

**ROUTING MISBEHAVIOR AND PERFORMANCE
ANALYSIS OF TEMPORALLY ORDERED
ROUTING ALGORITHM (TORA) IN MOBILE
ADHOC NETWORKS**

*Dissertation submitted to Jawaharlal Nehru University, in partial fulfillment of the
requirements for award of the degree of*

Master of Technology

in

Computer Science and Technology

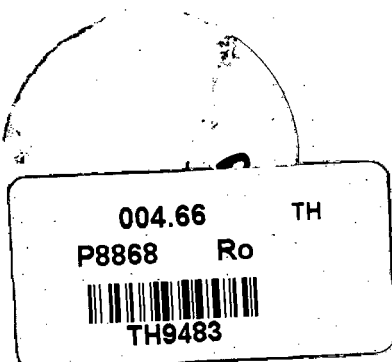
by

B. PRASANT KUMAR



**SCHOOL OF COMPUTER & SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI – 110067**

January 2002



1940

LIBRARY'S COPY

**ROUTING MISBEHAVIOR AND PERFORMANCE
ANALYSIS OF TEMPORALLY ORDERED
ROUTING ALGORITHM (TORA) IN MOBILE
ADHOC NETWORKS**

*Dissertation submitted to Jawaharlal Nehru University, in partial fulfillment of the
requirements for award of the degree of*

Master of Technology

in

Computer Science and Technology

by

B. PRASANT KUMAR



**SCHOOL OF COMPUTER & SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI – 110067**

January 2002

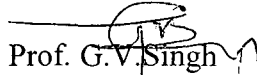


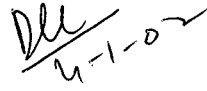
जवाहरलाल नेहरू विश्वविद्यालय
JAWAHARLAL NEHRU UNIVERSITY
School of Computer & Systems Sciences
NEW DELHI - 110 067, INDIA


CERTIFICATE

This is to certify that the project entitled “ **Routing Misbehavior and Performance Analysis of Temporally Ordered Routing Algorithm (TORA) in Mobile Ad-hoc Networks** ” being submitted by **B. Prasant Kumar** to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science & Technology, is a bonafide work carried out by him under the guidance and supervision of Prof. G.V.Singh and Dr. D.K.Lobiyal.

The matter embodied in the dissertation has not been submitted for the award of any other degree or diploma.


Prof. G.V. Singh
Professor, SC & SS,
Jawaharlal Nehru University
New Delhi – 67


Dr. D.K. Lobiyal
Asst. Professor, SC & SS
Jawaharlal Nehru University
New Delhi - 67


Prof. K.K. Bharadwaj
Dean, SC & SS,
Jawaharlal Nehru University,
New Delhi-110067.

**...dedicated to
my beloved parents**

ACKNOWLEDGEMENTS

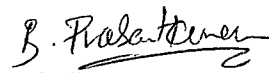
I would like to pay obeisance at the feet of my parents for their blessings that are always with me in all my aspirations including my academics.

I would like to sincerely thank my supervisors Prof. G.V. Singh, Dr D.K. Lobiyal, School of Computer And System Sciences, Jawaharlal Nehru University for the help, encouragement and support extended by them in successful completion of this project.

I would like to record my sincere thanks to the dean, Prof. K.K. Bharadwaj for providing the necessary lab facilities.

I will acknowledge the efforts and knowledge put by the development team of VINT project team at UC Berkeley, for developing the network simulator and making its source code available on the Internet as a freeware.

I take this opportunity to thank all the faculty members and friends for their help and encouragement during the course of the project.


B. Prasant Kumar

ABSTRACT

In this dissertation work the simulation study of routing misbehavior of Temporally Ordered Routing Algorithm (TORA) has been done by using the Network Simulator, NS2 developed at UC Berkeley. The network simulator NS2 provides the simulation environment for the ad-hoc networks.

The routing misbehavior is induced to the nodes in the ad-hoc network, by using the loss model in the network simulator. The following context is used in the simulation process: network size, mobility rate and traffic pattern. The routing misbehavior of TORA has been analyzed for the different combinations of the above mentioned parameters. The following parameters are evaluated to analyze The following context is used in the simulation process: network size, mobility rate and traffic pattern. The routing misbehavior of TORA has been analyzed for the different combinations of the above mentioned parameters.

Different types of the scenario files are generated to study the routing misbehavior by using the scenario file generator in the network simulator. A Tcl script that specifying the network configuration, different mobility and traffic scenarios is given as the input to the network simulator. After the simulation process the resulted data has been extracted and the routing misbehavior of the TORA is analyzed. Based on the results of the simulation, graphs are drawn for different scenarios.

CONTENTS

1. Introduction	1
1.1 Background	1
1.2 Need for Ad-hoc Networks	2
1.3 Mobile Ad-Hoc Network (MANET)	2
1.3.1 Characteristics of Mobile Ad-hoc Networks	3
1.3.2 Applications	4
1.4 Routing in Ad-hoc Networks	4
1.4.1 Desirable Properties of Ad-hoc Routing protocol	5
1.4.2 Destination Sequenced Distance Vector (DSDV)	6
1.4.3 Ad-hoc on Demand Distance Vector Routing(AODV)	7
1.4.4 Dynamic Source Routing (DSR)	8
1.4.5 Temporally Ordered Routing Algorithm (TORA)	8
1.5 Problem Definition	9
2. Routing Misbehavior in MANET	11
2.1 Watchdog	13
2.2 Pathrater	14
3. Temporally Ordered Routing Algorithm (TORA)	16
3.1 Overview of the Protocol	16
3.2 Creating routes	18
3.3 Maintaining routes	19
3.4 Erasing routes	22
3.5 Configuration	23

4. Simulation Environment and Methodology	26
4.1 Network Simulator	26
4.2 Mobile Node	29
4.3 Methodology	31
4.4 Simulation Overview	34
5. Simulation Study	35
5.1 Overview	35
5.2 Scenario Files Generation	38
5.3 Simulation Results	39
5.3.1 End to End Delay	39
5.3.2 Delivery Ratio	40
5.3.3 Overhead Ratio	42
5.3.4 Bandwidth Utilization	44
Conclusion	46
References	47
Appendix	49

Chapter 1

INTRODUCTION

1.1 Background

Mobile communication and wireless networking are becoming more and more popular now a days. There are two variations of mobile wireless networks. The first type is known as infrastructured network, with fixed and wired gateways. The bridges for these networks are known as base stations. A mobile unit within these networks connects to, and communicates with, the nearest base station that is within their communication range. As the mobile unit travels out of range of one base station and into the range of another, a “handoff” occurs from the old base station to the new, and the mobile host is able to continue communication seamlessly throughout the network. Applications of this type of network include office wireless local area networks (WLANs). The second type of mobile wireless network is the infrastructureless mobile network, known as an ad hoc network. Infrastructureless networks have no fixed routers, all nodes are capable to move and can be connected dynamically in an arbitrary manner. Nodes of these networks function as routers, which discover and maintain routes to other nodes in the network. Example applications of ad hoc networks are emergency search-and rescue operations, meetings or conventions in which persons wish to quickly share information.

As the cost of Wireless access drops, wireless communications could replace wired in many settings. The advantage of wireless is the ability to transmit data among the users in a common area while remaining mobile. Mobile nodes such as notebook computers, featuring powerful CPU’s large main memories, hundreds of megabytes of disk space are now easily affordable and are becoming quite common in everyday business and personal life. AT the same time, network connectivity options for use with mobile hosts have increased dramatically, including support for a growing number of wireless networking products based on radio and infrared.

1.2 Need for ad-hoc networks

Under some circumstances the mobility support is not possible. For example the access points are not set up due to low cost effect, poor performance or low usage. This may occur in some situations, such as in outdoor conferences, the mobile users will meet under circumstances that are not explicitly planned for and in which no connection to a standard wide area network such as the Internet is available. Emergency situations of natural disasters and military deploys in battlefield. So there is a need for ad-hoc network. With out the access points the mobile nodes under this environment must form an ad-hoc networks. These kind of networks of mobile nodes have become known as ad hoc networks. Since there is no fixed infrastructure, the wireless ad hoc networks can be deployed quickly. Such networks can be used in situations where either there is no other wireless communication infrastructure present or where such infrastructure cannot be used because of security, cost or safety reasons.

1.3 Mobile ad-hoc network (MANET)

A *Mobile Ad-hoc network* is a collection of mobile node dynamically forming a temporary network with out using any existing network infrastructure and centralized administration.

The areas in which there is little or no communication infrastructure or the existing infrastructure is expensive or in convenient to use, wireless mobile users may still be able to communicate through the formation of an ad-hoc network. The idea of ad-hoc networking is sometimes also called infrastructure less networking, since the mobile nodes in the network dynamically establish routing among themselves to form their own network.

Each node in the mobile ad-hoc network have a wireless interface and communicate with each other over either radio or infrared. The nodes are often mobile, but can be used to deploy relay points in areas where relay points might be

for other nodes. The nodes in ad-hoc network can act as both router and hosts, thus a node may forward packets between other nodes as well as run user applications.

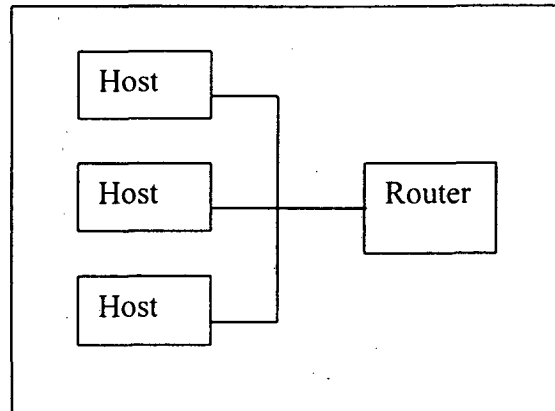


Figure 2: Block diagram of a mobile node acting as host and router.

1.3.1 Characteristics of mobile ad-hoc network

- There is no centralized administration or standard support services. Routing is distributed among the nodes, which can act as both hosts as well as routers.
- Possess dynamic topology due to the fact that nodes are free to move arbitrarily.
- Uses multi-hop routing in which nodes that are not in each other's transmission range can communicate through the intermediate nodes.
- Each node has a limited CPU capacity, storage capacity, battery power and bandwidth. This means power usage must be limited thus leading to a limited transmission range.
- These networks may consist of both unidirectional and bi-directional links.
- These networks are more prone to physical security threats than are fixed cable networks.

1.3.2 Applications

- Personal area networking - cell phone, laptop, earphone, wristwatch.
- Military environments - soldiers, tanks, planes.
- Civilian environments - taxicab network, meeting rooms, sports stadiums, Boats, small aircraft.
- Emergency Operations - search-and rescue, policing and fire fighting.

