

A Route-Control Mechanism for Improving the Performance of Transport Protocols in a MANET

Reuven Cohen and Anna Levin
Department of Computer Science
Technion, Israel

Abstract— We propose a new cross-layer mechanism, referred to as *Route-control*, for mobile ad-hoc networks (MANETs). This mechanism, which works in the Network and Transport layers, aims at enhancing the performance of MANETs' reliable Transport protocols. The main idea behind the proposed mechanism is to notify the sender when the packets of a Transport layer flow change their route. We show that the sender can benefit from this information when deciding whether to retransmit a missing segment or to wait, when estimating the RTT (Round Trip Time), and when deciding whether to change the congestion window.

I. INTRODUCTION

We consider Mobile Ad-hoc NETWORKS (MANETs) with hop-by-hop forwarding strategy, such as AODV [1], OLSR [2] and TBRPF [3]. Such networks are known to introduce a lot of challenges not only in the PHY and MAC layers, where schemes for efficient wireless communication are needed, but also in the Network and Transport layers, where protocols that work well in spite of node mobility are sought.

Researchers working on Transport layer issues in MANETs agree that running TCP in such networks is inefficient, and that new protocols are needed (e.g., [4], [5], [6], [7], [8], [9], [10], [11]). These protocols should be based on full separation between congestion control and loss control. The loss control sublayer sits on top of the congestion control sublayer. It decides when a new segment has to be sent or when an old one has to be re-sent. In both cases it submits the segment to the congestion control sublayer, responsible for deciding when the packet should actually be transmitted in accordance with the congestion control rules. Unlike in TCP, the loss of a segment will not necessarily affect the congestion control sublayer. The loss control sublayer should be based on selective repeat, as in TCP SACK [12]. This is required in order not to retransmit segments that are correctly received by the receiver.

In this paper we do not propose a new Transport protocol, but a new cross-layer mechanism, referred to as *Route-control*, which works in the Network and Transport layers. This mechanism enhances the performance of MANETs' reliable Transport protocols. The main idea behind the proposed mechanism is to notify the sender when the packets of the considered flow follow the same route and when they switch to a new route. The sender benefits from this information in several ways:

- 1) The sender will know whether to retransmit a missing segment or to wait because a packet carrying this segment still might be correctly received. If a segment is missing and the route is stable, the sender retransmits

it as soon as it gets a single ACK for a succeeding segment. This concept reduces the dependency on the receipt of duplicate ACKs, avoiding inefficient time-outs.

- 2) Its RTT estimation is enhanced. RTT estimation in MANETs is difficult due to frequent route changes and because it is difficult for the sender to know whether a change is due to mobility – in which case old information about the estimated RTT should be ignored – or due to emerging congestion – in which case old information should be taken into account using the moving average concept [13]. With *Route-control*, the sender knows **exactly** when the flow changes route, and it can immediately ignore an old RTT estimation.
- 3) Its response to congestion can be significantly enhanced, whether congestion control is based on the receipt of ACKs, as in the traditional TCP [14], [15], on explicit congestion notification [16], or on equations computed by every source [17], [18]. All these schemes are affected by the change in a flow route. With the proposed mechanism, the sender can ignore possible effects of route changes on the specific congestion control scheme.

The proposed mechanism is based on the cooperation of the internal nodes. These nodes need to signal to the receiver when they discover that a packet for a given source-destination pair is forwarded on a new route. This information is then signaled by the receiver to the sender in the Transport layer header, in the same way that congestion is signaled using the ECN mechanism [16]. No assumption is made regarding the underlying routing protocol. However, the mechanism is more effective for MANET's routing protocols that use hop-by-hop forwarding [1], [2], [3], and less effective for those using source-route forwarding [19], [20]. We also make no assumption about the Transport layer protocol. We show, however, how such a protocol can take advantage of our mechanism with respect to loss control, congestion control and RTT computation.

The rest of the paper is organized as follows. Section II presents related work. Section III presents the proposed mechanism, which contains three algorithms: one for the sender, one for the receiver, and one for every intermediate node. Section IV presents a selective repeat ARQ protocol using the proposed *Route-control* mechanism. In this section (Theorem 1) we formally prove one of the most important properties of the new mechanism: that the sender can decide when a single duplicate ACK is a sufficient indication of a lost seg-

ment and when it is not. Section V shows how the Route-control mechanism can improve both RTT computation and congestion control. Section VI presents our simulation study and Section VII concludes the paper.

II. RELATED WORK

The closest paper to ours is probably [10]. The authors propose a Transport layer approach to improve TCP performance by detecting and responding to out-of-order packet delivery events. The receipt of an out-of-order packet can be detected in two ways. The first is to use the TCP timestamp option. The second is to use a new field in the Transport header, referred to as a TCP packet sequence number. This counter is incremented by one whenever a segment is transmitted, even if this segment has already been transmitted in the past. The mechanism proposed in [10] has the advantage of being implemented in the Transport layer, whereas our mechanism is implemented in the Network layer and requires the support of the intermediate nodes. However, our Network layer mechanism has the important advantage of providing the Transport protocol with much more information than the one in [10]:

- 1) First, our mechanism informs the Transport layer that a new route is being used even when the change does not result in out-of-order routing. This is important because, as discussed in Section I, the Transport layer may have to initiate several actions in this case, such as resetting the congestion window and the round trip time estimate.
- 2) Second, our mechanism can be used to speed up the retransmission of a lost segment in case of stable routes. This improvement of the Transport protocol's performance, discussed in Section IV, is impossible with the mechanism proposed by [10], as it does not provide necessary information.

