Performance Enhancement of the Temporally-Ordered Routing Algorithm

[Honours Thesis]

Kwan Hui Lim
School of Computer Science and Software Engineering
The University of Western Australia
Crawley, WA 6009, Australia
kwanhui@graduate.uwa.edu.au

ABSTRACT

The Temporally-Ordered Routing Algorithm (TORA) is a highly adaptive distributed routing algorithm used in Mobile Ad hoc Networks (MANET) that is able to provide multiple loop-free routes to a destination [1, 2]. TORA is very dependent on the services provided by the Internet MANET Encapsulation Protocol (IMEP) to effectively carry out its three main functions: Route Creation, Route Maintenance, and Route Erasure. However, TORA does not scale well in networks with a large number of traffic connections compared to other MANET protocols. We performed a comprehensive analysis into the workings of IMEP to determine which behaviour of IMEP leads to the poor performance of TORA and narrowed down the problem in IMEP. We suggested two approaches to improve the operations of IMEP: by increasing the maximum number of object block message retransmissions and by using a random RETRANS_PERIOD. In addition, two different variations of TORA were developed to solve some of its performance problems: the network localization approach and selective node participation approach. The IMEP modifications and TORA variations resulted in an overall improvement in terms of packet delivery, routing overhead and packet latency in a variety of network scenarios.

Categories and Subject Descriptors

I.6.0 [**Simulation and Modelling**]: General; C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Algorithms, Experimentation

Keywords

TORA, IMEP, Wireless Networks, Simulations, Ad Hoc Protocols

1. REFERENCES

[1] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *INFOCOM '97: Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, pages 1405–1413, Washington, DC, USA, 1997. IEEE Computer Society.

[2] V. D. Park and M. S. Corson. Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification. Internet Draft, July 2001. Available from: http://tools.ietf.org/id/draft-ietf-manet-tora-spec-04.txt.

Performance Enhancement of the Temporally-Ordered Routing Algorithm

Kwan Hui Lim

This report is submitted as partial fulfilment
of the requirements for the Honours Programme of the
School of Computer Science and Software Engineering,
The University of Western Australia,
2007

Abstract

The Temporally-Ordered Routing Algorithm (TORA) is a highly adaptive distributed routing algorithm used in Mobile Ad hoc Networks (MANET) that is able to provide multiple loop-free routes to a destination. TORA is very dependent on the services provided by the Internet MANET Encapsulation Protocol (IMEP) to effectively carry out its three main functions: Route Creation, Route Maintenance, and Route Erasure. However, TORA does not scale well in networks with a large number of traffic connections compared to other MANET protocols. We performed a comprehensive analysis into the workings of IMEP to determine which behaviour of IMEP leads to the poor performance of TORA and narrowed down the problem in IMEP. We suggested two approaches to improve the operations of IMEP: by increasing the maximum number of object block message retransmissions and by using a random RETRANS_PERIOD. In addition, two different variations of TORA were developed to solve some of its performance problems: the network localization approach and selective node participation approach. The IMEP modifications and TORA variations resulted in an overall improvement in terms of packet delivery, routing overhead and packet latency in a variety of network scenarios.

Keywords: TORA, IMEP, Wireless Networks, Simulations, Ad Hoc Protocols
CR Categories: I.6 Simulation and Modelling, C.2.2 Network Protocols

Acknowledgements

I would like to extend my deepest appreciation to Prof. Amitava Datta for his constant encouragement, motivation and enlightening discussions. I wish to also thank my family, friends, and girlfriend for all of their support. This thesis would not have been possible without all of you.

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

A MANET consists of mobile nodes which are capable of wireless communications and moving in a random fashion [4]. The connectivity between the nodes forms the topology of the MANET and one of its characteristics is its dynamic topology due to the unpredictable and sudden movement of nodes. Another characteristic is the lower capacity of wireless links compared to their hardwired counterparts thus increasing the likelihood of traffic congestions. TORA [13, 12] is a highly adaptive distributed routing algorithm that is designed to work in such a network but its correct operation is dependent on certain services provided by IMEP [3].

TORA has been shown to perform well in terms of packet delivery for small networks with a small number of traffic connections but its performance deteriorates drastically with an increase in traffic connections [1, 2]. This problem has been attributed to certain characteristics of IMEP and how TORA contributes to these characteristics. We tackle the problem by first correcting the behaviour of IMEP and then modifying the TORA protocol.

The problem with IMEP is that it is prone to incorrectly detect link failures when there are many traffic connections. This incorrect detection of link failures causes TORA to react and results in additional routing overhead that prevents data packet from being sent. In addition, in high mobility networks, TORA suffers from a high routing overhead as the frequent link failures require it to constantly find alternate routes. However, TORA has an advantage over other on-demand routing protocols as it is able to maintain multiple routes to a destination. In addition, the protocol's localized reaction to link failures makes it highly scalable for implementation in large networks.

1.1 Overview of TORA

TORA is a distributed routing algorithm that is based on a family of link reversal algorithms and is able to provide multiple loop-free routes to any destination on-

demand. The availability of multiple paths is a result of how TORA models the entire network as a directed acyclic graph (DAG) rooted at the destination. Each node has a height associated with it and links between nodes flow from one with a higher height to one with a lower height. The collection of links formed between nodes forms the DAG and ultimately all nodes will have a route to the destination. For each possible destination required, a separate DAG needs to be constructed.

1.2 Related Work

1.2.1 A Query Localization Approach

Dharmaraju [7] proposed a *Query Localization* approach to improve the performance of TORA. This is based on the argument that TORA initializes a DAG for the entire network during route creation and this DAG includes nodes that never participate in communication. Subsequently, the maintenance of the DAG causes unnecessary routing overhead due to the maintenance of links to these nodes that never participate in communication.

The *Query Localization* approach involves the propagation of QRY packets in an expanding ring manner during route creation. This means that QRY packets are propagated for one hop, then two hops, and so on until it reaches the destination. The destination then replies with an UPD packet that establishes links only on nodes that previously received the QRY packet and initializes a DAG that is a portion of the entire network. If a node that is outside of this DAG suffers from a link breakage, it is unnecessary to initiate route maintenance and this results in less routing overhead being generated.

Dharmaraju also showed that the *Query Localization* approach results in a significant reduction of overhead (about 50%) over the original TORA. However, there were no experiments or results to compare its performance against original TORA in terms of packet delivery. The packet delivery metric is an equally important criterion in evaluating the performance of a protocol and should be included to give a thorough evaluation of the improvement achieved.

1.2.2 Routing Loops and Packet Queuing

Weiss, Hiertz, and Xu [18] pointed out that loops could be formed in TORA when control messages are not immediately delivered. They recommended a solution which involves IMEP providing TORA with additional information about how a

new neighbouring node is being discovered. This additional information enables TORA to determine whether or not to send an UPD packet which will remove any possible loop. Similarly, Das, Castañeda, and Yan [5] noted the possibility of short-lived inconsistencies about the direction of links in TORA that leads to loops as well.

Weiss, Hiertz, and Xu also noted that nodes constantly move about causing the temporary unavailability of routes to a destination. They suggested the intermediate queuing of data packets which allows packets to be queued for a short amount of time while routes could be discovered and subsequently packets sent.

1.3 Thesis Outline

Firstly, we introduce the TORA protocol and IMEP in Chapter 2 and describe the setup of the simulations in Chapter 3. Chapter 4 covers the analysis of IMEP and some of the approaches to overcome its problems. We recommend some modifications to TORA to enhance its performance in Chapter 5. In Chapter 6, we evaluate the performance of the modifications we recommended before concluding the thesis in Chapter 7.

CHAPTER 2

Background Information

TORA requires the services provided by IMEP to function properly. With reference to the Open Systems Interconnection (OSI) Model [6], IMEP sits below TORA with both protocol being at the network layer. In turn, IMEP uses the services provided by the Institute of Electrical and Electronics Engineers (IEEE) 802.11 Medium Access Control (MAC) protocol [8], a data link layer protocol. Both TORA and IMEP are covered in this chapter.

2.1 Temporally-Ordered Routing Algorithm

TORA is a distributed, on-demand routing protocol based on a link reversal algorithm and is able to provide multiple loop-free routes to a destination. The protocol is highly adaptive as it minimizes the reaction to link failure by using only a localized single pass of the distributed protocol. The following description of the TORA protocol is taken from Park and Corson [13, 12] who first proposed the protocol.

2.1.1 Notation and Assumptions

TORA models the network as a graph $G = (N, L)$ where N refers to the set of nodes and L refers to the set of links. Links are created and broken as nodes move about thus the set of links L changes with time. The following discussion of TORA are based on the following assumptions:

- Each node is identified by a unique node identifier (ID).
- Links are bi-directional (i.e. nodes that share a link can communicate with each other).
- The use of a lower level protocol that is able to:

- Ensure that nodes are always aware of their neighbours.
- Provide reliable and in-sequence delivery of packets.
- Use broadcasting for packet transmission (i.e. all neighbours of a node will receive a packet that is transmitted by the node).

In Section 2.2, we look at IMEP which is the lower level protocol that provides TORA with all these services.

Each link $(i, j) \in L$ is initially unassigned but may subsequently be assigned one of the following states:

- undirected.
- directed from node i to node j.
- directed from node j to node i.

Consider a link $(i, j) \in L$ that is directed from node i to node j, node i is termed the "upstream" neighbour of node j while node j is termed the "downstream" neighbour of node i. Correspondingly, the link (i, j) is termed the "downstream" link of node i and the "upstream" link of node j.

2.1.2 Basic Structure

For each destination that requires routing to, a logically separate version of TORA is executed by each node. The protocol performs three main functions: route creation, route maintenance, and route erasure. These three functions are facilitated by the use of three different control packets: query (QRY), update (UPD), and clear (CLR) packets.

When a node requires a route to a destination, route creation is initiated where QRY and UPD packets are used to establish directions on previously undirected links. This establishment of links results in a destination-oriented directed acyclic graph (DAG) where any packet sent will reach the destination as it is the only node with no downstream links. Topological changes caused by node mobility may cause a node to lose all of its downstream links and hence, its route to the destination. With the use of UPD packets, route maintenance ensures that the DAG is re-established to become destination-oriented within a finite time. In the event that a network partition is detected, route erasure uses CLR packets to ensure that all invalid routes are erased by setting all links in the partition to be undirected.

2.1.3 Description

Each node $i \in N$ is responsible for maintaining its height H_i at all times. The height is represented by an ordered quintuple $H_i = (\tau_i, oid_i, r_i, \delta_i, i)$ and the values have the following representation:

1. τ_i, is the "logical time" of the link failure.
2. oid_i, is the unique ID of the node which defined the new reference level.
3. r_i, is a single bit that divides the reference level into two sub-levels: non-reflected or reflected.
4. δ_i, is an integer used to order nodes with respect to the same reference level.
5. i, is the unique ID of the node itself.

The height of the node is defined by two parameters: a reference level and a delta with respect to the reference level. The first three values in the quintuple represent the reference level while the last two values represent the delta.

Each node i (other than the destination) maintains a link-state array with an entry $LS_{i,j}$ for each link $(i, j) \in L$, where j is a neighbour of i. The height of node i, H_i and that of its neighbour j, $HN_{i,j}$ determines the direction of the links, and is directed from the higher node to the lower node. If node i has a non-NULL height, it labels the link $LS_{i,j}$ as:

- Upstream (UP), if a neighbour j has a height higher than that of node i.
- Downstream (DN), if a neighbour j has a height lower than that of node i.
- Undirected (UN), if a neighbour j has a NULL height.

If node i has a NULL height and its neighbour j has a non-NULL height, the link $LS_{i,j}$ is labelled as downstream (DN) as its neighbour is considered lower.

Initially, all links are undirected thus all nodes in the network have a height set to NULL, $H_i = (-, -, -, -, i)$. An exception is the destination did whose height is always ZERO, $H_{did} = (0, 0, 0, 0, did)$. Each neighbour of the destination $j \in N_{did}$ sets the height $HN_{j,did}$ to ZERO, $HN_{i,did} = (0, 0, 0, 0, i)$ and labels their link $LS_{j,did}$ as downstream.

When a node i discovers a new neighbour $j \in N_i$, it establishes the new link $(i, j) \in L$ by adding a new height $HN_{i,j}$ and link-state $LS_{i,j}$ entry for neighbour j. If the new neighbour is not the destination, the height entry is set to NULL,

$HN_{i,j} = (-, -, -, -, i)$ and the link-state is labelled as undirected. If the new neighbour is the destination, the height entry and link-state is set as outlined above.

Route Creation

Route creation involves the process of requesting and replying of routes, which are achieved by the use of QRY and UPD packets respectively. A QRY packet consist of a destination-ID (*did*), which identifies the destination that requires routing to. An UPD packet consist of a *did* and the current height of the node broadcasting the packet. Each node i (other than the destination) maintains three values:

1. a route-required flag, RR_i, which is initially set to 0. A route-required flag is maintained for each destination.
2. the time at which the last UPD packet was broadcast.
3. the time at which each link (i, j) $\in L$, where $j \in N_i$, became active.

The route-required flag indicates whether a node is currently waiting for a route to the destination. The time in item 2 and 3 can be local to that particular node as it does not have to be synchronized between different nodes.

When a node needs a route to a destination (i.e. it has no directed links), it broadcasts a QRY packet and sets its route-required flag to 1. When a node i receives a QRY packet, it does one of the following:

- If the receiving node has no downstream neighbours and its route-required flag is 0, it sets its route-required flag to 1 and re-broadcasts the QRY packet. In this case, the receiving node does not have a route and attempts to find a route by propagating the QRY packet further.
- If the receiving node has no downstream neighbours and its route-required flag is 1, it ignores the QRY packet. This means that a QRY packet has already been broadcast and the receiving node is currently waiting for a route.
- If the receiving node has at least one downstream link and its height is NULL, it sets its height to $H_i = (\tau_j, oid_j, r_j, \delta_j + 1, i)$, where $HN_{i,j} = (\tau_j, oid_j, r_j, \delta_j, j)$ is the minimum height of its downstream neighbours, and broadcast an UPD packet. In this case, the receiving node does not have

a valid route but is able to route through one of its neighbours to the destination.

- If the receiving node has at least one downstream link and its height is non-NULL, it compares the time at which the last UPD packet was broadcast to the time which the link over which the QRY packet came from became active. If no UPD packet has been broadcast after the link over which the QRY packet came from became active, it broadcasts an UPD packet. Otherwise, it ignores the QRY packet as an UPD packet has already been broadcast.