1.4 Routing in Ad-Hoc Networks

Routing algorithms for wired networks are designed to operate in a largely unchanging topology and over high bandwidth links. They are, therefore, unsuitable for ad hoc networks because of the node mobility, which may lead to link failure and link repair. The rate of link failure and link repair may be high when nodes move fast. Hence specialized routing strategies are required to maintain route stability despite mobility. An additional limitation in ad hoc networks is limited battery power, which needs to be conserved by allowing sleep modes, and minimizing broadcasts.

The simplest approach to routing in a dynamic topology would be to flood the network with the packet to be sent, with the hope that it would eventually reach the destination. However, this is extremely inefficient. Routing protocols for ad hoc networks assume a rate of topology change not high enough to make flooding the only alternative and not low enough to make conventional routing protocols effective.

A routing protocol for ad hoc network must be distributed, since in view of the dynamic topology no centralized point of control is possible. It should generate routes quickly so that they can be used before the topology changes. Also, it must be bandwidth efficient and have minimal control overhead. There are two basic choices for routing in ad hoc networks Reactive and Proactive.

Proactive or table-driven protocols attempt to continuously maintain a consistent view of network topology, for which they rely on route table maintenance and periodic and/or event triggered updates. The advantage here is route is available whenever required. However, these protocols continuously waste network capacity to keep routing information current, even though most of this information becomes stale even before it is used, due to node mobility. Another disadvantage is that in these protocols, local topological changes affect the entire network.

Reactive or on-demand schemes, on the other hand, evaluate routes only when they are needed. On-demand protocols typically have three components.

1. Route discovery : done on demand using a global search mechanism.
2. Route maintenance : done only while the route is being used.
3. Route erasure : non-active routes are to be removed.

These protocols reduce redundant routing information in the network, do not waste capacity on updates, and allow nodes to save power by going into sleep or standby modes.

Some *hybrid protocols* attempt to get the best of both worlds by limiting proactive routing to the local neighborhood of a node and using efficient reactive routing beyond it.

1.4.1 Desirable properties of ad hoc routing protocols

Distributed operation :

The protocol should be distributed, It should not be dependent on a centralized controlling node.

Loop free :

The protocol should guarantee that the routers supplied are loop-free.

Demand based operation :

The protocol should be reactive, to minimize the control overhead in the network and not to waste the network resources more than necessary.

Unidirectional link support :

The radio environment can cause the formation of unidirectional links. Utilization of these links and not only the bi-directional links improves the routing protocol performance.

Security :

The radio environment is especially vulnerable to impersonation attacks, so to ensure the wanted behavior from the routing protocol, we need some sort of preventive security measures. Authentication and encryption is probably the way to go and the problem here lies within distributing keys among the nodes in the ad hoc network.

Power conservation :

The nodes in an ad hoc network are very limited in battery power and therefore use some sort of stand-by mode to save power. Therefore it is important that the routing protocol has support for these sleep-modes.

Multiple routes :

To reduce the number of reactions to topological changes and congestion multiple routes could be used. If one route has become invalid, it is possible that another stored route could still be valid and thus saving the routing protocol from initiating another route discovery procedure.

There are different routing protocols that are based on the above mentioned protocols.

1.4.2 Destination Sequenced Distance Vector Routing (DSDV)

DSDV is a proactive, table-driven hop-by-hop distance vector routing protocol that in each node has a routing table that for all reachable destinations stores the next-hop and number of hops for that destination. Each node periodically broadcast routing updates.

To guarantee loop-freedom DSDV uses a sequence numbers to tag each route. The sequence number shows the freshness of a route and routes with higher sequence numbers are favorable. A route R is considered more favorable than R' if

R has a greater sequence number or, if the routes have the same sequence number or, if the routes have the same sequence number but R has low hop-count. The sequence number is increased when a node A detects that a route to a destination D has broken. So the next time node A advertises its routes, it will advertise the route to D with an infinite hop-count and sequence number that is larger than before.

DSDV basically is distance vector with small adjustments to make it better suited for ad-hoc networks. These adjustments consist of triggered updates that will take care of topology changes in the time between broadcasts. To reduce the amount of information in these packets there are two types of update messages defined: full and incomplete dump. The full dump carries all available routing information and the incremental dump that only carries the information that has changed since the last dump [3].

1.4.3 Ad Hoc on Demand Distance Vector Routing (AODV)

AODV is a pure on demand, loop free routing protocol obtained by building on the DSDV algorithm. AODV only requests a route when needed and does not require nodes to maintain routes to destinations that are not actively used in communication. The algorithm uses different messages like Route Request (RREQ) and a hello message (RREP) to discover and maintain routes.

When a source node needs a route to some destination node, and does not already have a valid route to that destination, it initiates the *Route Discovery* process. It broadcasts a route request (RREQ) packet to its neighbors, which then forward the request to their neighbors, and so on, until either the destination or an intermediate node with a fresh enough route to the destination is located. Then a route is made available by unicasting a RREP back to the source.

In the *Route maintenance* process when a node detects that a route to a neighbor no longer is valid, it will remove the routing entry and send a link failure message, a triggered route reply message to the neighbors that are actively using the route, informing them that this route no longer is valid. The nodes that receive this message will repeat this procedure. The message will eventually be received by the

affected sources that can choose to either stop sending data or requesting a new route by sending a new RREQ [2].

1.4.4 Dynamic Source Routing (DSR)

Dynamic Source Routing algorithm belongs to the class of reactive protocols and allows nodes to dynamically discover a route across multiple network hops to any destination. Source routing means that each packet in its header carries the complete ordered list of nodes through which the packet must pass.

In *route discovery* phase the source node that needs a route to some destination node requests a route by broadcasting a Route Request (RREQ) packet. Every node receiving this RREQ searches through its route cache for a route to the requested destinations. DSR stores all known routes in its route cache. If no route is found, it forwards the RREQ further and adds its own address to the recorded hop sequence. This request propagates through the network until either the destination or a node with a route to the destination is reached. When this happens a Route Reply (RREP) is unicasted back to the originator. This RREP packet contains the sequence of network hops through which it may reach the target.

Route maintenance is the mechanism in which a source node S detects if network topology has changed, so that it can no longer use its route to some destination node D. When route maintenance detects a problem with a route in use, a route error packet is sent back to the source node. When this error packet is received, the hop in error is removed from this host's route cache, and all routes that contain this hop are truncated at this point [5].

1.4.5 Temporally Ordered routing Algorithm (TORA)

TORA is a highly adaptive, loop-free, distributed routing algorithm based on the concept of link reversal. It is source-initiated and provides multiple routes for any desired source-destination pair. The key design concept of TORA is the localization of control messages to a very small set of nodes near the occurrence of a

topological change. TORA performs three basic functions route creation, route maintenance, and route erasure.

During the *route creation* and *route maintenance* phases, nodes use a “height” metric to establish a directed acyclic graph (DAG) rooted at the destination. Then the links are assigned a direction, either upstream or down stream based on the relative height metric of neighboring nodes. If a route is broken with node mobility, route maintenance is necessary to re-establish the DAG rooted at the same destination. Upon failure of the last downstream link, a node generates a new reference level, which results in the propagation of that reference level by neighboring nodes, effectively coordinating a structured reaction to the link failure. Links are reversed to reflect the change in adapting to the new reference level.

The “height” metric in TORA is dependent on the logical time of link failure. The “height” metric consists of five elements, namely logical time of a link failure, the unique ID of the node that defined the new reference level, a reflection indicator bit, a propagation ordering parameter, and the unique ID of the node. TORA’s *route erasure* phase involves flooding a broadcast clear packet (CLR) throughout the network to erase invalid routes [15].

1.5 Problem Definition

The intermediate nodes play a vital role in network routing in mobile ad hoc networks. If a node is participating in routing means that it has accepted to forward packets that are destined to a destination through it. Whenever a node is agreed to forward packets and it is not doing so, then that node is said to be misbehaving. These misbehaving nodes may effect the efficiency of the routing algorithm. There comes the need for analysis of routing algorithm, when its nodes are misbehaving.

In this present work the routing misbehavior of Temporally Ordered routing Algorithm has to be studied by using the following quantitative parameters.

- Network size
- Bandwidth
- Mobility rate

- Traffic pattern
- Battery power

In the situation of routing misbehavior the performance of the Temporally Ordered Routing Algorithm has to be evaluated by using the following qualitative parameters.

- End to End Delay
- Delivery Rate
- Overhead Ratio
- Percentage Bandwidth Utilization

ROUTING MISBEHAVIOR IN MANET

Ad hoc networks maximize total network throughput by using all available nodes for routing and forwarding. Therefore, the more nodes that participate in packet routing, the greater the aggregate bandwidth, the shorter the possible routing paths, and the smaller the possibility of a network partition. A node may misbehave by agreeing to forward packets and then failing to do so, because it is overloaded, selfish, malicious, or broken.

- An overloaded node lacks the CPU cycles, buffer space or available network bandwidth to forward packets.
- A selfish node is unwilling to spend battery life, CPU cycles, or available network bandwidth to forward packets not of direct interest to it, even though it expects others to forward packets on its behalf.
- A malicious node launches a denial of service attack by dropping packets.
- A broken node might have a software fault that prevents it from forwarding packets.

Misbehaving nodes can be a significant problem. Ad hoc networks are vulnerable in many settings to nodes that misbehave when routing packets.

One solution to misbehaving nodes is to forward packets only through nodes that share an a priori trust relationship. A priori trust relationships are based on pre-existing relationships built outside of the context of the network (e.g. friendships, companies, and armies). The problems with relying on a priori trust-based forwarding are that

- It requires key distribution

- Trusted nodes may still be overloaded
- Trusted nodes may still be broken
- Trusted nodes may be compromised
- Untrusted nodes may be well behaved.

It may not be possible to exchange keys used to authenticate trusted nodes outside of the ad hoc network before the conference or disaster that requires an ad hoc network. If keys are not distributed ahead of time, then enforcing a priori trust-based forwarding requires a secure channel for key exchanges within the ad hoc network for authentication. Even if keys can be exchanged, a trusted node's security may be compromised, or a trusted node may be overloaded or broken as mentioned above. Finally, although relying on a priori trust-based forwarding reduces the number of misbehaving nodes, it also excludes untrusted well-behaved nodes whose presence could improve ad hoc network performance.

Another solution to misbehaving nodes is to attempt to forestall or isolate these nodes from within the actual routing protocol for the network. However, this would add significant complexity to protocols whose behavior must be very well defined. In fact, current versions of mature ad hoc routing algorithms, including DSDV, AODV, DSR, TORA and others only detect if the receiver's network interface is accepting packets, but they otherwise assume that routing nodes do not misbehave. Although trusting all nodes to be well behaved increases the number of nodes available for routing, it also admits misbehaving nodes to the network [14].

There are two mechanisms to detect and mitigate routing misbehavior, *watchdog* and *pathrater*. The *watchdog* identifies misbehaving nodes, while *pathrater* avoids routing packets through these nodes. When a node forwards a packet, the node's *watchdog* verifies that the next node in the path also forwards the packet. The *watchdog* does this by listening promiscuously to the next node's transmissions. If the

next does not forward the packet, then it is misbehaving. The pathrater uses this knowledge of misbehaving nodes to choose the network path that is most likely to deliver packets.

