Efficient Bandwidth in Mobile Ad Hoc Networks Using Genetic Algoritham

A.Anusha, CH.Veera Babu

Abstract*: Most of the existing routing protocols are designed primarily to carry best effort traffic and only concerned with shortest path routing. Little attention is paid to the issues related to the quality of services (QoS) requirement of a route. In this paper, we will consider the problem of searching for a route satisfying the bandwidth requirement in a mobile ad-hoc network. Unlike in a wired network, where the available bandwidth of a route is simply the minimum bandwidth of the links along the route, the calculation of the available bandwidth of a route in a mobile ad-hoc network has been proved to be complete. The Genetic Algorithm (GA) has successfully been applied to many famous Application problems in communication networks, such as the multicast routing problem. Recently, many researchers have attempted to adopt genetic algorithms to solve various problems existing in mobile ad hoc networks. This Genetic Algorithm executed in a centralized manner for the bandwidth calculation problem in the TDMA channel model. Extensive computer simulations are performed to compare the performance of our proposed GA method and that of other existing heuristic algorithms. Simulation results verify that our GA can produce larger bandwidth utilization than others.

I. INTRODUCTION

The research of MANETs has attracted a lot of attention recently. In particular, since MANETs are characterized by their fast changing topologies, extensive research efforts have been devoted to the design of routing algorithms/protocols for MANETs. A number of efficient routing algorithms/protocols have been developed. However, most of the existing routing algorithms/protocols have been designed primarily to carry best-effort traffic and only concerned with shortest-path routing. Little attention is paid to the issues related to the QoS requirements of a path. Only recently have people turned their attention to establish paths with QoS given in terms of bandwidth and delay. The purpose of this study is to search for a path that would meet the bandwidth requirements in a TDMA-based MANET.

To provide guaranteed bandwidth for the transmission of packets along a path from a source to a destination, the required amount of bandwidth needs to be reserved along the entire path. Before the required bandwidth of a path can be reserved, the available bandwidth along the path has to be calculated in order to determine whether the current bandwidth is enough. The BWC problem in a MANET is much more difficult than that in a traditional wired network. This is because the bandwidth in a MANET is usually shared among adjacent hosts due to the broadcast nature of the wireless medium, which is subject to collisions, losses, and the potential topological changes caused by host mobility. In fact, the calculation of the available bandwidth of a path in TDMA-based MANET is NP-complete. The objective of this study is to design an efficient heuristic algorithm to calculate the maximal available bandwidth along a path in a TDMA-based MANET

In a TDMA-based MANET, the transmission time scale is organized in frames of fixed length. Each frame consists of a fixed no. of time slots. The time slots are used for the transmission of the packets. Only one data packet used for the transmitted in each time slot. The bandwidth requirement is realized by reserving time slots on links. The half-duplex operation mode is assumed i.e., a host cannot be in in the transmitting mode and the receiving mode at the same time. For a given host, each slot in a frame can be marked with either “x”, implying that the slot is free, i.e., it is not used by any adjacent host of the host to receive or send packets. The slots that are free at both of the two ending host of the link are said to be the common free slots of the link. The available bandwidth of a link (the link bandwidth) is defined as the number of common free slots of the link.

Link Bandwidth, which is defined as the number of common free slots between its two ending hosts, the bandwidth of a path (path bandwidth) between two hosts can be defined as the no. of available slots between two hosts, However, unlike a wired network, where the bandwidth of the path is simply the minimum link bandwidth along the path, the finding of the available slots of a path in a TDMA-based MANET is NP-complete.

A weighted graph G=(V,E) is used to denote a MANET, where the node set V represents the mobile hosts, and the edge set E represents the links between nodes. Each frame in the MANET consists of m2 time slots; F indicates that the corresponding slot is free at both of the two ending nodes of the link.

Objective

The objective of this study is to design an efficient heuristic algorithm to calculate the maximal available bandwidth along a path in a TDMA-based MANET.

1.2 Existing System

Most of the existing routing algorithms/protocols in mobile ad hoc networks (MANETs) have been designed primarily to carry

Revised Manuscript Received on February 05, 2012.

Annadatha Anusha:computer science, JNTU(KAKINADA)/Stan’s engineering college:, Chirala, Country INDIA, 08147044396 No., LAKSHMIANUSHA25@GMAIL.COM

CH.Veera babu, :computer science, JNTU(KAKINADA)/Stan’s engineering college,:chirala,:India, 09963834485., veeruchinka”gmail.com

best-effort traffic and only concerned with shortest-path routing. Little attention is paid to the issues related to the quality-of-service (QoS) requirements of a path. In a TDMA-based MANET, as only a single transmission code is uniformly used, transmissions from different hosts are very likely to collide.

