Chapter 5

Routing Protocols for WANET: MATLAB Implementation

This chapter describes an overview of power control and optimization of the Reactive Routing protocols AODV in wireless ad hoc network using the traditional techniques. Study of AODV reactive routing protocol is done using Truetime toolbox. AODV Simulator is described to study and compare results of routing protocol.

Routing and power control issues present special challenges in ad hoc networks. Typically, every node in an ad hoc network serves as a router for other nodes, and paths from source to destination often require multiple hops. Compared to wired networks, wireless ad hoc networks have less bandwidth, longer paths, and less stable connectivity, all of which render routing protocols from wired networks less suitable for the wireless world. An intelligent routing strategy is required to efficiently use the limited resources while at the same time being adaptable to the changing network conditions such as: network size, traffic density and network partitioning [1, 2]. In parallel with this, the routing protocol may need to provide different levels of QoS to different types of applications and users.

5.1 Overview of AODV Routing Protocol

AODV stands for Ad-Hoc On-Demand Distance Vector [3] of a reactive type protocol, even though it still uses characteristics of a proactive protocol. AODV takes the interesting parts of DSR and DSDV [4], in the sense that it uses the concept of route discovery and route maintenance of DSR [5] and the concept of sequence numbers and sending of periodic hello messages from DSDV.

Routes in AODV are discovered and established and maintained only when and as long as needed. To ensure loop freedom sequence numbers, which are created and updated by each node itself, are used. These allow also the nodes to select the most recent route to a given destination node.

AODV takes advantage of route tables. In these it stores routing information as destination and next hop addresses as well as the sequence number of a destination. Next to that a node also keeps a list of the precursor nodes, which route through it, to make route maintenance easier after link breakage. To prevent storing information and maintenance of routes that are not used anymore each route table entry has a lifetime. If during this time the route has not been used, the entry is discarded [6].

5.1.1 Routing table

Each routing table entry contains the following information [7]:

- ✓ Destination
- ✓ Next hop
- ✓ Number of hops
- ✓ Destination sequence number
- ✓ Active neighbours for this route

- ✓ Expiration time for this route table entry

Expiration time, also called lifetime, is reset each time the route has been used. The new expiration time is the sum of the current time and a parameter called active route timeout. This parameter, also called route caching timeout, is the time after which the route is considered as invalid, and so the nodes not lying on the route determined by RREPs delete their reverse entries. If active route timeout is big enough Route Repair will maintain routes. RFC 3561 defines it to 3 seconds.

5.1.2 Control messages

An important feature of AODV is the maintenance of time-based states in each node: a routing entry not recently used is expired. In case of a route is broken the neighbours can be notified. Route discovery is based on query and reply cycles, and route information is stored in all intermediate nodes along the route in the form of route table entries. The following control messages are used:

1) Routing Request Message (RREQ) is broadcasted by a node requiring a route to another node.
2) Routing Reply Message (RREP) is unicasted back to the source of RREQ.
3) Route Error Message (RERR) is sent to notify other nodes of the loss of the link.
4) HELLO messages are used for detecting and monitoring links to neighbors.

AODV is a relative of the Bellman-Ford distant vector algorithm, but is adapted to work in a mobile environment. AODV determines a route to a destination only when a node wants to send a packet to that destination. Routes are maintained as long as they are needed by the source. Sequence numbers ensure the freshness of routes and guarantee the loop-free routing.

5.1.2.1 Routing request

When a route is not available for the destination, a route request packet (RREQ) is flooded throughout the network. The RREQ contains the following fields [7]:

Source Address	Request ID	Source Sequence No.	Destination Address	Destination Sequence No.	Hop Count

The request ID is incremented each time the source node sends a new RREQ, so the pair (source address, request ID) identifies a RREQ uniquely. On receiving RREQ message each node checks the source address and the request ID. If the node has already received a RREQ with the same pair of parameters the new RREQ packet will be discarded. Otherwise the RREQ will be either forwarded (broadcast) or replied (unicast) with a RREP message:

- If the node has no route entry for the destination, or it has one but this is no more an up-to-date route, the RREQ will be rebroadcasted with incremented hop count

- If the node has a route with a sequence number greater than or equal to that of RREQ, a RREP message will be generated and sent back to the source.

The number of RREQ messages that a node can send per second is limited.

There is an optimization of AODV using an expanding ring (ESR) technique when flooding RREQ messages [8]. Every RREQ carries a time to live (TTL) value that specifies the number of times this message should be re-broadcasted. This value is set to a predefined value at the first transmission and increased at retransmissions. Retransmissions occur if no replies are received. Historically such flooding used a TTL large enough - larger than the diameter of the network - to reach all nodes in the network, and so to guarantee successful route discovery in only one round of flooding [9]. However, this low delay time approach causes high overhead and unnecessary broadcast messages. Later, it was shown that the minimal cost flooding search problem can be solved via a sequence of flooding with an optimally chosen set of TTLs [9].

5.1.2.2 Routing reply

If a node is the destination, or has a valid route to the destination, it unicasts a route reply message (RREP) back to the source. This message has the following format.

Source Address	Destination Address	Destination Sequence No.	Hop Count	Life Time

The reason one can unicast RREP back is that every node forwarding a RREQ message caches a route back to the source node.

5.1.2.3 Route error

All nodes monitor their own neighborhood. When a node in an active route gets lost, a route error message (RERR) is generated to notify the other nodes on both sides of the link of the loss of this link.

5.1.2.4 HELLO messages

Each node can get to know its neighborhood by using local broadcasts, so-called HELLO messages. Nodes neighbors are all the nodes that it can directly communicate with. Although AODV is a reactive protocol it uses these periodic HELLO messages to inform the neighbors that the link is still alive. The HELLO messages will never be forwarded because they are broadcasted with TTL = 1. When a node receives a HELLO message it refreshes the corresponding lifetime of the neighbor information in the routing table [10, 11].

This local connectivity management should be distinguished from general topology management to optimize response time to local changes in the network.

5.1.3 Sequence numbers

The Ad Hoc On-Demand Distance Vector (AODV) routing protocol is intended for use by mobile nodes in an ad hoc network. It offers quick adaptation to dynamic link conditions, low processing and memory overhead, low network utilization, and establishment of both unicast and multicast routes between sources and destinations. It uses destination sequence numbers to ensure loop freedom at all times (even in the face of anomalous delivery of routing control messages), solving problems (such as ``counting to infinity'') associated with classical distance vector protocols.

5.1.3.1 Counting to infinity

The core of the problem is that when X tells Y that it has a path somewhere, Y has no way of knowing whether it itself is on the path - as Tanenbaum [12] notes. So if Y detects a link to Z is broken, but X still has a “valid” path to Z, Y assumes X in fact does have a path to Z. So X and Y will start updating each other in a loop, and the problem named ”counting to infinity” arises. AODV avoids this problem by using sequence numbers for every route, so Y can notice that X’s route to Z is an old one and is therefore to be discarded.

5.1.3.2 Time stamping

The sequence numbers are the most important feature of AODV for removing the old and invaluable information from the network. The destination sequence number for each destination host is stored in the routing table, and is updated in the routing table when the host receives the message with a greater sequence number. The host can change its own destination sequence number if it offers a new route to itself, or if some route expires or breaks.

Each host keeps its own sequence number, which is changed in two cases:

- Before the node sends RREQ message, its own sequence number is incremented
- When the node responds to a RREQ message by sending a RREP-message, its own sequence number becomes the maximum of the current sequence number and the node’s sequence number in the received RREQ message.

The reason is that if the sequence number of already registered is greater than that in the packet, the existing route is not up-to-date. The sequence numbers are not changed by sending HELLO messages [11].