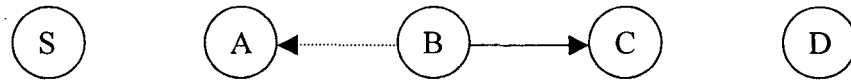


Figure 2.1 : Example for Watchdog.

2.1 Watchdog

The watchdog method detects misbehaving nodes as shown in Figure 2.1. Suppose there is a path from node S to D through intermediate nodes A, B, and C. Node A cannot transmit all the way to node C, but it can listen in on node B's traffic. When node A transmits a packet for B to forward to C, A can often tell if B transmits the packet. If proper encryption is not performed separately for each link, then A can also tell if B has tampered with the payload or the header.

The watchdog technique can be implemented by maintaining a buffer of recently sent packets and comparing each overheard packet with the packet in the buffer to see if there is a match. If so, the packet in the buffer is removed and forgotten by the watchdog, since, it has been forwarded on. If a packet has remained in the buffer for longer than a certain timeout, the watchdog increments a failure tally for the node responsible for forwarding on the packet. If the tally exceeds a certain threshold bandwidth, it determines that the node is misbehaving and sends a message to the source notifying it of the misbehaving node. The watch dog mechanism could be used to some degree to detect replay attacks but would require maintaining a great deal of state information at each node as it monitors its neighbors to ensure that they do not retransmit a packet that they have already forwarded. Also, if a collision has taken place at the receiving node, it would be necessary and correct for a node to retransmit a

packet, which may appear as a replay attack to the node acting as its watchdog. Therefore, detecting replay attacks would neither be an efficient nor an effective use of the watchdog mechanism.

To work properly, the watchdog must know where a packet should be in two hops. If the watchdog does not have this information, then a malicious or broken node could broadcast the packet to a non-existent node and the watchdog would have no way of knowing. Because of this illustration the watchdog works best on top of a source routing protocol.

2.2 Pathrater

The pathrater run at each node in the network, combines knowledge of misbehaving nodes with link reliability data to pick the route most likely to be reliable. Each node maintains a rating for every other node it knows about in the network. It calculates a path metric by averaging the node ratings in the path. Since the pathrater depends on knowing the exact path a packet has traversed, it must be implemented on top of source routing protocol.

The path rater assigns ratings to the node as follows. When a node in the network comes in the view of the pathrater, it assigns it a neutral rating of 0.5. A node always rates itself with a 1.0. By using this mechanism the pathrater selects the shortest length path if all other nodes that are neutral. The pathrater increments the ratings of nodes on all actively used paths by 0.01 at periodic intervals of 200 ms. A path is said to be active if a node has sent a packet within the previous rate increment interval. The maximum value a neutral node can attain is 0.8. A node's rating can be decremented by 0.05 when a link break during packet forwarding is detected and the node is unreachable. The lower bound rating of a neutral node is 0.0. The pathrater does not modify the ratings of nodes that are not currently in active use.

In the present work the routing misbehavior of Temporally ordered routing algorithm (TORA), has been analyzed in the context as mentioned before in the problem definition by using the network simulator NS2 developed at UC Berkely.

TEMPORALLY ORDERED ROUTING ALGORITHM

3.1 Overview of the protocol

Temporally Ordered Routing Algorithm (TORA) is a distributed routing protocol for multi-hop networks. The key concept in TORA is decoupling the generation of far-reaching control messages from the dynamics of the network topology. It is a member of the class referred to as link-reversal algorithm. TORA possesses the following attributes.

- Distributed execution
- Loop-free routing
- Multipath routing
- Reactive or Proactive route establishment and maintenance
- Minimization of communication overhead via localization of algorithmic reaction to topological changes.

TORA provides only the routing mechanisms and depends on Internet MANET encapsulation protocol (IMEP) for underlying functions. TORA has been designed to work on top of lower layer mechanisms or protocols (IMEP), that provide the following basic services between neighboring routers: link status sensing and neighbor discovery, reliable, in-order control packet delivery, link and network layer address resolution and mapping, security authentication. TORA modifies the partially link reversal algorithm in which any node other than the destination that has no outgoing links, reverses incoming links from only those neighbors who have not themselves reversed links previously. Unlike the partially link reversal algorithm TORA detects partitions and informs all the nodes in the partition about the partition. A logically separate version of TORA is run for each destination to which routing is required. The following version of TORA runs for a given destination.

TORA imposes logical directions to the links between routers to form a routing structure that is used to forward packets to the destination. A router imposes a direction either upstream or downstream to the link with a neighboring router based on the relative values of a metric associated with each router. A good analogy would be water flowing down the hill through pipes. Hilltop is the source, pipes are links, and pipe connections are nodes. TORA assigns level numbers to each node down the hill. When two intermediate nodes cannot communicate, the last node raises its level higher than any of its neighbors, so that water, which is data, flows back out of it[15]. The metric maintained by a router can conceptually be thought of as the router's "height" i.e., the links are directed from the higher router to the lower router. A router may only forward packets downstream.

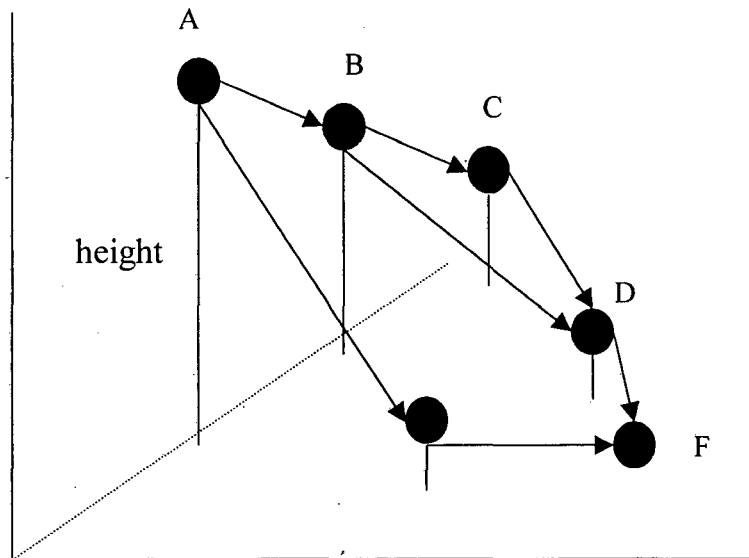


Figure 3: Routes created by TORA

The above figure depicts the directed acyclic graph (DAG) formed by TORA targeted at the destination 'F'. The relative heights of the nodes are also shown.

TORA can be separated into three basic functions. *Creating routes*, *Maintaining routes*, and *Erasing routes*. (The concept of Route Optimization is not considered in this thesis work, because it is under process.)

3.2 Creating Routes

Routes can be initiated on-demand by a source or proactively by a destination. In either case, routers select heights with respect to the given destination and assign directions to the links between neighboring routers.

In the *on-demand/reactive* mode, a source initiates the process by sending a QRY packet to its neighbors that identifies the destination for which a route is requested. The QRY packet is propagated out from the source until one or more routers that have a trusted route to the destination receive it. While forwarding the QRY packet, routers set a route-requested flag and discard any subsequent QRY packets received for the same destination. The route-requested flag suppresses redundant route requests and reduces the need for subsequent route requests when a destination is temporarily unreachable. Routers that have a trusted route to the destination respond to the QRY packet by sending an UPD packet to their neighbors that identifies the relevant destination and the height of the router sending the UPD packet. Routers also maintain the time at which an UPD packet was last sent to its neighbors and the time at which links to neighboring routers became active, to reduce redundant replies to a given route request. When a router with the route-requested flag set receives an UPD packet, it sets its height and sends an UPD packet to its neighbors that identify the relevant destination and the new height of the router sending the UPD packet. Thus, routers in the network select heights for the requested destination learn of their neighbors heights for the destination and assign link directions based on those heights. To ensure that a route request continues to propagate when the destination was initially unreachable, routers with the route-requested flag set must resend a QRY packet upon activation of a new link (i.e., discovery of a new neighbor).

In the *proactive mode*, the destination initiates the route creation process by sending an OPT packet that is processed and forwarded by neighboring routers. The OPT packet identifies the destination, the mode of operation for the destination and the height of the router sending the OPT packet. The OPT packet also contains a sequence number used to uniquely identify the packet and ensure that each router processes and forwards a given OPT packet from a destination at most once. As the OPT packet is forwarded, routers set their mode of operation correspondingly,

reselect their heights, and send an OPT packet to their neighbors that identifies the relevant destination and the new height of the router sending the UPD packet .

3.3 Maintaining Routes

Nodes that have height other than NULL only participate in maintaining routes. Any neighbor that is having height NULL is not used for the computations. Each node keeps the following values.

oid - The old unique ID of the node that defined the new reference level.

toi - A clock tag set to the time of the link failure, where nodes should have synchronized clocks with an external time source such as the Global Positioning System (GPS).

r - A reflection indicator bit.

delta - A propagation ordering parameter, height.

id - The current unique ID of the node itself.

The first three parameters represent reference level of height entry. A new reference level is defined each time a node loses its last downstream link due to a link failure. Last two values define an offset with respect to the reference level, which were the first three values.

Here height NULL is an unknown or undefined height (conceptually an infinite), ZERO is The assumed height of a given destination.

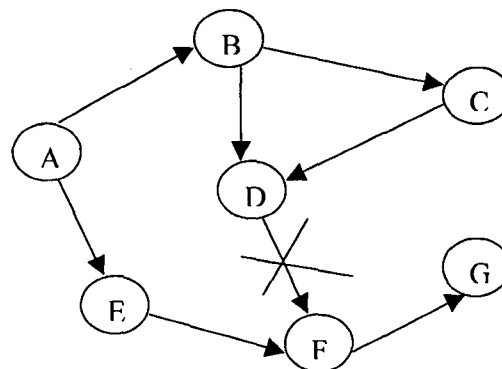


Figure 4: DAG inTORA when a route is broken.

When a node moves, the DAG route is broken and route maintenance is needed to reestablish a DAG for the same destination. When the last downstream link of a node fails, it generates a new reference level. This results in the propagation of that reference level by neighboring nodes as shown in Figure 4. Here, after the link failure between Nodes D and F happens, the new reference level is set as Node D. Links are reversed to reflect the change in adapting to the new reference level. This has the same effect as reversing the direction of one or more links when a node has no downstream links. In route maintenance phase, TORA floods a broadcast clear packet (CLR) to the network to erase invalid routes.

Each node, other than the destination that has no downstream links modifies its height as follows.

Case 1 (Generate):

Node i has no downstream links, due to a link failure.

$(\tau[i], \text{oid}[i], r[i]) = (t, i, 0)$, where t is the time of the failure.