Similarly, when a node i receives an UPD packet from a neighbour $j \in N_i$, node i first updates the height entry $HN_{i,j}$ in its height array with the height contained in the UPD packet it received and either:

- If its route-required flag is 1, node i sets its height to $H_i = (\tau_j, oid_j, r_j, \delta_j + 1, i)$, where $HN_{i,j} = (\tau_j, oid_j, r_j, \delta_j, j)$ is the minimum non-NULL height of its neighbours. Then, it updates all entries in its link-state array LS, reset the route-required flag to 0 and broadcast an UPD packet containing its new height.
- If its route-required flag is 0, node i updates the entry $LS_{i,j}$ in its link-state entry. This may cause node i to lose its last downstream link thus initiating route maintenance.

Route Maintenance

A node i that has lost all of its downstream links will initiate route maintenance if it has a non-NULL height. A node i has no downstream link if $H_i < HN_{i,j}$ for all non-NULL neighbours $j \in N_i$. Route maintenance is necessary to re-orient the DAG so that it becomes rooted at the destination but it is only performed for nodes that have a non-NULL height and no NULL heights are used for the computation. This process involves node i modifying its height, $H_i = (\tau_i, oid_i, r_i, \delta_i, i)$ according to one of the cases:

Case 1 (Generate): Node i has no downstream links due to a link failure.

- $(\tau_i, oid_i, r_i) = (\tau, i, 0)$, where t is the time of the link failure
- $(\delta_i, i) = (0, i)$

This case assumes that node i has at least one upstream neighbour and it defines a new reference level as its upstream neighbour needs to route through it to the destination. If node i has no upstream neighbour, it sets its height to NULL.

Case 2 (Propagate): Node i has no downstream links (due to the receipt of an UPD packet that caused link reversal) and the ordered sets (τ_j, oid_j, r_j) are not equal for all $j \in N_i$.

- $(\tau_i, oid_i, r_i) = max\{\tau_j, oid_j, r_j \mid j \in N_i\}$, where t is the time of the link failure
- $(\delta_i, i) = (min\{\delta_j\} - 1, i$, where δ_j is chosen from $j \in N_i$ with $(\tau_j, oid_j, r_j) = max\{(\tau_j, oid_j, r_j)\}$

Node i propagates the reference level of its highest neighbour and selects a height that is lower than all of its highest neighbours by using a lower delta. Since the reference levels are not equal for all $j \in N_i$, node i is actually performing partial link reversal where links to highest neighbours are maintained while links to all other neighbours are reversed.

Case 3 (Reflect): Node i has no downstream links (due to the receipt of an UPD packet that caused link reversal) and the ordered sets (τ_j, oid_j, r_j) are equal with $r_j = 0$ for all $j \in N_i$.

- $(\tau_i, oid_i, r_i) = (\tau_j, oid_j, 1)$
- $(\delta_i, i) = (0, i)$

In this case, all the neighbours of node i have the same reference level that is of a non-reflected sub-level (i.e. $r_j = 0$ for all $j \in N_i$). It propagates this reference level as a reflected sub-level by setting the bit r_i. This case facilitates the detection of a network partition in Case 4 (Detect).

Case 4 (Detect): Node i has no downstream links (due to the receipt of an UPD packet that caused link reversal), the ordered sets (τ_j, oid_j, r_j) are equal with $r_j = 1$ for all $j \in N_i$ and $oid_j = i$.

- $(\tau_i, oid_i, r_i) = (-, -, -)$
- $(\delta_i, i) = (-, i)$

In this case, a network partition has been detected as the last reference level defined by node i has been propagated back as a reflected sub-level by all of its neighbours. This means that node i initially tried to find a route through its neighbours but now, its neighbours are trying to find a route through node i. This indicates a network partition and route erasure must be initiated. By restricting the task of detecting network partitions to the originator of the reference level, the probability of a false detection is reduced significantly.

Case 5 (Generate): Node i has no downstream links (due to the receipt of an UPD packet that caused link reversal), the ordered sets (τ_j, oid_j, r_j) are equal with $r_j = 1$ for all $j \in N_i$ and $oid_j \neq i$.

- $(\tau_i, oid_i, r_i) = (t, i, 0)$, where t is the time of the link failure
- $(\delta_i, i) = (0, i)$

This means that node i experienced a link failure (but did not lose its last downstream link) between the time it propagated a reference level and all of its neighbours returned with a reflected sub-level. Node i propagates a new reference level as this does not indicate a network partition.

When cases 1, 2, 3, and 5 cause the computation of a new height, node i updates all the entries in its link-state array LS as the state of some links would have changed. Following which, node i broadcasts an UPD packet to all of its neighbours $j \in N_i$. When a neighbour j receives an UPD packet from node i, it updates the height $HN_{j,i}$ and link-state $LS_{j,i}$ entries. This subsequent broadcasting of UPD packets may cause a node i to lose all of its downstream links, thus requiring node i to modify its height as outlined in the cases above.

Route Erasure

Route erasure is initiated upon the detection of a partition (Case 4). Node i sets its height and the height entry for each neighbour $j \in N_i$ to NULL. However, if the destination is a neighbour, the corresponding height array is set to ZERO. Node i then updates its link-state array LS, and broadcasts a CLR packet that consists of a destination id, *did* and the reflected sub-level of node i, $(\tau_i, oid_i, 1)$. When a node i receives a CLR packet from a neighbour $j \in N_i$, it reacts according to one of the following:

1. If the reference level in the CLR packet matches that of node i, it sets its height and the height entry for each neighbour $j \in N_i$ to NULL (unless the

destination is a neighbour, in which case the corresponding height array is set to ZERO), updates all the entries in its link-state array *LS*, and broadcasts a CLR packet.

2. If the reference level in the CLR packet does not match that of node i, it sets the height entry for each neighbour $j \in N_i$ (with the same reference level in the CLR packet) to NULL and updates the corresponding link-state array entries.

This effectively erases all invalid routes since the height of all nodes in the network partition is set to NULL. If node i loses its last downstream link due to Condition 2, it reacts according to Case 1 of route maintenance.

2.2 Internet MANET Encapsulation Protocol

IMEP provides services that TORA requires such as Link/Connection Status Sensing and Broadcast Reliability. It also performs Message Aggregation to reduce the amount of overhead produced by TORA. Some of the other services that IMEP provide are Network-layer Address Resolution, Multipoint Relaying and Authentication. These other services are not as crucial to the performance of TORA as the former three and hence they are not discussed. The following description of IMEP is based on the internet draft of IMEP which was developed by Corson *et al.* [3].

2.2.1 Message Aggregation

Message Aggregation is the process by which IMEP encapsulates packets that are passed down by TORA and its own routing control packets into a single object block message. This minimizes the number of channel accesses needed since a single object block message is sent instead of multiple, smaller IMEP and TORA packets.

2.2.2 Link/Connection Status Sensing

Link/Connection Status Sensing provides TORA with accurate and up-to-date information on the status of links of a node to its neighbours. This mechanism is also able to determine if a link is bi-directional or uni-directional and TORA requires links to be bi-directional. This mechanism operates based on two methods of detecting link failures, namely the explicit method and implicit method.

Explicit Link Failure Detection Method

The explicit method of link failure detection that IMEP uses to determine link status information operates by having a node i to broadcast BEACON packets to its one-hop neighbours. When node i receives a reply in the form of an ECHO packet from a neighbour, it labels the link to that neighbour as bi-directional. Node i continues to send BEACON packets at every BEACON_PERIOD interval and sets a MAX_BEACON_TIME. This MAX_BEACON_TIME is made up of the maximum number of BEACON retransmissions multiplied by the BEACON_PERIOD. If it does not hear any ECHO packet after the MAX_BEACON_TIME, it labels that link as down and notifies TORA.

Implicit Link Failure Detection Method

The implicit method of link failure detection involves making use of other packets that IMEP sends such as the object block message created after Message Aggregation. In a network where there are many traffic connections, TORA sends packets down to IMEP for transmission frequently. IMEP then performs Message Aggregation before sending out the single object block message. Nodes that receive this object block message then reply with an ACK packet. The sending of the object block messages and replying with ACK packets mirrors that of the BEACON and ECHO packet used in the explicit method discussed earlier. Similarly, there is a MAX_RETRANS_TIME made up of the RETRANS_PERIOD between each retransmission multiplied by the maximum number of retransmissions. After the MAX_RETRANS_TIME, any neighbour that did not reply with an ACK packet has its link labelled as down and TORA will be notified.

2.2.3 Broadcast Reliability

Broadcast Reliability can be any mix of two delivery modes of broadcast or multicast and also two reliability modes of reliable or unreliable. Specifically, TORA requires Broadcast Reliability in the reliable and broadcast mode. The reliable mode is important for in-sequence delivery of messages and this is achieved by attaching a sequence number to each object block message produced after Message Aggregation. Any out-of-sequence object block message that arrives at a node is placed in a queue and passed to TORA only after the object block messages of the missing sequence numbers arrive. The broadcast mode requires all neighbouring nodes to acknowledge any object block message sent and this facilitates the implicit method of link failure detection in Link/Connection Status Sensing.

CHAPTER 3

Methods

In this chapter, we elaborate on how the performance of TORA/IMEP is analysed and the various simulation settings used. We briefly explain the measurement metrics used followed by a description of the approach we took to enhance the performance of TORA.

3.1 Simulation Environment

We used the *ns-2* network simulator [17] for our simulations. *Ns-2* is a discrete event simulator that allows the modelling of a variety of protocols over wired, wireless and satellite networks. It has the added advantage of a controlled environment where we can examine the characteristics of IMEP and TORA. The radio propagation properties of the mobile nodes are modelled after the Lucent WaveLan direct sequence spread spectrum radio [16].

The random waypoint model [10] is used to define the movement scenarios for nodes in the network. In this model, nodes start off at a random position before moving to a random destination at a random speed between 0 and some pre-defined maximum. Upon reaching the destination, it stops for a certain *pausetime* before moving to another random destination. This takes place for the entire duration of the simulation, which we set at 900 seconds.

We modelled nodes with a maximum speed of 1 m/s or 20 m/s, representing a low and high mobility network respectively. The topology size of the network was chosen at $1500m \times 300m$ for a network of 50 nodes, and $2121m \times 425m$ for a network of 100 nodes to maintain a node density of approximately $9000m^2$ per node. The pause times chosen were 0, 30, 60, 120, 300, 600, and 900 seconds where 0 and 900 represents constant mobility and no movement respectively. These simulation parameters are identical to that used by Broch *et al.* [1].

Traffic connections in the network are modelled using constant bit rate (CBR) sources where packets are sent to a particular destination at a constant rate until

BEACON_PERIOD	1 s
Maximum number of BEACON retransmissions	3 times
MAX_BEACON_TIME	3 s
object block message RETRANS_PERIOD	0.5 s
Maximum number of object block message retransmissions	3 times
MAX_RETRANS_TIME	3 s
Minimum message aggregation delay	150 ms
Maximum message aggregation delay	250 ms

Table 3.1: IMEP constants

the end of the simulation. We used CBR packets with a size of 64 bytes and sending rate of 4 packets per second.

As IMEP [3] does not specify the values for the constants used, we chose the values used by Broch *et al.* [1] for the purpose of benchmarking the performance of original TORA against our modified versions of IMEP and TORA. The values are as shown in Table 3.1.

3.2 Performance Measurement

The simulated protocols were evaluated using three metrics: packet delivery ratio, routing overhead, and average latency. Packet delivery ratio is derived from the total number of data packets received over the total number of data packets sent while routing overhead is calculated as the number of routing packets required per data packet sent. Average latency measures how long packets take on average to reach their destination from the originator.

3.3 Experimental Approach

A two-pronged approach is used to improve the performance of TORA: at the IMEP layer and the TORA layer itself. We first analyse IMEP in detail and determine how and why it fails in the detection of link failures. Next, we recommend approaches to modify IMEP such that those inefficiencies are corrected and show results that support our recommendations. The two approaches that we recommend are by using a random RETRANS_PERIOD and by increasing the maximum number of object block message retransmissions. Both approaches resulted in an improvement in packet delivery, routing overhead and average la-

tency for various scenarios with the latter approach performing better. These comparisons were performed on the original TORA protocol but using different modifications of IMEP. Correcting the behaviour of IMEP allowed us to concentrate on solving the problems faced by TORA on the basis that the underlying layer performs satisfactorily.

Using the corrected IMEP, we present two approaches to modify the TORA protocol itself: a network localization approach and a selective node participation approach. Both approaches solve the problem of high routing overhead in route maintenance by maintaining only a subset of the network as the DAG. Both approaches perform well with a large number of traffic connections with the network localization approach performing better. However, the network localization approach suffers from a drop in performance with a small number of traffic connections while the selective node participation approach maintains its performance.

CHAPTER 4

In-depth Analysis and Improvement of IMEP

In this chapter, we analyse IMEP, particularly in the way it detects link failures. Modifications to the existing IMEP protocol are suggested and its performance evaluated based on how TORA performs using the original IMEP against our modified IMEP.

Broch *et al.* and Broustis *et al.* [1, 2] noted that the loss of data, ACK, and BEACON packets due to heavy traffic causes IMEP to incorrectly detect link failures. There has been no study that shows how this error occurs or its impact on the performance of TORA. We examine that and suggest modifications to IMEP that result in TORA performing better.

4.1 Analysis of Explicit Link Failure Detection

One way that IMEP can incorrectly detect a link to have failed is when it does not receive ECHO packets in response to a BEACON packet it sent out. In a network with many traffic connections, IMEP is able to detect the status of links implicitly since many object block messages will be sent throughout the entire network. We prove this by showing that the explicit method of link failure detection has minimal effect on TORA.

4.1.1 Increasing the Maximum Number of BEACON Retransmissions

If the incorrect detection of link failures is caused by the inability of BEACON packets to reach nodes (thus they do not reply with ECHO packets), increasing the maximum number of BEACON retransmission solves the problem. This increases both the MAX_BEACON_TIME and the probability that nodes can

receive BEACON packets and subsequently reply with ECHO packets. However, our results show that even with that modification, TORA does not obtain any significant improvement regardless of network size, node mobility or the number of traffic connections simulated.

