MASTER'S THESIS

Routing and Quality of Service in Mobile AdHoc Networks with TORA/INORA

by Dinesh Dharmaraju
Advisor: John S. Baras

CSHCN MS 2002-1
(ISR MS 2002-1)

The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CSHCN/

Abstract

Title of Thesis: ROUTING AND QUALITY OF SERVICE IN MOBILE ADHOC NETWORKS WITH TORA/INORA

Degree Candidate: Dinesh Dharmaraju

Degree and year: Master of Science, 2002

Thesis directed by: John.S. Baras,

Department of Electrical and Computer Engineering

Mobile Adhoc NETworks(MANETs) are characterized by bandwidth constrained links, multiple hops and dynamic topologies. Routing and providing quality of service in these networks is a highly challenging task. In this thesis, we discuss the unicast routing in MANETs with enhancements to the Temporally Ordered Routing Algorithm (TORA) and quality of service at the network layer with INORA.

Temporally Ordered Routing Algorithm (TORA) is a highly distributed, scalable routing protocol for MANETs. We discuss improvements in the performance of TORA by Query Localization. We also discuss the improvements to TORA to remove a specific traffic instability problem in TORA. We also describe the proac-

tive operation of TORA and show by simulations that it is generally a good idea to have the gateway nodes in a MANET proactively perform route building and route maintenance.

We propose INORA, a network layer QoS support mechanism in adhoc networks, which makes use of the INSIGNIA in-band signaling mechanism and TORA. We present an effective coupling between TORA and INSIGNIA to get routes that are “best-able” to provide QoS requirements for a flow. INORA also provides congestion control. We present two schemes calles “Coarse feedback scheme” and “Fine feedback scheme” under the INORA framework. We show that under heavily loaded conditions, the INORA schemes perform better than when the signaling protocol and the routing protocol operate without feedback.

ROUTING AND QUALITY OF SERVICE IN MOBILE ADHOC NETWORKS WITH TORA/INORA

by

Dinesh Dharmaraju

Thesis submitted to the Faculty of Graduate School of the University of Maryland, College Park in partial fulfillment of the requirements for the degree of Master of Science, 2002.

Advisory Committee:

Dr. John. S. Baras, Chair

Dr. Richard. J. La

Dr. Sennur Ulukus

Acknowledgments

I am very thankful to my advisor, Dr. John Baras for directing my research, for supporting me and motivating me to excell. I would also like to thank Dr. Richard La for serving on my thesis committee and for the discussions on INORA. I am very thankful to Dr. Scott Corson for directing my research initially.

This research was suppported through collaborative participation with NASA NAG-59150 and NASA NCC-3528

I would like to thank Matthew Impett for initiating me to TORA and ns-2. I would like to thank Ulas Kozat for the valuable discussions and ideas.

I am thankful to Pedram Hovareshti and Ayan Roy Choudhary for their discussions during the development of INORA, for their help in validating the INORA simulations and helping me in getting the results. I would like to thank Vijay Bharadwaj and Jau-Ling Chou for their great help in the lab. I would also like to thank Harsh Mehta, Karthik Chandrasekar, Kyriakos Manousakis, Senni Perumal and Maben Rabi for the valuable discussions. I would also like to thank Sudhir Varma for his help during the preparation of the document. I would like to thank Majid Raissi-Dehkordi, Gun Akkor and Jia-Shiang Jou for giving me the traffic

models in ns-2 simulations. I am thankful to my roommates for their understanding support during my stay.

Finally, I would like to thank my Master, my parents, my sister, family and friends for always supporting, guiding and encouraging me and helping me become what I am now.

Dedication

To Master, Amma and Nanna

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The abundance and variety of information services provided by the Internet along with the possibility of access to services via light, hand-held, cord-less devices such as portable computers, mobile phones and personal digital assistants(PDAs), have transformed wireless communication systems into a prominent part of any state of art network. The studies and developments in wireless networking have primarily been driven by success of the dominant cellular architecture model. Thus, although significant progress has been achieved in the thorough understanding of wireless networking characteristics through the study of cellular systems, many of the developments are still not directly applicable to satisfy the needs of the wireless systems that require network architectures which may not follow the cellular paradigm. Such networks, sometimes referred to as wireless ad-hoc, or peer-to-peer, or multi-hop networks, consist entirely of wireless and often mobile nodes that may communicate either directly or via multiple hop paths that require

the support of intermediate nodes to achieve connectivity. Wireless Adhoc networks which have node mobility are called *Mobile Adhoc Networks*(MANETs).

Wireless ad-hoc networks are autonomous systems of fixed or mobile wireless nodes with routing capabilities, that may operate in a stand-alone fashion or as part of larger heterogenous network (e.g. in hybrid configurations). Although their development was initially driven by the needs of military networks(prior term used to describe them was *packet radio networks*), they are expected to embrace commercial systems as well, especially with the evolving use of personal communication services systems. It is envisioned that future applications will not be limited to the needs of the military (wireless digital battlefield, war-fighter's wireless internet etc.) but will include several civilian applications as well. For instance, they can be deployed in collaborative network scenarios (e.g. conferences or company meetings), where individual users need to share or exchange information without depending on a local network of access points. They are a viable solution in situations of emergency and rescue operations where the infrastructure-based network may not be available. Ad-hoc networks can also serve as platforms for micro-sensor networks that can be deployed in remote or inaccessible areas to collect, process and transmit various signals(e.g acoustic, seismic etc). for multiple purposes. And there are many potential applications such as home networks of heterogenous devices, industrial robotics and others.

The all-wireless architectures studied here exhibit several noticeable characteristics that make them quite different from the existing cellular systems and wireless LANs. In wireless ad-hoc networks the existence of a link between any

two nodes depends on a multitude of parameters, such as transmission power level, distance from the receiver, interference from other transmitters, propagation effects(e.g. multi-path, shadowing etc), type of antennas used (e.g. omnidirectional or highly-directional) etc. Nodes may move frequently and in an arbitrary fashion and /or may select to turn their power "OFF" at any time in order to conserve their battery reserves. Thus, the ad-hoc network topology is not stable, may change randomly and unpredictably and consists of varying capacity links.

Another crucial issue in wireless ad-hoc networks is the lack of central coordinator node. Although in some simulations, there may or may not be certain nodes in the role of local coordinators(similar to that of a base station), protocols designed to perform network control and signaling functions must operate in a distributed fashion. The overhead associated with collecting and maintaining global network state information prohibits the use of schemes that control operation through a central coordinator node. Moreover, distributed algorithms that do not depend on the status of a single node are not directly affected by individual node/link failures that occur often in such environments. The MANET working group [7] in the Internet Engineering Task Force is working on standardizing routing and other network layer protocols for MANETs. The Temporally Ordered Routing Algorithm (TORA) [5, 4, 6]is one such protocol. It is a highly distributed, scalable protocol. In this thesis, we present changes to the TORA routing protocol which causes in the improvement in its performance. Also, we describe a new network layer QoS mechanism called INORA which operates by effective interaction between TORA and INSIGNIA in-band signaling system.

1.1 Prior Work: Temporally Ordered Routing Algorithm (TORA)

1.1.1 Notation and Assumptions

A network is modeled as a graph G=(N,L), where N is the finite set of nodes and L is a set of initially undirected links. Each node $i \in N$ is assumed to have a unique node identifier(ID), and each link $(i, j) \in L$ is assumed to allow two-way communication (i.e nodes connected by a link can communicate with each other in either direction). Due to mobility of the nodes, the set of links L is changing with time (i.e. new links can be established and existing links can be severed). From the perspective of neighboring nodes, a node failure is equivalent to severing all links incident on that node. Each initially undirected link $(i, j) \in L$ may be subsequently be assigned one of the three states:

1. Undirected
2. Directed from *node i* to *node* j
3. Directed from *node* j to *node i*

If a link $(i, j) \in L$ is directed from *node i* to *node* j, *node i* is said to be "upstream" from *node* j, while node j is said to be downstream from node *i*. For each node i, the neighbors of i, $N_i \subset N$ is defined to be the set of nodes j such that $(i, j) \in L$. TORA requires the presence of an underlying link-level protocol, which ensures that the node i is always aware of its neighbors in the set N_i. It is also assumed

that all transmitted packets are received correctly and in the order of transmission. In current implementations of TORA, an underlying layer called *Internet Manet Encapsulation Protocol* (IMEP) is being used for reliable, in-order transmission of TORA packets. IMEP also gives TORA the neighborhood set N_i. Finally, since existing networks of this type typically employ omnidirectional antennas, it is assumed that when a node i transmits a packet, it is broadcast to all its neighbors in the set N_i .

1.1.2 Basic Structure of TORA

A logically separate version of TORA is run for each destination to which routing is required. Let us consider a single version running for a given destination.

The protocol can be separated into three basic functions:

- Creating Routes
- Maintaining Routes
- Erasing Routes.

Creating a route from a given node to the destination requires establishment of a sequence of directed links leading from the node to the destination. The function is only initiated when a node with no directed links requires a route to the destination. Thus, creating routes corresponds to assigning directions to links in an undirected network. The method used to accomplish this is a query/reply process,

which builds a directed acyclic graph (DAG) rooted at the destination.(i.e. destination is the only node with no downstream links). The protocol uses QRY and RPY packets for this functionality.

Maintaining routes refers to reacting to topological changes in the network in a manner such that routes to the destination are re-established within a finite time. This means that its directed portions return to a destination-oriented DAG within a finite time. TORA uses UPD packets for this functionality.

1.1.3 Description of the Protocol

At any given time, an ordered quintuple $H_i = (\tau_i, oid_i, r_i, \delta_i, i)$ is associated with each node $i \in N$. Conceptually, the quintuple associated with each node represents the "height" of the node defined by two parameters: a reference level and a delta with respect to the reference level. The reference level is represented by the first three values in the quintuple, while the delta is represented by the last two values. A new reference level is defined each time a node loses its last downstream link due to a link failure. The first value representing the reference level, τ_i, is the time tag set to the "time" of the link failure. The second value, oid_i, is the originator-ID(i.e.. the unique ID of the node which defined the new reference level). This ensures that the reference levels can be ordered lexicographically, even if link failures occur at a node simultaneously(i.e. with the same time tags). The third value, r_i, is a single bit used to divide each of the unique reference levels into two unique sub-levels. This bit is used to distinguish between the original reference level, and the higher reflected reference level. The first value representing the

delta, δ_i, is an integer used to order nodes with respect to a common reference level. This value is instrumental in the propagation of a reference level. Finally, the second value representing the delta i, is the unique ID of the node itself. This ensures that nodes with a common reference level and equal values of δ_i(and in fact all nodes) can be totally ordered lexicographically at all times.

Each node i (other than the destination) maintains its height, H_i. Initially, the height of each node in the network (other than the destination) is set to NULL. $H_i = (-,-,-,-,i)$. Subsequently, the height of each node i can be modified in accordance with the rules of the protocol. The height of the destination is always ZERO, $H_{i,did} = (0,0,0,0,did)$.

Each node i(other than the destination) also maintains a link-state array with an entry $LS_{i,j}$ for each link $(i,j) \in L$, where $j \in N_i$. The state of the links is determined by the heights H_i and $HN_{i,j}$, and is directed from the higher node to the lower node. If a neighbor j is higher than node i, the link is marked upstream (UP). If a neighbor j is lower than node i, the link is marked downstream (DN). When a new link $(i,j) \in L$ is established(i.e., node i has a new neighbor $j \in N_i$), node i adds entries for the new neighbor to the height and link-state arrays. If the new neighbor is the destination, the height entry is set to ZERO, $HN_{i,did} = (0,0,0,0,did)$; otherwise it is set to NULL, $HN_{i,j} = (-,-,-,-,j)$. The corresponding link-state, $LS_{i,j}$ is set as outlined above.

Route Creation

Creating routes uses QRY and UPD packets. A QRY packet consists of a destination-ID (did), which identifies the destination for which the algorithm is running. An UPD packet consists of a did, and the height of the node i which is broadcasting the packet, H_i.

Each node i (other than the destination) maintains a route-requested flag, RR_i, which is initially unset. Each node i (other than the destination) also maintains the time at which the last UPD packet was broadcast and the time at which each link $(i, j) \in L$, where $j \in N_i$ became active.

When a node with no directed links and an unset route-requested flag requires a route to the destination, it broadcasts a QRY packet and sets its route-requested flag. When a node i receives a QRY packet, it reacts as follows:

1. If the route-requested flag of the receiving node is set, it discards the QRY packet.

2. If the route-requested flag of the receiving node is not set and its height is non-NULL with $r = 0$, it first compares the time last UPD packet was broadcast to the time the link over which the QRY packet received became active. If a UPD packet has been broadcast since the time the link became active, it discards the QRY packet; otherwise, it broadcasts an UPD packet which contains its current height.

3. If the route-requested flag of the receiving node is not set and its height is non-NULL with $r = 0$, but it has a neighbor node whose height is non-

NULL with $r = 0$; it sets its height to $H_i = (\tau_j, oid_j, r_j, \delta_j + 1, i)$, where $HN_{i,j} = (\tau_j, oid_j, r_j, \delta_j, i)$ is the minimum height of its non-NULL neighbors with $r = 0$, updates all the entries in its link-state array LS and broadcasts a UPD packet which contains its new height.

4. If none of the above conditions hold true, the receiving node re-broadcasts the QRY packet and sets its route-requested flag.

If a node has the route-requested flag set when a new link is established, it broadcasts a QRY packet.

When a node i receives a UPD packet from a neighbor $j \in N_i$, node i first updates the entry $HN_{i,j}$ in its height array with the height contained in the received UPD packet and then reacts as follows:

1. If the route-requested flag of the receiving node is set and the height contained in the received UPD packet is non-NULL with $r = 0$, it sets its height to $H_i = (\tau_j, oid_j, r_j, \delta_j + 1, j)$-where $HN_{i,j} = (\tau_j, oid_j, r_j, \delta_j, j)$ is the height contained in the received UPD packet, updates all the entries in its link-state array LS, unsets the route-required flag and then broadcasts an UPD packet which contains its new height.

2. If the above condition does not hold true, the receiving node simply updates the entry $LS_{i,j}$ in its link-state array.

Route Maintenance

Route maintenance is only performed for nodes that have a non-NULL Height. Furthermore, any neighbor's height which is NULL is not used for the computations. A node i is said to have no downstream links if $H_i < HN_{i,j}$ for all non-NULL neighbors $j \in N_i$. This will result in one of five possible reactions depending on the state of the node and the preceding event. Each node (other than the destination) that has no downstream links modifies its height, $H_i = (\tau_i, oid_i, r_i, \delta_i, i)$ as follows:

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

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

$(\delta_i, i) = (0, i)$

i.e. node i defines a new reference level. The above assumes that node i has at least one upstream neighbor. If node i has no upstream neighbors, it sets its height to NULL.

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

$(\tau_i, oid_i, r_i) = max\{(\tau_j, oid_j, r_j) / j \in N_i\}$

$(\delta_i, i) = min\{(\delta_k |\ (\tau_k, oid_k, r_k) = max[\tau_j, oid_j, r_j] for\ j \in N_i) - 1, i\}$

In essence, node i propagates the reference level of its highest neighbor and selects a height which is lower than all neighbors with that reference level.

Case 3(Reflect): Node i has no downstream links(due to a link reversal following reception of a UPD packet) 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 essence, the same level(which has not been "reflected") has propagated to node *i* from all of its neighbors. Node *i* "reflects"back a higher sub-level by setting a bit *r*.

Case 4(Detect): Node *i* has no downstream links (due to a link reversal following the reception of an UPD packet), the ordered sets (τ_j, oid_i, r_i) are equal with $r_j = 1$ for all $j \in N_i$, and $oid_j = i$ (i.e., node *i* defined the level).

$(\tau_j, oid_i, r_i) = (-, -, -)$

$(\delta, i) = (-, i)$

In essence, the last reference level defined by node *i* has been reflected and propagated back as a higher sub-level from all its neighbors. This corresponds to detection of a partition. Node *i* must initiate the process of erasing invalid routes.

Case 5(Generate): Node *i* has no downstream links (due to link reversal following the reception of a UPD packet), the ordered sets (τ_j, oid_j, r_j) are equal with $r_j = 1$ for all $j \in N_i$, and $oid_j \neq i$ (i.e., node *i* did not define the level).

$(\tau_i, oid_i, r_i) = (t, i, 0)$ where *t* is the time of failure.

$(\delta_i, i) = (0, i)$

In essence, node *i* experienced a link failure(which did not require reaction) between the time it propagates a reference and the reflected higher sub-level returned from all neighbors. this is not necessarily a partition. Node *i* defines a new reference level.

Following the determination of its new height in cases 1, 2, 3 and 5, node i updates all the entries in the link-state array LS; and broadcasts an UPD packet to all the neighbors $j \in N_i$. The UPD packet consists of a did, and the new height of the node i which is broadcasting the packet, H_i. When a node i receives a UPD packet from a neighbor $j \in N_i$, node i updates the entries $HN_{i,j}$ and $LS_{i,j}$ in its height and link-state arrays. If the update causes a link reversal which results in node i losing its last downstream link, then it modifies its height as outlined as the cases above.

Route Erasure

Following the detection of a partition (case 4), node i sets its height and the height entry for each neighbor $j \in N_i$, to NULL (unless the destination is a neighbor, in which case the corresponding height entry is set to ZERO), updates all the entries in its link-state array LS, and broadcasts a CLR packet. The CLR packet consists of a did and the reflected reference level of node i, $(\tau_i, oid_i, 1)$. When a node i receives a CLR packet from a neighbor $j \in N_i$, it reacts as follows:

1. If the reference level in the CLR packet matches the reference level of node i, it sets its height and the height entry for each neighbor $j \in N_i$ to NULL (unless the destination is a neighbor, i which case the corresponding height entry 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 the reference level

of node i, it sets the height entry for each neighbor $j \in N_i$(with the same reference level as the CLR packet) to NULL, and updates the corresponding link-state array entries.

Thus, the height of each node in the portion of the network which was partitioned is set to NULL and all invalid routes are erased. If condition 2 causes node i to lose its last downstream link, it reacts as in case 1 of *Route Maintenance*.

Chapter 2

Query Localization in TORA

2.1 Introduction

The following problems exist with the querying mechanism in TORA:

- The extent of query propagation determines the size and complexity of the DAG, that gets built. The DAG built for *node j* might include nodes, that may never participate in communication with *j*.
- The maintenance of the DAG is an expensive affair. The topology changes in the underlying graph in the adhoc network trigger routing reactions. The route maintenance is performed by *update packets(UPDs)*. It has been found by simulations that the update packets contribute most to the routing traffic.
- If the *Route Requested flag*(RR flag) is set forever at a *node i*, any route

maintenance packet, sent out much later than the *route-building* phase can cause the DAG to be extended to node *i*. This is an additional overhead.

- Moreover, if the communication patterns are localized, flooding the query packets (QRYs) throughout the network caused considerable overhead. A practical illustration of this would be in a military application of MANETs on a battle field. The communication is mostly localized within the same unit.

To tackle these problems, we designed a query localization technique that would reduce the routing packet overhead in the case of limited and localized communication patterns.

2.2 Features of Query Localization

The following features were incorporated into the querying mechanism of TORA:

- Mention of node-id of the querying source.
- Sequence numbering in *Queries*(QRY's).
- Hop-count (Time to Live) for *Query packets*(QRY's).
- *Route Requested* (RR flags) which have finite expiration times.
- Separation of *route-building* and *route-maintenance* functions.

When a *node i* intends to start communicating with *node j,* it looks for the availability of a route in its routing table to the destination *j* . If it does not find any existent route (has NULL *Height*), it sends out a query(*QRY(i,seqno,*hc_1 *, j)*), where *seqno* stands for the sequence number of the query. hc_1 stands for the 1^{st} hop-count. Also, a timer is set with a value of QT_1. If the neighbors of node *i* do not have route to the destination, (have a NULL *Height*) they set their Route-Requested flag (RR flag) and start off the RR Timer (*RT*). If *node i* doesn't receive a reply from any of the nodes within the query time-out period, it sets out another query *QRY(i,seqno* $+ 1$*,*hc_2 *, j),* with a higher hop-count hc_2. The query timer in this case is set to a value of QT_2. When a *node l*, which has a route to the destination (has a non-NULL *Height*), it sets out a reply $RPY(H_l, l)$, where H_l is the Height of the node *l*.

The hop-count values of the queries are chosen such that $hc_1 < hc_2 < hc_3 <$ ..... $< hc_n$, where *n* is the diameter of the network.

