Efficient and Secured Multicasting in Ad Hoc Network

By

SEJAL L BUTANI

09MCES02

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May-2012

Efficient and Secured Multicasting in Ad Hoc Network

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

M.Tech. In Computer Science & Enginnering

By

Sejal Butani

09MCES02

Guide

Prof. Sharada Valiveti

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2012

UNDERTAKING

I, Sejal L. Butani, Roll No. 09MCES02, give undertaking that the Major Project entitled "Efficient and Secured Multicasting in Ad hoc Network" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Technology of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature

Date:

Place:

Certificate

This is to certify that the Major Project entitled "Efficient and Secured Multicating in Ad hoc Network" submitted by Sejal L. Butani(09MCES02), in degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Prof. Sharada Valiveti
Guide and Associate Professor,
Computer Science and Engineering Dept,
Institute of Technology,
Nirma University, Ahmedabad

Dr. S. N. Pradhan
Professor and PG-Coordinator,
Computer Science and Engineering Dept.,
Institute of Technology,
Nirma University, Ahmedabad

Prof. D. J. Patel
Professor and Head(CSE Dept.),
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr. K Kotecha
Director,
Institute of Technology,
Institute of Technology,
Nirma University, Ahmedabad

Acknowledgements

My deepest thanks to **Prof. Sharada Valiveti**, Associate Professor, Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad the Guide of the project that I undertook for giving her valuable inputs and correcting various documents of mine with attention and care. She has taken the pain to go through the project and make necessary amendments as and when needed.

My deep sense of gratitude to **Dr. S. N.Pradhan**, Professor and PG-Coordinator of Department of Computer Engineering and **Prof. D.J.Patel**, Professor and Head of Department of Computer science and Engineering, Institute of Technology, Nirma University, Ahmedabad for an exceptional support and continual encouragement throughout the Major project.

I would like to thank **Dr. Ketan Kotecha**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for his unmentionable support, providing basic infrastructure and healthy research environment.

Lastly and most importantly, no words are enough to acknowledge constant support and sacrifices of my family members, especially my parents and husband because of whom I am able to complete my dissertation work successfully.

- Sejal L. Butani
09MCES02

Abstract

An Ad hoc Network consists of a set of autonomous mobile nodes that communicates via multi-hop wireless communication in an infrastructure-less environment. In such a network, group communication takes place by implementing a multicasting technique. This multicasting technique is intended to provide energy and bandwidth efficiency with secure content delivery. The project concentrates on identifying such an efficient multicasting technique. On the basis of comparison of multicasting protocols, Protocol for Unified Multicasting through Announcement (PUMA) has been chosen for initial implementation. PUMA does not rely on any unicast routing approach. It delivers data at a higher efficiency, while also provides a tight bound for control overhead in a wide range of network scenarios. PUMA also provides high throughput and better End-to-End delay. Secure communication is a major concern in PUMA, especially because multicasting protocols are applied in areas such as audio/video conferencing, corporate communications, collaborative and groupware applications. For secure communication, the performance of RSA and ElGamal was examined; Findings say that RSA's execution time is comparatively less. To guarantee the data integrity in ad hoc networks, SHA-1 and MD5 are analyzed with RSA. Finally, the integration of PUMA has been done with RSA and SHA1 to guarantee more security.

Abbreviation Notation

PUMA ..... Protocol for unified multicasting through announcements
ODMRP ..... On-demand Multicast Routing Protocol
MAODV ..... Multicast Ad hoc On-Demand Distance Vector
NSMP ..... Neighbor Supporting Ad hoc Multicast Routing Protocol
CAMP ..... Core Assisted Mesh Protocol
DCMP ..... Dynamic Core based Multicast routing Protocol for ad hoc wireless networks
AMRIS ..... Ad hoc Multicast Routing protocol
MA ..... Multicast Announcement
NS ..... Network Simulator
CBR ..... Constant Bit Rate
PDF ..... Packet Delivery Fraction
CA ..... Certificate Authority
RSA ..... Rivest Shamir Adleman
SHA1 ..... Secure Hash Algorithm
MD5 ..... Message Digest-5

Contents

List of Tables

List of Figures

Chapter 1

Introduction

An Ad hoc Network consists of a set of autonomous mobile nodes that communicates via multi-hop wireless communication in an infrastructureless environment. It is an autonomous system in which mobile nodes connected by wireless links are free to move randomly and often act as routers at the same time. Ad hoc networks have become increasingly relevant in recent years due to their potential applications in military battlefield, emergency disaster relief, vehicular communications etc.

In ad hoc networks, nodes communicate with each other by way of radio signals, which are broadcast in nature. Broadcast is a unique case of multicast, wherein all nodes in the network should get the broadcast message. In ad hoc applications, collaboration and communication among a group of nodes are necessary. Instead of using multiple unicast transmissions, it is advantageous to use multicast in order to save network bandwidth and resources. Multicasting is a communication process in which the transmission of message is initiated by a single user and the message is received by one or more end users of the network. Multicasting in wired and wireless networks has been advantageous and used as a vital technology in many applications such as audio/ video conferencing, corporate communications, collaborative and groupware applications, stock quotes, distribution of software, news etc. Under multicast communications, a single stream of data can be shared with multiple recipients and data is only duplicated when required. Main purpose of multicasting is to provide multiple packets to multiple receivers using bandwidth and energy efficiently.

The Multicasting protocols can be classified into tree based and Mesh based protocols. The main objective of routing protocol structure is to efficiently deliver information to the members of the multicast group while avoiding non members. A tree based multicasting protocol maintains either shared based multicast tree or multiple multicast tree (one per each sender) to deliver information from senders to receivers of a multicast group. In a multicasting tree, there is usually only one single path between a sender and a receiver, while in a routing mesh protocol, there may be multiple paths between each sender-receiver pair. Routing meshes are thus more suitable than routing trees for system with frequently changing topology due to availability of multiple paths between a sender and a receiver. Example of tree based multicasting protocols are the multicast ad hoc on-demand distance vector protocol (MAODV), and AMRIS (Ad hoc Multicast Routing protocol). The well-known examples of mesh-based multicasting protocols are the Core assisted mesh protocol (CAMP), On-demand multicast routing protocol (ODMRP) and Protocol for unified multicasting through announcements (PUMA).

This thesis contains a survey of different multicasting protocols and their comparison. From the analysis, PUMA has been selected for efficient multicasting based on performance results. NS-2.35 simulator is used for simulation. Secure communication is also a major concern in PUMA because this protocol is used in many applications. Public key cryptography algorithms like RSA and ElGamal are used to add security in the existing PUMA architecture. Data integrity is also done in that using hash algorithm. The results are then analysed based on the performance metrics in order to have secure and efficient communication.

1.1 Objective of the Work

The main objective of the work is to build an efficient and secured multicast protocol in Ad hoc network.

Specific Objectives

- To minimize routing overhead.

- To maximize packet delivery fraction.
- To maximize Throughput.
- To secure Packet Delivery.
- To provide Data Integrity.
- To provide better end-to-end delay.

1.2 Motivation of the Work

Multicasting is an efficient solution for group communication on the Internet. Instead of sending a separate copy of data per receiver, a sender can send just a single copy, and the multicast routers in the network make copies and forward packets appropriately to all the receivers. Thus multicasting utilizes network resources such as bandwidth and buffer space efficiently, and reduces load at the sender(s) as well as the transmit routers. Security is also needed because it applied in many emerging application.

1.3 Scope of the work

The scope of this work is to increase efficiency and security in the system. And currently, the system is designed for PUMA only. There is still scope for improving the system for different protocols.

1.4 Thesis Organization

The rest of the thesis is organized as follows:

Chapter [2], Literature Survey, shows overview of different Multicasting Protocols and comparison between them.

Chapter [3], Overview of the PUMA protocol, focuses on the basic working of PUMA protocol.

Chapter [4], Study of NS-2 Simulator, focuses on basics of NS-2, a discrete event network simulator which is heavily used in ad-hoc networking research. Also it includes an overview of programming language and process of integrating new protocol.

Chapter [5], Simulation of protocols on NS-2.35, shows the simulation results of PUMA and MAODV on the basis of performance metrics like packet delivery fraction, throughput and routing overhead.

Chapter [6], Security requirements, shows why security is needed in multicasting ad hoc network.

Chapter [7], Public key cryptography and Data integrity, focuses on the security part. To secure communication using RSA and ElGamal algorithms is discussed in this chapter. And also, Data integrity is added using SHA-1 and MD5 algorithms.

Chapter [8], Integration of PUMA and Security, shows the integration of PUMA with security module. Finally, the performance results have been generated using execution time, Packet delivery fraction and End-to-End delay.

Finally, in chapter [9] concluding remarks and future work is presented.

Chapter 2

Literature Survey related to Multicasting Protocols

In the following subsections I am going to explain multicasting protocols in Ad-hoc network and then comparison of multicasting protocols.

2.1 ODMRP

ODMRP (On-demand Multicast Routing Protocol)[4] is mesh based, and uses a forwarding group concept (only a subset of nodes forwards the multicast packets). A soft state approach is taken in ODMRP to maintain multicast group members. No explicit control message is required to leave the group. In ODMRP, group membership and multicast routes are established and updated by the source upon demand. Similar to on-demand unicast routing protocols, a request phase and a reply phase constitute the protocol. When a multicast source has packets to send, it floods a member advertising packet to neighbor nodes with data payload attached. This packet, called JOIN_DATA, is periodically broadcast to the entire network to refresh the membership information, thus taxing bandwidth resources. The advantage of ODMRP is that it performs well in terms of packet delivery ratio in highly dynamic environments, because ODMRP provides route redundancy with a mesh topology that enables it to handle dynamic environments much better. The main disadvantage in ODMRP is that multicast routes and group membership are established and updated by the

source on demand, which can create congestion due to the significantly higher processing load the node must handle.

