the CMU wireless network extension [12] that implemented a set of ad hoc routing protocols (DSR [13], AODV [14], etc.) and 802.11 MAC layer, and a wireless channel model with a distance of 250m transmission range. We add our algorithm to the implementation of the NewReno TCP [15] in NS2. The ad hoc network model used in this study was a set of 20 nodes confined to a 1000m by 750m rectangular area. Mobility is simulated using the random waypoint mobility model over 50 different mobility scenarios for each maximum moving speed. A zero pause time was chosen to make the nodes move all the time. The maximum moving speed is set to 2, 10 and 20 m/s.

Routing protocols: DSR [13] and AODV [14] are used as on-demand routing protocols. On-demand protocols do not maintain routes between all the nodes in ad hoc network. Rather, routes are established when needed through a route discovery process in which a route request is broadcast. A route reply is returned either by the destination or by an intermediate node with an available route. Route error messages are used to invalidate routing table entries when link failures are detected.

Transport protocols: Mixed UDP and TCP [15] traffic are used. For UDP, we used the existing standard UDP protocol. For TCP, there are many TCP algorithms commonly used in the internet. One of the differences between these TCP versions lies in their methods of recovering from packet loss which supposedly comes from network congestion.. We took the NewReno version of the TCP protocol as our base case because it outperforms the previous versions like Reno and tahoe TCP, also, because its popularity in last years [16].

The traffic work load is a single TCP connection. Using an FTP, a file transfer was conducted over this connection for 120 seconds. This single TCP connection was joined by 10 CBR flows (2KBps each) that were established among 10 random pairs of nodes to induce background/cross traffic and congested conditions.

Performance metrics: The improvement percentage of throughput in NewReno TCP is our main metric. Our work is implemented in NewReno TCP and compared by the Fixed RTO [8] approach because it is a layered approach which is interested in adopting backoff value. Although the authors in [8] implemented Fixed RTO in Reno TCP, hence we have implementd it in NewReno TCP because of its good performance and popularity as we said earlier. We have measured the highest acknowledge

number that received by the TCP sender. The higher the acknowledge number, the better performance had TCP achieved.

The improvement is measured as follows:

$$imp\% = \frac{hack_{ABRA} - hack_{NewReno}}{hack_{NewReno}} * 100$$

where $imp\%$ is the improvement percentage, $hack_{ABRA}$ is highest acknowledge number of the ABRA and $hack_{NewReno}$ is the highest acknowledge number of NewReno TCP. This is done for each movement scenario. Then we take the average for these 50 scenarios.

5. Performance Results

As we have predicted earlier, our adaptive TCP can improve the TCP performance by trying to adapt the retransmission timer value so we can send packets as soon as possible when a broken route comes back, The results from running the simulation have validated this hypothesis. Figure 2 illustrates an example case on how our ABRA approach out-performs the NewReno TCP (under a particular scenario and setup).

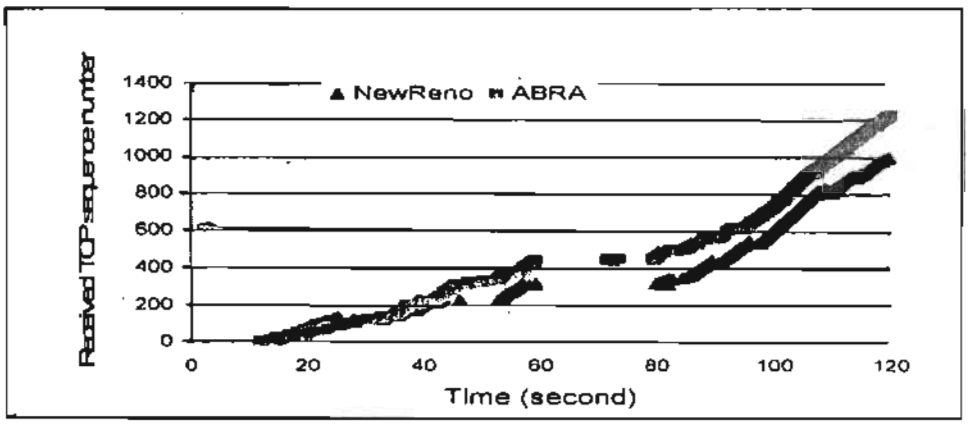

Figure 2: Comparasion between ABRA and NewReno TCP with respect to received TCP sequence number.

