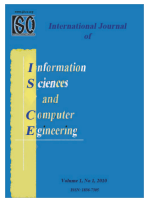
Information Sciences and Computer Engineering, Vol. 1, No. 1, (2010) 1–8

PHARUS

International Journal of
Information Sciences and Computer Engineering

journal homepage: http://www.ijisce.org



Collision Resistant Temporally Ordered Routing Algorithm

Vinay Thotakura, Mahalingam Ramkumar*

Department of Computer Science and Engineering, Mississippi State University, Box 9637, Starkville, MS 39762

***Abstract*– A broad class of mobile ad hoc network (MANET) applications will require every mobile node in a connected subnet to establish persistent paths with a single sink or gateway or command-post. Temporally ordered routing algorithm (TORA) is the only MANET routing protocol which has been explicitly designed to facilitate such a requirement. However, TORA has several shortcomings like i) the need for an expensive lower layer, the Internet MANET Encapsulation protocol (IMEP); and ii) substantial overhead in scenarios involving subnet partitions. We propose a novel protocol, collision-resistant TORA (CR-TORA), which is also designed to meet the same goal as TORA, viz., to lower overhead for establishing persistent paths from all nodes to a destination. However, CR-TORA overcomes the shortcomings of TORA. Simulations show that for the intended application scenario CR-TORA demands lower number of total packets to be transmitted to realize higher data packet throughput, with lower latency.**

1. Introduction

A mobile ad hoc network (MANET) is a collection of mobile nodes which can communicate with each other over multiple hops, using other nodes in the subnet as routers. Several proactive and reactive MANET routing protocols [1] have been proposed in the literature which permit any two nodes in the subnet to establish a path.

Among several applications of MANETs, many will require establishment of persistent paths from all mobile nodes to one, or a small number of (possibly mobile) sink(s). One example is a system that facilitates every mobile soldier to relay messages to a command post. Another is that of a sensor network used for monitoring mobile fauna, where mobile sensors periodically send data to a single sink. A third example is that of wireless gateways in rural areas, where multi-hop connectivity from the mobile nodes to the gateway can substantially enhance the coverage that can be provided by the gateway.

Temporally ordered routing algorithm (TORA) [2, 3] was explicitly designed for application scenarios requiring persistent paths from all mobile nodes to one sink. This is in contrast to other popular MANET routing protocols like dynamic source routing (DSR) [4], ad hoc on-demand distance vector (AODV) [5] and destination sequenced distance vector (DSDV) [6], which aim to establish paths between *all* nodes in a connected subnet.

*Corresponding author:
Email address: ramkumar@cse.msstate.edu, Ph: +1 6623258435

1.1. Contribution

The contribution of this paper is a novel routing protocol, collision-resistant TORA (CR-TORA), which shares a basic design goal with TORA - to lower overhead for establishing multiple persistent paths from all nodes to a destination. CR-TORA is however designed to overcome many shortcomings of TORA [7].

Firstly, TORA relies on a lower layer - the Internet MANET encapsulation protocol (IMEP) [8] - to provide a collision free environment for TORA. The overhead for IMEP itself can be substantial. CR-TORA is designed to obviate the need for an expensive lower layer to address collisions. Secondly, under conditions of network partitions TORA results in significant control message overhead and a long "settling time" before which nodes in a partitioned subnet can realize that they do not have a valid path to the destination. This can result in needless attempts to transmit data packets which are ultimately bound to fail. CR-TORA is designed to ensure that nodes recognize such partitions quickly. Thirdly, unlike TORA, CR-TORA does not require time-synchronization.

For the intended application model, where mobile nodes need to periodically relay data packets to a destination (command post, sink, gateway etc.), we show that CR-TORA outperforms TORA in every conceivable respect. Typically, compared to TORA, CR-TORA reduces the total number of transmissions by 60%; results in a 15% increase in throughput; and a 40% reduction in latency.

The rest of this paper is organized as follows. Section 2 is an overview of TORA. The proposed CR-TORA protocol is outlined in Section 3. Section 4 describes the simulation model used to evaluate TORA and CR-TORA for the desired application scenario. We compare TORA and CR-TORA under various mobility models and network density, on the basis of packet delivery ratios, total number of transmissions, and latency. Conclusions are offered in Section 5.

2. TORA

In [9] Gafni and Bertsekas proposed two distributed algorithms for the problem of "maintaining communication between the nodes of a data network and a central station in the presence of frequent topological changes." Temporally ordered routing algorithm [2] is based on a *half-reversal* algorithm in [9]. For a

subnet with N nodes and t destinations, t independent instances of TORA can be used - one for each destination. In the rest of this paper we shall assume a single destination ϕ.

2.1. Overview of TORA

In TORA the route to the destination ϕ from any node is based on the "height" of the node with respect to the destination ϕ. Every node stores heights of all neighbors for the destination, and based on such heights, determines its own height. If a node A has greater height than a neighbor B, then B is regarded as the downstream neighbor of A. Data packets can flow only downwards; i.e from upstream to downstream.