2.2 MAODV

MAODV (Multicast Ad hoc On-Demand Distance Vector)[3] builds a group tree, shared by all sources and receivers for a group. It uses a hard state maintenance approach. The group tree enables it to localize group joins and connection of newly active sources to the multicast tree, as well as, repairs when the tree becomes disconnected. The use of a shared tree and the localized connection and reconnection to the tree result in longer forwarding paths for data packets. Such paths have a higher likelihood of packet loss due to collisions, and higher end-to-end delay; they are also more likely to break which also leads to packet loss and a more frequent invocation of the route repair mechanisms within the protocol. MAODV requires the use of periodic neighbor detection packets for detection of broken links, and periodic group leader control packet floods for disseminating a multicast group's sequence number. MAODV creates a shared tree between the multicast sources and receivers for a multicast group. The root of each group tree is a multicast source or receiver for the group that has been designated as a group leader. Each data packet is forwarded to all nodes on this list except the node from which it was received. The packet is forwarded as either a unicast to each such neighbor, or as a broadcast, when it needs to be forwarded on to multiple nodes.

The advantage of MAODV is that routes are established on demand and destination sequence numbers are used to find the latest route to the destination. MAODV's main disadvantage is that it suffers from high End-to-End Delay since packets must travel longer paths within the shared tree. Also because of the higher network load caused by the large number of control and data transmissions, congestion may increase.[8]

2.3 NSMP

NSMP (Neighbor Supporting Ad hoc Multicast Routing Protocol)[7] is an extension to ODMRP aiming to restrict the flood of control packets to a subset of the entire network.

NSMP is a robust and efficient ad hoc multicast routing protocol. Mesh infrastructure used in NSMP has resilience against link failures. A soft state approach is used, and routes are built and maintained with basic route discovery and reply messages. NSMP also operates independent of unicast routing protocol. NSMP reduces the routing overhead by localizing route discovery and maintenance operations. NSMP attempts to achieve the route efficiency of the multicast tree while enjoying the robustness of the multicast mesh infrastructure. It is known that the mesh structure is more robust against topology changes than the tree structure. In selecting a route, NSMP prefers a path that contains existing forwarding nodes to reduce the number of forwarding nodes. This enhances route efficiency, leading to less contention and further to lower end-to-end delay.

2.4 CAMP

CAMP (Core Assisted Mesh Protocol)[2] uses a shared mesh structure. All nodes in the network maintain a set of tables with membership and routing information. CAMP uses hard state maintenance approach to support multicast group membership. Moreover, all member nodes maintain a set of caches that contain previously seen data packet information and unacknowledged membership requests. Cores are used to limit the flow of JOIN REQUEST packets.

CAMP possesses good control traffic scalability to facilitate the increased size of a multicast group. Since JOIN_REQUESTS are only propagated until they reach a mesh member, CAMP does not incur exponential growth of multicast updates as the number of nodes and group members increase, which represents a significant advantage with respect to bandwidth allocation and energy consumption. However, an important disadvantage of CAMP is that it employs a unicast routing protocol to handle network convergence and control traffic growth in the presence of mobility. Another major disadvantage of CAMP is that it assumes that the routing information from a unicast routing protocol is available and that the correct distance to the specified receiving node can be determined within a specified time. Finally, CAMP assumes that a wireless router contains a preexisting mapping service that provides group addresses which are identified by their specific names.

2.5 DCMP

DCMP (Dynamic Core based Multicast routing Protocol for ad hoc wireless networks)[7] is an extension to ODMRP that designates certain senders as cores and reduces the number of senders performing flooding. DCMP proposes a shared-mesh based approach for multicast communication. DCMP combines a mesh network approach with a tree routing concept: multiple core based trees are interconnected by a mesh. DCMP classifies senders into three categories: Active sources keep their multicast relations up to date by periodically flooding the network with control packets. Passive sources never actively participate in multicast delivery path creation. Passive sources are always associated with a core active source that forwards data packets for them. Core active sources incorporate the same functionality as active sources and additionally act as cores for passive sources. The control overhead in DCMP increases only moderately with an increasing number of sources, because DCMP allows multiple sources to become passive.

2.6 AMRIS

AMRIS (Ad hoc Multicast Routing protocol)[6] establishes a shared tree for multicast data forwarding. AMRIS does not require a separate unicast routing protocol. Each node in the network is assigned a multicast session ID number. The ranking order of ID numbers is used to direct the flow of multicast data.

The main difference between AMRIS and other multicast routing protocols is that each participant in the multicast session must have a session specific multicast session member id (msm-id). This msm-id provides each node with an indication of its "logical height" in the multicast delivery tree. The drawbacks of AMRIS are that each node must send a periodic beacon to signal their presence to neighboring nodes and that it is very sensitive to mobility and traffic load. The primary reasons for its poor performance are the number of necessary retransmissions and the size of beacons, both of which create overhead and can cause increased congestion.

2.7 PUMA

PUMA (Protocol for Unified Multicasting through Announcement) [2]does not require any unicast routing protocol to operate, or the pre-assignment of cores to groups. The section below shows PUMA operation in detail. PUMA derives from its use of very simple signaling (multicast announcements) to accomplish all the functions needed in the creation and maintenance of a multicast routing structure in a MANET. Multicast announcements are used to elect cores dynamically, determine the routes for sources outside a multicast group to unicast multicast data packets towards the group, join and leave the mesh of a group, and maintain the mesh of the group. PUMA protocol is advantageous due to its high packet delivery ratio and limited congestion. The comparison of multicasting protocols is shown in Table 2.1.

	ODMRP	MAODV	NSMP	CAMP	DCMP	AMRIS	PUMA
N/w topology	Mesh	Tree	Mesh	Mesh	Mesh	Tree	Mesh
Initialization Approach by	Source	Source	Source	Source & receiver	Source	Source	Receiver
Maintenance Approach	Soft State	Hard State	Soft State	Hard State	Soft State	Soft State	Soft State
Dependency	No	Yes	No	Yes	No	No	No
Loop Free	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Flooding of control packets	Yes	Yes	Yes	No	Yes	No	No
Independent Routing Protocol	Yes	Yes	Yes	No	Yes	No	Yes
Periodic Control Msg	Yes	Yes	Yes	No	Yes	Yes	Yes
Advantage	- Packet delivery ratio is better due to route redundancy	- Avoids sending duplicate packets to receivers - Routes are on demand	-Reduces overhead -Lower End-to-End delay -less contention	-Better bandwidth allocation - Good scalability because due to no flooding	Simultaneous data flow in case of link breaks - Improve scalability	-No loops -Link breaks are locally repaired - simplicity	-High data delivery ratio -Limited Control overhead
Disadvantage	- Create congestion due to high Processing load	-High End to End delay - High network load due to larger data & control x'mision	-Control Overhead high with increase of high network size	-Network convergence and control traffic growth in the presence of mobility	-Control overhead increase with increasing sources	-Waste of bandwidth -Slow rejoin scheme -Increased average hop distance	-No acknow-legement -No delivery Validation

Table 2.1:Comparison of Multicasting Protocols

2.8 Summary

In this chapter, different multicasting protocols and their comparison are discussed in details. After comparison of protocols, PUMA has been selected because it does not rely on any unicast routing approach. Here, CAMP follows unicast routing approach and this may incur considerable overhead in a large ad hoc network. ODMRP is improved version in terms of control packet overhead as compared to DCMP and NSMP. MAODV and AMRIS, on the other hand, are tree based protocols and they provide only a single route between senders and receivers.

Chapter 3

Study of NS-2 Simulator

For making simulation trustworthy, simulation process should be repeatable so that it can be used for further reviews, unbiased means should be used to variety of scenarios, rigorous must truly test the aspects of environment being studied and statistically sound. After detailed study of different existing simulators NS-2 was selected as simulator for implementation.

3.1 Network Simulator-2

NS-2 is an event driven packet level network simulator developed as part of the VINT project targeted at networking research. NS provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. NS began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995 ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. Currently ns development is supported through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researchers including ACIRI. NS has always included substantal contributions from other researchers, including wireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems.NS-2 has many and expanding uses including:

- To evaluate the performance of existing network protocols.

- To evaluate new network protocols before use.
- To run large scale experiments not possible in real experiments.
- To simulate a variety of IP networks.

NS-2 is available for both Windows and Linux platform. Under Linux, following components can be installed. [16]

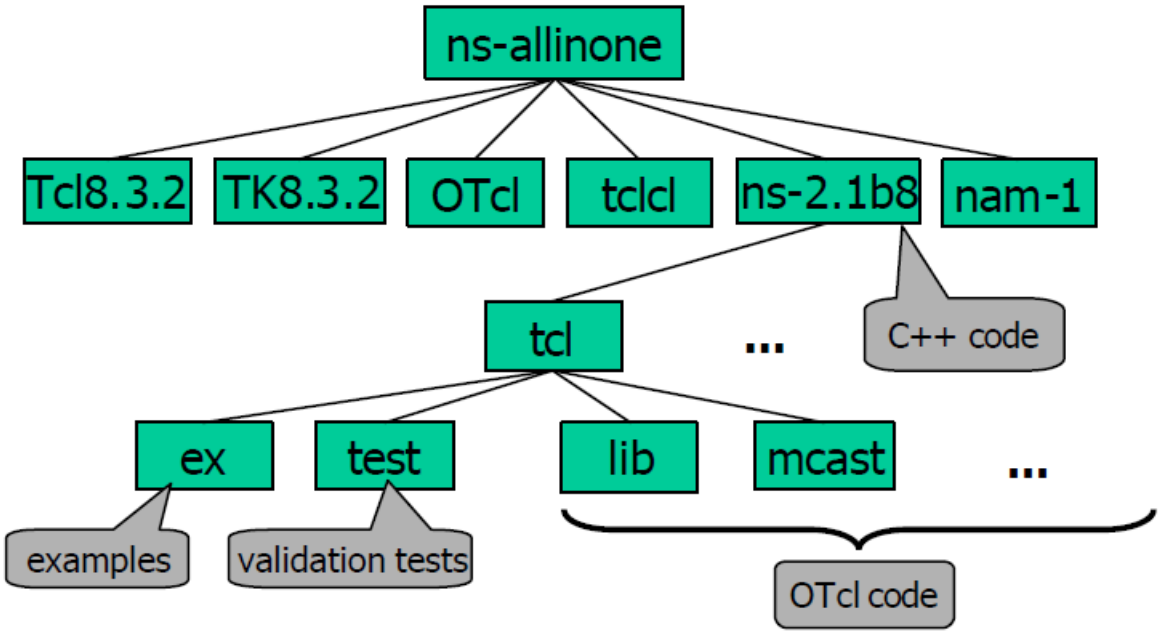


Figure 3.1: Directory structure of NS-2

3.1.1 Architecture of NS-2

As shown in the simplified user's view of Figure 3.2[23], NS is an Object-oriented Tcl (OTcl) script interpreter that has a simulation event scheduler and network component object libraries, and network set-up (plumbing) module libraries. To use NS-2, a user programs in the OTcl script language. An OTcl script will do the following.