5.1.4 Route discovery

Route discovery process starts when a source node does not have routing information for a node to be communicated with. Route discovery is initiated by broadcasting a RREQ message. The route is established when a RREP message is received. A source node may receive multiple RREP messages with

different routes. It then updates its routing entries if and only if the RREP has a greater sequence number, i.e. fresh information.

5.2 AODV Configuration Parameters

The default values for some important values associated with AODV protocol operations [13] shown in **Table 5.1**. A particular mobile node may wish to change certain of the parameters, in particular the NET_DIAMETER, MY_ROUTE_TIMEOUT, ALLOWED_HELLO_LOSS, RREQ_RETRIES, and possibly the HELLO_INTERVAL. In the latter case, the node should advertise the HELLO_INTERVAL in its Hello messages, by appending a Hello Interval Extension to the RREP message. Choice of these parameters may affect the performance of the protocol.

Active_Route_Timeout	3,000
Allowed_Hello_Loss	2
Bad_Link_Lifetime	2 * Rrep_Wait_Time
Bcast_Id_Save	30,000
Broadcast_Record_Time	Rrep_Wait_Time
Group_Hello_Interval	5,000
Hello_Interval	1,000
Mtree_Build	2 * Rev_Route_Life
My_Route_Timeout	2 * Active_Route_Timeout
Net_Diameter	35
Next_Hop_Wait	Node_Traversal_Time + 10
Node_Traversal_Time	40
Rev_Route_Life	Rrep_Wait_Time
Rrep_Wait_Time	3 * Node_Traversal_Time * Net_Diameter / 2
Rreq_Retries	2
Service_Addr_Timeout	300,000 sec
TTL_Start	1
TTL_Increment	2
TTL_Threshold	7

Table 5.1:Default value of parameters of AODV routing Protocol

AODV uses a static value for its route lifetime parameter called Active Route Timeout (ART) which indicates the time that the route can stay active in the routing table [14]. Route lifetime may be more accurately determined dynamically via measurement, instead of static value. Adaptive values for ART depending on the situation of the transmitter and intermediate nodes.

In most of the ad-hoc routing protocols, a static link lifetime (LL) is used for a newly discovered neighbors. Though this works well for networks with fixed infrastructures, it is inadequate for ad-hoc networks due to nodes mobility and frequent breaks of links. To overcome this problem, routing protocols with estimated LL using nodes affinity were introduced. In paper [15] methods are presented to estimate LL based on nodes affinity and then continually update those values depending on changes of the affinity.

Due to nodes' mobility, the efficiency of a dynamic ad hoc routing protocol depends highly on updating speed of network topology changes. To achieve continuous updated routing tables, the nodes

periodically broadcast short Hello messages to their neighbors. Although benefits of these messages have been proven, many studies show some drawbacks for these messages. In paper [16], adaptively optimization of the frequent needs of those messages using a fuzzy logic system is described. Extensive performance analysis via simulation proves the effectiveness of the proposed method to improve the accuracy of neighborhood information and, hence, overall network performance.

5.3 MATLAB

MATLAB is a high performance language for technical computing. It integrates computation, visualization and programming in and easy to use environment where problems and solutions are expressed in familiar mathematical notation. MATLAB is an interactive system while basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix a vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C. the name MATLAB is an abbreviation of matrix laboratory [17].

MATLAB features a family of add-on application specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulations and many others. The MATLAB system consists of five main pans, when MATLAB is started the MATLAB desktop appears, containing tools for managing files , variables and applications associated with MATLAB [18].

5.3.1 SIMULINK

Simulink is a software package for modeling, simulating and analyzing dynamic systems. It supports linear and non linear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multirate i.e. have different parts that are sampled or updated at different rates [19].

For modeling, Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click and drag mouse operations. It includes a comprehensive block library of sinks, sources, linear and non linear components and connectors. You can also customize and create your own blocks. For information on creating your own blocks, like separate writing S-Functions.

Models are hierarchical, so you can build models using both top down and bottom up approaches. You can view the system at a high level and then double click blocks to go down thorough the levels to see increasing level of model detail. This approach provides insight into how a model s organized and how its parts interact.

The model can be simulated using a choice of integration methods, either from the Simulink menus or by entering commands in the MATLAB Command window. The menus are particularly convenient of interactive work, while the command line approach is very useful for running a batch of simulations (for example using scopes and other display blocks, you can see the simulation results while the simulation is running, the simulation results can be put in the MATLAB workspace for post processing and visualization [17].

5.3.2 Truetime Toolbox

TrueTime (Cervin *et al.*, 2003;Andersson *et al.*, 2005)[20] is a Matlab/Simulink-based simulator for networked and embedded control systems that has been developed at Lund University since 1999.The use of the Matlab/Simulink-based [The Math works, 2001] simulator TRUETIME, which facilitates co-simulation of controller task execution in real-time kernels, network transmissions, and continuous plant dynamics is described. The simulator is presented in [Henriksson et al., 2003; Cervin et al. 2003; Henriksson et al., 2002; Andersson et al., 2005][21], but be aware that several differences from these papers exist.

The manual [20] describes the fundamental steps in the creation of a TRUETIME simulation. This include how to write the code that is executed during simulation, how to configure the kernel and network blocks, and what compilation that must be performed to get an executable simulation. The code functions for the tasks and the initialization commands may be written either as C++ functions or as Matlab M-files and both cases are described.

[22] has discussed networked control in industrial applications and the possibility to simulate the control systems by TrueTime, the Networked Control Systems, their characteristics and possibilities, the wired and also the wireless systems.[23] uses TrueTime toolbox (for Matlab) as a simple and easy way how to realize several network types. Also demonstrates how to setup simple TrueTime network control system with an explanation of its basic settings and parameters. In the last briefly compare simulation results of motor control system which uses different network types. [24] Describes, a higher level simulation platform for WSN is proposed based on the *TrueTime* toolbox. Relevant features include graphical representation of communication components, wireless communication and battery-driven operation. Special attention was paid to the 3D graphical interface, simulator interactivity and its extendibility. A TrueTime simulation model of the tunnel scenario is developed [25]. The TrueTime simulator allows concurrent simulation of the physical robots and their environment, the software in the nodes, the radio communication, the network routing, and the ultra-sound navigation system.

5.3.3 Software Requirements

TRUETIME currently supports Matlab 7.x (R14 and later) with Simulink 6.x and Matlab 6.5.1 (R13.1) with Simulink 5.1. Support for Matlab 6.1 (R12.1) with Simulink 4.1 was dropped in TRUETIME 1.3.

A C++ compiler is required to run TRUETIME in the C++ version. For the Matlab version, pre-compiled files are provided in the archive that is downloaded from the TRUETIME web site. The following compilers are currently supported (it may, of course, also work using other compilers):

• Visual Studio C++ 7.0 (for all supported Matlab versions) for Windows
• gcc, g++ - GNU project C and C++ Compiler for LINUX and UNIX

Before starting Matlab, you must set the environment variable TTKERNEL to point to the directory with the TRUETIME kernel files, $DIR/kernel. This is typically done in the following manner:

• Unix/Linux: export TTKERNEL=$DIR/kernel
• Windows: use Control Panel / System / Advanced / Environment Variables

Then add the following lines to your Matlab startup script. This will set up all necessary paths to the TRUETIME kernel files.

```
addpath([getenv('TTKERNEL')])
init_truetime;
```

Starting Matlab and issuing the command

```
>> truetime
```

From the Matlab prompt will now open the TRUETIME block library [ ] as in **Figure 2.2**.