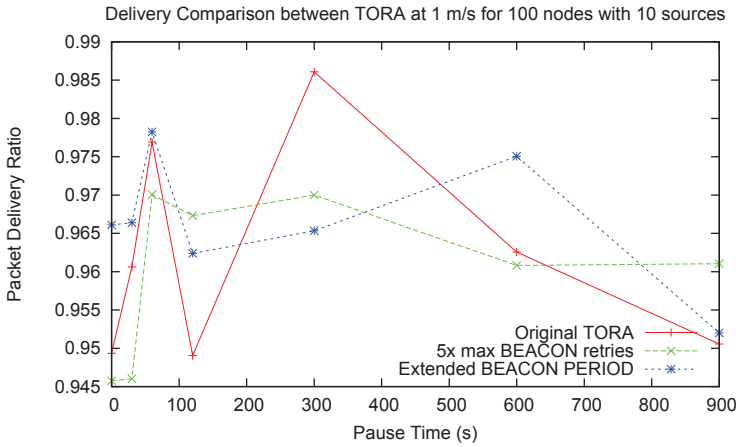

Figure 4.1: 100 nodes with 10 traffic connections at 1 m/s for the explicit method

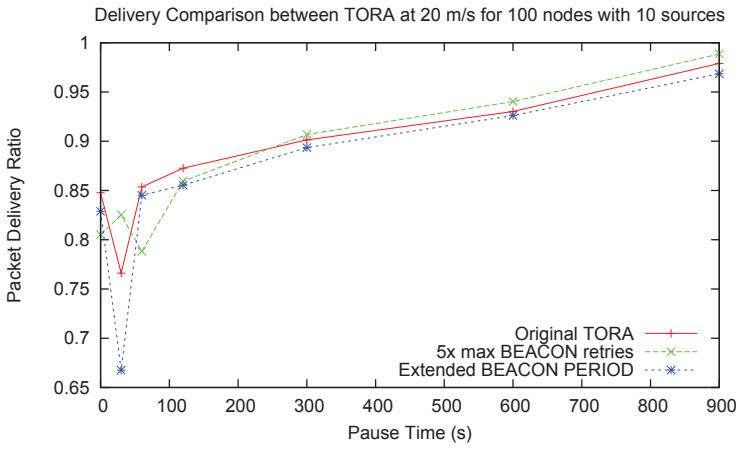

Figure 4.2: 100 nodes with 10 traffic connections at 20 m/s for the explicit method

Fig. 4.3 and 4.4 show that improvements of up to 80% can be observed for large networks with 30 traffic connections but this is only when the network is

static with a pause time of 900 seconds. Even with constant mobility at a low speed of 1 m/s, there is little performance improvement from this modification as shown in Fig. 4.3.

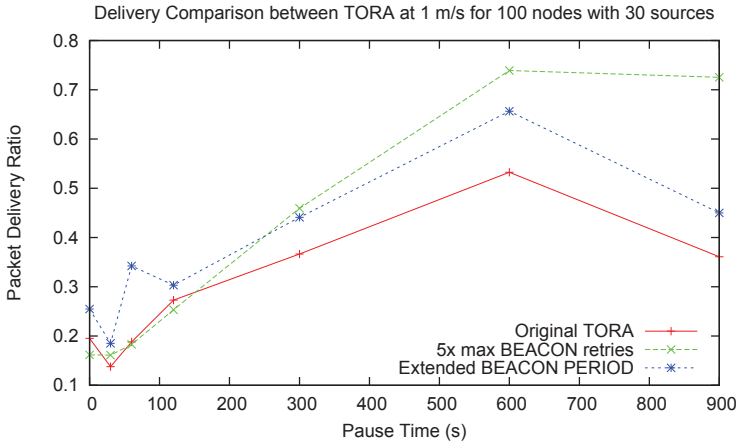

Figure 4.3: 100 nodes with 30 traffic connections at 1 m/s for the explicit method

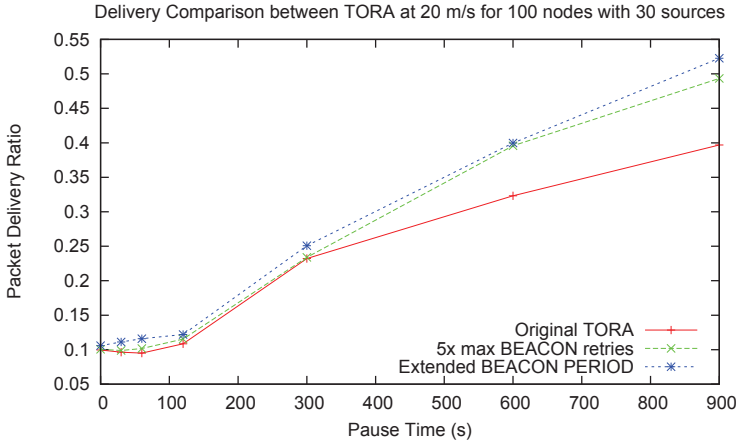

Figure 4.4: 100 nodes with 30 traffic connections at 20 m/s for the explicit method

4.1.2 Increasing BEACON_PERIOD

Increasing the BEACON_PERIOD tests if nodes receive the BEACON packets but are unable to access the physical channel (due to heavy traffic) to reply with ECHO packets. If that is true, extending the BEACON_PERIOD provides the nodes with a longer timeframe to reply and thus a higher probability to transmit the ECHO packet.

Fig. 4.1, 4.2, 4.3 and 4.4 show that increasing the BEACON_PERIOD does not bring about any significant performance improvement, as is the case with increasing the maximum number of BEACON retransmission.

The results show that the explicit method of link failure detection has little effect on the performance of TORA and is not the main cause of incorrect link failure detection. The reason is that in a network with many traffic connections, object block messages are sent frequently thus the implicit method of link failure detection takes precedence and the explicit method becomes redundant. If there is little traffic, nodes are able to both send and receive BEACON and ECHO packets successfully since there is little chance of the packets getting lost due to network congestions. Hence, the explicit method of link failure detection only plays an important role when the DAG is first initiated where TORA floods *query* packets to find routes to the destination. After which, actual traffic (i.e. object block message that are aggregated from IMEP/TORA control packets and data packets) starts and IMEP knows whether or not links are up (using the implicit method) via the ACKs for those object block messages.

4.2 Analysis of Implicit Link Failure Detection

If the implicit method of link failure detection is the main cause of the incorrect detection of link breakages, the main problem is ACK packets failing to reach the sender of the object block message. This causes the sender to think that neighbours which did not reply with an ACK packet are down. This can happen in two ways: inability of neighbouring nodes to send the ACK due to a congested physical channel or inability of the object block message to reach the neighbouring nodes (thus they do not reply with ACK packets).

Many overhead packets are generated due to the broadcast mode and reliable mode of IMEP that is required by TORA. When an object block message is broadcast, all neighbouring nodes are required to reply with an ACK packet. This causes a significant amount of traffic and it increases with the number of traffic connections as more traffic connections means the sending of more object

block messages.

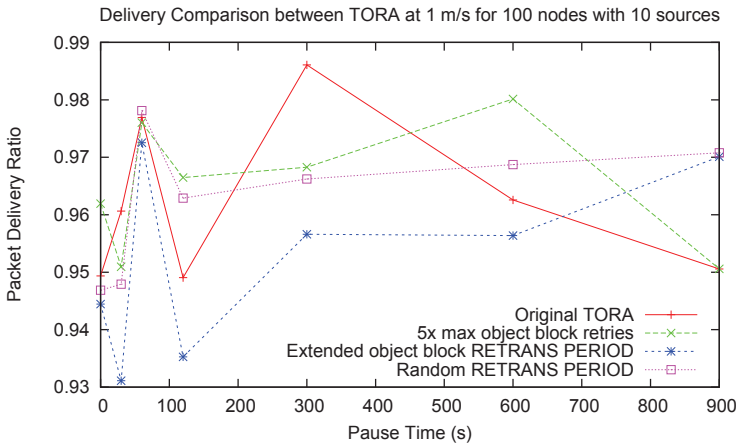

Figure 4.5: 100 nodes with 10 traffic connections at 1 m/s for the implicit method

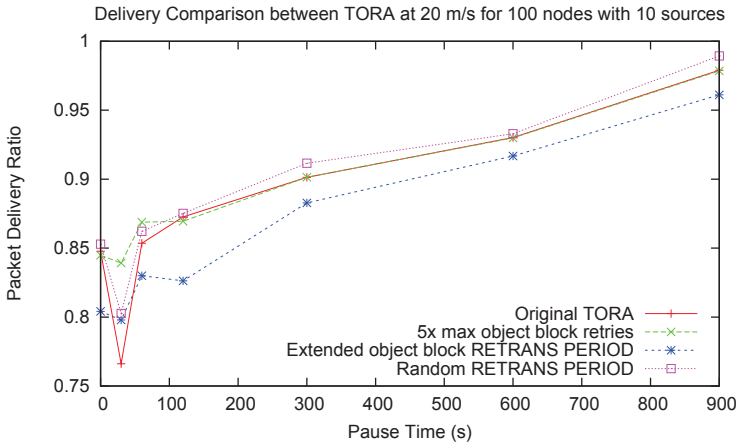

Figure 4.6: 100 nodes with 10 traffic connections at 20 m/s for the implicit method

4.2.1 Increasing the Maximum Number of Object Block Retransmissions

Adapting a modification similar to that in the explicit method of link failure detection, we now increase the maximum number of object block message retransmissions. Similarly, this increases the probability that the object block message sent by a node reaches all of its neighbouring nodes and allows them to reply with ACK packets. Recall that when a node sends out an object block message, it requires all of its neighbours to reply with an ACK packet within MAX_RETRANS_TIME. Any node that failed to do so has its corresponding link labelled as down. This is the broadcast mode required by TORA and it increases the chance that IMEP incorrectly detects a link breakage. We now go on to discuss the results obtained by modifying the implicit method of link failure detection in IMEP.

The results we obtained support our claim that the implicit method of link failure detection is the main cause of the incorrect detection of link breakages. Fig. 4.7 shows that TORA obtains performance improvement of up to 400% in a network of 100 nodes moving constantly at 1 m/s with 30 traffic connections. This is a huge performance increase compared to modifying the explicit method of link failure detection shown in Fig. 4.3. Even for a network of 100 nodes with a mobility of 20 m/s and 30 traffic connections, a performance improvement of 12% is observed at a pause time of 0 as shown in Fig. 4.8. This disparity in performance improvement of 12% to 400% is due to the fact that in high mobility networks, many actual link breakages occur and most of the link breakages reported by IMEP are correct. However, in low mobility networks, link breakages seldom occur but are still frequently reported (incorrectly) by IMEP. Hence, the performance improvement gained from correcting these detection errors is more significant in low mobility networks than high mobility ones.

The underlying reason for this increase in performance is that in the original protocol, the neighbouring nodes do not receive the object block messages and thus do not reply with an ACK packet within MAX_RETRANS_TIME. Once a node fails to reply with an ACK packet after MAX_RETRANS_TIME, the link to that node is labelled as down and TORA starts the route maintenance process thus creating even more overhead. By increasing the maximum number of object block message retransmissions, there are more chances for the object block messages to reach neighbouring nodes thus allowing them to reply with an ACK packet. This reduces the number of link failures that are incorrectly detected.

In the IEEE 802.11 MAC protocol, broadcasting utilizes virtual carrier sensing

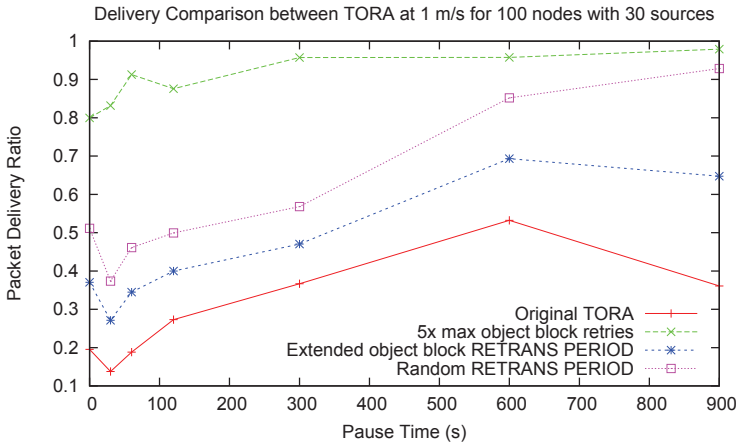

Figure 4.7: 100 nodes with 30 traffic connections at 1 m/s for the implicit method

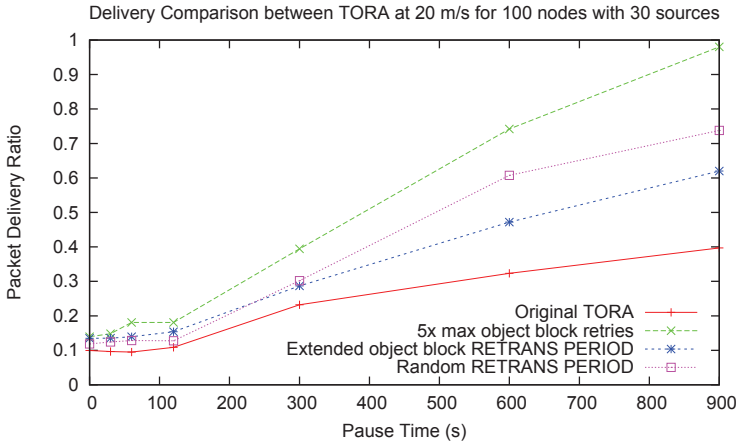

Figure 4.8: 100 nodes with 30 traffic connections at 20 m/s for the implicit method

and physical carrier sensing but not Request-To-Send (RTS) and Clear-To-Send (CTS) packets; nor does it require receiving nodes to acknowledge the reception of packets [8]. Hence, when IMEP sends packets down to the MAC layer for broadcasting, there is no indication of whether the packets reached the intended recipients or collided in the transmission medium. The following sequence of actions based on Fig. 4.9 shows how increasing the maximum number of object

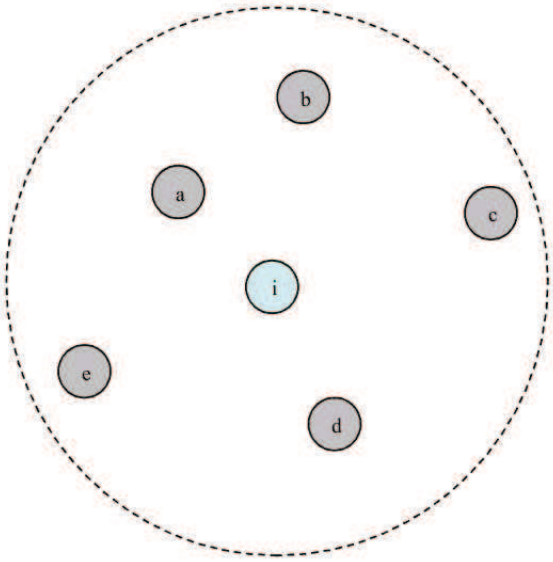

Figure 4.9: Node i and neighbouring nodes a, b, c, d, and e within its transmission range

block message retransmissions for IMEP results in an improvement for TORA.

