On the interaction of TCP and Routing Protocols in MANETs

S. Papanastasiou, L. M. Mackenzie, M. Ould-Khaoua
Department of Computing Science
University of Glasgow
Glasgow G12 8RZ, UK
{stelios, lewis, mohamed}@dcs.gla.ac.uk

Vassilis Charissis
Digital Design Studio
Glasgow School of Art
Glasgow G41 5BW, UK
v.charissis@gsa.ac.uk

Abstract

Recent research in mobile ad hoc networks (MANETs) has highlighted the detrimental effect of multiple retransmission timeouts (RTOs) on TCP throughput. However, the effect of the routing protocol's buffering strategy on this phenomenon has not been explicitly demonstrated or studied. In this paper, we present and analyse through simulations using functional testbed routing agents, TCP throughput performance during a route break event. The results of our analysis on three popular routing protocols, namely AODV, DSR and OLSR, produces insight into the different behavioural patterns of TCP during this event, and highlights the mechanisms of each routing protocol that affect it. Further, trade-offs in the choice of the routing parameters with respect to TCP performance are discussed.

1 Introduction

In its various forms, TCP has been widely used over the Internet and has become ubiquitous both in terms of application scope and popularity. Throughout its lifetime the TCP standard has been subject to several modifications and refinements that represent trade-offs involving implementation complexity, backwards compatibility and throughput performance. Due to its extensive use and implementation maturity on most platforms, TCP has become the focal point of much research work in MANETs, especially with regard to the effects of non-congestion related packet losses [2].

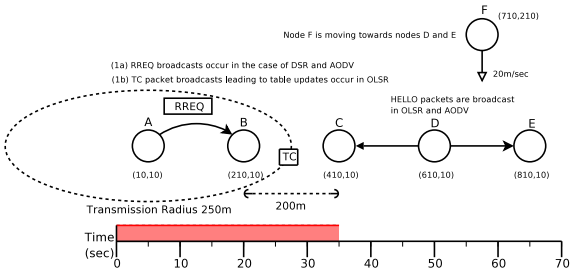
However, although previous TCP research has identified the leading cause of TCP's underwhelming performance in wireless multihop networks to be consecutive RTOs [3] experienced due to packet loss misinterpretation, it has not illustrated how this effect is handled by different routing protocols. Further, most previous simulation studies have used routing agents which do not implement the full RFC specification and simplify certain functionality in the agent itself, such as packet buffering, which may influence TCP performance in a significant way.

Motivated by the above observations, this paper presents a study on the effects of mobility on TCP, through its interplay with three popular routing mechanisms, namely AODV, DSR and OLSR. Such an approach provides insight on the potential performance discrepancies between routing agents and, more significantly, outlines the trade-offs involved in enabling optional features included in each routing protocol. In contrast to previous work [3], detailed traces are used to pinpoint the causes of the performance penalty incurred by the TCP agent with respect to inactivity periods and reaction to packet losses. Real implementations of the routing agents are used to drive the simulation in order to ensure correspondence with real-life testbeds [1].

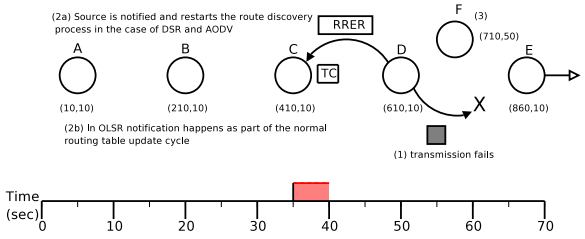
The rest of this paper is organised as follows. The next section presents the simulation scenario setup which remains in focus throughout this study. Sections 3, 4 and 5 present the interaction of TCP with the AODV, DSR and OLSR protocols respectively, during the route breakage scenario. The implications of the choice of the various routing parameters with respect to TCP performance are discussed and accounted for in the same section. Finally, Section 6 summarises the conclusions of this study and offers suggestions for future work.