- Initiates an event scheduler.
- Sets up the network topology using the network objects.

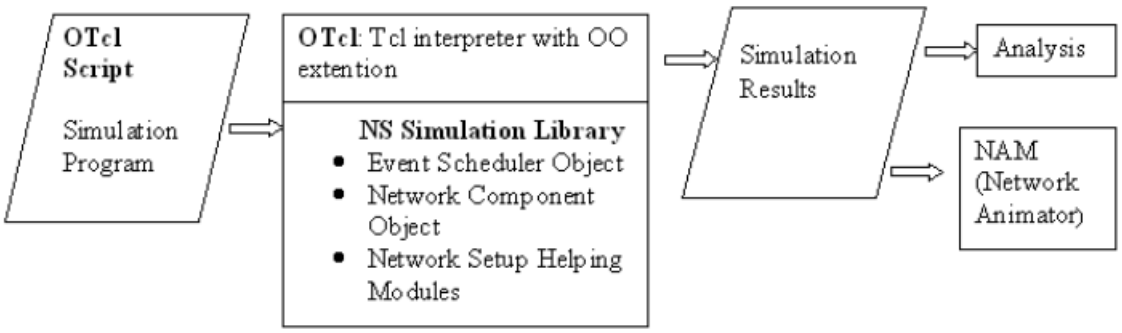


Figure 3.2: User's view of NS-2

- Tells traffic sources when to start/stop transmitting packets through the event scheduler.

A user can add OTcl modules to NS-2 by writing a new object class in OTcl. These then have to be compiled together with the original source code.
Another major component of NS besides network objects is the event scheduler. 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. The event scheduler in NS-2 performs the following tasks:

- Organizes the simulation timer.
- Fires events in the event queue.
- Invokes network components in the simulation.

Depending on the users purpose for an OTcl simulation script, following simulation results are stored as trace files, which can be loaded for analysis by an external application:

- A NAM trace file (file.nam) for use with the Network Animator Tool
- A Trace file (file.tr) for use with XGraph or GnuPlot.

3.1.2 C++/OTcl linkage

NS2 is written in C++ with OTcl interpreter as a front end. For efficiency reason, NS separates the data path implementation from control path implementations. Languages used with NS-2 are:

- Split-Language programming
  - Scripting Language (Tcl - Tool Command Language and pronounced as tickle)
  - System Programming Language (C/C++)
- NS is a Tcl interpreter to run Tcl Scripts: TclCL is the language used to provide a linkage between C++ and OTcl. Toolkit Command Language (Tcl/OTcl) scripts are written to set up/configure network topologies. TclCL provides linkage for class hierarchy, object instantiation, variable binding and command dispatching. OTcl is used for periodic or triggered events.

The following is written and compiled with C++

- Event Scheduler
- Basic network component objects.

These 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 and 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. It is also possible to add member functions and variables to a C++ linked OTcl object. [23]

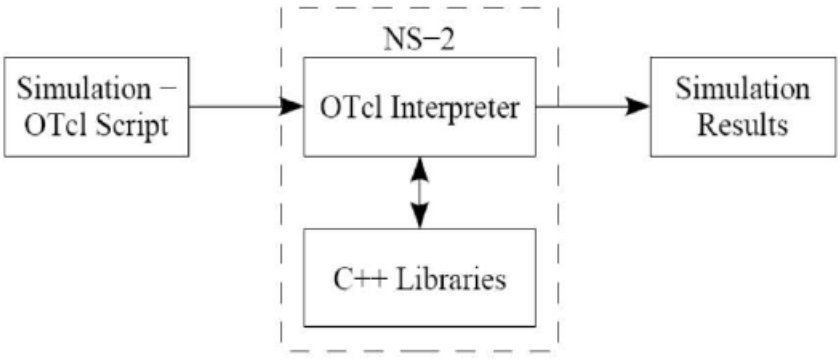


Figure 3.3: Architecture of NS-2

3.1.3 Network AniMation (NAM) Trace

NAM trace is records simulation detail in a text file, and uses the text file to play back the simulation using animation. NAM trace is activated by the command $ns namtrace-all $file, where ns is the Simulator handle and file is a handle associated with the file (e.g., out.nam in the above example) which stores the NAM trace infor- mation. After obtaining a NAM trace file, the animation can be initiated directly at the command prompt through the following command:

>> nam filename.nam

Many visualization features are available in NAM. These features are for example animating colored packet ows, dragging and dropping nodes (positioning), labeling nodes at a specified instant, shaping the nodes, coloring a specific link, and monitoring a queue.

3.2 Programming languages

This topic covers the basic elements of the programming languages, which are essential for developing NS2 simulation programs. These include Tcl/OTcl which is the basic building block of NS2 and AWK which can be used for post simulation analysis.

3.2.1 Tcl/OTcl Programming

NS2 uses OTcl to create and configure a network, and uses C++ to run simulation. All C++ codes need to be compiled and linked to create an executable file. Since the body of NS2 is fairly large, the compilation time is not negligible. A typical Pentium 4 computer requires few seconds (long enough to annoy most programmers) to compile and link the codes with a small change such as including int i=0; into the codes.

OTcl, on the other hand, is an interpreter, not a compiler. Any change in an OTcl file does not need compilation. Nevertheless, since OTcl does not convert all the codes into machine language, each line needs more execution time. In summary, C++ is fast to run but slow to change. It is suitable for running a large simulation. OTcl, on the other hand, is slow to run but fast to change. It is therefore suitable to run a small simulation over several repetitions (each may have different parameters).

NS2 is constructed by combining the advantages of these two languages. NS2 manual

provides the following guidelines to choose a coding language: Use OTcl for configuration, setup, or one time simulation, or to run simulation with existing NS2 modules. This option is preferable for most beginners, since it does not involve com- plicated internal mechanism of NS2. Unfortunately, existing NS2 modules are fairly limited. This option is perhaps not sufficient for most researchers. Use C++ when you are dealing with a packet, or when you need to modify existing NS2 modules.

Tcl is a general purpose scripting language. While it can do anything other lan- guages could possibly do, its integration with other languages has proven even more powerful. Tcl runs on most of the platforms such as Unix,Windows, and Mac. The strength of Tcl is its simplicity. It is not necessary to declare a data type for variable prior to the usage. At runtime, Tcl interprets the code line by line and converts the string into appropriate data type (e.g., integer) on the fly.

OTcl is an object-oriented version of Tcl, just like C++ is an object-oriented version of C. The basic architecture and syntax in OTcl are much the same as those in Tcl. The difference, however, is the philosophy behind each of them. In OTcl, the con- cepts of classes and objects are of great importance. A class is a representation of a group of objects which share the same behavior(s) or trait(s). Such a behavior can be passed down to child classes. In this respect, the donor and the receiver of the behaviors are called a superclass (or a parent class) and a subclass (or a child class), respectively. Apart from inheriting behaviors from a parent class, a class defines its own functionalities to make itself more specific. This inheritance is the very main concept for ansy OOP including OTcl.

3.3 AWK script

AWK is a general-purpose programming language designed for processing of text files. AWK refers to each line in a file as a record. Each record consists of fields, each of which is separated by one or more spaces or tabs. Generally, AWK reads data from a file consisting of fields of records, processes those fields with certain arithmetic or string operations, and outputs the results to a file as a formatted report.

To process an input file, AWK follows an instruction specified in an AWK script. An AWK script can be specified at the command prompt or in a file. While the strength of the

former is the simplicity (in invocation), that of the later is the functionality. In the later, the programming functionalities such as variables, loops, and conditions can be included into an AWK script to perform desired actions. In what follows we give a brief introduction to the AWK language.

AWK Programming Structure: The general form of an AWK program is shown below:

BEGIN <initialization>

<pattern1> <actions>

<pattern2> <actions>

. . .

END <final actions>

Prior to processing an input text file, AWK performs <initialization> specified in the curly braces located after the reserved word BEGIN. Then, for each record, it performs actions if the records match with the corresponding pattern. After process- ing the entire file, it performs <final actions> specified in the curly braces located after the reserved word END.

3.4 Summary

Here, NS-2 is selected as a network simulator tool,it is an open source system that is developed using C++ and Tool Control Language.So it is easy for user to add specific components to the system to serve their own purposes.

Chapter 4

PUMA OVERVIEW

PUMA uses a reactive routing protocol which discovers route only when it is required. It uses a receiver-initiated approach in which the receivers join the multicast group by using address of a special node, without the need for network-wide flooding of control or data packets from all the sources of the group. PUMA uses the shared Mesh based Multicast topology and eliminate the need for a unicast routing protocol and pre-assignment of cores to multicast groups. PUMA uses the soft state approach for Multicast group Maintenance where multicast group membership and its associated routes are refreshed periodically by flooding its Multicast Announcement (MA) packet. In Puma, nodes maintain a packet ID cache to drop duplicate data packets.

PUMA uses a single control message called Multicast announcement for all its functions. It uses Multicast announcement (MA) to create and maintain its multicast topology. Multicast announcement header is shown in fig 4.1. Each Multicast announcement (MA) specifies a mesh membership code, the distance to core, the address of the group and core node, a sequence number and a parent that states the preferred neighbor to reach the core. Mesh membership code field is set to 1 when a node wants to join into the multicast group. Distance to core field count hops from current node to core node. The Successive multicast announcements have a higher sequence number than previous multicast announcements sent by the same core.
Multicast Announcements are used to elect the core, find out the sources outside a multicast

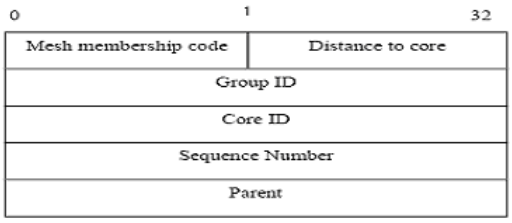


Figure 4.1: Multicast Announcement header

group to unicast data packets towards the group, Join and leave the mesh of a group and to maintain the mesh of the group.

