AN IMPROVED TCP CONGESTION CONTROL ROUTING PROTOCOL FOR WIRED-CUM-WIRELESS NETWORKS

A PROJECT REPORT

Submitted by

GLADSON V MANUEL (080105601032)

ANVIN JOHNSON (080105601005)

ALLEN JOSEPH (080105601003)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

ANNAI MATHAMMAL SHEELA ENGINEERING COLLEGE, NAMAKKAL

ANNA UNIVERSITY OF TECHNOLOGY

COIMBATORE-641047

APRIL 2012

ANNA UNIVERSITY OF TECHNOLOGY

COIMBATORE-641047

BONAFIDE CERTIFICATE

Certified that this project report **"AN IMPROVED TCP CONGESTION CONTROL ROUTING PROTOCOL FOR WIRED-CUM-WIRELESS NETWORKS"** is the bonafide work of **"GLADSON.V.MANUEL**, **ANVIN JOHNSON, ALLEN JOSEPH"** who carried out the project work under my supervision.

SIGNATURE

Prof.J.VIJIPRIYA,M.E.,MSc.,(Ph.D).,

SUPERVISOR

DEPT OF COMPUTER SCIENCE & ENGG

ANNAI MATHAMMAL SHEELA

ENGINEERING COLLEGE,

ERUMAPATTY,

NAMAKKAL DT.

SIGNATURE

Prof.J.VIJIPRIYA,M.E.,MSc.,(Ph.D).,

HEAD OF THE DEPARTMENT

DEPT OF COMPUTER SCIENCE & ENGG

ANNAI MATHAMMAL SHEELA

ENGINEERING COLLEGE,

ERUMAPATTY,

NAMAKKAL DT.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We take great pleasure in expressing our profound gratitude to all those who have helped us for the successful completion of the project.

We are grateful to our respected **Management, Annai Mathammal Sheela Engineering College, Namakkal,** on guiding light for us in all the activities we had in our college premises.

We would like to express our sincere thanks to our respected **Principal Prof. Dr.C.MANOHARAN,** for providing an opportunity to carry out the project work.

We are very much indebted to the **Head of the Department of Computer Science and Engineering Prof. J.VIJIPRIYA,M.E.,M.Sc.,(Ph.D).,** for her support and guidance towards us and for her gracious permission to undertake this project.

We would like to express our sincere gratitude to our project coordinators **Prof. K.SARAVANAN,M.E.,** and **Prof. A.ILLAYARAJAA,M.E., Department of Computer Science and Engineering** for their enormous motivation, timely suggestions and guidance.

We are indeed grateful to our project guide **Prof. J.VIJIPRIYA,M.E.,M.Sc.,(Ph.D)., Department of Computer Science and Engineering** who helped and encouraged us throughout this project.

Above all, we thank the almighty for giving us the courage and wisdom to undertake this project and complete it successfully.

Last but not the least, we owe special thanks to all others who helped us to bring this project successfully.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	Abstract	**i**
	List of Figures	**ii**
	List of Abbreviations	**iii**
1	**Introduction**	**1**
2	**System Analysis**	**2**
	2.1 Literature survey	2
	2.2 Existing System	6
	2.3 Proposed System	8
	2.4 Feasibility Study	10
3	**System Specification**	**11**
	3.1 Hardware Requirements	11
	3.2 Software Requirements	11
4	**Software Description**	**12**
	4.1 Network Simulator	12
5	**Project Description**	**16**
	5.1 Problem Definition	16
	5.2 Overview of the Project	18
	5.3 Module Description	20
	5.4 Data Flow Diagram	24
	5.5 Input Design	26
	5.6 Output Design	26

CHAPTER NO.	TITLE	PAGE NO.
6	**System Testing**	**27**
	6.1 Unit Testing	27
	6.2 Integration Testing	27
	6.3 Test Cases	28
7	**System Implementation**	**29**
8	**Conclusion & Future Enhancements**	**30**
	8.1 Conclusion	30
	8.2 Future Enhancements	31
9	**Appendix**	**32**
	9.1 Sample Code	32
	9.2 Screen Shots	43
10	**References**	**47**

Abstract

TCP is a connection oriented protocol that offer reliable data transfer. But TCP is not well suited for several emerging applications including streaming and real time audio and video because it increases end-to-end delay and delay variations. As the growth of networking has sparked, the demand for low cost routing and congestion control mechanism boomed to a higher level. This project is helps to increase the network throughput by efficient low cost routing and congestion control mechanism. The routing algorithm finds out optimal path and the congestion control algorithm, NRDC reduce the packet collision and packet loss. Simulations are done using ns2, a discrete event simulator. The result of simulation is compared with other TCP variants. The Comparison of simulation result shows that the proposed algorithm performs better in terms of throughput.

List of Figures

Figure No.	Figure Name	Page No.
5.2.1	System Architecture	18
5.4.1	Routing Algorithm	24
5.4.2	NRDC Algorithm	25

List of Abbreviations

WBA	-	Wireless Broadcast Advantage
WRP	-	Wireless Routing Protocol
AIMD	-	Additive Increase Multiplicative Decrease
AODV	-	Ad Hoc On-Demand Distance Vector Routing
BFA	-	Bellman-Ford Algorithm
DSDV	-	Destination-Sequenced Distance-Vector
ECN	-	Explicit Congestion Notification
EDR	-	Expected Data Rate
EEOR	-	Energy Efficient Opportunistic Routing
LCT	-	Link Cost Table
MRL	-	Message Retransmission List
NRDC	-	Newton Raphson Division Congestion
RTT	-	Round Trip Time
MANET	-	Mobile Ad Hoc Network
DSDV	-	Destination Sequenced Distance Vector
CBR	-	Constant Bit Rate
VBR	-	Variable Bit Rate
DVR	-	Distance Vector Routing
FIFO	-	First In First Out

CHAPTER 1

Introduction

Computer networks, both wired and wireless networks, have been growing tremendously in the last decade. Many wired networks, wireless networks, mobile ad hoc networks and wireless sensor networks are being developed which interconnects computers used by both private and public sectors. In most of these networks the protocol stack used for interconnection is TCP/IP. Due to this reason the TCP connection itself is facing several challenges such as packet loss, congestion, routing cost etc. This challenges lead to the throughput degradation of these TCP networks.

In this project we try to explore the possibility to find out an efficient route to transmit the packet without congestion and minimize packet loss. Here we have proposed a scheme of protocol to overcome the packet loss by applying a congestion control algorithm over the feasible and low cost route which we find out using the routing algorithm. In this project we perform a comparative study by coupling different routing algorithms (Bellman-Ford, DSDV) with congestion control algorithms to find out efficient method NRDC (Newton Rapson Division congestion control) for routing a packet data in the network. The proposed NRDC algorithm together with routing protocol to provide better performance under performance scenario such as throughput and node mobility.

CHAPTER 2

System Analysis

2.1 Literature Survey

The TCP [8] protocol actually corresponds to the transport layer of TCP/IP suite. TCP provides a communication service at an intermediate level between an application program and the Internet Protocol (IP). When an application program desires to send a large chunk of data across the Internet using IP, instead of breaking the data into IP-sized pieces and issuing a series of IP requests, the software can issue a single request to TCP and let TCP handle the IP details.

TCP is a reliable stream delivery service that guarantees that all bytes received will be identical with bytes sent and in the correct order. Since packet transfer is not reliable, a technique known as positive acknowledgment with retransmission is used to guarantee reliability of packet transfers. This fundamental technique requires the receiver to respond with an acknowledgment message as it receives the data. The sender keeps a record of each packet it sends. The sender also keeps a timer from when the packet was sent, and retransmits a packet if the timer expires before the message has been acknowledged. The timer is needed in case a packet gets lost or corrupted.

TCP is a connection-oriented protocol [11] that offers reliable data transfer as well as flow and congestion control. TCP maintains a congestion window that controls the number of outstanding unacknowledged data packets in the network. Sending data consumes slots in the window of the sender and the sender can send packets only as long as free slots are available.

The major drawback in the existing TCP is that the minimum network throughput. The existing system does not consider about the packet loss or packet congestion. The routing of packet data is only done. The collision also cannot be detected. This is performed on the basis that no congestion occur in the network.

On start-up, TCP [10] performs slowstart, during which the rate roughly doubles each round-trip time to quickly gain its fair share of bandwidth. In steady state, TCP uses the AIMD mechanism to detect additional bandwidth and to react to congestion. When there is no indication of loss, TCP increases the congestion window by one slot per round-trip time.

In case of packet loss, indicated by a timeout, the congestion window is reduced to one slot and TCP reenters the slowstart phase. Packet loss indicated by three duplicate ACKs results in a window reduction to half of its previous size.

In [11] the additive increase/multiplicative-decrease (AIMD) algorithm is a feedback control algorithm best known for its use in TCP Congestion Avoidance. AIMD combines linear growth of the congestion window with an exponential reduction when congestion takes place. Multiple flows using AIMD congestion control will eventually converge to use equal amounts of a contended link. The related schemes of multiplicative-increase/multiplicative-decrease (MIMD) and additive-increase/additive-decrease (AIAD) do not converge.

The approach taken is to increase the transmission rate (window size), probing for usable bandwidth, until loss occurs. The policy of additive increase may, for instance, increase the congestion window by a fixed amount every round trip time. When congestion is detected, the transmitter decreases the transmission rate by a multiplicative factor; for example, cut the congestion window in half after loss. The result is a saw-tooth behavior that represents the probe for bandwidth.AIMD requires a binary signal of congestion. Most frequently, packet loss serves as the signal; the multiplicative decrease is triggered when a timeout or acknowledgement message indicates a packet was lost. It is also possible for in-network mechanisms to mark congestion (without discarding packets) as in Explicit Congestion Notification (ECN).

Since there is no specific mechanism for congestion avoidance or congestion control the congestion or packet loss cannot be identified. Thus the existing system never consider the packet loss in data transmission.

WRP uses an enhanced version of the distance-vector routing protocol, which uses the Bellman-Ford algorithm to calculate paths. Because of the mobile nature of the nodes within the MANET, the protocol introduces mechanisms which reduce route loops and ensure reliable message exchange.WRP, similar to DSDV, inherits the properties of the distributed Bellman-Ford algorithm. To counter the count-to-infinity problem and to enable faster convergence, it employs a unique method of maintaining information regarding the shortest distance to every destination node in the network and the penultimate hop node on the path to every destination node. Since WRP, like DSDV, maintains an up-to-date view of the network, every node has a readily available route to every destination node in the network. It

differs from DSDV in table maintenance and in the update procedures. While DSDV maintains only one topology table, WRP uses a set of tables to maintain more accurate information. The tables that are maintained by a node are the following: distance table (DT), routing table (RT), link cost table (LCT), and a message retransmission list (MRL).

The Bellman–Ford algorithm computes single-source shortest paths in a weighted digraph. For graphs with only non-negative edge weights, the faster Dijkstra's algorithm also solves the problem. Thus, Bellman–Ford is used primarily for graphs with negative edge weights. It consists of the following steps:

1. Each node calculates the distances between itself and all other nodes within the AS and stores this information as a table.
2. Each node sends its table to all neighboring nodes.
3. When a node receives distance tables from its neighbors, it calculates the shortest routes to all other nodes and updates its own table to reflect any changes.