2 Scenario setup and TCP parameters

The simulation scenario used throughout this study involves a simple five node string (or chain) topology (nodes $A \rightarrow E$) and an additional node, F, which stands in close proximity between nodes D and E, as shown in Figure 1(a). The nodes along the



(a) Initial Topology



(b) Route breakage

Figure 1. The scenario used for TCP analysis

Table 1. AODV parameters used

Parameter	Value	Parameter	Value
Exp. ring search	ON	TTL Start	2
Local Repair	OFF	TTL Incr.	2
Active Route TO	5s	LL feedback	OFF
Grat. RREQ	OFF	HELLO int.	1s

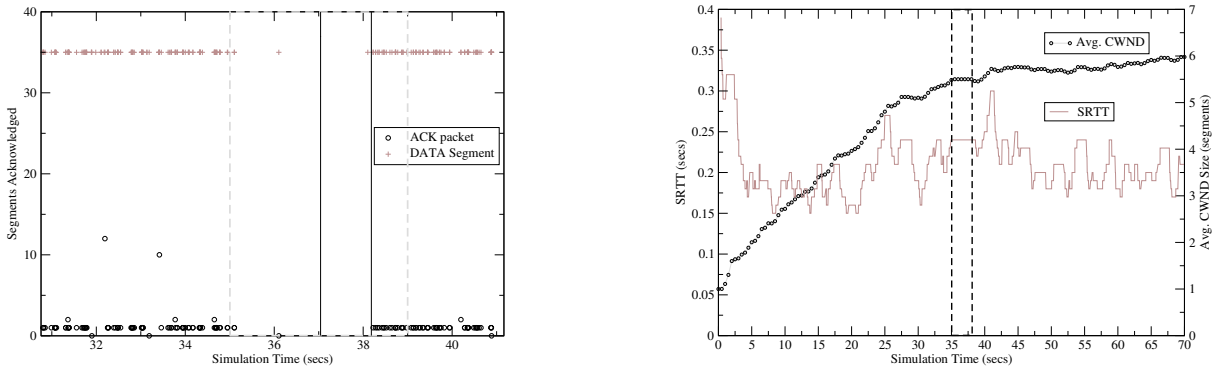
$A \rightarrow E$ string are spaced 200m apart and feature 2Mbps transceivers which nominally represent a good balance of bandwidth and transmission range. In this scenario, the transmission range of the transceivers is fixed at 250m using a flat, ideal signal propagation model which does not account for attenuation up to the transmission range limit and nullifies the signal strength beyond that threshold. Obviously, some signal degradation is always present in real life scenarios but this model is utilised here so as to isolate the effects of route breakage and disregard other effects such as those caused by interference and hidden terminals [2]. The simulation scenario lasts 100 seconds and throughout that time an FTP bulk transfer with an infinite backlog is established between the end points of the string topology, namely nodes A and E. The routing agents installed are, in turn, AODV-UU [6], DSR-UU [7] and OOLSR [5]. All three are working testbed reference implementations and are interfaced with the ns-2 simulator. The TCP agent used is TCP Reno. Note that our discussion holds true in the case of NewReno and SACK, however, Reno is chosen for demonstration purposes due to its simplicity and its role as a baseline for TCP performance studies.

The scenario proceeds as follows; Initially, the complete path is set-up in the form of the $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$ route and the bulk data transfer begins. Route discovery and maintenance occurs differently for each routing protocol; however, details are omitted here due to lack of space but can be found in the relevant RFCs. After a few seconds and once the bulk transfer has been initiated, node F moves closer and gets stationed between nodes D and E, thus getting well established within both nodes' transmission radii. At the 30 secs 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 the link becomes invalid. At its new destination node E is still a neighbour of node F but cannot be contacted by node D. This situation is exhibited in Figure 1(b). Eventually, a new route (through node F) is utilised to facilitate $A \rightarrow E$ communications.

HELLO packets are used to detect route failures in AODV and OLSR, whilst network layer ACKs are employed for that purpose in DSR. Link layer feedback, in all cases, is not considered. The intention is to avoid false route breakage notifications that occur when packets are dropped due to MAC layer miscoordination [2]. However, this is a loose assumption; our conclusions hold in the case of link layer feedback as well but further results are not included here due to space limitations.