The operations of Query Localization are described in the flow chart in figure 2.2. These modifications were implemented on a Linux test-bed of laptops and were tested in real-time. The protocol stack diagram for the Linux implementation of TORA is as shown in the figure . These modifications were also implemented and tested on the ns-2 simulator.

Merger of QRY's from two different sources

In the original TORA, the *Queries* (QRY's) were not associated with the query source. They were only associated with the destination. When a query for a

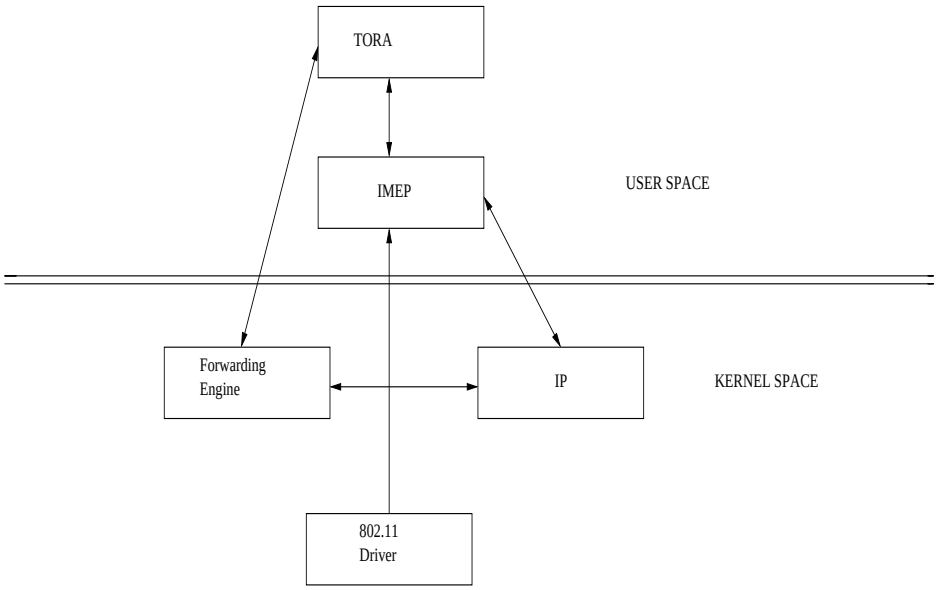


Figure 2.1: Protocol Stack for the Linux implementation of TORA

destination arrives at a node which has its *RR* flag set (i.e. a query has already passed that node for that particular destination), the query is dropped.

In the *Query-Localized* TORA, when a node with its RR flag set, receives a QRY, it allows the latest query to pass by. This is because the forwarding of the QRY will enable the nodes that are ahead to reset their RR-timers, which are very critical for the operation of the *Query-Localized* TORA. This is described in the figure .

2.2.1 Illustration of Query Localization

Consider an adhoc network with connectivity as illustrated in figure 2.4.

1. Node 9 wants to initiate a connection to node 7. TORA on node 9 sends

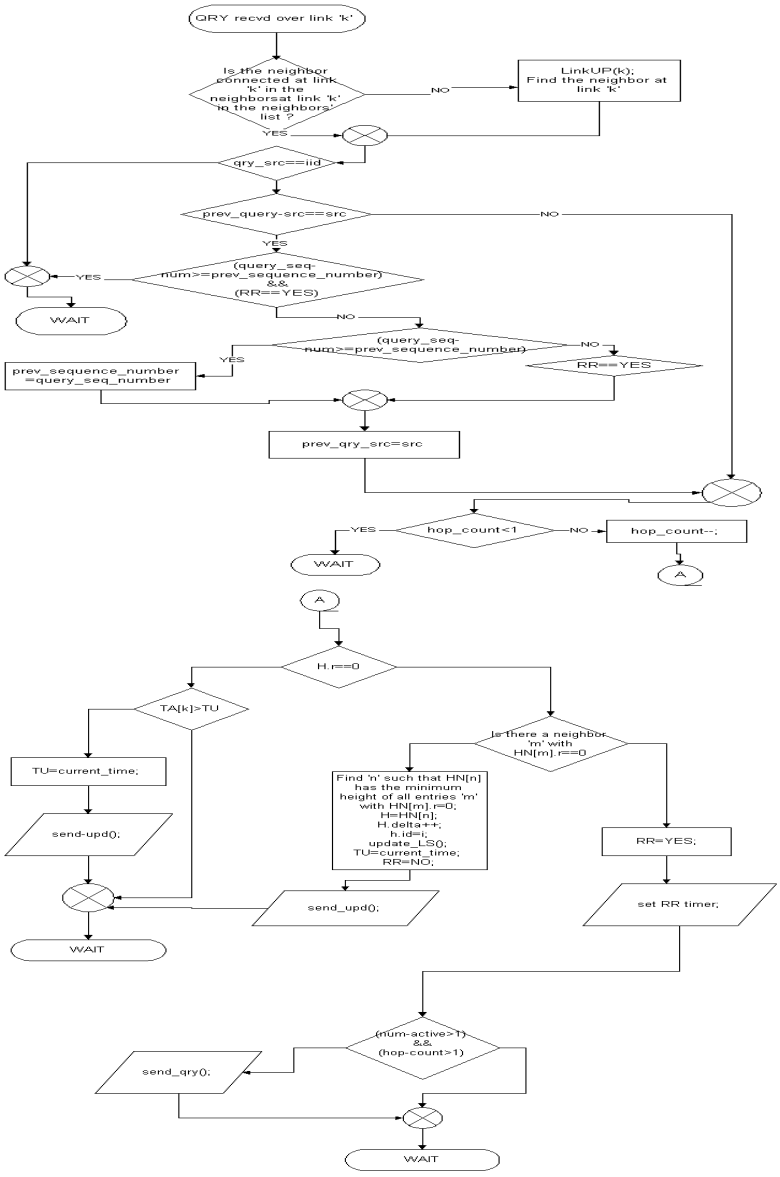


Figure 2.2: Flow chart for procedures executed on query reception

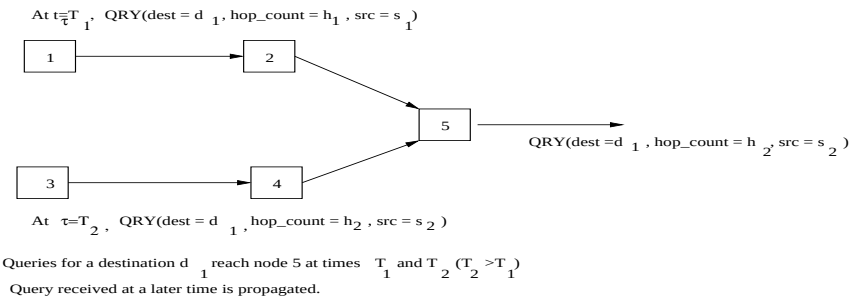


Figure 2.3: Merger of Queries from two different sources

out a *Query* (QRY) with the *src* field set to 9 and the *dest* field set to 7. The *hop_count* field is set to 1. (Note that the illustration here shows an arithmetic increase in the hop-counts. This can easily be extended to other kinds of increments).

2. The *Query-Timer* expires and the source node hasn't received a Reply (RPY). So, it initiates a new Query (QRY) with a *hop-count* field set to 2 as shown in figure 2.5.

3. The 3rd query(see figure 2.6) reaches the destination.

4. Reply (RPY) packets are broadcast by the destination 7. Nodes that have their RR flags set propagate the RPYs(see figure 2.7). The status of the RR flag at a node is determined by the *RR Timer (RT*).

5. Figure 2.8 illustrates the pruning of the DAG when the *query-localization* changes are applied.

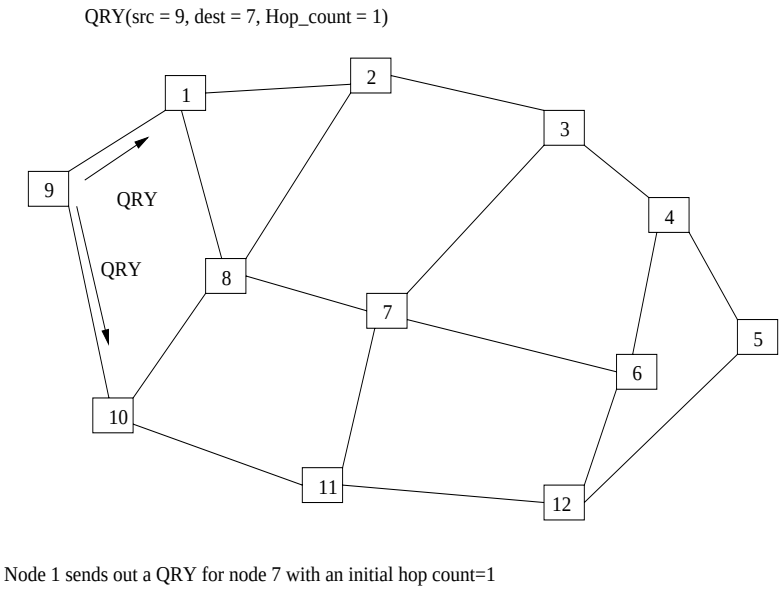


Figure 2.4: Node 9 is the source and node 7 is the destination.The first query is send out

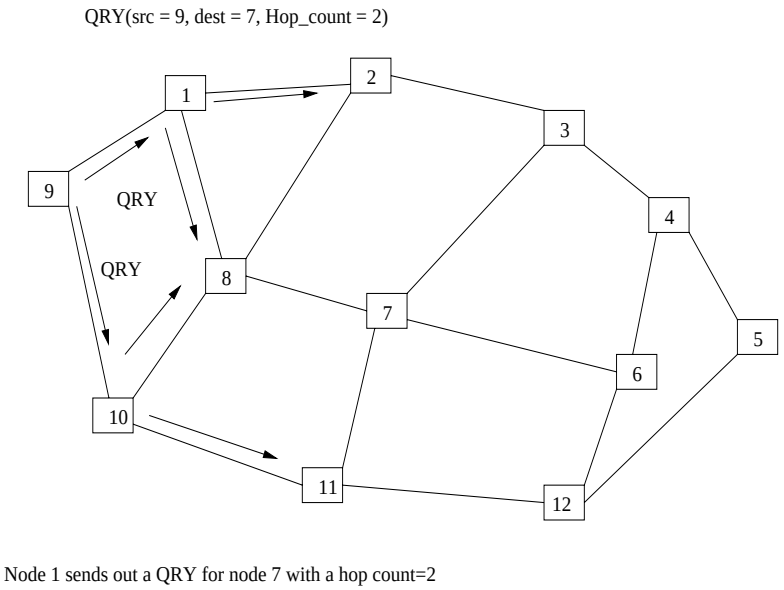


Figure 2.5: After the first Query timer expires, the second query is sent

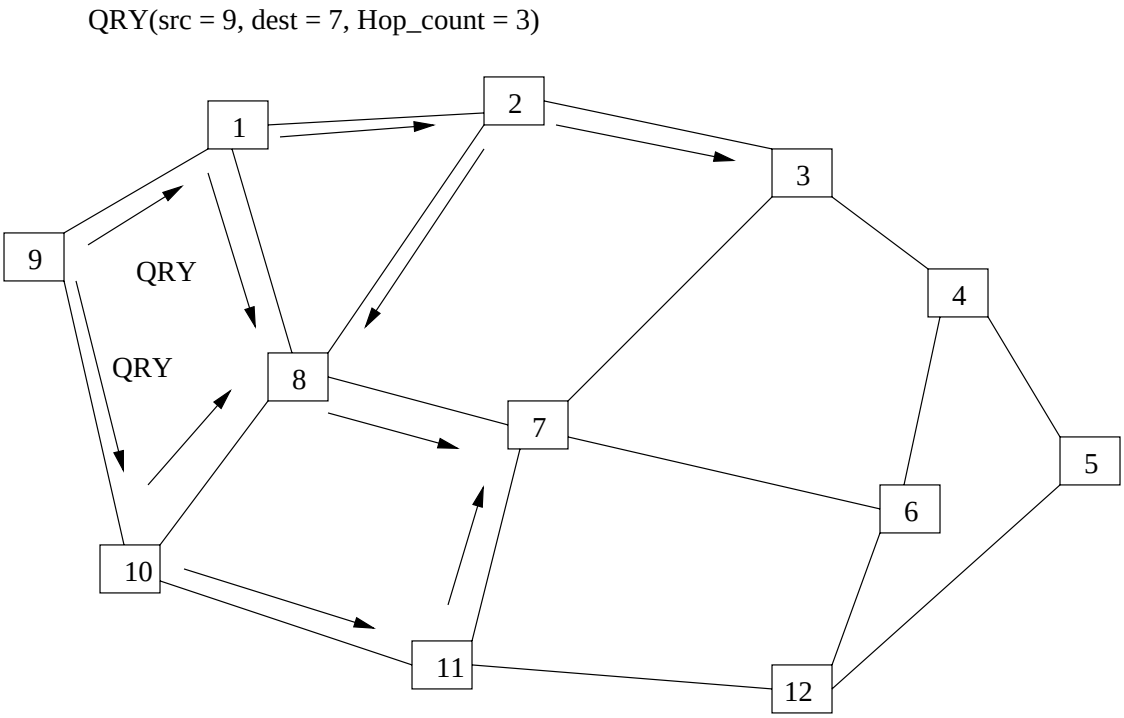


Figure 2.6: The expiration of the second Query timer causes the third query to be sent

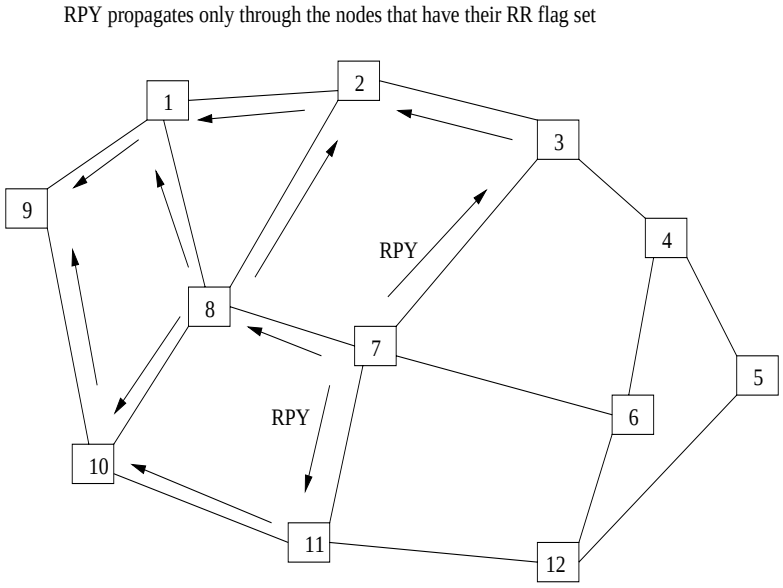


Figure 2.7: The RPY's propagate through the nodes that have their RR flag set

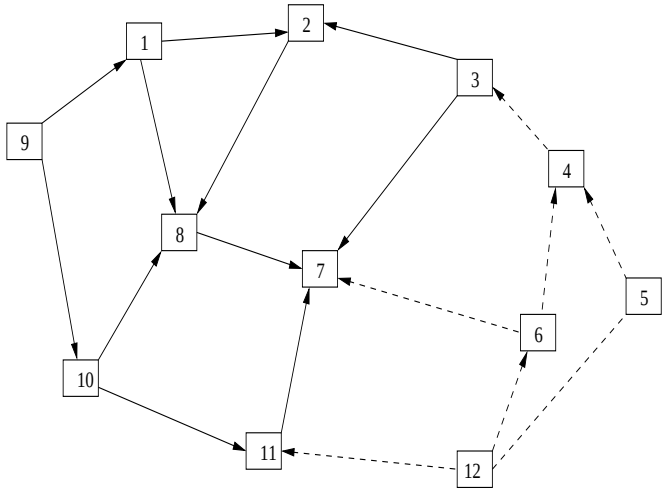


Figure 2.8: The DAG is pruned in query-localized TORA as compared to the original TORA

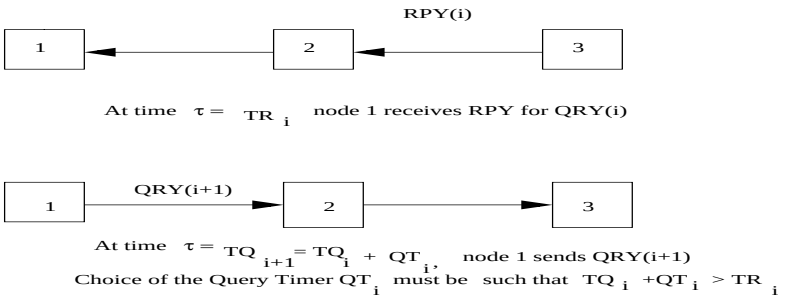


Figure 2.9: Choice of Query-Timer

2.2.2 Choice of Parameters

Hop-Count: This depends on the density of the mobile nodes in the network. For networks, that have uniform density of nodes, if we intend to have an arithmetic increase in the number of nodes covered in each *query-run*, we need to have the hop-count increase as $\sqrt{n}$, where n is the number of the query-run. i.e $hc_n \sim \sqrt{n}$.

Query-Timer: The choice of the query timer must be such that i^{th} query timer value should be greater than the expected round-trip time for covering hc_i hops. i.e. $QT_i > RTT_i$. Let $QT_i = k_q RTT_i$, where $k_q > 1$. k_q must be small enough to minimize delay in finding routes.

RR-Timer: The choice of the RR Timer must be such that the i^{th} RR-Timer value should be greater than the expected round-trip time for covering hc_i hops.

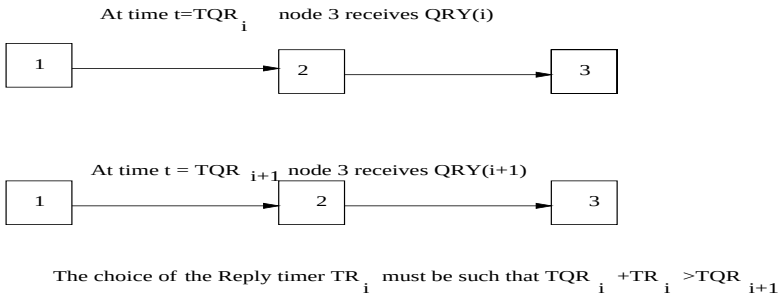


Figure 2.10: Choice of Reply-Timer

2.3 Simulations and Evaluation of Query Localization

The performance of TORA with the Query Localization incorporated has been tested by simulations in ns-2 simulator. We used the CMU wireless extensions for this purpose.[16]. 50 nodes are randomly placed in a rectangular area of 1500mx500m. Nodes move around this area in a random way-point model [17]with a maximum speed v_{max}= 40 m/s and pause time $p_t = 5sec$. Random waypoint model is described in the section 2.3.1. The communicating nodes among these 50 mobile nodes are picked at random. The source nodes generate CBR traffic with packet size 64 bytes and inter-packet interval is 0.1 sec$(5.12\ kbps)$. The communicating nodes are chosen at random amongst the 50 mobile nodes. IEEE 802.11 (operating in adhoc mode) was chosen as the underlying MAC layer. The transmission radius of the mobile nodes is chosen to be 250m. The underlying propagation model in the physical layer was chosen to be the *Two-Ray Ground Propagation Model*[8]. According to this model,

$P_r = (P_t * G_t * G_r * h_t^2 * h_r^2)/(d^4)$

where P_t is the transmission power. P_r is the power of reception. G_t is the transmitter antenna gain. G_r is the receiver antenna gain. h_t is the height of the transmitter antenna. h_r is the height of the receiver antenna.

A number of experiments were performed to compare the original TORA and the "Query-Localized" TORA. The results are explained in section 2.3.2.

2.3.1 Mobility model

Random Waypoint Model: The trajectory of a mobile node is specified by random way-points (x_1, y_1) , (x_2, y_2),.....,(x_n, y_n). A node moves from the i^{th}waypoint (x_i, y_i) to the $i+1^{th}$ waypoint (x_{i+1}, y_{i+1}) with a velocity $v = v_0$ where v_0 is chosen randomly between 0 and v_{max}. On reaching (x_{i+1}, y_{i+1}), the node pauses for the pause time p_t and changes direction towards the next random waypoint.

2.3.2 Simulation Results

The performance metrics that we consider for comparison are:

1. Routing Overhead

2. Percentage of packet delivery

Routing Overhead is the number of routing messages received by different nodes for a single data packet received.