4.1 Core Election procedure in PUMA

PUMA chooses a core for each multicast group in the network. Each connected component has only one core. If one receiver joins the group before other receivers, it becomes the core of the group. If several receivers join the group at the same time, the one with highest ID becomes the core of the group.

When a receiver needs to join a multicast group, it first determines whether it has received a multicast announcement for that group. If the node has received, then it takes on the core specified in the announcement it has received, and it transmits the multicast announcements that specifies the same core for the group. Otherwise it assumes itself as the core of the group and starts transmitting multicast announcement periodically to its neighbors stating itself as the core of the group and a hop count of 0 to itself. Nodes propagate multicast announcements based on the best multicast announcement they receive from their neighbors.

4.2 Propagation of Multicast announcements and establishment of Connectivity lists

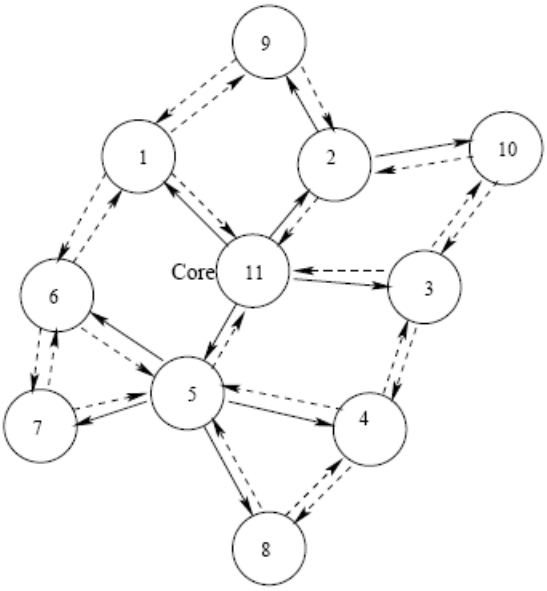
A node that believes itself to be the core of a group transmits multicast announcements periodically for that group. As the multicast announcement pass through the network, it establishes a connectivity list at every node in the network. Connectivity list is used to form a mesh structure and to route data packets from receivers to the core.

A node keeps track of the data from all the multicast announcements it receives from

its neighbors in the connectivity list. Fresher multicast announcements from a neighbor overwrite entries with lower sequence numbers for the same group. Hence all the nodes in a group store the recent information about a neighbor for each core in the group.
Each entry in the connectivity list also store the time when it was received, and the neighbor from which it was received. The node then generates its own multicast announcement based on the best entry in the connectivity list.
While electing core, on receiving a multicast announcement with higher core ID, all entries in the connectivity list with a lower core ID are erased. Hence all suitable entries in the connectivity list at any point of time have the same core ID and sequence number. Among these suitable entries, the entries with the shortest distance to the core would be qualified as the best entries, and the neighbors corresponding to these entries are called parents.
After selecting the best multicast announcement, the node generates the fields of its own multicast announcement in the following way:

- Core ID:The core ID in the best multicast announcement.
- Group ID: The group ID in the best multicast announcement.
- Sequence number:The sequence number in the best multicast announcement.
- Distance to core:One plus the distance to core in the best multicast announcement.
- Parent:The neighbor from which it received the best multicast announcement.
- Mesh member:A node sets its membership code field based on whether it is a mesh member or non-member.

After generating its own multicast announcement, it will broadcast the announcement to all its neighbors. The connectivity list stores information about one or more routes that exist to the core. When a core change occurs for a particular group, the node clears the entries of its old connectivity list and builds a new one, specific to the new core. Figure 4.2 illustrates the propagation of multicast announcements and the building of connectivity lists. The solid arrows indicate the neighbor from which a node receives its best multicast announcement. Node 6 has three entries in its connectivity list for neighbors 5, 1, and 7. However it chooses the entry it receives from 5 as the best entry, because it has the shortest



Connectivity List at node 6
Core Id = 11 Group Id = 224.0.0.1, Seq No 79

Neighbor	Multicast Announcement		Time (ms)
	Distance To Core	Parent	
5	1	11	12152
1	1	11	12180
7	2	5	12260

Figure 4.2: Dissemination of multicast announcements

distance to core and has been received earlier than the one from node 1. Node 6 uses this entry to generate its own multicast announcement, which specifies Core ID = 11, Group ID = 224.0.0.1, Sequence Number = 79, Distance to Core = 2 and Parent = 5. When a node wants to send data packets to the group, it forwards them to the node from which it received its best multicast announcement. If that link is broken, then it tries its next best and so on. Hence each node in the network has one or more routes to the core. The multicast announcement sent by the core has distance to core set to zero and parent field set to invalid address.

Multicast announcements are generated by the core every three seconds. After receiving a multicast announcement with a fresh sequence number, nodes wait for a short period (e.g. 100 ms) to collect multicast announcements from multiple neighbors before generating their

own multicast announcement.

When multiple groups exit, nodes collect all the fresh multicast announcements they receive, and broadcast them periodically for every multicast announcement interval. However, multicast announcement representing groups being received for the first time, resulting in a new core, or resulting in variations in the mesh membership code are forwarded immediately, without aggregation. This is to pass up delays in critical operations, when electing core node and establishing mesh structure.

4.3 Establishment and Maintenance

Initially, only receivers consider themselves as mesh members and set the mesh member flag to true in the multicast announcements they send. Non-receivers consider themselves mesh members if they have at least one mesh child in their connectivity list. A neighbor in the connectivity list is a mesh child if

- Its mesh member flag is set.
- The distance to core of the neighbor is larger than the nodes own distance to core .
- The multicast announcement corresponding to this entry was received in within a time period equal to two multicast announcement intervals.

If a node has a mesh child means a mesh member. Hence it lies on a shortest path from a receiver to the core. A node forwards a multicast data packet it receives from its neighbor if the parent for the neighbor is the node itself. Hence, multicast data packets move hop by hop, until they reach mesh members. The packets are then flooded within the mesh, and group members use a packet ID cache to detect and discard packet duplicates. Figure 4.3 is an example of PUMA mesh and data forwarding within the multicast group. Nodes O and Q marks in the multicast announcements that their parent is node N. Similarly, node P marks in its multicast announcement that its parent is K. Let nodes O and P are senders. Node N forwards a data packet from O, but not from P, because only O has informed N that it considers N as its parent. Although node J is not the parent of P, it forwards the packet when it receives it from P, because mesh members do not consult their connectivity list before forwarding a packet. As a result, receiver node I will get the packet early. Node

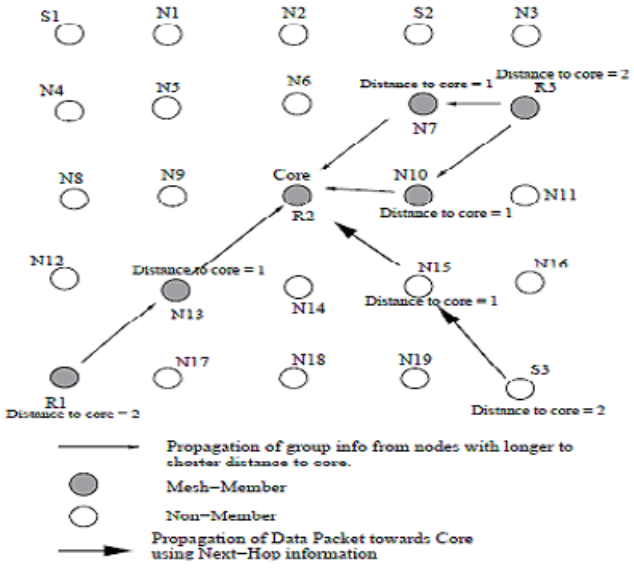


Figure 4.3: Mesh creation in PUMA

J does not rebroadcast the packet when it receives same packet from K due to duplicate packet checking.

4.4 Forwarding Multicast Data Packets

This parent field of the connectivity list entry for a particular neighbor corresponds to the node from which the neighbor received its best multicast announcement. This field allows nodes that are non-members to forward multicast packets towards the mesh of a group. A node forwards a multicast data packet it receives from its neighbor if the parent for the neighbor is the node itself. Hence, multicast data packets move hop by hop, until they reach mesh members. The packets are then flooded within the mesh, and group members use a packet ID to detect and discard packet duplicates.

4.5 Comparison of performance by structure

PUMA and ODMRP are both mesh-based protocols. However, every sender performs control packet flooding in ODMRP. Hence, depending on the number of senders there may be multiple nodes flooding the network periodically. In PUMA on the other hand, only

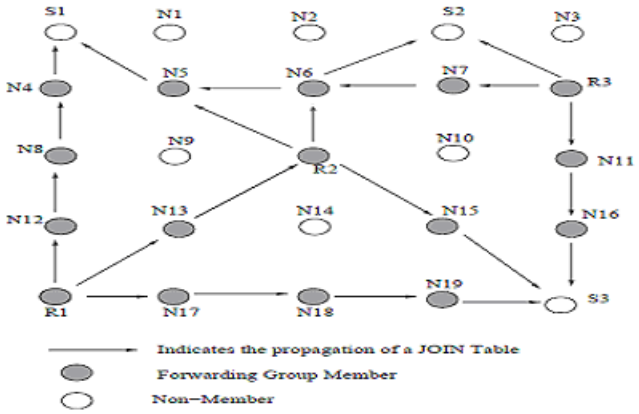


Figure 4.4: Mesh structure of ODMRP

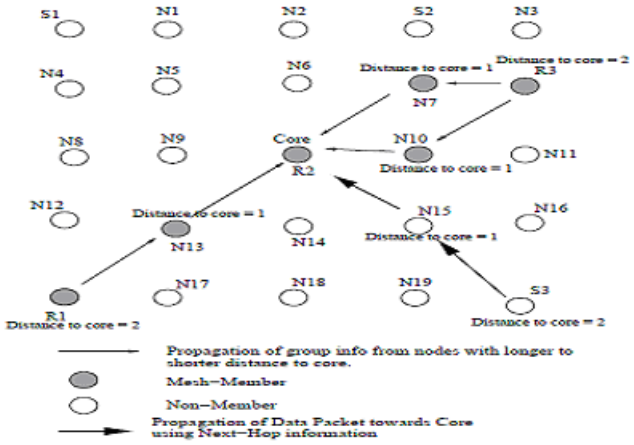


Figure 4.5: Mesh structure of PUMA

one node, i.e., the core floods the network. The mechanisms for establishing a mesh are also different in PUMA and ODMRP. The mesh constructed in ODMRP is sender-initiated whereas in PUMA it is receiver-initiated.

