

# On the evaluation of TCP in MANETs

Stylios Papanastasiou, Mohamed Ould-Khaoua, Lewis M. Mackenzie

Department of Computing Science

University of Glasgow

Glasgow, UK G128QQ

Email: {stelios,mohamed,lewis}@dcs.gla.ac.uk

**Abstract**—Past research efforts have denoted the problematic behaviour of traditional TCP agents in MANET environments and have proposed various remedies across the networking stack. However, there has not been an overall performance evaluation of different TCP agents under varying mobility conditions which takes into account past experiences in MANET evaluation. This work aims to rectify this shortcoming through detailed evaluation of prevalent TCP variants in different topology settings over the AODV routing protocol. Subsequent results reveal the performance merits of TCP Vegas and NewReno in MANETs with respect to Reno which is further explored and accounted for. Finally, insight is provided through extensive tracing on the interaction of TCP with the routing protocol.

## I. INTRODUCTION

Early in the course of MANET research, TCP behaviour came into focus as it became understood that TCP's misreaction to non-congestion induced packet losses results in suboptimal performance in such wireless environments [5]. Packet losses in wired networks are largely attributed to buffer contention at branching points in the network topology, but in mobile ad hoc networks losses due to errors or mobility may be at least as frequent. The appropriate reaction to each of these causes is different to that of contention losses but as TCP is unable to distinguish between them, it misreacts. Further research has revealed that the widely implemented 802.11 distributed MAC mechanism can severely penalise TCP flows when competing with other congestion aware or unaware traffic. This may lead to a "capture" effect whereas some TCP flow (or flows) severely under-utilise the offered bandwidth [17].

Several studies have examined the behaviour of TCP variants under specific conditions to highlight a particular performance aspect and suggest modifications. However, there have been few detailed comparison studies among different TCP variants. Earlier research work by Ramarathinam et al. [16] examined the goodput performance of TCP Tahoe, Reno, NewReno and SACK in a static multi-hop network under three different routing protocols and noted that overall, Reno is the best performer. Work by Xu et al. examined the behaviour of different TCP variants in the context of hidden terminal interference but only in the case of string topologies [18]. In the same work the apparent performance merit of Vegas was noted in string topologies and a simple modification was suggested to improve goodput for the Reno TCP variants. Finally, an evaluation of TCP variants over wireless networks has been performed by Grieco et al. in [8], where the relative

merits of TCP Westwood over Vegas and NewReno were discussed albeit without any mobility considerations.

In this study, we compare the throughput performance of TCP Vegas, Reno and NewReno in dynamic MANET scenarios. First, we highlight TCP's interaction with the routing protocol which provides insight for the discussion that follows. Then, the comparison is performed over the AODV routing protocol and the parameters chosen for the routing and transport agents are discussed in detail.

The rest of the paper is organised as follows. In the next section, an overview of typical TCP behaviour over AODV routing at the time of link breakage is presented; further, an effect of the routing cache on TCP behaviour previously unmentioned is identified. In Section II we present the simulation set up for the performance evaluation and justify the choice of simulation parameters. Section IV contains the simulation results for single TCP connections in a variety of MANET scenarios. In Section V, we include a discussion on the performance penalty incurred by the various TCP agents in our simulations. Finally, Section VI concludes the paper and offers suggestions for future work.

## II. AODV INTERACTION WITH TCP

This section examines the behaviour of TCP Reno in a simple mobile ad-hoc network scenario. This example is instructive on the challenges faced by TCP in a wireless multi-hop environment and provides insight on the interaction of TCP with the AODV routing mechanism. In particular, the discussion that follows aids comprehension when contemplating throughput performance issues with TCP in subsequent sections.