In several papers the authors propose schemes to be used by the intermediate node to inform the sender of route failure [4], [5]. With Explicit Link Failure Notification (ELFN) [5], a node that detects a link failure notifies the TCP sender. The sender then freezes its retransmission timer and periodically sends a probing packet until it receives an ACK from the destination, in which case it restores its retransmission timer and continues as normal. In [4], a feedback-based scheme called TCP-Feedback is proposed. With this scheme an intermediate node that detects a broken link sends a route failure notification (RFN) to the TCP sender. The sender freezes its state and resumes transmission only when it receives a route reestablishment notification (RRN) from the intermediate node. However, these schemes are orthogonal to the Route-control mechanism proposed in this paper because they are designed for routing protocols such as DSR [20], [19], whose forwarding strategy is source-route. In contrast, our mechanism is designed for routing protocols with hop-by-hop forwarding strategy, such as AODV [1], OLSR [2] and TBRPF [3]. In these protocols, the source is not aware of the route change, but only the intermediate nodes must know about the change in the link they use. Another difference is that the goal of [4], [5] is to improve congestion control, whereas our mechanism has an

additional goal: to improve the performance of the Transport protocol by giving the sender an early indication of the loss of a segment.

In addition to [4], [5], there are other proposals for improving congestion control in MANETs using intermediate nodes [6], [8]. A survey of these and other schemes can be found in [21]. Another relevant work is [9], where a new reliable Transport protocol for MANETs, called ATP, is proposed. ATP uses feedback from the network for three different purposes: determining the start-up rate of a connection, congestion detection and avoidance, and path failure notification. The first two feedbacks are orthogonal to the mechanism proposed in this paper and can be used together with it. The third, similarly to [4], [5], is designed to give the sender more information when a route is broken, in order to improve congestion control. It does not help in those cases where our mechanism helps: notifying the sender about the route change, and notifying about a segment loss without waiting for a time-out or dupACKs.

In [11], a mechanism called early packet loss notification (EPLN) is proposed. The purpose of EPLN is to make routing protocols aware of lost data packets and help reduce TCP timeouts for mobility-induced losses. Like the Route-control mechanism proposed in this paper, EPLN is also a cross-layer mechanism: it is implemented in the Network layer and conveys useful information to the Transport layer. Another similarity between Route-control and EPLN is that both help to reduce the time between the loss of a Transport segment and its retransmission. Still, there are many differences between the two mechanisms, the two most important of which are:

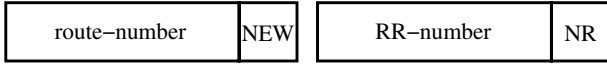
- Like [4], [5], EPLN was also developed for underlying routing protocols with source-route forwarding strategy. In contrast, our Route-control mechanism is designed for routing protocols with hop-by-hop forwarding, where the sender is not aware of the route change.
- EPLN plays a role when a route is broken. In contrast, the Route-control mechanism helps when a route changes.

One of the goals of our Route-control is to allow the sender to retransmit a lost segment after a single duplicate ACK, if the segment to which this ACK responds has been sent over the same route used by the lost segment. This target is also addressed by [7]. However, like most of the schemes discussed above, this one works only with source-route routing protocols.

III. THE ROUTE-CONTROL MECHANISM

A. Informal description of the mechanism

The proposed Route-control (RC) mechanism brings to mind three aspects of the TCP ECN (Explicit Congestion Notification) mechanism [16]: (a) it is implemented mainly in the Network layer and requires the cooperation of the intermediate nodes; (b) the indication provided by the intermediate nodes is mirrored from the receiver to the sender using the regular Transport layer ACK/NACKs; and (c) this information is mainly used by the Transport layer. Fig. 1 shows the new Network and Transport layer fields that are required by the RC mechanism. The two new Network layer fields are the route-number field and the NEW flag. The route-number field is a



(a) Network layer - data segments (b) Transport layer - ACK/NACK
 Fig. 1. New Network and Transport Layer Fields

counter held by a sending node for every receiving node in the network. Initially, it can be set to 0 or to random value, when the sending node bootstraps. Then, if the sending node is informed that the packets towards the particular destination have been routed over a new route, it increments this counter. The NEW is always set to 0 by the sending node. An intermediate node receiving a packet decides whether the next node for this packet is the same as for the previous packet of the same source-destination pair. If not, the NEW flag is set to 1.

The new fields in the Transport layer header (Fig. 1(b)) are related to the transmission of an ACK/NACK from the destination to the source. The RR-number (reverse route-number) equals to the route-number field of the segment to which this ACK/NACK responds, and the NR (new route) flag contains the value found in the NEW field of that segment.

The new Network layer and Transport layer fields can both be accommodated in the current TCP/IP packet structure. In the IP header, the new fields can replace the current Identification and Flags fields, which have been used in the past for packet segmentation and reassembly. In the TCP header, the new fields can be added using the TCP header extension.