5.3.4 Compilation

Since the TRUETIME archive contains pre-compiled files, no compilation is required to run TRUETIME with the M-file API. However, TRUETIME also supports simulations written in C++ code, which then must be compiled. In this case, you first need to configure your C++ compiler in Matlab. This can be done by issuing the command

```
>> mex -setup
```

In the setup, make sure that you change from the Matlab default compiler to a proper C++ compiler. For more detailed instructions on how to compile individual simulations, see Section 6 in this manual.

[23] Gives a brief introduction to the TrueTime simulator and then gives several examples on how TrueTime can be used to simulate networked control systems. Among the examples are time-triggered and event-based networked control and AODV routing in wireless ad-hoc networks.

The TrueTime Kernel block simulates a computer node with a generic real-time kernel, A/D and D/A converters, and network interfaces. The block is configured via an initialization script. The script may be parameterized, enabling the same script to be used for several nodes.

In this we tried to study working of AODV routing protocol using the Truetime toolbox available for MATLAB.

In this simulation node that requires a route to a destination node initiates route discovery by broadcasting an RREQ message to its neighbors. A node receiving an RREQ starts by updating its routing information backwards towards the source. If the same RREQ has not been received before, the node then checks its routing table for a route to the destination. If a route exists with a sequence number greater than or equal to that contained in the RREQ, an RREP message is sent back towards the source. Otherwise, the node rebroadcasts the RREQ. When an RREP has propagated back to the original source node, the established route may be used to send data. Periodic hello messages are used to maintain local connectivity information between neighboring nodes. A node that detects a link break will check its routing table to find all routes which use the broken link as the next hop. In order to propagate the information about the broken link, an RERR message is then sent to each node that constitutes a previous hop on any of these routes.

Each node also contains a periodic task, responsible for broadcasting hello messages and determines local connectivity based on hello messages received from neighboring nodes. Finally, each node has a task to handle timer expiry of route entries [11].

5.4 Simulation of Classical AODV using Truetime

The simulation model provides in Truetime for AODV routing protocol is shown in **Figure 5.1**. The simulation model creates the topology of the network consist of 7 nodes communicate with each other using wireless connections shown in **Figure 5.2.**

The Simulink model of AODV is as shown in **Figure 5.1** contains blocks for 7 nodes with their respective source and receiver blocks. The network block consists of information of Truetime wireless network block [26]. The information regarding number of nodes, transmission power, threshold

acknowledgement timeout, frame size and Pathloss are supplied to the system through this block. The node 1 is the source or transmitter and node is destination or the receiver of the system. The communication is performed between node number 1 and 7. The intermediate nodes are node number 2, 3, 4, 5 and 6. The route for communication is selected between node 1 and node 7 depending upon transmission power and threshold. The maximum signal reach is calculated and route for communication is selected.

5.4.1 Process to run simulation model

To run the simulation following commands execution require at the command prompt of Matlab after setting the current path in Matlab for kernel.

>>addpath(getenv('TTKERNEL'))

>> init_truetime

>> truetime

It will open the Truetime library, then open the AODV.mdl and initsim.m file.

First execute initsim.m file then run the AODV.mdl file to visualize the topology animation and the results on the command prompt.

This simulation scenario was carried out at power -8 db and threshold -48 db.

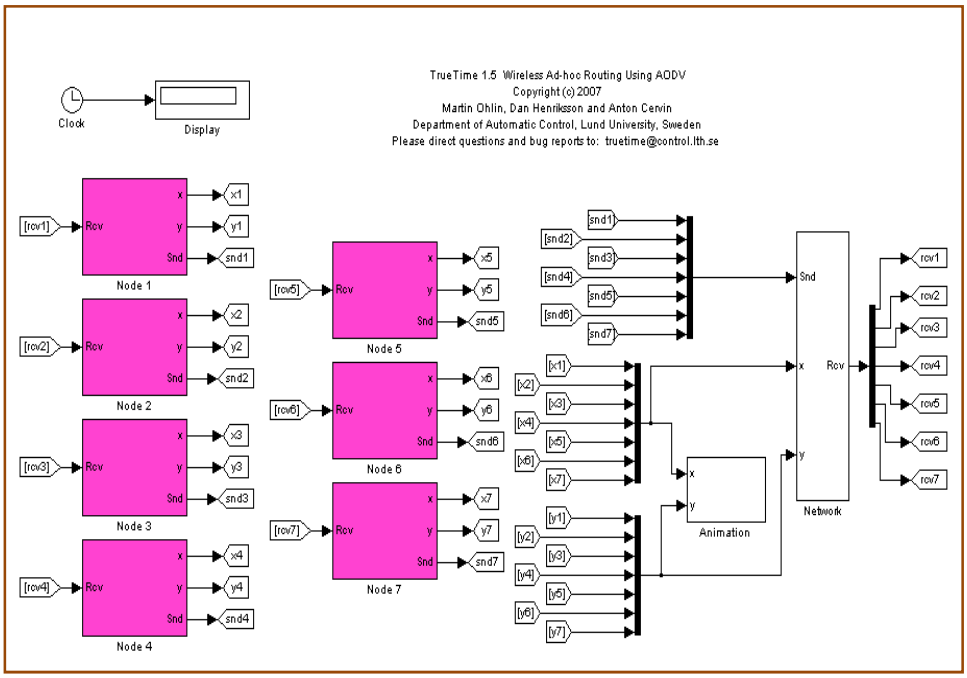


Figure 5.1 Simulink model in Truetime

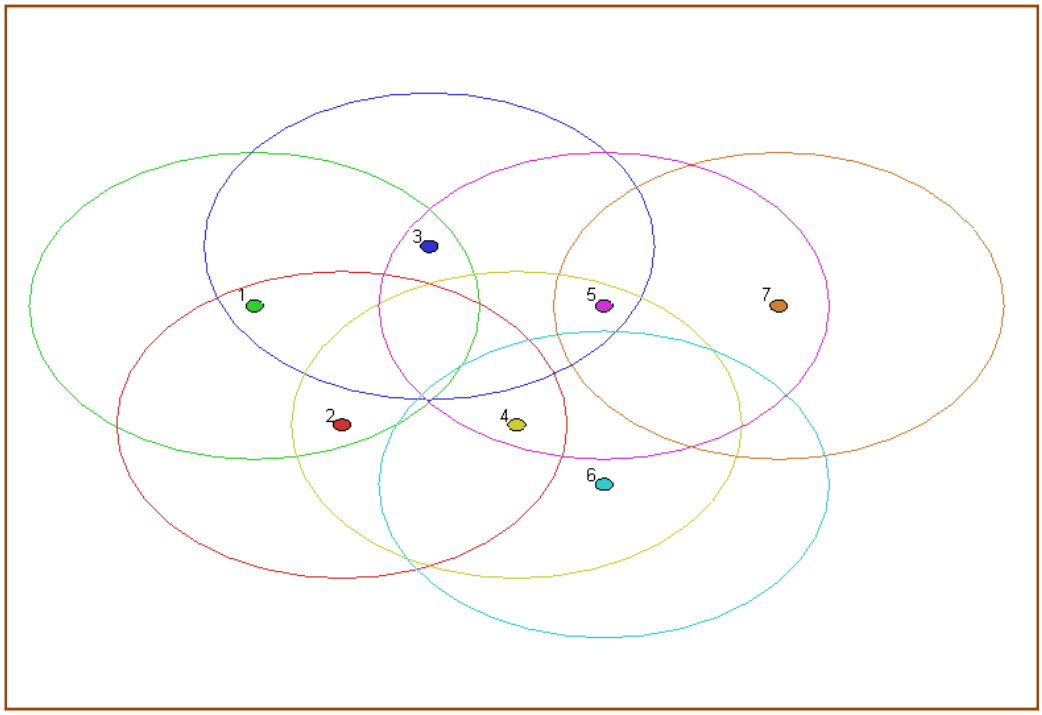


Figure 5.2: Topology created by Simulink model in Truetime

In the simulation scenario, the left-most node (node 1) sends data periodically to node 7 with period 0.5. The initial route that is established is 1 → 3 → 5 → 7. At time t=3, node 5 starts to move which eventually leads to the route breaking. At time t=10, node 6 repairs the route by moving in between node 4 and 7.