The height of a node i is a 5-tuple $(\tau, oid, r, \delta, i)$. The first three values (τ, oid, r) form the "reference level" (RL) where i) τ is the time of creation of the RL (TORA assumes some mechanism for time synchronization between nodes); ii) oid is the identity of the node that created the RL; and iii) r is a reflection bit. The value δ indicates the hop-count from the node which created the RL. For example, $(0, 0, 0, 5, A)$ is the height of the node A with respect to a RL $(0, 0, 0)$. The destination ϕ is always at a height ZERO: $(0, 0, 0, 0, \phi)$. The 5-tuple $(-, -, -, -, A)$ implies that A has a NULL height - or no forwarding path is available for A to the destination. Every node stores the height of all neighbors and categorizes them as downstream or upstream. If two neighbors have the same height (same RL and δ) the node with a smaller identity is considered to be downstream of the other.

Control Packets and IMEP: TORA employs three different types of control packets: OPT, UPD and CLR. In the proactive[1] mode of TORA (which is better suited for the intended application scenario) the destination periodically advertises its presence with an OPT packet with a fresh sequence number which includes its height $(0, 0, 0, 0, \phi)$. A one hop neighbor A of the destination rebroadcasts the OPT with a height $(0, 0, 0, 1, A)$. A node X, k hops away will typically broadcast an OPT with height $(0, 0, 0, k, X)$. TORA can afford to use such far-reaching OPT packets infrequently, as changes in topology that occur between two OPT updates are handled using localized UPD and CLR packets. One of TORA's design goals was to minimize the control packets required for dealing with topological changes.

Every control packet broadcast by a node is intended for every neighbor. TORA employs IMEP [8] to i) maintain a reliable delivery neighborhood (RDN) by keeping track of nodes within its range with bidirectional links; and ii) to provide a collision-free foundation for TORA, by sending/processing acknowledgements, and taking care of necessary retransmissions.

To lower overhead IMEP also attempts to encapsulate multiple small control/ acknowledgement packets into one. For example, when a node A submits a control packet to the IMEP layer for transmission, the IMEP layer may include acknowledgements for control packets previously received by A. Furthermore, in a scenario where A has neighbors B, C, D, and E, if only B and D had acknowledged a control packet sent by A, the IMEP layer may retransmit the control packet explicitly indicating only the identities C and E of neighbors which need to acknowledge the retransmission.

Route Maintenance: A node A may loose a downstream neighbor (say B) due to two reasons: i) *link failure*, where the IMEP layer of A informs A that the link to B does not exist anymore ; and ii) *link reversal*, where the previously downstream B declares that it is now upstream of A, by sending a UPD packet.

The TORA Algorithm

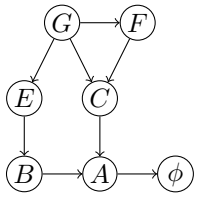
```
IF (link-failure)
   GENERATE-RL
ELSE //(link reversal)
   IF (all-neighbors-not-at-same-RL)
     PROPAGATE-RL
   ELSE
     IF (r == 0)
       REFLECT-RL
     ELSE
       IF (RL-created-by-me)
         CLEAR-RL
       ELSE
         GENERATE-RL
```

Route maintenance is triggered by a node only when it looses its last downstream neighbor. The reactions of a node to the trigger can be one of the following: i) GENERATE RL; ii) PROPAGATE RL ; iii) REFLECT RL; and iv) CLEAR RL.

A node i at a height $(\tau, oid, r, \delta, i)$ *generates* a new RL by sending a UPD with height $(\tau + 1, i, 0, 0, i)$. Node i *propagates* an RL by picking the highest RL among its neighbors, and among such neighbors, the neighbor with the least height. If the height of such a neighbor j is (rl, δ', j), where $rl = (\tau', oid', r')$ node i sends a UPD with height $(rl, \delta' - 1, i)$. If all its neighbors are at the same RL $(\tau', oid', r = 0)$, i *reflects* the level, by sending a UPD with height $(\tau', oid', r = 1, 0, i)$. To *clear* an RL i sends a CLR packet with height $(-, -, -, -, i)$.

2.2. TORA: Pros and Cons

Consider a simple subnet topology depicted alongside, where ϕ is the destination node.



After the OPT from the destination ϕ is propagated the heights of the nodes will be: $(rl, 0, \phi)$, $(rl, 1, A)$, $(rl, 2, C)$, $(rl, 2, B)$, $(rl, 3, G)$, $(rl, 3, E)$, and $(rl, 3, F)$, where $rl = (0, 0, 0)$. Assume that the link $C \rightarrow A$ goes down at time τ. The sequence of events that will transpire are then as follows:

a) C generates a new RL $rl' = (\tau, C, 0)$ by sending a UPD with height $(rl', 0, C)$; G takes no action as it still has downstream neighbors E and F;
b) F propagates the new RL by sending a UPD with height $(rl', -1, F)$; even after this link reversal G has a downstream neighbor E, hence no action is taken.

Clearly, TORA strives to reduce the number of control packets required to deal with changes in topology, and does this well. Where TORA does not do so well is in scenarios that result in partitioning of the subnet. To see this consider a scenario where A looses its link to destination ϕ at time τ. The sequence of events are now as follows:

a) $A \rightarrow$ UPD $(rl, 0, A)$ (A generates new RL $rl = (\tau, A, 0)$);
b) $B \rightarrow (rl, -1, B)$; $C \rightarrow (rl, -1, C)$ (B and C propagate new RL);

[1] In the *reactive* mode of TORA another control packet, query (QRY), is also used. A node with a NULL height can broadcast a QRY packet. If a node receiving a QRY packet has a NULL height it forwards the QRY; a node with a non-NULL height will respond with a UPD.