1. At time t=0, node i broadcast an object block message and sets MAX_RETRANS_TIME to 1.5. Nodes b, c, d, and e do not receive the object block message due to packet collision during transmission.

2. Node a receives the object block message successfully and replies with an ACK packet.

3. At time t=0.5, node i checks its list and discovers that nodes b, c, d, and e have not replied with an ACK packet. It proceeds to re-broadcast the earlier object block message. This time, nodes c, d, and e do not receive the object block message.

4. Node b receives the object block message successfully and replies with an ACK packet.

5. At time t=1.0, node i checks its list and discovers that nodes c, d, and e have not replied with an ACK packet. It proceeds to re-broadcast the earlier object block message. This time, nodes d, and e do not receive the object block message.

6. Node c receives the object block message successfully and replies with an ACK packet.

7. At time t=1.5, node i checks its list and finds out that nodes d and e have not replied with an ACK packet. Since MAX_RETRANS_TIME is equal to the current time, nodes d and e are labelled as down because they failed to reply within MAX_RETRANS_TIME.

If the maximum number of object block message retransmission is increased to 5, MAX_RETRANS_TIME will be 2.5 instead of 1.5. This allows node i to re-broadcast the object block message an additional two times. This provides nodes d and e with more chances to reply with an ACK packet and prevents IMEP from erroneously detecting that nodes d and e are down.

4.2.2 Increasing RETRANS_PERIOD

It is also important to determine if the object block message reached the neighbours but they were unable to send out an ACK packet in time due to a busy medium. If that is the case, increasing the RETRANS_PERIOD solves the problem. Fig. 4.7 shows an improvement of about 100% but the improvement is only a quarter of that gained by increasing the maximum number of object block message retransmissions. From this we can deduce that while extending the RETRANS_PERIOD offers a slight improvement, increasing the maximum number of object block message retransmissions produces a better result and hence is the main factor in improving IMEP.

From the results shown thus far, increasing the maximum number of object block message retransmissions brings about the largest improvement in TORA, followed by increasing the RETRANS_PERIOD. The fact that the former performs better shows that neighbouring nodes are unable to receive object block messages and thus do not reply with ACK packets. The improvements from modifying the implicit method of link failure detection further reinforces our original claim that in a network with many traffic connections, the implicit method takes precedence and the explicit method is hardly utilized, if at all.

4.2.3 An Approach based on Random RETRANS_PERIOD

This approach modifies IMEP such that each node is assigned a random object block message RETRANS_PERIOD instead of a fixed RETRANS_PERIOD for every node. This value is randomly generated from between 0.5 seconds to 0.8

seconds when the node first boots up. By assigning each node with a different RETRANS_PERIOD, we are reducing the probability of multiple nodes broadcasting at the same time and causing packet collisions.

Fig. 4.7 and 4.8 show that this approach increases the overall performance of TORA with a large number of traffic connections while Fig. 4.5 and 4.6 show that its performance remains relatively the same with a small number of traffic connections. Of particular interest is that this approach offers a higher performance increase of approximately 250% compared to 100% by increasing the RETRANS_PERIOD. However, this performance increase is much less than that gained by increasing the maximum number of object block message retransmissions, which is 400%.

CHAPTER 5

Heuristics to Improve TORA

Even with the improvements obtained with a more reliable IMEP, there are problems with the TORA protocol itself and we suggest solutions to fix it in this chapter. The process of route creation involves the propagation of QRY packets that initializes the entire network into a DAG but communication rarely involves nodes at the extreme ends of a DAG. A good example is when a node (the source) needs to communicate with another nearby node (the destination) but the DAG extends to nodes that are far away and do not participate in the communication between the source and destination. In the DAG created by original TORA, route maintenance can be triggered by nodes that do not participate in communication. This makes maintaining the DAG an expensive operation due to the unnecessary routing overhead produced. The approaches we recommend are generally based on the idea of reducing the size of the DAG that has to be maintained. A smaller DAG results in less routing overhead and this allows more data packets to be sent to their destination.

5.1 A Network Localization Approach

This approach works on the concept of initializing and maintaining a DAG that represents a portion of the network termed the localized network. Initializing a portion of the network reduces the routing overhead that could be caused by far away nodes that do not participate in communication. Some key features of this approach are:

- A *propagated_hopcount* (*ph*) variable in QRY packets that keeps track of the number of hopcounts travelled by the packet.
- A *DAG_hopcount* (*dh*) variable that represents how far the localized network stretches and also indicates how far UPD packet should be propagated. A unique *DAG_hopcount* is associated with each DAG.

- An *expiry_hopcount* (*eh*) variable in UPD packets that determines the number of hopcounts a packet will be propagated.

The route creation and route maintenance function of original TORA are modified and are discussed in greater detail. The route erasure function is not modified and is not discussed.

5.1.1 Route Creation

A node i that requires a route to a destination broadcasts a QRY packet with *propagated_hopcount* set to 0. This *propagated_hopcount* is incremented each time a node propagates the QRY packet. When a node with a valid route receives the QRY packet, it records *DAG_hopcount* as the value of *propagated_hopcount* in the QRY packet and propagates an UPD packet with *expiry_hopcount* set to *DAG_hopcount*. When a node receives the UPD packet, it records *DAG_hopcount* as the value of *expiry_hopcount* in the UPD packet, decrements *expiry_hopcount* and propagates the UPD packet if *expiry_hopcount* is greater than 0. This prevents the UPD packet from propagating too far and initializing the entire network.

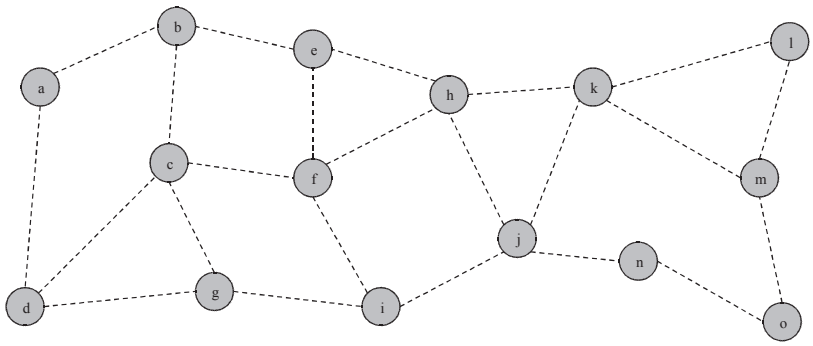

Figure 5.1: The initially uninitiated network

The following sequence of actions illustrates how route creation is performed using our network localization approach.

1. Initially, the network is uninitialized with all links undirected as shown in Fig. 5.1.

2. Node e requires a route to Node d and broadcasts a QRY packet with a *propagated_hopcount* of 0 as shown in Fig. 5.2.

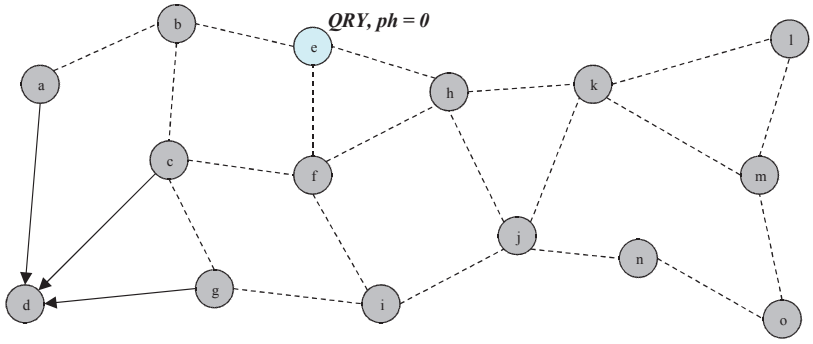

Figure 5.2: Node *e* needs a route to Node *d*, it initiates a QRY packet with a *propagated_hopcount* of 0

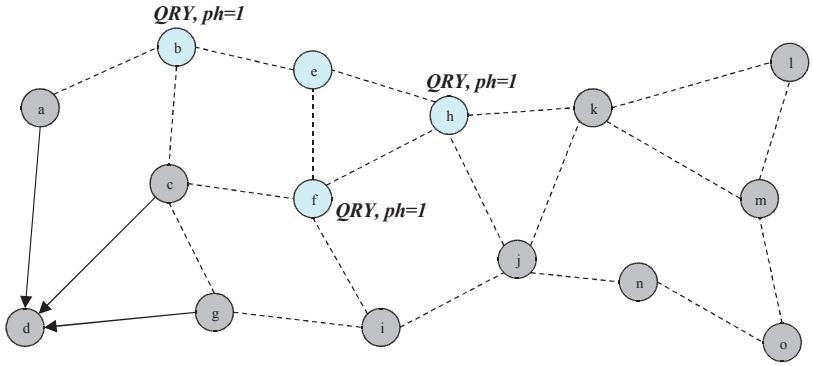

Figure 5.3: Node *b*, *f*, and *h* propagate the QRY packet

3. Node *b*, *f*, and *h* receive the QRY packet, increment *propagated_hopcount* to 1, and broadcast it as shown in Fig. 5.3.

4. Node *i*, *j*, and *k* receive the QRY packet, increment *propagated_hopcount* to 2, and broadcast it. Since node *a*, and *c* have a route to node *d*, they record *DAG_hopcount* as 2 (according to *propagated_hopcount*) and broadcast an UPD packet with an *expiry_hopcount* of 2 as shown in Fig. 5.4.

5. Node *l*, *m*, and *n* receive the QRY packet, increment *propagated_hopcount* to 3, and broadcast it. Node *b*, *f*, and *g* receive the UPD packet, decrement *expiry_hopcount* to 1, broadcast it, and record *DAG_hopcount* as 1 (based on the UPD packet received) as shown in Fig. 5.5.

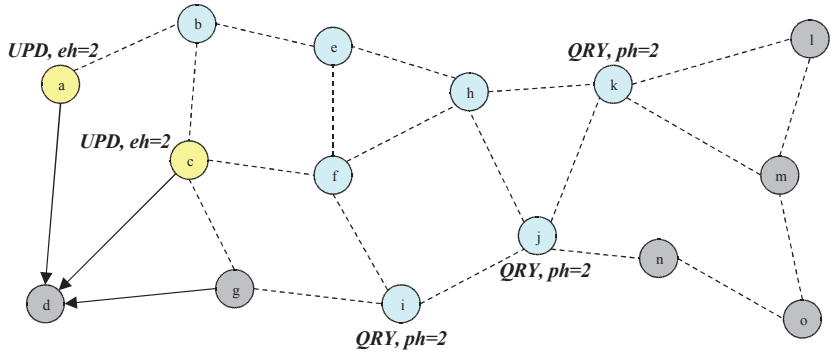

Figure 5.4: Node i, j, and k propagate the QRY packet. Node a, and c generate an UPD packet.

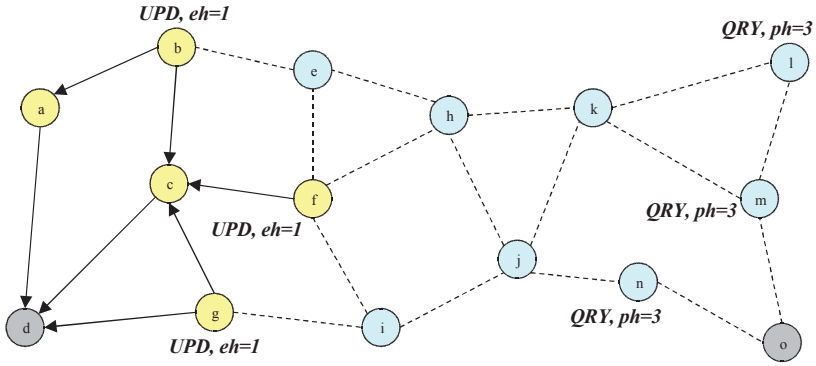

Figure 5.5: Node l, m, and n propagate the QRY packet. Node b, f, and g propagate the UPD packet.

6. Node o receives the QRY packet, increment *propagated_hopcount* to 4, and broadcast it. Node e, h, and i receive the UPD packet, decrement *expiry_hopcount* to 0, and record *DAG_hopcount* as 0 (based on the UPD packet received) as shown in Fig. 5.6. It does not broadcast the UPD packet as *DAG_hopcount* is 0.

In the end, only a portion of the entire network, the localized network is initialized and it consists of nodes a, b, c, d, e, f, g, h, and i but not nodes j, k, l, m, n, and o. This partial initialization of the network reduces the routing overhead (i.e. QRY and UPD packets) and the reduction increases with the size of the network.

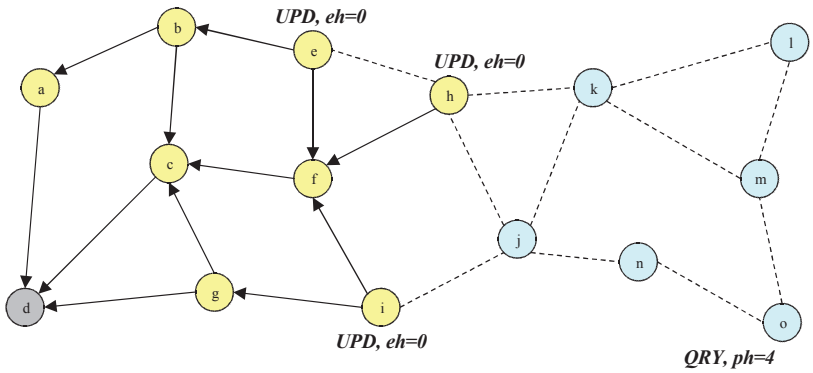

Figure 5.6: Node o propagates the QRY packet. Node e, h, and i propagate the UPD packet.

The building of this localized network facilitates the route maintenance function of TORA as we shall discuss next.