The scenario topology consists of a simple five node string topology (nodes  $A \rightarrow E$  and an additional node (node F) which is initially disconnected from the network, as shown in Figure 1(a). The nodes in the string are spaced 200m apart and feature standard Lucent WaveLan II [11] transceivers with bandwidth of 2Mbps. The transmission range of the transceivers is fixed at 250m using a flat, ideal signal propagation model under which there is no attenuation up to the transmission range limit and nullifies the signal strength beyond that limit. Obviously, such a propagation model cannot occur in reality as some signal attenuation is always present but it is utilised so as to isolate the effects of route breakage and disregard other effects such as those caused by interference [18]. To deal with the standard hidden terminal effect the

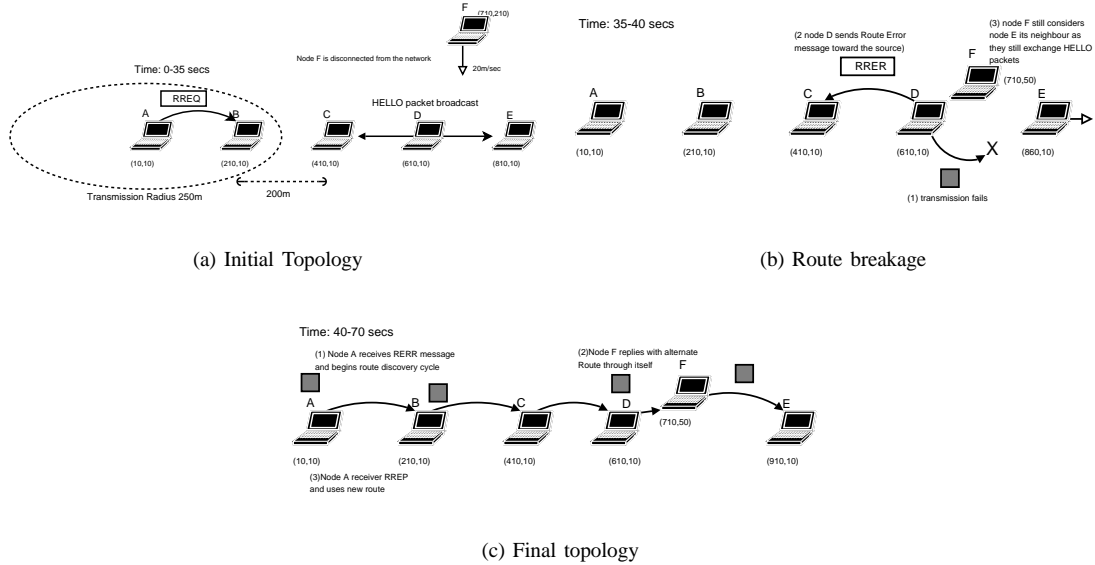


Fig. 1. A scenario depicting AODV operations after a route break

RTS/CTS mechanism is active throughout the simulation run. The routing protocol used is AODV-UU [12] which is a working AODV implementation utilised in conjunction with the ns-2 simulator [6] and which is implemented according to the corresponding RFC [15]. HELLO packets are used in this implementation to detect route failures, and in particular if a node has not broadcast a HELLO beacon for 5 seconds the link is considered obsolete. Link layer (LL) feedback is not considered in this experiment although a discussion follows later on the use of such feedback. A complete list of the AODV parameters used is presented in Table I.

The scenario setup is as follows; An FTP bulk transfer with an infinite backlog is established between the end points of the string topology, namely nodes A and E. Initially, AODV broadcasts an RREQ packet which is forwarded in sequence by each intermediate node until it reaches node E, which responds by issuing an RREP to A informing it of the  $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$  route. Note that node D is aware that E is its neighbour by the time the RREQ packet arrives. However, since the gratuitous RREP feature has been disabled in the AODV agents, only the intended destination may issue a reply to the discovery packet. After a few seconds and once the bulk transfer is on its way, node F moves and get stationed between nodes' D and E, thus entering both nodes' transmission radius. At the 30 sec mark node E starts moving horizontally away from Node D at 10m/sec until at 35 secs the signal of D no longer reaches E and hence the route becomes invalid. At its new destination node E is still a neighbour of node F but cannot be contacted by node D. Node D realises that its link to E has been invalidated when it notices the absence of HELLO packets (since MAC feedback has been disabled). A Route Error Packet is then sent back by node E to the source (node A) where a new route discovery cycle begins to

probe for an alternate route to the destination. Note that as the route error message propagates backwards in the route, all TCP packets buffered at each node using the invalidated route are dropped. Figure 1(b) outlines the repair procedure and depicts the route breakage. As soon as the RREQ packet reaches node E (through F), a RREP is sent from the destination (once again, Node E could have generated the RREP packet but gratuitous RREPs are disabled in this scenario). At about the 39 secs mark the route is restored and the TCP agent resumes transmission. There are no subsequent route breakages until the end of the simulation and the topology stands as in Figure 1(c).