A distance-vector routing protocol [7] is one of the two major classes of routing protocols, the other major class being the link-state protocol. Distance-vector routing protocols use the Bellman-Ford algorithm to calculate paths. Distance Vector means that Routers are advertised as vector of distance and Direction. Direction is simply next hop address and exit interface and Distance means hop count. Routers using distance vector protocol do not have knowledge of the entire path to a destination. Instead DV uses two methods:

1. Direction in which router or exit interface a packet should be forwarded.
2. Distance from its destination.

In distance vector routing, the least cost route between any two nodes is the route with minimum distance. In this protocol, as the name implies, each node maintains a vector (table) of minimum distance to every node. As the name suggests the DV protocol is based on calculating the direction and distance to any link in a network. The cost of reaching a destination is calculated using various route metrics. RIP uses the hop count of the destination whereas IGRP takes into account other information such as node delay and available bandwidth.

In data networking and queuing theory, network congestion [4] occurs when a link or node is carrying so much data that its quality of service deteriorates. Typical effects include queuing delay, packet loss or the blocking of new connections. A consequence of these latter two is that incremental increases in offered load lead either only to small increases in network throughput, or to an actual reduction in network throughput.

Network protocols which use aggressive retransmissions to compensate for packet loss tend to keep systems in a state of network congestion even after the initial load has been reduced to a level which would not normally have induced network congestion. Thus, networks using these protocols can exhibit two stable states under the same level of load. The stable state with low throughput is known as congestive collapse.

Routing [2] is the process of selecting paths in a network along which to send network traffic. Routing is performed for many kinds of networks, including the telephone network (Circuit switching), electronic data networks (such as the Internet), and transportation networks.

In packet switching networks, routing directs packet forwarding, the transit of logically addressed packets from their source toward their ultimate destination through intermediate nodes, typically hardware devices called routers, bridges, gateways, firewalls, or switches. General-purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time, but multipath routing techniques enable the use of multiple alternative paths.

Routing,[4] in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Because structured addresses allow a single routing table entry to represent the route to a group of devices, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging) in large networks, and has become the dominant form of addressing on the Internet, though bridging is still widely used within localized environments.

2.2 Existing System

In the existing system the routing of a data packet is made possible by a routing algorithm alone. Here the problem of packet loss or congestion is not addressed properly. Thus while transmitting a data packet, packet loss can be occurred and since there is no mechanism for congestion control or congestion avoidance. In the existing system packet loss cannot be identified easily and thus the number of retransmission is high. So the energy consumption for transmission and retransmission of data packet is very high in the existing system.

The existing system uses a fixed forwarder list to route the data packet. The protocols ExOR and MORE did not explore the benefit of selecting the appropriate forwarding list to minimize the energy cost. In contrast opportunistic routing like ExOR and MORE allows any node that overhears the transmission to participate in forwarding the packet. The existing system is clearly based on the wireless broadcast advantage (WBA) mechanism. The advantage of WBA is more obvious in a multi-hop wireless network, especially when a source node and the destination node are far way, i.e., the packet from the source node to a target node must be routed through a multi-hop path. In ExOR the source node selects a subset of its neighbouring nodes as forwarder list. The forwarder list is prioritized to indicate which nodes have higher priority to forward the packet. Then one or more nodes in the forwarder list, which received the packet successfully, will opportunistically act as new source nodes and route the packet to the target node.

The possibility for arising congestion is also not addressed in the system. The congestion in wireless network cause packet loss and node mobility. The congestion cause packets to be retransmitted which leads to high throughput degradation The existing system never considers this unwanted energy consumption due to the congestion which leads to the packet loss and packet retransmission. In this system each packet is retransmitted a minimal number of times, and covers the longest possible distance on each transmission. Some time is wasted by having the receiver broadcast packet information, but this is far less than the normal routing schemes, which can retransmit when an acknowledge message is lost. Since no mechanism for congestion control is used, retransmission is needed which leads to energy loss.

Drawbacks of Existing System

- **Energy consumption for routing is high.**

  The energy consumption for routing is high in the existing system since the nearby nodes can overhear the data transmission due to the wireless broadcast advantage (WBA). The energy needed to find an optimum path and to send the data packet is very high.

- **Packet loss is not considered.**

  In the existing system while transmitting data packets performance bottle neck may occur due to high data transmission rate and low window size. This may lead to packet loss or buffer overflow in the network. Also there is no proper mechanism for controlling this bottleneck.

- **Congestion can occur any time, cannot identify it.**

  Since there is no specific mechanism for congestion avoidance or congestion control the congestion or packet loss cannot be identified. Thus the existing system never consider the packet loss in data transmission.

- **Minimum network throughput.**

  The throughput of network is very low, since the energy consumption for routing is very high in the WBA concept and also congestion can occur in the network any time due to lack of proper congestion control mechanism. The traditional method of routing will lead to overhearing of packet transmission and cause high energy consumption. The packet loss and its retransmission minimize the network throughput.

2.3 Proposed System

The proposed system uses a method of prioritizing the forwarder list to reduce the energy consumption in routing the packets. In the traditional routing mechanism we use to transmit the packet to all its neighbouring nodes. The node which receives the packet successfully will act as the source node and route the packet to the target. This mechanism leads to the overhearing or increase the number of cluster heading. This increases the transmission energy consumption. Here we propose a system which uses some forwarder list of nodes in a prioritizing manner. By this method we can reduce the overhearing and thus the transmission energy can be reduced by avoiding the packet transmission to unwanted nodes. In the proposed system we use a routing algorithm which select and prioritize a forwarder list through which the transmission of data packet is done. This steadily decreases the transmission energy requirement. And suppose if the low cost path fail in transmission the next low cost path is selected and packet routing is done through this path with no delay.

In the proposed system we also introduce a separate mechanism for congestion control. Thus in the proposed system congestion is avoided. In addition collision and packet loss are also avoided. Since congestion is avoided we reduce the number of packet retransmission which leads to the high energy consumption. In this system we couple a congestion control algorithm with a routing algorithm. So that after finding an optimal low cost path for packet data transmission, congestion control is also done over this route. The congestion control mechanism helps to avoid the packet loss and packet collision while transmission.

We use Bellman-Ford algorithm to route the packets through the nodes in the network. For congestion control we proposed an algorithm (NRDC) which minimize the packet loss and increase the throughput.

The major advantage of our proposed system is that the network TCP throughput is increased by routing the packet energy efficiently with no packet collision or congestion.

Advantages of Proposed System

- **Avoid packet overhearing.**

  The existing system uses a fixed forwarder list to transmit data packet instead of transmitting the data packet to all the neighboring nodes. This cause cluster heading and increase the energy consumption for transmission. The proposed efficient routing avoids the overhearing of data packet by sending the packet only to the nodes in the selected low cost path.

- **Minimize energy cost.**

  The use of efficient routing algorithm and congestion control algorithm leads to minimize the energy cost. The routing algorithm helps to reduce the overhearing of packet and help to avoid forwarding the packet to unwanted nodes. This reduces the energy cost for routing. The congestion control algorithm helps to avoid the packet loss and its retransmission which require high energy.

- **Collision detection & avoidance.**

  The use of special mechanism for congestion control with the routing algorithm helps to avoid packet collision and packet loss. Thus the need for retransmission of lost packet can be avoided. The congestion algorithm will work over the selected route and avoid the occurrence of congestion in transmission of packet.

- **Maximize network TCP throughput.**

  By the coupling of a congestion control algorithm over the routing algorithm we maximized the network throughput. The congestion avoidance helps to avoid the packet loss and thus the retransmission is not needed. This maximizes the efficiency of the network. The routing algorithm makes the transmission fast and congestion free network is achieved.

2.3 Feasibility Study

A feasibility study is a preliminary study undertaken to determine and document a project's viability. The term feasibility study is also used to refer to the resulting document. The results of this study are used to make a decision whether to proceed with the project, or to leave it.

Economic Feasibility

This study is conducted to check the economic feasibility of the proposed system. We use NS2 and nam for the design and simulation of the proposed system. Since the software used for the development of the proposed system are all open source and free of cost, our project is economically feasible. Each and every tool for the development are available free of cost, so the expenditure for the development and simulation of the proposed system is low.

Operational Feasibility

The improved TCP congestion free routing protocol in wired cum wireless network which we develop can be easily implemented in a network for energy efficient routing of data packets. The simulation result which we got for the system shows that it is highly energy efficient and congestion free. The coupling of routing algorithm and congestion control algorithm is feasible in NS2.

Technical Feasibility

Our system is designed and simulated in NS2. The technique which we plan to use in our system is that to find out a route to send packet in an energy efficient manner by avoiding the packet overhearing and to apply the congestion control algorithm over this energy efficient route. From the preliminary feasibility study itself we find that the coupling of a congestion control algorithm over the routing algorithm is easily possible in the NS2. The simulation and performance analysis is easy in NS2. To analyze and compare the performance of different algorithm with the proposed algorithm can be very easily done in NS2 with the help of the x-graph feature available in the NS2.

CHAPTER 3

System Specification

3.1 Hardware Requirements

Processor : Core 2duo/dual core/AMD athelon

RAM : 1GB

Monitor : 15'' color

Keyboard : Standard 102 keys

Mouse : 3 buttons

Hard Disk : 120GB

3.2 Software Requirements

Operating System : Linux trisquel

Software : Network Simulator (allinone 2.35)

CHAPTER 4

Software Description

4.1 Network Simulator

NS is a discrete event simulator targeted at networking research. Ns provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. The ns-2 was built in C++ and provides a simulation interface through OTcl, an object-oriented dialect of Tcl. The user describes a network topology by writing OTcl scripts, and then the main ns-2 program simulates that topology with specified parameters. It runs on Linux, FreeBSD, Solaris, Mac OS X and on Windows using Cygwin. It is licensed for use under version 2 of the GNU General Public License. The simulation and performance analysis is easy in NS2. To analyze and compare the performance of different algorithm with our algorithm can be very easily done in NS2 with the help of the x-graph feature available in the NS2.

NS2 is an open-source simulation tool that runs on Linux. It is a discreet event simulator targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP, RTP and SRM over wired and wireless (local and satellite) networks. It has many advantages that make it a useful tool, such as support for multiple protocols and the capability of graphically detailing network traffic. Additionally, NS2 supports several algorithms in routing and queuing. LAN routing and broadcasts are part of routing algorithms. Queuing algorithms include fair queuing, deficit round-robin and FIFO.

The network simulator is a package of tools that simulates the behavior of networks. It create network topologies, log events that happen under any load and analyze events to understand the behavior such as queuing delay, congestion and packet loss.

Features

- Protocols: TCP, UDP, HTTP, Routing algorithms etc.
- Traffic Models: CBR, VBR, Web etc.
- Radio propagation, Mobility models.
- Energy Models.

- Topology Generation tools.
- Visualization tools.
- Extensibility.