Various routes are available for communication, those routes are 1-3-5-7, 1-2-4-6-7, 1-5-7 etc. With increase in power and threshold the range or signal reach increase and number of hop count decreases. At certain time the node 5 moves out of range of communication performed through route 1-3-5-7. After node 5 moves away, the node 6 moves towards node 5 and comes in route of 1-3-5-7. The node topology is as shown in **Figure 5.2**. The circle with different colors shows various nodes used for communication.

5.4.2 Simulation Results for Classical AODV

The results on the command window of the Matlab are as shown in **Figure 5.3**. It contains various information like time at which data is sent and received., the data with its size, the route through which communication is performed and availability of route, information about the route breakage, information regarding the hello messages sent by various nodes.

The default value of the parameters considered for AODV protocol are as shown in the **Table 5.2**. The routing table for each node participating in the network is described in **Table 5.3.**

Active_Route_Timeout	**3 Sec**
My_Route_Timeout	**6 Sec**
Hello_Interval	**1 Sec**
Allowed _Hello _Loss	**2 Sec**
Delete_Period	**2 Sec**
Symbol Period	**1.000000000000000e-06**
Number Of Bits	**500**
Bits/Symbol	**2**
Number Of Symbols	**250**
Threshold	**-48 Dbm**
Frequency	**2.4 GHz**

Table 5.2: Default value of parameters for AODV protocol

Parameters	**NODE-1**	**NODE-2**	**NODE-3**	**NODE-4**	**NODE-5**	**NODE-6**	**NODE-7**
Destination	7	7	7	7	7	7	1
Next Hop	2	4	5	6	7	7	6
Number Of Hops	4	3	2	2	1	1	4
Destination Sequence number	1	1	1	1	1	1	11
Expiry Time	32.5002	32.5003	12.0003	32.5004	12.0004	32.0005	32.5005
Neighbor	-	1	1	2	3	4	-
State	Valid	Valid	Invalid	Valid	Invalid	Valid	Valid

Table 5.3: Routing Table of Each Node

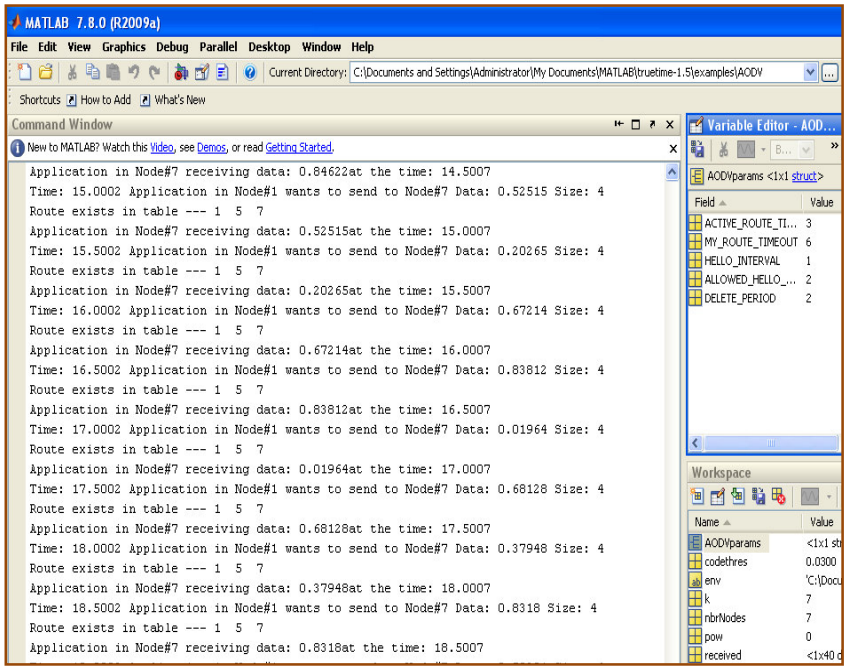


Figure 5.3: Results on Command window for AODV Simulink model

The performance of routing protocol is evaluated using several performance metrics to calculate best path for routing the packet to its destination. These metrics are a standard measurement that could be number of hops, which is used by the routing algorithm to determine the optimal path for the packet to its

destination. This performance metrics are packet delivery ratio, end to end delay and signal reach for the node.

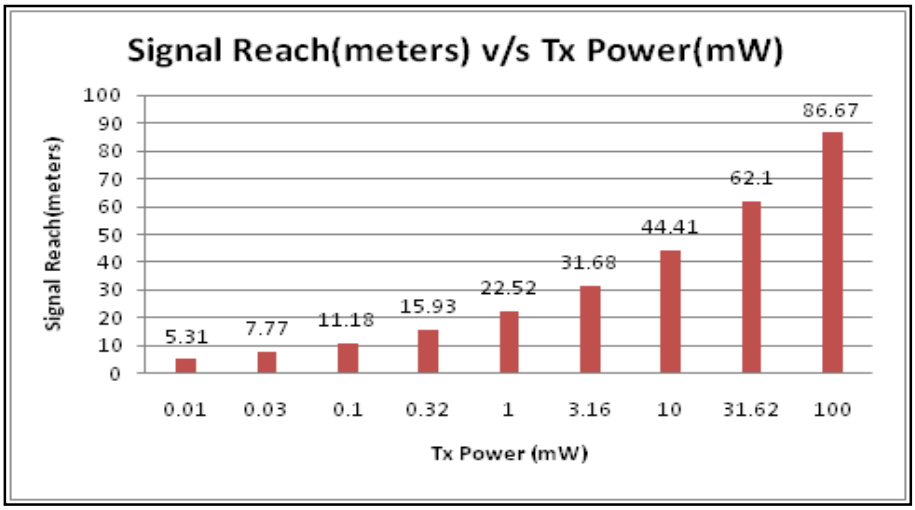


Figure 5.4: Signal Reach v/s Tx Power

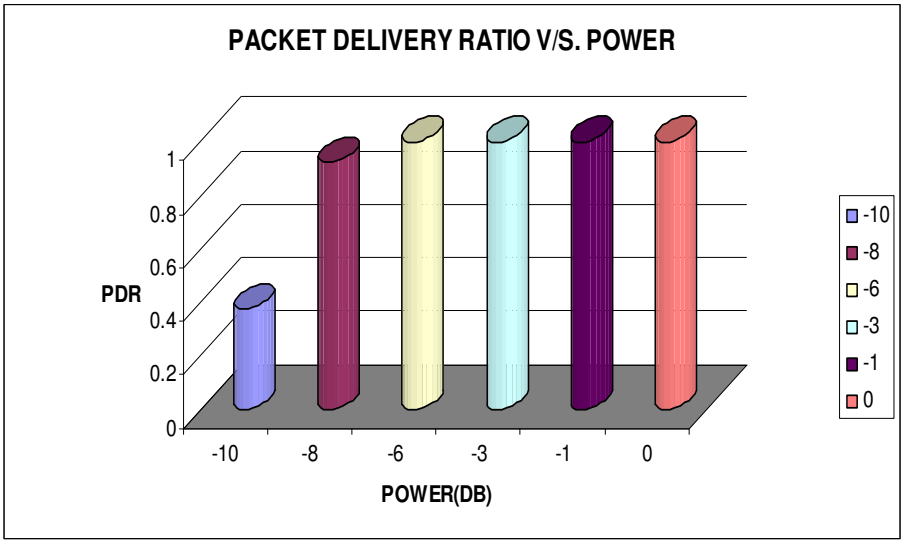


Figure 5.5: Packet delivery Ratio v/s Transmission Power

Figure 5.4 describes the Signal Reach v/s Tx power. From the figure one can observe that as the transmission power increases the transmission range of each node increases. So the signal reach of each node increases and the routing hops can be reduced. **Figure 5.5** shows the change in packet delivery ratio, initially whenever sending node setting the path at that time initially packet delivery ratio is less after that it increases and depend on the mobility of the nodes.