Figure 2.11 depicts the comparison of routing overhead between the original TORA and the "*Query Localized*" TORA. It can be seen that the routing overhead in TORA is reduced substantially (by about 50%) with the query-localized version of TORA. As explained in section 2.1, the major portion of routing overhead in TORA is due to the *Update*(UPD) messages. These messages are used to maintain the DAG routing structure. By reducing the extent to which the *Query*(QRY) messages propagate, we have pruned the DAG to a great extent as described in sections 2.2 and 2.2.1. By doing this, far fewer UPD messages are propagated. Hence there is considerably low overhead. We also notice that as the number of connections in the network increase, the routing overhead reduces. The routing overhead in TORA consists of a reactive portion(QRY's and RPY's) and a non-reactive portion(UPD's). As explained earlier, the non-reactive portion constitutes the majority of routing messages and it depends on the size of the DAG. As the number of connections increase, previously built DAG's are re-used (either in part or entirety), thus precluding the necessity for non-reactive overhead (UPDs). This causes the reduction in the routing overhead. We found that both the versions of TORA performed identically in terms of end-to-end delay of packets.

We varied the values of the query-timer and observed the performance of the *Query-Localized* TORA. It can be seen from the figure 2.12, the routing overhead is the least when the query timer is chosen as a value between 100msec and 200msec. The percentage of packet delivery is also the highest for the *Query-Timer* value in this region. This validates the point illustrated in section 2.2.2. The performance of the protocol was found to be insensitive to RR Timer (RT).

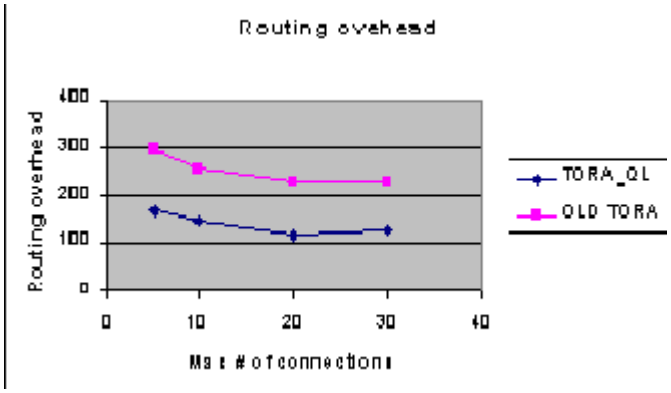


Figure 2.11: Comparison of Routing Overhead vs #of connections.

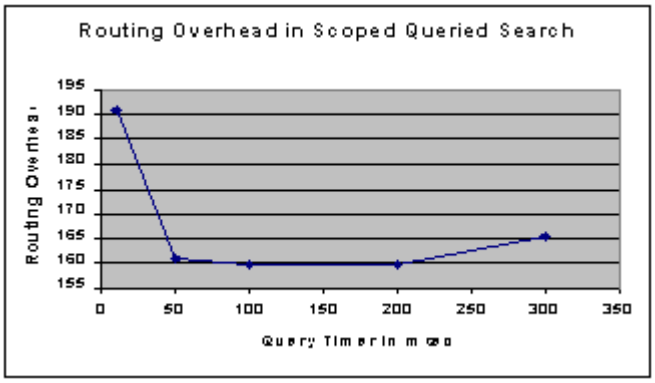


Figure 2.12: Routing Overhead vs Query Timer values

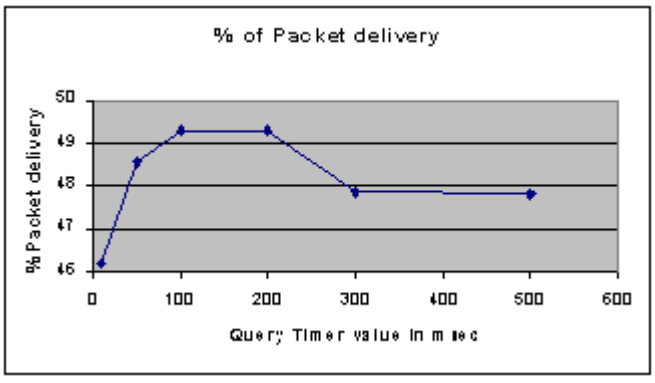


Figure 2.13: Packet Delivery Percentage vs Query Timer Values

Chapter 3

INORA - A Unified Signaling and Routing Mechanism for QoS Support in Mobile Adhoc Networks

3.1 Introduction

Providing quality of service (QoS) support for the delivery of real-time audio, video and data in mobile adhoc networks thus, presents a number of technical challenges. Mobile adhoc networks can be quite large, which makes the problem of network control very difficult. In fixed-wired networks, most QoS schemes use hard-state resource reservations and explicit *connection-establishment* and *connection tear-down* mechanisms. Dynamically changing topology of mobile adhoc networks due to mobility of the nodes calls for the soft-state reservation[1] of re-

sources across the network for providing QoS support, as against hard-state reservations in wired networks. QoS support can be provided at the MAC layer(E.g. MACA/PR[12]) or at the network layer[9][1][13]. In this chapter, we present a QoS framework at the network layer based on the TORA routing protocol and the INSIGNIA in-band signaling system.

3.2 Approaches to network layer QoS support in Mobile Adhoc Networks

Various network layer mechanisms have been proposed for QoS support in mobile adhoc networks. They can be be broadly categorized as the following depending on the degree of coupling between the QoS resource reservation mechanisms and routing protocol.

1. QoS Routing
2. QoS signaling with no interaction between the QoS resource reservation mechanism and the routing protocol.
3. QoS signaling with interaction between the QoS resource reservation mechanism and the routing protocol.

QoS Routing QoS routing protocols search for routes with sufficient resources for the QoS requirements. QoS routing protocols work with the resource man-

agement mechanisms to establish paths through the network that meet end-to-end QoS requirements, such as delay or jitter bounds, bandwidth demand.[14]

E.g.: CEDAR[9]

Here, the QoS provision mechanism is intrinsically tied to the routing protocol. QoS Routing is difficult in MANETs.

Firstly, the overhead of QoS routing is too high for the bandwidth limited MANETs because there needs to be some mechanism for a mobile node to store and update link information.

Secondly, because of the dynamic nature of MANETs, maintaining precise link information is very difficult.

Thirdly, the traditional meaning that the required QoS should be maintained once a feasible path is established is no longer true. The reserved resource may not be guaranteed because of the mobility-caused path breakage or power depletion of the mobile hosts.

QoS Signaling QoS signaling is used to reserve and release resources, set up, tear down and renegotiate flows in the network. Soft-state reservations are better in mobile adhoc networks, because of the highly dynamic conditions in the network. [2]

QoS Signaling without interaction between the QoS mechanism and the routing protocol The signaling mechanism can be operated independent of the routing protocol. The routing protocol provides the route between the source

and destination of a flow. The signaling protocol establishes resources along the route chosen by the routing protocol. Here, the routing protocol is completely decoupled from the signaling mechanism.

E.g. INSIGNIA[1][2]

QoS Signaling with interaction between the QoS mechanism and the routing protocol Here, there is a loose coupling between the QoS mechanism and routing protocol. The coupling is looser than in *QoS Routing.* The routing protocol provides a route between the source and destination of the flow. The signaling mechanism provides feedback to the routing protocol regarding the route chosen and asks the routing protocol for alternate routes if the route provided doesn't satisfy the QoS requirements. The INORA (INSIGNIA+TORA) scheme that is presented in this chapter belongs to this category. In INORA, INSIGNIA makes a call-back to TORA asking for alternate routes when the current route fails to meet the QoS requirements. TORA is a good choice for the routing protocol in this case. This is because, TORA operates by creating a routing structure called a *Directed Acyclic Graph* (DAG), which gives multiple routes from a source to the destination.

3.3 Overview of INSIGNIA-based Wireless flow management system

The INSIGNIA QoS signaling system is a part of the *wireless flow management* that supports the delivery of *adaptive real-time services* in dynamic mobile adhoc networks.

The goal of INSIGNIA-based *wireless flow management* is to support the delivery of adaptive real-time services to mobile adhoc hosts under time-varying conditions. The adaptive service model allows packet audio, video and real-time data applications to specify their maximum and minimum bandwidth needs. INSIGNIA plays a central role in the establishment of resources, at the intermediate routers between the source-destination pairs. Based on the availability of end-to-end resources, wireless flow management attempts to provide assurances for the minimum and maximum bandwidth needs depending on resource availability. In addition to supporting adaptive real-time services the service model also supports IP best-effort packet delivery.

The following are the main modules of the of the INSIGNIA-based *wireless flow management system*:

- *Packet Forwarding Module*: This classifies the incoming packets and forwards them to the appropriate module(viz. routing, INSIGNIA, local applications, wireless packet scheduling modules). Signaling messages are processed by INSIGNIA and the data packets delivered locally or forwarded to the packet scheduling module.

- *Routing Module*: This is a routing protocol which dynamically tracks the changes in an adhoc network topology making the routing table visible to all the intermediate forwarding modules. (E.g. INSIGNIA, packet forwarding). Wireless flow management requires the availability of such a MANET routing protocol. e.g. Temporally Ordered Routing Algorithm (TORA), Dynamic Source Routing Protocol (DSR), AdHoc On Demand Distance Vector Routing Protocol (AODV).

- *INSIGNIA Module*: This establishes, restores, adapts and tears down real-time flows. Flow restoration algorithms respond to dynamic route changes due to mobility. Adaptation algorithms respond to changes in available bandwidth. Based on an in-band signaling approach that explicitly carries the control information in the IP packet header, flows can be rapidly established, restored, adapted and released in response to wireless impairments and topology changes. Because of this dynamic environment, network management is based on soft-state, which is well suited to managing flow state in mobile adhoc networks.

- *Admission Control Module*: This module allocates bandwidth to flows based on the adaptive real-time service maximum and minimum bandwidth request in the data packet. Once resources have been allocated, they are periodically refreshed by the soft-state mechanism through the reception of data packets. Admission control is testing is based on the measured channel capacity/utilization and requested bandwidth.

- *Packet Scheduling Module*: This responds to location dependent channel conditions experienced in wireless networks. Without taking the channel state into account, a mobile node may receive significantly less service than it is supposed to, while another node may get more.

- *Medium Access Controller Module:* This provides quality of service driven access to the shared wireless media for adaptive real-time services and best-effort services.

The wireless-flow management system is illustrated in figure 3.1

The INSIGNIA[2] in-band signaling system plays an important role in establishing, adapting, restoring and terminating end-to-end reservations for flows. INSIGNIA is designed to be light-weight in terms of the amount of bandwidth consumed for network control. It operates by setting up soft-state reservations for a flow across the path from the source of the flow to the destination of the flow in a mobile adhoc network. INSIGNIA uses the IP Options field in the IP header to convey the signaling information. See fig.3.2. The following are the IP options fields:

- **Service Mode:** When a source node wants to establish a reserved QoS flow to a destination node, it sets the RES bit of the INSIGNIA IP option service mode of a data packet and sends the packet toward the destination. On reception of a RES packet, the intermediate nodes execute *admission control* to accept or deny the request. When a node accepts a reservation request, resources are committed and subsequent packets are scheduled accordingly.

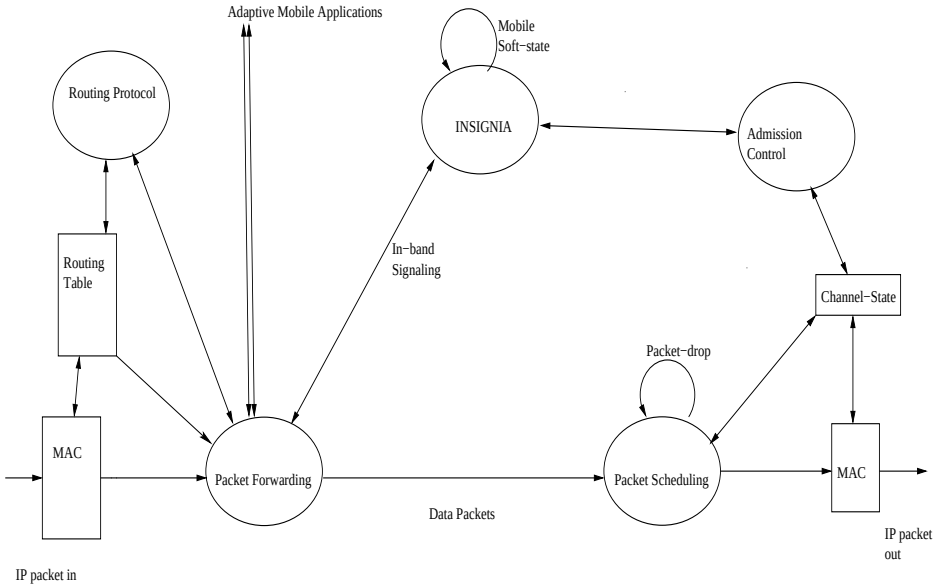


Figure 3.1: Wireless Flow Model for INSIGNIA

If the reservation is denied, packets are treated as *best effort mode* (BE) packets.

- **Payload Type:** This option carries an indication of the payload type, which identifies whether the packet is of the type *base QoS* (BQ) or *enhanced QoS* (EQ)[2]

- **Bandwidth Request:** The bandwidth request allows us to specify its maximum (MAX) and minimum (MIN) bandwidth requirements for adaptive services. During request establishment, the bandwidth indicator reflects the resource availability at the intermediate nodes along the path between source-destination pairs of different flows.

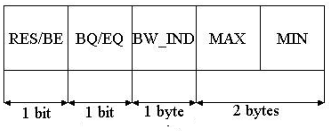


Figure 3.2: INSIGNIA IP options

3.3.1 Admission Control

A source of a QoS flow sets out data packets with its service mode IP options field set to RES. All the intermediate nodes which receive packets with their *service mode* field set to RES perform admission control. At the first node where the admission control fails, the service mode is changed to BE (best effort).

Admission control failure occurs when either of the following occurs:

- The node is unable to allocate at least the minimum required bandwidth (BW_{min}) for the flow.
- There is congestion at the node, i.e the queue-size at the node has exceeded a threshold.$(Q > Q_{th})$

In fig.3.3, we illustrate the connectivity of a MANET with a graph. The source of a QoS flow is node 1. The destination is node 5. Let the path given by the routing protocol be $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$. node 4 is the first node at which an admission control failure occurs because of either of the conditions mentioned above. The reserved flow turns into a *best effort* flow.

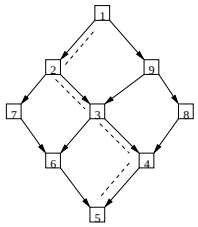


Figure 3.3: INSIGNIA-Admission Control fails at node 4

3.3.2 QoS Reporting

QoS reporting is used to inform source nodes of the ongoing status of the flows. Destination nodes actively monitor ongoing flows, inspecting the *status information* (e.g. Bandwidth Indicator) and *measured delivered QoS* (e.g packet loss, throughput etc.). Although the QoS reports are basically generated periodically according to the sensitivity of application, QoS reports are sent immediately when required. The source, on the reception of a QoS report indicating a flow degrade from *reserved* to *best effort* flow may downgrade the flow. Here the feedback is end-to-end from the source to destination. INSIGNIA doesn't take any help from the network with regard to redirecting the flow along routes which are able to provide the required QoS guarantees. In INORA (See section 3.4) we describe a mechanism that takes help from the network and the feedback about the capability of intermediate nodes to admit flows is given to the routing protocols on a hop-by-hop basis.

3.4 INORA

In INORA, we make use of feedback on a per-hop basis to direct the flow along route that is able to provide the QoS requirements of the flow. We make use of the INSIGNIA in-band signaling system and TORA[4] routing protocol in the INORA scheme. The wireless flow management system described in the section 3.3 modified to give feedback from the signaling system to the routing protocol. This is illustrated in figure 3.3.

TORA operates by creating a *Directed Acyclic Graph* (DAG) rooted at the destination as described in the section 1.1.3. We use this routing structure to direct the flow through routes that are able to provide the resources for the flow according to the QoS requirements of the flow. We present two schemes under the INORA framework.

1. Coarse feedback scheme.

2. Fine feedback scheme.

3.4.1 Coarse Feedback Scheme

The operations of the coarse-feedback scheme of INORA are described illustrated through the following example :

Consider a QoS flow being initiated with node 1 as the source and node 5 as the destination.

1. Let the DAG created by TORA be as illustrated in fig.3.5

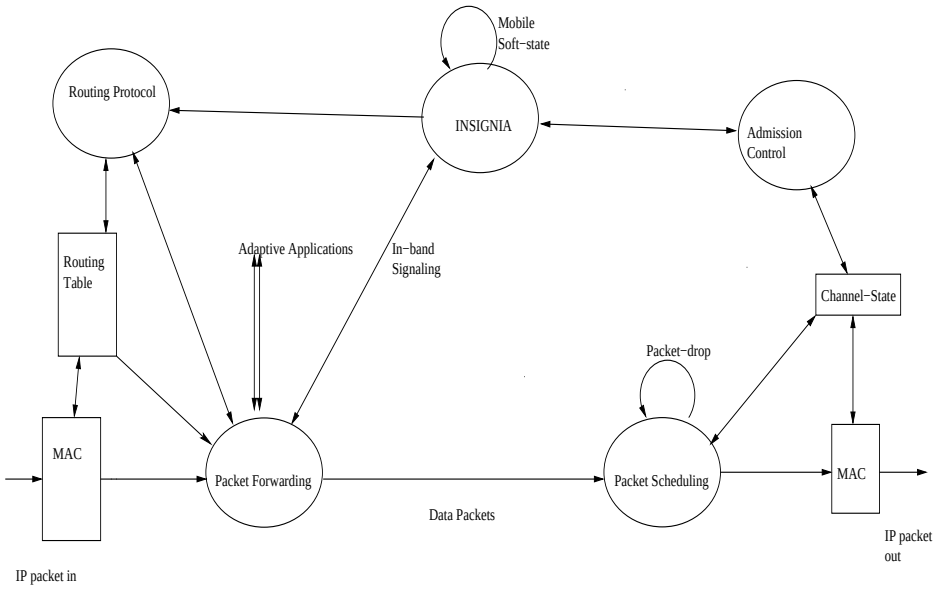


Figure 3.4: Wireless Flow Management system in INORA

2. Let $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$ be the path chosen by the TORA routing protocol.(See fig. 3.5)

3. INSIGNIA tries to establish soft-state reservations for the QoS flow along the path. Node 4 is the first node at which admission control for the flow fails, (because of either condition mentioned in section 3.3.1Node 4 sends an out-of-band *Admission Control Failure* (ACF) message to its previous hop (node 3).(See fig.3.6)

4. Node 3 realizes that the next hop 4 is not good for the current flow and re-routes the flow through another downstream neighbor (node 6) provided by TORA. (See fig.3.7)

5. If node 6 is able to admit the flow, the flow gets the required reservations all

along the path. The new path would be $1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 5$(See fig.3.7)

6. If node 6 is unable to admit the flow, it sends an ACF message to node 3(its previous hop).(See fig.3.8)

7. Node 3 realizes that it has exhausted all the downstream neighbors that it was provided by TORA. So, it sends a Cumulative *Admission Control Failure* message to its previous hop (node 2), indicating that none of its down-stream neighbors can accommodate the flow.(See fig.3.9)

8. Node 2 now, tries with its other down-stream neighbors for the possibility of a path that can give the required reservations to the flow.

The following things can be noted:

- As a result of this scheme, it is possible that different flows between the same source and destination pair can take different routes, as can be seen from fig.3.10, that to go from node 1 to node 5, flow 1 takes the path $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$ and flow 2 takes the path $1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 5$

- While INORA is trying to find a good route for the flow following admission control failure at an intermediate node, the packets are transmitted as *best effort* (BE) packets from the source to the destination. It should also be noted that there is no interruption in the transmission of a flow that has not been able to find a route in which resources have been reserved all the way from the source to the destination.

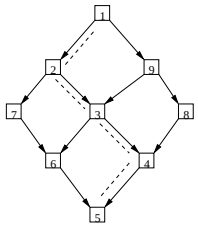


Figure 3.5: INORA Coarse-Feedback
node 4 is a bottle-neck node. Admission Control Fails at 4

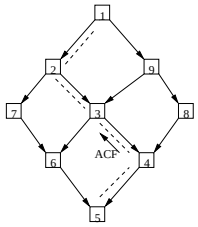


Figure 3.6: INORA Coarse-Feedback
node 4 sends an out-of-band ACF to the previous hop (node3)

- Because of the nature of the *Directed Acyclic Graph* (DAG), INORA tries to get a route which satisfies QoS requirements locally. When this fails, the search for a route which satisfies the QoS requirement becomes more global. In the worst case, we would have searched the entire DAG for a QoS route.