Figures 4.4[3] and 4.5[3] illustrate the mesh established by ODMRP and PUMA respectively, where nodes R1, R2 and R3 are receivers and nodes S1, S2 and S3 are senders. The forwarding group of ODMRP contains 16 nodes, whereas the mesh of PUMA contains only 6 nodes. Hence, a data packet sent by node S3 is retransmitted by 16 nodes in ODMRP, whereas in PUMA is is retransmitted only by 7 nodes. PUMA tends to concentrate mesh redundancy in the region where receivers exist by including all shortest paths from each receiver to the core, which is also a receiver. On the other hand, the mesh in ODMRP (i.e.,

the forwarding group) is simply the union of the shortest paths connecting all senders to all receivers. This can lead to a significant and unnecessary data packet overhead if all senders are also not receivers.

4.6 Summary

The chapter presented the overview of PUMA protocol. PUMA follows receiver-initiated approach. Receiver initiated approach has advantage over source-initiated approach like Basically the source does not know the receiver set and source does not have to process acks from each receiver. PUMA is also reactive protocol, which works faster as compared to proactive protocol. In this chapter, PUMA provides less routing overhead as compared to ODMRP based on the Mesh structure. Now, NS-2.35 is used for further implementation in PUMA.

Chapter 5

Implementation in Network Simulator-2.35

PUMA and MAODV are both receiver-oriented protocols. However, PUMA is a mesh-based protocol and provides multiple routes from senders to receivers. MAODV, on the other hand, is a tree based protocol and provides only a single route between senders and receivers.

5.1 Installation for MAODV implementation

In this section, I have simulated MAODV under network simulator NS- 2.35 with fedora 11 operating system, and then compared the results of MAODV with that of PUMA.
First, the NS2 simulator must be installed. Download the NS2 package from http: //www.isi.edu/nsna and type ../install. in the directory of extraction folder: ns-allinone-2.35.
The source code for MAODV is not available in NS-2.35 so I have added/updated some files in AODV protocol which is already available in ns-2.35. In MAODV Package, there are a total of 18 files:

1. aodv.h;
2. aodv.cc;
3. aodv_mcast.cc;

4. aodv_mtable_aux.cc;
5. aodv_mtable_aux.h;
6. aodv_mtable.cc
7. aodv_mtable.h;
8. aodv_packet.h;
9. aodv_rqueue.cc;
10. aodv_rqueue.h;
11. aodv_rtable.cc;
12. aodv_rtable.h;
13. cmu_trace.cc;
14. wireless_phy.cc;
15. wireless_phy.h;
16. node.cc;
17. node.h;
18. ns_mcast.tcl.

Copy files 1) 12) to directory ./NS-2.35/aodv/;
copy file 13) to directory ./NS-2.35/trace/;
copy files 14) and 15) to directory ./NS-2.35/mac/;
copy files 16) and 17) to directory ./NS-2.35/common;
and finally copy file 18) to directory ./NS-2.35/tcl/mcast/.
The default configuration in aodv.h includes:
#define AODV_LOCAL_REPAIR
#define AODV_LINK_LAYER_DETECTION
#define IMPROVEMENT

#define MULTICAST

#define PREDICTION

#define AODV_LOCAL_REPAIR. and #define AODV_LINK_LAYER_DETECTION. are the original configuration options in aodv.h. If I compile the protocol with only these two options, only unicast traffic can be simulated, and the simulation results should be the same as the results generated by the original protocol implementation distributed in version NS-2.35

#define IMPROVEMENT. is used only for unicast packets to be transmitted for the second time when the first-time transmission failed. If the second-time transmission still fails, the packets will be dropped. I give the packets one more chance to be sent out, because I observe that in version NS-2.35, the packet delivery ratio will be improved dramatically by trying this. I explicitly add the multicast configuration as #define MULTICAST.. The proactive approach feature can be controlled by #define PREDICTION.. So if the proactive approach is included without multicast traffic, the proactive route maintenance for unicast traffic will be invoked.

There is another feature, added as #define UPPER-LEVEL-RECEIVE. It allows the upper level agent such as a UDP agent to receive the multicast data packets. The agent port number must be known in advance and defined in aodv.h. This feature is only used for multicast traffic, used for multicast group member receiving multicast data packets at the same port number. In order to make it work, the traffic file must be carefully organized (please see the example below). The top-level Makefile (in directory ns-2.35) needs to be changed, adding aodv-mcast.o, aodv-mtable.o, and aodv_mtable_aux.o to the compilation list. Under directory ./ns-2.35/, type .make clean., to remove all *.obj files, the recompile NS2 with .make. to get the new MAODV implementation.

5.2 Installation for PUMA implementation

In this section, I have simulated PUMA under network simulator NS-2.35. PUMA is compatible with NS-2.35. To add PUMA patch I need to do following steps.

1.Move puma.patch to ñs/

"ñs" is the folder just above "ns-2.35"

2.Run patch -p0 ¡puma.patch

This will install PUMA on "ns-2.35/puma"

3. Go to "ns-2.35" and run make

This will compile PUMA

4. Try PUMA in the "ñs/ns-2.35/tcl/ex" folder.

5.3 Simulation Method and Environment

The selected protocols are evaluated using Network simulator (NS-2) of 25-125 nodes incrementing by 25 nodes within 1000m * 1000m area. Simulation runs for 100 seconds. The mobility model is selected as Random Way Point model. In this mobility model a node randomly selects a destination and it moves in the direction of the destination with a speed uniformly chosen between the minimal speed and maximal speed. After it reaches the destination, the node stays there for a pause time and then moves again. Each node moves randomly with a speed of minimum 1 to maximum 10 m/s and stays at the same place with a pause time 1s. The Distributed Coordinated Function (DCF) of IEEE 802.11 for wireless LANs is assumed as the MAC layer protocol. The Two Ray Ground model is selected for the propagation. OmniAntenna is selected for the direction antenna model.

A traffic generator was developed to simulate constant bit rate (CBR) sources. The size of data payload was 512 bytes. The senders were chosen randomly among multicast members who in turn were chosen with uniform probability among network hosts. The member nodes join the multicast session at the beginning of the simulation and remain as members throughout the simulation.

The node movement scenario files and CBR (constant Bit Rate) pattern scenario files are created by using following commands.

To create Mobile node Movement Scenario files, the command line that needs to be run under directory: ns-allinone-2.35/ns-2.35/indep-utils/cmu-scen-gen/setdest:

./setdest [-n num-of-nodes] [-p pausetime] [-s maxspeed] [-t simtime] [-x maxx] [-y maxy] > [output-file]

And, to create CBR traffic scenario files, under directory: ns-allinone-2.35/indep-utils/cmu-

scen-gen/
ns cbrgen.tcl [-type cbr—tcp] [-nn nodes] [-seed seed] [-mc connections] [-rate packet/second for one connection] > [output-file]

5.4 Performance Metrics

A trace file contains a lot of information which may not be required to analyze the performance of the protocol. Some amount of information of trace file that is sufficient to predict the efficiency of the protocol. The following performance metrics that are needed to be taken into consideration in order to analyze and compare the performance of MAODV and PUMA are:

0. **Routing Overhead-** The ratio of the number of routing packets delivered and forwarded to destination to the number of routing packets and traffic packets received.

0. **Throughput-**The ratio of the total size of received data packets to the duration of time when it receives.

0. **Packet Delivery Fraction (PDF)-** The ratio of the number of data packets expected to be received to the number of data packets delivered to the destinations.

0. **End-to-End Delay-** The average time taken by a data packet to arrive in the destination. It also includes the delay caused by route discovery process and the queue in data packet transmission. Only the data packets that successfully delivered to destinations that counted.

5.5 Performance Results

I evaluated the performance of PUMA and compared it with MAODV in terms of routing overhead, throughput, packet delivery fraction and End-to-end delay in NS-2.35. The obtained results are illustrated in Fig. 5.1, Fig. 5.2, Fig. 5.3 and Fig. 5.4.

Based on the simulation results shown in Fig. 5.1 the routing overhead of PUMA is compared with MAODV for varying number of nodes. For increasing number of nodes, the

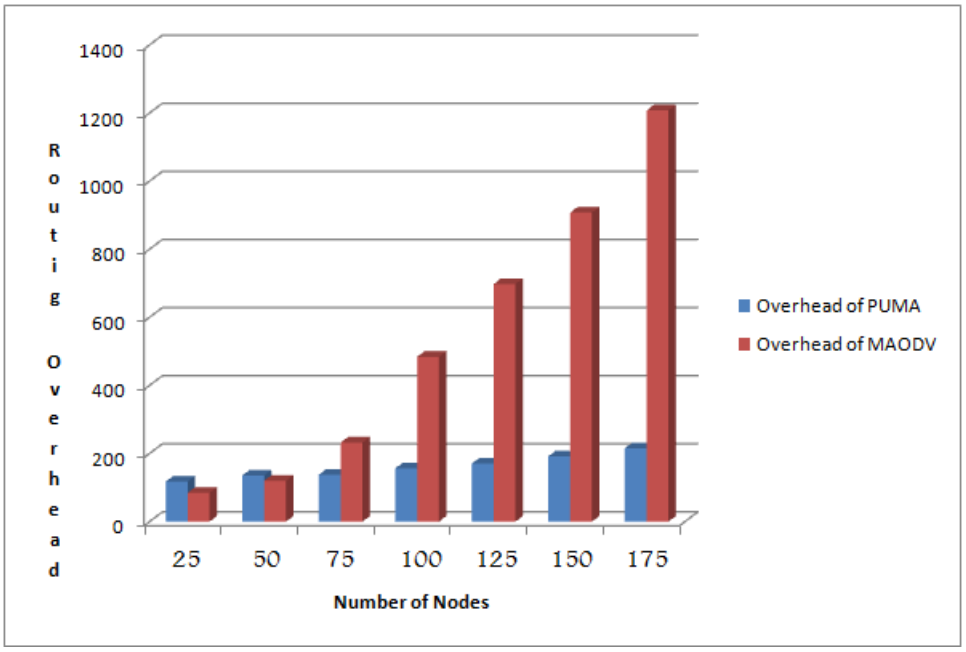


Figure 5.1: Performance Comparison of Routing overhead

routing overhead is increased in MAODV for varying number of nodes. So, MAODV incurs far more overhead compared to PUMA. Fig. 5.2, shows the Throughput analysis. For increasing number of nodes the throughput of PUMA is higher than the MAODV.