An example of the manipulation of the route-number field is shown in Fig. 2(a). The first Transport segment is sent with route-number= X and NEW=0. An intermediate node v realizes that the previous packet for the same source-destination pair was forwarded to a different node from the one to which this packet is to be sent. Therefore, it sets NEW=1. The receiver sends an ACK with RR-number= X and NR=1, confirming the receipt of this segment and indicating to the sender that a segment was sent over a new route, instead of X . Node v continues to set NEW=1, and receiver sets RR-number= X and NR=1, until v receives the packet with new route-number. This allows our mechanism to tolerate the loss of the packets between v and the destination, or the loss of the ACKs. When the sender receives the first ACK, it sets route-number= $X+1$. The next packet routed through v contains a route number greater than X , therefore NEW remains 0. In Fig. 2(b) the first ACK is lost. In this case, the sender increments the route-number for the destination only after getting the second ACK/NACK.

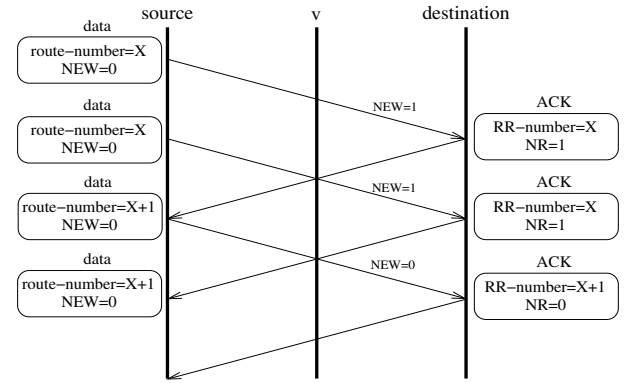
B. A formal description of the mechanism

We now give the formal description of the proposed Route-control mechanism. The algorithm presented below is divided into three parts: the first has to be executed by the sender, the second by the receiver, and the third by the intermediate nodes upon receiving a packet of the considered connection.

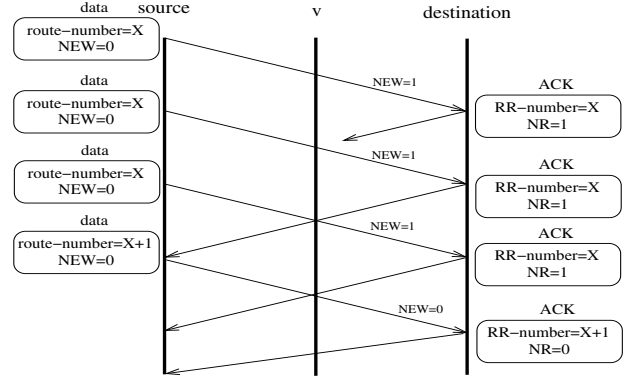
Algorithm 1: (the Route-control mechanism)

Sender s :

- S1 For every destination d , the sender maintains a route-number(s_d) counter. ■



(a) The first ACK is not lost



(b) The first ACK is lost

Fig. 2. An example of the change in the route-number field

- S2 Every data segment i is transmitted to d in a Network layer packet whose route-number field contains the current value of s_d and the NEW field is equal to 0.
 - S3 When an ACK is received from d with RR-number= s_d and NR=1, route-number is incremented.
- Receiver d :**
- R1 When d receives a segment i from s , it responds with an ACK/NACK, where RR-number=route-number(s_d) and the NR=NEW field of segment i .
- Intermediate node v :**
- I1 An intermediate node v maintains a table with an (s, d) -entry for every pair of nodes whose packet has been received by v in the last time-out period of Δ seconds. This entry indicates the next node to which the last (s, d) -packet was sent by v , the s_d value in the header of that packet, and NEW flag value with which the packet was forwarded.
 - I2 Consider an (s, d) -packet received by v . If v has no (s, d) -entry in its table, i.e., no (s, d) -packets were received in Δ seconds, the NEW is unchanged. Else, the NEW is set depending on: (a) the NEW and the route-number fields values in the previous (s, d) -packet; and (b) whether the next node for the (s, d) -packet has been changed. Table I addresses reaction of v to the possible combinations of these factors.
 - I3 Queue the packet for routing to the next node. FIFO queuing is maintained for all the (s, d) -packets. ■

case	Conditions			Action
	next node of previous packet	route-number of previous packet	NEW field of previous packet	
I2-a	indifferent	= the route-number of current packet	NEW=1	set NEW ← 1
I2-b	same as of current packet	< the route-number of current packet	NEW=1	don't change NEW
I2-c	not the same as of current packet	indifferent	indifferent	set NEW ← 1
I2-d	same as of current packet	indifferent	NEW=0	don't change NEW

TABLE I
THE VARIOUS CASES FOR STEP I2 IN ALGORITHM 1

IV. A SELECTIVE REPEAT ARQ PROTOCOL USING THE ROUTE-CONTROL MECHANISM

Reliable transport protocols for MANET are most likely to employ the concept of selective repeat ARQ. Therefore, while the discussion in this section is relevant also for go-back-N, or for the combination of go-back-N with selective repeat (as in TCP Reno), we consider here a pure selective repeat protocol. The principles of the considered protocol are as follows:

Algorithm 2: (a selective repeat algorithm for MANET based on the Route-control mechanism)

- SR-1 When a new segment i arrives at the receiver, an ACK is generated. This ACK describes the full state at the receiver, namely, correct reception of i , missing segments with sequence numbers smaller than i ¹, and sequence numbers of the segments greater than i that have already been received.
- SR-2 When a new segment can be transmitted into the network, subject to the congestion control rules², the sender either retransmits an old segment or transmits a new one according to the following rule. Let LOST be the set of segments whose last transmission is considered lost. If LOST is empty, then a new segment is transmitted. If LOST is not empty, the segment with the smallest sequence number is retransmitted.