- Also, the scope of search for the routes is the DAG. INORA only chooses an appropriate route from the set of routes given by TORA. It doesn't trigger any route-querying mechanism to find new routes which will be good *QoS-wise*.

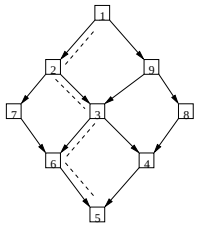


Figure 3.7: INORA Coarse-Feedback
node 3 redirects the flow to node 6.

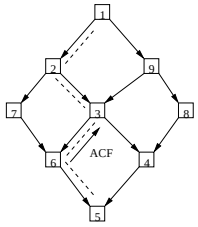


Figure 3.8: INORA Coarse-Feedback
If node 6 fails to admit the flow, 6 sends an ACF message to 3

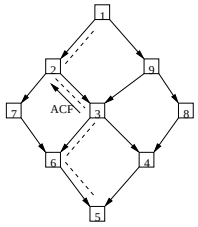


Figure 3.9: INORA Coarse-Feedback
node 3, having exhausted all its next-hops, sends an a cumulative ACF to its previous hop 2.

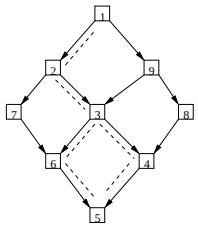


Figure 3.10: INORA Coarse-Feedback
Different flows between same source-destination pair can take different routes

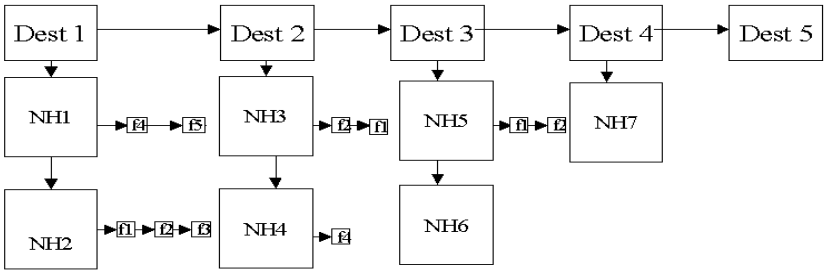


Figure 3.11: TORA Routing Table in INORA

Implementation Details

When a node X receives an Admission Control Failure (ACF) message from its downstream neighbor Y, it blacklists the downstream neighbor Y. Associated with the black-list entry, is a timer, which makes sure that the downstream neighbor Y is black-listed long enough. The node Y must be black-listed for the expected period of time required by INORA to search for a QoS route. This time is $O(E)$, where E is the number of links in the network at any given time. The TORA routing table is restructured in INORA as shown in fig.3.11

Associated with every destination, there is a list of next hops which is created by TORA. With the feedback that TORA receives from INSIGNIA in INORA, TORA associates the *next-hops* with the flows that they are suitable for. So, a routing look-up in INORA is based on the ordered pair $(destination, flow)$. If TORA doesn't have the information about the best route for the given flow, the routing look-up is just based on the destination. In that case, TORA gives the downstream neighbor with the least *Height*[4] metric. If any of the nodes is not INORA aware, normal operations of INSIGNIA and TORA continue.

3.4.2 Class-Based Fine Feedback Scheme

In this scheme, we divide the (BW_{min}, BW_{max}) interval into N classes, where BW_{min} is the minimum bandwidth required by a flow and BW_{max} is the maximum bandwidth required by the QoS flow. The IP options field in the IP header which carries the INSIGNIA information, now carries an additional *class* field. This field signifies the amount of bandwidth that has been allocated for the flow along the path.

The operation of the protocol is illustrated by the following example:

Consider a QoS flow being initiated with node 1 as the source and node 5 as the destination, with minimum bandwidth requirement BW_{min} and maximum bandwidth requirement BW_{max}. Let the flow be admitted with class m $(m < N)$ at node 1.

1. Let the DAG created by TORA be as shown in fig.3.12
2. Let $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$ be the path chosen by the routing protocol.(See fig. 3.12)
3. INSIGNIA tries to establish soft-state reservations for the QoS flow along the path.
4. Node 2 is able to admit the flow with class m as was requested by its previous upstream hop, node 1.
5. Suppose that node 3 has admitted the flow with class l, but has not been able to allocate the bandwidth of class m, as requested by its previous hop 2.$(l < m)$ (See fig.3.13)

6. Node 3 now, sends an *Admission Report message* $(AR(l))$ to the upstream previous hop(node 2), indicating its ability to give *class l* bandwidth to the flow.(See fig. 3.13)

7. Node 2 splits the flow in the ratio of l to $m-l$ and forwards the flow to node 3 and node 7 respectively, in that ratio. This means that the flow of class m has been split into two flows of class l and $m-l$ and is forwarded to nodes 3 and 7 respectively.(See fig. 3.14)

8. Suppose that node 7 is unable to give *class* $(m-l)$ as requested by the upstream previous hop 2, but is only able to give class n $(n < m-l)$. 7 sends an Admission Report message $(AR(n))$ to the upstream previous hop , node 2.(See fig.3.15)

9. Now node 2, realizing that its downstream neighbors have been unable to give the *class m*, which it was requested, informs its ability to give a class $l+n$ $(l+n < m)$ by sending a cumulative *Admission Report* $AR(l+n)$ to its previous hop 1. (See fig. 3.16)

10. Now, node 1 tries to find another downstream neighbor, which might be able to accommodate the flow with class $(m-(l+n))$

The following things can be noted:

- When a node is unable to admit a flow, either due to its inability to give the flow the requested minimum bandwidth or due to congestion at a node, it is not able to allocate the minimum bandwidth BW_{min} required by the flow,

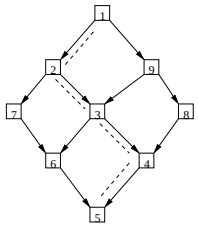


Figure 3.12: INORA Fine-Feedback
node 3 has admitted the flow with class l, but is not able to give the bandwidth-class that the node 2 (previous hop) is able to give, say m, $m > l$

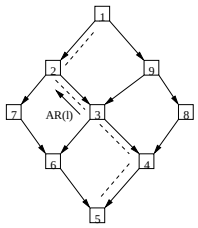


Figure 3.13: INORA Fine-Feedback
node 3 sends Admission Report $AR(l)$ to previous hop (node 2)

the *Admission Control Failure* messages as in the *coarse-feedback* scheme described in section 3.4.1are sent. So, the *fine-feedback scheme* is a super-set of the *coarse-feedback scheme*.

- *Fine-feedback scheme*, like the *coarse-feedback scheme* first tries to search for a QoS route, which can give the requested bandwidth *class* locally. The search becomes more global if it is not able to find the QoS route which gives the required cumulative class locally.

- A single flow can get split, and the packets can take different routes from the source to the destination. (See fig.3.17)

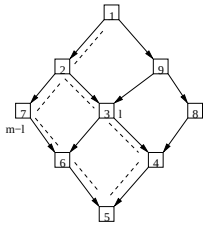


Figure 3.14: INORA Fine-Feedback
node 2 splits the flow among the next hops, 7 and 3 in the ratio $m-l$ to l

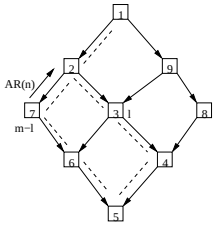


Figure 3.15: INORA Fine-Feedback
node 7 is unable to give $m-l$, but only $n < m-l$. It sends $AR(n)$ upstream

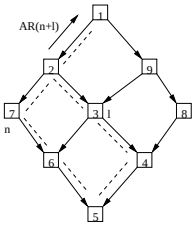


Figure 3.16: INORA Fine-Feedback
node 2 sends $AR(n+l)$ indicating the bandwidth that it can support

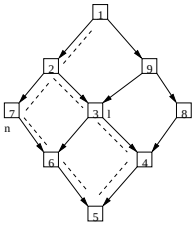


Figure 3.17: INORA Fine-Feedback
A single flow gets split and takes different paths to the destination

Implementation Details

Consider the example mentioned in 3.4.2. When node 2 receives an $AR(l)$ from node 3 and $AR(n)$ from node 7, indicating the ability of the downstream neighbors to give class n and class l to the flow as against the requested class m $(l+n<m)$, node 2 makes a note of the class, that each downstream neighbor has been able to allocate in the *Class Allocation List* and associates timers with those entries. The TORA routing tables here, are similar to the *coarse-feedback scheme* as illustrated in fig. *3.11*. There is an additional *class* field in the *flow* entries of the routing table. The routing table look-ups are made on the basis of the ordered 3-tuple $(destination, flow, class_{req})$ where

$destination$ stands for the destination for which we are looking up routes.

$flow$ stands for the flow for which we are looking up routes.

$class_{req}$ stands for the bandwidth class requested by the flow.

3.5 Simulations

We performed *ns-2* simulations to evaluate the INORA framework. The INSIGNIA code was obtained from COMET group, Columbia University [15]. The TORA *ns-2* code from CSHCN, University of Maryland[18] was used. We made modifications to the INSIGNIA and TORA code to incorporate the INORA scheme. CMU Monarch wireless extensions [19]for *ns-2* were used. We ran experiments with the INORA schemes(*coarse-feedback* and *fine-feedback*), and the original INSIGNIA and TORA, running independent of each other without the feedback.

In the INORA *fine-feedback scheme,* we chose the number of classes, $N = 5$

The node mobility follows *Random Way-point Model*(see section 2.3.1). The underlying MAC layer is IEEE 802.11 and the wireless propagation model is *the Two-Ray Ground Propagation model* as explained in section 2.3.

We have 10 flows, 3 of which have QoS requirements and the remaining 7 flows don't have QoS requirements. The sources generate CBR traffic. The simulations have been run for a simulation time of 300sec. We considered two different scenarios. Scenario B has QoS sources transmitting at a higher data rate and the QoS flows have a higher reservation requirements.

Scenario_A: The 50 nodes are spread out randomly in a rectangular grid of $1500mX500m$. The nodes more with a velocity uniformly distributed between 0-40m/s. The 3 QoS flows generate traffic at a data rate of $81.92.kbps$ The 7 non-QoS flows generate traffic at a data rate of 40.96kbps. The QoS flows ask for a reservation of $BW_{min} = 81.92$ $kbps$, and $BW_{max} = 163.84\ kbps$

Scenario_B: The 50 nodes are spread out randomly in a rectangular grid of $1500mX500m$. The nodes move with a velocity uniformly distributed between 0-40 m/s. The 3 QoS flows generate traffic at a data rate of $136.533\ kbps$. The 7 non-QoS flows generate traffic at a data rate of $40.96\ kbps$. The QOS flows ask for a reservation of $BW_{min} = 136.533$ $kbps$, $BW_{max} = 273.066\ kbps$

Scenario_C: The 50 nodes are spread out randomly in a rectangular grid of

$1500mX300m$. The nodes move with a velocity uniformly distributed between 0- 20 m/s. The 3 QoS flows generate traffic at a data rate of 81.92 $kbps$. The 7 non-QoS flows are generated at a data rate of $81.92kbps$. The QoS flows request for a reservation of $BW_{min} = 81.92kbps$ and $BW_{max} = 163.84kbps$.

3.5.1 Results

We evaluate the performance of INORA schemes by observing the end-to-end delay of the packets and the control message overhead.

The average end-to-end delay for QoS flows in different schemes for scenario_C is shown in Table 3.1. We see that the INORA *coarse-feedback* has lesser average delay than INSIGNIA and TORA operating without feedback. The INORA fine-feedback scheme performs better than the INORA *coarse-feedback* scheme. This is because the INORA feedback schemes try to find paths which can allocate the requested bandwidth reservations to the QoS flows. The *fine-feedback* scheme does this in a much fine-grained manner when compared to the *coarse-feedback* scheme. So, we have the fine-feedback scheme performing better than the *coarse-feedback* scheme.

Table 3.2 shows the average end-to-end delay experienced by all packets (from both QoS and non-QoS flows). We see that the INORA schemes perform better than INSIGNIA and TORA operating without feedback. It can be seen that the average delay is reduced by 80% from the no feedback case in the INORA *coarse-feedback* scheme. By trying to find the paths which can allocate the re-

Table 3.1: Average delay of QoS packets

QoS Scheme	Avg. end-to-end delay(sec)
No feedback	0.049
Coarse feedback	0.043
Fine feedback	0.034

Table 3.2: Average delay of all packets(QoS/non-QoS packets)

QoS Scheme	Avg. end-to-end delay(sec)
No feedback	0.108
Coarse feedback	0.021
Fine feedback	0.062

quired bandwidth to the flows and by performing load balancing in the network, the INORA schemes ensure that the overall congestion in the network is reduced. So, we have a lesser end-to-end delay for the packets. We find that INORA *fine-feedback* scheme has higher average end-to-end delay (for QoS and non-QoS packets together) when compared to the INORA *coarse-feedback* scheme. This is because the INORA fine-feedback scheme does fine-grained feedback(by splitting the flows along different paths), which benefits the QoS flows more at the cost of non-QoS flows. Table 3.3 shows the overhead in the INORA schemes. We find that the number of INORA control messages transmitted per every QoS data packet delivered is more for the fine-feedback scheme as compared to the coarse-feedback scheme. This is expected because of the additional Admission Report messages for fine-grained control in the fine feedback scheme.

We find that the delays in scenario_A and scenario_B are higher than in scenario_C. This is because of higher mobility in scenario_A and scenario_B. Also, because of the larger area, there are more partitions in scenario_A and scenario_B.

Table 3.3: Overhead in INORA schemes

QoS Scheme	No. of INORA pkts/data pkt
Coarse feedback	0.0174
Fine feedback	0.1833

Now, we compare the performance of the 3 QoS schemes between scenario_A and scenario_B. Scenario_B generates traffic at a higher data rate and the reservation requirements are also more than in scenario_A.

We find that INORA with coarse-feedback and fine-feedback schemes gives almost the same packet delivery rate as INSIGNIA and TORA acting without feedback, in both Scenario_A and Scenario_B. (See fig.3.18 and fig.3.19)

The average delay on a per-flow basis for QoS flows in Scenario_A is shown in fig.3.20. The average delay on a per-flow basis for non-QoS flows is shown in fig.3.21. The average delay on a per-flow basis for QoS flows in scenario_B is shown in fig.3.22. The average delay on a per-flow basis for non-QoS flows is as shown in fig.3.23. It can be seen that the delay is flow dependent. The INORA schemes do better *average delay-wise* for most of the flows when compared to INSIGNIA and TORA running without interaction. Also, INORA does better when there are higher bandwidth requirements (Scenario_B) than when the flows have lower bandwidth requirements (Scenario_A). The INORA fine-feedback scheme does better when compared to INORA coarse-feedback scheme in Scenario_B.

The plot of average delay vs. simulation time in scenario_A for all data packets (QoS and non-QoS) is as shown in 3.24. The same plot in Scenario_B is shown in fig.3.25. In Scenario_B , the INORA fine-feedback scheme does the best, fol-

lowed by INORA coarse-feedback scheme and then, followed by INSIGNIA and TORA running without feedback.

INORA schemes work better in scenario_B than in scenario_A. This shows that as the network gets more heavily loaded, and when the QoS flows have higher bandwidth requirements, having an interaction between the routing protocol and the QoS signaling system gives better performance. Also by using the INORA fine-feedback scheme in higher loaded scenarios, we have good effects of fine-tuned load balancing.

The additional overhead incurred in the INORA schemes over INSIGNIA and TORA running independently of each other for Scenario_A is as shown in fig. 3.26. The additional overhead incurred in INORA schemes over INSIGNIA and TORA running independently in Scenario_B is as shown in fig. 3.27. As expected, INORA fine-feedback scheme has larger messaging overhead when compared to the INORA coarse-feedback scheme in both Scenario_A and Scenario_B. It can be seen that the ratio of INORA overhead in coarse-feedback to fine-feedback increases from scenario_A and scenario_B. This shows that even the fine-feedback scheme performs better in heavily loaded networks than in lightly loaded networks in terms of overhead.

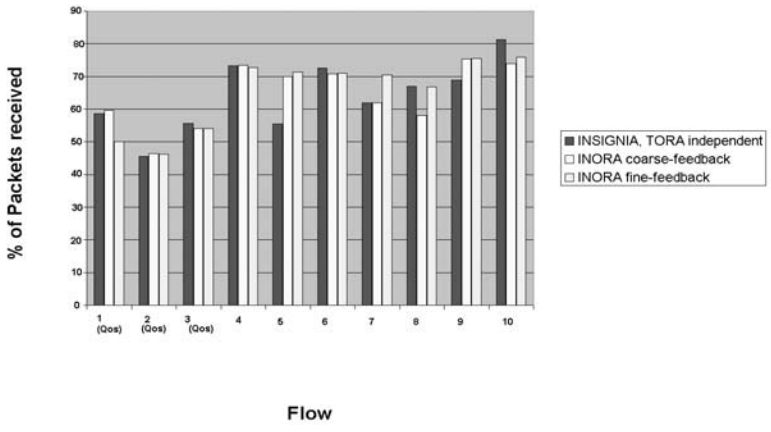


Figure 3.18: Percentage of Packets delivered(Scenario_A)

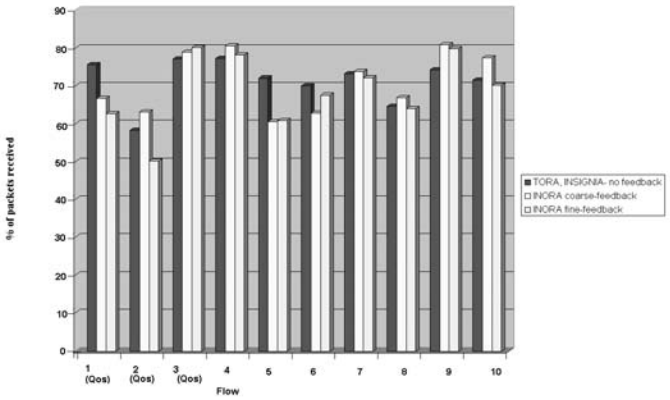


Figure 3.19: Percentage of packets delivered (Scenario_B)

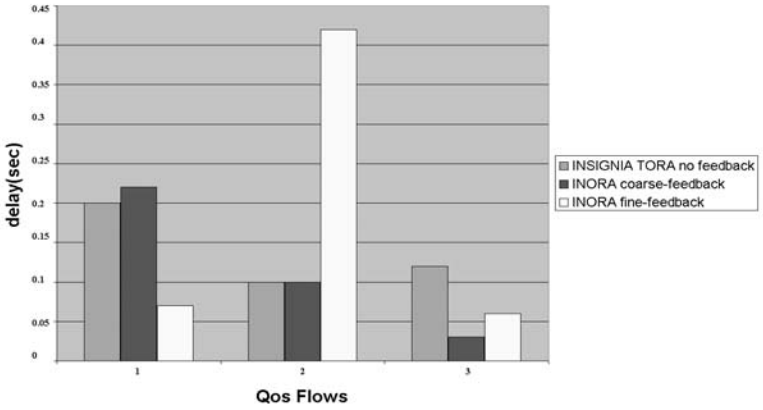


Figure 3.20: Average Delay of QoS packets (Scenario_A)

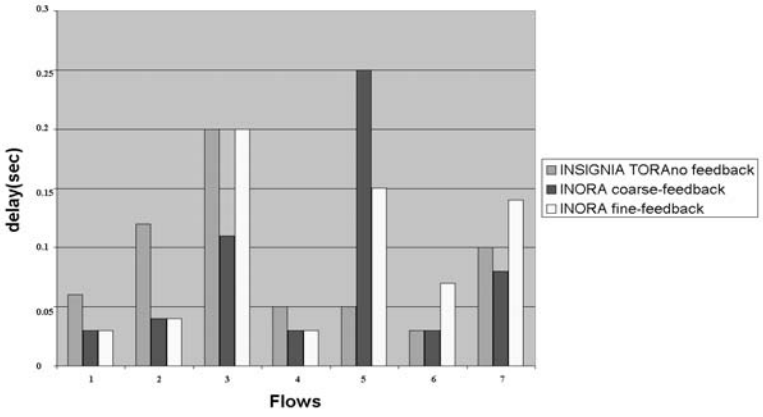


Figure 3.21: Average Delay of non-QoS packets(Scenario_A)