c) $E \rightarrow (rl, -2, E)$ and $F \rightarrow (rl, -2, F)$ (E and F propagate new RL);
d) $G \rightarrow (rl', 0, G)$ (G reflects RL as $rl' = (\tau, A, 1)$);
e) $F \rightarrow (rl', -1, F), C \rightarrow (rl', -1, C)$ and $E \rightarrow (rl', -1, E)$ (F, C and E propagate reflected RL);
f) $B \rightarrow (rl', -2, B)$ (B propagates reflected RL);
g) $A \rightarrow$ CLR (A detects partition);
h) $B \rightarrow$ CLR; $C \rightarrow$ CLR (propagating CLR in the isolated subnet);
i) $G \rightarrow$ CLR, $F \rightarrow$ CLR; $E \rightarrow$ CLR (propagating CLR in the isolated subnet);

Note that a UPD with a new RL has to reach all nodes in the partition. The RL is then reflected and returned to the originator of the RL, which detects a partition. Following this the originator of the RL creates a CLR packet which needs to be propagated throughout the partitioned subnet. If the average per-hop processing delay is Δ, and L is the longest path in the partitioned subnet, for a time of up to $3L\Delta$ several nodes in the partitioned subnet may possess non-NULL heights which are actually invalid. During this time data packets created by nodes in the partition may be relayed back and forth till the time the nodes realize that they do not actually have a valid height. Another disadvantage of TORA is the need for some mechanism for time-synchronization between nodes to correctly interpret the field τ in an RL.

Collisions: Perhaps the most acute of TORA's shortcomings is its susceptibility to collisions. Once again consider a scenario above where A reverses its link to C by sending a UPD packet. Now assume that the packet was lost due to collision. At this point A assumes that C is its downstream neighbor, while C assumes that A is downstream, thereby creating a simple loop. More complex loops can also result due to collision [7]. That TORA by itself was not designed to address collisions is the reason that mandates a lower layer for this purpose. While TORA itself requires low overhead for control packets, when considered together with the lower layer for sending / processing acknowledgements to / from every neighbor, TORA becomes far less appealing.

3. Collision Resistant TORA

Like the proactive version of TORA, CR-TORA is intended for application scenarios where all mobile nodes require to send data packet to a single (possibly mobile) destination. Similar to proactive TORA, CR-TORA employs CLR, UPD and OPT packets, and data flows only downwards.

The primary difference between CR-TORA and TORA is that CR-TORA (as the name implies) has some in-built features to handle collisions, and thereby eliminates the need for a lower layer like IMEP to address collisions. In doing so CR-TORA also lowers the control packet overhead and the settling time during network partitions, and eliminates the need for time-synchronization.

Some of the other salient differences between TORA and CR-TORA are:

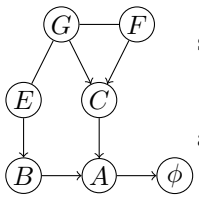
i) CR-TORA does not use reference levels; the height of a node i is a single value δ_i which is typically the number of hops from the destination.
ii) Two neighbors i and j at the same height (or $\delta_i = \delta_j$) do *not* consider each other as upstream/downstream based on their identities. In CR-TORA a node i is downstream of j only if $\delta_i < \delta_j$.

3.1. CR-TORA: Principle of Operation

Recall that in TORA a node responds to link-failures or link-reversals by *generating* an RL, or *propagating* an RL, or *reflecting* an RL, or *clearing* an RL. In CR-TORA nodes respond by *creating* CLR, or *propagating* CLR, or *creating* UPD, or *propagating* UPD, or *retransmitting* CLR.

The basic principle of operation of CR-TORA is as follows. A node losing its last downstream link *creates* a CLR packet. If the receipt of a CLR packet leaves a node X with no downstream neighbor, X *propagates* the CLR right away. However, if X has other downstream neighbors it starts a timer which runs for a duration T_W. While the timer is running X may continue to receive other control packets. If (during this time) other control packets cause X to loose its last downstream neighbor, X propagates a CLR right away. On the other hand, if X retains one or more downstream neighbors after the timer expires, X *creates* a UPD. Neighbors of X with NULL heights then *propagate* the UPD. Most often CLR packets are created when a node looses its last downstream neighbor. However there are other scenarios (which could result due to collisions) which would also mandate creation of a CLR packet, or retransmission of a CLR packet.

To facilitate comparison of TORA and CR-TORA we shall once again consider the same example subnet considered for TORA in Section 2.2. At this point we shall ignore collisions (a more detailed description of CR-TORA follows in Section 3.2). After the OPT from the destination ϕ is propagated the heights of the nodes will be $(0, \phi)$, $(1, A)$, $(2, B)$, $(2, C)$, $(3, E)$, $(3, G)$, and $(3, F)$. Note that in CR-TORA links between nodes that have same height are undirected (unlike TORA).



Assume that as earlier, the link $C \rightarrow A$ goes down. The typical sequence of events in CR-TORA will be as follows:

a) $C \rightarrow$ CLR (C *creates* CLR);
b) $F \rightarrow$ CLR; $G \rightarrow$ CLR; (F and G *propagate* CLR); E waits for a time T_W ;
c) (No clear from B) $E \rightarrow$ UPD, $\delta_E = 3$ (E *creates* a UPD);
d) $G \rightarrow$ UPD, $\delta_G = 4$ (G *propagates* UPD);
e) $F \rightarrow$ UPD, $\delta_F = 5$; $C \rightarrow$ UPD, $\delta_C = 5$ (F and C *propagate* UPD).

