Mobile Ad Hoc Networks Routing Misbehaviors and their Detections

Dr Sudan Jha
Professor, School of Computer Engineering
KIIT University, Patia
Bhubaneswar, Odisha, India

Abstract—In general, routing protocols for MANETs (Mobile Ad Hoc Networks) are designed based on the assumption that all participating nodes are fully cooperative. However, due to the open structure and scarcely available battery-based energy, node misbehaviors may exist. One such routing misbehavior is that some selfish nodes will participate in the route discovery and maintenance processes but refuse to forward data packets. In this paper, we propose the 2ACK scheme that serves as an add-on technique for routing schemes to detect routing misbehavior and to mitigate their adverse effect. The main idea of the 2ACK scheme is to send two-hop acknowledgment packets in the opposite direction of the routing path. In order to reduce additional routing overhead, only a fraction of the received data packets are acknowledged in the 2ACK scheme. Analytical and simulation results are presented to evaluate the performance of the proposed scheme.

Keywords- MANET, Mobile Ad Hoc Networks, Networks, Routing

I. Introduction

The system is acknowledgement based approach for detection of routing misbehavior in MANET. MANET is expanded as Mobile Ad-hoc Network which is a collection of mobile nodes which communicate with each other via wireless links. In between nodes there will be routers. When packets are transmitted, from one node to another node, the packets from source to destination may be transmitted through intermediate nodes. For transmitting packets from source to destination there may be several routes. In this context routing is important operation in the function of MANET. The operation of MANET also doesn't depend on existing infrastructure base stations. Thus routing is dynamic in MANET. Since it is a wireless network, network topology rapidly changes. Now, routing of packets becomes more complex in this MANET which is large scale mobile dynamic network. Even in MANET there may be two types:

1. Closed MANET
2. Open MANET

In closed MANET all nodes co-operate each other with common goals. In open MANET there is no coordination among nodes. This again increases complexity of routing. In this environment of complex routing there may be routing misbehavior. Routing misbehavior means the direction of routing to destination suddenly changes in a wrong way. Now, we are introducing a system which detects misbehavior more efficiently and accurately with acknowledgement. This scheme is also called 2ACK scheme. The idea is when a packet forwards to next hop successfully the next hop will send an acknowledgement to the previous node to indicate the route is correct up to so far. This system is implementing 2ACK scheme using DSR protocol.

1.1 Purpose

The main purpose of the system is detection of routing misbehavior in a complex large scale heterogeneous mobile dynamic network in transmitting data packets from source to destination. It also aims to detect misbehavior of routing more efficiently, accurately, fastly. Its objective and main purpose is low false alarm rate, low missed detection rate. The purpose of using 2ACK scheme is to enhance the effect of DSR protocol.

1.2 Existing System

The existing system works with DSR algorithm in which we have the overhead in sending full data packet as acknowledgement. This can be eliminated in our proposed system. And in this we also detect the link misbehaviors.

1.3 Proposed System

In our proposed system we are using a special scheme called 2ACK in order to detect the routing misbehaviors in mobile ad-hoc networks. This scheme is an add on technique to the dsr algorithm.

1.4 Project Scope

The scope of this system is to provide efficient, accurate routing of data packets in heterogeneous mobile dynamic network MANET, so that the users working at different workstations in MANET cannot suffer from congestion and delay in transmission.

II. Literature Review

2.1 Introduction

A Review of Literature is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it's worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts in Literature Survey.

2.2 Technical Survey

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. Understand the different technologies involved in the proposed system before commencing the project we have to be very clear about what are the technologies that are to be required for the development of the new system. Find out whether the organization currently possesses the required technologies. Is the required technology available with the organization?

2.3 Operational Survey

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. Have the user been involved in the planning and development of the project? Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project. Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

2.4 Economic Survey

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.

2.5 The 2ACK scheme

The watchdog detection mechanism has a very low overhead. Unfortunately, the watchdog technique suffers from several problems such as ambiguous collisions, receiver collisions, and limited transmission power. The main issue is that the event of successful packet reception can only be accurately determined at the receiver of the next-hop link, but the watchdog technique only monitors the transmission from the sender of the next-hop link. Noting that a misbehaving node can either be the sender or the receiver of the next-hop link, we focus on the problem of detecting misbehaving links instead of misbehaving nodes. In the next-hop link, a misbehaving sender or a misbehaving receiver has a similar adverse effect on the data packet: It will not be forwarded further. The result is that this link will be tagged. Our approach discussed here significantly simplifies the detection mechanism.

Details of the 2ACK Scheme

The 2ACK scheme is a network-layer technique to detect misbehaving links and to mitigate their effects. It can be implemented as an add-on to existing routing protocols for
MANETs, such as DSR. The 2ACK scheme detects misbehavior through the use of a new type of acknowledgment packet, termed 2ACK. A 2ACK packet is assigned a fixed route of two hops (three nodes) in the opposite direction of the data traffic route.

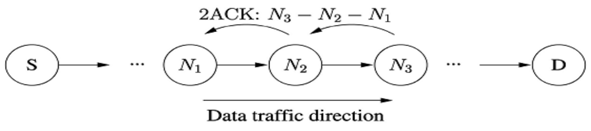


Fig 2.1 the 2ACK scheme

Suppose that N1, N2, and N3 are three consecutive nodes (triplet) along a route. The route from a source node, S, to a destination node, D, is generated in the Route Discovery phase of the DSR protocol. When N1 sends a data packet to N2 and N2 forwards it to N3, it is unclear to N1 whether N3 receives the data packet successfully or not. Such an ambiguity exists even when there are no misbehaving nodes. The problem becomes much more severe in open MANETs with potential misbehaving nodes. The 2ACK scheme requires an explicit acknowledgment to be sent by N3 to notify N1 of its successful reception of a data packet: When node N3 receives the data packet successfully, it sends out a 2ACK packet over two hops to N1 (i.e., the opposite direction of the routing path as shown), with the ID of the corresponding data packet. The triplet (N1 -> N2 -> N3) is derived from the route of the original data traffic. Such a triplet is used by N1 to monitor the link N2 -> N3. For convenience of presentation, we term N1 in the triplet (N1 -> N2 ->N3) the 2ACK packet receiver or the observing node and N3 the 2ACK packet sender.

Such a 2ACK transmission takes place for every set of triplets along the route. Therefore, only the first router from the source will not serve as a 2ACK packet sender. The last router just before the destination and the destination will not serve as 2ACK receivers.4
To detect misbehavior, the 2ACK packet sender maintains a list of IDs of data packets that have been sent out but have not been acknowledged. For example, after N1 sends a data packet on a particular path, say, (N1 ->N2 ->N3) it adds the data ID to LIST i.e., on its list corresponding to N2 ->N3.