Components of Network Simulator

- ns-2
- nam
- xgraph
- Tcl/Tk
- otcl
- perl
- Tcl/CL
- Tcl-debug

Motivation for Simulations

- Cheap does not require costly equipment
- Complex scenarios can be easily tested
- Results can be quickly obtained – more ideas can be tested in a smaller timeframe
- The real thing isn't yet available
- Controlled experimental conditions – Repeatability helps aid debugging
- Disadvantages: Real systems too complex to model.

TCL

- Variables: set x 1
  set y $x
- Arrays: set a(0) 1
- Printing: puts "$a(0) \n"
- Arithmetic Expression: set z = [expr $y + 5]
- Control Structures: if {$z == 6} then { puts "Correct!"}
  for {set i =0} {$i < 5} {incr i }{
  puts "$i * $i equals [expr $i * $i]"

```
}
```

- Procedures:
```
proc sum {a b} {
return [expr $a + $b]
}
```

Creating Event Scheduler

- Create event scheduler: set ns [new simulator]
- Schedule an event: $ns at <time> <event> – event is any legitimate ns/tcl function
- Start Scheduler

```
$ns at 5.0 “finish”
$ns run
proc finish {} {
global ns nf
close $nf
exec nam out.nam &
exit 0
}
```

Tracing

- All packet trace

```
$ns traceall[open out.tr w]
<event> <time> <from> <to> <pkt> <size>
<flowid> <src> <dst> <seqno> <aseqno>
+ 0.51 0 1 cbr 500 0 0.0 1.0 0 2
- 0.51 0 1 cbr 500 0 0.0 1.0 0 2
r 0.514 0 1 cbr 500 0 0.0 1.0 0 0
```

- Variable trace

```
set par [open output/param.tr w]
$tcp attach $par
$tcp trace cwnd_
$tcp trace maxseq_
$tcp trace rtt_
```

Tracing and Animation

- Network Animator

```
set nf [open out.nam w]
$ns namtraceall
$nf
proc finish {} {
global ns nf
close $nf
exec nam out.nam &
exit 0
}
```

Creating topology

- Two nodes connected by a link
- Creating nodes

```
set n0 [$ns node]
set n1 [$ns node]
```

- Creating link between nodes

```
$ns <link_type> $n0 $n1 <bandwidth> <delay> <queuetype>
$ns duplexlink $n0 $n1 1Mb 10ms DropTail
```

Creating TCP Connections

- Create TCP agent and attach it to the node

```
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
```

- Create a Null Agent and attach it to the node

```
set null0 [new Agent/TCPSink]
$ns attach-agent $n1 $null0
```

- Connect the agents

```
$ns connect $tcp0 $null0
```

CHAPTER 5

Project Description

5.1 Problem Definition

The minimum network throughput and energy consumption is one of the most important problems in the transmission of data packet in a network system. The traditional routing algorithm use WBA concept to route a packet from source to destination. Here in this method a data packet is needed to transmit over each and every nearby node even if these nodes are not taking part in the routing. This process of sending data packet to nodes which are not present in the forwarder list will lead to high routing energy conception.

The state of art of the network congestion shows that it is a very difficult problem because there is no way to determine the network condition. The congestion occurs when there is a lot of traffic in the network. Rapidly increasing bandwidths and great variety of software applications have created a recognized need for increased attention to TCP congestion control mechanism. The existing system does not consider about the packet loss and collision. It does not consist of any congestion control algorithm. This result in excessive packet loss and this leads to the necessity for retransmission of these lost packets. In most of the existing congestion control algorithms the increasing of congestion window linearly with time increase the bandwidth of the TCP connection and when the congestion is detected, the window size is multiplicatively reduced by a factor of two. This steadily decreases the throughput of the TCP connection.

The existing system uses a fixed forwarder list to route a data packet from source to the destination. By using a proper and efficient routing mechanism we can minimize the energy for routing. And at the same time with the help of a congestion control algorithm we can overcome the packet loss and packet collision. The following are the major problem in the existing system:

- Excessive queuing delays.

When the sender transmits packets in full line and the receiver cannot hold the entire packets, congestion occurs and it leads to delay in packet sending. The delayed packet forms a queue in the network. Since the loss notification is improper in the existing system, the

retransmitted packets will again enter in the queue. the queuing delay (or queuing delay) is the time a job waits in a queue until it can be executed. It is a key component of network delay.

This term is most often used in reference to routers. When packets arrive at a router, they have to be processed and transmitted. A router can only process one packet at a time. If packets arrive faster than the router can process them (such as in a burst transmission) the router puts them into the queue (also called the buffer) until it can get around to transmitting them.The maximum queuing delay is proportional to buffer size. The longer the line of packets waiting to be transmitted, the longer the average waiting time is; and when the buffer fills the router must drop packets.

- Energy loss due to retransmission.

Since there is no mechanism for congestion control, packet loss will occur and it is necessary to retransmit the lost packet. When this occurs in high rate a fair amount of energy is needed to retransmit the packets to the specified destinations. By specifying a congestion free route by prioritize forwarder list we can overcome the need of packet retransmission. Packet loss can be caused by a number of factors including signal degradation over the network medium due to multi-path fading, packet drop because of channel congestion, corrupted packets rejected in-transit, faulty networking hardware, faulty network drivers or normal routing routines. Thus in this case retransmission is essential.

- To find out the optimum forwarder list by low energy consumption.

The traditional method of routing by Wireless Broadcast Advantage (WBA) is to send a packet to all its adjacent nodes and the one which got the packet successfully will act as the source node and route the packet to the target node. By finding out an optimum forwarder list of nodes we can effectively reduce this energy for packet transmission to all its adjacent nodes. While considering an optimum forwarder list we should consider not only the expected cost but also the increment cost by adding n node in the list.

5.2 Overview of the Project

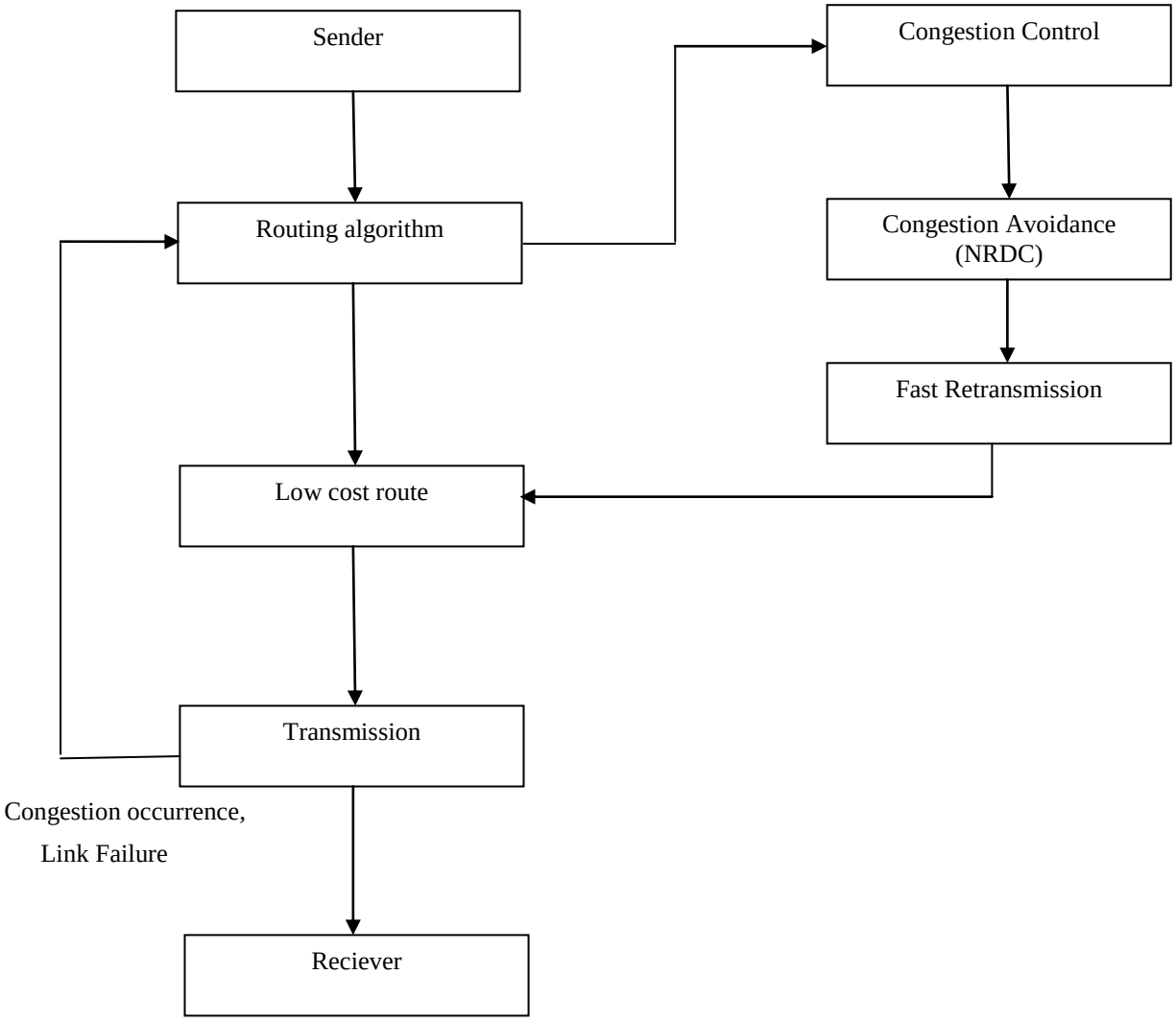


Fig.5.2.1 System Architecture

In this project sender sends the packets to the destination through a low cost route found out with the help of a routing algorithm. The mechanism of congestion control helps to avoid the packet loss and packet congestion in the network. The congestion control algorithm which we use for this purpose is the NRDC congestion control algorithm. The NRDC algorithm avoids the possibility of packet loss and congestion by fast retransmission technique. Thus the packet is transmitted to the destination with minimum packet loss and high throughput. While transmission if a link failure occurs the routing protocols automatically selects the next low cost route and perform the above congestion control technique for the transmission of the data packets.

5.3 Module Description

Modules

1. Experimental setup
2. Routing
3. Congestion Control
4. Result

Experimental Setup

Experiment 1

The wired network topology consist of eight nodes. Each node is connected with neighbouring nodes with 100MB bandwidth and 20ms delay duplex wired connection. The drop tail queue and distance vector routing algorithm are used. Here we conduct many tests to analyze and compare the performance of NRDC algorithm with other existing congestion control algorithms BIC, ILLINOIS in the wired networks.

Experiment 2

Network topology consists of two wired nodes, one router, one base station, two mobile nodes. First two wired nodes are connected with a router with 100MB bandwidth and 20 ms delay duplex wired connection. The router is connected with a base station 10mb and 200 ms delay duplex wired connection. two nodes and base stations are connected with a wireless duplex links..Senders send number of flows towards corresponding receiver which the travels the bottleneck router and base station in between. Here base station which uses routing protocol DSDV to route the packets to its correct receiver. Traffic use FTP. A lot of experiments are carried out in the topology to test the performance of TCP with efficient routing and NRDC congestion control mechanism. The results are compared with existing TCP such as BIC and ILLINOIS using NS2.