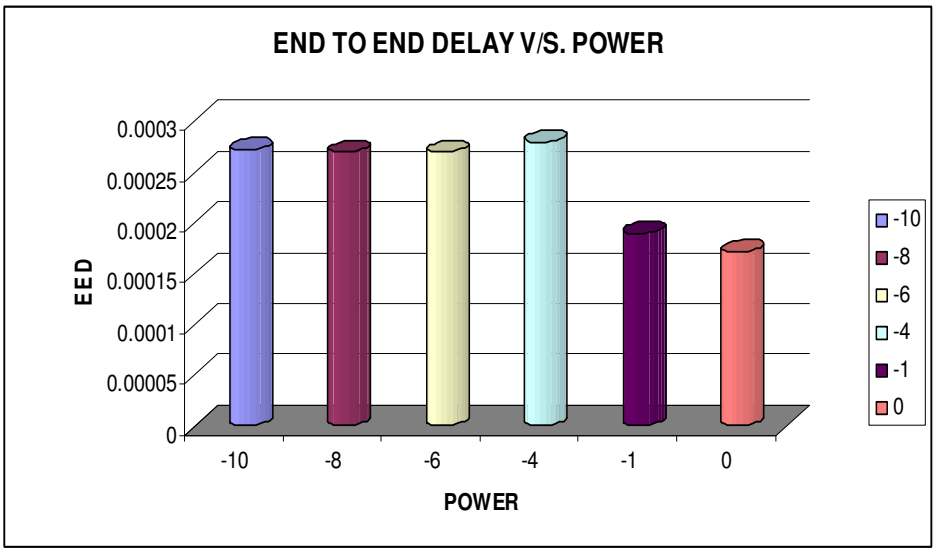


Figure 5.6: End to End delay (sec) v/s Power (dbm)

Figure 5.6 shows the average end to end delay v/s change in power, it explains that as the Tx power increases the average end to end delay decreases because as the transmission power increases signal reach of each node also increases and due to that the number of hops reduces between source node and the destination node which cause to reduce the end to end delay.

5.5 Simulation of Fuzzy based AODV

Recently, many researchers adaptively optimize the ad hoc routing protocols functions and parameters using the Fuzzy Reasoning Algorithm (FRA). The routing protocol parameters can be determined more accurately and dynamically by FRA, instead of static values. The application of FRA to Ad-Hoc network problems allows us to specify these parameters using "if...then..." type of linguistic rules.

In wireless Ad-Hoc networks, Ad-Hoc On-demand Distance Vector (AODV) routing protocol uses static value for 'Active-Route-Timeout' (ART) which indicates the lifetime of an active route in the routing table. As accurate and dynamic ART is more suitable than static one, the fuzzy logic system is used to obtain adaptive ART (fuzzy ART) values [27].

The fuzzy reasoning is used to dynamically configure the protocols parameters instead of using static values. The dynamical configuration can adapt to the changing of the network topology and improve the protocol performance. Using static parameters for the protocols in ad hoc environments that suffer from frequent change of network topology and different traffic intensity may degrade the routing protocols performance [28].

[29] Used a fuzzy reasoning to dynamically configure five routing parameters of AODV (Ad hoc On-demand Distance Vector) routing protocol. They used mathematical models to represent nodes

moving mode and their traffic mode. These models were used to categorize the network environments to nine categories. The fuzzy reasoning was used to estimate the nodes membership degree in these environments. Depending on the node membership degree, the values of the protocol parameters are increased or decreased.

Actually, the fuzzy reasoning can be used more effectively to accurately calculate the real values of protocol parameters that map the status of the node and its links. [29] did this to dynamically adapt routes lifetimes (Natsheh et al., 2005; Natsheh et al., 2006a), "Hello" messages interval time (Natsheh et al., 2006b), and active queue management for congestion control (Natsheh et al., 2007).

This report is concentrating on fuzzy logic implementation on AODV protocol to optimize the maximum signal reach from source to destination. The signal reach is optimized using transmission power and threshold of the signal. With change in signal reach, the hop count also changes. The increases in signal reach results with decrease in number of hop count. The inputs of fuzzy logic are transmission power and threshold with output as signal reach. The mamdani inference system is used to fuzzify the data available to the fuzzy system.

5.5.1 Fuzzy Inference System

A fuzzy engine is typified by the inference system that includes the system rule base, input membership functions that fuzzify the input variables and the output variable de-fuzzification process. Fuzzification is a procedure where crisp input values are represented in terms of the membership function, of the fuzzy sets. The fuzzy logic controller triangular membership functions are defined over the range of the fuzzy input values and linguistically describe the variable's universe of discourse. Following the fuzzification process the inference engine determines the fuzzy output using fuzzy rules that are in the form of "If Then Rules". De-fuzzification is then used to translate the fuzzy output to a crisp value [30].

The signal reach is optimized using fuzzy logic by using transmission power as input and threshold as other input of the system. With increase in transmission power and threshold, the signal reach is maximized. The fuzzy inference system with the inputs: as power, threshold and output as reach of signal from transmitter with mamdani type of inference system. The input1 and input 2 are transmission power and threshold respectively. The output of the system is signal reach.

The respective types of variables for fuzzy inference system, their membership function with the range of each variable are shown in **Table 5.4**.

Types of variables	FIS variables	Range			Types of membership functions
		Low	Medium	High	
Input 1	Power	-10 to -6	-8 to -2	-4 to 0	TRIMF
Input 2	Threshold	-51 to -49.5	-50 to -48	-48.5 to -47	TRIMF
Output	Signal reach	10 to 15	12 to 20	18 to 25	TRIMF

Table 5.4: Fuzzy Inference System

The range of membership function for input 1 which is power varies from -10 to 0 divided into low, medium and high with triangular type of membership function. The range for membership function for threshold is -51 to -47 with triangular membership function. The signal reach is in range of 10 to 25 with triangular membership function.

5.5.2 Rule Base

The rule base for the Fuzzy Inference System is shown in the **Table 5.5** with three variables classified into three Membership functions low, medium and high. The membership functions used are triangular membership function.