With the rules above, the main issue is to determine when to add a segment into the LOST set. Of course, one way is to have a time-out for every outstanding segment:

- SR-3 Every segment i in the sender window is associated with a timer $T[i]$ that indicates the last transmission time of i . Let $RTO[t]$ be the value of the sender retransmission time-out at time t for the considered destination. If $T[i] + RTO[t] \leq t$ holds at time t , segment i must be added to LOST and $T[i]$ is reset.

An RTO-based retransmission mechanism is compulsory in ARQ protocols. However, it should not serve as the only mechanism due to its inefficiency. Determining the value of $RTO[t]$ is a tough issue in MANETs, where not only the network load

¹In the following discussion and for the rest of the paper, a smaller sequence number indicates an earlier (older) segment.

²We assume that the sender maintains a $cwnd$ variable, whose role – as in TCP – is to estimate the amount of outstanding data the sender is allowed to have from the perspective of congestion control. For the sake of simplicity, $cwnd$ indicates the number of segments and not the number of bytes.

is dynamic, but also the link capacities. An inaccurate RTO value causes significant throughput degradation, either due to very long idle times before necessary retransmissions [22], or due to unnecessary retransmissions [23].

In order to reduce the dependency on the time-out mechanism, TCP learns about a loss from the arrival of segments with sequence numbers greater than the next expected one. Since IP does not guarantee in-order delivery, a standard TCP sender retransmits a segment only upon receiving three out-of-order segments, a concept known as “3 duplicate ACKs”[14].

The proposed RC mechanism improves the performance of a MANET’s Transport layer in the following ways:

- If a segment is lost but the route has not changed, the sender knows this and retransmits the lost segment immediately after a **single** dupACK. In [22] it is shown that in such a case throughput can increase twofold.
- If a segment is lost when a route changes, the sender knows this and responds by either waiting for more ACKs, or by ignoring dupACKs and waiting for an RTO.

The implementation of the second idea is straightforward, whereas the first idea can be implemented as follows:

- SR-4 The sender keeps the route-number of the last transmission of every outstanding segment. It also remembers for every segment in LOST, whether it was inserted to LOST due to RTO or due to a valid NACK.
- SR-5 A valid NACK is received for a segment in the following way. Suppose the sender receives an ACK for segment i , which also indicates that j is missing (i.e., it is a NACK for j). Suppose also that:
 - (a) The RR-number, X , in the ACK, is equal to the route-number of i ’s last transmission.
 - (b) The last transmission of i was conducted after the last transmission of j , $T[i] > T[j]$.
 - (c) The last transmission of j contains the same route-number, X , as the last transmission of i , and the ACK for i returns with $NEW = 0$.
 - (d) The last transmission of i was its only transmission, or it was a retransmission due to a valid NACK for i (and not due to an RTO).

Then, regardless if j is bigger or smaller than i , the NACK is valid, and j should be added to LOST. ■

Note that the above algorithm does not require the sender to know whether the ACK/NACKs are received in order.

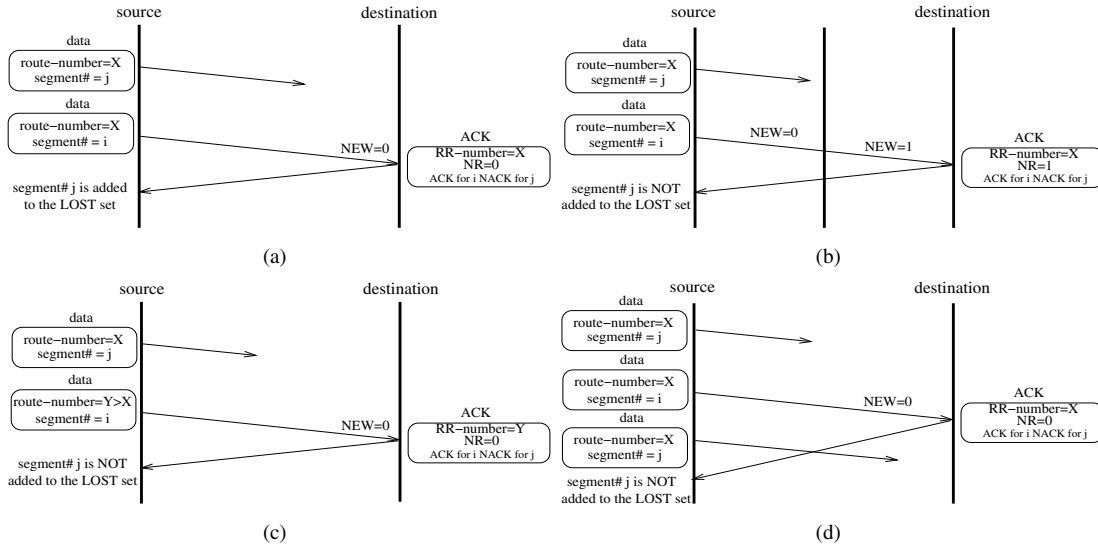


Fig. 3. Four possible cases for an ACK with a NACK

Some scenarios related to Alg.2 are illustrated in Fig. 3. Fig. 3(a) presents the case where all the conditions above are met, and segment j is therefore added to LOST. Fig. 3(b) presents the case where i is transmitted over a new route, despite the fact that it was sent with the same route-number as j . This implies that condition (c) is not met and j is not added to LOST when the ACK for i is received. This is because the sender is unable to decide whether the NACK was originated by a packet loss or by reordering. Fig. 3(c) presents the case where segment i is transmitted with a route-number larger than that of j . Hence, condition (c) is not met and j is not added to LOST. Finally, in Fig. 3(d) condition (d) does not hold.