A counter of forwarded data packets, Cpkts, is incremented simultaneously. At N1, each ID will stay on the list for τ seconds, the timeout for 2ACK reception. If a 2ACK packet corresponding to this ID arrives before the timer expires, the ID will be removed from the list. Otherwise, the ID will be removed at the end of its timeout interval and a counter called Cmis will be incremented. When N3 receives a data packet, it determines whether it needs to send a 2ACK packet to N1. In order to reduce the additional routing overhead caused by the 2ACK scheme, only a fraction of the data packets will be acknowledged via 2ACK packets. Such a fraction is termed the acknowledgment ratio, Rack. By varying Rack, we can dynamically tune the overhead of 2ACK packet transmissions. Node N1 observes the behavior of link N2 -> N3 for a period of time termed Tobs. At the end of the observation period, N1 calculates the ratio of missing 2ACK packets as Cmis/Cpkts and compares it with a threshold Rmis. If the ratio is greater than Rmis, link N2 -> N3 is declared misbehaving and N1 sends out an RERR (or the misbehavior report) packet. Since only a fraction of the received data packets are acknowledged, Rmis should satisfy Rmis > 1 - Rack in order to eliminate false alarms caused by such a partial acknowledgment technique. Each node receiving or overhearing such an RERR marks the link N2 ->N3 as misbehaving and adds it to the blacklist of such misbehaving links that it maintains. When a node starts its own data traffic later, it will avoid using such misbehaving links as a part of its route. The 2ACK scheme can be summarized in the pseudo code provided in the appendix for the 2ACK packet sender side (N3) and the observing node side (N1).

III. Design And Analysis

Design is a meaningful engineering representation of something that is to be built. Software design is a process through which the requirements are translated into a representation of the software. Design is the place where quality is fostered in software engineering. Design is the perfect way to accurately translate a customer's requirement in to a finished software product. Design creates a representation or model, provides detail about software data structure, architecture, interfaces and components that are necessary to implement a system. This chapter discusses about the design part of the project. Here in this document the various UML diagrams that are used for the implementation of the project are discussed.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has 14 types of diagrams divided into two categories. Seven diagram types represent structural information, and the other seven represent general types of behavior, including four that represent different aspects of interactions. Some of these diagrams can be categorized hierarchically as below:

3.1 Use-case Diagram

A **Use Case Diagram** in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Since our system mainly detects routing misbehavior in mobile ad-hoc networks. Here the role and responsibility of mobile nodes is very important. By using routing protocols the mobile nodes continuously communicating with other nodes and involves in detection of misbehavior of nodes and mitigation. According to this information the following actors are involved in this system.
1. Source node
2. Destination node
3. Intermediate node

Source node Here source node is a node which may be a computer or mobile device or any router which tries to communicate with destination node and involves in detecting routing misbehavior, routing traffic before forwarding packets (data traffic) to the next node in a path to destination.

Destination node It is also either a computer or mobile device or any router which is sending feedback, acknowledgement with communication from source node before receiving data packets and after receiving data packets.

Intermediate node Intermediate node is an observer node or router in between source node and destination node. In general it receives the data packet from source node and forwards it to other nodes towards the destination. Sometimes it may not forward the data packet due to selfishness.

In our system we may have the following use cases

1. Detecting misbehavior
2. Finding performance in routing layer
3. Communicating with other nodes
4. Establishing network connection with other nodes
5. Sharing resources
6. Sending acknowledgement
7. Computing routing table
8. Authenticating packets
9. Sending packets
10. Receiving packets
11. Partial data forwarding
12. Data forwarding in TCP traffic
13. Data forwarding in UDP traffic
14. One way acknowledgement
15. Two way acknowledgement
16. UDP connection
17. TCP connection
18. Route discovery
19. Route maintenance
20. Retransmit
21. Buffering

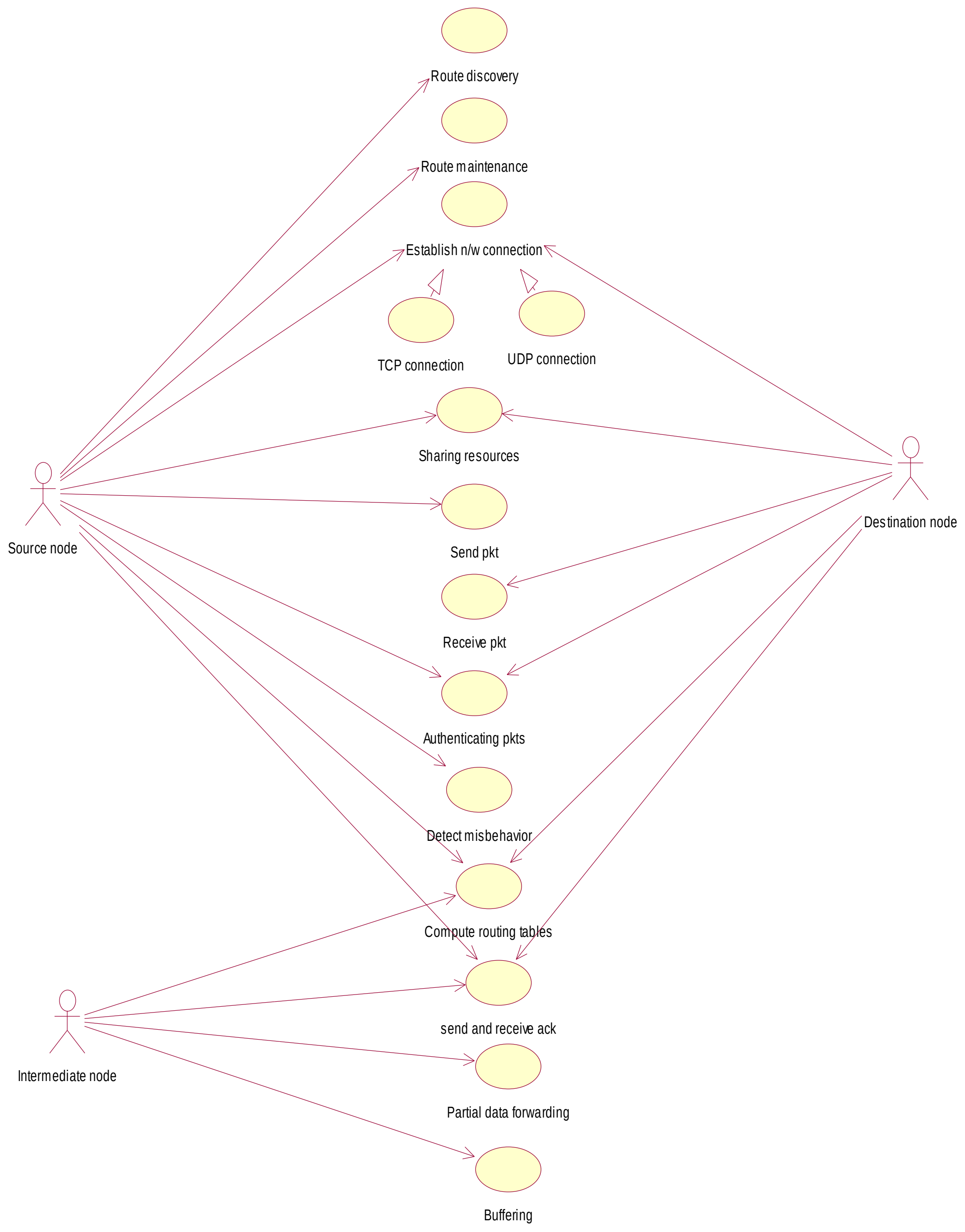


Fig 3.1 Use Case Diagram

S.No	Class Name	Attributes	Operations

1	Network	networktype
		no.ofnodes
			establishCon()
			tcpCon()
			udpCon()
			changeCon()
			disconnect()

2	Node	nodeno
		nodeaddress
			findNextNode()
			communicate()
			sendAck()
			dropPkt()

3	SourceNode	address
		timeout
		targetaddress
		delay
			sendPkt()
			setValidity()
			setTimer()
			clearTimer()
			receiveAck()