$(\delta[i], i) = (0, i)$

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

Case 2 (Propagate):

Node i has no downstream links, due to a link reversal following reception of an UPD packet and the ordered sets $(\tau[k], \text{oid}[k], r[k])$ are not equal for all neighbors k .

$(\tau[i], \text{oid}[i], r[i]) = \max\{(\tau[k], \text{oid}[k], r[k]) \text{ of all neighbors } k\}$

$(\delta[i], i) = (\delta[m]-1, i)$, where m is the lowest neighbor with the maximum reference level defined above.

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

TH-9483

Case 3 (Reflect):

Node I has no downstream links due to a link reversal following reception of an UPD packet and the ordered sets $(\tau[k], \text{oid}[k], r[k])$ are equal with $r[k] = 0$ for all neighbors k .

$$(\tau[i], \text{oid}[i], r[i]) = (\tau[k], \text{oid}[k], 1)$$

$$(\delta[i], i) = (0, i)$$

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 the bit r .

Case 4 (Detect):

Node i has no downstream links, due to a link reversal following reception of an UPD packet, the ordered sets $(\tau[k], \text{oid}[k], r[k])$ are equal with $r[k] = 1$ for all neighbors k , and $\text{oid}[k] = i$ (i.e., node i defined the level).

$$(\tau[i], \text{oid}[i], r[i]) = (-, -, -)$$

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

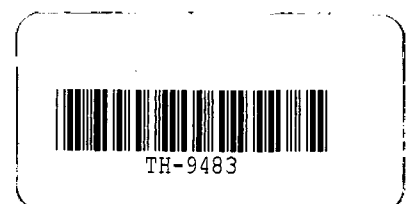
In essence, the last reference level defined by node i has been reflected and propagated back as a higher sub-level from all of its neighbors. This corresponds to detection of a partition.

Case 5 (Generate):

Node i has no downstream links due to a link reversal following reception of an UPD packet, the ordered sets $(\tau[k], \text{oid}[k], r[k])$ are equal with $r[k] = 1$ for all neighbors k , and $\text{oid}[k] \neq i$ (i.e., node i did not define the level).

$$(\tau[i], \text{oid}[i], r[i]) = (t, i, 0), \text{ where } t \text{ is the time of the failure}$$

$$(\delta[i], i) = (0, i)$$



In essence, node i experienced a link failure between the time it propagated a reference level and the reflected higher sub-level returned from all neighbors. This is not necessarily an indication of a partition. Node i defines a new reference level.

After the determination of its new height in cases 1, 2, 3, and 5, node i updates all the entries in its link-status table; and broadcasts an UPD packet to all neighbors k . The UPD packet consists of the destination-ID, j , and the new height of the node i that is broadcasting the packet. When a node i receives an UPD packet from a neighbor k , node i reacts as described in the creating routes section and in accordance with the cases outlined above. In the event of the failure a link (i, k) that is not its last downstream link, node i simply removes the entries $HT_NEIGH[k]$ and $LNK_STAT[k]$ in its height and link-status tables.

3.4 Erasing Routes

In TORA there is a potential for oscillations to occur, especially when multiple sets of coordinating nodes are concurrently detecting partitions, erasing nodes, and building new routes based on each other.

Following the detection of a partition as in case 4, node i sets its height and the height entry for each neighbor k to NULL, other than the destination j if it is a neighbor, in which case the corresponding height entry is set to ZERO), updates all the entries in its link-status table, and broadcast a CLR packet. The CLR packet consists of the destination-ID, j , and the reflected reference level of node i , $(\tau[i], \text{oid}[i], 1)$. In actuality the value $r[i] = 1$ need not be included since it is always 1 for a reflected reference level. When a node i receives a CLR packet from a neighbor k it reacts as follows:

- a) 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 k to NULL (unless the destination j is a neighbor, in which case the corresponding height

entry is set to ZERO), updates all the entries in its link-status table and broadcasts a CLR packet.

b) 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 *k* (with the same reference level as the CLR packet) to NULL and updates the corresponding link-status table entries. Thus, the height of each node in the portion of the network that was partitioned is set to NULL and all invalid routes are erased. If (b) causes node *i* to lose its last downstream link, it reacts as in case 1 of maintaining routes.

3.5 Configuration

Each router is configured with a router ID (RID), which is unique among the routers. Each interface “*i*” of a router consists the following parameters configured.

IP_ADDR[*i*] IP address of the interface.

ADDR_MASK[*i*] Address mask of interface.

PRO_MODE[*i*] Reactive or proactive mode of operation.(0 = OFF, 1 = ON).

MODE_SEQ[*i*] Sequence number associated with mode of interface “*i*”.

For each interface, a network route corresponding to the address and mask of the interface may be added to the routing table. For each interface, a router maintains a sequence number that is incremented upon changes to the interface mode or operation

For each destination “*j*”, a router maintains the following state variable.

HEIGHT[*i*] This router’s height metric for routing to “*j*”.

PRO_MODE[*i*] Sequence number of the most recent mode for “*j*”.

RT_REQ[*i*] Indicates any route request to “*j*” is pending.

TIME_UPD[*i*] Time last UPD packet sent to “*j*” by this router.

For each destination “j”, a router maintains a separate instance of the following state variables for each neighbor “k”.

HT_NEIGH[j][k] The height metric of neighbor “k”.

LNK_STAT[j][k] The assigned status of the link to neighbor “k”.

TIME_ACT[j][k] Time the link to neighbor “k” became active.

Each router may maintain the following auxiliary variables for each destination to which routing is required.

NUM_ACTIVE[j] Number of neighbors, i.e. active links.

NUM_DOWN[j] Number of links marked DN in the LNK_STAT table.

NUM_UP[j] Number of links marked UP in the LNK_STAT table.

The entry in the LNK_STAT table is maintained in accordance with the following rule.

If HT_NEIGH[k] = NULL then LNK_STAT[k] = UN

else if HEIGHT = NULL then LNK_STAT[k] = DN

else if HT_NEIGH[k] < HEIGHT then LNK_STAT[k] = DN

else if HT_NEIGH[k] > HEIGHT then LNK_STAT[k] = UP

The Query (QRY) packet consists of the fields:

Version # - The TORA version number.

Type - The TORA packet type.

Reserved - Field reserved for future use.

Destination IP Address - The IP address for which a route is being requested.

The Update (UPD) packet consists of the fields:

Version # - The TORA version number.

Type - The TORA packet type.

Reserved - Field reserved for future use.

Destination IP Address - The IP address for which a route is being requested.

Mode Sequence # - Sequence number associated with the subsequent mode.

The 'tau', 'oid', 'r', 'delta', 'id' values associated with the destination IP address and mask, of the router sending the UPD.

The Clear (CLR) packet consists of the fields:

Version # - The TORA version number.

Type - The TORA packet type.

Reserved - Field reserved for future use.

Destination IP Address - The IP address for which a route is being requested.

The 'tau', 'oid', 'id' values associated with the destination IP address and mask, of the router sending the UPD.

SIMULATION ENVIRONMENT AND METHODOLOGY

In this dissertation work to study the effect of routing misbehavior in Temporally Ordered Routing Algorithm (TORA), a network simulator is needed which can provide the mobile wireless radio environments. The network simulator should provide the mobility features to simulate the networks like ad hoc networks, which has dynamic topology in which nodes may enter or leave the network at any time. The Network Simulator, NS2 is the one that is the result of an effort of research and development that is administrated by researches at UC, Berkely.

4.1 Network Simulator

NS is a discrete event simulator targeted at networking research. NS simulates variety of IP networks. It implements network protocols such as TCP and UPD, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms such as Dijkstra, and more. NS also implements multicasting and some of the MAC layer protocols for LAN simulations. The simulator is written in C++ and a script language called OTcl.

NS is an an Object oriented Tcl(OTcl) script interpreter that has an event scheduler, network component objects and network setup modules called plumbing modules. The user writes an Otcl script that defines the network (number of nodes, links), the traffic in the network (sources, destinations, type of traffic) and which protocols it will use. The OTcl script initiates an event scheduler, sets up the network topology using the network objects and the plumbing functions in the library, and tells traffic sources when to start and stop transmitting packets through the event scheduler. One can create new object either by writing a new object or by making a compound object from the object library, and plumb the data path through the object.

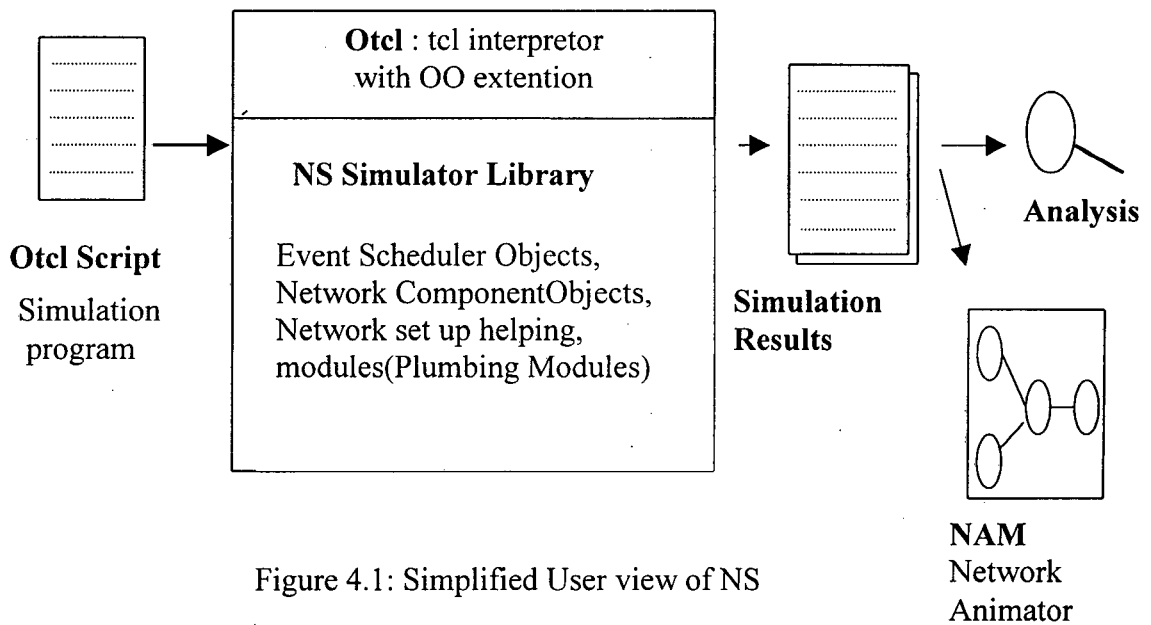


Figure 4.1: Simplified User view of NS

An event in NS is a packet ID that is unique for a packet with scheduled time and the pointer to an object that handles the event. In NS the event scheduler keeps track of simulation time and fires all the events in the event queue scheduled for the current time by invoking appropriate network components, usually the ones who issued the events, and left them do the appropriate action associated with packet pointed by the event. Network components communicate with one another passing packets, however this does not consume actual simulation time. All the network components that need to spend some simulation time handling a packet, use the event scheduler by issuing an event for the packet and waiting for the event to be fired to itself before doing further action handling the packet. For example, a network switch component that simulates a switch with 15 microseconds of switching delay issues an event for a packet to be switched to the

scheduler as an event 15 microsecond later. The scheduler after 15 microsecond dequeues the event and fires it to the switch component, which then passes the packet to an appropriate output link component. Another use of an event scheduler is timer. For example, TCP needs a timer to keep track of a packet transmission time out for retransmission. Timers use event schedulers in a similar manner that delay does. The only difference is that timer measures a time value associated with a packet and does an appropriate action related to that packet after a certain time goes by, and does not simulate a delay[6].