Theorem 1: Every segment inserted by the sender into the LOST set due to a valid NACK is a segment for which all previous transmissions have indeed been lost.

Proof: Suppose the theorem is incorrect. Let segment j be the first one to enter the LOST by mistake at time t_5 due to reception of NACK. This NACK is sent at t_3 as a response to segment i , sent at t_2 (Fig. 4(a)). This implies the transmission of j that started at $t_1 < t_2$ and received at $t_4 > t_3$.

We prove that there is no transmission of i after t_2 . Segment i cannot be sent after t_5 , when the ACK for i is received. It cannot be transmitted during (t_2, t_5) because: (a) if i is transmitted due to a valid NACK for i , then this NACK precedes the incorrect NACK for j at t_5 , contradicting our assumption; (b) if i is transmitted after an RTO, then the insertion of j at t_5 into the LOST contradicts step SR-5(d) of Alg.2. By step SR-5(b) of Alg.2, the fact that the last transmission of i is conducted at t_2 and the insertion of j into LOST at t_5 implies that there is no transmission of j after t_2 , see Fig. 4(b).

Because the ACK for i returns with NR=0, i reaches the receiver with NEW = 0, by step R1 of Alg.1. Suppose, i is routed along the path $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_N \rightarrow d$, where $N \geq 1$ and $v_1 = s$. By steps I2-a and I2-b in Table I, every node v_i forwards a previous (s, d) -packet towards d through node v_{i+1} . Therefore, i belongs to a burst forwarded along $v_1 \rightarrow \dots \rightarrow v_N \rightarrow d$ with a route-number = X . By step I3

of Alg.1, each of these packets is either dropped or received in-order by d . This implies that j , sent at t_1 , does not belong to this burst, and it is sent before the first packet, p , of this burst (see Fig. 4(c)). Let v_i be the first node from $v_1 \dots v_N$ that transmits p over a new route. By step I2-c of Alg.1, v_i sets the NEW flag of p to 1. By step I2-a, v_i also sets the NEW of all the other (s, d) -packets in this burst to 1, as they all have the route-number X . However, this contradicts the fact that i , transmitted at t_2 , is received with NEW = 0. ■

V. ENHANCING RTT COMPUTATION AND CONGESTION CONTROL USING THE ROUTE-CONTROL MECHANISM

This section extends the proposed RC mechanism in order to enhance RTT computation and congestion control in MANETs. To this end, we use the Network layer fields shown in Fig. 1(a) in the ACK/NACK segments as well. This allows the sender to decide whether the received ACK/NACK has travelled on the old or on a new route.

A. RTT computation

Several papers have shown that rapid changes in the RTT of a TCP connection cause significant throughput degradation [24], [25]. In order to avoid unnecessary retransmissions, RTO is set to a value that is much larger than the estimated RTT. Hence, retransmissions of lost segments are delayed and throughput is negatively affected.

Every reliable Transport protocol must use a retransmission time-out mechanism for preventing deadlocks. Defining a good value for the RTO is important and difficult. It is important, because a premature RTO results in unnecessary retransmissions and an unnecessary congestion window decrease, which are translated into significant throughput degradation. It is difficult, in particular in MANETs, because the RTO is determined from the RTT (round-trip time) estimation, but the latter is affected by the load variance and by the nodes mobility.

In order to correctly match a segment and its NACK, the RTT of a retransmitted ACK is not taken into account by current TCP implementations, according to ‘‘Karn’s algorithm.’’

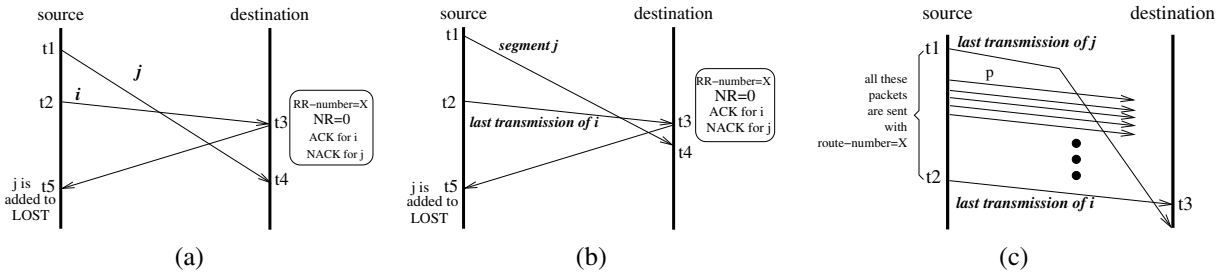


Fig. 4. The proof of Theorem 1

With the proposed RC mechanism, an ACK can be **correctly** associated with the segment to which it responds, because the ACK contains the route-number of the received data segment.

If a segment is routed over a different path than the previous segments of the same connection, its estimated RTT can significantly differ from the actual value [23]. Using the proposed RC mechanism, a sender knows whether the ACK it has received and/or the data segment to which this ACK responds have travelled over a new path. If this is the case, the sender ignores the previous RTT computation, and starts over using the RTT sample of the new ACK.