5.1.2 Route Maintenance

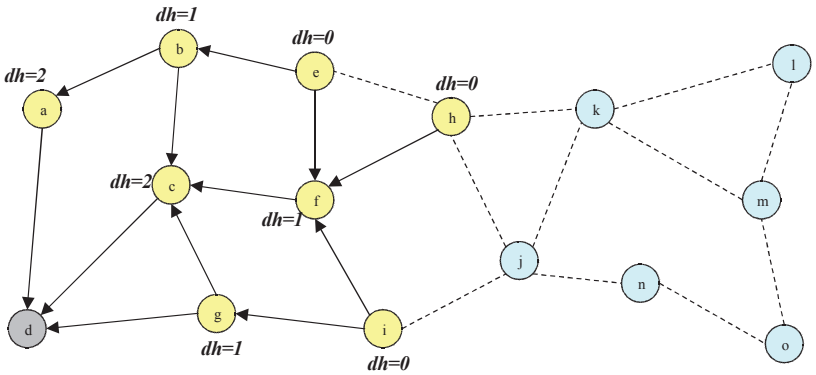

Figure 5.7: Localized network with *DAG_hopcount* labelled

We now illustrate how the network localization approach restricts route maintenance to the portion of the network that requires it most (i.e. nodes that participate or facilitate communication between the source and destination nodes).

1. Fig. 5.7 shows the localized network resulting from route creation using the

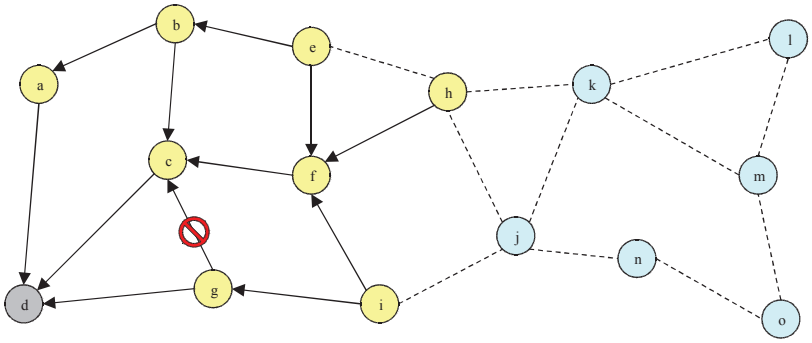

Figure 5.8: The link (c, g) fails

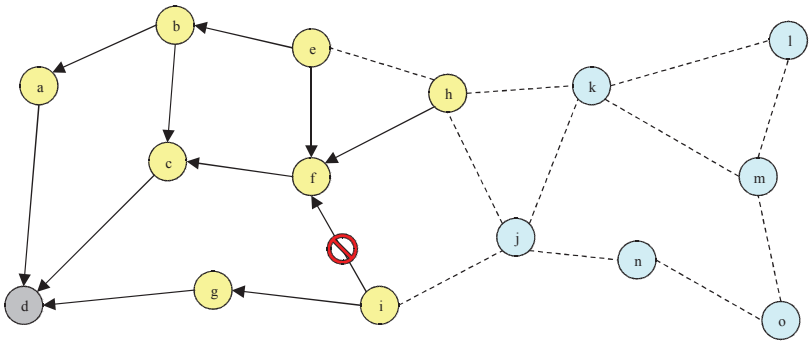

Figure 5.9: The link (f, i) fails

network localization approach. The *DAG_hopcount* represents the lifetime (in terms of hopcounts) of UPD packets used during route maintenance.

2. The link (c, g) fails but no action is needed as both nodes have at least one downstream link as shown in Fig. 5.8.

3. The link (f, i) fails but no action is needed as both nodes have at least one downstream link as shown in Fig. 5.9.

4. The link (d, g) fails. Node g loses its last downstream link and initiates an UPD packet with *expiry_hopcount* initiated to 1 (i.e. the value of *DAG_hopcount*), and broadcast it as shown in Fig. 5.10

5. Fig. 5.11 shows that after receiving the UPD packet, node i reverses its link

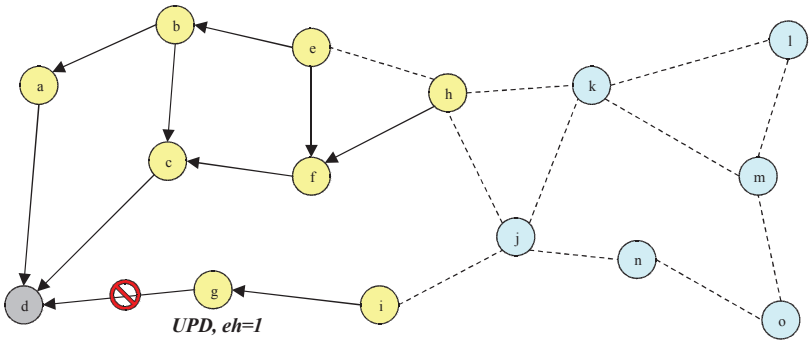

Figure 5.10: The link (d, g) fails and node g initiates an UPD packet

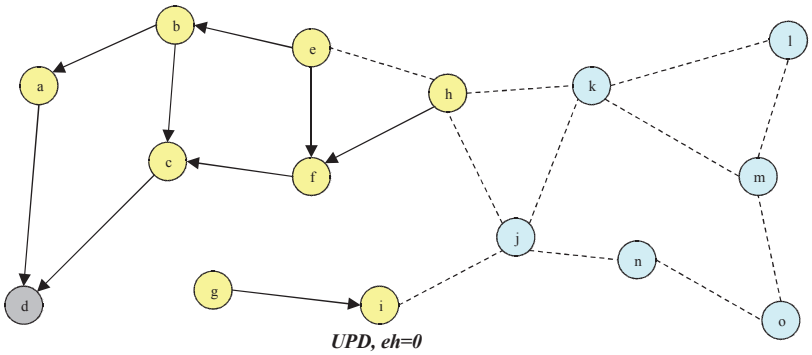

Figure 5.11: The link (i, g) is reversed and node i loses its last downstream link

and loses its last downstream link. As node i has a *DAG_hopcount* of 0, it does not propagate the UPD packet but sets the link (i, g) to undirected as shown in Fig. 5.12.

In this example, we can see that route maintenance is restricted to within the localized network and does not extend to the portion of the uninitiated network. The use of *DAG_hopcount* in nodes and *expiry_hopcount* in UPD packets prevents UPD packets from propagating beyond the localized network and causing the initialization of nodes outside the localized network.

If a link that is outside the localized network fails, the nodes involved do not perform route maintenance. Consider nodes l and m in Fig. 5.12, the failure of link

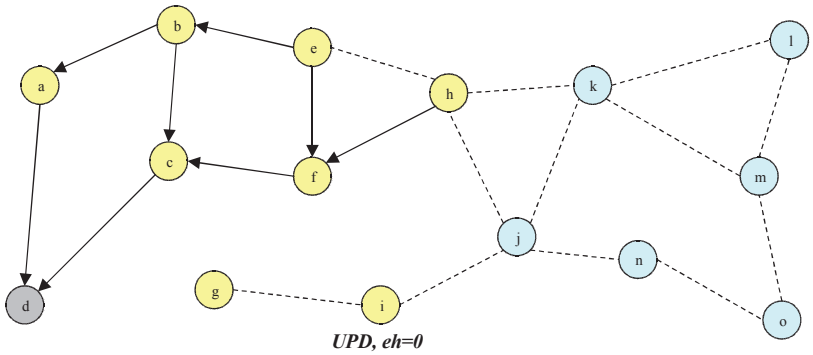

Figure 5.12: The link (i, g) is set as undirected

(l, m) does not initiate route maintenance since both nodes have a NULL height. This reduces the routing overhead and enables more packets to be delivered. In larger networks, this improvement is significant as the localized network excludes more of such nodes than in a smaller network.

5.2 A Selective Node Participation Approach

This approach is tailored for networks with a moderate to high node density where each node i has multiple neighbours since many nodes will be within the transmission range of node i. After route creation, the failure of node i possibly results in route maintenance for all neighbours of node i thus causing a substantial amount of overhead in terms of UPD packets. Our approach aims to reduce the number of nodes that are redundant to the DAG but still maintain the overall integrity of the DAG (i.e. the existence of multiple routes to the destination). Some key features of our approach are:

- A *node_status* variable stored in nodes (with respect to a particular destination) that states whether it participates as part of the network.
- A *probability_active* constant that determines the probability that a node is assigned an active status.

From the uninitialized network shown in Fig. 5.13, we apply the selective node participation approach to obtain the DAG shown in Fig. 5.14. This DAG is rooted at node t and the route creation process was initiated by node a. As

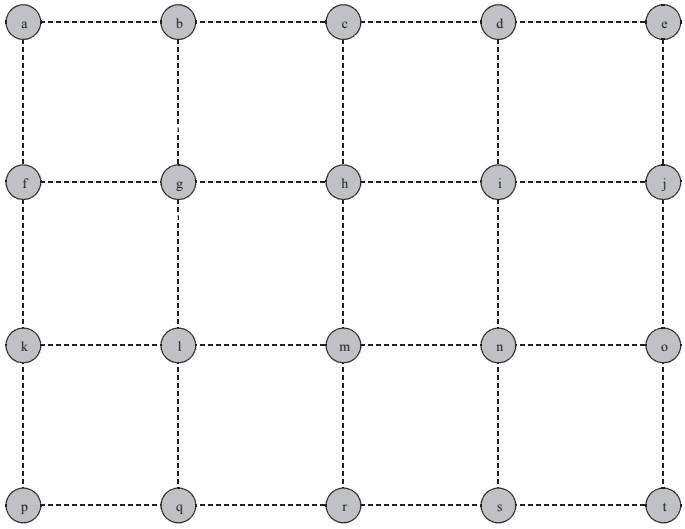

Figure 5.13: Node e requires a route to node t

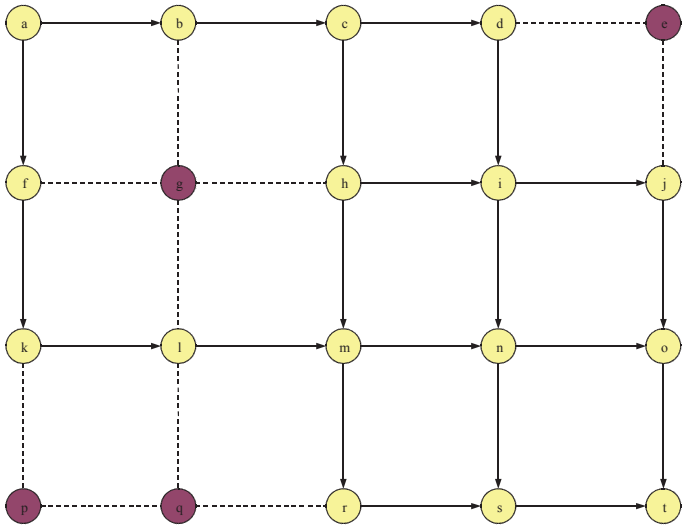

Figure 5.14: Network initialized using the selective node participation approach

Fig. 5.14 shows, there exists multiple routes to node t even though we excluded nodes e, g, p, and q from the DAG.

5.2.1 Algorithm Execution

Our approach differs from original TORA as nodes are assigned either an active, inactive, or unassigned status. All nodes are given an unassigned status when they first boot up. During route creation, a node propagates QRY packets when it requires a route to a destination. When a node i receives a QRY packet, it performs as follows:

1. If *node_status* is active, the QRY packet is processed as per original TORA.
2. If *node_status* is inactive, the QRY packet is ignored.
3. If *node_status* is unassigned and one of its neighbour is the destination or source, *node_status* is set as active and the QRY packet broadcast.
4. If *node_status* is unassigned and none of its neighbour is the destination or source, *node_status* is randomly set as active or inactive according to *probability_active* and the QRY packet is broadcast if the *node_status* assigned is active.

The flooding of QRY packets sets *node_status* in all nodes to either active or inactive. This reduces the QRY packets propagated compared to original TORA since nodes assigned an inactive status no longer propagate QRY packets. This approach is very sensitive to the value of *probability_active* as a high value makes it perform like original TORA while a low value causes unnecessary network partitions. This shows that the approach is very dependent on the node density of the network. We found that a *probability_active* value of 0.8 works well and used that in our simulations.

Route maintenance and route erasure do not involve any nodes with a *node_status* of inactive. This does not affect route maintenance as a node i has many neighbours, and multiple routes exists through those neighbours. Network partitions are not unnecessarily created since those nodes do not participate in the DAG and hence there is no need to erase their heights.

CHAPTER 6

Experiments and Results

In this chapter, we evaluate and compare the performance of original TORA, our approaches to modifying TORA and IMEP, and other leading on-demand protocols such as the DSR protocol [9, 11] and AODV protocol [15, 14]. The approaches of IMEP compared are that by increasing the maximum number of object block message retransmissions and by using a random RETRANS_PERIOD while that for TORA is the network localization approach and selective node participation approach. The performance of these protocols are evaluated using three metrics: packet delivery ratio, routing overhead, and average latency.

6.1 Packet Delivery Ratio

Even though TORA does not perform as well as DSR and AODV, we managed to obtain improvements in certain scenarios that brings it close to DSR and AODV. Some examples are shown in Fig 6.4 and 6.7 where we managed to increase the packet delivery ratio of TORA to 0.8, or just 20% less than that of DSR and AODV.

6.2 Routing Overhead

Fig. 6.11, 6.12, 6.15, and 6.16 show the reduction in routing overhead of original TORA against that of TORA running on top of our two modified versions of IMEP. Comparing the two versions of modified IMEP, the approach by increasing the maximum number of object block message retransmissions generates less routing overhead compared to that of using a random RETRANS_PERIOD.