Threshold / Power	Low	Medium	High
Low	Low	Low	Low
Medium	Medium	Medium	Medium
High	Medium	High	High

Table 5.5: Rule Base

5.5.3 Rule Viewer and Surface Viewer

The rule viewer of the fuzzy system with 9 various rules are shown in Figure 6.7. The transmission power and threshold are two inputs to the system and signal reach is the output of the system. The rule viewer for fuzzy logic system is as shown in **Figure 5.7.**

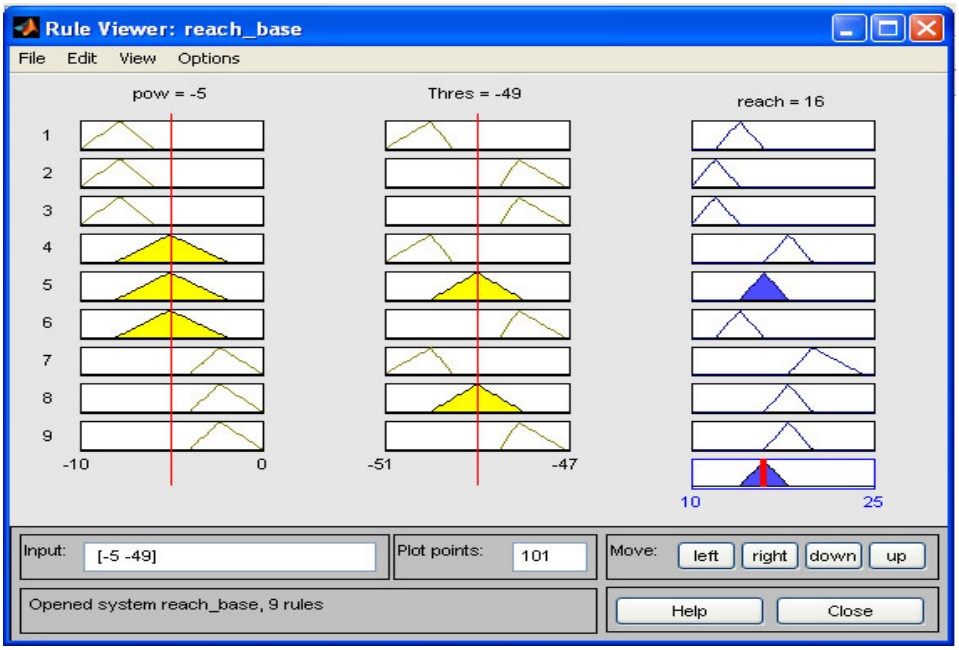


Figure 5.7: Rule Viewer

The surface viewer of the fuzzy system is as shown in **Figure 5.8** for system with 9 rules, 2 inputs and single output.

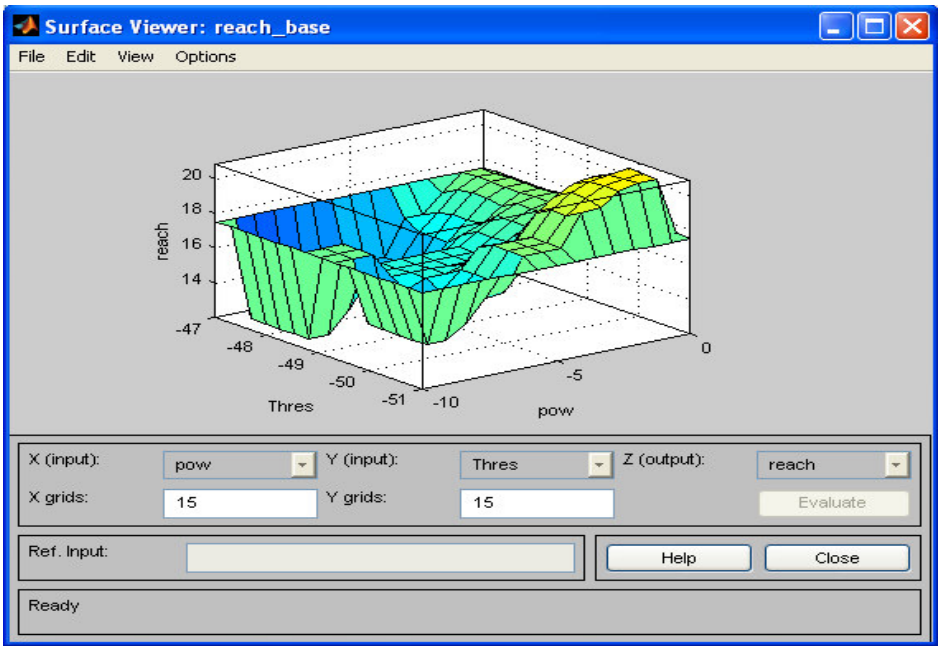


Figure 5.8: Surface Viewer

5.6 ANFIS AODV

The signal reach of AODV protocol is also optimized using ANFIS [31] with same inputs, transmission power and threshold. The implementation of ANFIS is accomplished using ANFIS editor available in MATLAB. The Sugeno type of inference system is used.

The acronym ANFIS derives its name from adaptive Neuro-fuzzy inference system. Using a given input/output data set, the toolbox function ANFIS constructs a fuzzy inference system (FIS) whose membership function parameters are tuned (adjusted) using either a back propagation algorithm alone, or in combination with a least squares type of method. This allows your fuzzy systems to learn from the data they are modeling. The signal reach is also optimized using ANFIS with transmission power and threshold as input. Thus signal reach is output of the Neuro-Fuzzy system and inputs are transmission power and threshold shown in **Figure5.9 (a).**

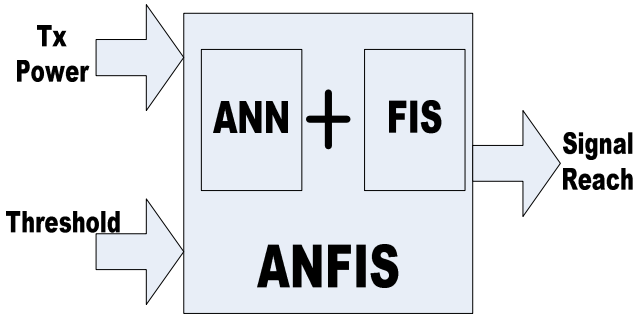


Figure 5.9(a): Block schematic of ANFIS

Input output relations (mapping) in the form of traditional mathematical modeling is replaced by ANFIS learning the synaptic weights by undergoing a training process. ANFIS has built in adaptability or can be trained to modify the weights with the change in environment. It is possible to share the algorithm and structures to have a seamless integration of modules. In traditional AODV, the value assumed to be fixed value. ANFIS is used in our work to make the decision for the Hello time interval based on other parameter.

ANFIS is a network-type structure similar to that of a neural network, which maps inputs through input membership functions and associated parameters, and then through output membership functions and associated parameters to outputs, can be used to interpret the input/output map. The parameters associated with the membership functions will change through the learning process. The computation of these parameters (or their adjustment) is facilitated by a gradient vector, which provides a measure of how well the fuzzy inference system is modeling the input/output data for a given set of parameters. ANFIS uses either back propagation or a combination of least squares estimation and backpropagation for membership function parameter estimation.

By typing anfisedit on command window, opens the ANFIS Gui editor [17]. The network is required to be trained as preliminary step. This is done by supplying training data to the network which helps in training the network. The training pairs are loaded in workspace.

The Sugeno type of inference system is used by ANFIS editor. The structure generated by ANFIS is as shown in Figure 5.9(b).