During this delivery effort, the TCP agent is unable to distinguish among the different causes of packet loss. Certain packet losses derive from the inability of the distributed MAC algorithm to properly coordinate packet transmissions among stations, which is mostly attributed to the exponential "waiting period" backoff of the 802.11 protocol as shown in [3], [17]. Further, the routing protocol's reaction to route breakages explicitly causes packet loss as each station processing the RERR message empties its queue buffer of packets utilising the invalid route. All such packet losses are interpreted by TCP as signs of congestion despite the fact that it does not occur at any point during the simulation (no node's buffer becomes full at any time). This misdiagnosis impedes TCP performance and

TABLE I

AODV PARAMETERS USED IN THE EXPERIMENTS

Parameter	Value	Parameter	Value
Exp. ring search	ON	TTL Start	2
Local Repair	NO	TTL Increment	2
Active Route Timeout	5 secs	LL feedback	NO
Gratuitous RREQ	OFF	HELLO interval	1 sec

its effects are further compounded by reliance on the coarse grained RTO timer to become aware of the route's restoration (there is no explicit cross-layer notification from the routing protocol of the route's status). Several research studies [1], [20] have shown that it is particularly desirable for TCP to maintain explicit awareness of the route's status, so that lost packets from route breakages do not spuriously activate congestion avoidance.

Figure 2(a) displays the DATA packet/ACK exchange (and thus, indirectly, the goodput) of a TCP Reno agent in the string topology scenario when the packet size is 1460 bytes. Each marked ACK point in the graph corresponds to a single ACK received at the source which acknowledges a range of packets. A value of 0 denotes a dupACK (since it does not acknowledge any new segments); a value of 1 denotes the normal TCP cycle since every ACK acknowledges a single additional segment (delayed ACKs are not used in our simulations); a value greater than 1 denotes that a packet which filled-in a discontinuous series of received segments packets at the destination's buffer was received and successfully acknowledged. The DATA Segment marks at the top of the graph indicate the times when a TCP DATA segment was launched by the sender. Of particular interest is the region at 35-39 secs (indicated by a grey box in Figure 2(a)) where the ACK flow stops since the route is considered invalid. The period of disconnection, that is the period when the routing protocol determines that the route has become invalid until it registers its restoration, is denoted by the two solid vertical lines in the graph. Further note the discrepancy between the time of the actual route failure (at 35s) and the time it takes for the routing protocol to detect it and initiate a new route discovery procedure (37.1s mark). Apart from a stray dupACK received before the RERR notification could propagate to the source, there is no newly ACKed traffic during that period.

An interesting interplay between the TCP's exponential RTO backoff can be observed in this case. The packets sent after the 35 secs mark as well as ACKs in flight are mostly lost. These losses cause TCP to retransmit at the 36 sec mark after experiencing an RTO. This retransmitted packet is lost on its way at node B, which by this time has received the RERR packet forwarded by node C. The new RTO timer backs off exponentially and is set to approximately 2 secs. Thus, it expires shortly after the 38 secs mark by which time the AODV agent at node A is aware of the route breakage and has already initiated the route discovery process. The subsequent retransmitted TCP packet (after the RTO) is *buffered* at the source node whilst the route discovery process finds a new route. A new route is discovered 70ms later (the discovery process started earlier than this) and the packet is launched 30 ms afterwards by the routing agent. If a subsequent RTO had occurred, say because the route became invalid once again, TCP's exponential back-off would have necessitated TCP to remain inactive for a longer period of time than before (approximately 4secs in this case) even though the route might have been repaired in the mean time. The lack of useful feedback between the routing (AODV) and the transport

agents is a well known problem in MANETs and has been discussed in previous work [4], [20]. However, no previous research mentions that, at a rudimentary level, the negative effect of consecutive RTOs does not take place if the TCP agent launches the packet after a route breakage has been detected by the routing protocol (as in the example mentioned above). In such a case, the buffering of the packet by the routing entity circumvents the damaging effects of consecutive RTOs if the packet and its accompanying ACK are successfully transmitted once the route has been restored.