4	IntermediateNode	address
		counter
		observperiod
		timeout
			forwardPkt()
			buffering()
			findProbability()
			report()

5	DestinationNode	address
		srcaddress
			receiveData()
			sendAck()

6	Route	distance
		avgnoofhops
			findRoutetoDestination()
			findRoutetoNextHop()
			updateRoutingTable()
			changeRoute()

7	Acknowledgement	ackratio
		senderaddresss
		receiveraddress
		ackno
		acktype
			authenticateAck()
			fragmentPkt()

8	Packet	Pktid
		Pktsize
			partitioning()
			assembling()

A **class diagram** in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. The class diagram is the main building block in object oriented modeling. They are being used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed. In the class diagram these classes are represented with boxes which contain three parts:

- The upper part holds the name of the class
- The middle part contains the attributes of the class, and
- The bottom part gives the methods or operations the class can take or undertake

The main classes that are identified in our system design are as follows:

- Network
- Node
- SourceNode
- DestinationNode
- IntermediateNode
- Acknowledgement

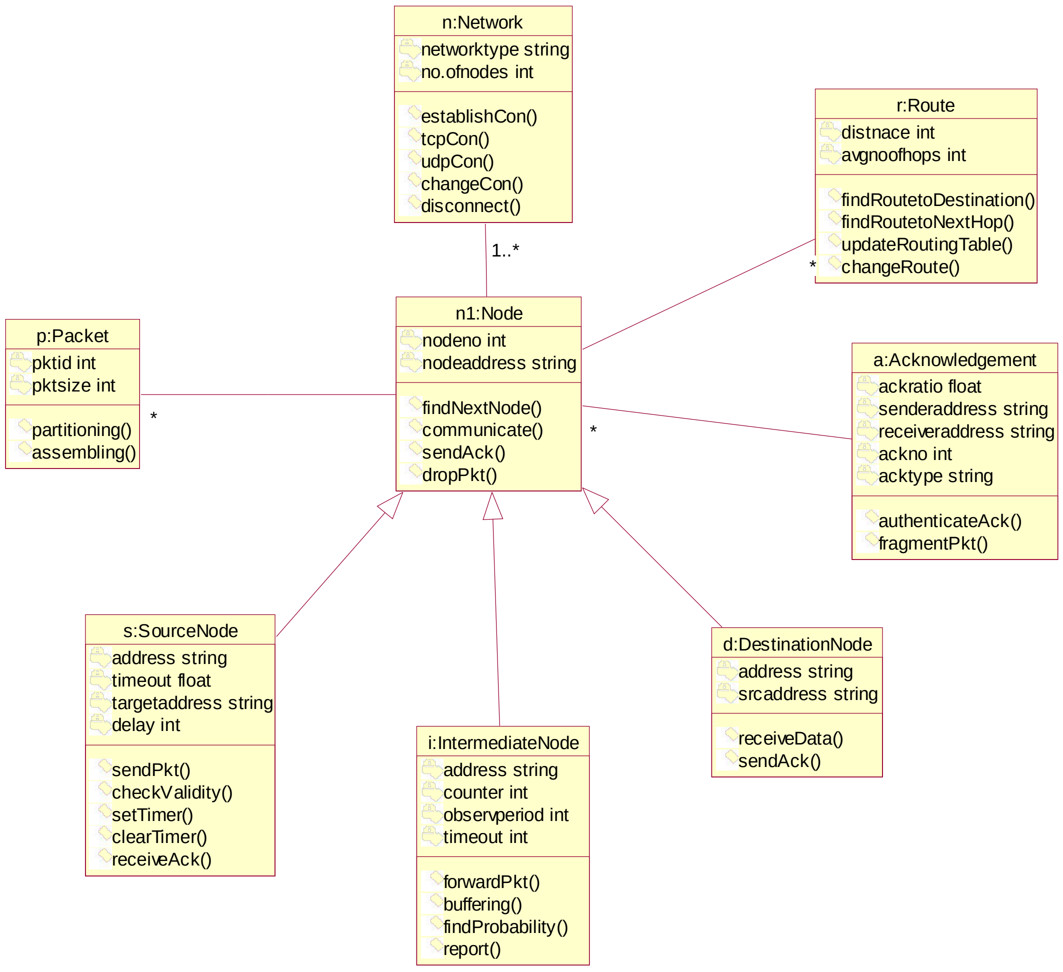


Fig 3.2 Class Diagram

Description: This class diagram contains different classes such as network, node, packet, route, acknowledgement etc. The node may be either source or intermediate or destination node. These nodes are connected in network. And the packets can be routed from one node to another node using different routes; after packet has been received acknowledgement is sent to sender node.

3.3 Sequence Diagram

A **sequence diagram** in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Sequence diagram for network connection establishment

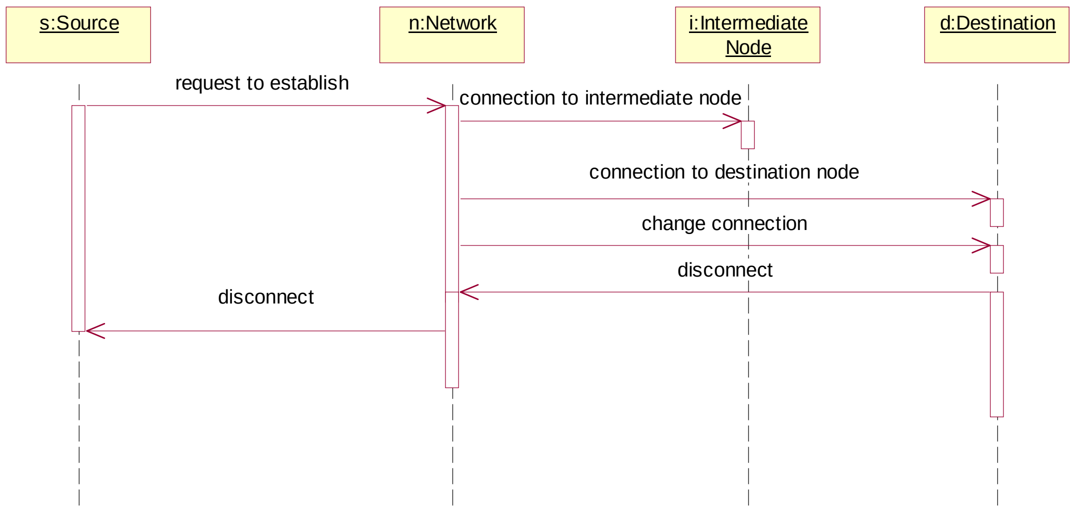


Fig 3.3 Sequence diagram for connection establishment

Description This sequence diagram involves the objects of source, network, intermediate node and destination. Source request for network establishment, then the network connect the source to intermediate node and then to destination node. If no connection is needed then it disconnect the connection.