Based on the simulation results shown in Fig. 5.3, the packet delivery fraction of PUMA is higher than MAODV for varying number of nodes.

Here, higher End-to-end delay values imply that routing protocol is not fully efficient and causes a congestion in the network. As against the MAODV, PUMA exhibits lesser values of End-to-end delay.

Table 5.1: Performance Comparison of Routing Overhead

Number of nodes	PUMA	MAODV
25	117.84	84.75
50	135.72	120.3
75	138.03	232.36
100	156.71	484.05
125	170.62	697.96
150	191.53	907.68
175	215.24	1207.91

Table 5.2: Performance Comparison of Throughput

Number of nodes	PUMA	MAODV
25	163.96	50.21
50	209.86	69.49
75	400.70	97.29
100	651.07	134.67
125	783.65	187.32
150	940.43	247.78
175	1105.34	498.23

Table 5.3: Performance Comparison of Packet Delivery Fraction

Number of nodes	PUMA	MAODV
25	3.3854	3.0391
50	8.0537	4.3438
75	12.7654	6.2938
100	15.4042	8.6952
125	16.9352	11.3456
150	18.2847	13.1853
175	20.7893	15.2651

Table 5.4: Performance Comparison of End-to-End Delay

Traffic Load	PUMA	MAODV
10	0.01	0.01
20	0.013	0.02
30	0.02	0.028
40	0.025	0.038
50	0.028	0.044
60	0.03	0.052

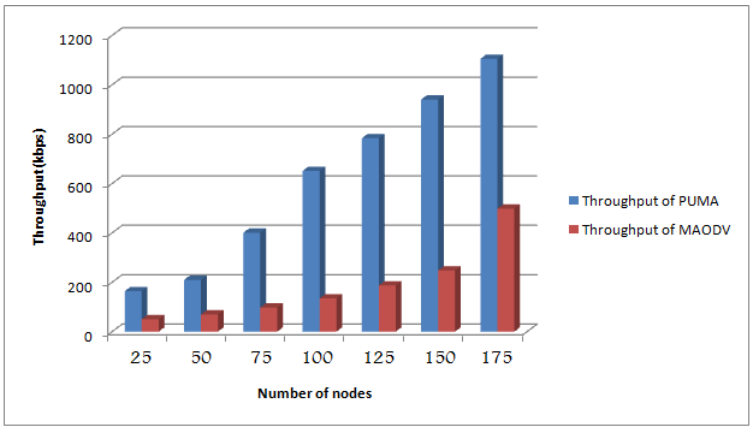


Figure 5.2: Performance Comparison of Throughput

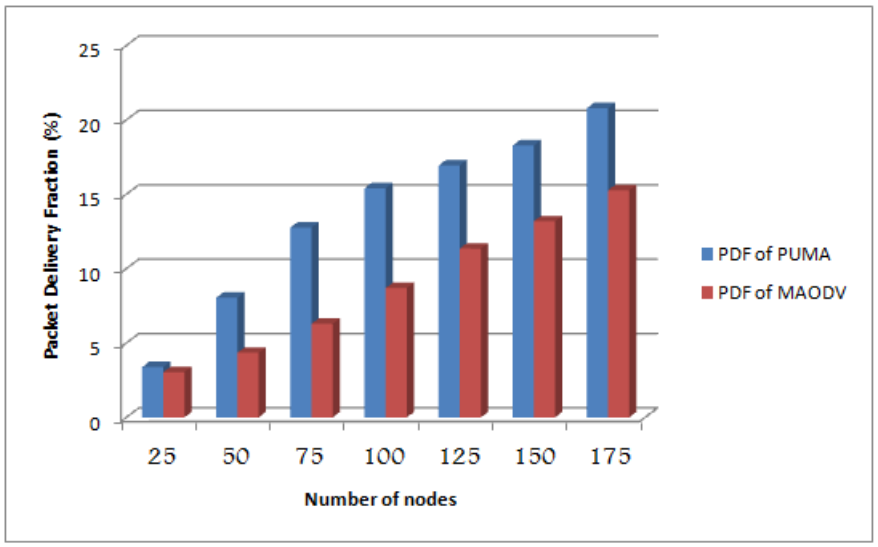


Figure 5.3: Performance Comparison of PDF

5.6 Summary

In this chapter, PUMA incurs far less overhead as compared to MAODV. It has higher packet delivery fraction and throughput. The lesser values of End-to-end delay implies a better performance than other protocol. So, PUMA has been selected for further implementation.

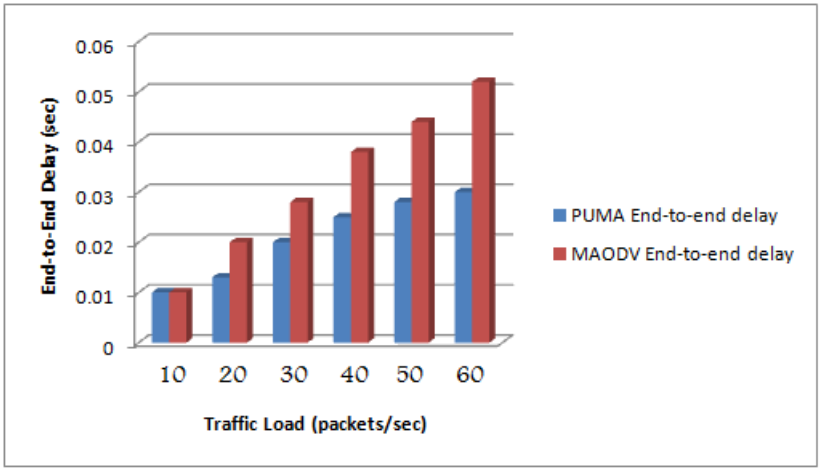


Figure 5.4: Performance Comparison of End-to-End delay

Chapter 6

Security Requirements

Secure communication is a major concern in PUMA, especially because multicasting protocols are applied in areas such as audio/ video conferencing, corporate communications, collaborative and groupware applications. Some of the unique characteristics of ad hoc network that pose a strong challenge to the design of the secure multicast routing protocols include: open peer-to-peer network architecture, shared wireless medium, demanding resource constraints, and dynamic network topology.

To ensure secure communication, the selected cryptography algorithm must have following requirements:

- Confidentiality - Protecting the data from all nodes except the intended receiver.
- Authentication - Proving one's identity.
- Integrity - Ensuring no unauthorized alteration of data.
- Non repudiation - Preventing an entity from denying previous commitments or actions.

Now, one of the major issues in PUMA is how to provide better security that is needed. A proper Cryptography scheme is thus a critical factor for success of PUMA.

Chapter 7

Cryptography

There are many types of Cryptography Schemes to ensure security, such as symmetric key algorithms, asymmetric key algorithms, message digests. Symmetric and asymmetric key algorithms provide secrecy. Message digests are used for authentication.
The purpose of this project was to determine which algorithm performs better for given input data and to conduct a performance analysis of various security algorithms.Symmetric key algorithms use the same key for encryption and decryption. In public key algorithms, the encryption and decryption keys are different. It is not feasible to derive the decryption key from the encryption key. Some of the public key algorithms are RSA and ElGamal.

7.1 Public key Cryptography

In public key algorithms, encryption and decryption keys are different. It is not feasible to derive the decryption key could not feasibly be derived from the encryption key. Public key cryptography requires each user to have two keys: a public key, used by the entire world for encrypting messages to be sent to the user, and a private key, which the user needs for decrypting messages.

7.1.1 RSA

RSA is one of the most popular and successful public-key cryptography algorithms. The algorithm has been implemented in many commercial applications. It is named after its inventorś Ronald L. Rivest, Shamir, and Leonard Adleman. They invented this algorithm

in the year 1977. They utilized the fact that when prime numbers are chosen as a modulus, operations behave "conveniently". They found that if we use a prime for the modulus, then raising a number to the power (prime 1) is 1.RSA algorithm and its security simply capitalize on the fact that there is no efficient way to factor very large integers. The RSA scheme is a block cipher. Each plaintext block is an integer between 0 and n - 1 for some n, which leads to a block size <= log2 (n). The typical block size for RSA is 1024 bits. This algorithm is still in use because of its security and easy implementation. RSA strengths make it most suitable for resource-constrained systems. RSA provides greater security for a given key size and can be efficiently and compactly implemented. These attributes make it well suited for systems with constraints on processor speed, security, power consumption, bandwidth, and memory. The RSA algorithm has been implemented in many applications and it is currently one of the most popularly used encryption algorithm. The security of the RSA algorithm lies in the fact that there is no good way of factoring numbers. No one till now knows a way to factorize a number into its prime factors. As long as no one finds a way RSA will be safe and will be one of the best encryption algorithms in use.

7.1.2 ElGamal

ElGamal is based on the difficulty of solving the discrete logarithm problem. The ElGamal algorithm can be used for both encryption and decryption. The security of the ElGamal scheme relies on the difficulty of computing discrete logarithms over GF(p), where p is a large prime. Prime factorization and discrete logarithms are required to implement the RSA and ElGamal cryptosystems.

7.1.3 Performance Comparison of RSA and ElGamal

We measured the encryption and decryption times of RSA and ElGamal on the client and server and graphs were plotted representing the measured times. In Fig. 6 and Fig. 7, we see the encryption and decryption times increased with the varying number the nodes.

In ElGamal, the ciphertext is twice as long as the plaintext, which is a disadvantage as compared to RSA algorithm. From the figure we see, RSA takes less execution time as compared to ElGamal. So, here RSA is selected for further implementation.