In a scenario where A looses its link to destination ϕ (causing a subnet partition) the sequence of events that transpire in in CR-TORA will be:

a) $A \rightarrow$ CLR (A creates CLR);
b) $B \rightarrow$ CLR; $C \rightarrow$ CLR (B and C propagate CLR);
c) $E \rightarrow$ CLR; $F \rightarrow$ CLR; $G \rightarrow$ CLR (E, F and G propagate CLR).

As can be seen readily by comparing the above sequence of events with those for TORA (in Section 2.2), TORA generates

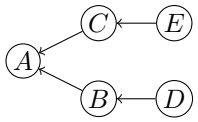
lower number of control packets compared to CR-TORA when there is no network partition. However, this does not necessarily mean that CR-TORA has a higher overhead under such scenarios as we have ignored the overhead for IMEP in TORA. Even if IMEP overhead is ignored, TORA still generates a substantially higher number of control packets compared to CR-TORA when partitioning occurs in the subnet.

3.2. The CR-TORA Protocol

In the rest of this section we shall take a more in-depth look at the CR-TORA protocol.

3.2.1. CLR Event Identifiers (CEI)

In CR-TORA a CLR is created in response to a specific event - like a broken link. We shall see soon that there are two other specific events that lead to creation of a CLR. A node creating a CLR assigns a unique CLR event identifier (CEI) to the event.



For example, if the link $B \rightarrow A$ goes down, B will set its height to NULL, and create a CLR with CEI α. Node D, which on receipt of the CLR-α, has lost its only downstream neighbor sets its height to NULL and simply *propagates* CLR-α. On the other hand, if the node A goes down, two different CEIs will be created: one by B, say CLR-α, due to the loss of the link $B \rightarrow A$, and one by C, say CLR-β, due to the loss of the link $C \rightarrow A$.

Unlike UPD reference heights in TORA, CR-TORA CEIs are *not* tied to time - or time synchronization is not necessary in CR-TORA. The only requirement is that no two CEIs should be the same. A simple strategy to accomplish this is choose the CEI by concatenating the identity of the creator with a sequence number maintained by the creator. In this case a CLR created by a node C will have a CEI $C \parallel q_c$. The next CLR created by C will have a CEI $C \parallel (q_c + 1)$. In the rest of this paper we shall simply employ lower case Greek letters to represent CEIs.

3.2.2. CLR-List and UPD-List

Every node maintains a CLR-list and UPD-list, both being a list of CEIs. Both lists are emptied from time-to-time, under different circumstances. The CLR-list C_i of node i is a list of CEIs made known to i, through CLR packets received by i. Both CLR and UPD packets broadcast by a node i will include its CLR-list[2] C_i:

i) a CLR from i is of the form $[i, CLR, C_i]$;
ii) a UPD from i is of the form $[i, UPD, \delta_i, C_i]$.

When a node i with CLR-list C_i receives a packet $[j, CLR, C_j]$, it adds all CEIs in C_j that were not already in its CLR-list C_i to its CLR-list. Thus, after the CLR is received $C_i = C_i \cup C_j$.

A UPD is *created* by node i only after its T_W-timer fires. The timer is started when i receives a CLR, and it still has at least one downstream neighbor. If i has at least one downstream neighbor left even after the timer fires, i creates a UPD which includes C_i. As soon as the UPD is sent, i i) creates a UPD-list $\mathcal{U}_i = C_i$; and ii) empties CLR-list C_i.

A node *propagating* a UPD merely empties its CLR-list - it does *not* create a UPD-list. A node will propagate a UPD only if it had a NULL height before it received the UPD. A node i at a NULL height receiving a UPD $[j, UPD, \delta_j, C_j]$ verifies if $C_i \subset C_j$ (in other words, i checks if all CEIs known to i are "addressed" by the UPD by j). Only if $C_i \subset C_j$, i sets its height to $\delta_i = \delta_j + 1$, propagates UPD $[i, UPD, \delta_i, C_i]$, and empties its CLR-list C_i. On the other hand, if $C_i \not\subset C_j$ node i will *retransmit* CLR $[i, CLR, C_i]$.

3.2.3. Other CLR Creation Scenarios

When a node i with a non-NULL height receives a UPD $[j, UPD, \delta_j, C_j]$. Assume that j was i's last downstream neighbor, *and* the height δ_j announced by j is such that $\delta_j \geq \delta_i$ (or this UPD causes i to loose its last downstream neighbor); Now i *creates* a CLR (with a new CEI). Such a scenario arises when the CLR sent by j is lost due to collision, and i creates an UPD assuming j as its downstream neighbor; which (UPD generated by i) is then propagated by j.

When a node i with a non-NULL height receives a CLR $[j, CLR, C_j]$ which leads to the loss its last downstream neighbor, i checks if any of the CEIs in C_j is also present in i's UPD-list $\mathcal{U}_i$. If not (or $C_j \cap \mathcal{U}_i = \emptyset$) i propagates CLR C_i. On the other hand, if $C_j \cap \mathcal{U}_i \neq \emptyset$, then i *creates* a new CEI. The new CEI is added to i's CLR-list C_i before i broadcasts CLR C_i. When a node creates a new CEI its CLR-list and UPD-list are emptied.