Sequence diagram for data transmission

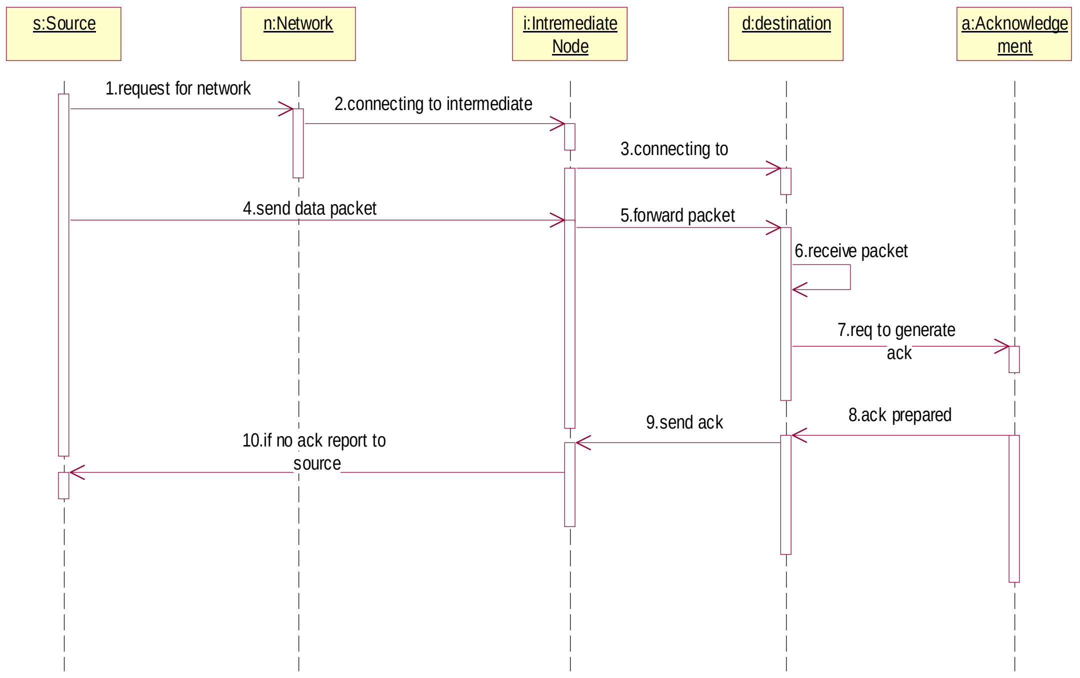


Fig 3.4 Sequence diagram for data transmission

Description In this case source is connected to destination through several intermediate nodes. Then source sends packet to intermediate nodes and they forward the packet to destination node. The destination node receive packet and generates acknowledgement and this acknowledgement is sent back to previous nodes.

Sequence diagram for routing packets

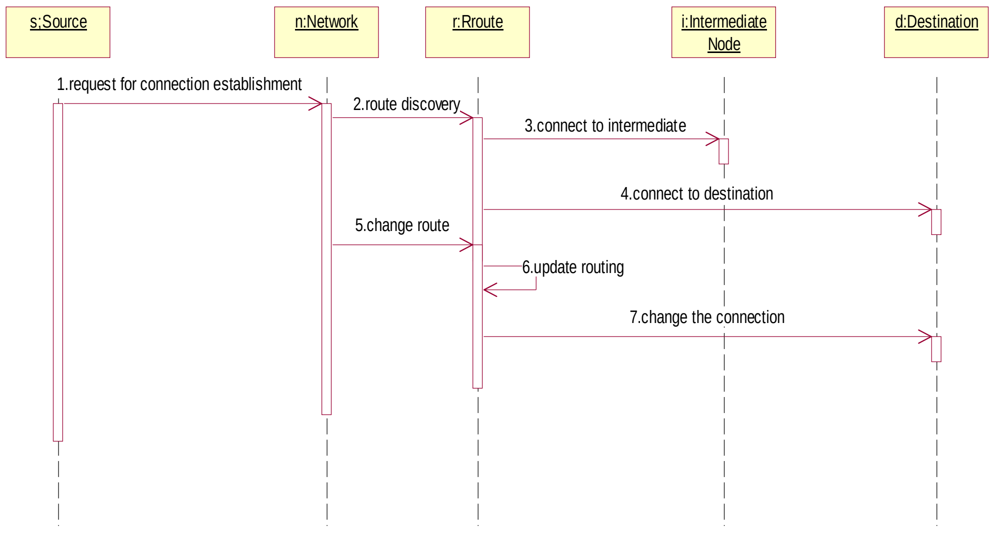


Fig 3.5 Sequence Diagram for routing packets

Description This diagram shows that at first network is established, then route is discovered to connect to destination to route packets. In the network when the routes are changed then the routing tables are updated, as a result new routes are established.

3.4 Collaboration Diagram

A **Collaboration diagram** also known as Communication diagram models the interactions between objects or parts in terms of sequenced messages. Communication diagrams represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behavior of a system. Communication diagrams show a lot of the same information as sequence diagrams, but because of how the information is presented, some of it is easier to find in one diagram than the other. Communication diagrams show which elements each one interacts with better, but sequence diagrams show the order in which the interactions take place more clearly.

Collaboration Diagram for network connection establishment

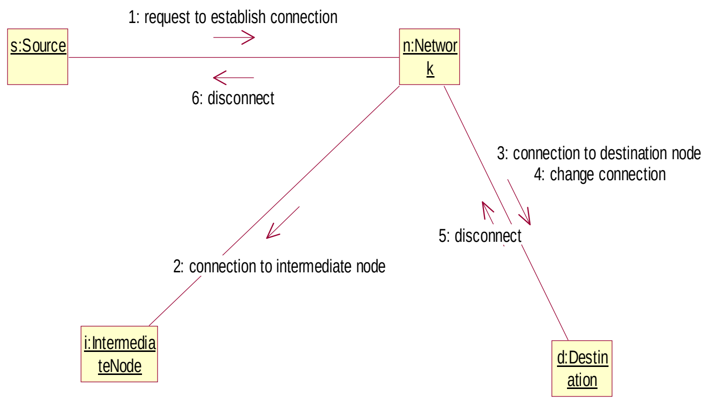
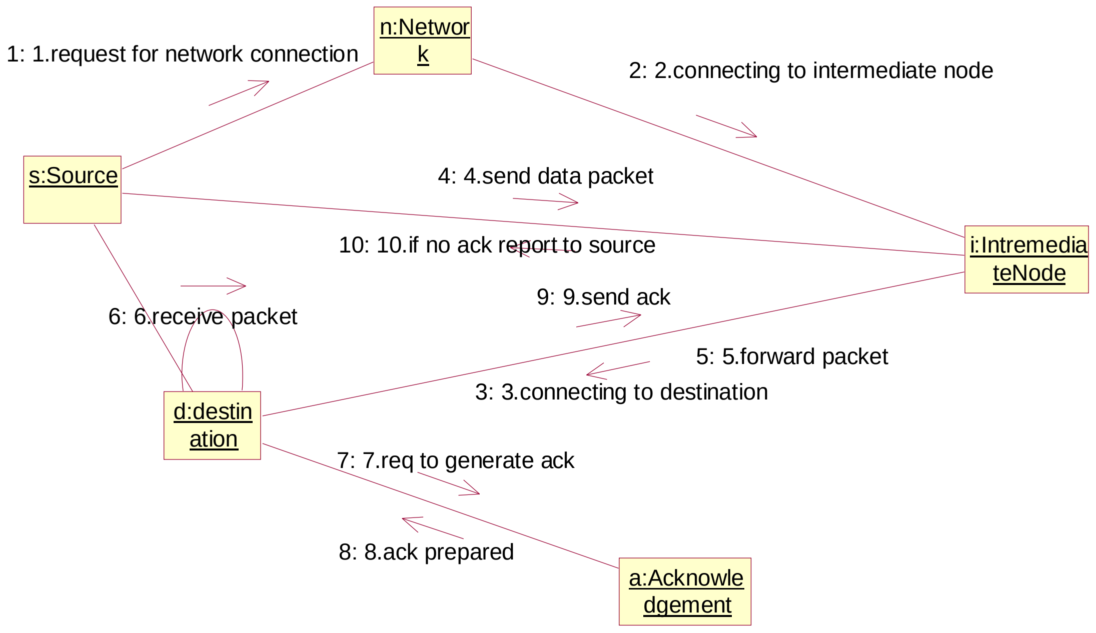