Routing

Routing is the process of selecting paths in a network along which to send network traffic. The routing protocol which we use bellman-ford algorithm. Bellman–Ford is in its basic structure very similar to Dijkstra's algorithm, but instead of greedily selecting the minimum-weight node not yet processed to relax, it simply relaxes all the edges, and does this $|V| - 1$ times, where $|V|$ is the number of vertices in the graph. The repetitions allow minimum distances to accurately propagate throughout the graph, since, in the absence of negative cycles, the shortest path can only visit each node at most once. Unlike the greedy approach, which depends on certain structural assumptions derived from positive weights, this straightforward approach extends to the general case.

The working procedure for bellman-ford algorithm is as follows.

```
Step 1: initialize graph
  for each vertex v in vertices:
     if v is source then v.distance := 0
     else v.distance := infinity
        v.predecessor := null

Step 2: relax edges repeatedly
  for i from 1 to size(vertices)-1:
     for each edge uv in edges: // uv is the edge from u to v
        u := uv.source
        v := uv.destination
        if u.distance + uv.weight < v.distance:
           v.distance := u.distance + uv.weight
           v.predecessor := u

Step 3: check for negative-weight cycles
    for each edge uv in edges:
     u := uv.source
     v := uv.destination
     if u.distance + uv.weight < v.distance:
        error "Graph contains a negative-weight cycle"
```

A distributed variant of the Bellman – Ford algorithm is used in distance-vector routing protocols, for example the Routing Information Protocol (RIP). The algorithm is distributed because it involves a number of nodes (routers) within an Autonomous system, a collection of IP networks typically owned by an ISP. It consists of the following steps:

- Each node calculates the distances between itself and all other nodes within the AS and stores this information as a table.
- Each node sends its table to all neighbouring nodes.
- When a node receives distance tables from its neighbours, it calculates the shortest routes to all other nodes and updates its own table to reflect any changes.

DSDV [12] is an adaptation of classical distance vector routing protocol to ad hoc networks. In DSDV, two routing tables are maintained at each of the nodes. One of them is the routing table, which contains a complete list of addresses of all other nodes in the network. The other contains the setting time data for each destination node. It is used to determine the time for update advertisement. The routing of updates and packets between nodes is based on these tables. Along with each node's address, the routing table contains the address of next hop, route metric, destination sequence number, etc. Route updates are broadcasted periodically or scheduled as needed in the network. Routes are always selected with later sequence number. If the sequence numbers are identical, the route with smallest metric will be selected. These criteria guarantee loop-free routes, but they also induce routes fluctuation. DSDV is a bi-directional protocol, which unavoidably has the unidirectional links problem in ad hoc networks.

Congestion control

A congestion control mechanism is given to the efficient low cost routing path which is find out by the routing algorithm. The congestion control algorithm which we use in this system is the NRDC algorithm, which is based on the additive increase and multiplicative decrease algorithm. The size of the congestion window is increased steadily when no congestion is occurred. The approach taken is to increase the transmission rate (window size), probing for usable bandwidth, until loss occurs. The policy of additive increase may, for instance, increase the congestion window by a fixed amount every round trip time. When congestion is detected, the transmitter decreases the transmission rate by a multiplicative factor; for example, cut the congestion window in half after loss. The result is a saw-tooth

behaviour that represents the probe for bandwidth. Frequently, packet loss serves as the signal; the multiplicative decrease is triggered when a timeout or acknowledgement message indicates a packet was lost.

The modified mechanism which we use for the additive increase of the congestion window size is the Newton Raphson method. The existing equation for the additive increase of congestion window size is replaced by the modified Newton Raphson method. The additive increase/multiplicative-decrease (AIMD) algorithm is a feedback control algorithm best known for its use in TCP Congestion Avoidance. AIMD combines linear growth of the congestion window with an exponential reduction when congestion takes place.

The procedure of NRDC algorithm is as following:

Adjust sending rate according to an additive increase/multiplicative decrease algorithm

- Start slowly, increase gradually to find equilibrium
- Add a small amount to the sending speed each time interval without loss
- For a window-based algorithm $W_{i+1} = W_i(\alpha - (2*W_i))$ each RTT, where $\alpha > 1$ typically
- Respond to congestion rapidly
- Divide sending window by some factor $\beta=2$ each interval loss seen
- For a window-based algorithm $W_i = W_i / \beta$ each RTT, where $\beta = 2$ typically.

Result: TCP Throughput

A lot of experiments on wired network topology and wired cum wireless network topology are carried out to test performance of NRDC–TCP with existing tcp variants BIC and ILLINOIS with efficient routing protocol DVR and DSDV respectively using NS2. From xgraph a better performance of NRDC-TCP agent is observed and gives better performance in the value of mean throughput obtained after a simulation of 50 milliseconds.

5.4 Data Flow Diagram

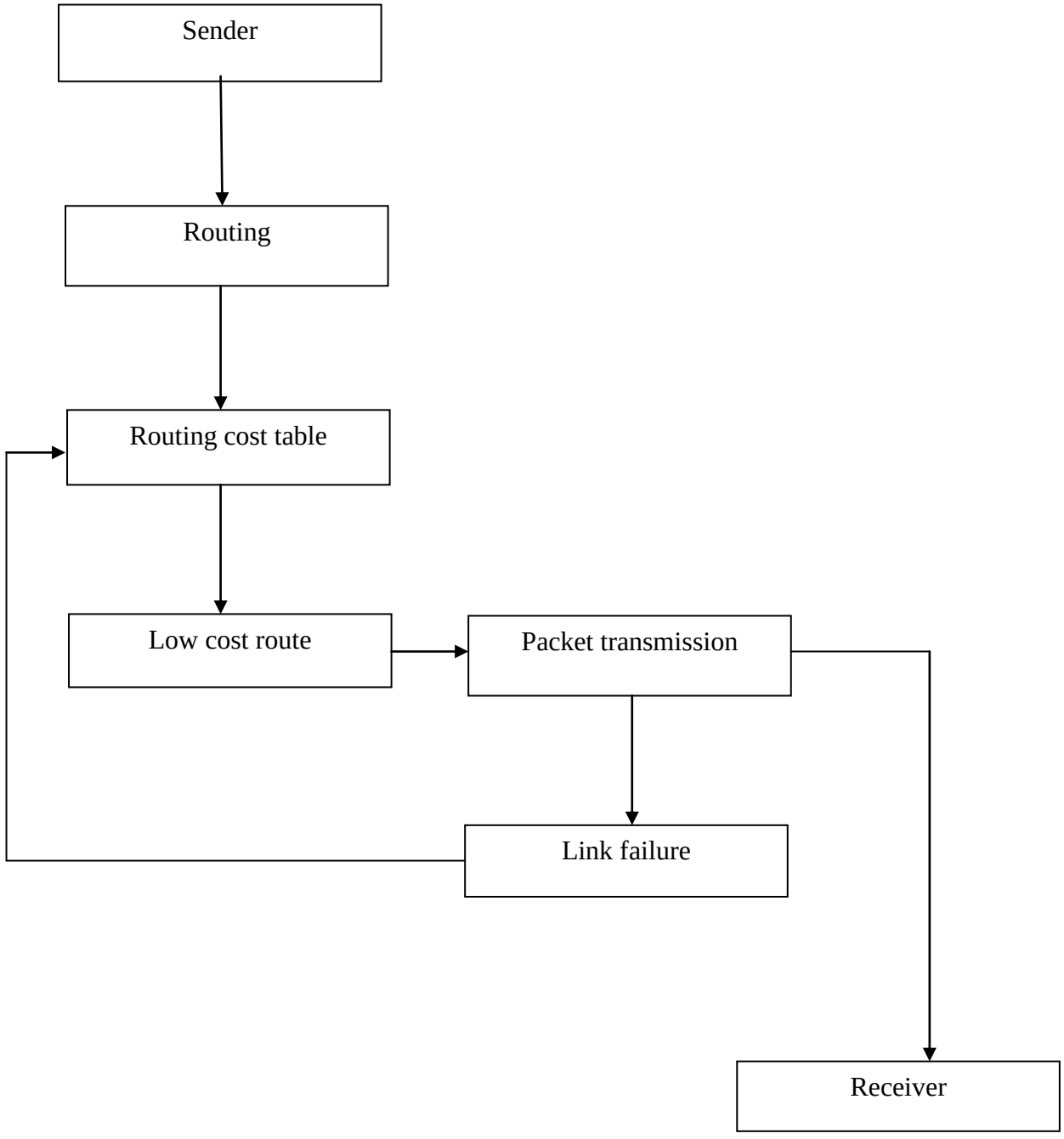


Fig. 5.4.1 Routing Algorithm

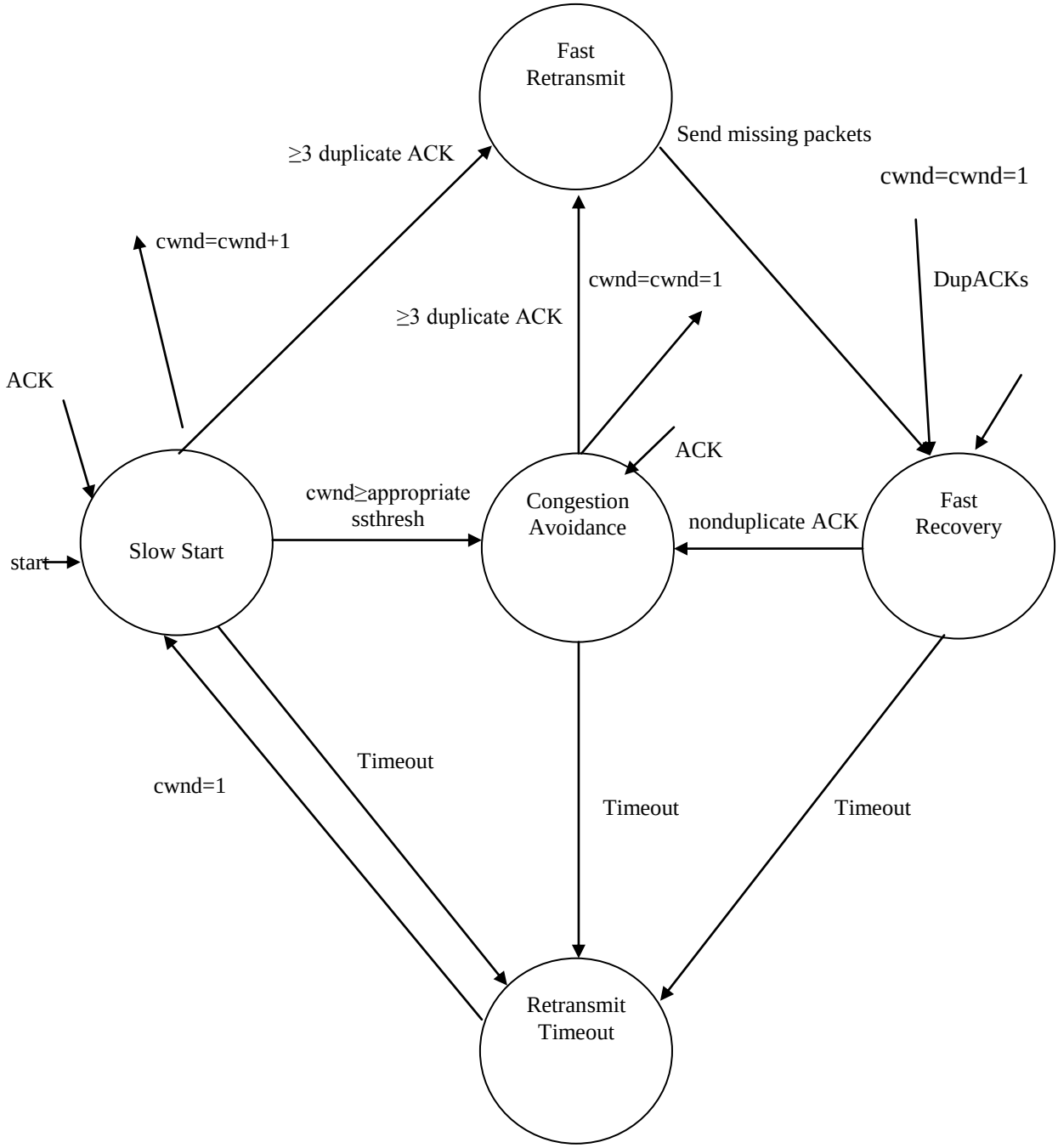


Fig. 5.4.2 NRDC Algorithm

5.5 Input Design

Input design is a part of overall system design. The main objective during the input design is given below. To provide a cost effective method of input. To achieve the highest possible level of accuracy. To ensure that the input is acceptable and understood by the system. In this project the input design is made to its maximum level by specifying the maximum packet size, window size and the type of queue for each node individually. And also the traffic start and stop time with link failure for a small period of time is done.