NS separates the data path implementation from control path implementations. In order to reduce packet and event processing time (not simulation time), the event scheduler and basic network component objects in the data path are written and compiled using C++. The compiled objects are made available to the OTcl interpreter through an OTcl linkage that creates a matching OTcl object for each of the C++ objects. The OTcl linkage also makes the control functions and the configurable variables specified by the C++ object act as member functions and member variables of the corresponding OTcl object.

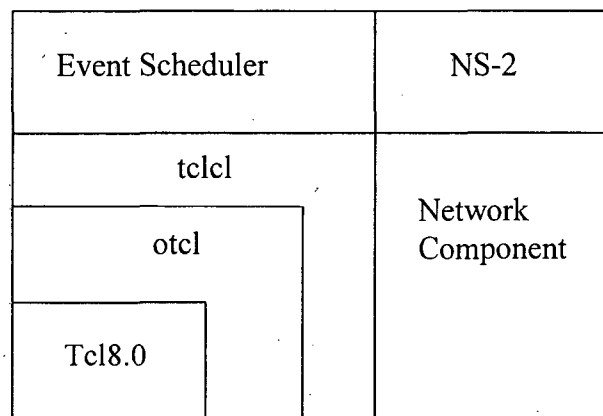


Figure 4.2 : Architectural View of NS

The above figure shows the general architecture of NS. A general user can be thought of standing at the left bottom corner, designing and running simulations in Tcl using the simulator objects in the OTcl library. The event schedulers and most of the network components are implemented in C++ and available to OTcl through an OTcl linkage that is implemented using tclcl.

When a simulation is completed, NS produces one or more text-based output files called trace files that contain the detailed simulation data. The data can be used for simulation analysis. The output data can be used as input to the graphical simulation display tool called *Network Animator* (NAM). Nam has a nice graphical user interface similar to that of a CD player which provides play, fastforward, rewind, pause button options. The network animator also graphically presents the information such as throughput and number of packet drops at each link, although the graphical information cannot be used for accurate simulation analysis.

4.2 MobileNode

MobileNode is the basic NS node object with added functionalities like ability to transmit and receive on a channel that allows it to be used to create mobile, wireless simulation environments. The mobility features including node movement, periodic position updates, maintaining topology boundary etc are implemented in C++ while plumbing of network components within MobileNode itself have been implemented in OTcl.

Each node is a mobile and independent entity that is responsible for computing its own position and velocity as a function of time. The mobility features including node movement, periodic position updates, maintaining topology boundary etc are implemented in C++ while plumbing of network components within Mobile Node itself have been implemented in OTcl [9].

Each mobile node uses a routing agent for the purpose of calculating routes to other nodes in the ad hoc network. Packets are sent from the application and are received by the routing agent. The agent decides a path that the packet must travel in order to

reach its destination and stamps it with this information. It then sends the packet down to the link layer. The link layer level uses an Address Resolution protocol (ARP) to decide the hardware addresses of neighboring nodes and map IP address to their correct interfaces. With this information the packet goes to the interface queue and awaits a signal from the Multiple Access Control (MAC, IEEE 802.11) protocol. When the MAC layer decides it is ok to send it onto the channel, it fetches the packet from the queue and hands it over to the network interface that in turn sends the packet onto the radio channel. Then the packet is delivered to all network interfaces. Each network interface stamps the packet with the receiving interfaces properties and then invokes the propagation model.

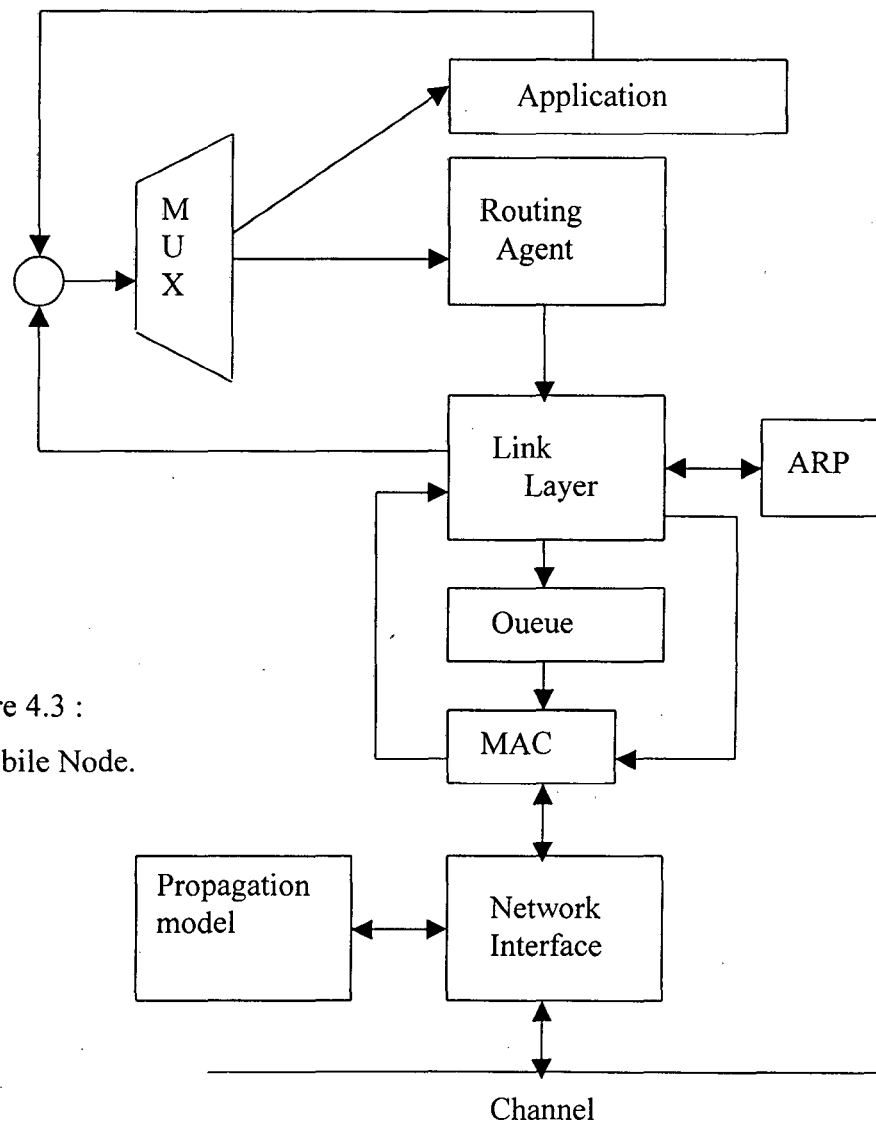


Figure 4.3 :
A Mobile Node.

4.3 Methodology

In this dissertation work, the routing misbehavior of Temporally Ordered routing Algorithm has to be studied using the Network Simulator. As the network simulator does not facilitate an intermediate node to misbehave by dropping packets that are intended to a destination, the algorithm has to be modified to incorporate routing misbehave mechanism. Therefore, to study the impact of routing misbehavior on the TORA the NS needs to have a support for node misbehavior. For this a node misbehaving extension has to be introduced in Network Simulator NS2.

The misbehavior can be induced by adding a parameter to the Temporally Ordered Routing Algorithm at the level of node configuration. The parameter has to be set in the Tcl script, which is an input to the Network Simulator. The parameter will be set or unset based on the assumption that a particular node may misbehave or not. For a particular node if the parameter value is set then that node has to misbehave by dropping the data packets with out forwarding them further. The number of nodes that have to misbehave will be selected randomly by using the random number generator in NS.

After inducing the routing misbehavior in TORA, the effect of routing misbehavior has to be studied with the following parameters that should be varied.

Network size:

Measured in terms of number of nodes. As the network size increases, the number of misbehaving nodes also increases. The size of the network will have its own impact on the performance of the algorithm. Simulation has to be done for different sized networks, with a varying percentage of nodes that are misbehaving.

Traffic pattern

It deals with the number of packets are moving in the network i.e., how effective is a protocol in adapting to non uniform or bursty traffic patterns. When ever in a network with large number of nodes and with less traffic, then some nodes may not participate in routing. If the traffic in a network is bursty then it may lead to congestion.

Bandwidth

One of the factors that can effect the performance of the routing algorithm is the bandwidth that a network is providing. The performance of algorithm has to be tested for different values of bandwidth.

Mobility rate

The rate, at which the nodes are moving around the network, makes an impact on the routing in an ad hoc network. If the node mobility is high, it may lead to the network partition. Then the route maintenance mechanism is needed, and delay in the network will be increased.

Battery power

Every node needs battery power in forwarding and receiving packets. The battery power for a node is limited. In forwarding and receiving packets, the energy of the node will be consumed. It may some times leads to out of battery life and dropping of packets. Some nodes may try to save their batter life to use itself, with out wasting it for other's packets.

End to End delay

End to end delay is referred to as the time taken to delivery packets. Data delay can be divided into queuing delay and propagation delay.

Delivery rate

It is referred as the ratio of number of messages received by destination to the number of messages sent by the source per unit time.

Overhead ratio

It can be defined as the number of control and data bits transmitted per the actual data bits delivered.

Percentage bandwidth utilization

It is the ratio of the overall bandwidth utilized to the total bandwidth available.

The routing misbehavior of TORA has to be analyzed by taking different combinations of the above mentioned parameters. As the nodes in the ad hoc networks are highly mobile in nature, there should be a support for node mobility. To provide the node mobility and the traffic pattern, NS provides two types of scenario files called the node-movement file and the traffic-pattern file. One can create his own scenario files according to his requirement.