Figure 2(b) shows the SRTT measurements as realised by the Reno TCP agent. The time frame for the route failure is denoted by a dashed box in the same graph. The RTT samples freeze for some time as the route is restored (denoted by the plateau at around 35-38 secs in the SRTT graph) point of route. After the route has been restored the new SRTT measurements are not significantly different to previous ones as the path is only extended by a single hop. It is noteworthy however that the measurements vary significantly which may not be ideal for delay or jitter sensitive applications.

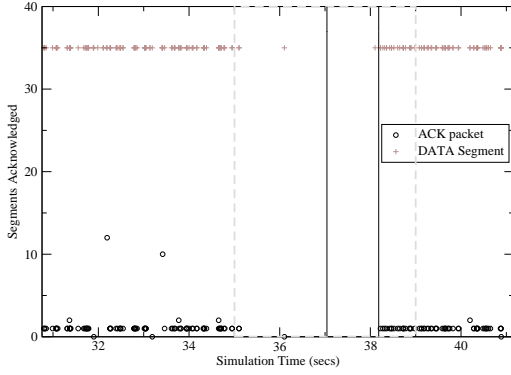
In the same figure, a graph of the average congestion window size is overlaid. Previous research has revealed that TCP does not behave optimally when used under distributed MAC mechanisms such as those employed by the 802.11 protocol. TCP stabilises at a large average cwnd which maintains more packets in the pipe than is optimal [7] for such distribution arbitration to function properly, especially when the path is long. For an IEEE 802.11 receiver the optimal window size for a topology of 4 hops would be 1 [2]; here an average cwnd of 6 segments is observed.

Finally, it is of particular note that during the course of the experiments packet loss is still evident, even for packets that are not broadcast (i.e. DATA packets, not just HELLO and RTS/CTS). This is surprising considering that interference is not evident in this scenario (due to the signal propagation model chosen) and still the distributed mechanism nevertheless fails to coordinate transmissions effectively.

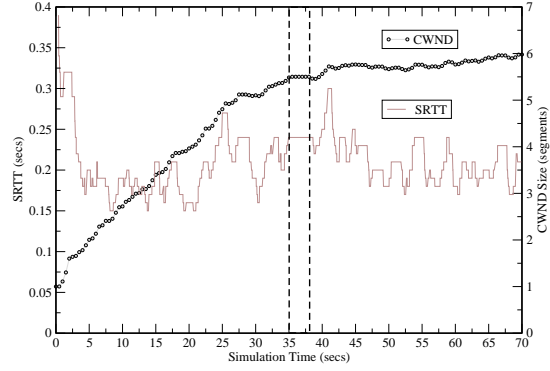
### III. SIMULATION SETUP

The general evaluation of the various TCP agents in the following section was conducted with the ns-2 simulator version 2.28 [6]. The AODV parameters are the same as the ones used in Section II. The signal propagation model used is the Two-Ray Ground model where signals propagate from sender to receiver in an open environment and over two possible paths; one by a direct ray and one that is reflected from the ground. Essentially, this model is representative of environments where a strong line of sight is present but ground reflections also influence path loss. This is the standard propagation model used in TCP evaluation over MANETs ([7], [18]), although there are several others such as Ricean/Rayleigh fading, Shadowing etc.

The simulation area chosen is a flat strip area of 1500x300m where 50 nodes are placed randomly. The mobility model used is the Random Waypoint Model with parameters set to reflect mobility ranging from walking to vehicular speeds. The exact



(a) ACKs acknowledged and data pkt send



(b) Smoothed RTT and CWND evolution

Fig. 2. Goodput and delay o

TABLE II  
SIMULATION PARAMETERS

Parameter	Value
Pause Times	0 (continuous motion)
Max. Node Speed	2, 5, 10, 15, 20 m/s
Mean Node Speed (0p)	2(1.44), 5(3.33), 10(6.62) m/sec 15(9.10), 20(13.78) m/sec
Simulation Time	900 secs
<b>TCP Parameter</b>	<b>Value</b>
Min. RTO	200ms
Max. RTO	60secs (RFC 2988)
RTO Timer Granularity	10ms (Linux kernel 2.4)
Delayed ACKs	disabled

simulation environment parameters are portrayed in Table II. Note that an increase of the setting of maximum speed in the mobility model is not indicative of a significant increase in the mean node speed as believed previously [9]. Research by J. Yoon et al. in [19] contains a thorough discussion of the issue as well as a proposed solution which is applied in the topologies used for our simulations. Thus, in this series of simulations the mean node speed increases in concert with the maximum node speed parameter, which in turn implies a real increase in mobility. Previous TCP evaluations over MANETs ([3], [4], [9]) have not considered this limitation of the Random Waypoint Model. The average mean node speed for the topologies used in our simulations are shown in Table II at rows 3 and 4 inside parenthesis next to the maximum node speed parameter.