We now present an algorithm for RTT estimation:

Algorithm 3: (using the RC for enhancing RTT estimation)

For every outstanding segment i , the sender records the local time of its last transmission, in addition to the route-number for this transmission. Consider an ACK received for i :

- If the RR-number of this ACK is not equal to the route-number of i 's last transmission, the ACK is not taken into account for the RTT computation, as it responds to an earlier transmission of i .
- Else, if NR=0 in this ACK and NEW=0 in the network layer header of the ACK, the time difference between the last transmission of i and the current time is considered a new RTT sample. This sample is combined into the previous RTT estimation, e.g., using the algorithm in [13].
- Else, the data segment and/or its ACK have travelled over a new route. Therefore, the previous RTT computation should be ignored. And the current RTT sample is considered the first sample for the new route. ■

B. Enhancing Congestion Control

The end-to-end Transport layer congestion control scheme for best-effort traffic is vital to the success of the TCP/IP in the Internet. Therefore, it plays a key role in any MANET architecture. An important issue in such a scheme is how the sender learns about a network congestion and adjusts its rate. There are several approaches to address this issue:

Reno In the standard TCP Reno [14], the sender learns about the network load from the ACKs it receives. Specifically, (a) the receipt of an in-order ACK indicates that the network is not congested and $cwnd$ can increase; (b) the receipt of several dupACKs means the network is “slightly” congested and $cwnd$ is halved; (c) the loss of several packets during one RTT, means heavy congestion and $cwnd$ is set to 1.

ECN An optional TCP/IP congestion control mechanism is Explicit Congestion Notification [16]. With this scheme, a congested router sets a flag in the packet's IP header telling the receiver to reduce the load. This message is reported back to the sender using a flag in the TCP header. The sender responds by halving $cwnd$, as it does after the receipt of 3 dupACKs.

RTT In TCP Vegas [26], congestion control is based on precise measurement of the RTT. An increase in the RTT indicates to the sender that congestion has developed and that $cwnd$ has to be reduced.

EB The equation-based approach for TCP congestion control [17], [18] is based on the following steps: (a) the receiver measures the loss event rate and feeds it to the sender; (b) the sender measures the RTT; (c) the loss rate and RTT are fed into the equation, which outputs the acceptable transmit rate; (d) the sender adjusts its transmission rate accordingly.

Deciding on the best congestion control scheme for MANET, or developing a new one, is beyond the scope of this paper. In fact, a congestion control scheme for a MANET depends on many parameters: the underlying routing protocol, the network traffic, the density of the network, and so forth [21]. The purpose of the following discussion is to show how each of the standard congestion control schemes can significantly benefit from the Route-control mechanism in a MANET.

- 1) The RC mechanism notifies the sender when its packets are forwarded over a new route. This is important information because the load over the new route may differ greatly from the load over the old one. This important aspect of the RC mechanism can play a significant role in each of the congestion control schemes described above.
- 2) The RC mechanism allows the sender to distinguish between a dupACK indicating one or more packet losses and a dupACK that might be caused by the out-of-order routing. This has a bearing on all the congestion control schemes – in particular TCP-Reno – that view a duplicate ACK as an indication of a single packet loss.
- 3) As explained earlier, the RC mechanism allows the sender to much better estimate the RTT. This is crucial to each of the congestion control schemes discussed above, and in particular to the EB and the RTT schemes.

VI. SIMULATION STUDY

The purpose of this section is to better understand the behavior of various protocols with the proposed RC mechanism, and

the conditions where they can benefit from it. We used ns2 for this study and augmented it with our own node mobility generator. This generator produces random vectors of movement, accordingly to the commonly used Random Waypoint model [27]. In addition, it controls the percentage of mobile and static nodes and uses transmission ranges to analyze connectivity.

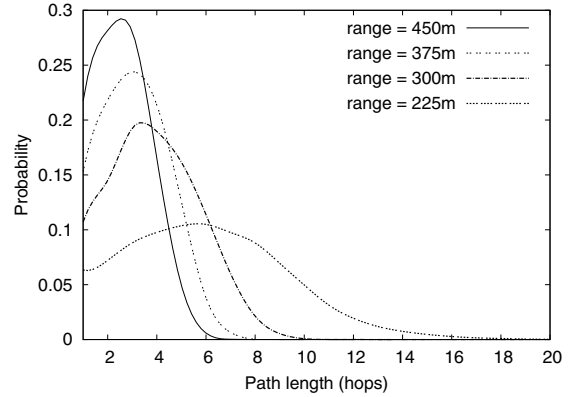
The grid size for all the simulations was set to 1500×1500 m. We placed 100 nodes on the grid half of which are static and the other half move with constant speed in a random direction. A pair of nodes was randomly chosen to be source and destination in each experiment, and the path between the two nodes was analyzed. We ran 100 experiments for each instance and computed the average.

We first report results on the path length distribution and on the stability of the paths. The graph in Fig. 5(a) shows the path length distribution for the 25m/s velocity. Its y-axis indicates the probability for a specific path length value. The graph in Fig. 5(b) shows the distribution of the time between two consecutive path changes when the velocity is 12.5m/s. Its y-axis indicates the probability for a given value of time between two consecutive path changes between a source-destination pair. In both graphs we show several curves for different transmission ranges. As expected, the increase in the transmission range leads to the decrease of the path length. Moreover, Fig. 5(b) shows that a small transmission range causes more frequent changes on the paths between source-destination pairs. For example, the probability that the path will change after 2 seconds is about 0.25 when the range is 450m and 0.5 when it is 225m.