Thus, while a UPD is created by a node i only under one condition (after the T_W timer fires, if i retains a downstream neighbor), a CLR with a new CEI is created by a node i under three conditions:

i) i looses its last downstream link;
ii) i, which had a downstream neighbor j, receives a CLR which renders it with no downstream neighbor, *and* at least one of the CEIs that accompany the CLR packet by j is included in the UPD-list of i. Such a scenario can occur if i had prematurely created a UPD assuming that j is still downstream (the CLR sent by j earlier may have been delayed or lost due to collision).
iii) When i receives a UPD packet from its last "downstream" neighbor j, node i finds that $\delta_j \geq \delta_i$ (or i's belief that j was downstream is recognized to be wrong).

3.3. The CR-TORA Algorithm

In CR-TORA nodes react to some triggers like loss of last downstream link, firing of the T_W timer, or receipt of control packets like CLR, UPD and OPT, depending on the state of the node. The state of a node i is defined by i) latest OPT sequence number ϕ_q; ii) height - δ_i; iii) a neighbor-table with list of neighbors and their height (we shall represent the height of a neighbor j as H_j^i); iv) a CLR-list C_i, and v) a UPD-list $\mathcal{U}_i$. In addition, vi) a binary flag DN represents that the node has a down-stream neighbor; vii) a binary flag TR represents that the timer is currently running. In the pseudo-code describing the CR-TORA algorithm below, we shall use the notation $\delta_{i\_}$ to represent the height of a node i *before* it received a control packet (after the packet is processed, the height δ_i may change).

[2] As we shall see later in Section 4 of this paper, in our simulations the average number of CEIs that accompany a CLR or a UPD packet was found to be less than 2; thus the size of UPD and CLR packets in both TORA and CR-TORA are comparable.

1. Node i looses last downstream link:
CREATE CLR;flush $C_i, \mathcal{U}_i$;
2. Node i receives $[j, CLR, C_j]$**:**
$H_j^i = NULL$;
IF ($\delta_{i\_} == NULL$) return;
$C_i = C_j \cup C_i$;
IF (DN&(!TR)) start-timer;
IF (!DN)
IF ($C_j \cap \mathcal{U}_i == \emptyset$) PROPOGATE CLR;
ELSE {CREATE CLR;flush $C_i, \mathcal{U}_i$;}

3. Timer fires:
IF (DN)
{CREATE UPD;$\mathcal{U}_i = C_i$;flush C_i;}
4. Node i receives $[j, UPD, \delta_j, C_j]$**:**
$H_j^i = \delta_j$;
IF ($\delta_{i\_} == NULL$)
IF ($C_i \subset C_j$) PROPOGATE UPD;
ELSE RETX CLR;
ELSE IF (!DN)
{CREATE CLR;flush $C_i, \mathcal{U}_i$;}

An OPT packet relayed by a node i is of the form $[i, OPT, \delta_i, \phi_q]$, where ϕ_q is the OPT sequence number chosen by the destination.

5. i receives $[j, OPT, \delta_j, \phi_q{}']$
IF ($\phi_q{}' > \phi_q$)
{$\phi_q = \phi_q{}'$; $H_j^i = \delta_j$;
$\delta_i = \delta_j + 1$; flush $C_i, \mathcal{U}_i$
SEND $[i, OPT, \delta_i, \phi_q]$;}
IF ($\phi_q{}' = \phi_q$) $H_j^i = \delta_j$;

A node i receiving an OPT $[j, OPT, \delta_j, \phi_q{}']$ with a fresh sequence number, i) sets its height to $\delta_i = \delta_j + 1$; ii) retransmits the OPT; and iii) empties its CLR-list and UPD-list. The CLR-list and the UPD-lists of a node i are also flushed clear when a i creates a new CLR. When a node propagates a CLR its UPD-list is cleared.

4. Simulations

Simulations were carried out to evaluate the performance of TORA and CR-TORA using random realizations of subnet topologies with mobile nodes.

4.1. Simulation Environment

The simulating environment generates $N = 150$ randomly placed nodes in square region with edges of size 500 meters. The range of each node was assumed to be R meters (simulations were performed for $R = 55, 60, 65$). The destination is randomly chosen. Every node attempts to send data packets to the destination periodically - once every T_D seconds on an average (simulations were performed for $T_D = 1, 2, 3$ seconds). However, a node sends a data packet only if it has a non-NULL height. All simulation runs were performed for a network time of 100 seconds.

Mobility: To model mobility, at random instances of time some nodes were moved by random distances in X and Y directions (distance uniformly distributed between ±15 meters); some nodes were turned off; some of the nodes that are currently off were turned on and relocated at a random position. We simulated two mobility models. In the model M-I with lower mobility, on an average, during every second, i) $M_m = 7.5\%$ of the total number of nodes were moved; ii) $M_o = 3.75$ % of the nodes were turned off; and 50 % of nodes that are currently off are turned on. For the higher mobility model (Model M-II) the parameters were $M_m = 15\%$ and $M_o = 7.5\%$. In applying the mobility model, the only difference between the destination and the other nodes is that the destination is never turned off.

MAC Layer: The channel bit-rate was assumed to be 2Mb/s. Nodes employ p-persistent CSMA with $p \approx 1/20$. The carrier sense delay was assumed to be $\tau_{cs} = 1\mu sec$ sec. In other words, two nodes within the range of each other may not sense each other's transmission if they begin their transmissions within $\tau_{cs} = 1\mu sec$ of each other[3]. If a node senses that the channel becomes available at a time t, it begins its transmission at a time $t + x\tau_{cs}$ where x is random, and uniformly distributed between 1 and 20. A packet is received successfully by a node (without collision) only if not more than one of the receiver's neighbors (the sender of the packet) was transmitting during the entire duration of the packet. Most collisions at a node occur due to the "hidden station problem" - overlapping transmissions from neighbors of a node who are not within each other's range.