3 TCP and AODV

First, the AODV protocol is utilised as the routing agent, in the simple scenario described above. Its parameters are set as in Table 1. Figure 2(a) displays the DATA packet/ACK exchange (and thus, indirectly, the throughput) of the TCP agent in the string topology scenario. 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



(a) ACKs received and data pkts sent

(b) Smoothed RTT and CWND evolution

Figure 2. Throughput and delay of TCP Reno over AODV during a route break

(delayed ACKs are not used in this simulation). 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 became 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 35 secs) and the time it takes for the routing protocol to detect it and initiate a new route discovery procedure (37.1 secs mark). The delay is attributed to the absence of link-layer feedback and the use of HELLO packets, which represents a trade-off between frequency of updates and overhead. Apart from a stray dupACK received before the RERR notification could propagate to the source, there is no newly ACKed traffic during that period.

Also, an interesting interplay between the TCP's exponential RTO backoff and the routing mechanism can be observed. The packets sent after the 35 secs mark as well as ACKs in flight are mostly lost. These losses cause TCP to retransmit at 36 secs 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 then 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 (by the discovery process which had started earlier) 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 backoff would have necessitated TCP to remain inactive for a longer period of time than before (approximately 4 secs in this case) even though the route might have been repaired in the meantime. The lack of useful feedback between the routing and transport agents is a well known problem in MANETs and has been discussed in previous work [3]. 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 smoothed round trip time (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 being restored (denoted by the plateau at around 35-38 secs in the SRTT graph). 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 (cwnd) 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 [4] 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 a single segment [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 flat signal propagation model chosen) and the distributed mechanism nevertheless fails to coordinate transmissions effectively. Previous work has noted such losses when interference is present [2] but we confirm the phenomenon in a flat signal propagation scenario.

4 TCP and DSR

This section discusses the same scenario with a DSR agent in place at participating nodes. All other parameters are identical to the previous Section. As a replacement to link layer feedback, network layer ACKs are utilised. These are part of the draft specification and may be demanded by the source. Such network layer functionality makes DSR deployable across systems that do not feature equivalent lower layer functionality.

Passive ACKs occur between the two entities of a point-to-point dyad, when the packet originator overhears the neighbouring destination forwarding the packet after sometime. This implies that the destination had correctly received the packet earlier, but on the downside, it requires omnidirectional antennas and both source and destination to be within transmission range of each other. Further, it is possible for network layer feedback to be complemented by passive ACKs. In particular, the node may attempt several times to first transmit and then wait for a passive ACK before requesting a network layer ACK, thus avoiding the network layer ACK overhead. The list of DSR parameters set for this simulation scenario is presented in Table 2.

The focal point of interest, again, is the route break that happens at 35 secs. In particular, at 35.9 secs and

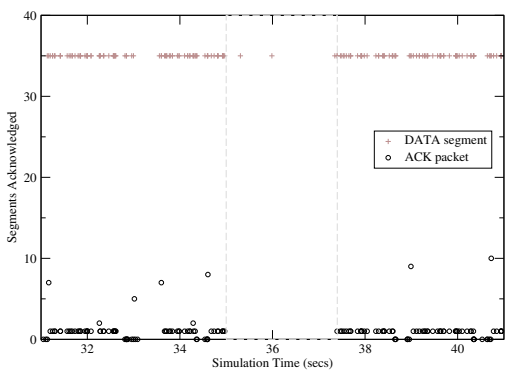
Table 2. DSR parameters used

Parameter	Value	Parameter	Value
Passive ACKs	ON	Flowstate	OFF
Promiscuous Listening	ON	Send. Buffer Lifetime	30s
Salvage pkts	ON	Alt. Routes	ON
Snoop routes	ON	LL feedback	OFF