For each simulation run a TCP connection is set between two randomly selected nodes and an FTP transfer session is initiated for the duration of the simulation. As such, there is a single source of TCP regulated traffic with an infinite backlog in the network. The TCP packet size is set to 1460 bytes and other parameters are set as in Table II. The performance metric is *goodput* which is defined as the number of packets successfully transmitted by the sender for which an ACK has been received. Retransmissions (spurious or otherwise) do not

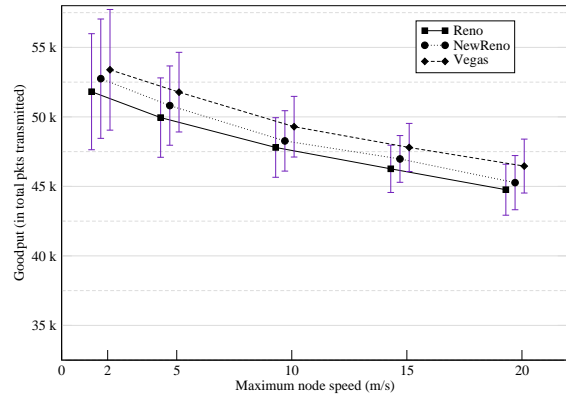


Fig. 3. Goodput against maximum node speed for different TCP agents with 0 sec pause time in strip (1500x1300m) topologies

contribute to the metric; each packet's contribution to the advancement of the sliding window is only measured once. For each pause time and maximum node speed combination the average goodput of 50 topologies is calculated and a 90% confidence interval for each is produced (shown as standard error bars in the relevant figures). The different TCP variants are analysed on the same topologies so as to ensure fairness and relevance of the results. A paired t-test is performed on the observations to denote if there are statistically significant differences in the performance of the TCP agents. The overall simulation (and connection) time is set to 900 seconds.

#### IV. SIMULATION RESULTS

The goodput results for the TCP agents under examination, namely Reno, NewReno, and Vegas, in strip topologies are contained in Figure IV as a function of goodput over maximum node speed under continuous mobility conditions. The error bars represent 90% confidence intervals.

In these experiments, TCP Vegas seems to perform significantly better than NewReno; however, the difference could potentially diminish if the agent is adversely affected by other

factors, such as the interaction with other traffic in the network. Nonetheless, the results presented here confirm Vegas' competent performance (on par with NewReno) under various mobility conditions when interaction with other traffic is not taken into account. This expands on previous work comparing Vegas with Reno [13] and reveals that Vegas' performance merits are equivalent or more pronounced than NewReno's when single connections are considered. In our experiments Vegas had consistently higher goodput than NewReno (2-4%) and Reno (3-8%). TCP Reno was a significantly worse performer than the other two variants and that observation held true for all mobility conditions. The detailed results of the t-test are omitted here due to lack of space.

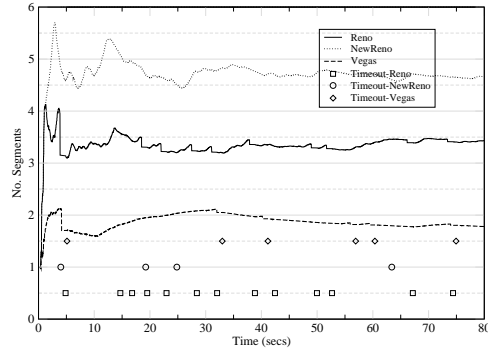
The usage of strip (1500x300m) topologies in our experiments, is a decision widely shared in TCP MANET investigations in the past ([3], [17]) under the assumption that such an area encourages the formation of longer paths than a square one. Even though the endpoints of the simulated transfer are randomly chosen, the average path length is, in the case of our experiments 2.5 hops. This is observed throughout the simulation runs, i.e. across 250 topologies (50 topologies for each of the 5 combinations of max. speed setting). Since recent research has shown that TCP under-performs due to interference which is particularly evident for routes  $\geq 3$  hops ([3], [14]), it becomes an interesting future prospect to examine the behaviour of the aforementioned TCP agents under longer routes.