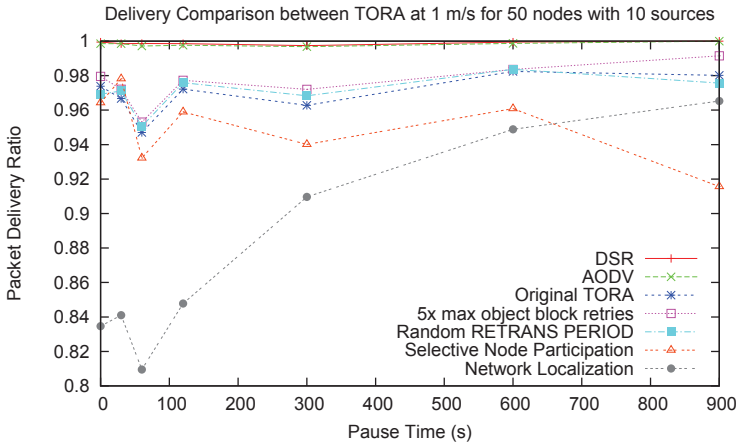

Figure 6.1: Packet delivery ratio for 50 nodes with 10 traffic connections at 1 m/s

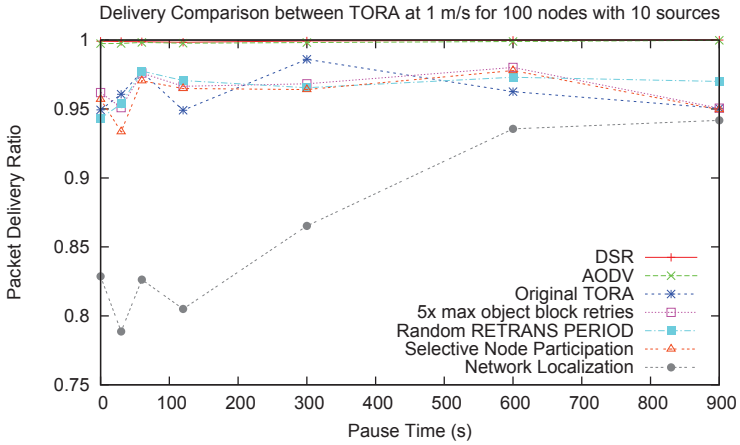

Figure 6.2: Packet delivery ratio for 100 nodes with 10 traffic connections at 1 m/s

6.3 Average Latency

Fig. 6.25 shows that in the case of original TORA, there are a few outliers in the form of packets that take between 64 and 512 seconds to reach their destination. These outliers are the main cause of a high average packet latency of 31 seconds as shown in Fig. 6.24 for a network of 100 nodes, mobility of 20 m/s, and pause

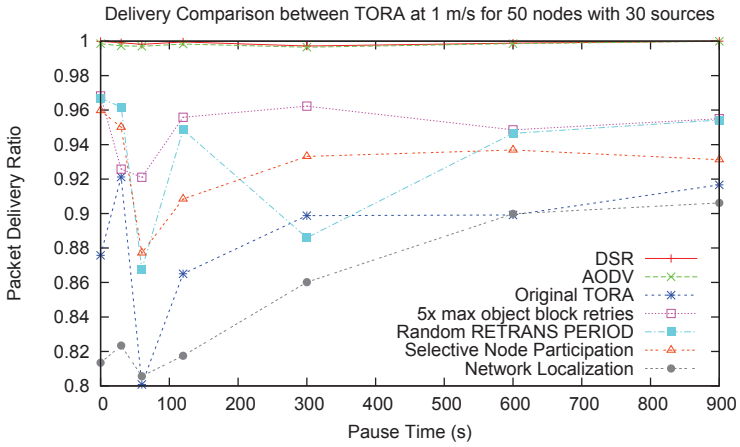

Figure 6.3: Packet delivery ratio for 50 nodes with 30 traffic connections at 1 m/s

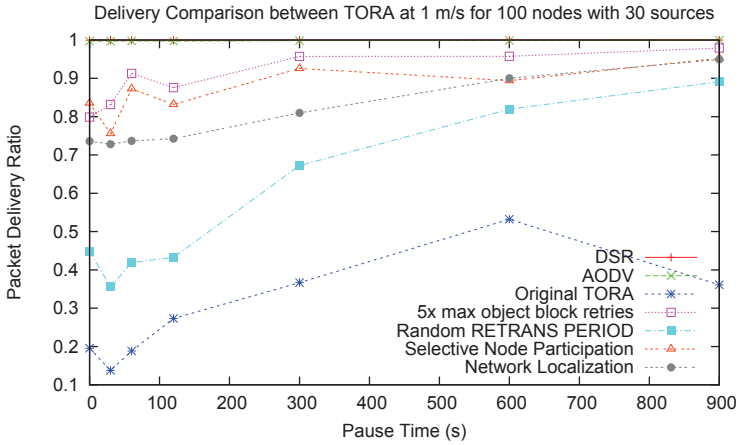

Figure 6.4: Packet delivery ratio for 100 nodes with 30 traffic connections at 1 m/s

time of 30 seconds. These packets belong to intermediate nodes that cannot find a route to the destination due to the incorrect detection of link failures by IMEP. Due to the unavailability of routes, these nodes store the packets in a queue until a route is found. By the time a route is available, the packets may have already been in the queue for an extended amount of time thus resulting in a long latency.

Our modified versions of IMEP also results in reductions in latency of up to

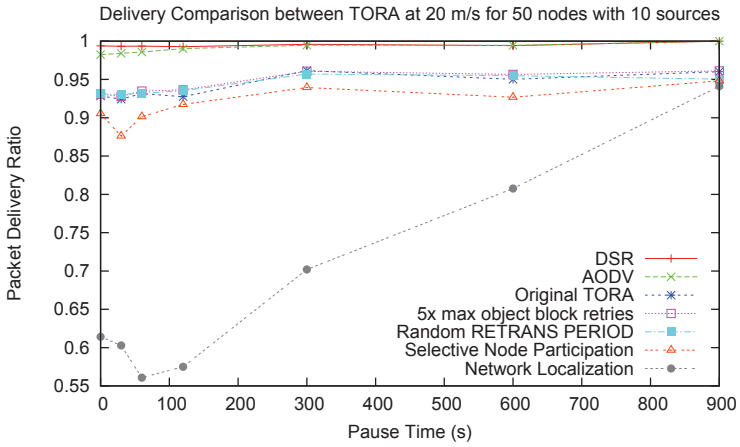

Figure 6.5: Packet delivery ratio for 50 nodes with 10 traffic connections at 20 m/s

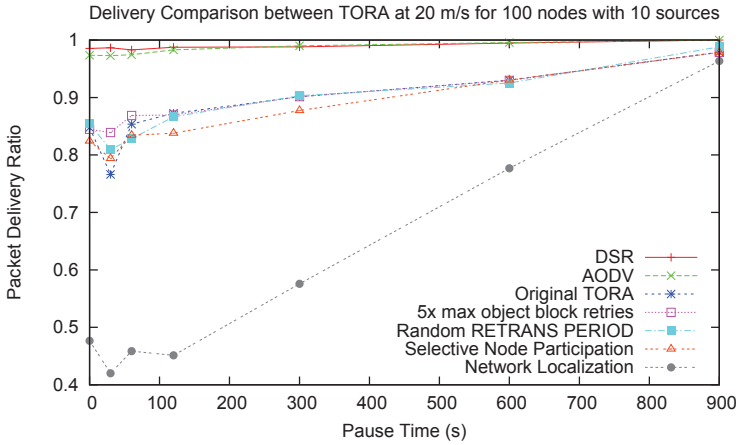

Figure 6.6: Packet delivery ratio for 100 nodes with 10 traffic connections at 20 m/s

34% as shown in Fig. 6.24. Even with a pause time of 30 seconds, our two approaches of increasing the maximum number of object block message retransmissions and by using a random RETRANS_PERIOD reduced the average packet latency by 31% and 24% respectively. This phenomenon is best explained by Fig. 6.25 which shows that while our approaches do not reduce the number of outliers (i.e. packets with a latency of between 64 and 512 seconds), it increases

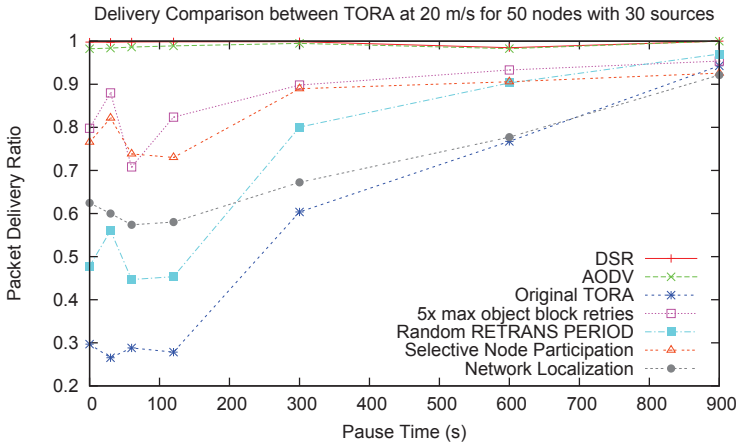

Figure 6.7: Packet delivery ratio for 50 nodes with 30 traffic connections at 20 m/s

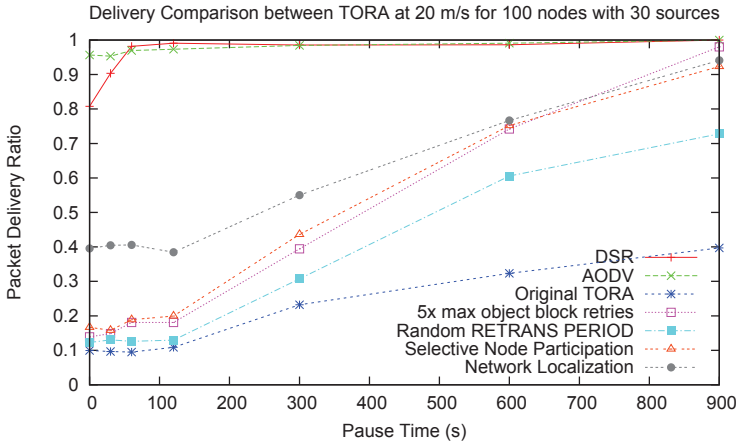

Figure 6.8: Packet delivery ratio for 100 nodes with 30 traffic connections at 20 m/s

the number of packets delivered within 1 second. This is partly because of the improvement in packet delivery ratio which means that more packets can be delivered and these packets that can now be delivered are done so within 1 second.

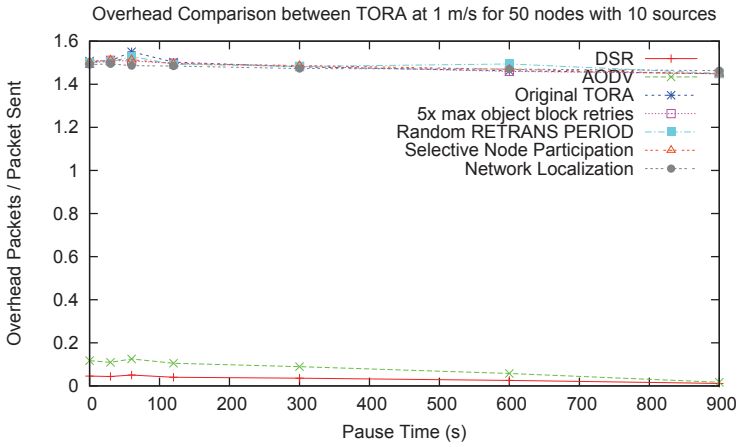

Figure 6.9: Routing overhead for 50 nodes with 10 traffic connections at 1 m/s

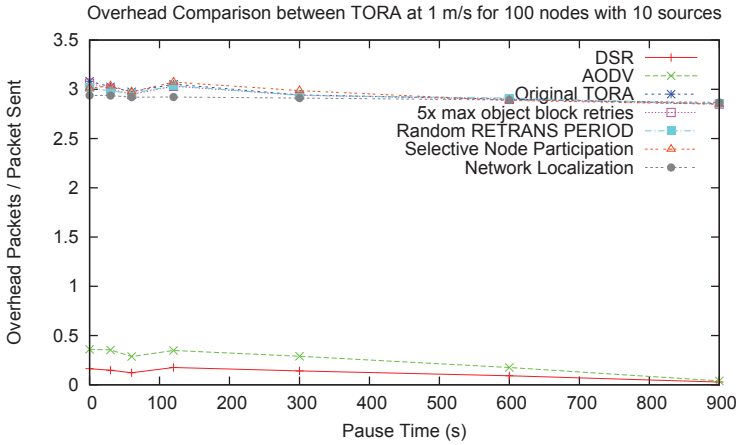

Figure 6.10: Routing overhead for 100 nodes with 10 traffic connections at 1 m/s

6.4 Discussion

The routing overhead graphs compare well with the corresponding packet delivery ratio graphs such that when there is an increase in packet delivery, there is a corresponding decrease in routing overhead. These two sets of results support our initial claim that a large amount of routing overhead prevents data packets

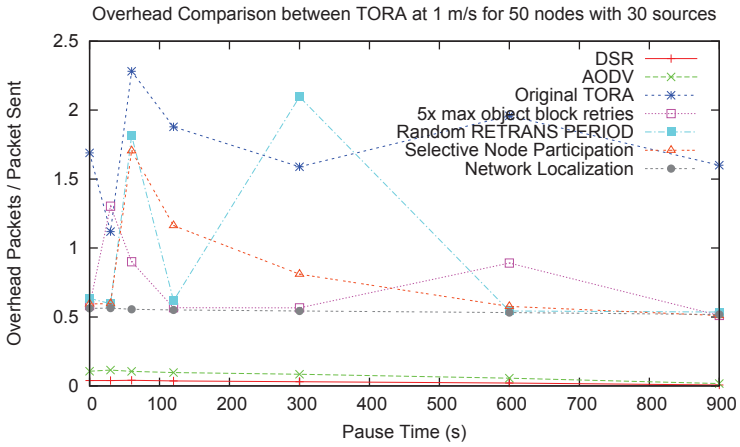

Figure 6.11: Routing overhead for 50 nodes with 30 traffic connections at 1 m/s

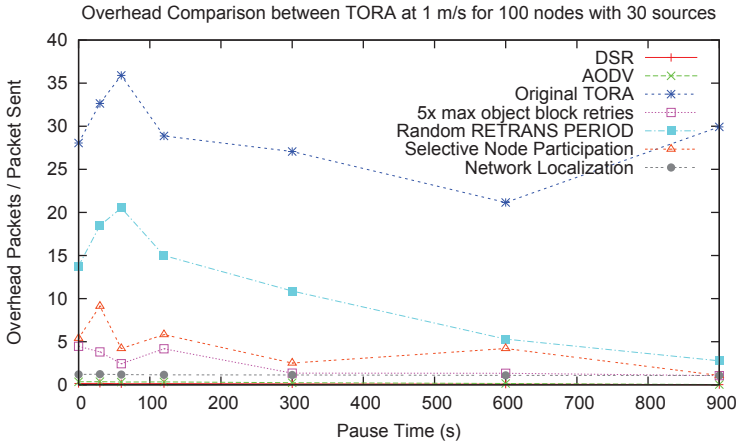

Figure 6.12: Routing overhead for 100 nodes with 30 traffic connections at 1 m/s

from reaching their destinations.