1.3 Problems in Existing System

i. The Quality of the service is less.
ii. Only single transmission code is used
iii. The collisions may occur due to the single transmission code.

1.4 Proposed System

In this study, searching for a path to satisfy bandwidth requirements in a time-division-multiple-access-based (TDMA–based) MANET has been the main goal. This is an NP-complete problem.

i. The genetic algorithm (GA) has been successfully applied to many well-known NP-complete problems in communication networks, such as the multicast routing problem.

ii. In this study, the principle of GA is used to design an efficient heuristic algorithm, which is executed in a centralized manner, for the bandwidth calculation (BWC) problem in a TDMA-based MANET.

II. SYSTEM ANALYSIS

2.1 Introduction

The research of MANETs has attracted a lot of attention recently. In particular, since MANETs are characterized by their fast changing topologies, extensive.

2.2 Study of the System

Modules Description

The main modules involved in the project are

Modules:

i. Representation of the Chromosome
ii. Input and the First Generation of Chromosomes
iii. Evaluation of a Chromosome
iv. Termination Judgment
v. Reproduction
vi. Crossover Operation on the Selected Chromosomes
vii. Mutation Operation on the Selected Genes

The description of the modules is given below

i.Representation of the Chromosome

I n the GA for the BWC problem, a chromosome is represented by a string of integers with length k = m1 × m2. The value of each gene in a chromosome (i.e., each entry in the matrix) is 0, 1, or 2. An entry with value 0 (1, 2) means its corresponding slot is unreservable (reservable, unavailable).

ii. Input and the First Generation of Chromosomes

The input of the GA is the common free slot matrix of a given path. The matrix is first transformed to a set of chromosomes forming the initial generation. The transformation rule is as follows: Scan every entry in the given common free slot matrix first from left to right, and then from top to down. If an entry is marked with "×," then its value is re-assigned to 2; otherwise, its value is randomly re-assigned to either 0 or 1.

iii. Termination Judgment

The purpose of termination judgment is to determine if the GA has reached the terminal condition. The terminal condition of the proposed GA is a pre-defined value on the number of generations.

2.3 Feasibility Report

i.Economic Feasibility

Economic feasibility attempts 2 weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

ii.Operational Feasibility

Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation?

Here are questions that will help test the operational feasibility of a project:

Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.

Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.

iii.Technical Feasibility

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, .at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis.

III. SOFTWARE REQUIREMENTS SPECIFICATION

SOFTWARE REQUIREMENTS SPECIFICATION

A software requirements specification is developed as a consequence of analysis. Review is essential to ensure that the developer and customers have the same perception.

Software requirements specification (SRS) is the starting point of the software development activity. The Software Requirements Specification is

produced at the culmination of the analysis task. The introduction of the software requirements specification states the goals and objectives of the software, describing it in the context of the computer-based system. The SRS includes an information description, functional description, behavioral description, validation criteria.

A requirement is a statement about what the proposed system will do that all stakeholders agree must be made true in order for the customer's problem to be adequately solved. Requirements can be divided into two major types, functional and non-functional. Requirements documents normally include both.

3.1 Hardware Requirements:

Hard Disk : 4GB
Processor : Pentium Celeron Onwards
RAM : 128 MB
Recommendable –
Hard Disk : 4GB
Processor : Pentium Celeron Onwards
RAM : 512 MB

Software Requirements:

Language : Java
Operating System : Windows XP.

3.2 Functional Requirements

Performance Requirements

This part of the SRS specifies the performance constraints on the software system, specifying the requirements relating to the performance characteristics of the system.

Two types of performance requirements*:*

1. Static Requirements
2. Dynamic Requirements

Static Requirements are those that do not impose constraint on the execution characteristics of the system. These include requirements like the number of terminals to be supported, the number of simultaneous users to be supported, and the number of files that the system has to process and their sizes.

Dynamic Requirements are which specify constraints on the execution behavior of the system. These typically include response time and throughput constraints on the system. Response time is the expected time for the completion of an operation under specified circumstances. Throughput is the expected number of operations that can be performed at a time.

3.3 Non Functional Requirements:

SECURITY:

This application is a secure One. That is the system cannot be used by a third party who was not a registered one or else an authorized one.