4.4 Simulation Overview

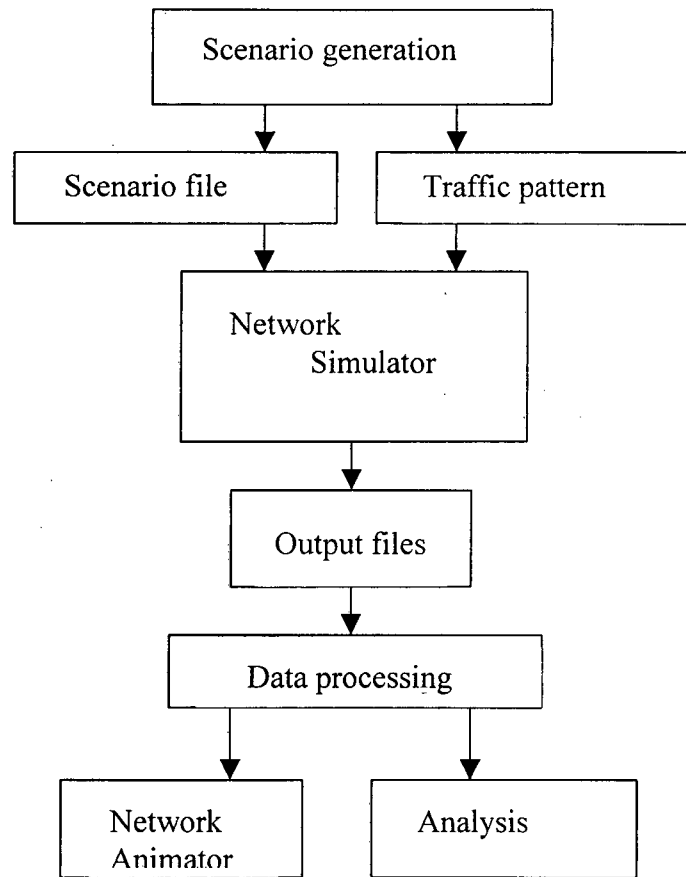


Figure 4.4 : Simulation overview.

The above figure shows the overview of simulation process done in this work. First the scenario files will be generated, which are used for node mobility and traffic pattern as mentioned above. The OTcl script with the modified node configuration uses these scenario files. Then the OTcl script will be given as an input to the Network Simulator. The output files that result from the simulation process will be given as input to the Network Animator. By using the data from the output file, the above mentioned parameters will be evaluated.

Chapter 5

SIMULATION STUDY

5.1 Overview

In this dissertation work the Temporally Ordered Routing Algorithm (TORA) has been simulated and the routing misbehavior of TORA has been studied. The Network Simulator developed at UC Berkeley, is used in our simulation study. The simulations are conducted on the Intel PC: Pentium-III processor, 1 GHz speed, 256 MB RAM, on Linux Operation System.

As mentioned in the problem definition the routing misbehavior of TORA has been analyzed with the specified parameters. The parameters used in this simulation are selected based on the view of efficiency of the routing algorithm in the context of misbehavior. The metrics used to study the efficiency of the routing algorithm are end to end delay, delivery ratio, overhead ratio, and percentage of bandwidth utilization. These metrics have been measured over some parameters that describe the characteristic behavior of an ad hoc network and can be varied in a controlled way. The parameters used in this work are network size, bandwidth, traffic pattern and the mobility rate. The battery power is not considered as the limited resources put a bound on the size of trace file. The trace file grows into GBs. It may be possible if the simulation is run in the distributed environment.

In this simulation work the way of inducing routing misbehavior of the algorithm has been changed due to the time constraint. The misbehavior on the nodes was not introduced the way as it is described in Chapter4. A loss model has been implemented, which will drop the packets at specified range of drop percentage. In NS the loss model is implemented on the node, so that the percentage drop of packets on the nodes can be considered as misbehavior.

The sample Tcl script:

The following sample tcl script is used to set the parameters of the network.

```
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(ant) Antenna/OmniAntenna
set val(ll) LL
set val(ifq) Queue/DropTail/PriQueue
set val(ifqlen) 50
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(rp) DSR
set val(nn) 100
set val(x) 670
set val(y) 670
set val(stop) 200.0
set val(tr) /root/prasant/kumar.tr
set val(nam) /root/prasant/.nam
set val(traffic) "/root/ns-allinone-2.1b8a/ns-2.1b8a/tcl/mobility/scene/test-cbr-25-5"
set val(movement) "/root/ns-allinone-2.1b8a/ns-2.1b8a/tcl/mobility/scene/mv-25-5"
```

The command *set* is used to set the values of the parameters. For example *set val(nn) 25*, sets the value of the network size to 25. *set val(ifq) 50* sets the queue length to 50. As the queue length is set to 50, during the simulation the queue can handle a maximum of 50 packets only. If the queue is full, then the node starts dropping packets.

The other parameters that are fixed during the entire simulation are,

Channel	Wireless Channel
Propagation Model	TwoRayGround
Antenna	OmniAntenna
Queue	DropTail/PriQueue
Queue Length	50
Physical Layer	WirelessPhysical
Mac	802.11

The simulations are run on a grid of size 670X670 and for a time period of 200 seconds with the fixed bandwidth of 1 Mbps.

The LossModel is implemented in the network as shown below.

```
proc UniformErr { } {  
  
    set loss_module [new ErrorModel]  
  
    $loss_module set rate_ 0.3  
  
    $loss_module unit packet  
  
    $loss_module ranvar [new RandomVariable/Uniform]  
  
    return $loss_module  
}
```

Above specified function, induces the packet losses in to the network. Changing the value for the set rate function can vary the percentage packet drop. The drop can be defined on the packet or on time. In the present work, it is defined on packet. In the above module it is set to 30% [9].

5.2 Scenario files generation

The mobility and the traffic pattern are created using the facility provided by NS. The random traffic connections of TCP type or CBR type can be created using the *cbrgen.tcl* file, which comes with NS2, and the node movement scenarios can be generated using *setdest* command in NS. In the present simulation CBR (constant bit rate traffic) is used.

The results of the simulation are two files which contain the resulted data. The output files are called as *trace* file and *nam* file. The *nam* file can be given as an input to the Network Animator, which comes as a part of NS2, to view the simulation graphically. The *trace* file can be used to analyze the simulation results. Each time a simulation is run then the results are written to the trace file specified.

After running the Network Simulator the trace files are studied properly and routines are written to extract the required data from those files. Based on that extracted data for different parameters, the graphs are drawn.

The following steps are followed in the simulation of the Temporally Ordered Routing Algorithm (TORA) to evaluate end-to-end delay, delivery rate, overhead ratio and bandwidth utilization.

- Network size is varied from 10 to 50 nodes, while the mobility rate and the traffic pattern are fixed at 5 m/sec and 5 packets/sec respectively.
- Mobility rate is varied from 1 m/sec to 25 m/sec, while the network size and the traffic pattern are fixed at 25 and 5 packets/sec respectively.
- Traffic pattern is varied from 5 packets/sec to 120 packets/sec, while the network size and the mobility rate are fixed at 25 and 5 m/sec.

5.3 Simulation Results

5.3.1 End-to-End Delay:

Based on the values calculated for end-to-end delay with the variable network size, mobility rate and traffic pattern the following graphs are drawn.

Figure 5.1 shows the end-to-end delay for variable network size, mobility rate and traffic patterns with the percentage of node misbehavior. The end-to-end delay is small i.e. small than 0.5 up to 20% node misbehavior with a varying network size. After 20% node misbehavior the delay is increased in the case of 50 nodes network size. With the increased network size the number of hops for a destination increase, thus the end-to-end delay also increases.

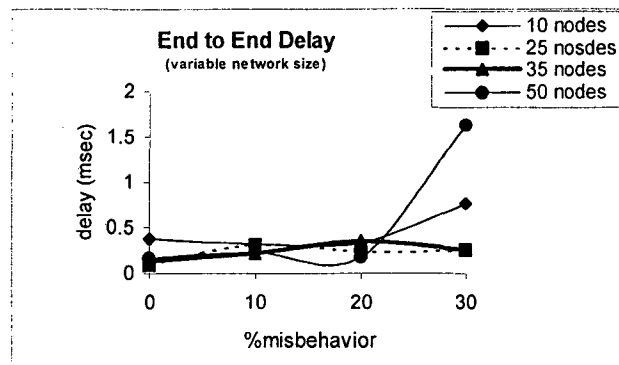


Figure 5.1 : End to End delay Vs percentage of node misbehavior.
(With the variable network size).

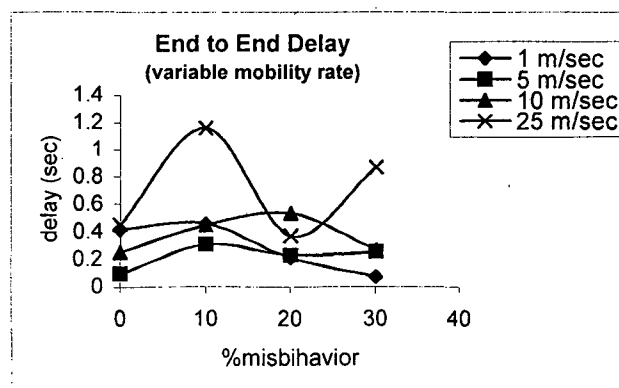


Figure 5.2 : End to End delay Vs percentage of node misbehavior.
(With the variable mobility rate)

In figure 5.2 The end-to-end delay is varying with different % node misbehavior differently in the case of variable mobility rate. As the speed increases the link failures may occur that lead to route reestablishment, which increases the delay. That depends on the direction, and with the speed of the nodes moving around the network.

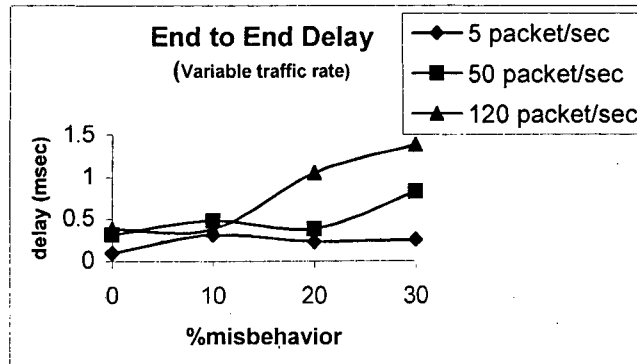


Figure 5.3 : End to End delay Vs percentage of node misbehavior.
(With the variable traffic rate).

In the above figure 5.3 with variable traffic pattern, the end-to-end delay is almost constant for a traffic of 5 packets/sec and it is increased for traffic of 50 and 120 packets/sec. This is because of the increase in traffic that causes congestion which lead to the delay of packets reaching their destination.

5.3.2 Delivery Ratio

Based on the values calculated for delivery ratio with the range of percentage of node misbehavior from 10 to 30% with an increment of 10%, for variable network size, mobility pattern and traffic pattern, the following graphs are drawn.

In the following figure 5.4 the graph shows that the delivery rate is high at 0% node misbehavior for different sized networks. With the increase in % node misbehavior, the delivery rate of the packets is decreased gradually. The network with 10 nodes is showing small delivery rate of packets when the % node misbehavior is increasing, but with a network of size 50 nodes the delivery rate of packets is high. The

increased network size causes the increase in the number of sources that are sending packets which leads to the high traffic through some nodes and resulting into decrease in no of packets received.