1: request to establish connection
s:Source
n:Network
6: disconnect
3: connection to destination node
4: change connection
5: disconnect
2: connection to intermediate node
i:IntermediateNode
d:Destination

Fig 3.6 Collaboration Diagram for network connection establishment

Collaboration Diagram for data transmission



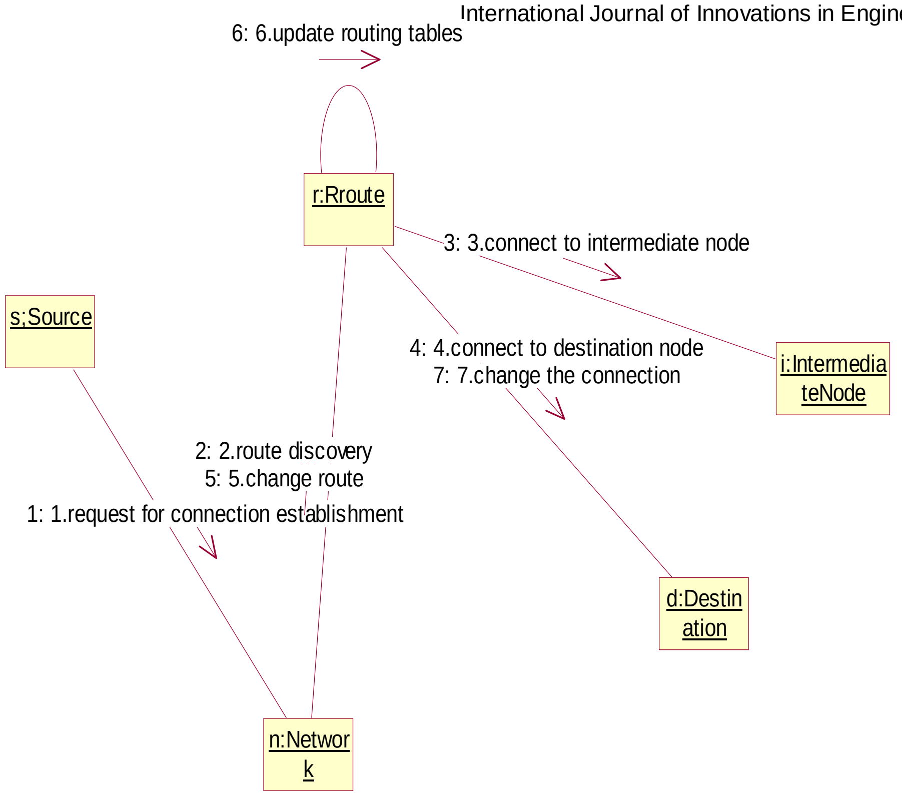

6: 6.update routing tables
r:Rroute
3: 3.connect to intermediate node
s;Source
4: 4.connect to destination node
7: 7.change the connection
i:IntermediateNode
2: 2.route discovery
5: 5.change route
1: 1.request for connection establishment
d:Destination
n:Network

Fig 3.7 Collaboration Diagram for data transmission

Collaboration Diagram for routing packets

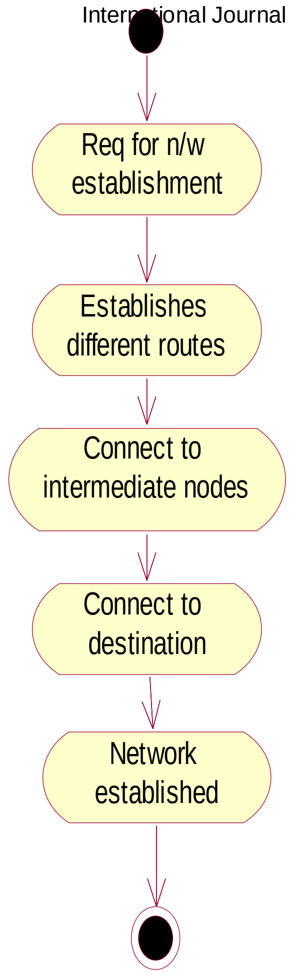

Req for n/w establishment
Establishes different routes
Connect to intermediate nodes
Connect to destination
Network established

Fig 3.8 Collaboration Diagram for routing packets

3.5 Activity Diagram

An activity is an ongoing non atomic execution with in a state machine. Activity ultimately results in some action, which is made up of executable atomic operations that results in a change in state of the system or the return of a value.

Activity diagram for network connection establishment

Fig 5.9 Activity diagram for connection establishment

Description This activity diagram shows that first the source request for network establishment. Then it establishes different routes to connect to intermediate nodes and destination node. Finally the connection is established between source and destination through several intermediate nodes.