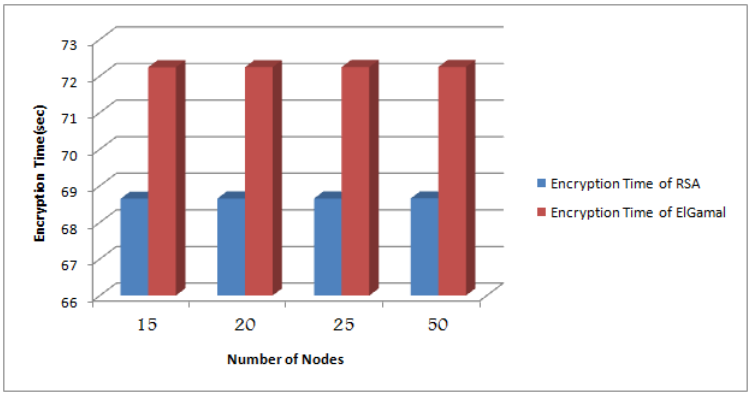


Figure 7.1: Comparison of Encryption time

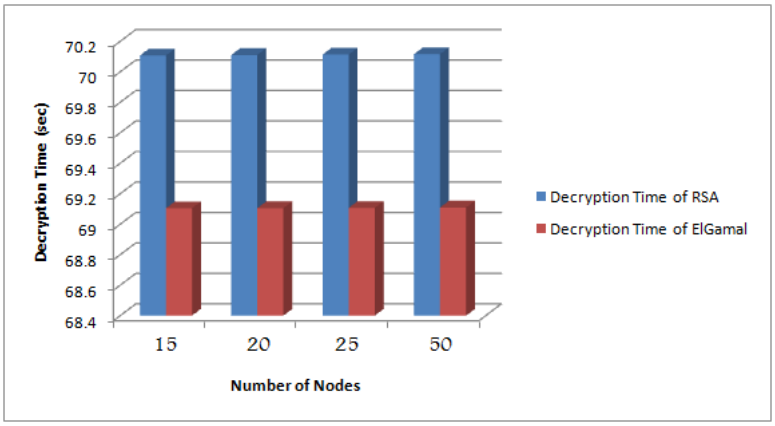


Figure 7.2: Comparison of Decryption time

7.2 Integrity in Multicast Ad hoc Network

Integrity plays an important role in ad-hoc networks. To overcome man-in-the-middle attack in mobile-ad-hoc networks, SHA-1 and MD-5 algorithms are used. Secure hash functions or message digests work on an authentication scheme and do not require encrypting the entire message. SHA-1 and MD5 are algorithms for computing a 'condensed representation' of a message or a data file. The 'condensed representation' is of fixed length and is known as a 'message digest' or 'fingerprint'. Here, SHA-1 generates 160 bits of message digest and MD5 generates 128 bits of message digest.

7.2.1 SHA-1 (Secure Hash Algorithm-1)

SHA1 was developed by the NSA. To overcome man-in-the-middle attack in mobile-ad-hoc networks, SHA-1 algorithm is used. SHA-1 stands for "secure hash algorithm-1". Normally, hop count field is mutable in nature. To protect this hop count value, hash values are found by using SHA-1 algorithm for those fields. Here, the packets are sent along with the hashed values of hop count field. Now, the malicious nodes, which forwards the false routing information, can be effectively defended.

SHA1 processes input data in 512-bit blocks, and generates 160-bit message digests.

Step:1 SHA1 pads the message by adding a 1 bit to the end, followed by as many 0 bits as needed to make the length a multiple of 512 bits.

Step 2: Then a 64-bit number containing the message length before padding is ORed into the low-order 64 bits.

step 3:Then initialize the SHA-1 buffer: (The 160-bit buffer is represented by five four-word buffers (A, B, C, D, E) used to store the middle or finally results of the message digests for SHA-I functions and they are initialized to the following values in hexadecimal.)

step 4:Process message in 16-word blocks (The heart of the algorithm is a module that consists of four rounds of processing 20 steps each.)

step 5:After all 512-bits blocks have been processed, the output of the last block is the 160-bits message digest. These message digest values are sent along with the packets .So, the packets which are sent by malicious nodes are suppressed. Thus, the integrity is ensured.

7.2.2 MD5

MD5 is the fifth in a series of message digests designed by Ronald Rivest. It operates by mangling bits in a sufficiently complicated way so every output bit is affected by every input bit.

MD-5 algorithm performs following steps:-

step 1: It starts out by padding the message to a length of 448 bits. Then the original length of the message is appended as a 64-bit integer to give a total input whose length is a multiple of 512 bits.

step 2:This is followed by as many zeros as are required to bring the length of the message

up to 64 bits fewer than a multiple of 512. The remaining bits are filled up with a 64-bit integer representing the length of the original message.

step 3: Initialize MD Buffer(The processing of a message block consists of four similar stages, termed rounds; each round is composed of 16 similar operations based on a non-linear function F, modular addition, and left rotation. A four-word buffer (h0, h1, h2, h3) is used to compute the message digest. Here each of h0, h1, h2, h3 is a 32-bit register.)

step 4: Process Message in 16-Word Blocks(There is procedure to break the 512 bit chunk message into sixteen 32-bit words.)

step 5: After all 512-bits blocks have been processed, the output of the last block is the 128-bits message digest. These message digest values are sent along with the packets .So, the packets which are sent by malicious nodes are suppressed. Thus, the integrity is ensured.

7.2.3 Working of RSA after binding with SHA-1 and MD5

In further implementation, we have done the binding of SHA-1 and MD5 with RSA and measured the performance of RSA after binding.

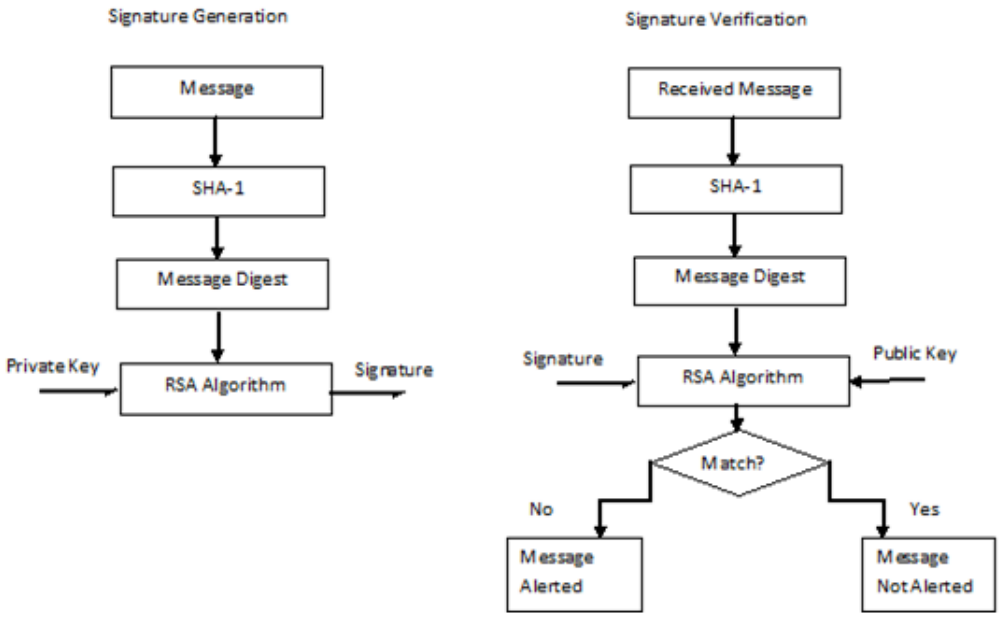


Figure 7.3: Working of SHA-1 after binding with RSA algorithm

Here, the working of SHA-1 and MD5 after binding with RSA is shown in Fig. 8 and Fig. 9 respectively. Working of both the algorithms is similar. Basically the process

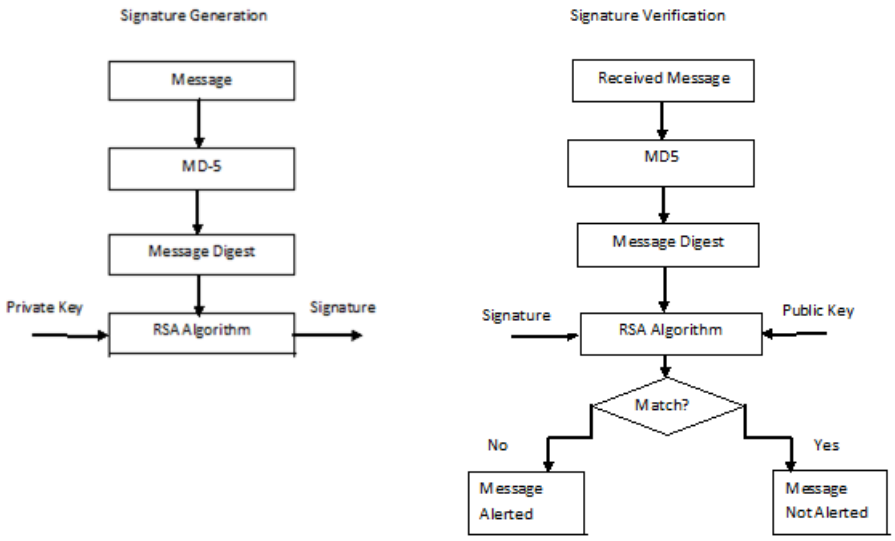


Figure 7.4: Working of MD5 after binding with RSA algorithm

can be implemented in two phases. First phase is signature generation, which is done on encryption side (sender side) and Second is signature verification, which is done on decryption side (receiver side). In signature generation, first the message goes to SHA-1 or MD5 algorithm and then algorithm produces the message digest. After that, message digest goes to RSA algorithm and produces the signature using public key. In second phase signature verification, after receiving message from the sender side, the received message goes to again SHA-1 or MD5 algorithm and produces message digest. Then, message digest goes to RSA algorithm and produces another signature using public key. Now, if signatures at the sender and the receiver side are same, then message is not altered. But if signatures don't match, then the message is altered.

7.2.4 Performance Analysis

The Verification time using RSA and SHA-1 is higher than for RSA and MD5. Verification times for both RSA and SHA-1 and RSA and MD5 increased with increasing key size. This larger digest size of RSA and SHA-1 makes it stronger against attacks. These results suggest RSA and SHA-1 is more secure than RSA and MD5.