We have also conducted experiments with the same TCP and AODV parameters in a 1000x1000 strip simulation area. The motivation is that such area has also been popular in previous research works [9]. The derived conclusions are identical to the ones observed for the strip (1500x300m) topology and are omitted here for brevity. The average path size of that series of experiments was 2.45 hops which implies that the hop length conditions were similar to those of the square area topologies. It might be inferred that if future research is to concentrate on TCP performance over long paths, communicating pairs would have to be carefully chosen to be consistently far apart.

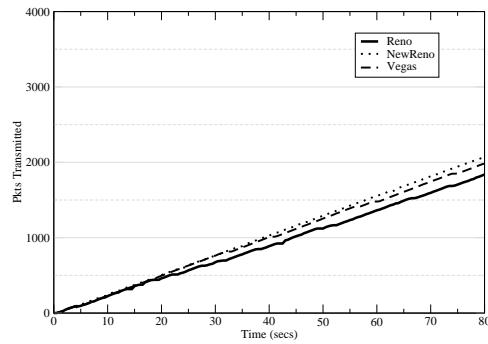
All TCP variants present the common trait of decreasing performance as mobility increases. This corroborates the observations made by previous work [9] and is attributed at TCP's inability to distinguish between packet loss causes. In previous work we have shown that the congestion window of TCP Vegas in MANETs is maintained at a lower value than that of Reno controlled variants because of the proactive nature of Vegas' congestion avoidance mechanism [14]. This accounts in part for Vegas' competent goodput behaviour. TCP NewReno's overall better performance over Reno may be accounted by the former's ability to "plug-in" multiple missing segments in a congestion window's worth of data without resulting to the coarse grained RTO timer like Reno. A more thorough examination of the TCP agents' behaviour in a specific short path scenario is included in the next section.

## V. ANALYSIS OF SHORT PATH BEHAVIOUR

This segment describes in the behaviour of the three protocols along a small connection path which provides insight on the overall performance behaviour observed during simulations.



(a) Running avg of packets in flight per TCP agent



(b) Total segments transmitted over time for each TCP agent

**Fig. 4.** Traces of the first 80 secs of a single TCP connection in a 5-hop string topology

First, assume a string topology of 5 hops with an initiated end-to-end connection. Figure 4(a) shows the running average of the number of packets in flight during the first 80 secs of the connection for Reno, NewReno and Vegas. It also denotes the instances where a retransmission timeout occurred for each agent. These RTOs were caused by multiple packet losses due to the effects of interference [14] (there is no mobility in the static string topology). Figure 4(b) shows the total no. of segments transmitted for each TCP agent over connection time. By the end of the 80 secs, TCP Reno has transmitted 10% fewer packets than NewReno or Vegas. This is due to the diminishing effects of RTOs which result in the under-utilisation of the path.

In this example, TCP Reno experiences 13 RTOs, NewReno 4 and Vegas 6. Even though TCP NewReno maintains more packets in the pipe than NewReno and Vegas, its ability to recover from multiple losses within the same congestion

window allows it to avoid multiple or consecutive RTOs. Specifically, TCP Vegas, NewReno and Reno spend 7, 4 and 13 seconds in RTO inactivity. The minimum RTO is set in this case to 200ms; several implementations set this to 1 sec adhering to the original RFC [10] which aggravates the impact of single or consecutive RTOs.

The above observation largely accounts for the performance discrepancy between the three TCP variants. Whenever low mobility is observed the paths can be considered relatively stable and thus the above behaviour largely applies. In cases of high mobility it can be said that with respect to the reaction to route breakages all three variants behave similarly; packet losses are treated as congestion losses and each one uses the same exponential back off mechanism as a response.

## VI. CONCLUSION

This research has examined the performance of three TCP variants over dynamic topologies in MANETs. The discussion included detailed traces of a route breakage scenario, studied the effect on TCP throughput and provided insight on the interplay of route buffering and the transport agent. Further, TCP Reno, NewReno and Vegas were evaluated in dynamic topologies over square and strip simulation areas. The performance merit of TCP Vegas in terms of throughput was noted as well as the almost equivalent effectiveness of TCP NewReno. Reno was shown conclusively to be the worst performer under all mobility conditions. An explanation of discrepancy of the TCP agents' performance over a stable path was provided.