The Application also manages all the required Security features incorporated in the database with required verifications and password checks.

RELIABILITY:

The proposed system is a reliable one too. That is once the recruiter poses a request or else a job seeker post details, with in no matter of time they will get the response. The Application Software is very reliable in handling data. It been designed such that, it even handles junk/ garbage data and has some robust options like code protection, error trapping etc.

SCALABILITY:

The recently developed system can be easily extended in future without much modification to the system, which makes it easily scalable

SAFETY:

This system was designed in such a way that everything will be on the safe side even during the situations causing disturbances.

PRIVACY:

The system is very much private because it is available to only those who have been registered with the system.

ADAPTABILITY:

Oracle provides options to communicate with existing systems for data migration on ease.

CONSISTENCY:

Advantages of using a consistent format–front end and interactive module to be emphasized.

MAINTAINABILITY:

The application software has a flexible design enabling future up gradation as per the running requirement

USER FRIENDLINESS:

The front-end design has a very user-friendly interface. It is very easy for a native user to understand and operate. The major operational functions are integrated into general keys for easy interaction with the software.

IV. SYSTEM DESIGN

4.1 INTRODUCTION:

Design is essentially a bridge between requirement specification and the final solution for specifying the requirements. The goal of the design process is to produce a model or representation of a system, which can be used later to build the system. The design of a system is correct if a system built precisely according to the design satisfies the requirements of that system. A design should clearly be verifiable, complete (implement all the specifications) and traceable (all design elements can be traced to some requirements). However, the two most important things that concern designers are efficiency and simplicity. Efficiency of any system is concerned with the proper use of scarce resources by the system.

System Design specification forms the third stage of system development process. System Design involves the development of internal design specifications to accomplish the functions stated in the System Requirement Specification (SRS), i.e., it translates the functional requirements to make things operation. The system design is a solution to proper approach. This will provide a comprehensive definition of the system internals with respect to oriented functions to be performed. Architecture defined as the grouping of functions into programming modules. The purpose of the document is to present the functional requirements in a clear, concise and unambiguous manner. The idea behind such a document is to give

the programmer a clear picture of what exactly is to be done in the development process.

The benefits of System Design Specifications are:

- It reduces software development effort.
- It resolves major design consideration.
- It provides a well-organized baseline for changing during implementation and maintenance.

Technical Criteria for Design:

- A design should exhibit a hierarchical organization that makes intelligent use of control among elements of software.
- A design should be modular, i.e. the software should be logically portioned into elements that perform specific functions and sub functions.
- A design should contain both data and procedural abstractions.
- A design should lead to modules that exhibit independent functional characteristics
- A design should lead to interface that reduce the complexity of connection between modules and with the external environment.

4.2 Design Principles:

Software design is both a process and a model. The design process is a set of iterative steps that enable the designer to describe all aspects of the software to be built. Basic design principles enable the software engineer to navigate the design process. Some of the design principles are

- The design should be traceable
- The design should not reinvent the wheel
- The design should "minimize the intellectual distance between system and the problem as it exists in the real world".
- The designer should exhibit uniformity and integrity.

V. DETAILED DESIGN

Logical design

Software is fundamental characteristic of computer software. Software structure is to decompose the complex groups of module into sub modules i.e., Process, Menus, Inputs and Reports.

Since the low level design concentrates on various input and outputs that are related to a process, it is emphasized that each form or a process that is appearing in the project module should be identified for the various data coming into the form and the data that in being generated as output of the screen. Hence then low level design is done by identifying various tables, their columns where they form the required outputs. Since the present system is designed to make user friendlier and to avoid ambiguity, where in many processes exist.

Conceptual Design

Conceptual design is the process of acquiring and evaluating documenting and then validating what the user envisions being the business relation. It identifies the user and business requirements of the application and leads to a business solution as seen by the user.

Physical Design

The purpose of physical design is to translate the logical design into a solution that can be implemented effectively, according to performance, administration and development process requirements .This physical view should correctly implement the desired system behavior while meeting the constraints imposed by the technology.

4.3 UML Diagrams

(Unified Modeling Language)

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

- **User Model View:**

i. This view represents the system from the user's perspective.
ii. The analysis representation describes a usage scenario from the end-users perspective.

- **Structural model view:**

i. In this model the data and functionality are arrived from inside the system.
ii. This model view models the static structures.

- **Behavioral Model View:**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

- **Implementation Model View:**

In this the structural and behavioral as parts of the system are represented as they are to be built.