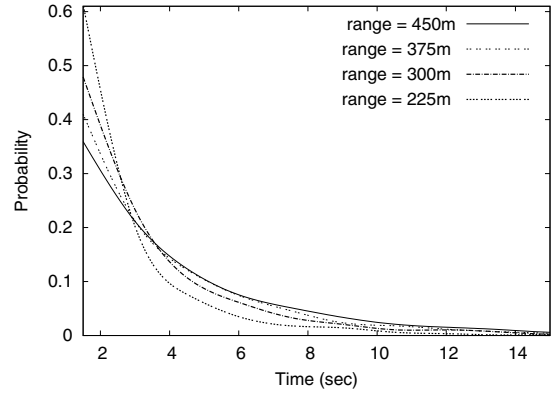
After defining the movement scenarios, we implemented the Route-control mechanism in ns2. The bandwidth of the links was set to 5Mb/s, the transmission range to 225m and the average loss rate per hop varied between 0 and 2%. The speed of the mobile nodes varied between 2.5 and 25m/s. We first consider packet loss due to a communication failure and not due to mobility. We see that very often RC abolishes the need to wait for 3 duplicate ACKs and the transport protocol is able to retransmit the lost packet after the first dupACK. Consequently, many retransmission timeouts (RTOs) are avoided.

In Fig. 6 we show the number of RTOs and the percentage of RTO reduction as a function of the average per-hop link loss rate during the transmission of a 100Mb file. We show several curves for two mobile node velocities. Fig. 6(a) shows the number of RTOs for TCP SACK with and without Route-control (RC). It is easy to see that when the nodes move faster or when the link loss rate increases, the number of RTOs is larger. Also, as the loss rate grows, so does the effect of RC on the number of RTOs. For example, for a speed of 2.5m/s and an average loss rate of 2% per each hop, RC reduces the number of RTOs from 130 to 85. The percentage of improvement for SACK and New-Reno is shown in Fig. 6(b).

Fig. 6(b) shows that RC reduces the number of RTOs for SACK by almost 40% when the per-hop link loss rate is 1%. For the same settings, RC reduces the number of RTOs for New-Reno by only 20%. This can be explained by the New-Reno's impatient timeout, which does not allow the sender to



(a) Path length distribution



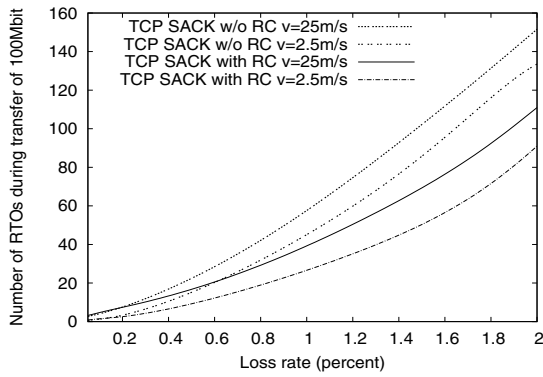
(b) Time between consecutive path changes

Fig. 5. Patterns of MANET behavior derived from simulations

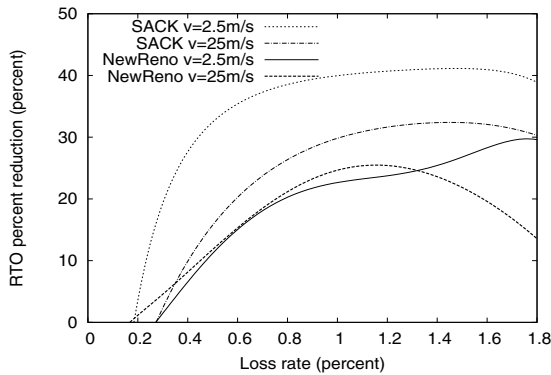
stay in the Fast Recovery stage for more than one RTO. Thus, early detection of packet loss by RC helps to avoid timeout in fewer cases, resulting in less significant contribution to New-Reno.

Another enhancement of the proposed RC scheme is more accurate RTT estimation. Fig. 7 shows the difference between smooth RTT (SRTT) measurement based on the algorithm in Section V and the traditional SRTT estimation with TCP SACK. The graph in Fig. 7(a) shows typical SRTT estimations vs. an RTT sample for a single simulation run. The curve of RC SRTT is very close to the one of the RTT measurements. This indicates that RC responds almost immediately to an RTT change, while the original SRTT estimation algorithm is more rigid and cannot tolerate frequent changes in a path length.

Fig. 7(b) summarizes the difference between the estimated SRTT and the measured RTT for 100 runs. It shows the variance of the SRTT normalized with respect to the RTT expectation, i.e., $Var(SRTT)/E[RTT] = E[(SRTT - RTT)^2]/E[RTT]$. As before, we see that the variance of the SRTT with RC is smaller than without it. For example, the variance without RC is 0.9, when the velocity is 25m/s and the average per-hop loss rate is 1%. For the same settings with RC, the variance is only 0.2. We also see that higher node mobility causes higher SRTT variance. This can be explained by more frequent changes in RTT, to which SRTT algorithm cannot easy adapt. Moreover, a high average packet loss rate affects SRTT variance, as it reduces the number of samples available for SRTT calculation.