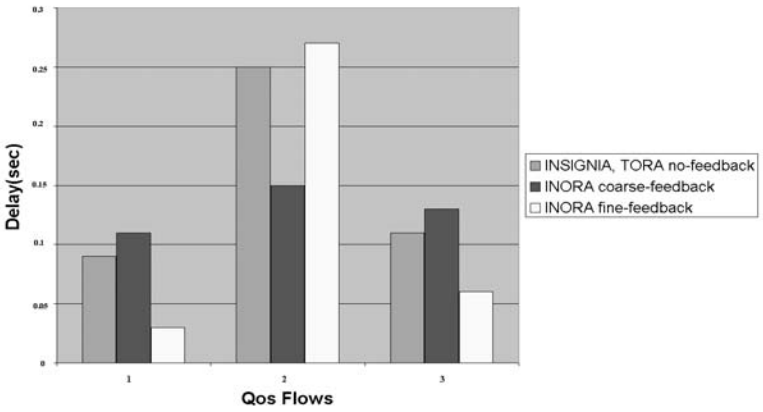


Figure 3.22: Average Delay of QoS packets(Scenario_B)

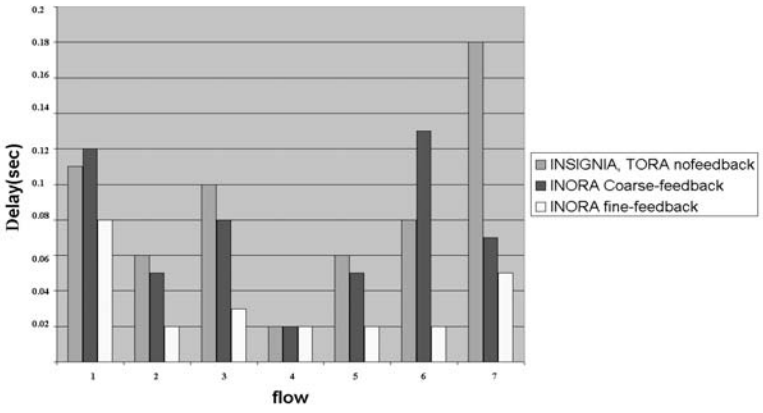


Figure 3.23: Average Delay of non-QoS packets(Scenario_B)

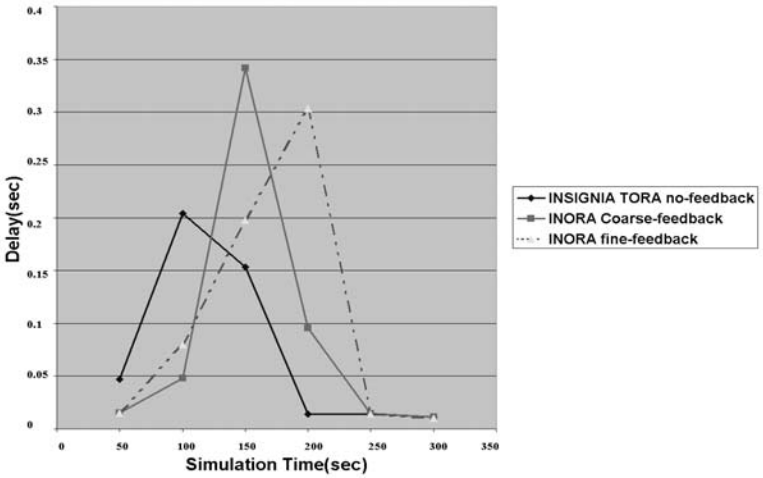


Figure 3.24: Average Delay of all the packets(Scenario_A)

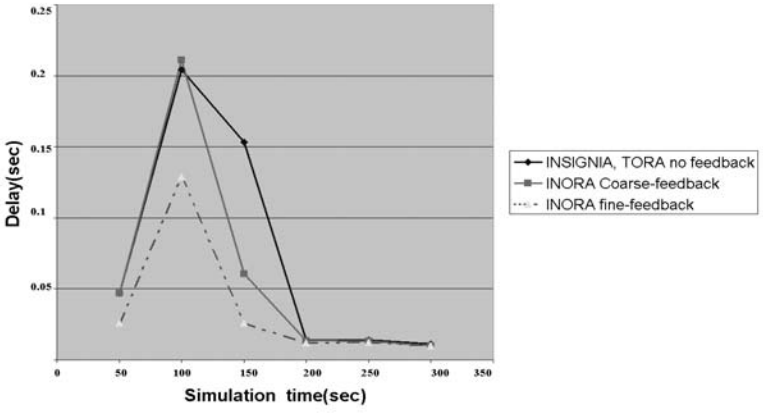


Figure 3.25: Average Delay of all the packets(Scenario_B)

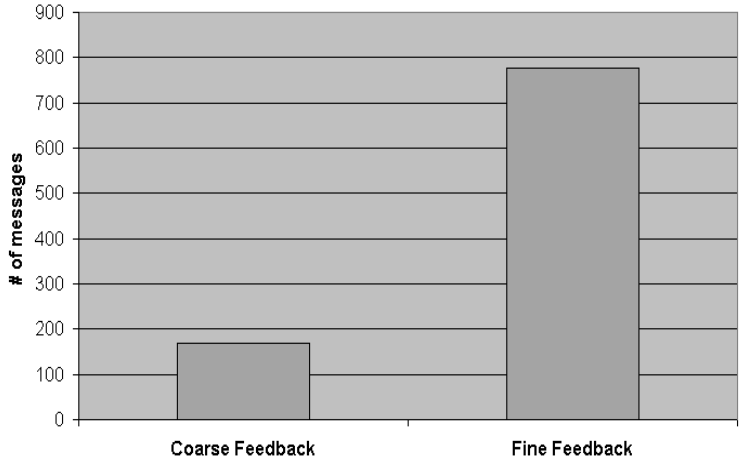


Figure 3.26: Overhead in INORA(Scenario_A)

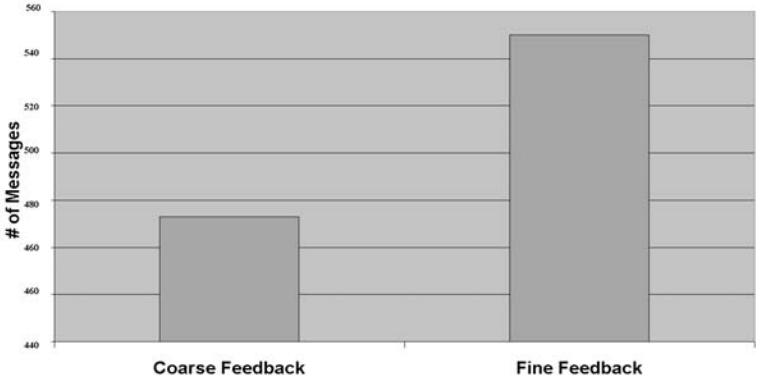


Figure 3.27: Overhead in INORA(Scenario_B)

Chapter 4

Enhancements to TORA

4.1 Introduction

In this chapter we describe a problem which causes routing instability in MANETs which use TORA. We describe the enhancements that were made to TORA to fix this problem. We also made changes to TORA to separate out the functions of *Route Creation* and *Route Maintenance*. We evaluate the enhanced TORA against the other standard MANET routing protocols. In the end, we describe the proactive operation of TORA. We evalute the proactively operated TORA against the non-proactive TORA.

4.2 Routing Instability Problem in TORA

4.2.1 Problem Description

Consider a heavily loaded and congested network as shown in fig. 4.1. It consists of two portions portion A and portion B connected by a single link with node P as the connecting node. Portion A is built off the reference $R_1 = (\tau_1, P, r_p)$. Destination is in the portion B. The DAG rooted at the destination is as shown in the fig. 4.1.

1. At time $\tau = \tau_2$, node Q loses its last downstream neighbor due to a link failure. (See fig. 4.2)

2. Node Q reverses its upstream links by generating a new reference $R_2 = (\tau_2, Q, 0)$ and sends a $UPD(UPD_1)$. UPD_1 hasn't been delivered to the neighbors yet. (Because of congestion in the network). (See fig. 4.3)

3. At time $\tau = \tau_3$, the link between node *P* and portion *B* fails. Partition of the network occurs.(see fig. 4.4)

4. Route Erasure mechanism triggers a *CLR* (CLR_1) for reference $R_1 = (\tau_1, P, r_P)$. All the nodes that have R_1 as their reference $NULL$ their *heights.* (See figure 4.5)

5. At time $\tau = \tau_4$, seeing that all its neighbors have $NULL$ heights, node *Q* $NULL$s its *Height* and being the generator for reference R_2, it sends out a $CLR(CLR_2)$ for the reference R_2 . (see figure 4.6)

6. At time $\tau = \tau_5$, UPD_1 gets delivered to the portion A.(IMEP, the underlying layer ensures reliable transmission of the routing packets). Now, routes are built off the new reference R_2.(See figure 4.7)

7. At time $\tau = \tau_6$, a *QRY* (QRY_1) arrives for the destination at the portion A. This is replied to by any node which has a non-NULL *Height* built off R_2. (See figure 4.8)

8. At time $\tau = \tau_7$, another node *R* loses its last downstream link and sends out an $UPD(UPD_2)$ after generating a new reference $R_3 = (\tau_7, Q, 0)$. This is not delivered yet.(See figures 4.9 and 4.10)

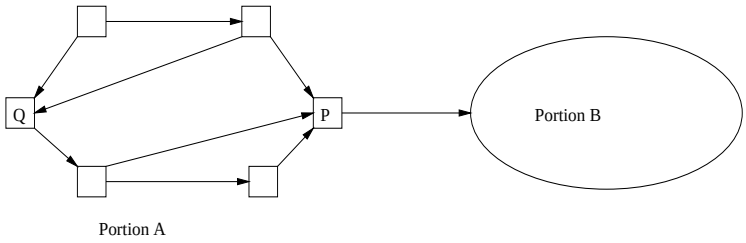
9. At time $\tau = \tau_8$, a $CLR(CLR_2)$ for reference R_2 is delivered. All the nodes that have their reference as R_2 NULL their *Height*s. (See figure 4.11)

10. At time $\tau = \tau_9$, Node *R* (which generated the *Height* reference R_3*) NULLs* its Height, because it is surrounded by NULL nodes.(See figure 4.12)

11. At time $\tau = \tau_{10}$, UPD_2 which was generated for R_3 is delivered. (See figure 4.13)

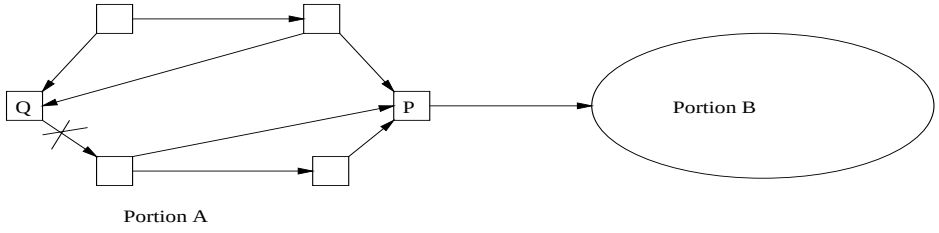
12. At time $\tau = \tau_{11}$, this is followed by another query for the same destination QRY_2, which is replied to by node *P* which has a non-NULL *Height* built off R_3. (See figure 4.14)

This leads to an infinite sequence of routing events ordered according to their delivery time as shown below:



Consider a heavily loaded, congested network. Portion A is built off reference $R_1 = (\tau_1, P, r_P)$. Destination is in portion B.

Figure 4.1: MANET Topology for Routing Instability Problem



At time $\tau = \tau_2$, node Q loses its last downstream neighbor due to a link failure.

Figure 4.2: MANET Topology for Routing Instability Problem

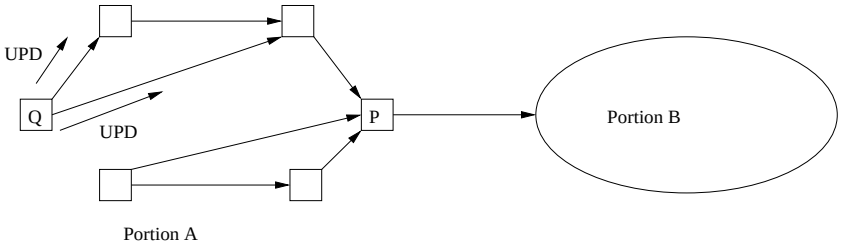
$CLR_1 \rightarrow UPD_1 \rightarrow QRY_1 \rightarrow CLR_2 \rightarrow UPD_2 \rightarrow QRY_2 \rightarrow$ .....

This problem manifests as large IMEP packets, This is because the underlying layer, IMEP aggregates TORA messages before sending them into the network. Because of this, the already heavily loaded network breaks down. This problem was first observed by Matt Impett during ns-2 simulations of TORA. The sequence of events is illustrated in figures 4.1 through 4.12 .

4.2.2 Solution to the Routing Instability Problem in TORA

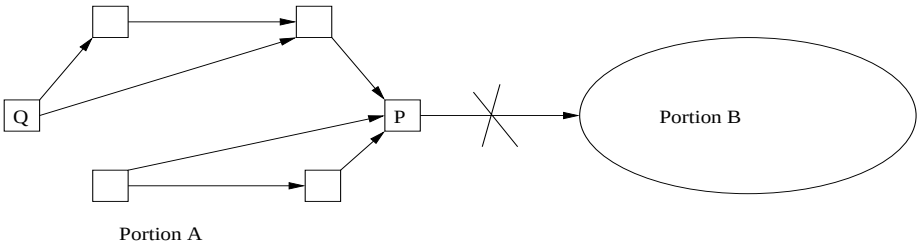
We propose the following modifications in TORA to tackle the above problem

At the time $\tau = \tau_5$(as described in section 4.2.1), when node *Q* finds all its



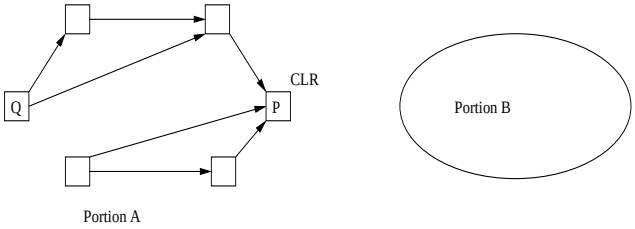
node Q reverses its upstream links by generating a new reference $R_2 = (\tau_2, Q, 0)$. Q sends a UPD(UPD1).

Figure 4.3: MANET Topology for Routing Instability Problem



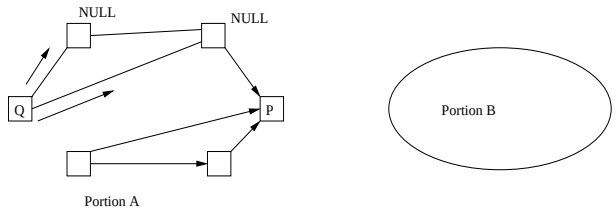
At time $\tau=\tau_3$, the link between P and portion B fails. Partition occurs in the network.

Figure 4.4: MANET Topology for Routing Instability Problem



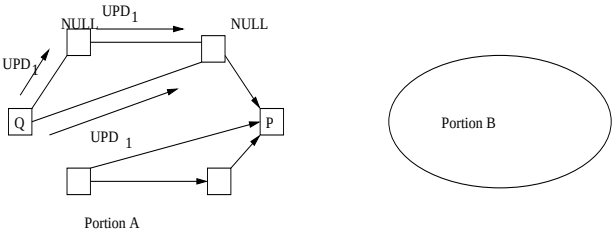
Route Erasure mechanism triggers a CLR(CLR$_1$) for the reference $R_1 = (\tau_1, P, r_p)$. All nodes that have R_1 as their reference NULL their Heights.

Figure 4.5: MANET Topology for Routing Instability Problem



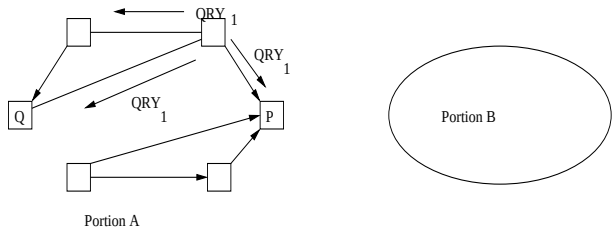
At time $\tau=\tau_4$, seeing that all its neighbors have NULL Heights, node 'Q' NULLs its Height. It sends out a CLR(CLR $_1$) for the reference R_2.
This is not delivered yet.

Figure 4.6: MANET Topology for Routing Instability Problem



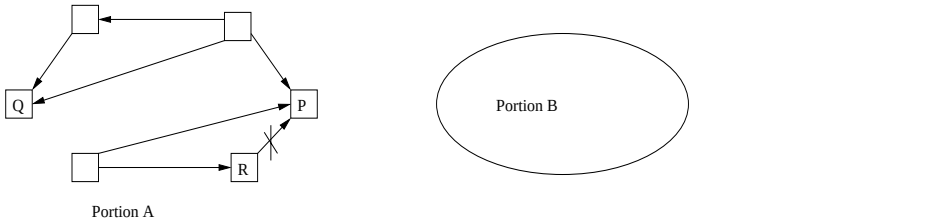
At time $\tau=\tau_5$, . UPD $_1$ gets delivered around portion A.

Figure 4.7: MANET Topology for Routing Instability Problem



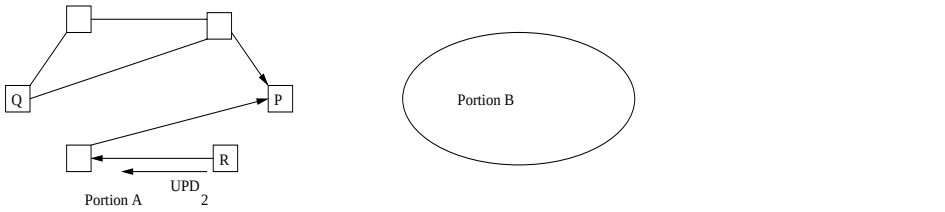
At time $\tau=\tau_6$, a QRY(QRY $_1$) arrives for the destination at Portion A.

Figure 4.8: MANET Topology for Routing Instability Problem



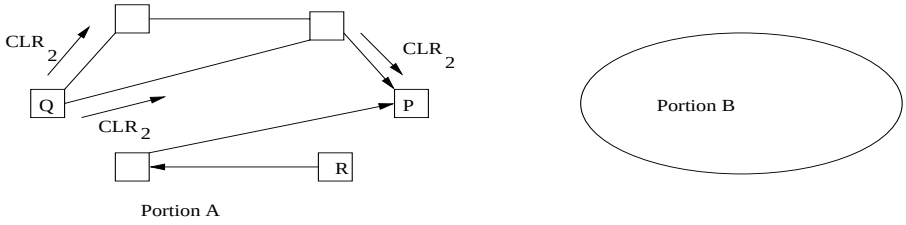
At time $\tau=\tau_7$, node R loses its last downstream link and sends out on UPD(UPD $_2$) after generating the new reference $R_3=(\tau_7, Q, 0)$

Figure 4.9: MANET Topology for Routing Instability Problem



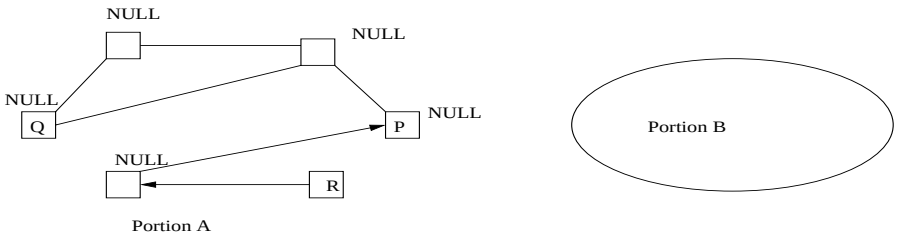
At time $\tau=\tau_7$ another node R loses its last downstream link and sends out on UPD(UPD $_2$) after generating a new reference $R_3=(\tau_7, Q, 0)$. . This is not delivered yet.

Figure 4.10: MANET Topology for Routing Instability Problem



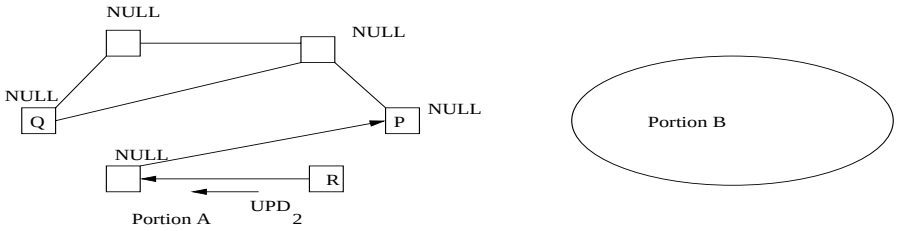
At time $\tau=\tau_8$, the CLRfor R_2(CLR $_2$) is delivered. All nodes with reference R_2 NULL their Heights.