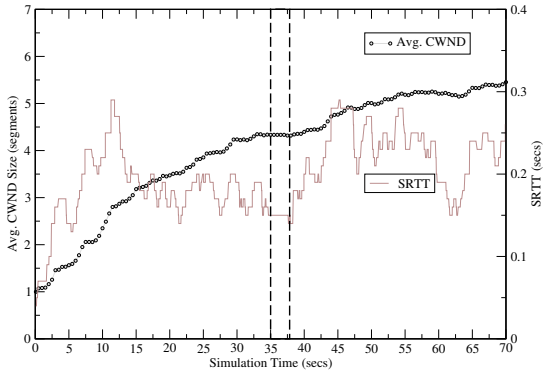
after several failed attempts to receive a network layer ACK, the routing agent at node D, realises that the link has been invalidated. Then, it produces a RERR towards the source (node A) but, unlike AODV, does not drop packets. Instead, all the packets making use of the invalidated link are placed in the maintenance buffer and the routing agent consults its routing cache and discovers that node F is a neighbour of node E; i.e. there is an potential alternative route. The information on this route was obtained at around 1 sec time when due to the MAC layer's inability to coordinate transmissions, the network layer ACKs between D and E failed to be transmitted and node E erroneously believed its link to node D to have become invalid. So, at E's subsequent discovery process for a route to A, both nodes D and F responded. Node E, then opted to utilise the path proposed by node D, which offered the shortest route, but as a side-effect node D became aware of node F's neighbouring status to node E.

Subsequently, at link breakage time there is no need for node D to initiate route discovery to eventually "salvage" the packets in its buffer that had their route invalidated by the obsolete $D \rightarrow E$ link. Instead, node D makes use of the alternate route through F by replacing the old path in the packets header with the new one. Node E also realises at 36.7 secs that the link to D has been severed, but shortly after (at 37.2 secs) receives the rerouted packet from node F and becomes aware of the new route (A, B, C, D, F, E). This use of route caching results is beneficial; in Figure 3(a), the area surrounded by the dashed box represents the time needed for the route to be re-established (approx. 2 secs) which is lower than the one for AODV in the previous Section (approx. 3 secs). More importantly, the TCP agent does not experience subsequent RTOs; the route discovery and salvaging operation is fast enough for TCP to exhibit only one RTO backoff. The caching of packets in the maintenance buffer during the route break as well as their subsequent forwarding also helps avoid consecutive RTOs.

The effect on TCP's SRTT estimator as well as the average cwnd size is shown in Figure 3(b). As compared to AODV, the plateau in the SRTT graph (denoted by the black dashed box) shortly after the 35



(a) ACKs received and data pkts sent



(b) Smoothed RTT and CWND evolution

Figure 3. Goodput and delay of TCP Reno over DSR during a route break

secs mark is not as noticeable, as TCP inactivity due to RTO backoffs does not last as long. Note that although in this case the use of cached routes is beneficial, the utilisation of a stale route would have had the opposite effect; the agent would send the packet along a non-existent path and would have to wait for a period of time before realising that the path was an invalid one, which could lead to consecutive RTOs at the source.

During the 35-39 secs time frame, the RERR response as was originally produced at approx. 36 secs by node E (after the route break) is propagated toward node A and causes the intermediate nodes to place packets making use of the invalidated $D \rightarrow E$ link in their respective maintenance buffers. When node D makes use of the alternate route (via node F), node C, which has already propagated the RERR on its way to the source, overhears the transmission and learns about the new route. When node A receives the RERR packet, it re-initiates the route discovery process and receives a gratuitous reply from node C. This 'domino' effect of link information updates accounts for the quick route restoration.

5 TCP and OLSR

This section contains a description of TCP's behaviour over OLSR as exhibited over the same scenario depicted in the previous two sections. The particular parameters used in this scenario are depicted in Table 3 and are the defaults set by the reference implementation [5] according to the RFC.

Initially, and during the first few seconds of the setup, HELLO messages are broadcast from each node,

Table 3. OLSR parameters used

Parameter	Value	Parameter	Value
HELLO interval	1s	TC interval	5s
Willing to route	ALL	Max. Jitter	250ms
Hysteresis	OFF	MPR Cover.	1
Neigh. Hold	6s	Refresh int.	2s

which declare their immediate neighbours. After the first exchange of these messages, subsequent broadcasts also include two hop neighbour information. Hence, after a short time interval, each node maintains enough information to declare a set of Multipoint relays (MPRs) which cover its two hop neighbours and which is advertised using Topology Control (TC) packets, which are distributed network wide. In this case, the MPR set of each node contains only a single neighbour, as only one is necessary to reach all nodes within a two hop radius. A notable exception is node C which has two nodes in its MPR set, namely B and D.