Activity diagram for data transmission

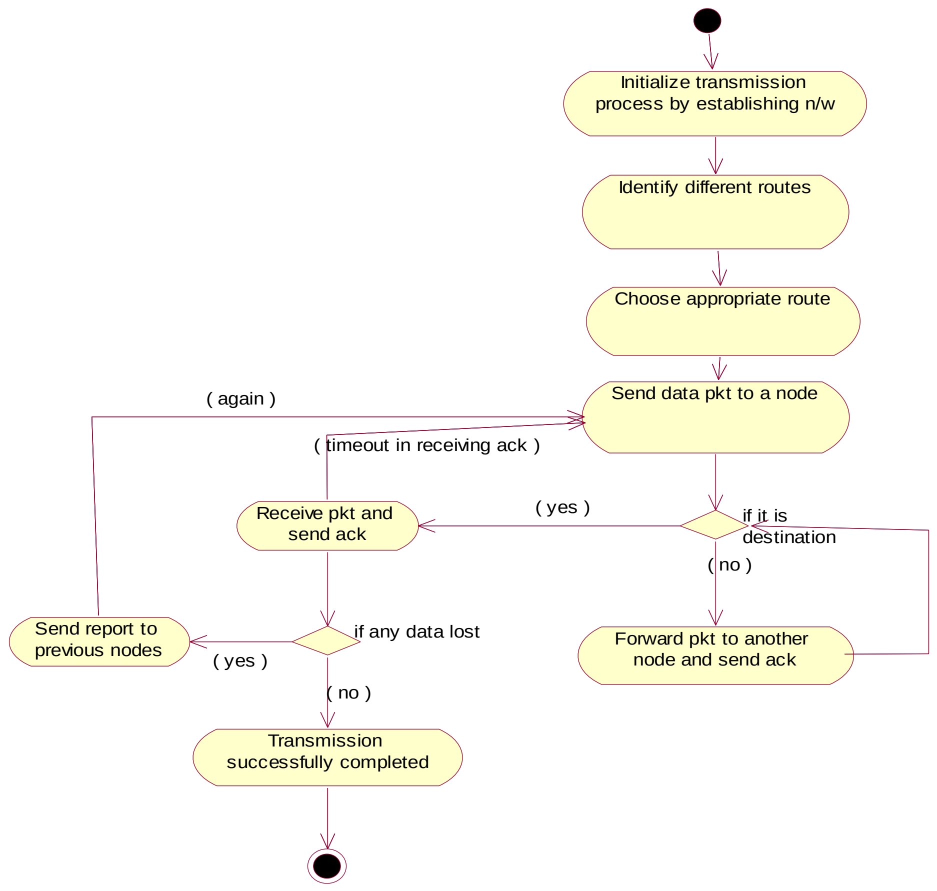

Initialize transmission process by establishing n/w
Identify different routes
Choose appropriate route
Send data pkt to a node
(again)
(timeout in receiving ack)
(yes)
if it is destination
Receive pkt and send ack
(no)
Forward pkt to another node and send ack
if any data lost
Send report to previous nodes
(yes)
(no)
Transmission successfully completed

Fig 3.10 Activity diagram for connection establishment

Description Here source initializes the transmission process by establishing the network. It identifies different routes and select appropriate route to send packet to destination. After sending the packet that packet will be received by another node , if the receiving node is destination then it will send the acknowledgement to sender node. If the receiving node is intermediate node then the packet will be forwarded to next node in the network towards the destination. If there are any data losses that will be reported to previous nodes, otherwise i.e., if no data losses are there then data transmission is completed successfully.

Activity diagram for routing packets

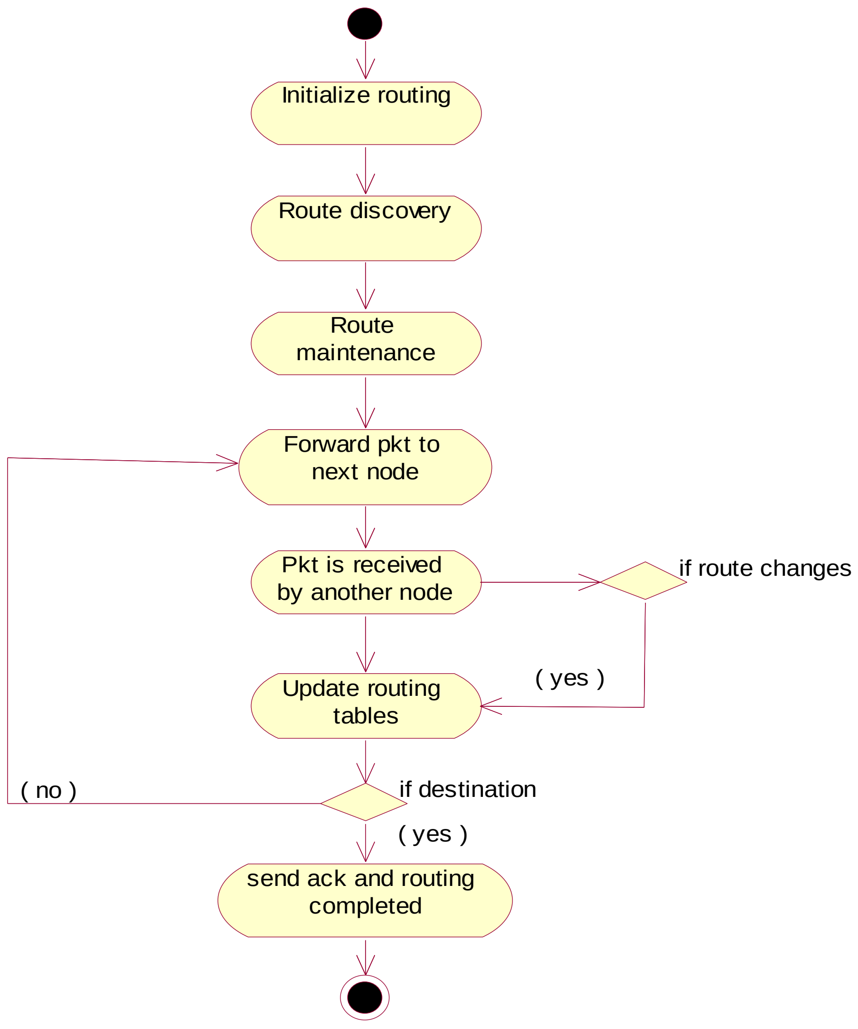


Fig 3.11 Activity diagram for connection establishment

Description Here the source node initializes the routing process by route discovery and route maintenance steps. By selecting better route packets will be transmitted. Due to dynamic changes in network the routing tables are updated regularly and based on those tables subsequent transmissions will be done.

3.6 Component Diagram

A** Component diagram **in the Unified Modeling Language, depicts how components are wired together to form larger components and or software systems. Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. Components diagrams can be used to illustrate the structure of arbitrarily complex systems.

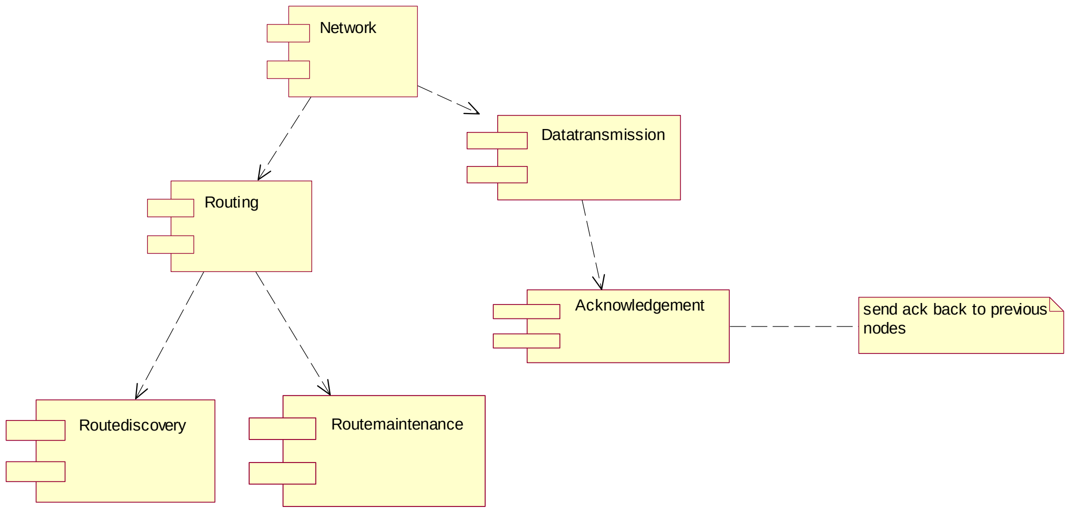


Fig 3.12 Component Diagram

Description The component diagram of our system involves the components such as network, routing, datatransmissions, and acknowledgement. The routing component associated with route discovery and route maintenance components. Network component is connected to routing and datatransmission components. Datatransmission involves packet and acknowledgement transmissions.