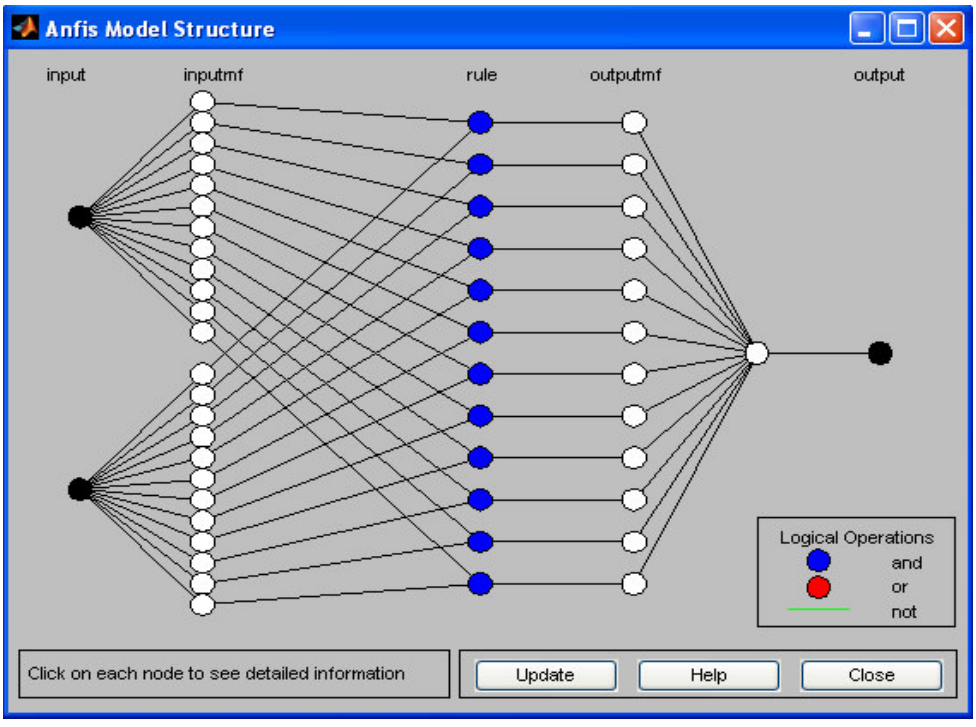


Figure 5.9(b): ANFIS model structure

The MATLAB ([17], [18]) is used to load the training pairs decided by the Sugeno Fuzzy Inference System (FIS) to train the neural network using ANFIS Editor [30].

The sub clustering option is used to generate FIS with 0.001 of error tolerance and 10 epochs. The graph of error v/s epochs is as shown in **Figure 5.10**.

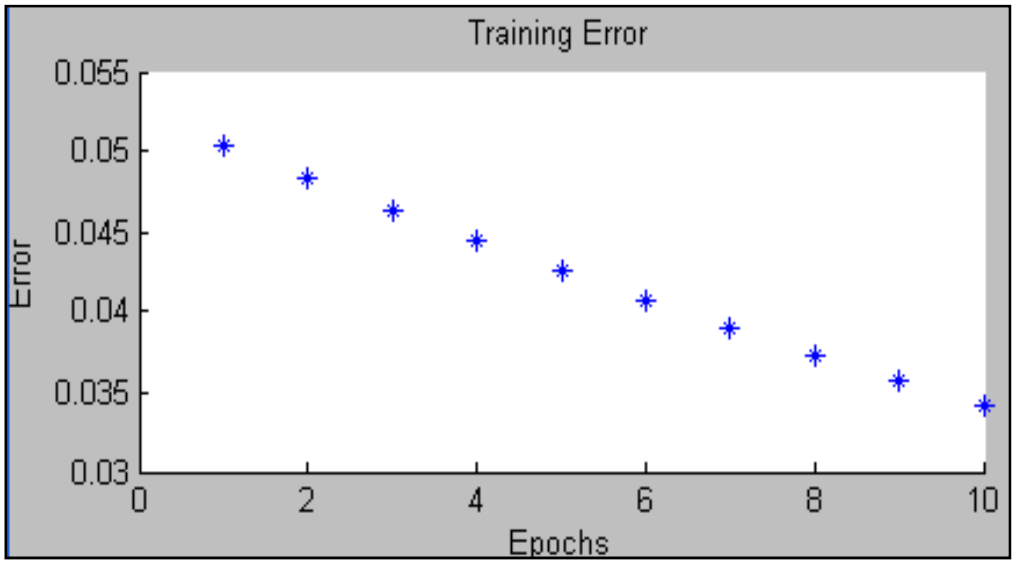


Figure 5.10 Error v/s Epochs

The surface viewer for ANFIS editor is as shown in following **Figure 5.11**.

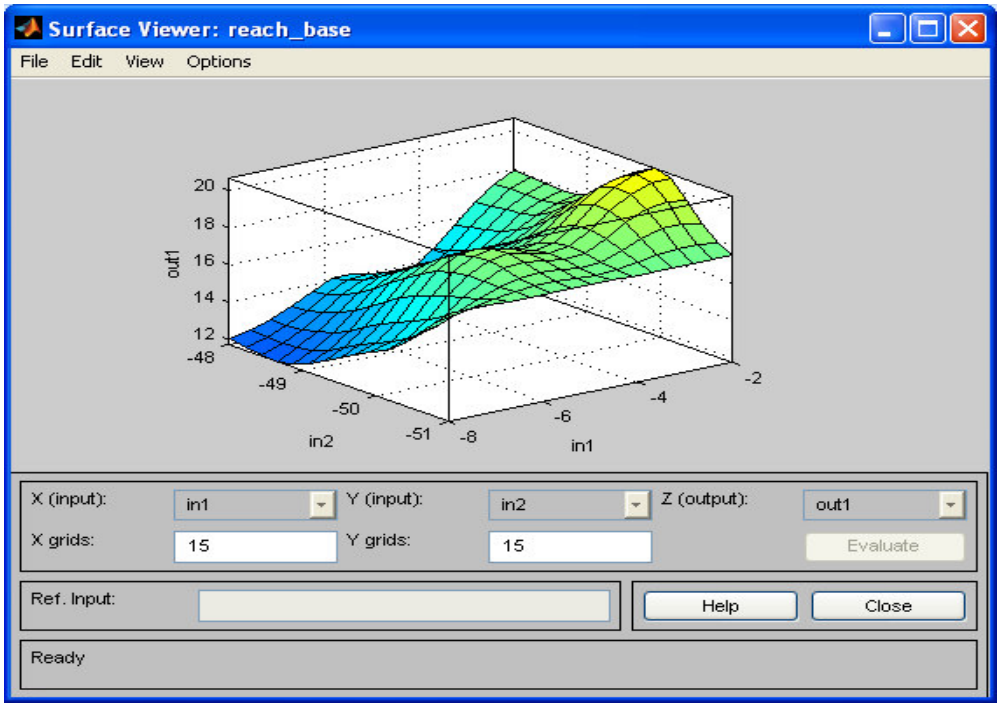
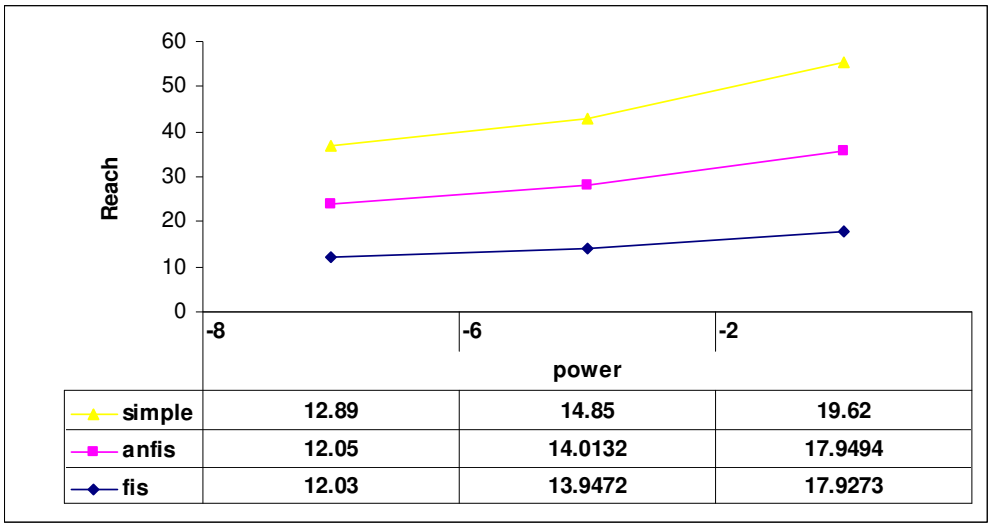


Figure 5.11: Surface viewer for ANFIS

5.7 Comparative Analysis: Classical & Soft AODV[1]

The graph showing comparison between Simple AODV, Fuzzy AODV and ANFIS AODV is as shown in **Figure 5.1**2.