Figure 4.11: MANET Topology for Routing Instability Problem



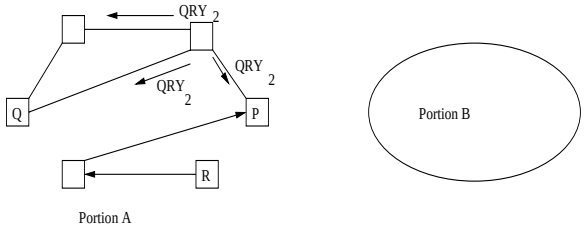
At time $\tau=\tau_9$, node R, which is surrounded by NULL nodes NULLs its Height. .

Figure 4.12: MANET Topology for Routing Instability Problem



At time $\tau=\tau_{10}$, UPD_2 which was generated for R_3 is delivered. .

Figure 4.13: MANET Topology for Routing Instability Problem



At time $\tau=\tau_{11}$, this is followd by another query for the same destination (QRY_2). This will be replied by node P, which has a non−NULL Height built off reference R_3

Figure 4.14: MANET Topology for Routing Instability Problem

neighbors with *NULL* heights, it doesn't set its *Height* to NULL. Similarly, node *R* NULLs its *Height* when it finds itself surrounded by NULL nodes. NULLing of the *Height* is permissible for a node, only on the explicit reception of a CLR for the reference of the node and a CLR will be generated only when a partition occurs.

Separation of Route-Building and Route Maintenance Procedures

We also incorporated the use of separate message types for the *Route-Building* and *Route-Maintenance* procedures in TORA(that have been described in section 1.1.3). An explicit *RPY* would be used for *Route-Building* as against an *UPD* packet as used in the original TORA. This helps the mobile nodes to know whether the Height change of their neighbors has been triggered as a response to an explicit *Query*, or has been due to a *Route Repair* mechanism. So, the node can choose to adapt/not adapt propagate/drop the routing message.

The flow charts describing these changes are illustrated in figures 4.15 through 4.22. These changes were implemented on the ns-2 simulator.

4.3 Performance Evaluation of the new TORA

To evaluate the performance of the new TORA, we tested it in a scenario which causes the routing instability problem described in section 4.2.1 to occur. We chose the ns-2 simulator for the simulations.