The two approaches of modifying IMEP correct the link failure detection mechanism and allow more packets to be delivered while reducing the amount of routing overhead due to unnecessary route maintenance. The approach of increasing the maximum number of object block message retransmissions performs better than that by using a random RETRANS_PERIOD in all three metrics

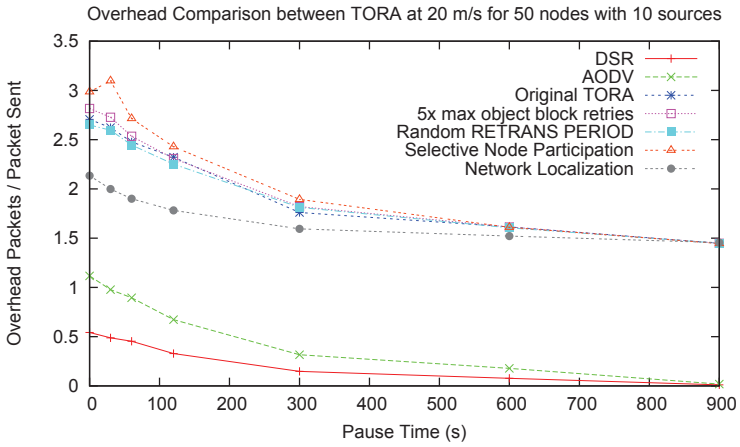

Figure 6.13: Routing overhead for 50 nodes with 10 traffic connections at 20 m/s

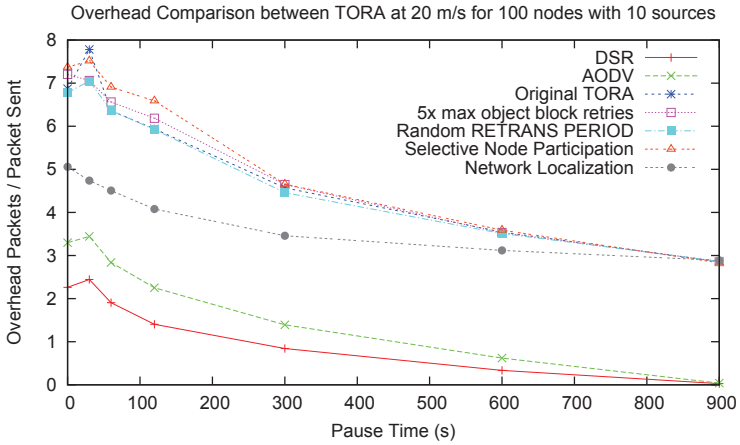

Figure 6.14: Routing overhead for 100 nodes with 10 traffic connections at 20 m/s

measured. These results highlight the importance of having a reliable IMEP layer that can correctly detect link failures. This prevents unnecessary route maintenance and allows packets to be delivered to their destination successfully due to a less congested medium.

As Fig. 6.8 shows, there is little improvement from modifying IMEP for high mobility networks with many traffic connections thus there is a need to modify

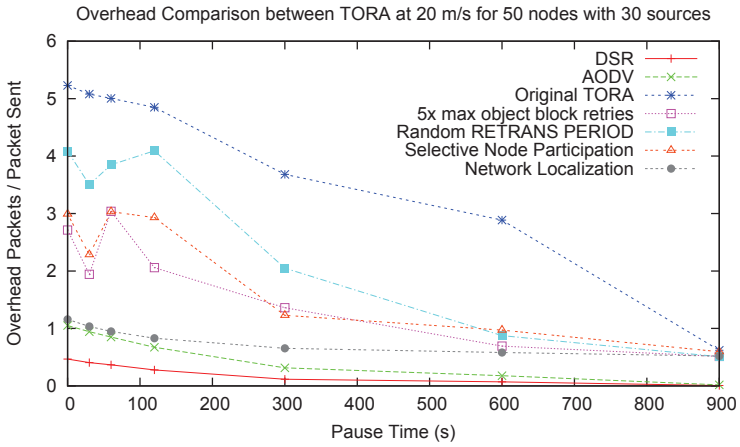

Figure 6.15: Routing overhead for 50 nodes with 30 traffic connections at 20 m/s

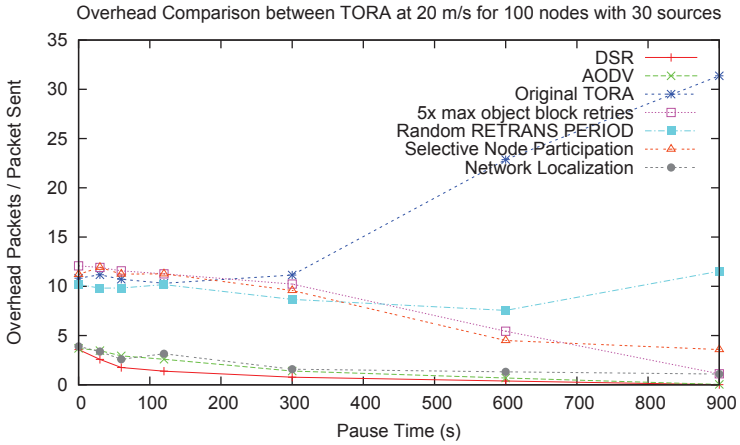

Figure 6.16: Routing overhead for 100 nodes with 30 traffic connections at 20 m/s

the TORA protocol. For that purpose, we developed the network localization approach and selective node participation approach. The network localization approach performs well for networks with many traffic connections but not when there are few traffic connections. It produces little routing overhead at less than 5 routing packets per data packet sent regardless of network size, number of traffic connections or mobility. These results show that this approach is best

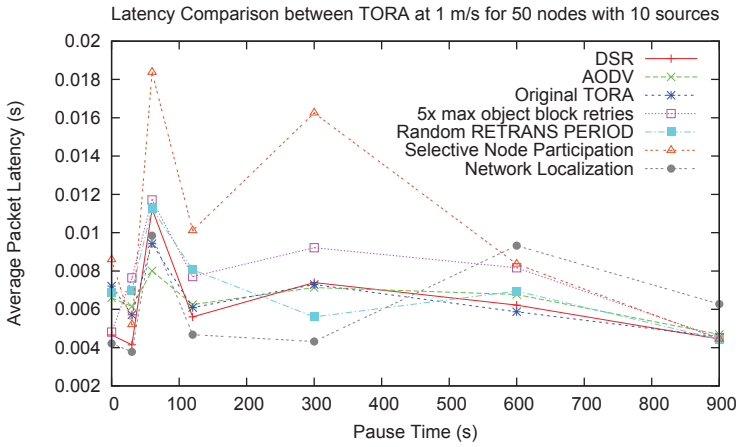

Figure 6.17: Average latency for 50 nodes with 10 traffic connections at 1 m/s

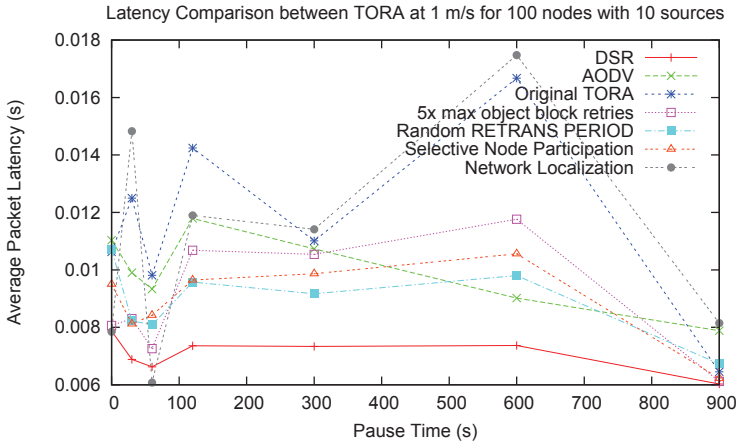

Figure 6.18: Average latency for 100 nodes with 10 traffic connections at 1 m/s

suited for use in large networks with many traffic connections. The selective node participation approach offers a smaller improvement for networks with many traffic connections compared to the network localization approach but it does not suffer from any decrease in performance for networks with few traffic connections.

The poor performance of the network localization approach for networks with few traffic connections is mainly because in a localized network, a network par-

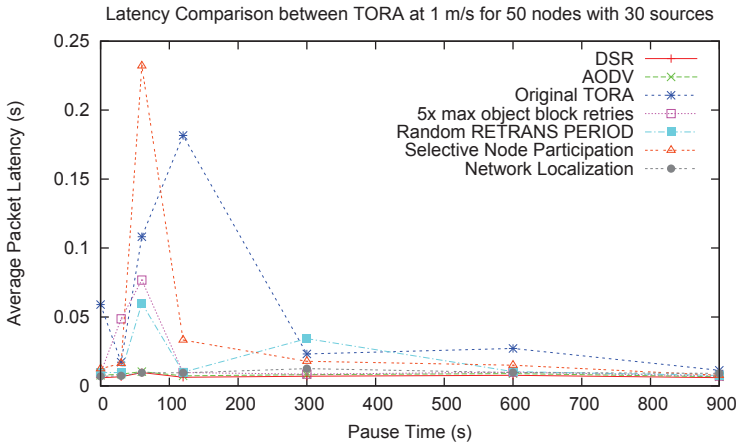

Figure 6.19: Average latency for 50 nodes with 30 traffic connections at 1 m/s

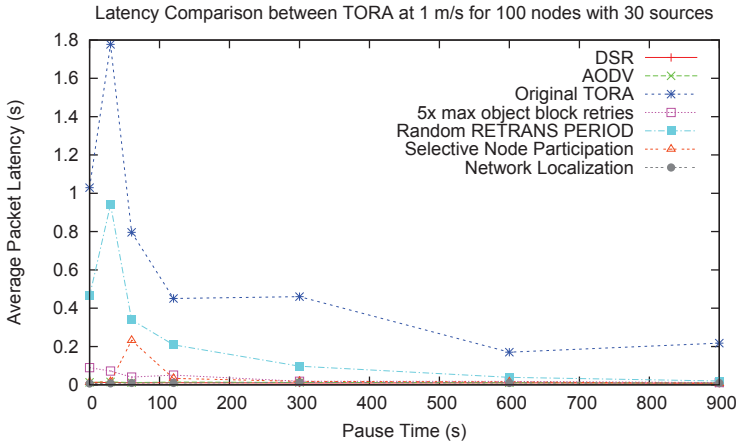

Figure 6.20: Average latency for 100 nodes with 30 traffic connections at 1 m/s

tition can be easily formed once the destination moves away. This causes the subsequent data packets to be dropped until the destination moves back into the localized network.

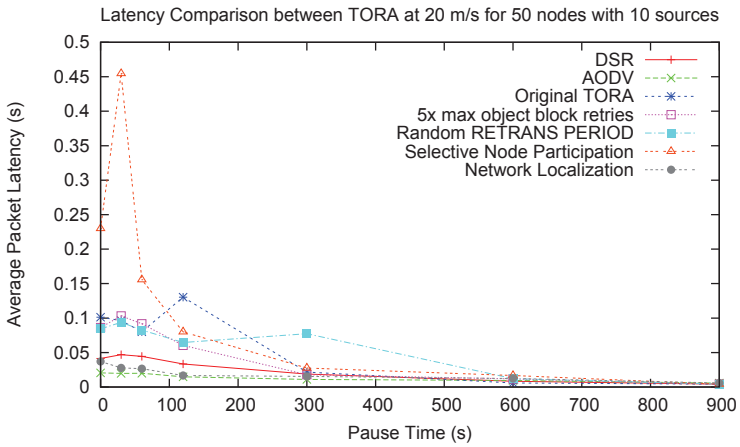

Figure 6.21: Average latency for 50 nodes with 10 traffic connections at 20 m/s

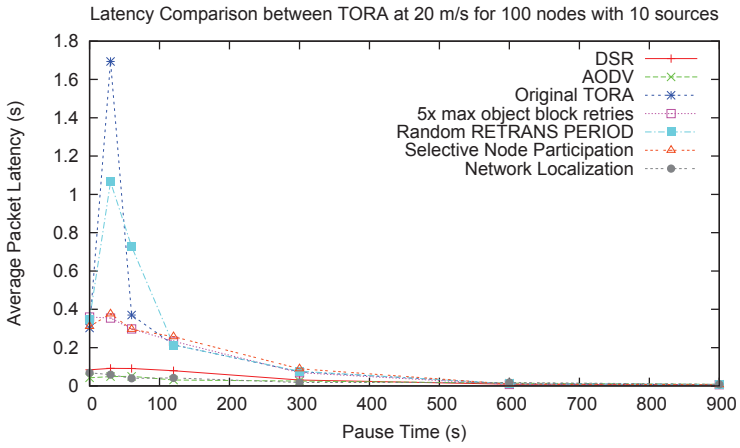

Figure 6.22: Average latency for 100 nodes with 10 traffic connections at 20 m/s

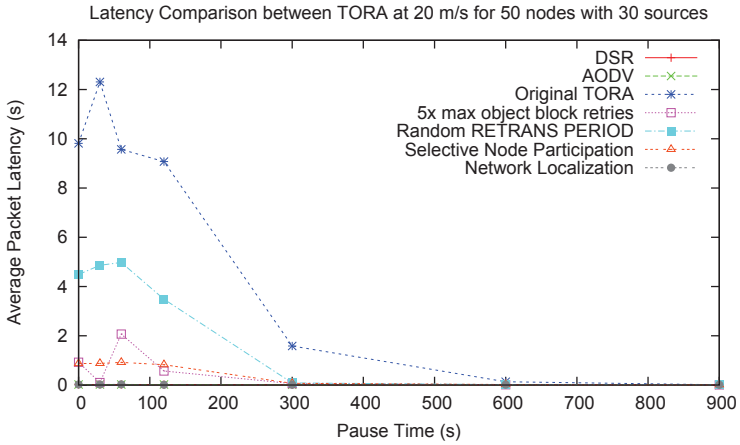

Figure 6.23: Average latency for 50 nodes with 30 traffic connections at 20 m/s

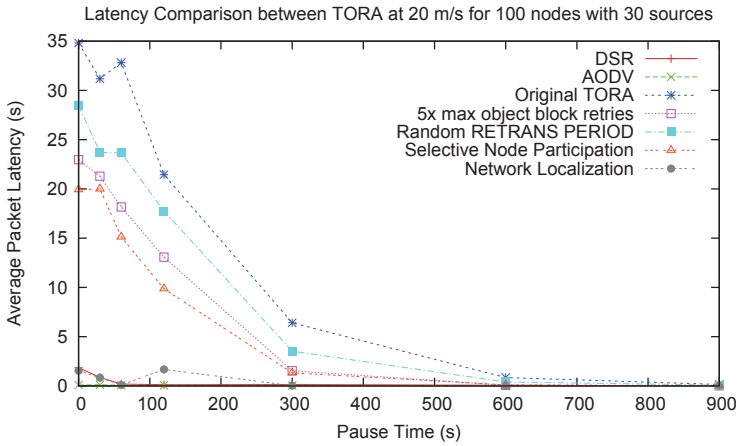

Figure 6.24: Average latency for 100 nodes with 30 traffic connections at 20 m/s

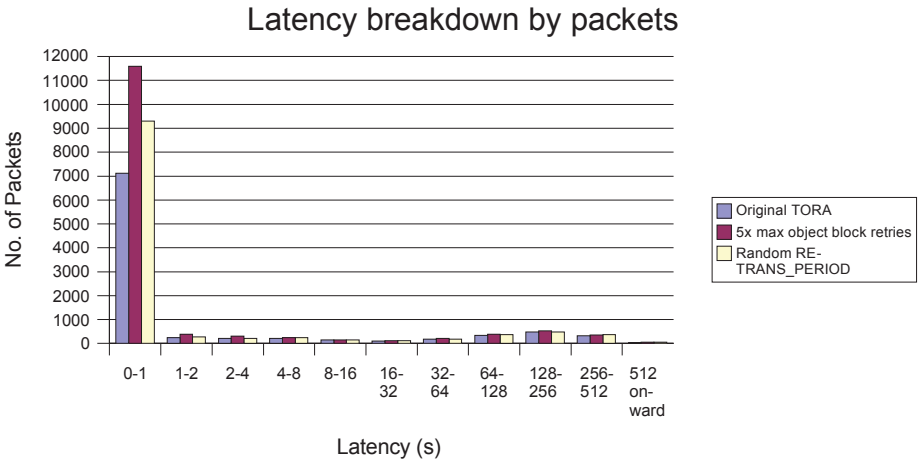

Figure 6.25: Breakdown of packet latency for 100 nodes with 30 traffic connections at 20 m/s and pause time of 30 s