Eventually, and at the 35 secs mark, a link failure occurs between node D and E, due to E's movement away from its neighbour. Essentially, the absence of HELLO packets is noted in the given refresh interval (2 secs) so the route $D \rightarrow E$ is determined to be invalid, by the routing agents of both D and E. Therefore, packets utilising the route in D and E are dropped which leads to consecutive RTOs at the TCP agent, as the route is not restored fast enough for incoming packets to activate the dupACK heuristic. In particular, TCP experiences the first RTO at 35.6 secs and then consecutive ones at 36.9, 39.5 and 44 secs. The DATA-ACK exchange during that time period is shown in Figure 4(a), where the isolated DATA transmissions shown corre-

spond to packets launches triggered by RTOs. Specifically, at the 35.6, 36.9 and 39.5 time marks, the DATA segments transmitted are not followed by an ACK response as they are lost upon transmission from node D to E. These particular losses are link layer losses; delivery is attempted but there is no lower level MAC-ACK response from the destination (E), as it has moved outside D's transmission radius. The unACKed DATA packet launched at about 44 secs, is discarded by node C which has yet to discover through TC exchanges the new route through node F.

The long TCP inactivity period, as denoted by the grey box between 35-55 secs in Figure 4(a), corresponds to TCP inactivity noted immediately after the route failure and is due to three reasons. First, failed transmissions are realised with the granularity of the refresh interval (2 secs, or the equivalent of two HELLO packet launch cycles), which leads to consecutive RTO's as packets are lost in failed MAC transmissions. Second, the route restoration period which qhappens with the dissemination of TC packets has by default a coarse granularity so that several TC transmissions may be "bundled" together to avoid excessive overhead as discussed below. Third, the lack of packet caching compounds the RTO issue, as ongoing TCP transmissions end up in packet drops and cause further timeouts until a new route is found. Even upon the route's restoration, TCP's RTO timer has to expire before a new "probing" packet is launched, as TCP is unaware that previous packet losses were due to link failure and attributes them to congestion.

Figure 4(b) denotes the smoothed RTT and CWND evolution experienced by the TCP agent. The time frame of inactivity due to the route failure is denoted by the dashed box (35-55 secs). The plateaus in the CWND and RTT diagrams are noticeably larger than the ones for AODV and DSR in the previous sections, as the transfer of data stalls for a longer period of time.

This long period of inactivity is due to the large TC update time interval (set to a default 5 secs) which in turn means that information on invalid links take several seconds to propagate. The OLSR RFC makes provisions for an immediate TC message launch mechanism which allows generation of TC messages as soon as the node's neighbourhood changes. However, since these transmissions are broadcast network-wide, there are significant overhead savings if they are "bundled" together, which makes delaying them desirable. The trade-off to consider is thus overhead versus lower notification delay.

To illustrate the above point more aptly we have conducted the same experiment with reduced TC transmission interval and immediate notifications (i.e. no "bundling" of TC segments). Figure 4(a) shows the total no. of TCP segments acknowledged when the TC update interval changes from 5 to 3 secs and when immediate TC updates are used. When there is a decrease in the update interval or immediate TCP updates are allowed, consecutive RTO's are avoided and total time spent in RTO backoff is reduced (as shown in Table 4). Note that the default update period of 5 secs radically under-performs in this case with noticeably longer periods of TCP inactivity (as denoted by the flat line segments of the graph in Figure 4(a)).

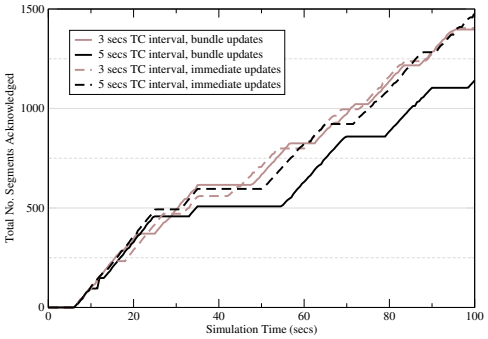
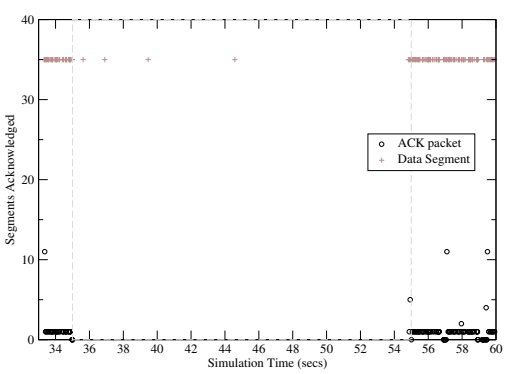