4.1.1. Control, Data and HELLO Packets

For both TORA and CR-TORA the destination sends OPT packets once every $T_{opt} = 5$ seconds. The duration of control packets were assumed to be 0.25 *msec* (about 64 bytes). The HELLO packets were $0.0625msec$ long (about 16 bytes). IMEP ACK packets (only for TORA) were also assumed to be of $0.0625msec$ duration. For both protocols we assumed a random processing delay in each node, uniformly distributed between 1 and $5msec$.

In TORA IMEP attempts to encapsulate multiple control/ACK packets into one IMEP packet. To offer a collision-free environment for TORA, IMEP sends ACKs for every control packet received. If a node has not heard an ACK from one or more of its neighbors within a time $T_{ack} = 20$ ms, it retransmits the control packet and explicitly indicates the identities of nodes which had not acknowledged the previous transmission. Only such nodes will need to send an ACK for the retransmitted packet. We limited the number of retransmissions to 2.

Data packets were of duration 1 *msec* (about 256 bytes). The maximum duration of IMEP packets was set at 1.0625 *msec* (272 bytes) to permit an ACK for a data packet to be sent along with the data packet. Only CSMA (no RTS/CTS handshake is used) was used even for data packet transmissions due to the relatively small size of packets.

In both TORA and CR-TORA a node A with multiple downstream neighbors chooses the one with the least height as the next hop to forward a data packet. If the data packet transmitted by A to a neighbor B is deemed unsuccessful, then A sends the data packet to its next downstream neighbor (if available). In both protocols a node A sending a data packet to a neighbor B attempts to overhear the retransmission of the data packet within a duration T_{ack}, failing which the data packet is retransmitted. The number of retransmissions are limited to 2 before the data transmission is deemed unsuccessful.

For maintaining dynamic list of neighbors every node tries to break silence once every T_s seconds. If a node A has not had the

[3] $1\mu sec$ is a conservative estimate given that the propagation delay for the maximum distance of 65 m is less than $0.25\mu sec$.

need to transmit a control or data packet (or ACK in TORA) in the last T_s seconds, A sends a HELLO packet to notify its presence to its neighbors. If a node A has not heard a transmission from a neighbor B for more that $2T_s$ seconds, the neighbor B is removed from A's neighbor table (if B happened to be A's last downstream link, link-failure route maintenance activities are triggered).

4.2. Results

Many simulation runs were performed; each run was for a network time of 100 seconds. The simulations were instantiated by the destination, by sending an OPT packet. Some of the parameters that were measured by the simulations were

a) $N_{tot} = N_{ctrl} + N_{opt} + N_{ack} + N_{dat}$: total number of packets: which is the sum total numbers of control packets, OPT packets, ACK packets, and data packets.
b) N_{tx}: total number of transmissions; for CR-TORA $N_{tot} = N_{tx}$; for TORA $N_{tx} < N_{tot}$ as packets queued for transmission can be aggregated by the IMEP layer.
c) n_{dat}: number of data packets instantiated by all nodes; note that $n_{dat} << N_{dat}$ as each of the unique n_{dat} data packets will need multiple transmissions/retransmissions over multiple hops.
d) n_{suc}: total number of data packets reaching the destination;
e) t_{lat}: average latency for data packets.
f) n_{ev}: total number of last-downstream-link-loss events (in both TORA and CR-TORA it is such events that trigger a sequence of localized control traffic).

Table 4.2 gives a detailed comparison of TORA and CR-TORA for one specific choice of parameters: viz, $T_{opt} = 5$, $R = 60$, and $T_D = 1$ second for two mobility models M-I and M-II.

Ideally, in the span of 100 seconds, each node should have created one data packet - or $100N = 15000$ data packets should have been created. However, as data packets are sent by a node only when they have a non-NULL height, the actual number of created data packets is less than 15,000. Ultimately the intent is to increase the number of packets received by the destination, while lowering the cost. In the TORA subnet for the mobility model M-I, the destination receives 7071 packets, compared to 8742 in the CR-TORA subnet. One simple measure of the "cost" is the total number of transmissions (by all nodes together). In the TORA subnet 624,869 TORA packets are aggregated into 446,861 IMEP transmissions compared to 231,169 packets (and the same number of transmissions as there is no aggregating lower layer) in CR-TORA. Furthermore, the average latency in TORA is 0.09 seconds while it is only 0.05 seconds in CR-TORA. Thus, CR-TORA outperforms TORA in every conceivable respect. CR-TORA results in a 15% higher throughput with a 40% reduction in latency, for 40% of the cost.

Data Packets: It is interesting to note that slightly more data packets were *created* in the TORA subnet ($n_{dat} = 13615$) compared to $n_{dat} = 13229$ for CR-TORA. The reason for this is TORA's long settling time during subnet partitions, during which nodes possess a non-NULL height while they do not actually have a physical path. This is also one reason for the substantially higher number of data packet transmissions N_{dat} in TORA compared to CR-TORA (about 1.7 times higher).