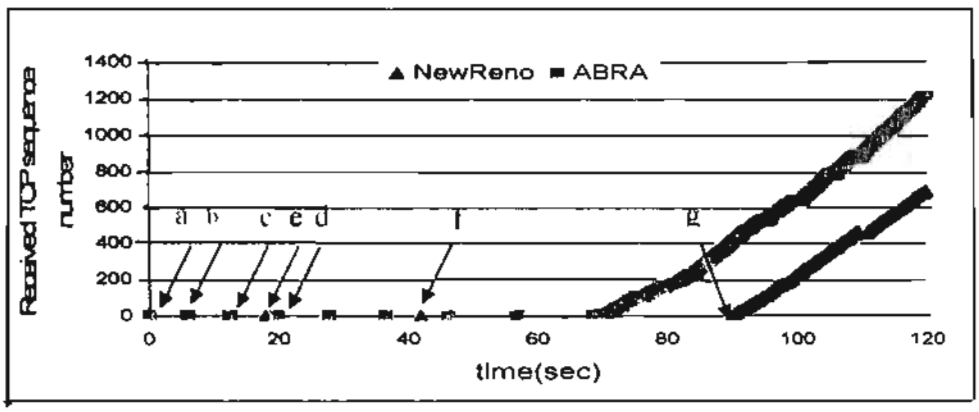

Figure 3 (a)

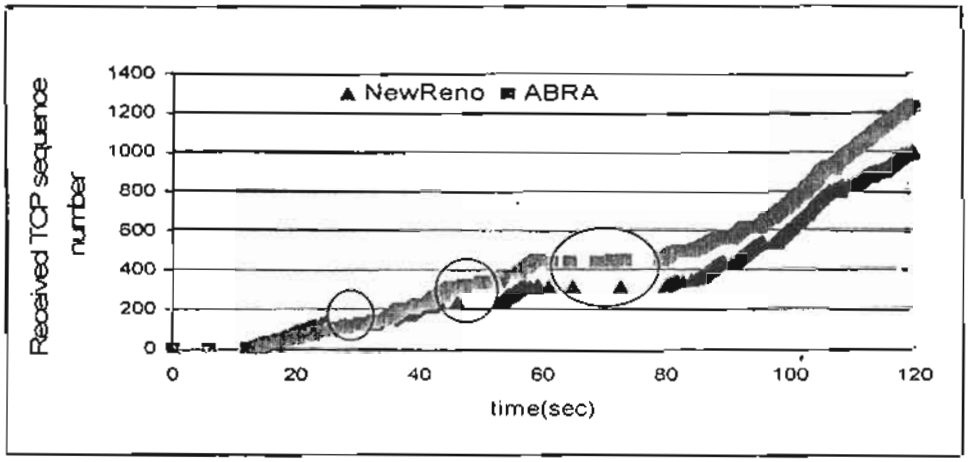

Figure 3 (b)

Figure 3: *An example shows the interval of retransmitting the unacknowledged packet*

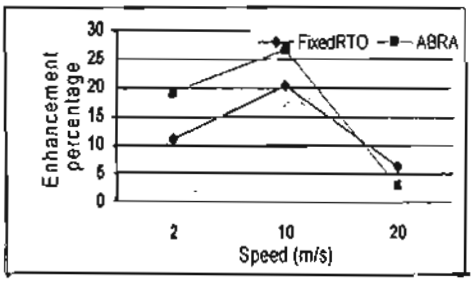

Figure 4 Enhancement percentage at different speeds using DSR

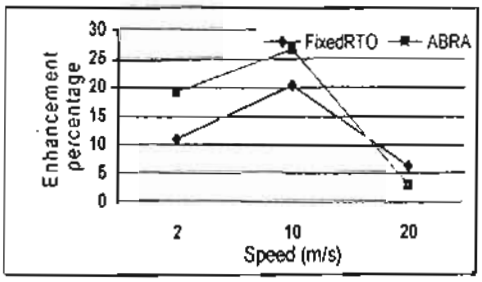

Figure 5: Enhancement percentage at different speeds using AODV

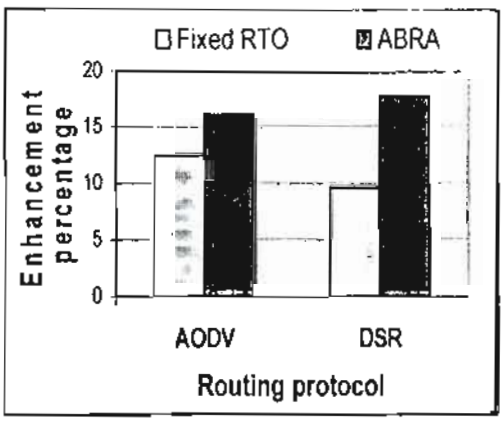

Figure 6: Total enhancement percentage average

Routing protocol	max speed (m/s)	Enhancement %	
		Fixed RTO	ABRA
AODV	2	10.92	19.1
	10	20.18	26.57
	20	6.12	2.81
AODV average		**12.46**	**16.08**
DSR	2	5.1	37.62
	10	22.88	17.34
	20	0.61	-1.02
DSR average		**9.59**	**17.72**

Table 1: Results summary.