Environmental Model View:

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

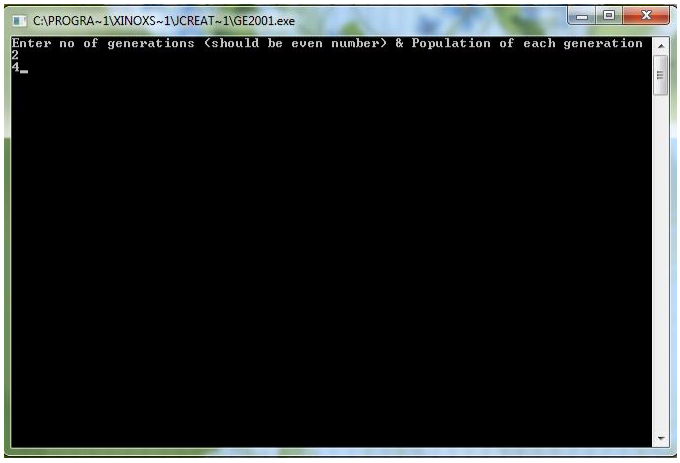
UML is specifically constructed through two different domains they are:

i. UML Analysis modeling, this focuses on the user model and structural model views of the system.
ii. UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

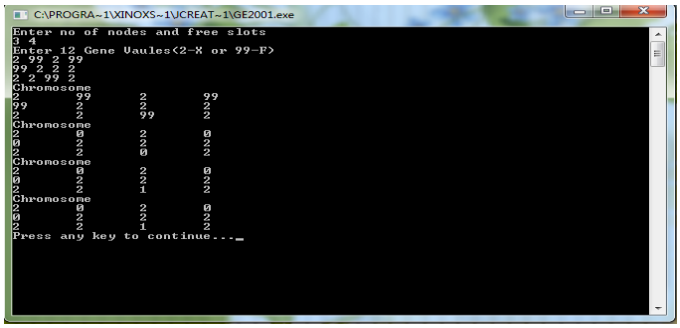
- **Use case Diagrams** represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.
- **Actors** are external entities that interact with the system. Examples of actors include users like administrator, Employee …etc., or another system like central database.

VI. SAMPLE SCREENS

Input for the no. of Generations and Population of Each Generation:

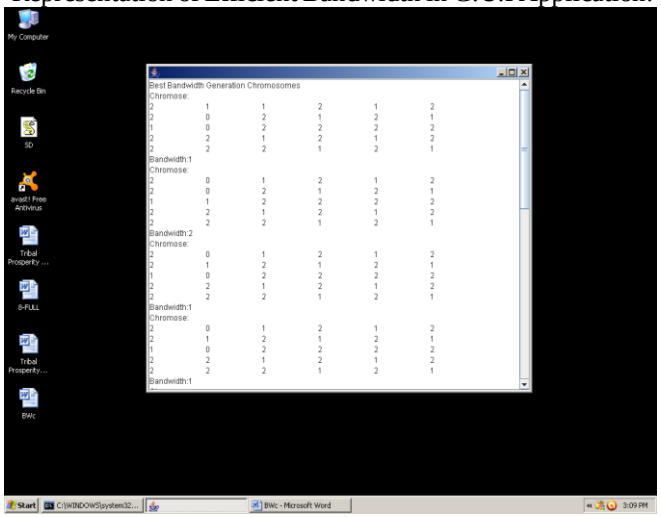


Input for no of nodes &Free Slots:

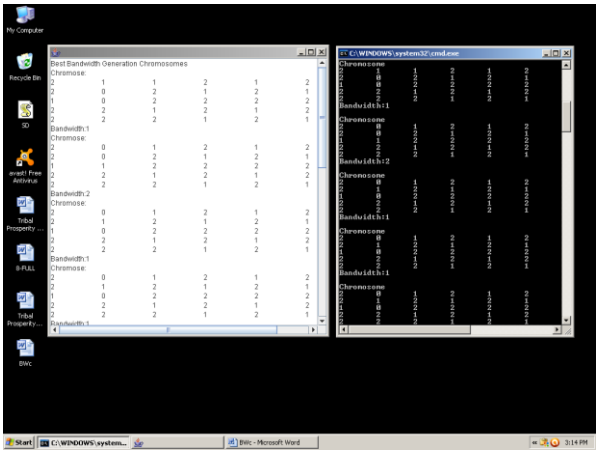


Representation of Efficient Bandwidth in Console Application:

Representation of Efficient Bandwidth in G.U.I Application:



Representation of Efficient Bandwidth in G.U.I & Console Applications:



VII. SYSTEM TESTING

Testing

Software Testing is the process used to help identify the correctness, completeness, security, and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors. Quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and behavior of the product against a specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation, not merely a matter of creating and following routine procedure. One definition of testing is "the process of questioning a product in order to evaluate it", where the "questions" are operations the tester attempts to execute with the product, and the product answers with its behavior in reaction to the probing of the tester[citation needed]. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product—putting the product through its paces. Some of the common quality attributes include capability, reliability, efficiency, portability, maintainability, compatibility and usability. A good test is sometimes described as one which reveals an error; however, more recent thinking suggests that a good test is one which reveals information of interest to someone who matters within the project community.

7.1 Introduction

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually

manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

Software testing may be viewed as a sub-field of Software Quality Assurance but typically exists independently (and there may be no SQA areas in some companies). In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the amount of faults that end up in the code or deliver faster.

Regardless of the methods used or level of formality involved the desired result of testing is a level of confidence in the software so that the organization is confident that the software has an acceptable defect rate. What constitutes an acceptable defect rate depends on the nature of the software. An arcade video game designed to simulate flying an airplane would presumably have a much higher tolerance for defects than software used to control an actual airliner.

A problem with software testing is that the number of defects in a software product can be very large, and the number of configurations of the product larger still. Bugs that occur infrequently are difficult to find in testing. A rule of thumb is that a system that is expected to function without faults for a certain length of time must have already been tested for at least that length of time. This has severe consequences for projects to write long-lived reliable software.

A common practice of software testing is that it is performed by an independent group of testers after the functionality is developed but before it is shipped to the customer. This practice often results in the testing phase being used as project buffer to compensate for project delays. Another practice is to start software testing at the same moment the project starts and it is a continuous process until the project finishes.

Another common practice is for test suites to be developed during technical support escalation procedures. Such tests are then maintained in regression testing suites to ensure that future updates to the software don't repeat any of the known mistakes.

It is commonly believed that the earlier a defect is found the cheaper it is to fix it.

In counterpoint, some emerging software disciplines such as extreme programming and the agile software development movement, adhere to a "test-driven software development" model. In this process unit tests are written first, by the programmers (often with pair programming in the extreme programming methodology). Of course these tests fail initially; as they are expected to. Then as code is written it passes incrementally larger portions of the test suites. The test suites are continuously updated as new failure conditions and corner cases are discovered, and they are integrated with any regression tests that are developed.

Unit tests are maintained along with the rest of the software source code and generally integrated into the build process (with inherently interactive tests being relegated to a partially manual build acceptance process).

The software, tools, samples of data input and output, and configurations are all referred to collectively as a test harness.

A. White-box and black-box testing

To meet Wikipedia's quality standards, this section may require cleanup. Please discuss this issue on the talk page, and/or replace this tag with a more specific message.

White box and black box testing are terms used to describe the point of view a test engineer takes when designing test cases. Black box being an external view of the test object and white box being an internal view. Software testing is partly intuitive, but largely systematic. Good testing involves much more than just running the program a few times to see whether it works. Thorough analysis of the program under test, backed by a broad knowledge of testing techniques and tools are prerequisites to systematic testing. Software Testing is the process of executing software in a controlled manner; in order to answer the question “Does this software behave as specified?” Software testing is used in association with Verification and Validation. Verification is the checking of or testing of items, including software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques as reviews, inspections, walk-through. Validation is the process of checking what has been specified is what the user actually wanted.

- Validation: Are we doing the right job?
- Verification: Are we doing the job right?

In order to achieve consistency in the Testing style, it is imperative to have and follow a set of testing principles. This enhances the efficiency of testing within SQA team members and thus contributes to increased productivity. The purpose of this document is to provide overview of the testing, plus the techniques.

At SDEI, 3 levels of software testing is done at various SDLC phases

- Unit Testing: in which each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented.
- Integration testing: in which progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a whole.
- System testing: in which the software is integrated to the overall product and tested to show that all requirements are met

A further level of testing is also done, in accordance with requirements:

- Acceptance testing: upon which the acceptance of the complete software is based. The clients often do this.