Future research prospects include the quantification of the causes of packet loss and the examination of aggregate multiple flow behaviour. It may be of particular interest to explore to what degree hidden terminal effects, or other MAC layer incompatibilities are responsible for the degradation in TCP throughput, compared to the decline caused by route breakages. Finally, there is significant value in examining the behaviour of multiple TCP flows for different TCP agents; however, such an examination should be accompanied by some metric of interaction between the flows in order for the results to be meaningful.

## REFERENCES

- [1] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash. A feedback based scheme for improving TCP performance in ad-hoc wireless networks. In *Proceedings of the 18th annual International Conference on Distributed Computing Systems*, pages 472–479, May 1998.
- [2] K. Chen, Y. Xue, and K. Nahrstedt. On Setting TCP's Congestion Window Limit in Mobile Ad Hoc Networks. In *Proceedings of the 38th annual IEEE International Conference on Communications ICC 2003*, pages 1080–1084, May 2003.
- [3] K. Chen, Y. Xue, S. H. Shah, and K. Nahrstedt. Understanding bandwidth-delay product in mobile ad hoc networks. *Computer Communications*, 27(10):923–934, June 2004.
- [4] T. D. Dyer and R. V. Boppana. A comparison of TCP performance over three routing protocols for mobile ad hoc networks. In *Proceedings of the 2001 ACM International Symposium on Mobile ad hoc networking & computing*, pages 56–66. ACM Press, 2001.
- [5] H. Elaarag. Improving TCP performance over mobile networks. *ACM Computing Surveys (CSUR)*, 34(3):357–374, 2002.
- [6] K. Fall and K. Varadhan. The ns manual - the VINT project. <http://www.isi.edu/nsnam/ns/ns-documentation.html>.

- [7] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. The impact of multihop wireless channel on TCP throughput and loss. In *Proceedings of Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, volume 3, pages 1744–1753, March 2003.
- [8] L. A. Grieco and S. Mascolo. Performance evaluation and comparison of westwood+, new reno, and vegas tcp congestion control. *SIGCOMM Comput. Commun. Rev.*, 34(2):25–38, 2004.
- [9] G. Holland and N. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 219–230. ACM Press, 1999.
- [10] U. o. S. C. Information Sciences Institute. *Transmission Control Protocol*. Internet Draft, <http://www.ietf.org/rfc/rfc793.txt>, September 1981. Request For Comments.
- [11] A. Kamerman and L. Monteban. WaveLAN II: A high-performance wireless LAN for unlicensed band. *Bell Labs Technical Journal*, 2(3):118–133, Summer 1997.
- [12] H. Lundgren, E. Nordstr&#246;, and C. Tschudin. Coping with communication gray zones in iee 802.11b based ad hoc networks. In *WOWMOM '02: Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*, pages 49–55. ACM Press, 2002.
- [13] S. Papanastasiou and M. Ould-Khaoua. Exploring the performance of TCP Vegas in Mobile Ad hoc Networks. *International Journal of Communication Systems*, 17(2):163–177, 2004.
- [14] S. Papanastasiou and M. Ould-Khaoua. TCP Congestion window evolution and spatial reuse in MANETs. *Journal of Wireless Communications and Mobile Computing*, 4(6):669–682, 2004.
- [15] 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.
- [16] V. Ramarathinam and M. A. Labrador. Performance Analysis of TCP over Static Ad Hoc Wireless Networks. In *Proceedings of the 12th International Conference on Parallel and Distributed Computing Systems (PDCS-2000)*, pages 410–415. ACTA Press, 2000.
- [17] K. Xu, M. Gerla, L. Qi, and Y. Shu. Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 16–28. ACM Press, 2003.
- [18] S. Xu and T. Saadawi. Performance evaluation of TCP algorithms in multi-hop wireless packet networks. *Wireless Communications and Mobile Computing*, 2(1):85–100, March 2002.
- [19] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proceedings of Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, volume 2, pages 1312–1321, March 2003.
- [20] X. Yu. Improving TCP performance over mobile ad hoc networks by exploiting cross-layer information awareness. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 231–244. ACM Press, 2004.