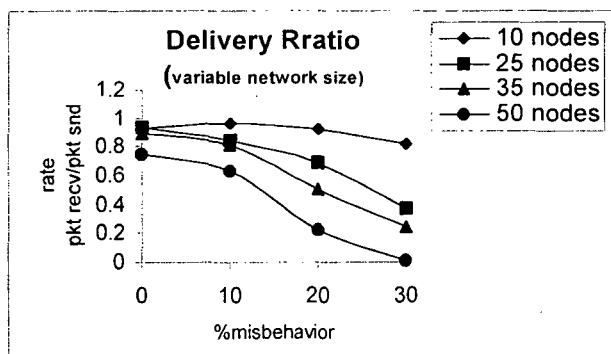


Figure 5.4 : Rate of delivered of packets Vs percentage of node misbehavior.
(With the variable network size).

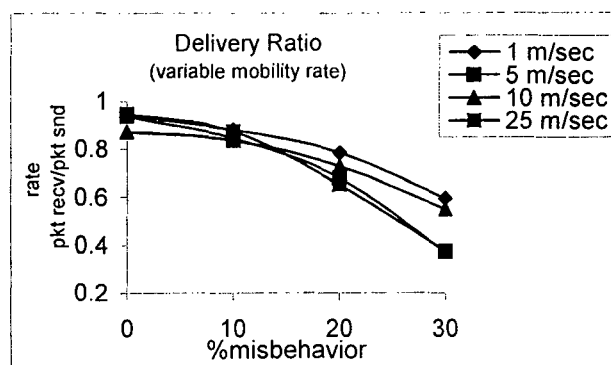


Figure 5.5 : Rate of delivered packets Vs percentage of node misbehavior.
(With the variable mobility rate).

In the above figure 5.5 the delivery ratio is gradually decreasing with the increasing percentage of node misbehavior. The decrease in the delivery ratio is mainly due to the 'IMEP' hello packets. Since the algorithm is reactive, its buffers are filled with the packets for finding routes frequently. IMEP layer floods back the route with 'CLEAR' packets when the destination is unreachable. Dropping the packets(both

control and data) increases the retransmission of packets thus further decreasing the delivery ratio.

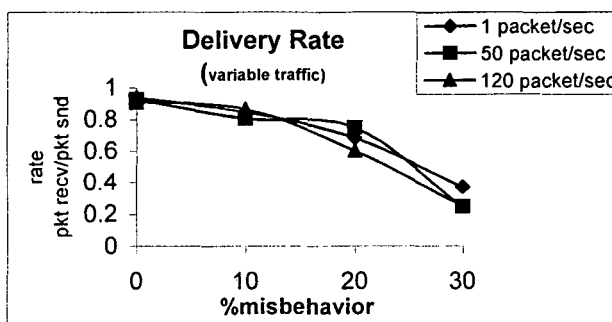


Figure 5.6 : Rate of delivered packets Vs percentage of node misbehavior.
(With the variable traffic rate).

The above figure 5.6 shows the delivery ratio for variable traffic patterns with varying percentage of node misbehavior. The delivery rate is high for the network with 0% node misbehavior and there is a gradual decrement of delivery rate with the increasing percentage of node misbehavior. The delivery ratio is smaller for the network with bursty traffic compared to the other traffic. The increase of the traffic rate causes more collisions, which decreases the delivery ratio.

5.3.3 Overhead Ratio

The following figure shows the overhead ratio for varying percentage of node misbehavior in different sized networks. The overhead ratio of the network with 10 nodes is smaller compared to the other(big) sized networks. With the increase in the size of the network, the overheads for creating and maintaining routes are high. Thus the overhead ratio is increased.

The following figure 5.7 shows the overhead ratio for varying percentage of node misbehavior in different sized networks. The overhead ratio of the network with 10 nodes is smaller compared to the other(big) sized networks. With the increase in the size of the network, the overhead for creating and maintaining routes is high.

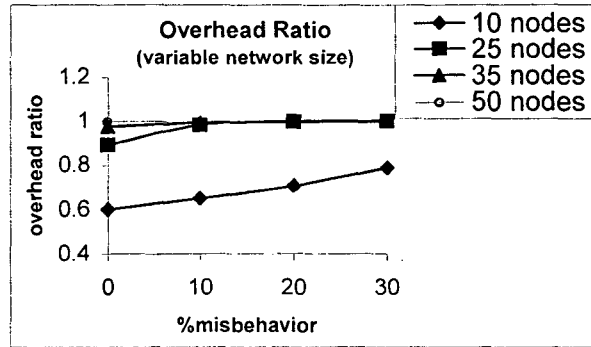


Figure 5.7 : Overhead ratio Vs percentage of node misbehavior.
(With variable network size).

The following figure 5.8 shows the overhead ratio for varying mobility rate with different percentage node misbehavior. With the increase in %node misbehavior, the overhead is increased gradually. The increased mobility rate in a network leads to route failures. This results into increased overhead of the control packets. Frequent control packets increase the collision in the network. At 0% node misbehavior, the network with speed 5 is having smaller overhead compared to the other networks.

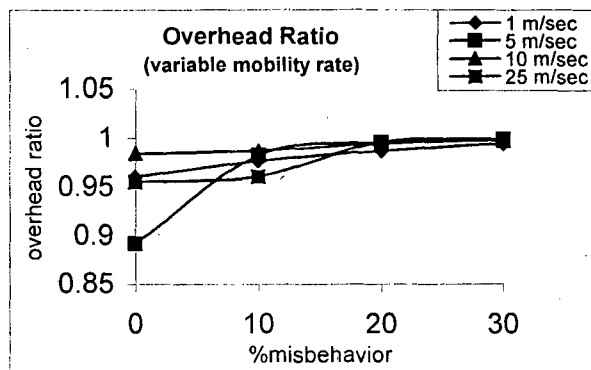


Figure 5.8 : Overhead ratio Vs percentage of node misbehavior with variable network size.

The following figure 5.9 shows the overhead ratio for varying %node misbehavior with variable traffic pattern. The overhead for the traffic of 1packet/sec is smaller

compared to the traffic 50 and 120 packets/sec. With the % node misbehavior, the overhead ratio is increased. The increased traffic in the network causes the congestion in the network, which increases the overheads.

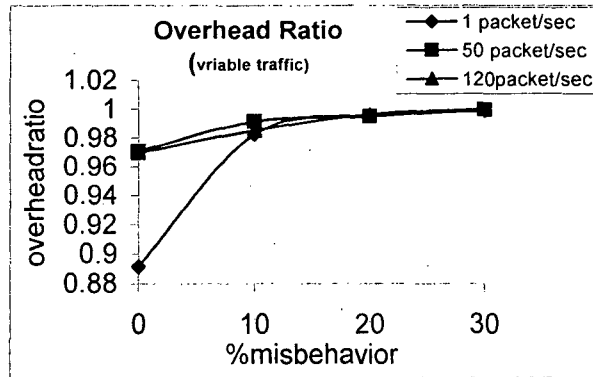


Figure 5.9 : Overhead ratio Vs % node misbehavior with variable traffic pattern.

5.3.4 Bandwidth Utilization

The following figures are drawn based on the values calculated during the simulation process.

The following figure 5.10 is drawn for bandwidth utilization over the %node misbehavior with variable network size. The bandwidth utilization is decreased with the increasing percentage of node misbehavior. The network size of 25 nodes is showing the gradual decrement in bandwidth utilization. With the increased network size the number of sources sending packets is increased, the control packets are also increased which increase the collisions and congestion in the network preventing the transmission of data packets.

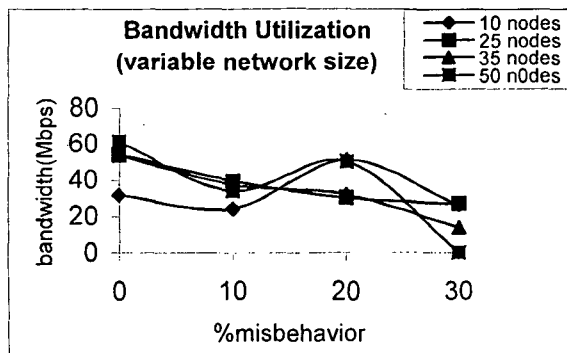


Figure 5.10 : Bandwidth utilization Vs %node misbehavior with variable network size.

The following figure 5.11 shows the bandwidth utilization for varying percentage of node misbehavior, with variable mobility rate. The bandwidth utilization decreases with the % node misbehavior for some extent and again it is increased. The threshold for the bandwidth utilization may be 20% node misbehavior since the trend in all the curves shows sudden change at this point.

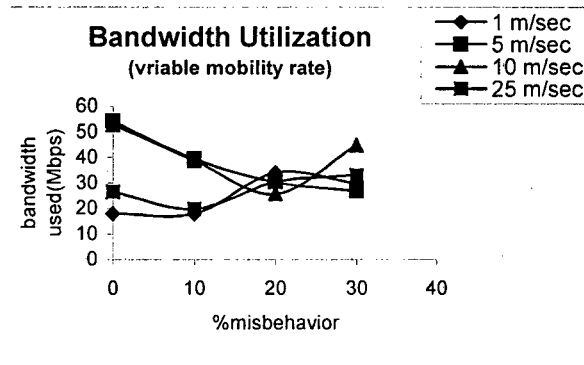


Figure 5.11 : Bandwidth utilization Vs %node misbehavior with variable mobility rate.

The following figure shows the bandwidth utilization over %node misbehavior with variable traffic pattern. The bandwidth utilization is increased with the increasing %node misbehavior. Network with small traffic shows the high utilization over the high traffic due to the control message overhead in the high traffic networks. For traffic of 20 packet/sec it is showing the disordered variation of bandwidth utilization.

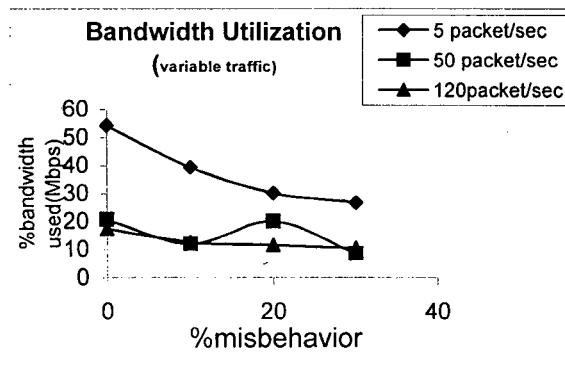


Figure 5.12 : Bandwidth utilization Vs % node misbehavior with variable traffic.

Conclusion

The present work has simulated the routing misbehavior of Temporally Ordered Routing Algorithm (TORA) used in mobile ad-hoc networks. The simulation results show varying characteristics of the network in different simulation scenarios. The simulation results have been represented with the help of graphs. These graphs show some regular trends where it has been easier to make categorical conclusions about the misbehavior of the algorithm. But in some cases it is quite difficult to make definite conclusions since the results shows irregular patterns. Attempt has been made to conclude about such results but they are not definite.