A SQL query on the database to be certain of the values It is used almost exclusively of client-server testers or others who use a database as a repository of information, but can also apply to a tester who has to manipulate XML files (DTD or an actual XML file) or configuration files directly. It can also be used of testers who know the internal workings or algorithm of the software under test and can write tests specifically for the anticipated results. For example, testing a data warehouse implementation involves loading the target database with information, and verifying the correctness of data population and loading of data into the correct tables.

B. 7.2 Test levels

- Unit testing tests the minimal software component and sub-component or modules by the programmers.
- Integration testing exposes defects in the interfaces and interaction between integrated components (modules).
- Functional testing tests the product according to programmable work.
- System testing tests an integrated system to verify/validate that it meets its requirements.
- Acceptance testing can be conducted by the client. It allows the end-user or customer or client to decide whether or not to accept the product. Acceptance testing may be performed after the testing and before the implementation phase. See also Development stage
    - Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.
    - Beta testing comes after alpha testing. Versions of the software, known as beta versions, are released to a limited audience outside of the company. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

It should be noted that although both Alpha and Beta are referred to as testing it is in fact use emersion. The rigors that are applied are often unsystematic and many of the basic tenets of testing process are not used. The Alpha and Beta period provides insight into environmental and utilization conditions that can impact the software.

After modifying software, either for a change in functionality or to fix defects, a regression test re-runs previously passing tests on the modified software to ensure that the modifications haven't unintentionally caused a regression of previous functionality. Regression testing can be performed at any or all of the above test levels. These regression tests are often automated.

C. 7.3 Test cases

A test case is a software testing document, which consists of event, action, input, output, expected result and actual result. Clinically defined (IEEE 829-1998) a test case is an input and an expected result. This can be as pragmatic as 'for condition x your derived result is y', whereas other test cases described in more detail the input scenario and what results might be expected.

It can occasionally be a series of steps (but often steps are contained in a separate test procedure that can be exercised against multiple test cases, as a matter of economy) but with one expected result or expected outcome. The optional fields are a test case ID, test step or order of execution number, related requirement(s), depth, test category, author, and check boxes for whether the test is automatable and has been automated. Larger test cases may also contain prerequisite states or steps, and descriptions. A test case should also contain a place for the actual result. These steps can be stored in a word processor document, spreadsheet, database or other common repository. In a database system, you may also be able to see past test results and who generated the results and the system configuration used to generate those results. These past results would usually be stored in a separate table.

VIII. CONCLUSION

CONCLUSION

This system has been computed successfully and was also tested successfully by taking “test cases”. The project is developed using Java in Windows environment. The goal that achieved Optimum utilization of resources and less processing time.

- In this study, the principle of ga is used to design an efficient heuristic algorithm, which is executed in a centralized manner, for the bandwidth calculation (BWC) problem in a TDMA-BASED MANET.
- Extensive computer simulations are performed to compare the performance of the proposed GA with that of another heuristic algorithm. Simulation results verify that the GA can produce a larger bandwidth.

Unlike in a wired network, where the available bandwidth of a path is simply the minimum link bandwidth along the path, the calculation of the available bandwidth of the path in a TDMA-Based-MANET is NP-complete. Based on the principle of GA, in this study, an efficient centralized heuristic algorithm is designed to solve the BWC problem in a TDMA-Based-MANET. Extensive computer simulations are performed to compare the performance of the new GA with

that of the forward algorithm. Simulations results verify that the proposed GA can produce a larger bandwidth.

IX. FUTURE ENHANCEMENTS

FUTURE ENHANCEMENTS

- The above project is able to perform some operations that to for identification of those names of original copy that to be protected of copyright of our own application.
- To rectify the BANDWIDTH problem in MANET, Create the connection dynamically.
- These to be developed and implemented in audio and video application. In these application we are using RSA algorithm that to develop the entire application.

X. BIBLIOGRAPHY

BIBLIOGRAPHY

1. 1 .Banerjee N, and Das, S.K., 2001, "MODERN: Multicast on-Demand QoS-based Routing in Wireless Networks" Proceedings of the IEEE VTS 53rd Vehicular Technology Conference.
2. Chen S. and Nahrstedt, K, "Distributed Quality of Service Routing in Adhoc Networks", Proceedings of the IEEE International Conference On Communications.
3. Gen M. and Cheng R., Genetic Algorithms and Engineering Design John Wiley and Sons.
4. Lin H.C. and Fung, P.C., "Finding Available Bandwidth in Multihop Mobile Wireless Networks" Proceedings of the IEEE VTS 51rd Vehicular Technology Conference.