5.6 Output Design

Output design generally refers to the results and information that are generated by the system; output is the main reason for developing the system and the basis on which they evaluate the usefulness of the system in the network. The output consist of a simulation of the network, performance analysis of the designed algorithm with other existing algorithms to analyst the efficiency and throughput.

CHAPTER 6

System Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.

Each module in the proposed system is tested as a single unit of module to ensure the correctness of the system.

6.2 Integration Testing

The integration testing involves the testing of the order in which the different modules are combined to produce the functioning whole. Integration testing generally through light on the order of arrangement of units, modules such as routing and congestion control in this project.

The proposed system consists of four modules. All these modules are tested individually and now it is integrated together to check for any errors in the system.

6.3 Test Cases

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not.

In each test case, you'll have a list of test items or functions or features that you want to evaluate. Each test item should include not just an action, but the success criteria, and if you want to get more sophisticated, criticality. For example, you might want to make sure a business-critical application still works after a network change. So you'd arrange to have the application owners create a transaction or operate the application. Along with that, you'd make sure they describe a successful transaction prior to execution so there's no question about whether or not it works. And you also want to know if the test fails, if it's important enough to cause your whole change to fail and be backed out. This is because sometimes

your change might be more important, and there might be a workaround for the problem.

In the proposed system we should check the system with different TCP agents at the nodes. We also change the size of data flow, window size. We analyse the performance of system with different types of queue.

CHAPTER 7

System Implementation

Topology Initialization

We can design a topology with required number of nodes in the Network Animator (NAM). Topology can be designed using the tools available in the NAM by just clicking required tools with mouse. The links between the nodes and the proper agents for each links can also be defined in this process.

System Execution

The simulation of the system is executed by few commands on the linux terminal. First we should navigate to the location where the nam and ns file is stored. For this we can use the command cd directoryname. The command ns filename.ns can be used to view the simulation of the network system.

Including the routing algorithm

Open the ns file from the location and including the following command after the node initialization, we can include a routing algorithm to find out the optimum low cost route in the network.

```
$ns rtproto DV
```

Including the congestion control algorithm

Open the ns file from the location and including the following command after the node initialization, we can include the congestion control algorithm to reduce the packet loss and to increase the network throughput.

```
$ns at 0 "$tcp1 select_ca NRDC"
```

CHAPTER 8

Conclusion & Future Enhancements

8.1 Conclusion

We have designed a system for increasing the network throughput and to reduce the packet loss in a TCP network. This system can be implemented in any TCP networks to increase the overall performance and the throughput of the network transmission. The simulation result of our system clearly shows that it is an efficient and elegant system to reduce the packet loss and the network congestion. This system gives a high level of throughput in the transmission of data packet in the TCP network system. This system uses the combination of two algorithms, one for routing and the other for congestion control, in order to perform congestion check and congestion control over the efficient low cost route which we find out by using the routing algorithm. The NRDC algorithm which we established in this system finds out efficient route and applies congestion avoidance over that route.

The proposed system performs the task of controlling the congestion window size by considering round trip time and acknowledgement for the packet before the timeout. Thus we get an efficient system for the transmission of data packet in a TCP network system with low routing cost and less packet loss with maximum throughput. The comparison of our system with other few existing system show its performance efficiency.

8.2 Future Enhancements

The future enhancement of our project is that by improving the equations for increasing and decreasing the congestion window size, we can reduce the packet loss. Our system shows a packet loss of 8%. We use the simplified Newton Raphson method's relation for the additive increasing of the congestion window size for each round trip time. In this system we derive new equation only to control the additive increasing of the size of the congestion window. But we never consider about the equation for multiplicative decrease of the size of the congestion window. By improving the equation for controlling the size of congestion window we can improve the throughput some more and to reduce the cause of data packet loss. In future we will apply NRDC-TCP protocol in wireless networks, MANETs, wireless sensor networks, satellite networks, multicast routing networks.

CHAPTER 9

Appendix

9.1 Sample Code

```
# Create a new simulator object.
set ns [new Simulator]
# Create a nam trace datafile.
set namfile [open ppp.nam w]
$ns namtrace-all $namfile
set f0 [open dvrc.thro w]
set f1 [open dvr.thro w]
# Create wired nodes.
set node(8) [$ns node]
## node(8) at 601.859375,575.906250
$node(8) set X_ 601.859375
$node(8) set Y_ 575.906250
$node(8) set Z_ 0.0
$node(8) color "black"

set node(7) [$ns node]
## node(7) at 577.453125,575.562500
$node(7) set X_ 577.453125
$node(7) set Y_ 575.562500
$node(7) set Z_ 0.0
$node(7) color "black"

set node(6) [$ns node]
## node(6) at 562.671875,555.281250
$node(6) set X_ 562.671875
$node(6) set Y_ 555.281250
$node(6) set Z_ 0.0
$node(6) color "black"
```

```
set node(5) [$ns node]
## node(5) at 571.953125,594.812500
$node(5) set X_ 571.953125
$node(5) set Y_ 594.812500
$node(5) set Z_ 0.0
$node(5) color "black"

set node(4) [$ns node]
## node(4) at 537.578125,578.656250
$node(4) set X_ 537.578125
$node(4) set Y_ 578.656250
$node(4) set Z_ 0.0
$node(4) color "black"

set node(3) [$ns node]
## node(3) at 512.484375,563.187500
$node(3) set X_ 512.484375
$node(3) set Y_ 563.187500
$node(3) set Z_ 0.0
$node(3) color "black"

set node(2) [$ns node]
## node(2) at 530.015625,600.656250
$node(2) set X_ 530.015625
$node(2) set Y_ 600.656250
$node(2) set Z_ 0.0
$node(2) color "black"

set node(1) [$ns node]
## node(1) at 488.078094,588.281250
$node(1) set X_ 488.078094
$node(1) set Y_ 588.281250
$node(1) set Z_ 0.0
```

```
# wireless3.tcl

# simulation of a wired-cum-wireless topology running with mobileIP
#=====================================================================
# Define options
# =====================================================================
set opt(chan)   Channel/WirelessChannel        ;# channel type
set opt(prop)   Propagation/TwoRayGround       ;# radio-propagation model
set opt(netif)  Phy/WirelessPhy                ;# network interface type
set opt(mac)    Mac/802_11                     ;# MAC type
set opt(ifq)    Queue/DropTail/PriQueue        ;# interface queue type
set opt(ll)     LL                             ;# link layer type
set opt(ant)    Antenna/OmniAntenna            ;# antenna model
set opt(ifqlen)         50                     ;# max packet in ifq
set opt(nn)             6                      ;# number of mobilenodes

set opt(adhocRouting)   DSR                    ;# routing protocol
set opt(cp)     ""                             ;# cp file not used

set opt(sc)     ""                             ;# node movement file.
set opt(x)      15                             ;# x coordinate of topology
set opt(y)      15                             ;# y coordinate of topology
set opt(seed)   0.0                            ;# random seed
set opt(stop)   300                            ;# time to stop simulation
set opt(ftp1-start)      0.0
set opt(ftp2-start)      0.0
set num_wired_nodes      7
#set num_bs_nodes       2  ; this is not really used here.
#
=====================================================================
# check for boundary parameters and random seed
if { $opt(x) == 0 || $opt(y) == 0 } {
        puts "No X-Y boundary values given for wireless topology\n"
}
if {$opt(seed) > 0} {
```

```
        puts "Seeding Random number generator with $opt(seed)\n"
        ns-random $opt(seed)


}
# create simulator instance
set ns_   [new Simulator]
$ns_ color 1 Orange
$ns_ color 2 Blue
$ns_ color 3 Black
$ns_ color 4 Green
$ns_ color 5 Yellow
$ns_ color 6 RED
# set up for hierarchical routing
$ns_ node-config -addressType hierarchical
AddrParams set domain_num_ 2          ;# number of domains
lappend cluster_num 1 1             ;# number of clusters in each domain
AddrParams set cluster_num_ $cluster_num
lappend eilastlevel 7 7            ;# number of nodes in each cluster
AddrParams set nodes_num_ $eilastlevel ;# of each domain
set tracefd  [open wireless3-out.tr w]
set namtrace [open out.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)
set tr0 [open simple0.data w]
set tr1 [open simple1.data w]
set tr2 [open simple2.data w]
set tr3 [open simple3.data w]
set tr4 [open simple4.data w]
set tr5 [open simple5.data w]
set f0 [open nrdc.thro w]
set f1 [open aodv.thro w]
set f2 [open cuaodv.thro w]
set f3 [open htcp.thro w]
```

```
set f4 [open highspeed.thro w]
set f5 [open scalar.thro w]
# Create topography object
set topo   [new Topography]
# define topology
$topo load_flatgrid $opt(x) $opt(y)
# create God
#   2 for HA and FA
create-god [expr $opt(nn) + 2]
#create wired nodes
set temp {0.0.0 0.0.1 0.0.2 0.0.3 0.0.4 0.0.5 0.0.6 };# hierarchical addresses
for {set i 0} {$i < $num_wired_nodes} {incr i}

{
   set W($i) [$ns_ node [lindex $temp $i]]
}
# Configure for ForeignAgent and HomeAgent nodes
$ns_ node-config -mobileIP ON \
                -adhocRouting $opt(adhocRouting) \
                -llType $opt(ll) \
                -macType $opt(mac) \
                -ifqType $opt(ifq) \
                -ifqLen $opt(ifqlen) \
                -antType $opt(ant) \
                -propType $opt(prop) \
                -phyType $opt(netif) \
                -channelType $opt(chan) \
                        -topoInstance $topo \
                -wiredRouting ON \
                        -agentTrace ON \

                -routerTrace OFF \
                -macTrace OFF
```

```
# Create HA and FA
set HA [$ns_ node 1.0.0]
#set FA [$ns_ node 2.0.0]
$HA random-motion 0
#$FA random-motion 0
# Position (fixed) for base-station nodes (HA & FA).
$HA set X_ 50.000000000000
$HA set Y_ 50.000000000000
$HA set Z_ 0.000000000000
#$FA set X_ 650.000000000000
#$FA set Y_ 600.000000000000
#$FA set Z_ 0.000000000000

# create a mobilenode that would be moving between HA and FA.
# note address of MH indicates its in the same domain as HA.
$ns_ node-config -wiredRouting OFF
#Mobile node 0
set MH0 [$ns_ node 1.0.1]
set node_(0) $MH0
set HAaddress0 [AddrParams addr2id [$HA node-addr]]
[$MH0 set regagent_] set home_agent_ $HAaddress0
# movement of the MH
$MH0 set Z_ 0.000000000000
$MH0 set Y_ 0.000000000000
$MH0 set X_ 120.000000000000
#Mobile node 1
set MH1 [$ns_ node 1.0.2]
set node_(1) $MH1
set HAaddress1 [AddrParams addr2id [$HA node-addr]]
[$MH1 set regagent_] set home_agent_ $HAaddress1
# movement of the MH
$MH1 set Z_ 0.000000000000
$MH1 set Y_ 0.000000000000
$MH1 set X_ 140.000000000000
```

```
set ftp5 [new Application/FTP]
$ftp5 attach-agent $tcp5
$ns_ at $opt(ftp1-start) "$ftp5 start"
$tcp5 set fid_ 6# source connection-pattern and node-movement scripts
if { $opt(cp) == "" } {
        puts "*** NOTE: no connection pattern specified."
set opt(cp) "none"

} else {

        puts "Loading connection pattern..."

        source $opt(cp)}

if { $opt(sc) == "" } {

        puts "*** NOTE: no scenario file specified."
set opt(sc) "none"

} else {
 puts "Loading scenario file..."
 source $opt(sc)
        puts "Load complete..."


}
# Define initial node position in nam
#Mobile node 2
set MH2 [$ns_ node 1.0.3]
set node_(2) $MH2
set HAaddress2 [AddrParams addr2id [$HA node-addr]]
[$MH2 set regagent_] set home_agent_ $HAaddress2
# movement of the MH
$MH2 set Z_ 0.000000000000
$MH2 set Y_ 15.000000000000
$MH2 set X_ 150.000000000000
#Mobile node 3
set MH3 [$ns_ node 1.0.4]
```