3.7 Deployment Diagram

A **Deployment diagram** in the Unified Modeling Language serves to model the physical deployment of artifacts on deployment targets. Deployment diagrams show "the allocation of Artifacts to Nodes according to the Deployments defined between them." Deployment of an artifact to a node is indicated by placing the artifact inside the node. Instances of nodes (and devices and execution environments) are used in deployment diagrams to indicate multiplicity of these nodes. For example, multiple instances of an application server execution environment may be deployed inside a single device node to represent application server clustering.

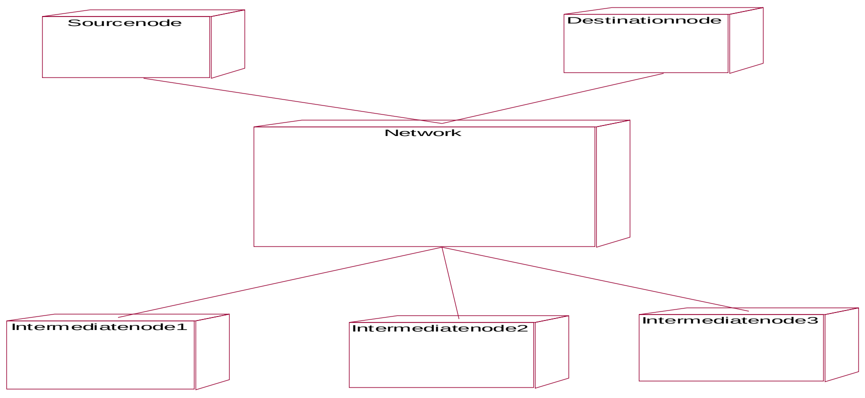


Fig 3.13 Deployment Diagram

Description This deployment diagram shows the different nodes that involved in our system. This contains the source node, intermediate node and destination nodes connected to network, which leads to the connections between source and destination source and intermediate nodes or intermediate nodes and destination.

IV. System 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. In general, software engineers distinguish software faults from software failures. Our project" Visual cryptography For Cheating Prevention" is tested with the following testing methodologies.

4.1 Developing Methodologies

The test process begins by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework for developing the test methodologies.

4.2 Acquire and study the test strategy

A team very familiar with the business risks associated with the software normally develops test strategy, the test team develops tactics. Thus the test team needs to acquire and study the test strategy. The test tactics are analyzed and studied for finding our various test factors, risks and effects. The risk involved in our project is implementing the acknowledgement scheme. So, the proper knowledge about the testing strategies should be gained in order to avoid such high level risks.

4.3 Determine the type of development project

The type of the development refers to the platform or methodology for developing the project. As it is been a simulation project we go for the prototyping. The prototypes are simply predefined structure or model, which can be used for further modeling. By using the prototypes we can modify the existing module of the application for some other specific operations. Here the test tactics is to verify that all the tools are used properly and to test functionality.

4.4 Determine the type of software system

The type of software system relates to the type of processing which will be encountered by the system. In this project, the software system we prefer to use is Java . We have chosen Java for its portability and its support to graphics & flash specifically for simulation.

4.5 Determine the scope of the software system

The scope of the project refers to the overall activities or operation to be included into the system being tested. The scope of the new system varies from that of the existing one. In the existing system, a large overhead occurs in sending acknowledgements. In this project, the acknowledgement transmission is optimal because only fraction of data packet is transmitted as acknowledgement rather than full packet.

4.6 Identify the tactical risks

The tactical risk is the subsets at a lower level of the strategic risks. The risks related to the application and its methodologies are identified. The risk involved in our project is implementing the identifying the misbehaving link.

4.7 Determine when the testing should occur

In the above processes we have identified the type of processing, scope and risks associated with our project. The testing can occur throughout all the phases of the project. During the analysis phase, the testing strategy and requirements are determined. In design phase, the complexities in design with respect to the requirements are determined and structural and functional test conditions are also tested. During implementation, the design consistency is determined. In test phase, the overall testing of the application is being done and previously the adequacy of the testing plan is also determined. In maintenance phase, the testing for modifying and reusing the system is done.

4.8 Build the system test plan

The test plan of the project should provide all the information on the application that is being tested. The test plan is simply a model that has to be followed during the progression of the testing. The test plan consists of the sequential set of procedures to test the application. Initially, the identification process of misbehaving links are tested. Then the test is carried out for 2ack scheme.

4.9 Build the unit test plan

In this case we are dividing the system into three different components or units each having specific functions. The three different components of the system are connection establishment, nodes events handling and change the connection. These units have their own test plan. The main purpose of the unit test plan is to eliminate the errors and bugs during the initial stage of the implementation. As the errors get debugged in the initial stage, the less complex the overall testing after integrating all the units of the system. The unit testing plan can be either simple or complex based on the functionality of that unit.

4.10 Testing Technique – Tool selection Process

In this process the appropriate testing process is selected from various testing methodologies such as prototyping model, waterfall model etc and the selection is done by the means of analyzing the nature of the project. We go for Waterfall model.

4.11 Select test factor

This phase selects the appropriate test factor. The particular module of the project which is essential for the testing methodologies is sorted out first. This will help the testing process to be completed within time. The test factors for our project include encoding, verification and decoding process.

4.12 Determine SDLC phase

This phase involves the structural testing of the project which will be used for easy implementations of the functions. Though structural testing is so much associated with the coding phase, the structural testing should be carried out at all the phases of the lifecycle. These evaluates that all the structures are tested and sound.

4.13 Identify the criteria to test

In this phase the testing unit is trained with the necessary constraints and limit with which the project is to be tested. In our project the testing unit is trained to test whether all the misbehaving nodes are correctly identified or not.

4.14 Select type of test

Individual responsible for testing may prefer to select their own technique and tool based on the test situation. For selecting the appropriate testing process the project should be analyzed with the following three testing concepts:

- Structural versus functional testing
- Dynamic versus static testing
- Manual versus automatic testing

After analyzing through the above testing concepts we divided to test our project in Waterfall model testing methodology.