Figures (3-a) shows a route failure at beginning of transmission. We can notice how the NewReno TCP retransmission timer value is doubled at points (a,b,c,d) when the timer expires and need to retransmit the unacknowledged packet. On the other hand, ABRA TCP at points (e,f,g) does not double the value of the retransmission timer each time it expires but it depends on *SRTT* to set the backoff value. We notice that ABRA resumes normal sending at a time of 69 seconds, but NewReno TCP resumes normal sending at a time of 90 seconds. We can easily conclude that the route comes back between 56 and 69 seconds which was early captured by ABRA. This shows ABRA TCP begins normal transmission as soon as possible. The same behavior can be shown in figure (3-b) where the circled region shows a route failure during data transmission. As noted previously, ABRA outperforms NEW Reno and tries to resume normal transmission early.

Figure 4 shows the enhancement percent average using the DSR routing protocol. Both FixedRTO and ABRA TCP outperform NewReno especially at 2 and 10 m/s, but at 20 m/s they behave like NewReno. ABRA TCP outperforms FixedRTO at low speed and is slightly below at medium and fast speed.

Figure 5 shows the enhancement percent average using AODV routing protocol. Both algorithms outperform NewReno. ABRA outperforms FixedRTO at low and medium speed and is small below FixedRTO at high speed.

Figure 6 shows the overall enhancement percent average of all speeds. FixedRTO outperforms NewReno by 12.5% for AODV and 9.6% for DSR. ABRA TCP outperforms NewReno by 16% for AODV and 17.7% for DSR. Table 1 summarizes the overall results mentioned before.

6. Conclusions

In this paper, an approach to improve the TCP performance over mobile ad hoc networks has been proposed. The proposed approach adapts the value of retransmission timer value to retransmit the unacknowledged packet as soon as possible after route reestablishment. Our simulation results showed that a significant improvement of approximately 17% can be obtained in the TCP throughput.

Our approach does not rely on the feedback from lower layers or from the network. As we have pointed out before, the feedback mechanism can be difficult to implement and expensive to deploy. The obvious tradeoff is that a feedback-based approach is more accurate because the information is directly from the network. So the conclusion is, for improving TCP over ad hoc network, the feedback-based approach should be used if available, otherwise, our approach can work on any environment and still deliver a significant improvement.

More simulations are to be done in future to explore the effect of ABRA TCP on other ad hoc routing protocols. In order to accurately simulate the realistic congested network envi-

ronment, there is a need to experiment with multiple TCP flows.

References

1 K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback based scheme for improving TCP performance in ad hoc wireless networks". IEEE Personal Communications Magazine, 8(1):34-39,Feb. 2001.

2 G. Holland and N. Vaidya. "Analysis of TCP performance over mobile ad hoc networks". In Proceedings of ACM MobiCom 99, Seattle, Washington, Aug. 1999.

3 J. Liu and S. Singh. "ATCP: TCP for mobile ad hoc networks". IEEE Journal on Selected Areas in Communications, 19(7): 1300-1315, July 2001.

4 Ahmad Al Hanbali, Eitan Altman, And Philippe Nain, "A Survey Of Tcp Over Ad Hoc Networks", IEEE communications Surveys & Tutorials, Third Quarter 2005, Volume 7, No. 3

5 K. Chandran et al., "A Feedback-Based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks", Proc. Intl. Conf. Distributed Computing Systems (ICDCS98), Amsterdam, Netherlands, May 1998.

6 G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks", ACM Wireless Networks, vol. 8, no. 2, Mar. 2002, pp. 275-88.

7 D. Kim, C. Toh, and Y. Choi, "TCP-BuS: Improving TCP Perform-ance in Wireless Ad Hoc Netwo-rks", J. Commun. and Net., vol. 3, no. 2, June 2001, pp. 175-86.

8 T. Dyer and R. Boppana, "A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad Hoc Networks", Proc. ACM MOBIHOC, Long Beach, CA, USA, 2001, pp. 56-66.

9 F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad Hoc Networks with Out-of-order Detection and Response", Proc. ACM MOBIHOC, Lausanne, Switzerland, June 2002, pp. 217-25.

10 V. Paxson and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, Standards Track, NASA GRC/BBN, Nove-mber 2000.

11 Network simulator ns2.29, URL: http://www.isi.edu/nsnam/ns/index.html, Dec-ember 2005.

12 Monarch project, Wireless and Mobility Extensions to ns-2. URL: http://monarch.cs.cmu.edu/-cmu-ns.html.

13 D. B. Johnson, D. A. Maltz and Y. C. Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)". Internet Draft, draft-ietf-manet-dsr-10.txt, July 2004, work in progress.

14 C. E. Perkins, E. M. Belding-Royer and S. R. Das. "Ad hoc On-Demand Distance Vector (AODV) Routing", Request For Comments, http://www.ietf.org/rfc/rfc3561.txt , July 2003, Experimental RFC.

15 S. Floyd, T. Henderson and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 3782, Standards Track, April 2004.

16 Stylianos Papanastasiou, "Investigating TCP Performance in Mobile Ad Hoc Networks", thesis for doctor of philosophy degree, faculty of Information and Mathematical Sciences, university of Glasgow, October 2006.