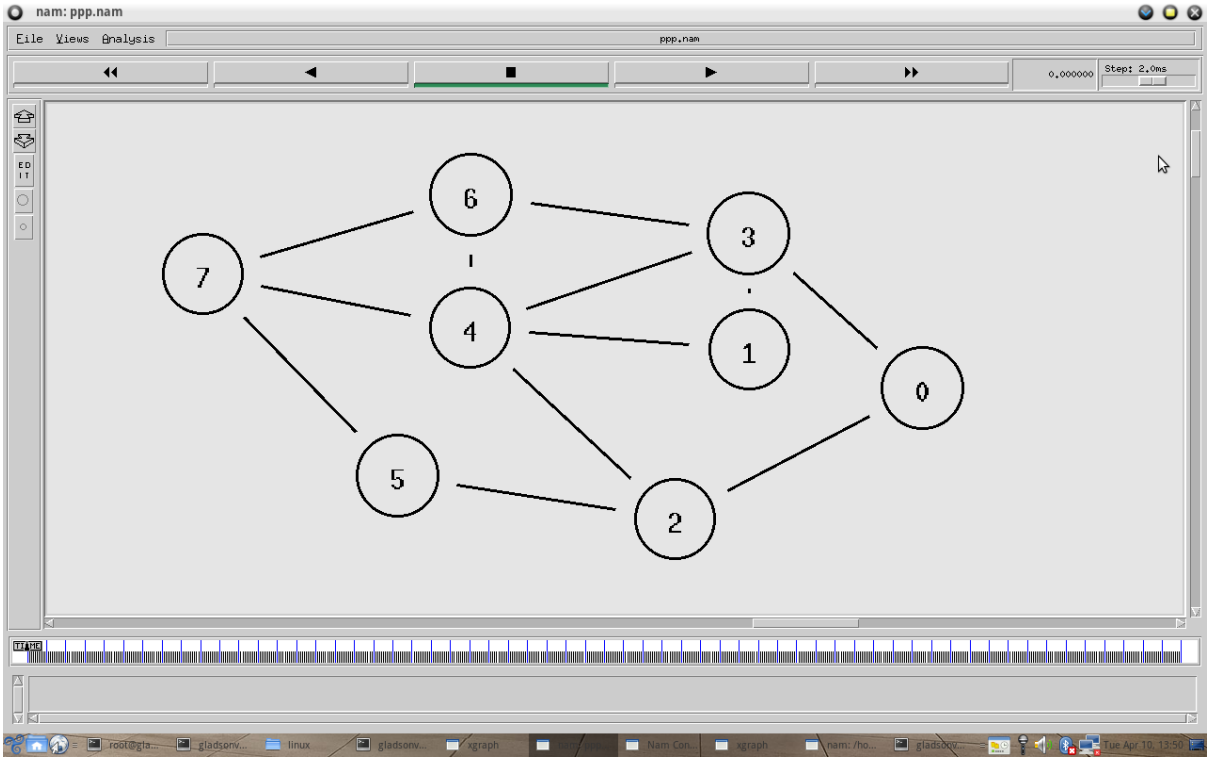
```
set node_(3) $MH3


set HAaddress3 [AddrParams addr2id [$HA node-addr]]
[$MH3 set regagent_] set home_agent_ $HAaddress3
# movement of the MH
$MH3 set Z_ 0.000000000000
$MH3 set Y_ 30.000000000000
$MH3 set X_ 150.000000000000
#Mobile node 4
set MH4 [$ns_ node 1.0.5]
set node_(4) $MH4
set HAaddress4 [AddrParams addr2id [$HA node-addr]]
[$MH4 set regagent_] set home_agent_ $HAaddress4
# movement of the MH
$MH4 set Z_ 0.000000000000
$MH4 set Y_ 40.000000000000
$MH4 set X_ 140.000000000000
$ns attach-agent $node(8) $sink1

$sink0 set packetSize_ 210

set tcp1 [new Agent/TCP/Linux]
$ns attach-agent $node(1) $tcp1
$tcp1 set timestamps_ true
$ns at 0 "$tcp1 select_ca NRDC"


$ns color 1 "Green"
$tcp1 set fid_ 2
$tcp1 set packetSize_ 1040
$tcp1 set window_ 1000
$tcp1 set windowInit_ 1
$tcp1 set maxcwnd_ 0


# Create traffic sources and add them to the agent.
```

```
set traffic_source(2) [new Application/FTP]
$traffic_source(2) attach-agent $tcp1
$traffic_source(2) set maxpkts_ 256

# Connect agents.
$ns connect $tcp1 $sink1

$ns rtproto DV

# Traffic Source actions.
$ns at 0.000000 "$traffic_source(1) start"
$ns at 60.000000 "$traffic_source(1) stop"
# Traffic Source actions.
$ns at 0.000000 "$traffic_source(2) start"
$ns at 60.000000 "$traffic_source(2) stop"
set last_ack0 0
set last_ack1 0


proc record {} {
    global ns tcp0 tcp1 sink1 sink0 f0 f1
    global last_ack0 last_ack1


  set now [$ns now]
    set time 0.5

   set c0 [$tcp0 set ack_]
   set d0 [$tcp0 set packetSize_]

  if { $c0 >0} {
        set e0 [expr $c0 - $last_ack0]
        puts $f0 "$now [expr $e0*$d0*8/$time/1000000]"
        set last_ack0 $c0
```

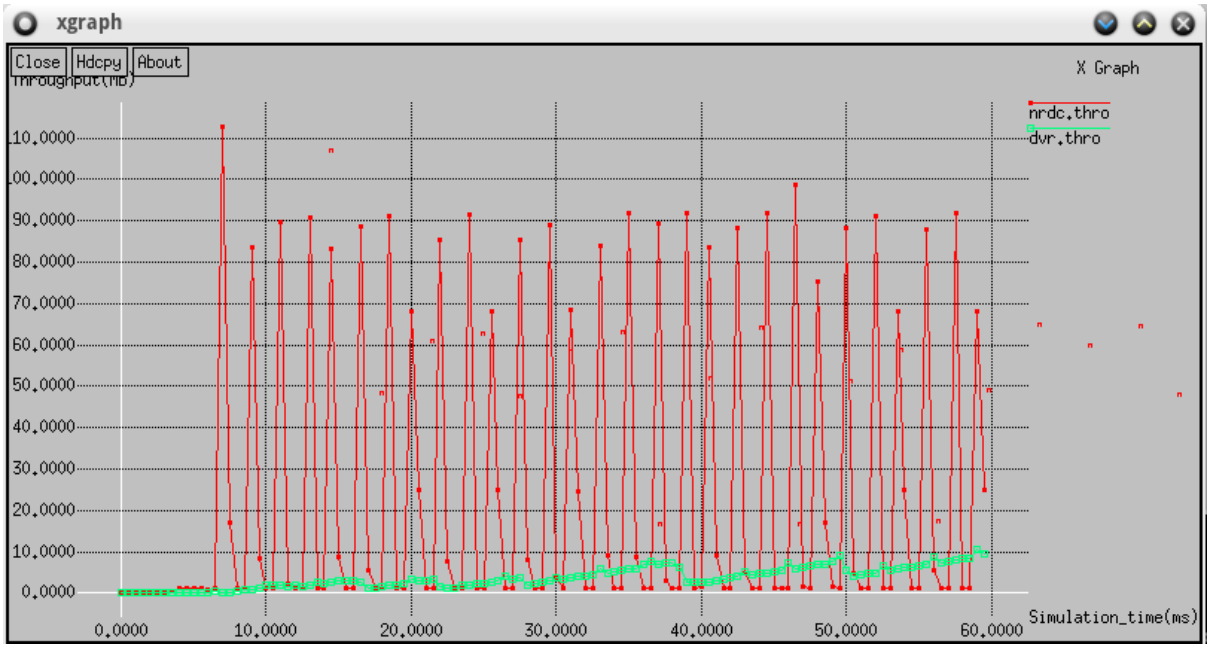
```
proc record {} {

  global ns_ tcp0 tcp1 tcp2 tcp3 tcp4 tcp5 sink0 sink1 sink2 sink3 sink4 sink5 tr0 tr1 tr2 tr3 tr4 tr5 f0
f1 f2 f3 f4 f5
global last_ack0 last_ack1 last_ack2 last_ack3 last_ack4 last_ack5
 set now [$ns_ now]

 set time 0.5
 set cwin0 [$tcp0 set cwnd_]
 set bytes0 [$sink0 set bytes_]

set nrexmitpack0 [$tcp0 set nrexmitpack_]
 set cwin1 [$tcp1 set cwnd_]
 set bytes1 [$sink1 set bytes_]
 set nrexmitpack1 [$tcp1 set nrexmitpack_]
 set cwin2 [$tcp2 set cwnd_]

 set bytes2 [$sink2 set bytes_]
set nrexmitpack2 [$tcp2 set nrexmitpack_]

 set cwin3 [$tcp3 set cwnd_]
 set bytes3 [$sink3 set bytes_]
set nrexmitpack3 [$tcp3 set nrexmitpack_]
set cwin4 [$tcp4 set cwnd_]
 set bytes4 [$sink4 set bytes_]
 set nrexmitpack4 [$tcp4 set nrexmitpack_]
set cwin5 [$tcp5 set cwnd_]
 set bytes5 [$sink5 set bytes_]
 set nrexmitpack5 [$tcp5 set nrexmitpack_]
 set c0 [$tcp0 set ack_]
 set d0 [$tcp0 set packetSize_]
 } else {

    puts $f0 "$now 0"
      }
  set c1 [$tcp1 set ack_]
  set d1 [$tcp1 set packetSize_]

 if { $c1 >0} {
```

```
        set e1 [expr $c1 - $last_ack1]
        puts $f1 "$now [expr $e1*$d1*8/$time/1000000]"
        set last_ack1 $c1
       } else {
        puts $f1 "$now 0"
       }

   $ns at [expr $now+$time] "record "
}

# Run the simulation
proc finish {} {
          global ns namfile f0 f1
          $ns flush-trace
          close $namfile
       close $f0
       close $f1
          exec nam -r 2000.000000us ppp.nam &
          exec xgraph dvrc.thro dvr.thro -m -x Simulation_time(ms) -y Throughput(Mb) -
geometry 800x400 &
          exit 0
          }
$ns at 0.0 "record"
$ns at 60.000000 "finish"
$ns run
```