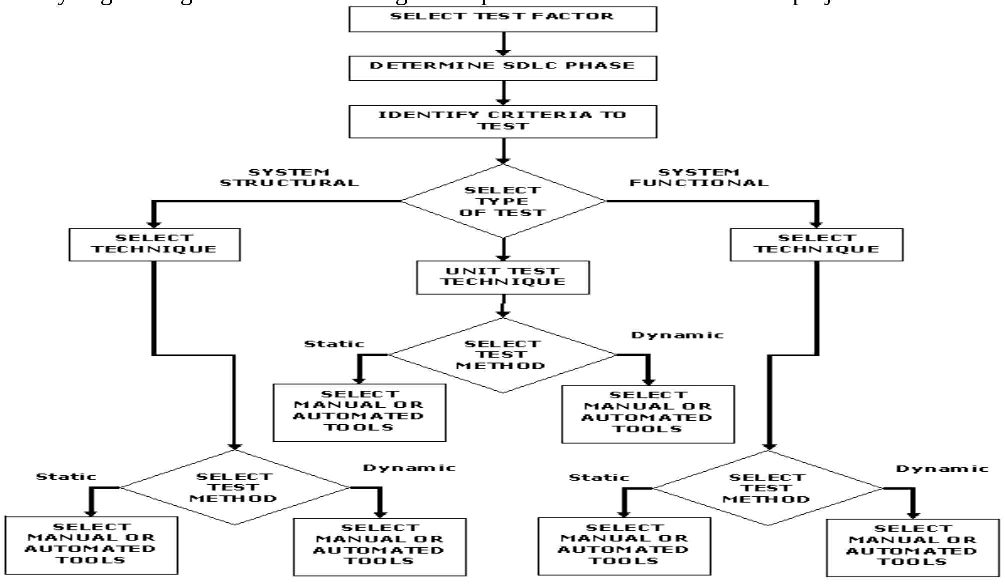


Fig 7.1 Testing technique and tool selection process

Structural Testing

Structural analysis based test sets are tend to uncover errors that occur during coding of the program. The properties of the test set are to reflect the internal structure of the program. Structural testing is designed to verify that the developed system and programs work as specified in the requirement. The objective is to ensure that the product is designed structurally sound and will function correctly.

Functional Testing

Functional testing ensures that the requirements are properly satisfied by the application system. The functions are those tasks that the system is designed to accomplish. This is not concerned with how processing occurs but rather with the results of the processing. The functional analysis based test sets tend to uncover errors that occurred in implementing requirements or design specifications.

Select technique

After selecting the appropriate testing methodology we have to select the necessary testing technique such as stress testing, execution testing, recovery testing, operation testing, compliance testing and security testing. We are performing operation testing by testing whether all the components perform its intended operations.

Select test method

We have to select the testing method which is to be carried out throughout the lifecycle. The two different methods are static and dynamic. Dynamic testing needs the program to be executed completely before testing. This is a traditional concept where the faults detected at the end will be very hard to rectify. In static process the program is tested for each and every line and the testing process is allowed to pass through only after rectifying the occurred fault. These make this process more expensive, so a combination of both static and dynamic testing method.

Mode of testing

It is necessary to select the test mode in which the testing method to be carried out. The two different modes are manual and automated tool. The real time projects needs frequent interactions. So, it is impossible to carry out the testing process by means of automated tool. Our project uses manual testing.

Unit test technique

This phase examines the techniques, assessment and management of unit testing and analysis. Testing and analysis strategies are categorized according to whether they goal is functional or structural or combination of these. It will assist a software engineer to define, conduct and evaluate unit tests and to assess new unit test techniques.

System Testing

Once the entire system has been built then it has to be tested against the "System Specification" to check if it delivers the features required. It is still developer focused, although specialist developers known as systems testers are normally employed to do it. In essence System Testing is not about checking the individual parts of the design, but about checking the system as a whole. In effect it is one giant component. System testing can involve a number of specialist types of test to see if all the functional and non-functional requirements have been met.

Regression Testing

This involves assurance that all aspects of an application system remain functional after testing. The introduction of change is the cause of problems in previously tested segments. It is retesting unchanged segments of the application system. It normally involves rerunning tests that have been previously executed to ensure that the same results can be achieved currently as achieved when the segments were last tested.

V. CONCLUSIONS

Mobile Ad Hoc Networks (MANETs) have been an area for active research over the past few years due to their potentially widespread application in military and civilian communications. Such a network is highly dependent on the cooperation of all of its members to perform networking functions. This makes it highly vulnerable to selfish nodes. One such misbehavior is related to routing. When such misbehaving nodes participate in the Route Discovery phase but refuse to forward the data packets, routing performance may be degraded severely.

We have proposed and evaluated a technique, termed 2ACK, to detect and mitigate the effect of such routing misbehavior. The 2ACK technique is based on a simple 2-hop acknowledgment packet that is sent back by the receiver of the next-hop link. Compared with other approaches to combat the problem, such as the overhearing technique, the 2ACK scheme overcomes several problems including ambiguous collisions, receiver collisions, and limited transmission powers. The 2ACK scheme can be used as an add-on technique to routing protocols such as DSR in MANETs. Extensive simulations of the 2ACK scheme have been performed to evaluate its performance. Our simulation results show that the 2ACK scheme maintains up to 91 percent packet delivery ratio even when there are 40 percent misbehaving nodes in the MANETs t. The regular DSR scheme can only offer a packet delivery ratio of 40 percent. One advantage of the 2ACK scheme is its flexibility to control overhead with the use of the Rack parameter.

VI. FUTURE ENHANCEMENTS

In this work, we have focused only on link misbehavior. It is more difficult to decide the behavior of a single node. This is mainly due to the fact that communication takes place between two nodes and is not the sole effort of a single node. Therefore, care must be taken before punishing any node associated with the misbehaving links. When a link misbehaves, either of the two nodes associated with the link may be misbehaving. In order to decide the behavior of a node and punish it, we may need to check the behavior of links around that node. This is a potential direction for our future work**.** The 2ACK scheme has been implemented on top of DSR. It is also possible to implement the 2ACK scheme over other routing schemes. The main challenge is how to derive the triplet information so that the 2ACK sender and the observing node are informed of such information. Knowledge of topology of the 2-hop neighborhood may be used. In our future work, we will investigate how to add the 2ACK scheme to other types of routing schemes and open networks. Theoretical analysis of the performance gain of the 2ACK scheme is of interest as well.

REFERENCES

[1] Kejun Liu, Jing Deng, Member, IEEE, Pramod K. Varshney, Fellow, IEEE, and Kashyap Balakrishnan, Member, IEEE

[2] L. Buttyan and J.-P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," ACM/Kluwer Mobile Networks and Applications, vol. 8, no. 5, 2003.

[3] K. Balakrishnan is with the Security Services Group, Deloitte and Touche
LLP, 1750 Tysons Boulevard, Suite 800, McLean, VA 22102.
E-mail: kbalakrishnan@deloitte.com.

[4] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," Proc. MobiCom, Aug. 2000.

[5] V.-N. Padmanabhan and D.-R. Simon, "Secure Traceroute to Detect Faulty or Malicious Routing," SIGCOMM Computer Comm. Rev., vol. 33, no. 1, Jan. 2003.

[6] V. Srinivasan, P. Nuggehalli, C.F. Chiasserini, and R.R. Rao, "Cooperation in Wireless Ad Hoc Networks," Proc. INFOCOM, Mar.-Apr. 2003.

[7] D. Johnson, D. Maltz, Y.C. Hu, and J. Jetcheva, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," Internet draft, Feb. 2002.