While both TORA and CR-TORA do not strive to determine the shortest path, the average path length for TORA is 9.9 hops compared to 7.1 for CR-TORA. This is in part due to the fact that a node does not consider a same height neighbor as downstream. Typically, CR-TORA heights reflect the actual number of hops between the node and the destination. Lower path lengths obviously lead to in lower latency and data traffic. Furthermore, the substantially higher *Ndat* in TORA also results in more packets being queued, leading to increased latency.

Route Maintenance Overhead: In both TORA and CR-TORA a series of route maintenance steps are trigerred by an event where a node looses its last downstream neighbor. The number of such events n_{ev} were also measured during the simulations. While both TORA and CR-TORA were simulated for the exact same network topology (and mobility) note that the value n_{ev} is lower in TORA (198) compared to CR-TORA (259); as CR-TORA does not consider a neighbor of the same height as downstream, the loss of the last downstream neighbor occured more often in CR-TORA. However, even while more maintenance activities are trigerred, CR-TORA mandates lower number of maintenance packets. Apart from data packets, the TORA subnet invoked $624869 - 312374 = 312,495$ "other" packets (HELLO, OPT, control packets like UPD and CLR, and IMEP ACK). In CR-TORA the number of "other packets" is substantially lower at $48,873$.

Effect of Mobility: As can be seen from Table 1 CR-TORA outperforms TORA by similar margins even for the scenario with increased mobility (model M-II instead of M-I). While increased mobility results in increased overhead and lower throughput for both TORA and CR-TORA, the increase in TORA overhead ($866082 - 624869 = 241,213$) is 10 times greater than the corresponding increase in CR-TORA ($254318 - 231,169 = 23,149$). This significant increase in TORA traffic also affects the latency t_{lat} in TORA which is increased by 0.02sec (compared to a mere $0.002sec$ increase in CR-TORA).

4.2.1. Effect of Network Density

In our simulations network density is controlled by adjusting the value of the range R. For the three of choices of $R = 55, 60, 65$ meters, the total number of nodes connected to the destination (averaged over many random realizations) are 94, 128 and 141 (63%, 85%, and 94%) respectively. As can be seen from Figure 1(a) CR-TORA offers greater throughput in all three cases.

Figure 1(b) depicts the maintenance traffic generated for different network densities. An increase in network density (and hence connectivity) will lead to lower number of maintenance events n_{ev} leading to lower maintenance traffic. However increased network density can also lead to higher probability of collisions, and consequently an increase in traffic due to retransmissions. The total number of control packets N_{ctrl} reduced with increased network density for CR-TORA; on the other hand, in TORA N_{ctrl} increases for high network densities due to a large number of collisions induced by IMEP retransmissions.

Increased connectivity resulted in an increase in the total number of data packets created (n_{dat}) in both TORA and CR-TORA, and consequently led to greater data traffic (N_{dat}). As can be seen from Figure 1(c) the increase in N_{dat} with network density

Table 1. TORA (T) vs CR-TORA (CR-T) for two mobility models M-I and M-II.

		N_{tot}/N_{tx}	n_{dat}	n_{suc}	N_{dat}	t_{lat}	n_{ev}
M-I	T	624,869/446,861	13,615	7071	312,374	0.09	198
M-I	CR-T	231,169	13,229	8742	182,296	0.05	259
M-II	T	866,082/632,243	13,261	6232	459,083	0.11	397
M-II	CR-T	254,318	12,917	7621	195,461	0.052	512

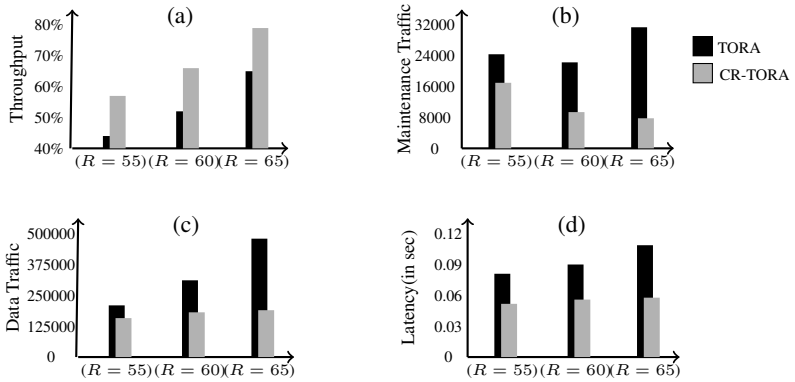


Fig. 1. Comparison of TORA and CR-TORA for different network densities; (a): Throughput; (b): Maintenance Traffic; (c): Data Traffic; (d): Latency.

is substantially higher for TORA, due to the higher number of collisions.

If we ignore collisions, one would expect lower latency with higher R due to a reduction in the number of hops. However increase in collisions with network density will create more retransmissions and thus increase the latency. As can be seen in Figure 1(d), in FR-TORA the two opposing effects almost balance out each other, causing only a very small increase in latency with increasing network density. On the other hand, due to the substantially higher number of collisions in TORA, the latency in TORA increases more rapidly with network density.

Recall that the TORA subnet generates more "useless" data packets (which will ultimately be undeliverable) as nodes possess invalid non-NULL heights for long durations under scenarios involving network partitions. We expect this phenomenon to be more pronounced for low density networks where network partitions are more likely to occur. As expected, for $R = 55$ the total number of data packets originated by the TORA and CR-TORA subnet are $n_{dat} = 13282$ and $n_{dat} = 11588$ respectively. The total number of data packets delivered are $n_{suc} = 5835$ for TORA and $n_{suc} = 6612$ for CR-TORA.