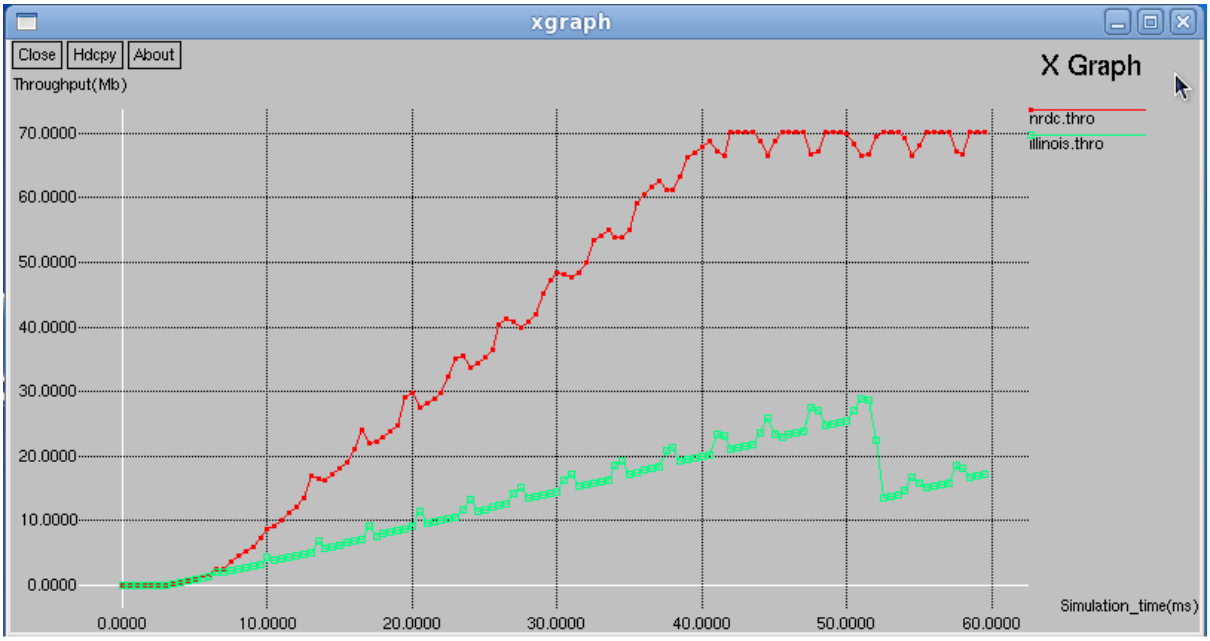
9.2 Screen Shot



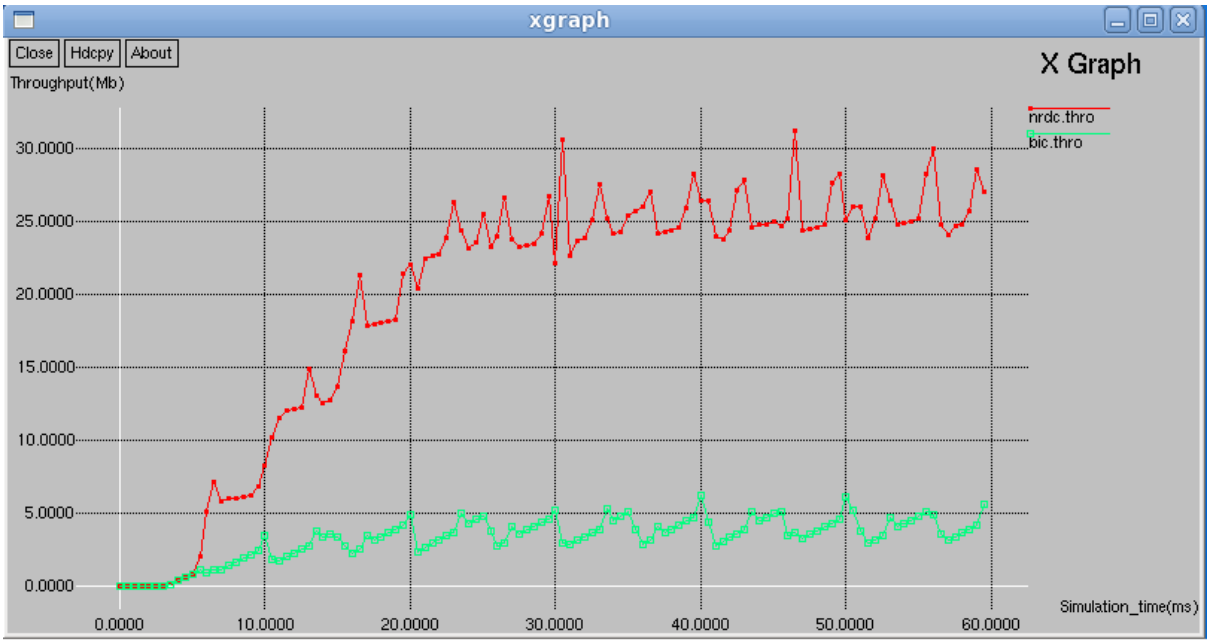
Wired Network Topology



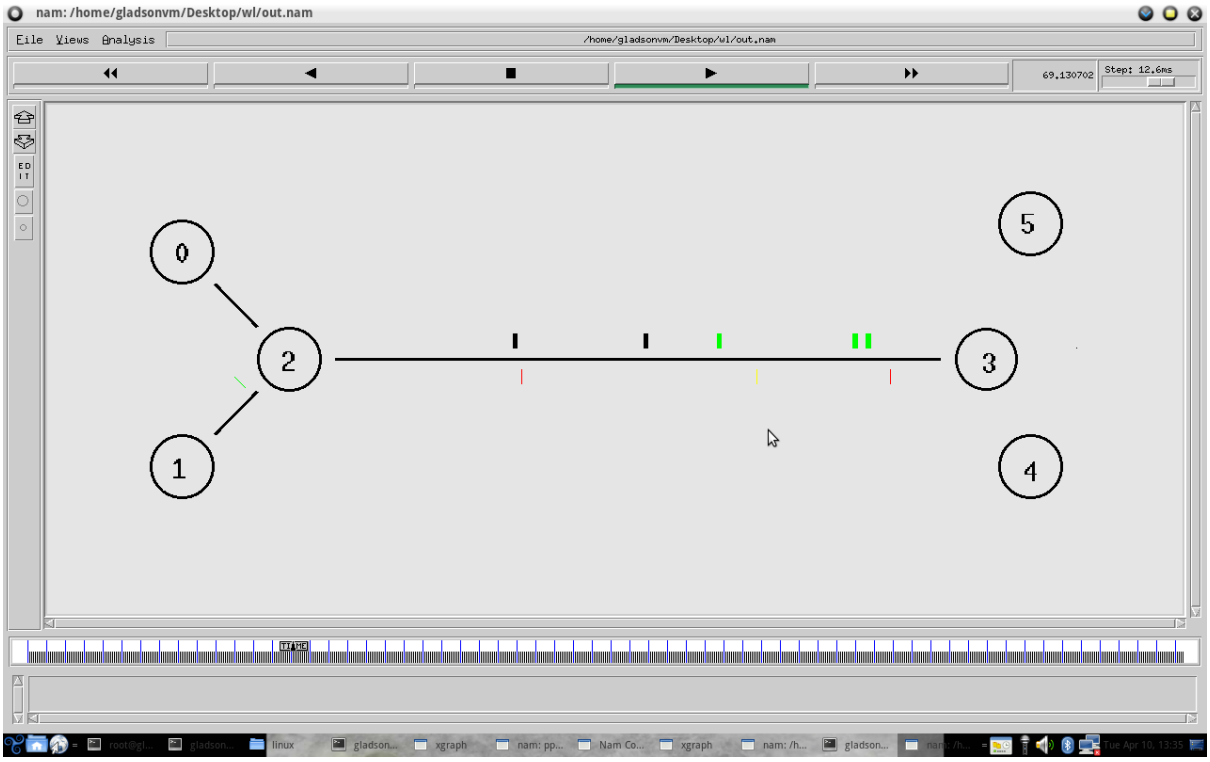
Comparison of NRDC with DVR



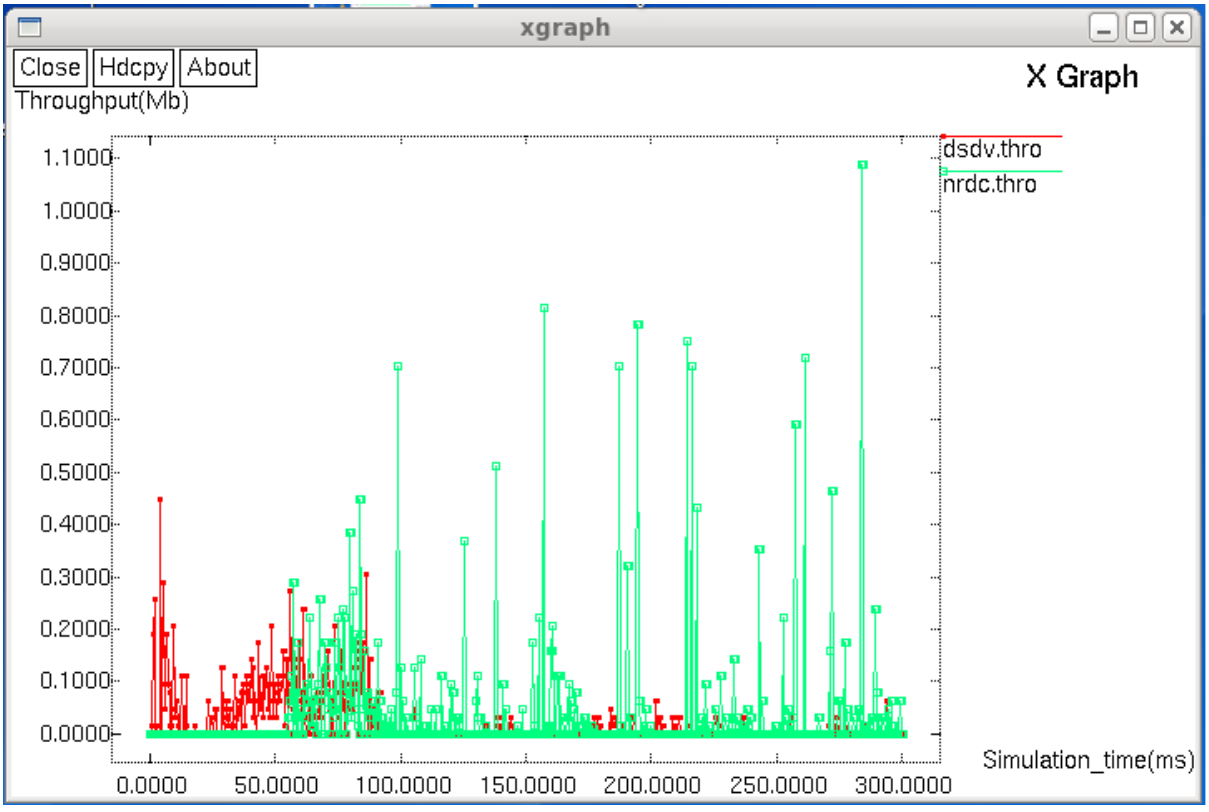
Comparison of NRDC with ILLINOIS



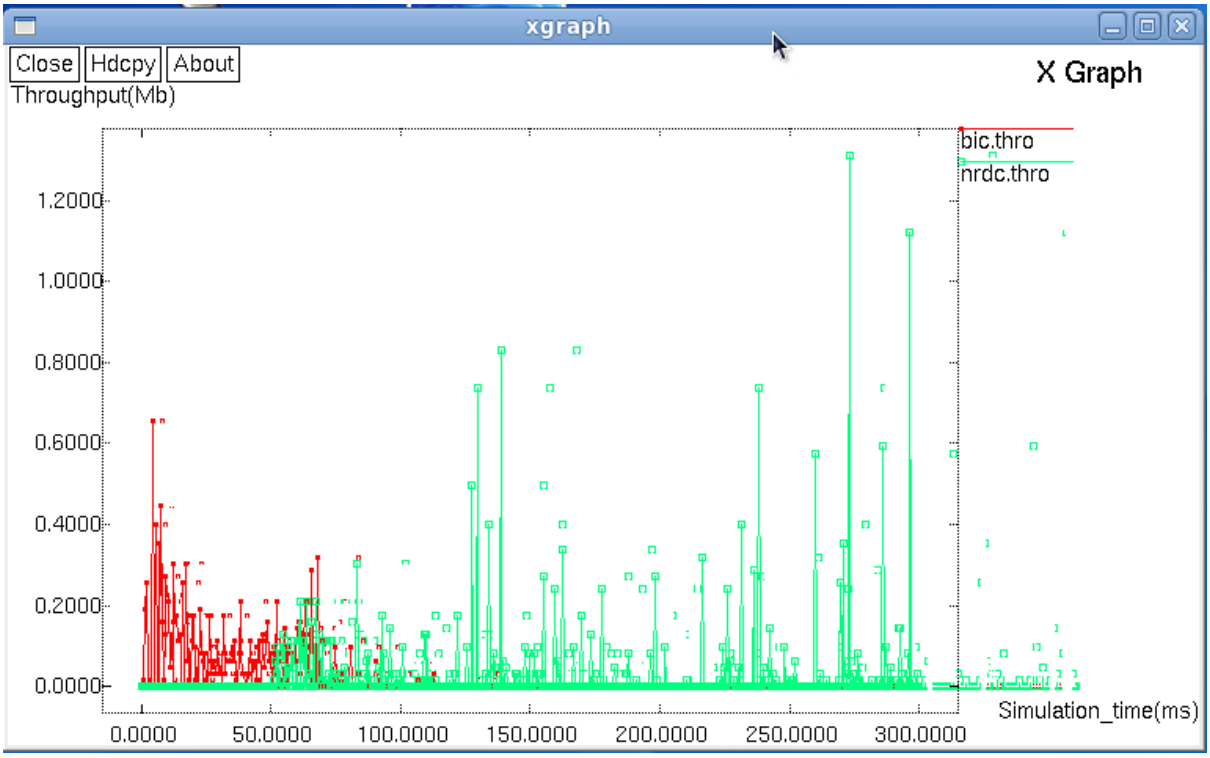
Comparison of NRDC with BIC



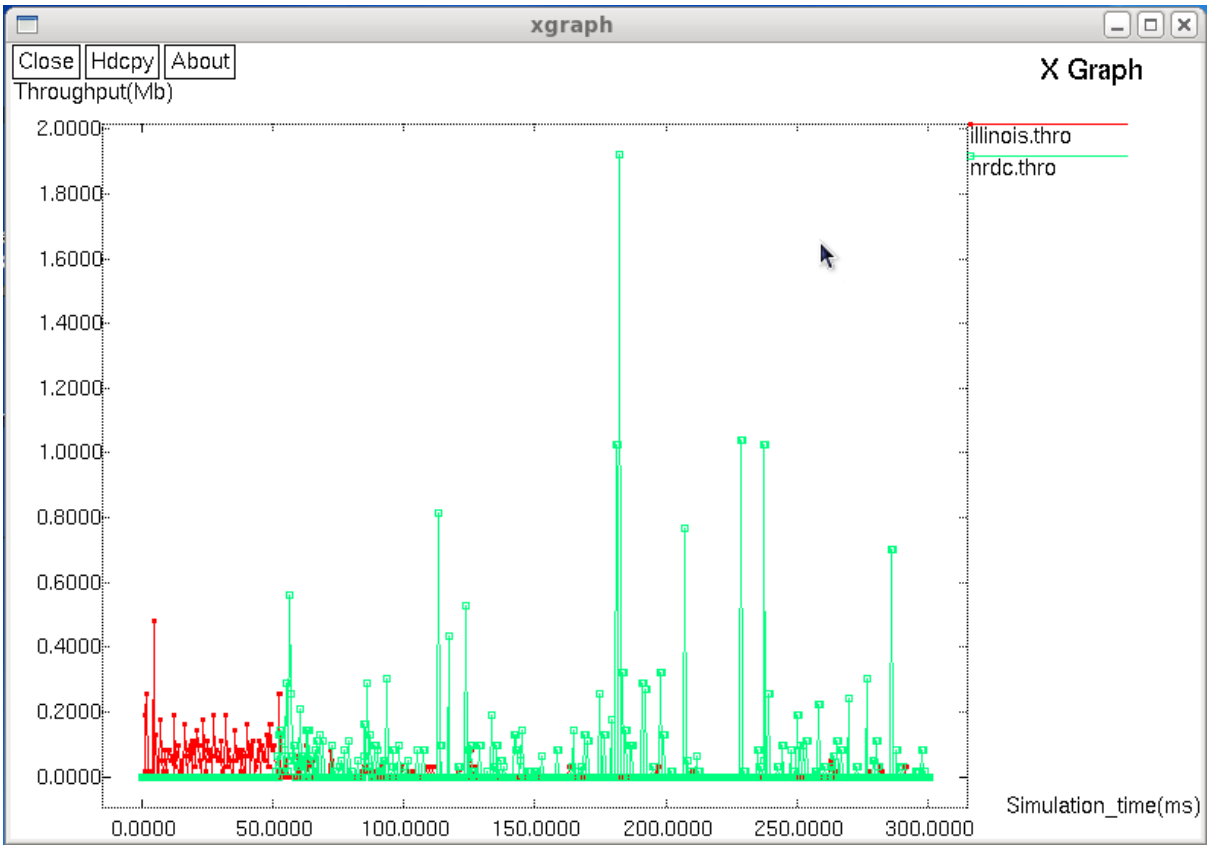
Wired cum Wireless Network Topology



Comparison of NRDC with DSDV



Comparison of NRDC with BIC



Comparison of NRDC with ILLINOIS

CHAPTER 10

References

1. Yu Wang, Member, IEEE, Weizhao Wang, Student Member, IEEE, and Xiang-Yang Li, Member, IEEE, Efficient Distributed Low-Cost Backbone Formation for Wireless Networks ; VOL. 17, NO. 7, JULY 2006.

2. Jun Cheol Park and Sneha Kumar Kasera School of Computing, University of Utah; Expected Data Rate: An Accurate High-Throughput Path Metric For Multi-Hop Wireless Routing, the IEEE SECON 2005 proceedings.

3. Haibo Zhang and Hong Shen; Energy-Efficient Beaconless Geographic Routing in Wireless Sensor Networks, VOL. 21, NO. 6, JUNE 2010.

4. C. Wang, Member, IEEE, B. Li, Senior Member, IEEE, K. Sohraby, Upstream Congestion Control in Wireless Sensor Networks Through Cross-Layer Optimization VOL. 25, NO. 4, MAY 2007.

5. Fernando Paganini, Senior Member, IEEE, and Enrique Mallada, A Unified Approach to Congestion Control and Node-Based Multipath Routing IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 17, NO. 5, OCTOBER 2009.