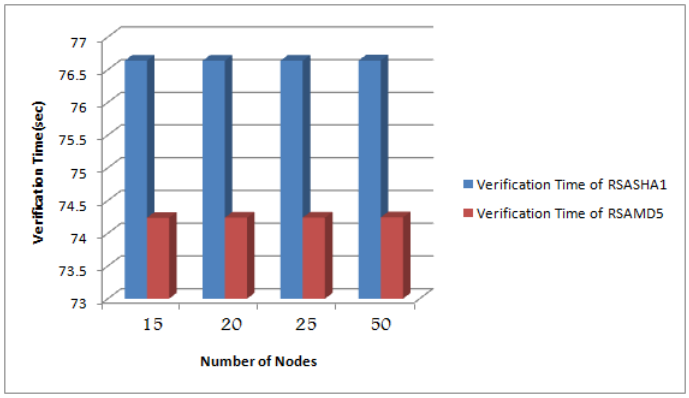


Figure 7.5: Comparison of Verification time

7.3 Summary

This chapter presented the overview of public key cryptography. From the public key cryptography, RSA takes less execution time as compared to ElGamal. So, RSA is selected for further implementation. Data integrity is done by SHA-1 and MD5 algorithms. Based on the performance results, RSA and SHA-1 is more secure as compared to RSA and MD5. In further chapter, RSA and SHA-1 with PUMA is discussed.

Chapter 8

Integration of security in PUMA

After the integration of PUMA with RSA and SHA-1 algorithm, that leads to its secured version PUMA-Safe. PUMA-Safe provides better security as compared to existence PUMA. Here, the Fig. 8.1 and Fig. 8.2 shows the performance results after integration. PUMA-Safe provides more packet delivery fraction as compared to original PUMA as shown in Fig. 8.1. It also provides less end-to-end delay as shown in Fig. 8.2, which leads to better performance of PUMA-Safe.

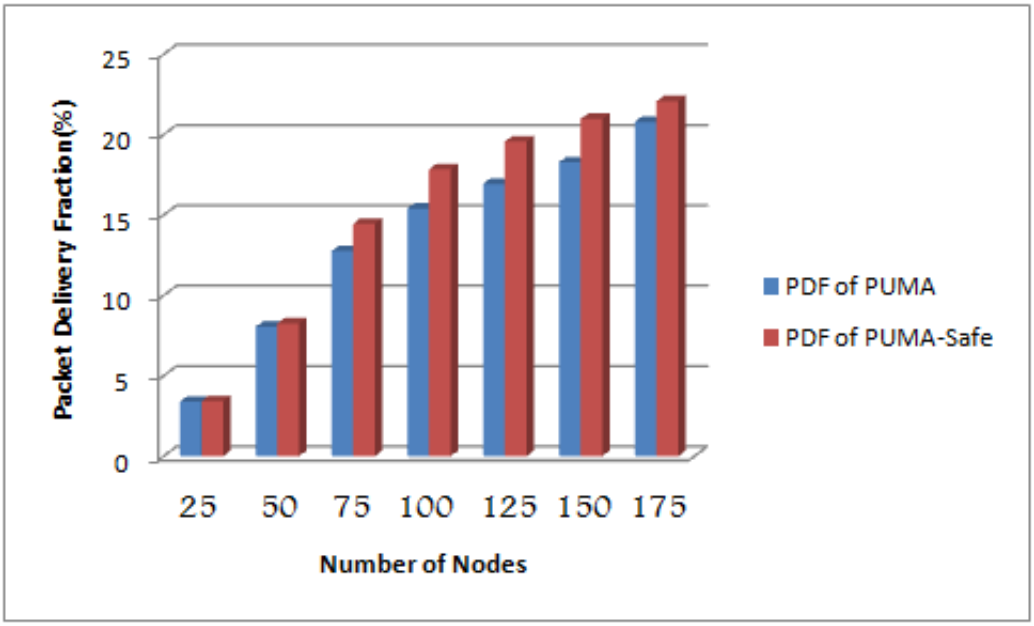


Figure 8.1: Performance comparison of PDF

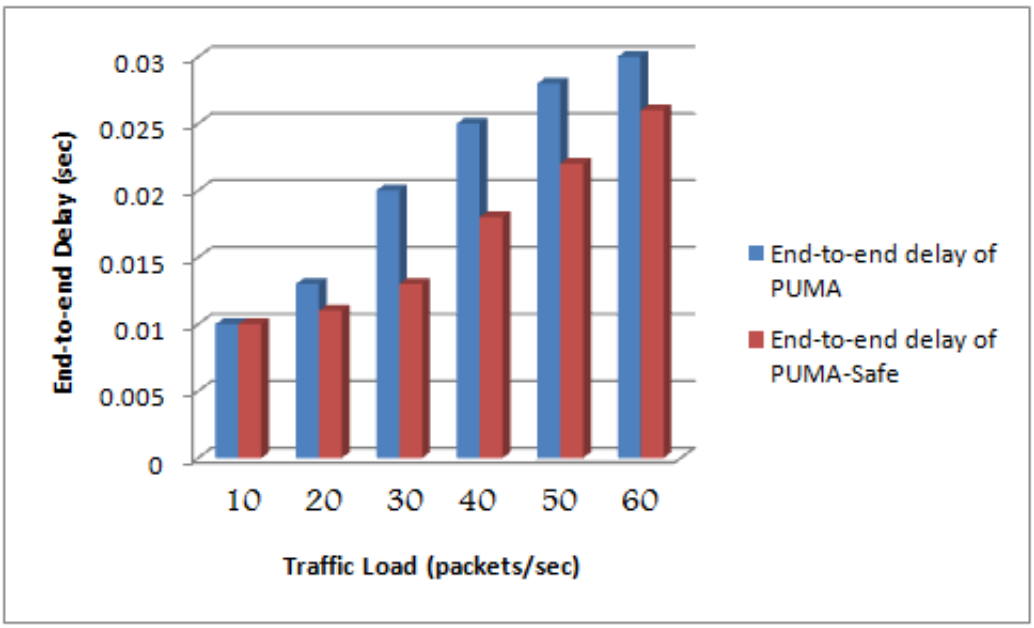


Figure 8.2: Performance comparison of End-to-end delay

Chapter 9

Conclusion and Future work

9.1 Conclusion

PUMA(Protocol for unified multicasting through announcements) is chosen for multicast ad hoc network based on comparison of various multicasting protocols. PUMA provides less routing overhead, maximum throughput and maximum packet delivery fraction as compared to other protocols. RSA and SHA-1 is more secure as compared to RSA and MD5. After integration of PUMA with RSA and SHA-1(PUMA-Safe), it provides better packet delivery fraction and end-to-end delay as compared to PUMA.

9.2 Future Work

Future work will mainly focus on the extension of PUMA-Safe to provide security from more attacks.

Web References

[17] http://mohit.ueuo.com/AWK_Scripts.html

[18] http://www.isi.edu/nsnam/ns/doc/node184.html

[19] http://en.wikipedia.org/wiki/List_of_ad_hoc_routing_protocols

[20] http://en.wikipedia.org/wiki/List_of_ad_hoc_routing_protocols

[21] web2.uwindsor.ca/courses/cs/aggarwal/cs60375/NS/NS2.ppt

[22] http://www.cubinlab.ee.unimelb.edu.au/~jrid/Docs/Manuel-NS2/node234.html

[23] http://CryptographicServices/92f9ye3s.aspx.html

[24] http://en.wikipedia.org/wiki/messagedigest

References

[1] Busola S. Olagbegi, Natarajan Meghanathan, "A Review of the Energy Efficient and Secure Multicast Routing Protocols for Mobile Ad hoc Networks ", - (GRAPH-HOC) Vol.2, No.2, June 2010 1,2Jackson State University, 1400 Lynch St, Jackson, MS, USA.

[2] R. Vaishampayan, J.J. Garcia-Luna-Aceves, "Energy Efficient and Robust Multicast Routing in Mobile Ad Hoc Networks",Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems, 2004.

[3] Ravindra Vaishampayan,"Efficient and Robust Multicast Routing in Mobile Ad Hoc Networks",-.University Of California Santa Cruz March 2006.

[4] A. M. Hegland, E. Winjum-University Graduate Center at Kjeller," Evaluation of Key Management in ad hoc Networks for Emergency and Rescue Operations"

[5] Zhou, L. and Haas, "Securing Ad Hoc Network. IEEE Networks, Vol. 13, No. 6.

[6] J-P. Hubaux, L. Buttyn and S. Capkun,"The Quest for Security in Mobile Ad Hoc Net- works, ACM.", 2001

[7] Zheng,S.R. 2004, "The Technique of Ad Hoc Network, Posts & Telecom press."

[8] Thomas Kunz and Ed Cheng, "Multicasting in Ad-Hoc Networks: Comparing MAODV and ODMRP,", Carleton University.

[9] E. Cheng, "On-demand multicast routing in mobile ad hoc networks", M.Eng. thesis, Carleton University, Department of Systems and Computer Engineering, 2001.

[10] Brian Neil Levine, J.J. Garcia-Luna-Aceves, "A comparison of reliable multicast protocols", - Computer Engineering Department, School of Engineering, University of California, Santa Cruz, Santa Cruz, CA 95064 USA - Multimedia Systems c Springer-Verlag 1998

[11] H.Lou and s.Lu,"Ubiquitous and Robust Authentication Services for Ad hoc Wireless Networks", Technical Report, 2000

[12] Den Boer B, Bosselaers A.,"Collisons for the compression function of MD5, Advances in cryptography",EUROCRYPT'93,p-293-304.

[13] Schneier Bruce, "Opinion:Cryptanalysis of MD5 ans SHA: Time for a new standard", April 19,2004.

[14] Levent Ertaul, Nitu Chavan, "Security of Ad Hoc Networks and Threshold Cryptography," California State University, Hayward.

[15] Marianne A. Azer,Sherif M. El-Kassas,Magdy S. El-Soudani, "SECURITY SCHEMES IN AD HOC NETWORKS A SURVEY AND NEW CHALLENGES",

[16] Jianping Wang, ns-2 Tutorial (2), Multimedia Networking Group, The Department of Computer Science, UVA

Index