Figure 5. Total no. of TCP segments acknowledged over simulation time

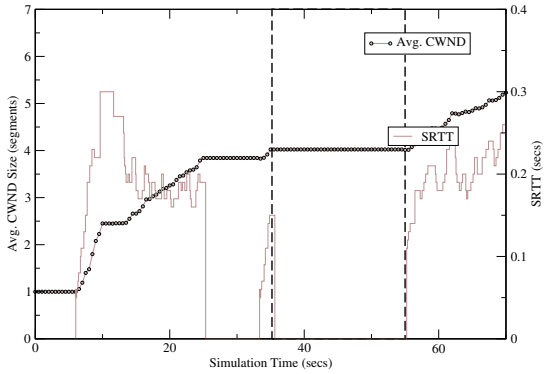
Table 4. Time spent in RTO for different OLSR TC transmission intervals and update methods

OLSR parametera	Time in RTO
5s TC interval - bundle	55.57s
5s TC interval - immediate	41.07s
3s TC interval - bundle	41.26s
3s TC interval - immediate	40.38s

Also note that since the reference implementation does not cache packets, unlike DSR which features a maintenance buffer, there is no forwarding of "salvaged packets" which could help avoid consecutive RTOs. Finally, the two other plateaus in Figure 4(b) merit some explanation. The first one at 0-6 secs is due to the startup period needed for the HELLO packets and TC exchanges to take place and setup the route. OLSR needs this warm-up period (unlike DSR and AODV which immediately start a short route discovery phase), and TCP activity does not occur until a valid path is discovered. The latter plateau at 26-33 secs is due to dropped HELLO messages due to mis-coordination of the MAC protocol. This phenomenon has been observed in the other two routing agents but the slower



(a) ACKs received and data pkts sent



(b) Smoothed RTT and CWND evolution

Figure 4. Throughput and delay of TCP Reno over OLSR during a route break

route restoration and lack of packet caching of OLSR causes TCP to under utilise the route for longer in such occurrences. Note however, that OLSR is designed to be used in dense networks where multiple flows are the norm rather than the exception and hence the default parameters are set to reflect such an environment. This above comments are not a critique on the choice of parameters, but an observation on their poor reflection on TCP performance in this scenario.

6 Conclusions

This paper has presented an overview of TCP behaviour in a simple mobility scenario under three popular routing protocols in MANETs, namely AODV, DSR and OLSR. The subsequent study of traces, has revealed that DSR interacts with TCP more efficiently than the other two protocols as it salvages packets and restores the route quickly in the examined topology. AODV was shown to behave competently by avoiding RTOs through caching at the source of outgoing TCP packets when it has been informed of that a route breakage has occurred. Finally, OLSR, by default, maintains sub-optimal parameters for this scenario type as it tuned for significant traffic loads over dense topologies. We have shown how a simple parameter tweak may improve OLSR's performance in this special case.

In the future, we plan to examine how consecutive RTOs affect TCP behaviour in general mobility scenarios. In particular, we aim to identify if different packet caching and route restoration mechanisms exhibited by routing protocols, affect TCP performance in significant way for a variety of topological conditions. Finally, as the routing agents used in this simulation study are actual test-bed ready implementations, we are in the process of verifying our simulation results in a real life environment and thus validate our conclusions.

References

[1] G. Anastasi, E. Borgia, M. Conti, and E. Grego. IEEE 802.11 Ad Hoc Networks: Performance Measurements. In *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03)*, pages 758–763, May 2003.

[2] 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.

[3] 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.

[4] 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.

[5] HIPERCOM Project. OOLSR. http://hipercom.inria.fr/OOLSR/.

[6] H. Lundgren, E. Nordström, and C. Tschudin. Coping with communication gray zones in ieee 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.

[7] E. Nordström. DSR-UU: A Dynamic Source Routing protocol implementation. http://core.it.uu.se/AdHoc/DsrUUImpl.