However, it is relevant to give if some conclusions which have been drawn in the present simulation. The simulation results are shown that the end-to-end delay is high in the case of large sized networks. The overhead ratio is very high for TORA due to its link reversal process and its dependency on the IMEP layer. There is a gradual decrement in the delivery ratio with the increasing % misbehavior in the network. In the case of bandwidth utilization, the disordered variation is observed.

Since no work can be completed within a specified period in its totality, the present work in the similar lines has its own limitations. Following are some of the limitations on the present work:

- The routing misbehavior has not been introduced in its true sense.
- The misbehavior of the algorithm has not been evaluated in the context of varying battery power.

Therefore, it is very obvious the project can be extended to overcome the above mentioned limitations. However, the work can have additional scope for future extensions that has not been considered in the scope of the present work. For example more parameters can used for evaluation as well as other network context can also be considered.

References

- [1] Andrew S. Tanenbaum, “ *Computer Networks* ”, PHI publications, 3rd edition 2000.
- [2] Charles E. Perkins, Elizabeth M. Royer, Samir R. Das, “ *Ad Hoc on Demand Distance Vector (AODV) Routing*”, Internet Draft, March 2001, www.ietf.org/internet-drafts/draft-ietf-manet-aodv-08.txt.
- [3] Charles E. Perkins, P. Bhagwat, “ *Highly dynamic DSDV for Mobile computer*”, SIGCOM' 94 conference on Communication Architecture, protocols and Applications, www.cs.umd.edu/projects/meml/papers/Sigcomm94.ps
- [4] Charles E. Perkins, “ *Mobility support, Mobile IP and Wireless Channel Support for ns-2* ”, www.svrloc.org/~charliep/mobins2
- [5] David B Johnson, David A. Maltz, Yih-Chun Hu , Jorjeta G. Jetcheva “ *The Dynamic Source Routing algorithm for Mobile Ad hoc networks* ” , Internet Draft from IETF MANET Working Group, www.ietf.org/internet-drafts/draft-ietf-manet-dsr-05.txt
- [6] Jae Chung, Mark Claypool, “ *NS by Example*”, WPI Computer Science, www.nile.wpi.edu/NS.
- [7] J. Broch, D. Maltz, D. Johnson, Y. HU, J. Jetcheua, “ *A Performance Comparison of Multi-hop Ad-hoc Network Routing Protocols*”, ACM/IEEE International Conference on Mobile Computing and Networking

- [8] Jochen H Shiller ,“*Mobile Communications*”, PHI publications
- [9] Kevin Fall, Kannan Varadhan, ”*Ns Manual*”, VINT project, May 2001, www.mach.cs.berkeley.edu/ns/ns-man.html.
- [10] Mark Gries, “ *Tutorial for Network Simulator NS-2*”
www.isi.edu/nsnam/ns/tutorial
- [11] Nitin H. Vaidya. Texas A & M University "*Mobile Ad-Hoc Networks: Routing, Mac and Transport issues*", www.cs.tamu.edu/faculty/vaidya.
- [12] Padmini Mishra "*Routing Protocols for Ad Hoc Mobile wireless Networks*", www.cis.ohio-state.edu/~jain/cis788-99/adhoc_routing/index.html.
- [13] Santhosh R. Thampuran, "*Routing Protocols for AD Hoc Networks of Mobile Nodes*", www.unix.ecs.umass.edu/~sthampur/Papers/AdhocRouting.pdf
(Mobicom 98).
- [14] Sergio Marti, T. J. Giuli, Kevin Lai, Mary Baker, "*Mitigating Routing Misbehavior in Mobile Ad Hoc Networks*", MOBICOM 2000.
- [15] V Park, S Corson, "*Internet Draft for TORA*", IETF MANET working group, July, 2001, www.ietf.org/internet-drafts/draft-ietf-manet-tora-spec-04.txt

Appendix – Simulation Results

Simulation results for different combinations of parameters are as follows.

Network Size

Network Size : 10 nodes , mobility rate : 5 m/sec , traffic rate 5 packets/sec

% Misbehavior		0%	10%	20%	30%
End to End delay		0.380863	0.323345	0.333100	0.770480
Delivery Rate	Packets Receive	4096	14968	947	390
	Send	4400	15562	1027	476
	Ratio	0.930909	0.96183	0.9221032	0.8193277
Overhead Ratio	IMEP	6438	31669	3331	2567
	Ack	9981	32782	2071	819
	Cbr	10915	34472	2237	909
	Ratio	0.6006804	0.651626	0.7071606	0.788358
Bandwidth utilization		31.729567	24.215420	51.241595	26.044625

Network size: 25, Mobility rate: 5 m/sec, Traffic rate: 5 packet/sec

% Misbehavior		0%	10%	20%	30%
End to End delay		0.094213	0.3.9999	0.230929	0.256310
Delivery Rate	Packets Received	2358	849	365	100
	Sent	2517	1004	536	268
	Ratio	0.936830	0.845618	0.680970	0.373134
Overhead ratio	IMEP	43197	134786	214611	218029
	Ack	5787	2161	832	200
	Cbr	5962	2435	950	267
	Ratio	0.891494	0.982319	0.9956098	0.998800
Bandwidth utilization		54.276883	39.417504	30.134930	26.700258

Network size: 35 nodes, Mobility rate: 5 m/sec, Traffic rate: 5 packets/sec

% Misbehavior		0%	10%	20%	30%
End to End delay		0.137098	0.222659	0.354189	0.259673
Delivery Rate	Packets Receive	1095	475	84	25
	Send	1226	588	167	101
	Ratio				
Overhead ratio	IMEP	122270	166460	192923	173814
	Ack	2726	1135	157	39
	Cbr	3092	1377	195	61
	Ratio				
Bandwidth utilization		53.662760	37.550720	31.978321	13.653988

Network size: 50 nodes, Mobility rate: 5 m/sec, Traffic rate: 5 packets/sec

% Misbehavior		0%	10%	20%	30%
End to End delay		0.166839	0.232544	0.179148	1.636332
Delivery Rate	Packets Receive	776	369	42	2
	Send	1044	588	186	125
	Ratio				
Overhead ratio	IMEP	476376	481404	356807	345425
	Ack	1866	863	68	2
	Cbr	2183	1036	120	21
	Ratio				
Bandwidth utilization		61.197789	33.925967	50.279611	0.056998

Mobility Rate

Mobility rate: 1 m/sec, Network size: 25 nodes, Traffic rate: 5 packets/sec

% Misbehavior		0%	10%	20%	30%
End to End delay		0.41842	0.462412	0.210958	0.075833
Delivery Rate	Packets Receive	9308	2185	478	190
	Send	9905	2480	603	319
	Ratio	0.9727	0.881048	0.786184	0.595611
Overhead ratio	IMEP	586854	25444	98854	119941
	Ack	22634	5725	1102	550
	Cbr	24573	6271	1332	680
	Ratio	0.959983	0.881048	0.786184	0.994388
Bandwidth utilization		18.003753	17.87702	33.965075	29.693296

Mobility rate: 5 m/sec, Network size: 25 nodes, Traffic rate: 5 packet/sec

% Misbehavior		0%	10%	20%	30%
End to End delay		0.094213	0.3.9999	0.230929	0.256310
Delivery Rate	Packets Received	2358	849	365	100
	Sent	2517	1004	536	268
	Ratio	0.936830	0.845618	0.680970	0.373134
Overhead ratio	IMEP	43197	134786	214611	218029
	Ack	5787	2161	832	200
	Cbr	5962	2435	950	267
	Ratio	0.891494	0.982319	0.9956098	0.998800
Bandwidth utilization		54.276883	39.417504	30.134930	26.700258

Mobility rate: 10 m/sec, Network size: 25 nodes, Traffic rate: 5 packets/sec

% Misbehavior		0%	10%	20%	30%
End to End delay		0.252054	0.452259	0.534623	0.273466
Delivery Rate	Packets Receive	1802	1377	630	137
	Send	2065	1646	864	248
	Ratio	0.872639	0.836574	0.7291667	0.5524194
Overhead ratio	IMEP	283933	266475	298351	138193
	Ack	4345	3217	1412	269
	Cbr	4658	3571	1691	339
	Ratio	0.984099	0.836574	0.7291667	0.5524194
Bandwidth utilization		52.895884	38.840091	25.6543	44.762236

Mobility rate: 25 m/sec, Network size: 25 nodes, Traffic rate: 5 packets/sec

% Misbehavior		0%	10%	20%	30%
End to End delay		0.453123	0.161250	0.370702	0.87195
Delivery Rate	Packets Receive	12652	3287	436	117
	Send	13355	3756	667	311
	Ratio	0.947361	0.875133	0.653673	0.3762058
Overhead ratio	IMEP	593601	18769	280743	293662
	Ack	27227	7177	978	217
	Cbr	29180	7984	1202	339
	Ratio	0.955108	0.960411	0.995752	0.998807
Bandwidth utilization		26.52135	19.770335	30.296733	32.954490

Traffic rate

Traffic rate: 5 packet/sec Network size: 25, Mobility rate: 5 m/sec

% Misbehavior		0%	10%	20%	30%
End to End delay		0.094213	0.3.9999	0.230929	0.256310
Delivery Rate	Packets Received	2358	849	365	100
	Sent	2517	1004	536	268
	Ratio	0.936830	0.845618	0.680970	0.373134
Overhead ratio	IMEP	43197	134786	214611	218029
	Ack	5787	2161	832	200
	Cbr	5962	2435	950	267
	Ratio	0.891494	0.982319	0.9956098	0.998800
Bandwidth utilization		54.276883	39.417504	30.134930	26.700258

Traffic rate: 50 packets/sec, Network size: 25, Mobility rate: 5 m/sec

% Misbehavior		0%	10%	20%	30%
End to End delay		0.315016	0.48037	0.383796	0.826866
Delivery Rate	Packets Receive	2344	828	354	97
	Send	2528	1029	474	388
	Ratio	0.927215	0.804663	0.746835	0.25
Overhead ratio	IMEP	177981	250098	171168	858269
	Ack	4736	1751	710	226
	Cbr	5506	2216	837	297
	Ratio	0.970748	0.991278	0.995154	0.9996542
Bandwidth utilization		20.662393	12.091401	20.126017	8.566709

Traffic rate: 120 packets/sec, Network size: 25 nodes, Mobility rate: 5 m/sec

% Misbehavior		0%	10%	20%	30%
End to End delay		0.380338	0.379613	1.050428	1.387349
Delivery Rate	Packets Receive	2123	1105	257	97
	Send	2335	1276	427	382
	Ratio	0.909208	0.865987	0.601874	0.253927
Overhead ratio	IMEP	163984	178564	203561	882062
	Ack	4628	2616	531	202
	Cbr	5266	2715	742	287
	Ratio	0.969714	0.985236	0.996378	0.9996748
Bandwidth utilization		17.437513	12.754895	11.601876	10.595743