(a) Number of RTOs during 100Mb transmission



(b) Percent of RTOs reduced in TCP SACK and New-Reno

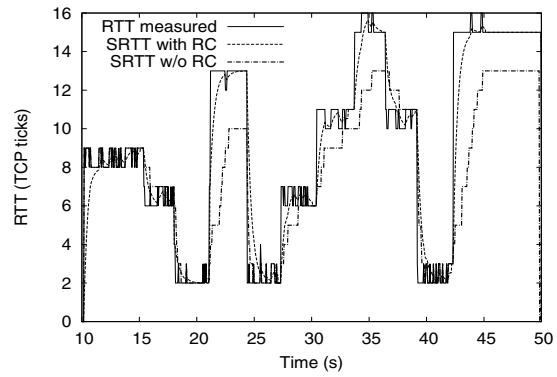
Fig. 6. Reduction in number of RTOs with Route-control mechanism

VII. CONCLUSIONS

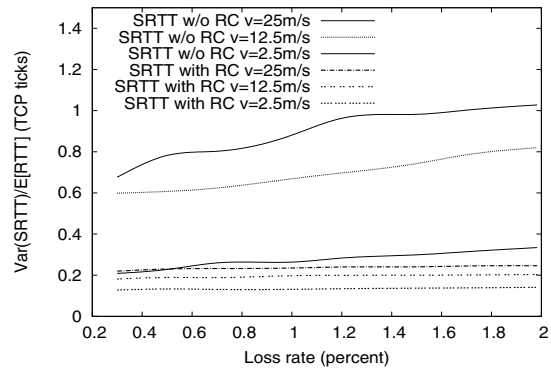
We proposed a new cross-layer mechanism, referred to as Route-control (RC), for MANETs. This mechanism notifies the sender when the packet sent to a given destination have been forwarded on a new route. The sender can benefit from this information when deciding whether to retransmit a missing Transport layer segment or to wait, also when estimating the RTT, and when deciding whether to change the congestion window. Using simulations, we showed that a sender can use RC to retransmit a lost segment as soon as it receives the first duplicate ACK, which helps to avoid many unnecessary RTOs. We also showed that RC improves the sender's RTT estimation, which is also translated to fewer RTOs.

REFERENCES

- [1] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," RFC 3561, July 2003.
- [2] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," RFC 3561, Oct. 2003.
- [3] R. Ogier, F. Templin, and M. Lewis, "Topology dissemination based on reverse-path forwarding (TBRPF)," RFC 3684, Feb. 2004.
- [4] K. Chandran, and et al., "Feedback-based scheme for improving TCP performance in MANET," *IEEE Pers. Comm. Mag.*, Feb. 2001.
- [5] G. Holland and N. H. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," in *MobiCom*, Aug. 1999.
- [6] D. Kim, C. Toh, and et al., "TCP-bus: Improving TCP performance in wireless ad-hoc networks," in *ICC (3)*, 2000, pp. 1707 – 1713.
- [7] D. Kim, C. Toh, and H. Jong, "An early retransmission technique to improve TCP performance for mobile ad hoc networks," in *PIMCR '04*.
- [8] D. Sun and H. Man, "ENIC - an improved reliable transport scheme for mobile ad hoc networks," in *GlobeCom*, 2003.
- [9] K. Sundaresan, V. Anantharaman, H. Hsieh, and R. Sivakumar, "ATP: a reliable transport protocol for ad-hoc networks," in *MobiHoc*, 2003.



(a) Single run comparison of SRTT with and w/o Route-control



(b) Variance of SRTT with and w/o Route-control

Fig. 7. SRTT with and w/o Route-control

- [10] F. Wang and Y. Zhang, "Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response," in *MobiHoc'02*.
- [11] X. Yu, "Improving TCP performance over mobile ad hoc networks by exploiting cross-layer information awareness," in *MobiCom '04*.
- [12] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," RFC-2018, Oct. 1996.
- [13] V. Paxson and M. Allman, "Computing TCP's retransmission timer," RFC 2988, Nov. 2000.
- [14] M. Allman, V. Paxson, and W. Stevens, "Tcp congestion control," RFC-2581, Apr. 1999.
- [15] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno modification to TCP's fast recovery algorithm," RFC-3782, Apr. 2004.
- [16] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," RFC 3168, Sept. 2001.
- [17] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *SIGCOMM*, 2000.
- [18] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specification," RFC 3448, Jan. 2003.
- [19] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, 1996, vol. 353.
- [20] D. Johnson, Y. Hu, and D. Maltz, "The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4," RFC 4728, Feb. 2007.
- [21] C. Lochert, and et al., "A survey on congestion control for MANETs," *Wiley Wireless Communications and Mobile Computing*, June 2007.
- [22] R. Cohen and S. Ramanathan, "Tuning TCP for high performance in hybrid fiber coaxial broadband access networks," *Trans. on Net.*, Feb.'98.
- [23] M. Yavuz and F. Khafizov, "TCP over wireless links with variable bandwidth," in *Vehicular Technology Conference*, 2002.
- [24] T. D. Dyer and R. V. Boppana, "A comparison of TCP performance over three routing protocols for mobile ad hoc networks," in *MobiHoc '01*.
- [25] S. Bohacek and et al., "A new TCP for persistent packet reordering," *IEEE/ACM Trans. on Net.*, vol. 14, no. 2, pp. 369–382, 2006.
- [26] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *SIGCOMM '94*, pp. 24–35.
- [27] J. Broch and et al., "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *MobiCom*, 1998, pp. 85–97.