4.2.2. Waiting Time T_W

One new parameter introduced by CR-TORA is the waiting time T_W. Recall that if a node receiving a CLR packet has other downstream nodes it waits for a duration T_W before it sends a UPD. We experimented with various waiting times ranging from 10 to 20 *msec*. Our simulations show that a waiting time of 15 *msec* maximized throughput (fraction of data packets that reached the destination). However the performance of CR-TORA is not very sensitive to this parameter. Between 10 to 20 *msec* the worst and best case scenarios varied only by less than 5%.

5. Conclusions

For a connected ad hoc subnet with N nodes, most MANET routing protocols attempt to facilitate paths between all possible ($\binom{N}{2}$) pairs of nodes. This is however an overkill for many practical MANET applications where every node requires a path only to a single sink. TORA has been explicitly designed to facilitate multiple paths from all nodes to a single sink. While some authors have compared the performance of TORA with other routing protocols with mixed results [10, 11, 12], such efforts often unfairly assume an application setting with the need to establish paths between *all* $\binom{N}{2}$ pairs of nodes, thereby ignoring scenarios for which TORA is uniquely well suited (a small number of mobile/fixed destinations and a large number of mobile nodes).

The main pitfall of TORA is its reliance on an expensive lower layer. As is evident from the simulation results presented in Section 4, IMEP retransmissions following collisions is the primary reason for poor performance of TORA - both in terms of increase in latency and the number of transmissions. Unfortunately, it is not possible to use TORA without IMEP as collisions can create loops in the TORA routing protocol [7].

The main motivations for CR-TORA were two-fold: that the goal of TORA is an important one for many application scenarios, and that there are compelling reasons to eliminate the need for an expensive lower layer. Consequently, the primary difference between CR-TORA and TORA is that CR-TORA has inbuilt features that address collisions, and thereby eliminates the need for IMEP. Apart from eliminating the need for a lower layer the other beneficial properties of CR-TORA are i) lower settling time during network partitions (leading to less instances of "useless" data traffic); ii) lower latency due to shorter path lengths, which is primarily attributable to the fact that same height neighbors are not considered as downstream; and iii) lack of the need for time-synchronization.

Simulations show that, compared to TORA, CR-TORA reduces the number of total number of transmissions by 60%; results in a 15% increase in throughput; and a 40% reduction in latency. CR-TORA is thus a promising protocol for many practical MANET/sensor networks.

References

[1] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, pp. 46–55, 1999.

[2] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *INFOCOM*, pp. 1405–1413, 1997.

[3] V. Park and S. Corson, "Temporally ordered routing algorithm (tora) version 1 functional specification," in *IETF MANET, Internet Draft*, 2001.

[4] P.Johanson and D. Maltz, *Dynamic source routing in ad hoc wireless networks*. Kluwer Publishing Company, 2001.

[5] C. Perkins, E.Royer, and S. Das, *Ad hoc On-Demand Distance Vector (AODV) Routing*. Aug 2002. Internet Draft, draft-ietf-manet-aodv-11.txt.

[6] C. Perkins and P. Bhagvat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *ACM SIGCOMM Symposium on Communication, Architectures and Applications*, 1994.

[7] E. Weiss, G. R. Hiertz, and B. Xu, "Performance analysis of temporally ordered routing algorithm based on ieee 802.11a," in *IEEE 61st Vehicular Technology Conference (VTC)*, May 2005.

[8] M. S. Corson, S. Papademetriou, P. Papadopoulos, V. D. Park, and G. Qayyum, "An intemet manet encapsulation protocol (imep) specification," in *Interact-Draft*, August 1998. draft-ieff-manet-imep-spee- 01.txt.

[9] E. M. Gafni and P. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," *IEEE Transactions on Communications*, vol. 29, pp. 11–18, 1981.

[10] V. Park and S. Corson, "A performance comparison of the temporally-ordered routing algorithm and ideal link-state routing," *Proceedings of IEEE International Symposium on Systems and Communications, IEEE Computer*, pp. 592–598, 1998.

[11] Q. He, H. Zhou, H. Wang, and L. Zhul, "Performance comparison of two routing protocols based on wmn," in *2nd International Conference on Wireless Broadband and Ultra Wideband Communications*, (Sydney, Australia), 2007.

[12] J. Broch, D., A. Maltz, D. B. Johnson, Y.-C. Hu, and J. G. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *In Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, (MobiCom 98)*, pp. 85–97, October 1998.

Mahalingam Ramkumar is an Associate Professor in the Department of Computer Science and Engineering, Mississippi State University. He received his Ph.D. in Electrical Engineering from New Jersey Institute of Technology, Newark, NJ, (2000), MS in Electrical Communication Engineering from Indian Institute of Science, Bangalore, India (1997) and Bachelor's Degree in Engineering from University of Madras, India (1987). His research interests include Trustworthy Computing, Network Security, Cryptography, and Mobile Ad hoc Networks.

Vinay Thotakura completed his B-tech degree in Computer Science with distinction from VR-SEC, Nagarjuna University, India, in 2005. He received his M.S. degree in Computer Science from Mississippi State University, MS, in 2009. He is currently a research assistant and Ph.D. candidate in the Department of Computer Science and Engineering at Mississippi State University. He research interest is securing MANET routing protocols using trustworthy MANET modules.