6. M. Chandrasekaran And R.S.D Wahida Banu, Performance Evaluation Of Polynomial Congection Control Algorithms Mimd-Poly And PIPD-Poly In TCP Networks, VOL. 20, NO. 5, JUNE 2009.

7. VVangBo, HuangChuanhe, YangVVenzhong, VVangTong, School of Computer, VVuhan University, Trust Opportunistic Routing Protocol in Multi-hop Wireless Networks, VVuhan, China, 430072.

8. Kostas Pentikousis, State University Of New York At Stony Brook, Tcp In Wired-Cum-Wireless Environments, IEEE Communications Surveys, http://www.comsoc.org/pubs/surveys , Fourth Quarter 2000.

9. Claudio Casetti Politecnico Di Torino, Italy, Mario Gerla, Ucla Computer Science Department, Usa, Saverio Mascolo, Politecnico Di Bari, Italy, M.Y. Sanadidi And

Ren Wang, Ucla Computer Science Department, TCP WESTWOOD: End-To-End Congestion Control For Wired/Wireless Networks, Usa.

10. M.Kalpana And Dr.T.Purusothaman, IJCSNS International Journal of Computer Science and Network Security, Performance Evaluation Of Exponential TCP/IP Congestion Control Algorithm, VOL.9 No.3, March 2009.

11. Neng-Chung Wang, Chi-Len Chiou, Performance Enhancement For TCP Over Dynamic Bandwidth Environment, Juy 2006.

12. Guoyou He Destination-Sequenced Distance Vector (DSDV) Protocol Networking Laboratory Helsinki University of Technology, ghe@cc.hut.fi