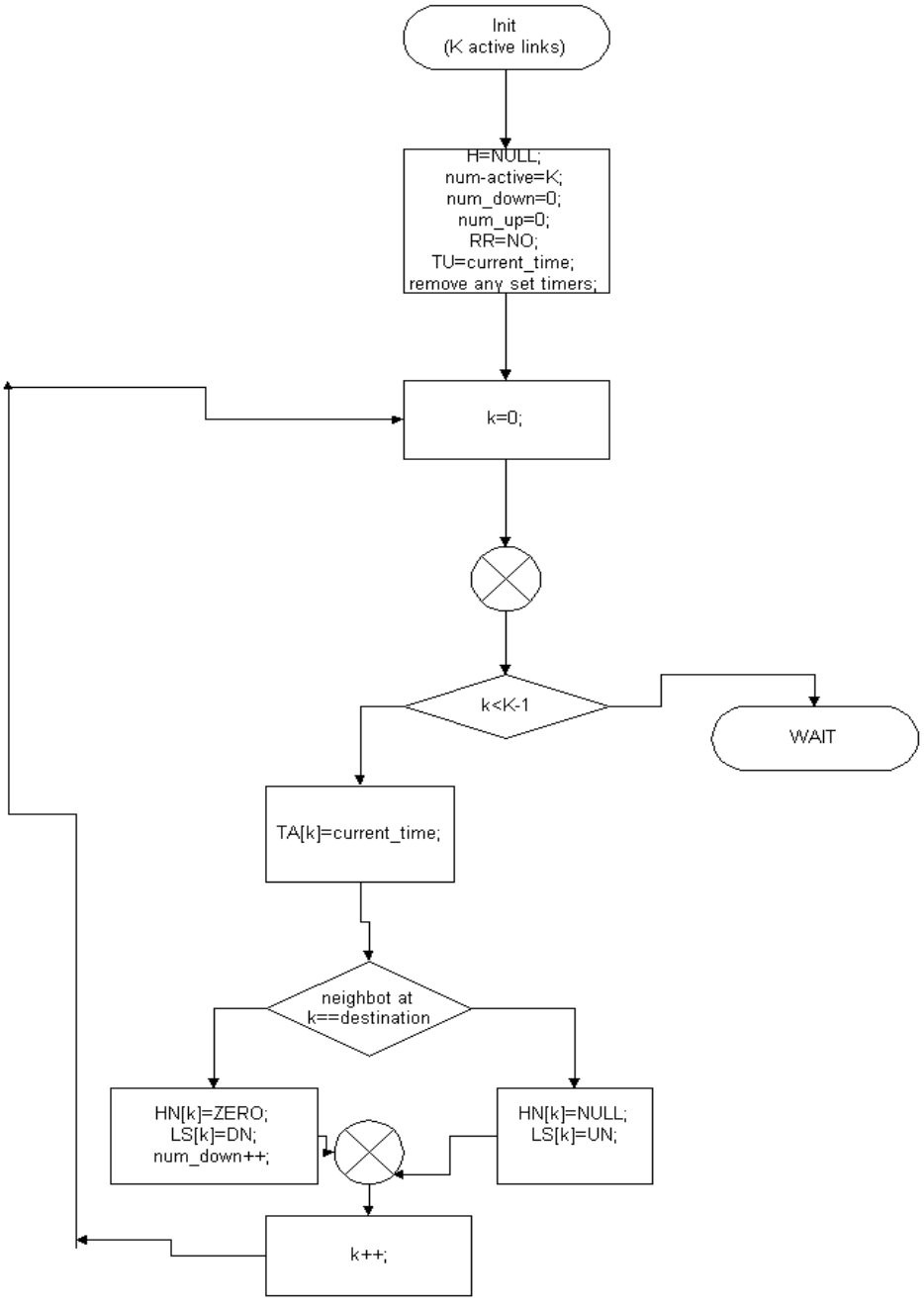


Figure 4.15: Procedure executed on initial boot-up

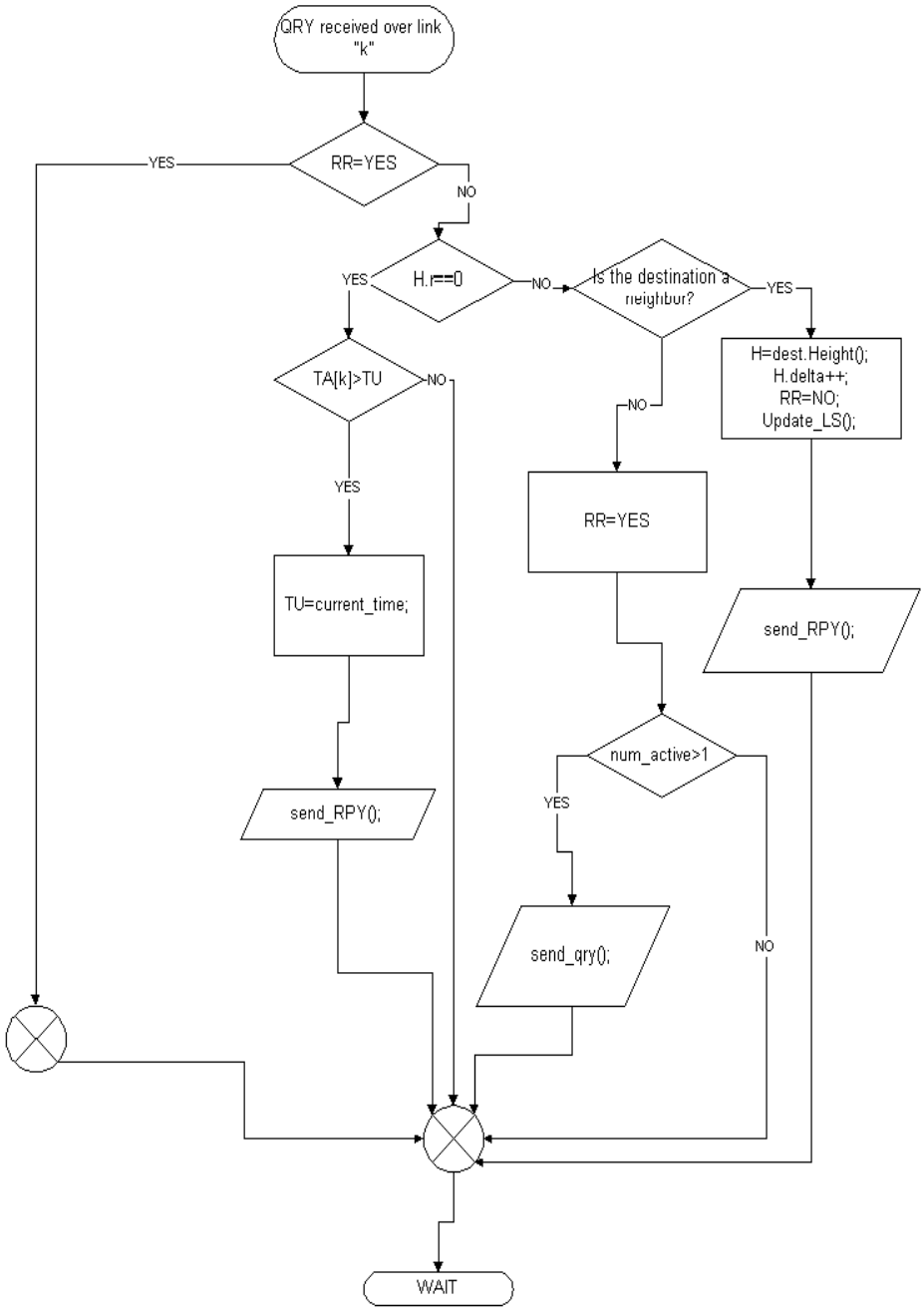


Figure 4.16: Procedure executed on the reception of a query

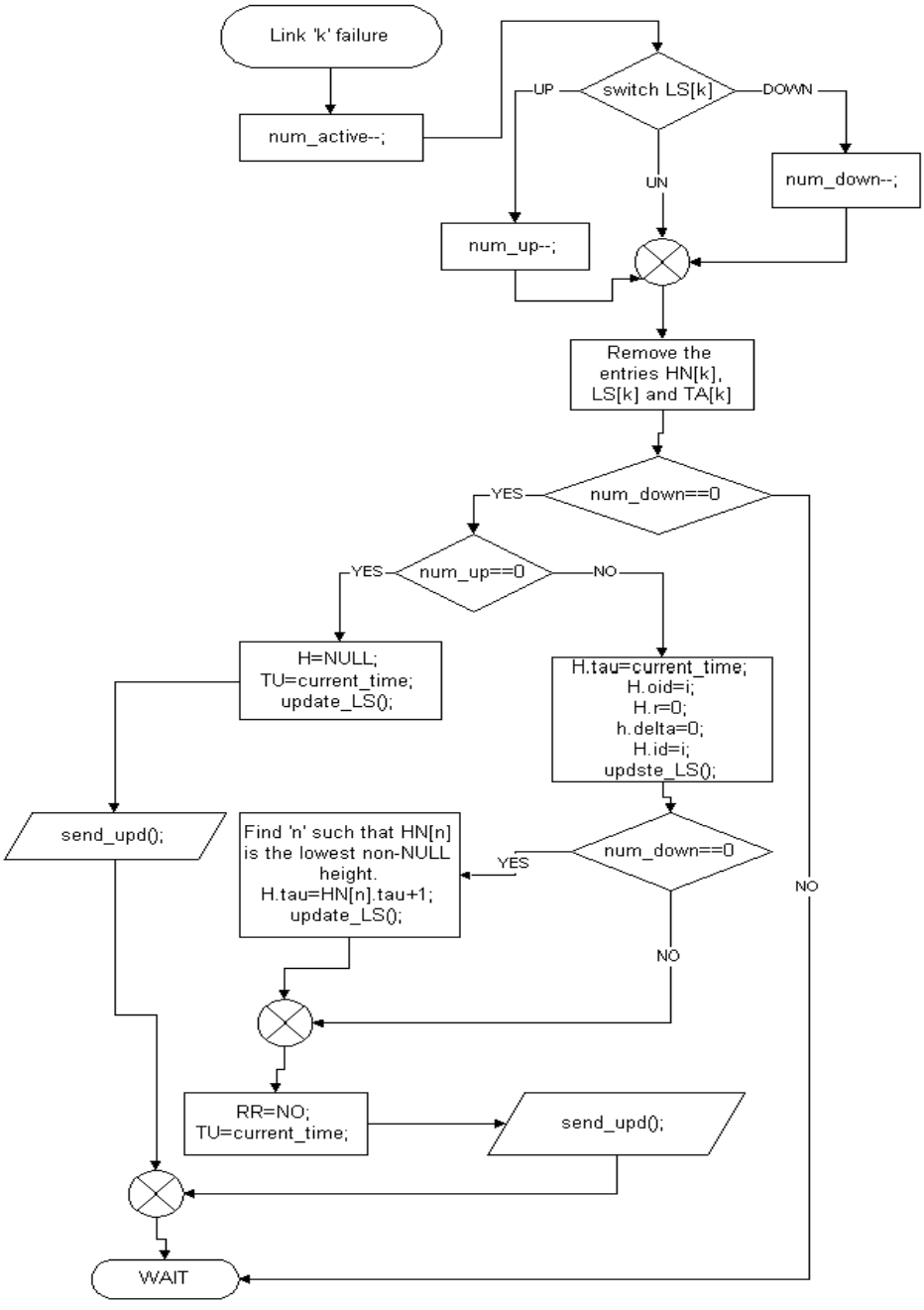


Figure 4.17: Procedure executed on a link failure

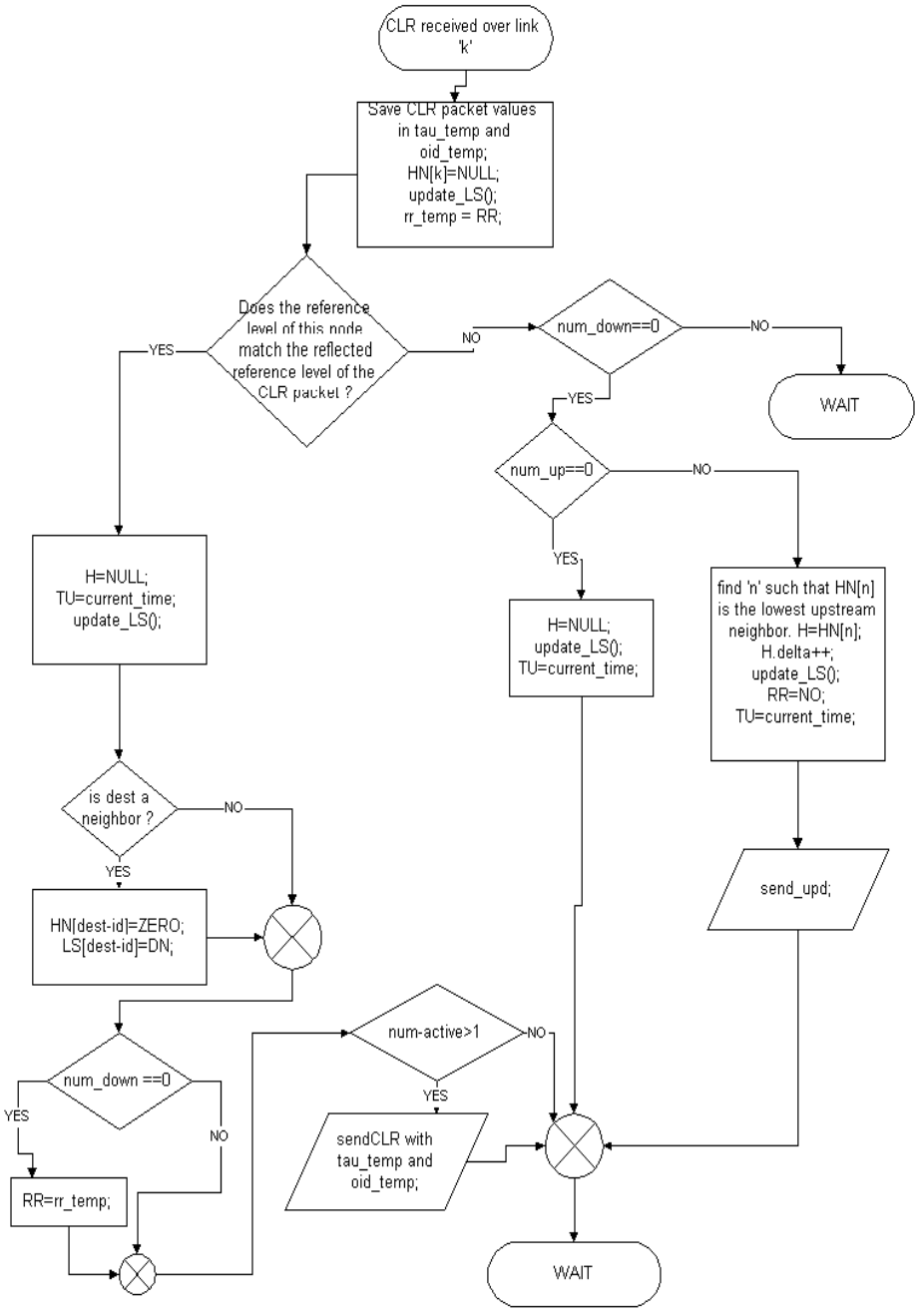


Figure 4.18: Procedure executed on the reception of *Clear*(CLR) messages

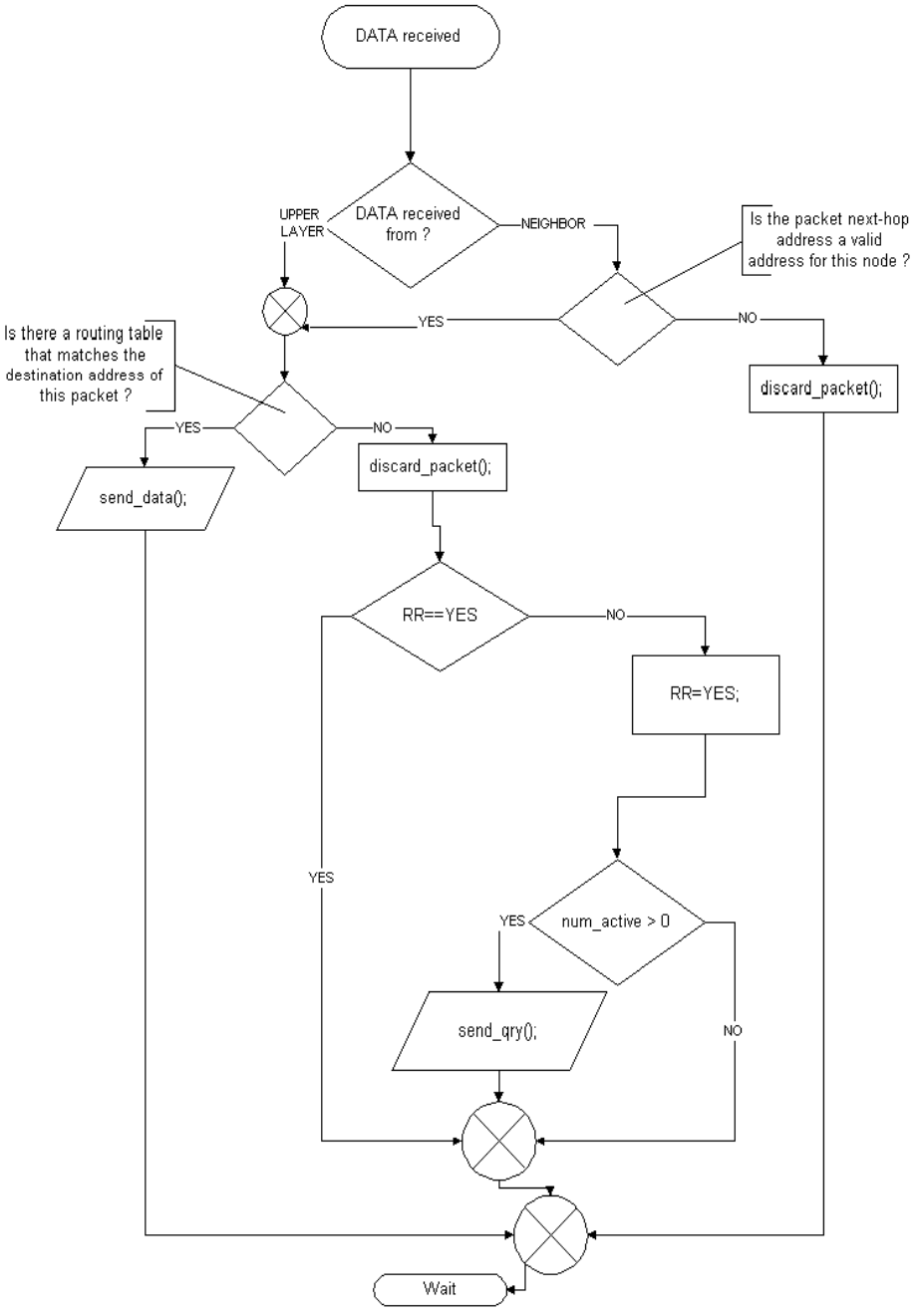


Figure 4.19: Procedure executed on the reception of a data packet

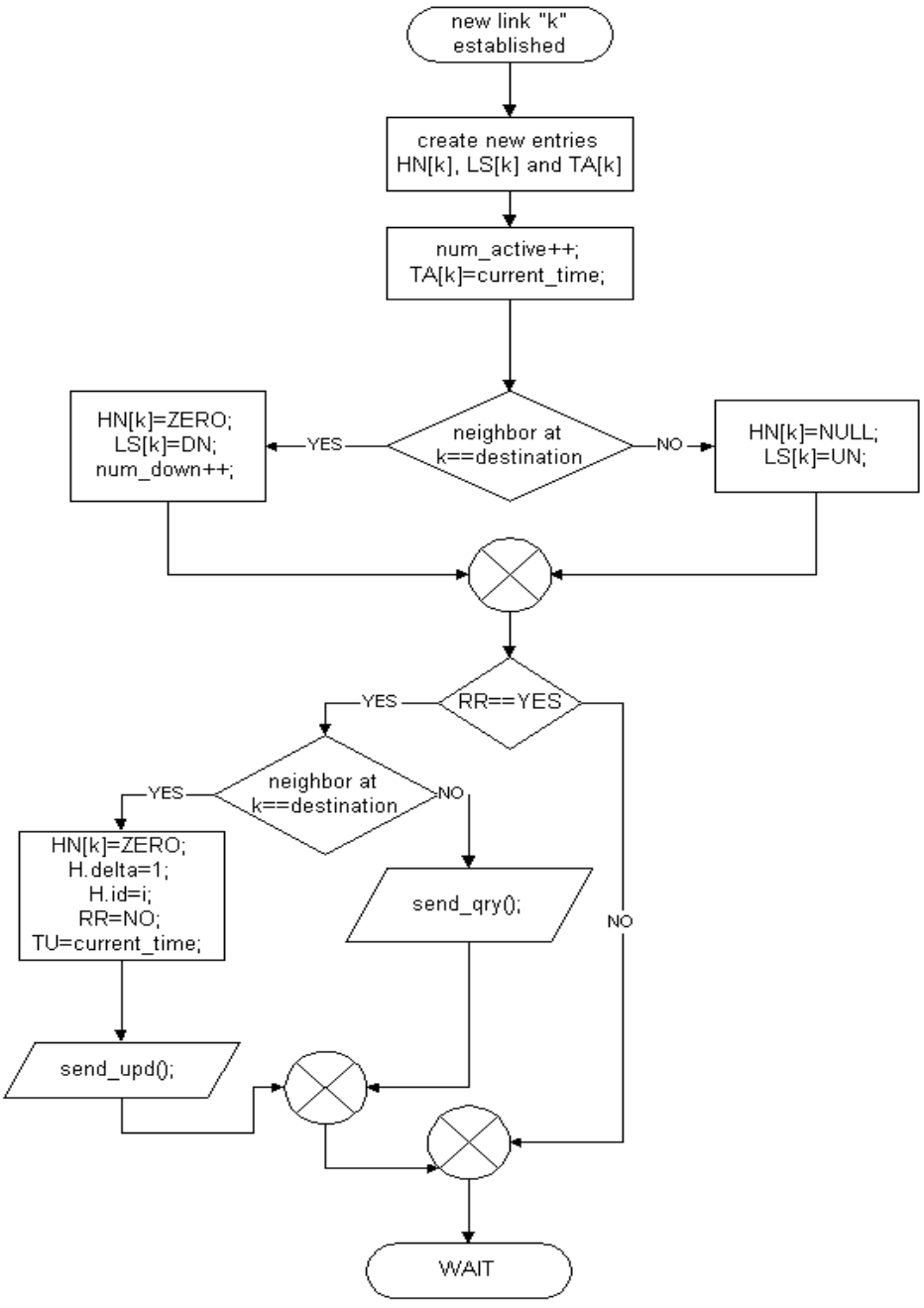


Figure 4.20: Procedure executed on a link coming up between two nodes

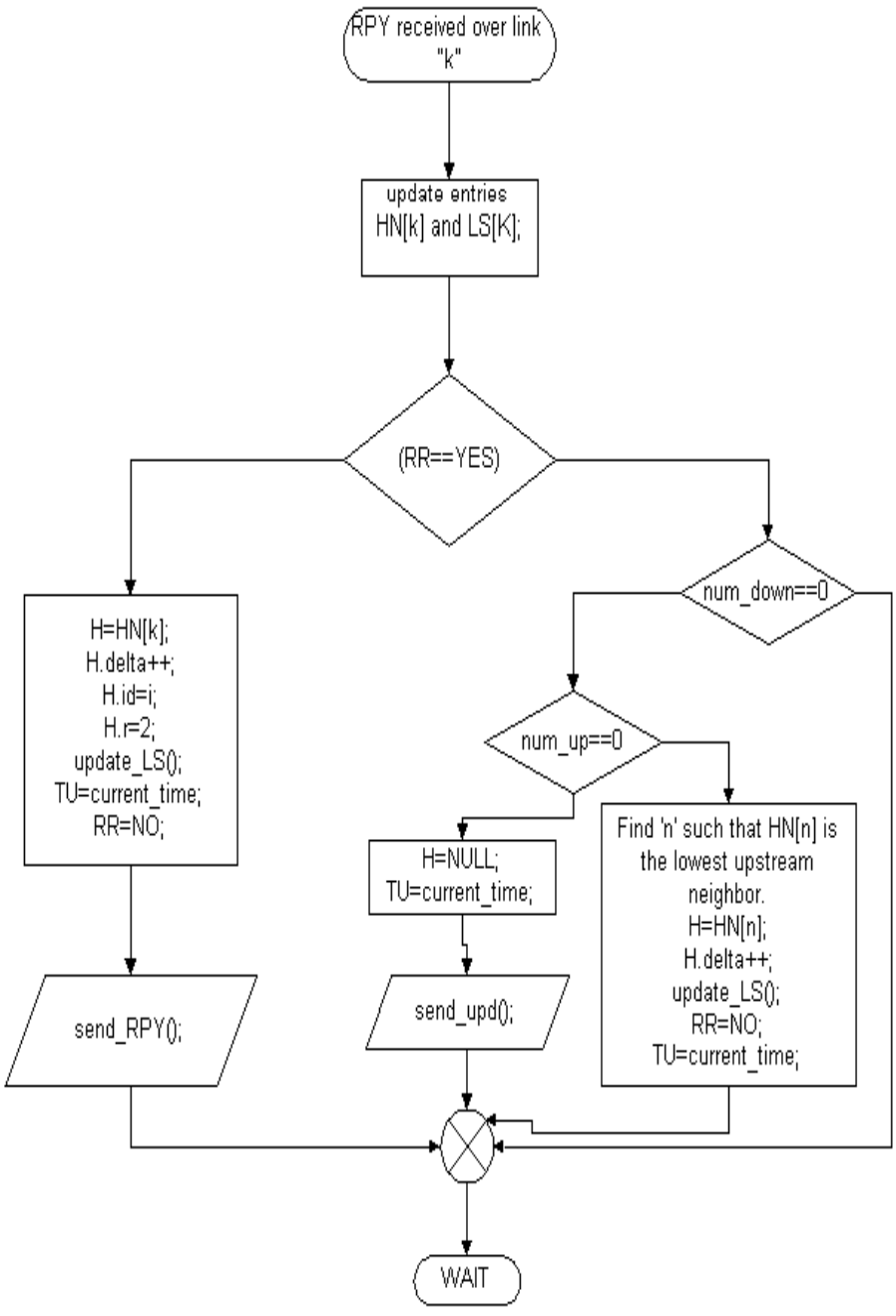


Figure 4.21: Procedure executed on the reception of a *Reply*(RPY) packet

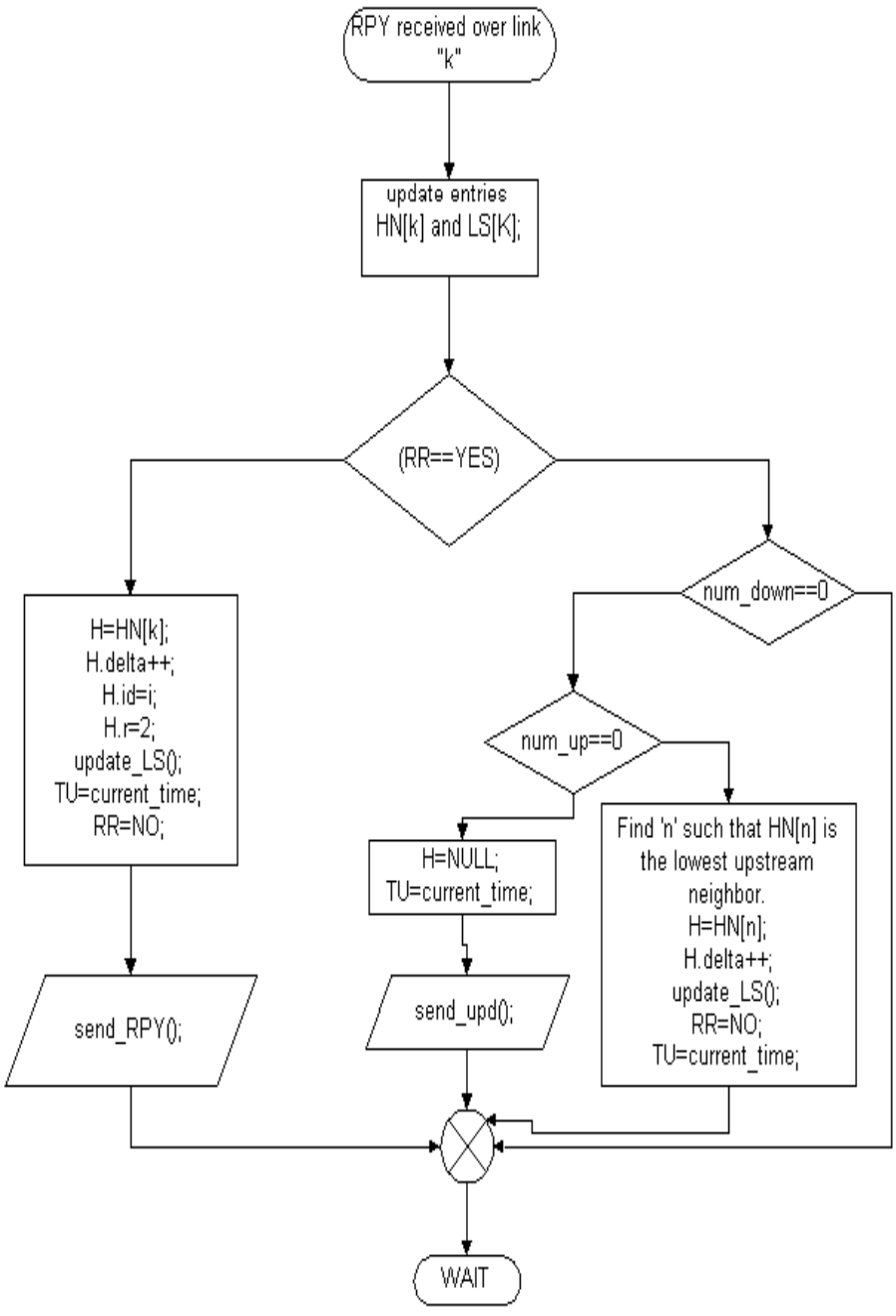


Figure 4.22: Procedure executed on the reception of an *Update*(UPD) packet

4.3.1 Description of the simulation scenario

The simulations were carried out in a 1500*m*x500*m* rectangular grid area. There are 50 nodes placed randomly in this area and move around with a maximum velocity of 40 m/s in a *Random way-point* model. The physical layer description and the MAC layer description are same as in chapters 2 and 3.

Communication Pattern

Out of the 50 nodes, 20 nodes engage in communication.

This communication pattern was provided by Majid-Raissi Dehkordi and Gun Akkor. Each of the 20 communicating nodes generate voice, data and content-delivery traffic. Such a communicating node switches between these services with a silence period between each change of service. The transition between the three services and the silence state for each node is defined by the continuous-time Markov chain as described in figure 4.23. The descriptions of the various traffic models are as follows:

1. **Voice:** The activity model for voice is an ON/OFF model with a constant transmission rate of 8kbps during the ON periods. The ON periods are exponentially distributed with a mean of 0.35 sec and the OFF periods have the same type of distribution, but with a mean of 0.65 sec. Thus, the effective bandwidth used by the voice source is: $8kbpsx(0.35/(0.35+0.65)) = 2.8kbps$

2. **Data:** The activity model for data in an ON/OFF model with a constant

transmission rate of 64kbps during the ON periods. The ON periods are distributed according to a Pareto distribution with the shape parameter 1.5 and average value 0.8 seconds. The OFF periods follow the same distribution, but with an average of 20 seconds.

3. **Content Delivery:** The activity model for content delivery is a constant bit rate connection with a high rate of 300kbps. The effective bandwidth for this type of source is therefore equal to this number because there are no OFF periods in this case.

The parameters of the Markov Chain are selected as follows:

$1/\lambda_s$= avg. time in "silence" state = 49.5 sec

$1/\lambda_v$= avg. time in "voice" state = 30 sec.

$1/\lambda_d$=avg. time in "data" state = 120 sec.

$1/\lambda_c$=avg. time in "content delivery" state = 13.33 sec

$p_v = 0.944$, $p_d = 0.037$, $p_c = 0.019$

These parameters result in the following stationary probability distributions:

$P_s = 0.6$, $P_v = 0.344$, $P_c = 0.003$

These parameters have been chosen such that the average load generated by a node at a particular time is around 40-50 kbps per each service. The traffic load offered by the CBR traffic is as shown in figure 4.24. The traffic load offered by the voice traffic is as shown in the figure 4.25. The traffic pattern of the content-delivery traffic is as shown in the figure

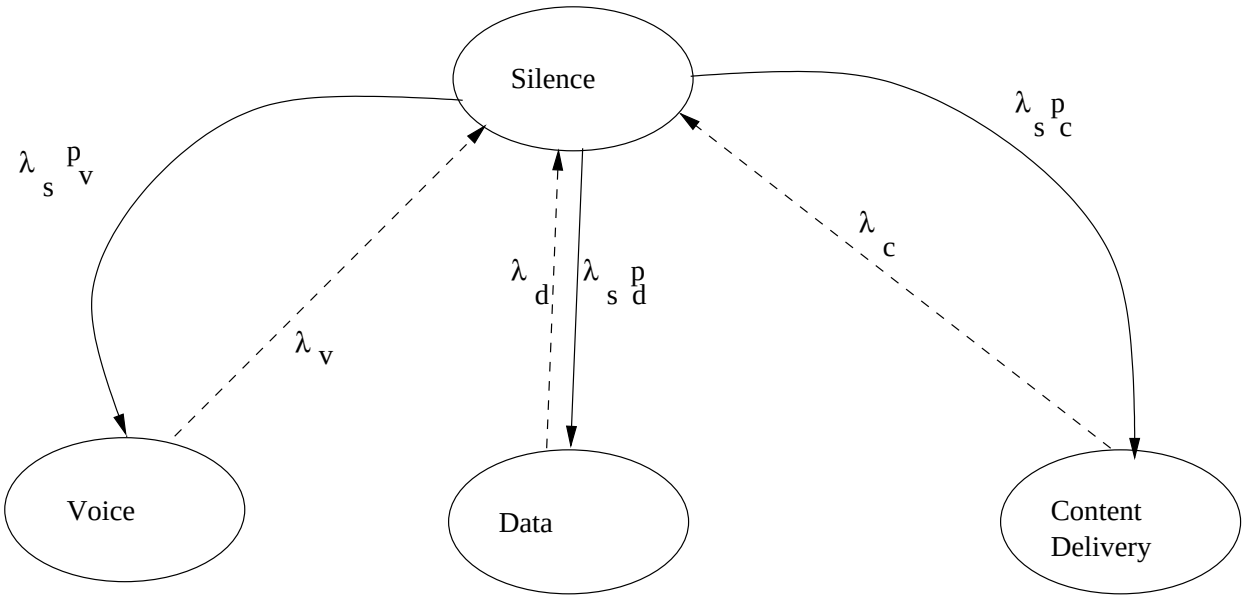


Figure 4.23: Markov Chain depicting the transition between the different traffic types

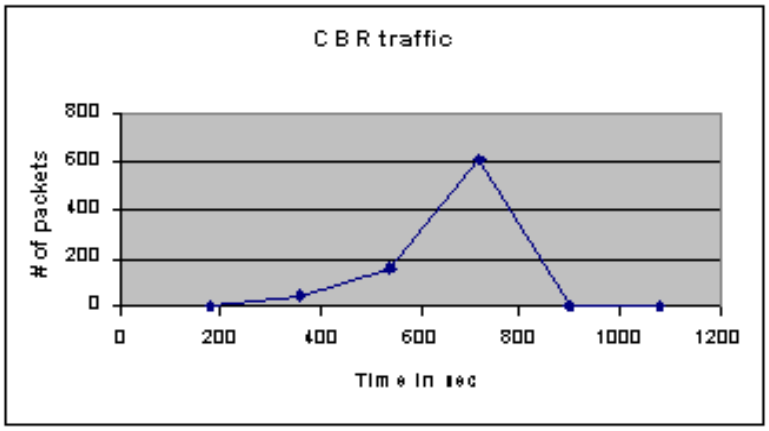


Figure 4.24: Traffic load offered by the Content delivery traffic

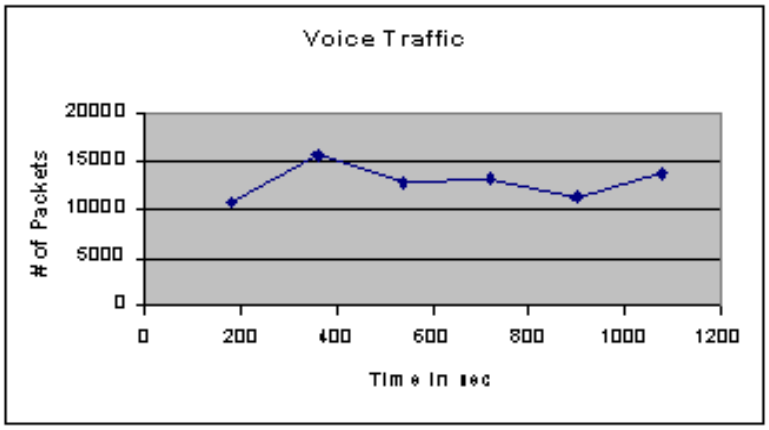


Figure 4.25: Traffic load offered by the Voice traffic

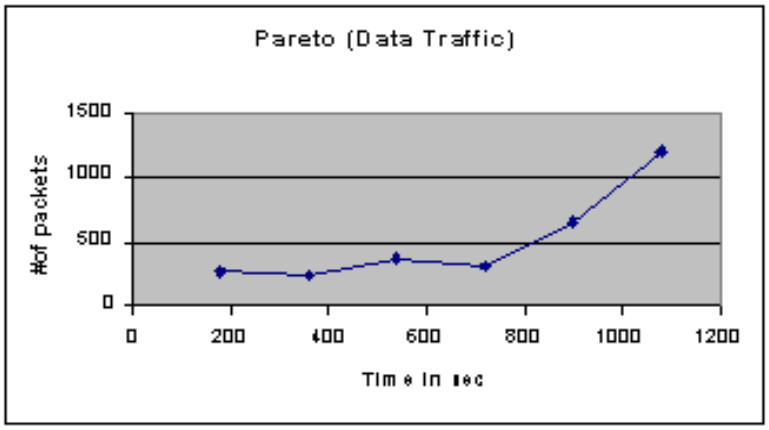


Figure 4.26: Traffic load offered by the data traffic

4.3.2 Performance of the enhanced TORA

We compared the performance of new TORA against other routing protocols for adhoc networks (AODV and DSR).