CHAPTER 7

Conclusion

The way IMEP incorrectly detects link failures is part of the reason for the poor performance of TORA in networks with many traffic connections. We narrowed down the implicit method of link failure detection in IMEP as the main factor and recommended two approaches to correct this error. One approach is by increasing the maximum number of object block message retransmissions and the other by using a random RETRANS_PERIOD. The improvement obtained from correcting IMEP is substantial but it begins to diminish for large, high mobility networks with many traffic connections. This is due to the frequent link breakages that causes route maintenance and high routing overhead that prevents data packets from reaching the destination. Modifications to the TORA protocol are necessary to solve this problem of high routing overhead.

The network localization approach solves the problem of high routing overhead in large, high mobility networks with many traffic connections by initializing only a part of the network as the DAG. This approach reduces routing overhead tremendously compared to original TORA but works well only for networks with many traffic connections. Another approach we developed is the selective node participation approach that is catered towards networks with a moderate to high node density. These two approaches extend upon the modifications we made to IMEP and involve modifying the TORA protocol itself.

On the whole, we have seen that modifications to both the IMEP and TORA protocol are necessary to increase the overall performance of TORA. Modifications to IMEP increases TORA's performance in low mobility networks and modifications to TORA itself are necessary for high mobility networks.

A possible area for future work include the development of a new protocol to replace IMEP. This new protocol will make use of the RTS/CTS mechanism of IEEE 802.11 MAC protocol to provide TORA with accurate information on the status of its links. This requires the new protocol to use single-casting when sending data packets instead of broadcasting in the implicit method of link failure detection in IMEP. In short, the protocol will be a streamlined version of IMEP

that provides the services that TORA requires.

Another possible future work would be the enhancement of the network localization approach such that it offers a performance improvement even in networks with few traffic connections. The problem with the approach is that when the destination moves away from the localized network, a network partition is created thus data packets dropped. A possible solution is for the destination to broadcast *update* packet at regular intervals so as to re-initialize the localized network in the event that it moves away.

APPENDIX A

Original Honours Proposal

Title: Performance Enhancement of the TORA Protocol

Author: Kwan Hui Lim

Supervisor: Associate Professor Amitava Datta

Background

A Mobile Ad hoc Network (MANET) consists of mobile nodes which are capable of wireless communications and moving in a random fashion [4]. The connectivity between the nodes forms the topology of the MANET and one of its characteristic is its dynamic topology due to the unpredictable and sudden movement of nodes. Another characteristic is the lower capacity of wireless links compared to their hardwired counterparts thus increasing the likelihood of traffic congestions.

The Temporally-Ordered Routing Algorithm (TORA) protocol [13, 12] is designed to work in such a network. It is based on a link reversal algorithm and provides multiple loop-free routes to the destination on-demand. TORA operates based on three main mechanisms; route creation, route maintenance and route erasure. The route creation mechanism is first initiated and query packets are flooded to search for routes while update packets are propagated back if there is a valid route. After which, route maintenance is carried out whereby link reversal creates alternate routes for nodes that have lost their last route to a destination. Lastly, route erasure ensures that all invalid routes in a network partition are erased to prevent the formation of loops.

Broch *et al.* [1] have shown that TORA's packet delivery ratio drops drastically in high mobility networks with a large number of sources. This has been attributed to the failure of TORA to converge due to the increased congestion that comes with a higher number of sources.

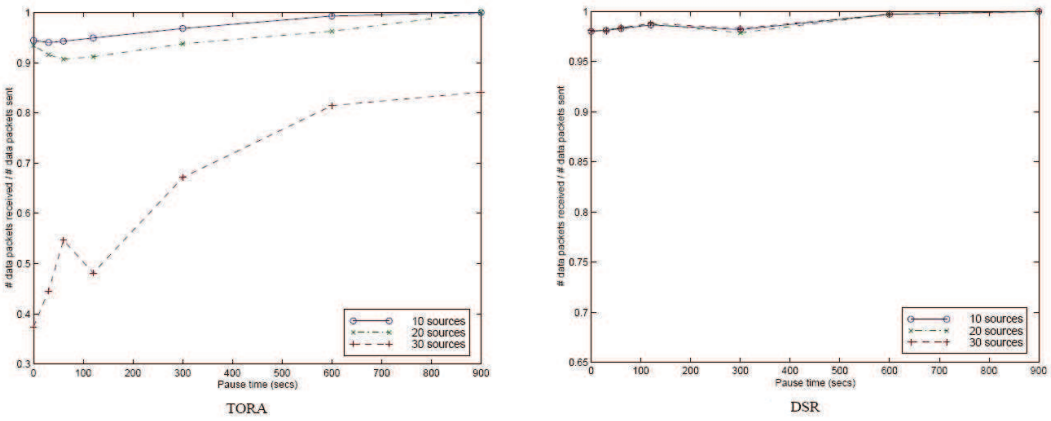

Figure A.1: Packet delivery ratio as a function of pause time [1]

On the contrary, the Dynamic Source Routing (DSR) protocol [9, 11] offers a consistent performance, regardless of both the mobility of the network and the number of sources. The routing overhead of DSR also increases consistently with a corresponding increase in the number of sources.

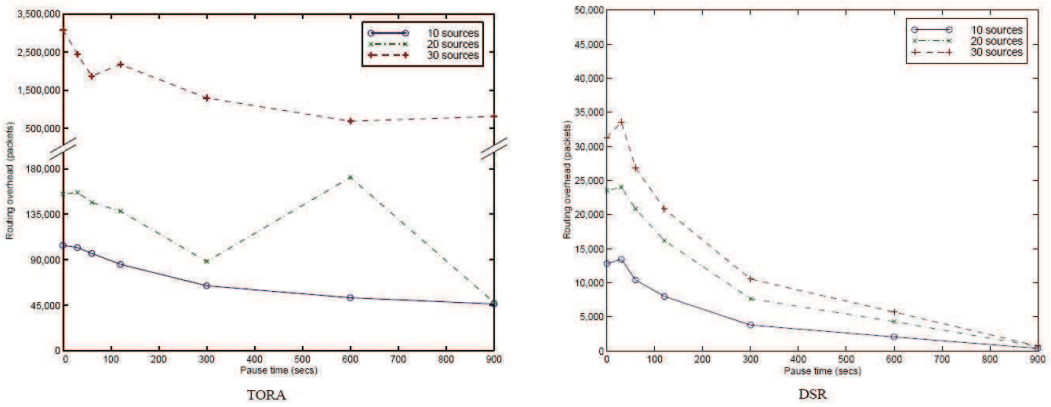

Figure A.2: Routing overhead as a function of pause time [1]

TORA experiences a great increase in the amount of routing overhead with an increase in the number of sources from 20 to 30. With 30 source nodes, TORA suffers from traffic congestion which causes the collision and subsequent loss of HELLO and ACK packets used for neighbour discovery. The loss of these packets lead to the erroneous belief that links were broken thus initiating the route maintenance phase of TORA, causing even more UPD packets to be sent.

Aim

In this project, I will modify the existing TORA protocol to enhance its performance to a level comparable to that of other MANET protocols. These performance enhancements of TORA would be further divided into:

- Increasing packet delivery ratio in high mobility networks.
- Reduction of the amount of routing overhead generated.

The end results will be a modified TORA protocol that is comparable to the DSR protocol in terms of packet delivery ratio and generates a less amount of overhead as compared to the current implementation of TORA. The modified TORA protocol will also be highly scalable and offer a consistent performance even in large networks with high node mobility.

Method

The ns-2 network simulator [17] will be the main tool used in this project for the simulation of the wireless networks and the various scenarios possible. The random waypoint model [10] will be used in the generation of movement scenarios for the nodes.

Firstly, experiments will be carried out to determine the performance of the TORA protocol in different scenarios. These experiments will show us how TORA performs in comparison to other MANET protocols, mainly the DSR protocol. In addition, we will also be able to identify the scenarios where TORA performs badly such as in a network with high mobility nodes, a large number of sources, etc.

Following which, we will proceed to make modifications to the original TORA protocol. Using this modified TORA protocol, we will run simulations to evaluate its performance in comparison with the original TORA protocol as well as other routing protocols.

Software and Hardware Requirements

This project will require the use of the *ns-2* network simulator (version 2.29 or later) for the purpose of evaluating the performance of the various MANET routing protocols in comparison with our implemented solution. In addition, the

use of the C++ programming language will be required for making modifications to the current implementation of the TORA protocol.

In terms of hardware requirements, an Intel P4 3.0+ GHz computer with at least 1GB of RAM and 20GB of free harddisk space would suffice. Currently, the CSSE laboratories are fitted with terminals of this configuration.

Task

Tasks	Completed By
Literature Review and Research Proposal	August 2006
Research Proposal Talk	August 2006
Understand TORA implementation and codes	September 2006
Revision of Research Proposal	October 2006
Formulation and Testing of Solutions	November 2006
Performance Evaluation of Solutions	January 2007
Drafting of Dissertation	April 2007
Submission of Dissertation	May 2007
Seminar Presentation	May 2007
Poster Submission	May 2007

Bibliography

[1] Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y.-C., and Jetcheva, J. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking* (New York, NY, USA, 1998), ACM Press, pp. 85–97.

[2] Broustis, L., Jakllari, G., Repantis, T., and Molle, M. A Comprehensive Comparison of Routing Protocols for Large-Scale Wireless MANETs. In *SECON '06: 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks* (2006), IEEE Computer Society, pp. 951–956.

[3] Corson, M. S., Papademetriou, S., P. Papadopoulos, V. P., and Qayyum, A. An Internet MANET Encapsulation Protocol (IMEP) Specification. Internet Draft, February 1999. Available from: http://www.tools.ietf.org/html/draft-ietf-manet-imep-spec-01.

[4] Corson, S., and Macker, J. RFC 2501: Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. Request for Comments, January 1999. Available from: http://www.ietf.org/rfc/rfc2501.txt.

[5] Das, S. R., Castaneda, R., and Yan, J. Simulation-based Performance Evaluation of Routing Protocols for Mobile Ad hoc Networks. *Mobile Networks and Applications 5*, 3 (2000), 179–189.

[6] Day, J. D., and Zimmermann, H. The OSI Reference Model. In *Proceedings of the IEEE* (1983), vol. 71, pp. 1334– 1340.

[7] Dharmaraju, D. Routing and Quality of Service in Mobile AdHoc Networks with TORA/INORA. Master's thesis, University of Maryland, 2002.

[8] IEEE Computer Society LAN MAN Standards Committee. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Standard, 1999.

[9] Johnson, D. B. Routing in Ad Hoc Networks of Mobile Hosts. In *Workshop on Mobile Computing Systems and Applications* (Santa Cruz, CA, USA, 1994), pp. 158–163.

[10] Johnson, D. B., and Maltz, D. A. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, Imielinski and Korth, Eds., vol. 353. Kluwer Academic Publishers, 1996.

[11] Johnson, D. B., Maltz, D. A., and Hu, Y.-C. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). Request for Comments: 4728, July 2004. Available from: http://www.tools.ietf.org/html/rfc4728.

[12] Park, V. D., and Corson, M. S. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *INFOCOM '97: Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution* (Washington, DC, USA, 1997), IEEE Computer Society, pp. 1405–1413.

[13] Park, V. D., and Corson, M. S. Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification. Internet Draft, July 2001. Available from: http://tools.ietf.org/id/draft-ietf-manet-tora-spec-04.txt.

[14] Perkins, C. E., Belding-Royer, E. M., and Das, S. R. Ad hoc On-Demand Distance Vector (AODV) Routing. Request for Comments: 3561, February 2003. Available from: http://www.tools.ietf.org/html/rfc3561.

[15] Perkins, C. E., and Royer, E. M. Ad-hoc On-Demand Distance Vector Routing. In *WMCSA '99: Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications* (New Orleans, LA, USA, 1999), pp. 90–100.

[16] Tuch, B. Development of WaveLAN, an ISM band wireless LAN. *AT&T Technical Journal 72*, 4 (1993), 27–37.

[17] UCB, LBNL, and VINT. The ns-2 Network Simulator. Internet, April 2007. Available from: http://nsnam.isi.edu/nsnam/index.php/Main_Page.

[18] Weiss, E., Hiertz, G. R., and Xu., B. Performance Analysis of Temporally Ordered Routing Algorithm based on IEEE 802.11a. In *VTC '05: Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st* (2005), vol. 4, pp. 2565– 2569.