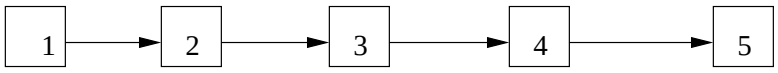
The following metrics were used for comparison:

1. **Average end-to-end delay experienced by packets belonging to different kinds of traffic**

2. **Total Goodput**: Goodput of the traffic is defined as the total number of data packets received by the destination nodes(x) divided by the total number of packets received(data packets(y)+routing packets(z)). $Goodput = x/(y+z)$ This metric penalizes long routes. It also penalizes the protocol with larger control packet transmissions. In the figure 4.27 that during the calculation of goodput, the packet reception is counted 5 times in the denominator and once in the numerator of the goodput expression.

It can be seen from figure 4.28 that the goodput of the new version of TORA is comparable to AODV and DSR. The goodput of TORA is lesser than AODV and DSR. This is because TORA has a greater overhead than AODV and DSR. This is because TORA creates a DAG structure and provides multiple routes. As we saw in chapter 3, this can be exploited to provide QoS facilities.

Figure 4.29 shows that the new TORA gives reasonable end-to-end delay characteristics for voice packets. It can be seen that DSR and AODV have a high delay during the initial portion of the simulation, while TORA maintains the same range of end-to-end delay throughout.



Node 1 is the source of a connection and node 5 is the destination for the connection.

Figure 4.27: Calculation of Goodput

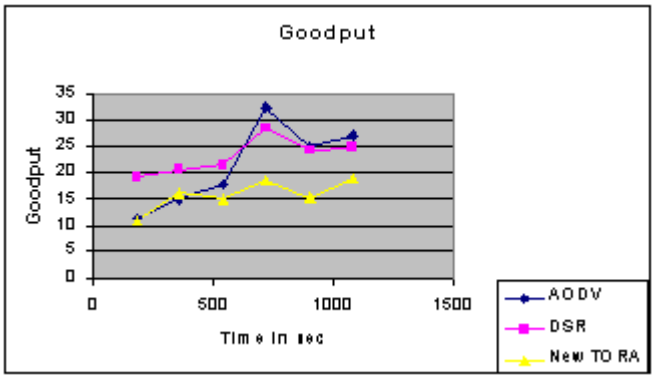


Figure 4.28: Goodput

Figure 4.30 shows that new TORA performs comparable to AODV and DSR in the delay for data packets. Again, AODV performs better than TORA which performs better than DSR.

Figure 4.31 shows the delay experienced by content delivery packets in AODV, DSR and TORA. It can be seen that AODV performs better than TORA which performs better than DSR.

4.4 Proactive Operation of TORA

As described in the section 1.1.3, TORA is an on-demand (reactive) routing protocol where, routes are built on-demand. But, TORA can also be operated in

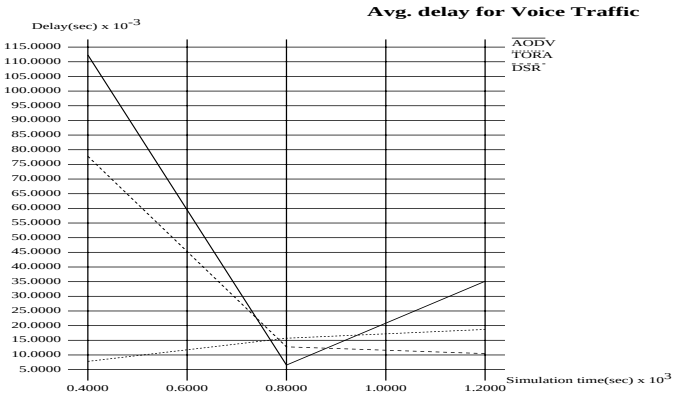


Figure 4.29: Delay Experienced by Voice Packets

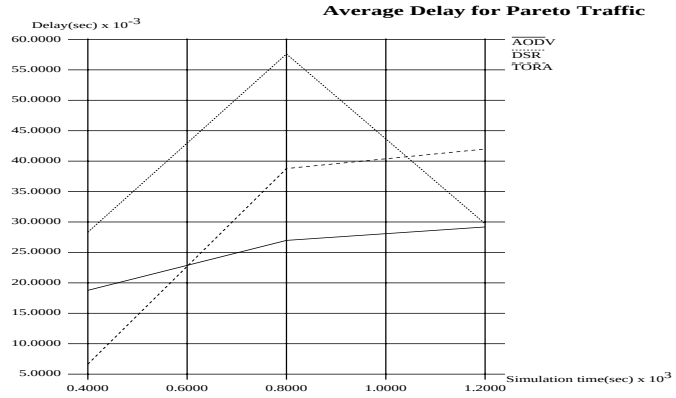


Figure 4.30: Delay Experienced by Data Packets

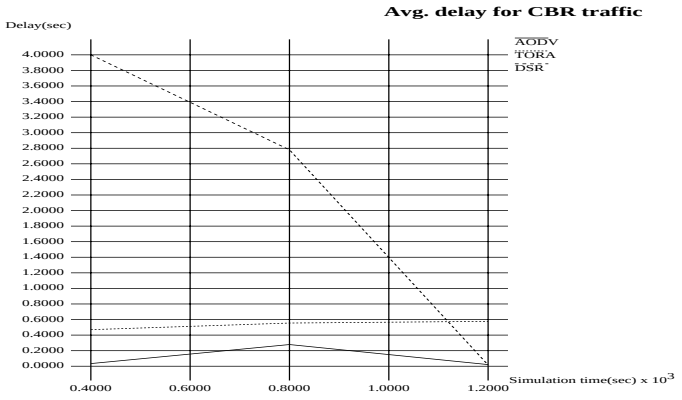


Figure 4.31: Delay Experienced by Content Delivery Packets

proactive mode[4]. Some selective nodes can proactively initiate the construction of DAGs for them as the destinations. This operation is possible by the introduction of an additional packet type called *Optimization* (OPT) packets. OPT packets are send out periodically by selective nodes. The OPT packets have the following fields:

- **Destination IP address:** This field denotes the node which originated the OPT packet.
- **Height:** This field denotes the *Height* metric of the neighboring node from which the OPT packet was received.
- **Optimization Sequence Number:** This field denotes the sequence number of the OPT packet.

- **Optimization Interval:** This field denotes the time interval between the production of two OPT packets.

On the reception of an OPT packet, a node compares the *Optimization sequence number* of the OPT packet which it has received to that of the OPT packet that it has previously propagated. If the *Optimization sequence number* of the packet received is greater than that of the OPT packet that it had propagated previously, it sets its *Height* with the same reference as the Height of the OPT packet it has received. It then increments the δ field of the *Height.* The unique id field of the Height is set to the id of itself. The node then propagates the OPT packet by setting the Height field in the OPT packet to its own *Height* by broadcasting to its neighbors.

It should be noted that since the selective destination nodes are initiating the build-up of the DAG rooted at themselves, the reference part of the Height propagated by the DAG will always be *zero-reference* $(0, 0, 0)$. The proactive building of the DAG will typically be initiated by the nodes that act as gateways to the external networks (or the Internet) from the MANET. These will be nodes that will be very frequently accessed. So, it makes sense to have a DAG proactively built to these nodes, and not wait until an explicit connection establishment trigger the route building process to these nodes. The procedure executed on the reception of an OPT packet is as shown in the figure 4.32.

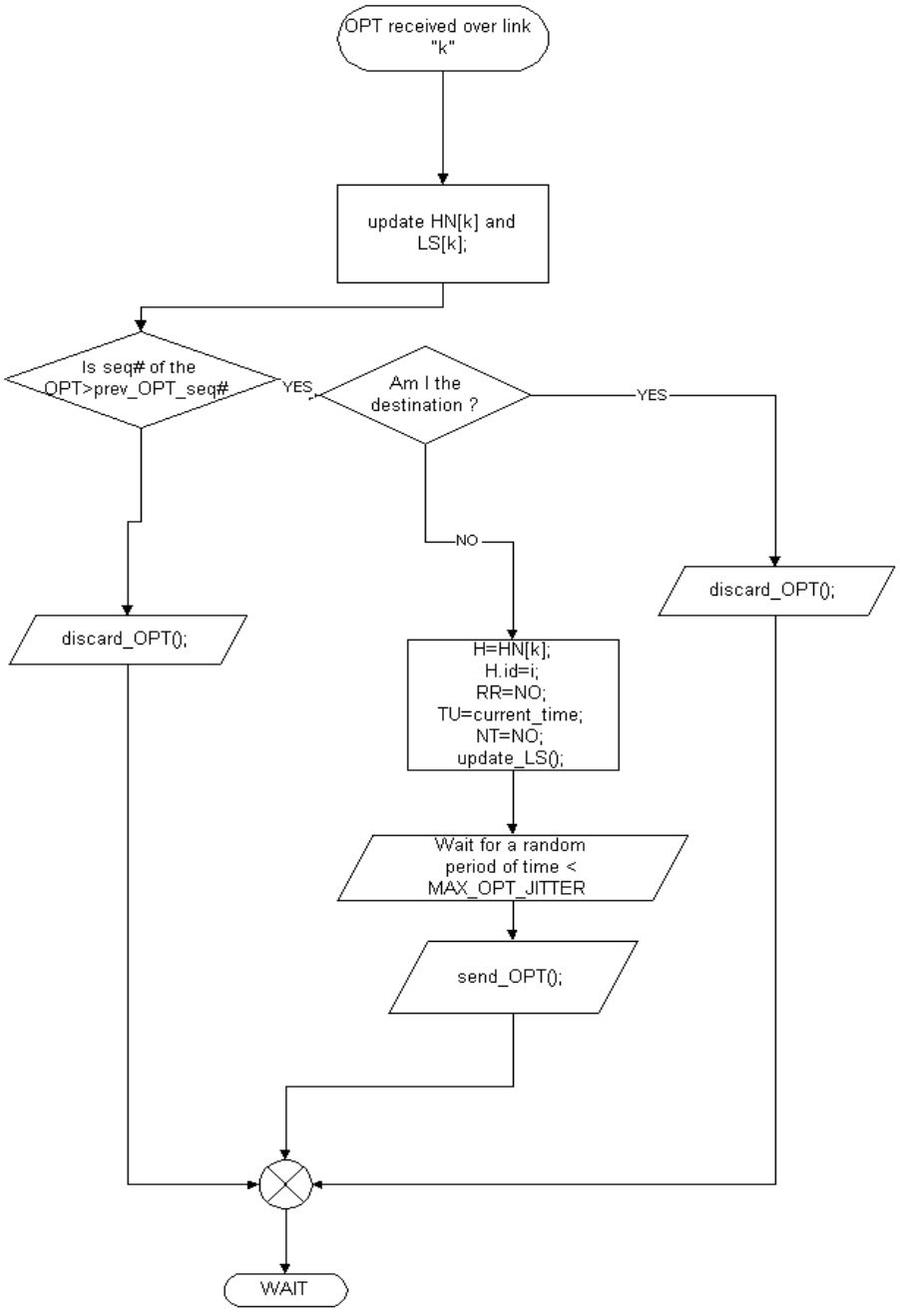


Figure 4.32: Procedure executed on the reception of an OPT packet.

4.4.1 Performance Evaluation of Proactively Operated TORA.

The performance of proactively operated TORA was tested against the non-proactively operated TORA by simulations in the ns-2 simulator. The simulation scenario was a 1500mx300m rectangular two dimensional grid. 50 nodes were randomly placed in this area. The mobility pattern of these nodes was random way-point model as described in section 2.3.1. The maximum speed of a mobile node is 20m/s. The wireless propagation model was chosen to be similar to that mentioned in the section 2.3. The transmission range of a single wireless node is 250 m as in the previous simulations. A single node was chosen as the gateway node which initiates proactive construction of the DAG to itself. We performed several experiments by varying the number of connections. The gateway node was always chosen as the destination. The source nodes generate CBR traffic with packet size 64 bytes and inter-packet interval is 0.1 sec. This corresponds to a bandwidth of $5.12 kbps$. We choose routing-overhead as a measure for performance comparison of the two versions of TORA. In the proactively operated TORA, the *Optimization Interval* (the interval between the generation of two consecutive OPT packets) was chosen to be 30 sec.

From fig. 4.33, we can see that as the number of connections increases, the routing overhead in both versions of the protocol decreases. This is due to the phenomenon of many connections using the single DAG mentioned in the section 2.3.2. This phenomenon is more evident in this simulation because all the connections use a single node (the gateway node) as the destination, and hence use the same DAG. We see that in the case of lower number of connections, the routing

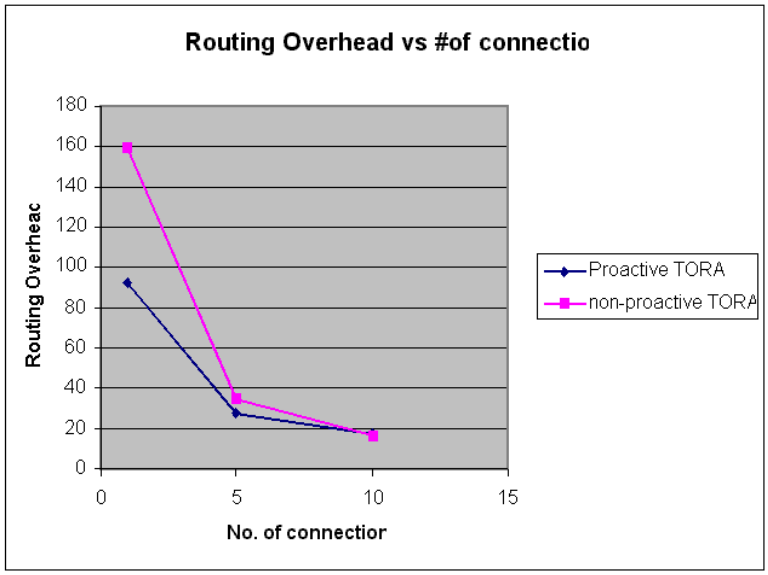


Figure 4.33: Routing Overhead in Proactive and non-proactive TORA

overhead is considerably lower in the proactive-TORA when compared to the non-proactively operated TORA. This is because in the proactively operated TORA, the destination proactively builds a DAG for itself and maintains it proactively (by the usage of OPT packets). This precludes the necessity for the route-building overhead(QRY and RPY packets) . This also reduces the route-maintenance overhead (UPD packets) considerably. We also find that as the number of connections increase, the routing overhead in the proactively operated TORA becomes greater than the non-proactively operated TORA. This is because, in non-proactively operated TORA, as the number of connections increase, the extent of the DAG increases, and more source nodes lie on the DAG. So, they don't have to initiate the route-building mechanism. Also, the reactive nature of TORA makes sure that UPD packet are generated only when a link failure causes a node to lose its last

downstream link. In the proactive version of TORA, the OPT packets are launched at regular intervals, irrespective of whether or not a topology change occurs.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In Chapter 1, we have provided an introduction to adhoc networking in general and mobile adhoc networks(MANETs) in particular. We also provided a description of Temporally Ordered Routing Algorithm(TORA) which is a highly distributed routing protocol for MANETs.

In Chapter 2, we presented the *Query-Localization* techniques for TORA. We have shown by simulations that the routing overhead is reduced by about 50% by using the Query-Localization techniques. We have also shown that the choice of the Query-Timer plays an important role in the performance of *Query-Localized* TORA.

In Chapter 3, INORA, a QoS support mechanism using INSIGNIA in-band QoS signaling system and TORA routing protocol for adhoc networks has been

proposed. The implementation and an evaluation of INORA has also been presented. We have shown by simulations that INORA schemes do well compared to when INSIGNIA and TORA have no feedback. In particular, INORA schemes perform very well in networks that are heavily-loaded and where the QoS flows have higher bandwidth requirements.

In Chapter 4, we presented the enhancements to TORA incorporated for solving a specific routing instability problem and also for providing a separation of *Route-Creation* and *Route-Maintenance* functions by introducing an additional control packet(RPY). We have shown by simulations that the new version of TORA compares well against other standard adhoc routing protocols (AODV and DSR). Also, we presented the proactive operation of TORA. We have shown that the gateway nodes in a MANET network can proactively build a DAG for themselves and cause a considerable reduction in routing overhead.

5.2 Future Work

In wireless networks, congestion at a wireless node is related to congestion in its one-hop neighborhood. i.e. wireless networks are best treated as a union of neighborhoods defined by the transmission radius of a node, rather than a graph consisting of nodes joined by point-to-point links. So, the congestion at a node is intrinsically related to congestion in its neighborhood. A suitable mechanism needs to be incorporated in INORA to reflect this fact, so that congested neighborhoods can be avoided by QoS flows.

A good method of calculating the bandwidth available to a wireless node also needs to be found, rather than the current method of finding it in an indirect fashion by the node listening promiscuously listening to the packet transmissions in its neighborhood.

In wireless networks, the different layers of the protocol stack cannot function in isolation with each other. This is because the higher layer functions directly depend on the lower layers. So, there needs to be an explicit coupling between the different layers to achieve efficient performance in wireless networks.

Bibliography

[1] S-B. Lee, G-S. Ahn, X. Zhang, A. T. Campbell,"INSIGNIA: An IP-Based Quality of Service Framework for Mobile ad Hoc Networks,"*Journal of Parallel and Distributed Computing*, Vol 60 No 4, April 2000

[2] S-B. Lee, A.T. Campbell,"INSIGNIA:In-band signaling support for QoS in mobile ad hoc networks", *Proc of 5th International Workshop on Mobile Multimedia Communications (MoMuC, 98)*, Berlin, Oct. 1998

[3] G-S. Ahn, A.T. Campbell, S-B. Lee, X.Zhang, "INSIGNIA,"*Internet Draft*, draft-ietf-manet-insignia-01.txt, Oct 1999

[4] V. Park, S. Corson, "Temporally Ordered Routing Algorithm(TORA) version 1 functional specification, draft-ietf-manet-tora-spec-04.txt,"July 2001

[5] V. Park, "A highly adaptive distributed routing algorithm for mobile wireless networks" Masters Thesis, Dept. of Electrical and Computer Engineering, University of Maryland 1997.

[6] V. Park, S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks", *IEEE Infocomm 1997*.

[7] MANET Working Group , URL: IETF.http://www.ietf.org/html.charters/manet-charter.html

[8] T.S. Rappaport, Wireless Communications: Principles and Practice, *Prentice Hall*, p.69-122, p.136-196, 1996

[9] P. Sinha, R. Sivakumar, V. Bhargavan, "CEDAR: a Core-Extraction Distributed Ad hoc Routing Algorithm,"*IEEE Infocom '99*, New York, NY.

[10] V. Bhargavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Media Access Protocol for Wireless LANs," *Proc. of ACM SIGCOMM '94*, pp.212-225, 1994.

[11] V. Bhargavan, "Performance of multiple access protocols in wireless packet networks," *International Performance and Dependability Symposium*, Durham, North Carolina, sept 1998

[12] C.R. Lin and M. Gerla, "MACA/PR: An Asynchronous Multimedia Multihop Wireless Network," *Proceedings of IEEE INFOCOM '97*, 1997.

[13] S. Chen, and K. Nahrstedt, "Distributed Quality-of-Service Routing in AdHoc Networks," *IEEE Journal on Special Areas in Communications*, Vol. 17, No. 8, August 1999.

[14] K. Wu, J. Harms, "QoS Support in Mobile Ad Hoc Networks," *Crossing Boundaries- the GSA Journal of University of Alberta*, Vol. 1, No, 1, Nov. 2001, pp.92-106

[15] INSIGNIA *ns-2* source code from Comet Group, Columbia University URL:http://comet.ctr.columbia.edu/ns_source_code.html

[16] CMU Monarch Project Extensions to ns-2

[17] J. Broch, D. Maltz, D. Johnson, Y. Hu, “A Performance Comparison of Multi-Hop Wireless AdHoc Routing Protocols”

[18] TORA *ns-2* source code from CSHCN, University of Maryland, College Park. URL:http://www.cshcn.umd.edu/tora.shtml

[19] Monarch *ns-2* wireless extensions from Monarch group, Carnegie Mellon University.URL:http://www.monarch.cs.cmu.edu/cmu-ns.html