	-8	-6	-2
simple	12.89	14.85	19.62
anfis	12.05	14.0132	17.9494
fis	12.03	13.9472	17.9273

Figure 5.12: Comparisons between Simple AODV, Fuzzy AODV and ANFIS AODV

The Neuro-Fuzzy is also applied to the network for optimizing signal reach. The inputs are same as that of fuzzy logic system. The analysis of the results obtained with fuzzy system and Neuro-fuzzy system says that the signal reach was optimized more efficiently by Neuro-Fuzzy system than fuzzy logic system.

5.8 User Interface Design

A graphical user interface (GUI) [32] is a user interface built with graphical objects; the components of the GUI; such as buttons, text fields, sliders, and menus. If the GUI is designed well-designed, it should be intuitively obvious to the user how its components function.

Applications that provide GUIs are generally easier to learn and use since the person using the application does not need to know what commands are available or how they work. The action that results from a particular user action can be made clear by design of the interface.

MATLAB implements GUIs as figure windows containing various styles of uicontrols objects. We must program each object to perform the intended action when activated by user of the GUI. All of these tasks are simplified by GUIDE, MATLAB's Graphical User Interface Development Environment.

[1] Published in Proceeding of International conference Paper entitled "Signal Reach Computation for AODV Power Threshold in WANET employing Fuzzy Logic" in the WORLDCOMP'10, The 2010 World Congress in Computer Science, Computer Engineering, and Applied Computing held on July 12-15, 2010, Las Vegas, USA in ICWN: Wireless Networks Division page:183-187.

5.8.1 Implementation of GUI

Creating a GUI involves following basic tasks:

- Laying out the GUI components.
- Programming the GUI components.
- Saving and running the GUI.

GUIDE primarily is a set of layout tools. However, GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI. This M-file provides a framework for the implementation of the callbacks; the functions that execute when users activate components in the GUI. GUIDE stores GUIs in two files, which are generated the first time we save or run the GUI:

- **FIG-*file*:** A file with extension .fig that contains a complete description of the GUI figure layout and the components of the GUI: push buttons, menus, axes, and so on. When we make changes to the GUI layout in the Layout Editor, out changes are saved in the FIG-file.
- **M-*file*:** A file with extension .m that contains the code that controls the GUI, including the callbacks for its components. This file is referred to as the GUI M-file. When we first run or save a GUI from the Layout Editor, GUIDE generates the GUI M-file with blank stubs for each of the callbacks. We can than program the callbacks using the M-file editor. Following Figure 5.13 illustrate the parts of GUI implementation.

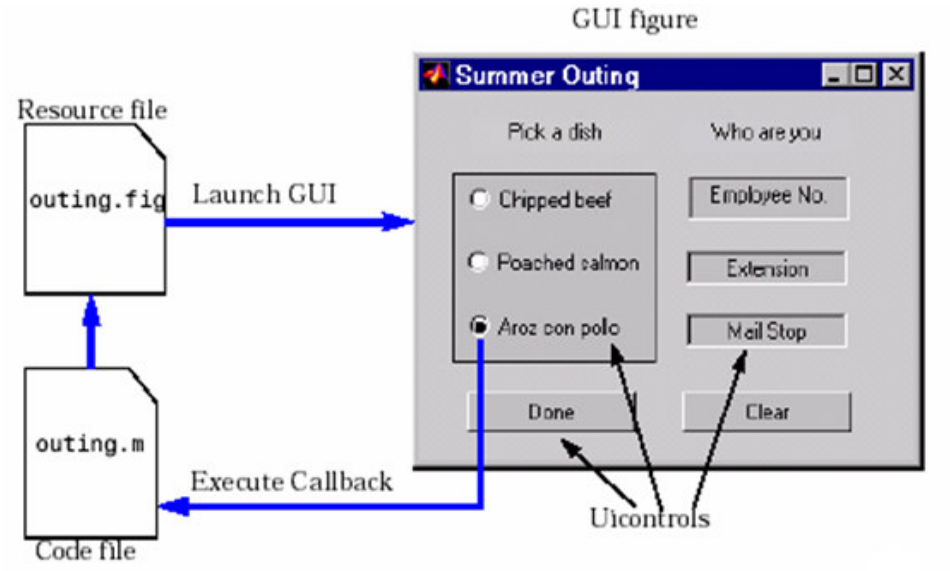


Figure 5.13: Parts of GUI implementation

The layout Editor enables us to layout a GUI quickly and easily by dragging components, such as push buttons, pop-up menus, or axes, from the component palette into the layout area.

5.8.2 Graphical User Interface for Simulator

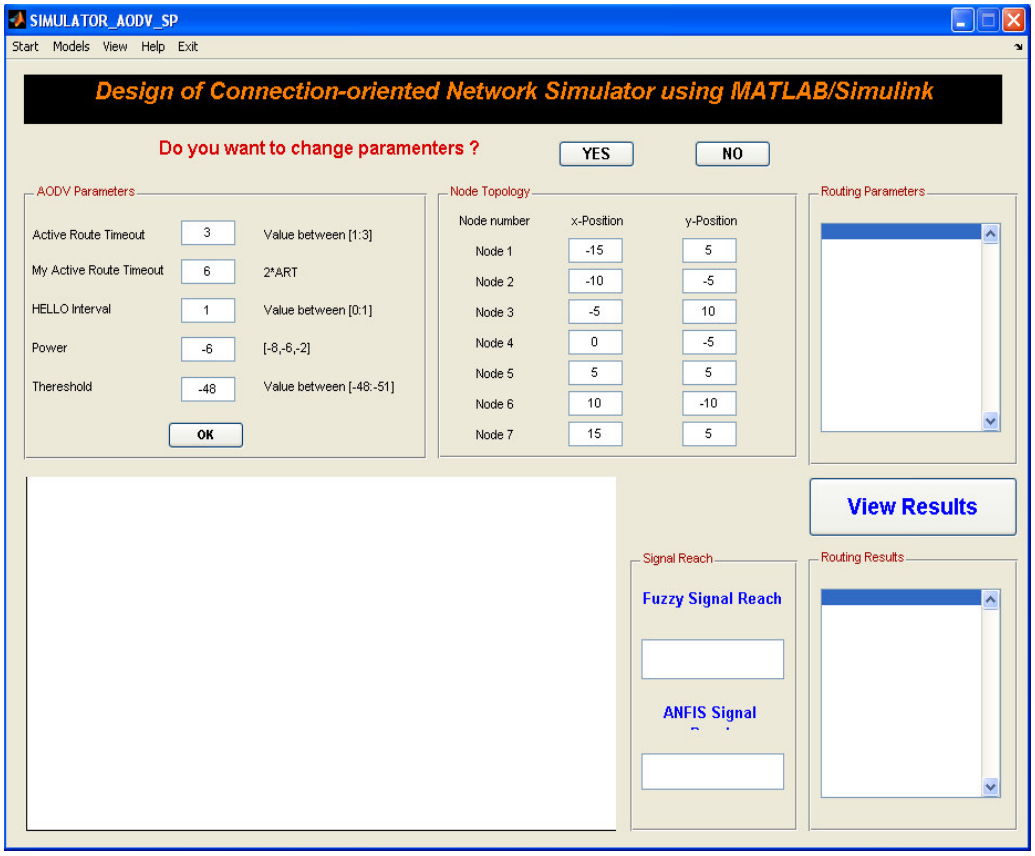


Figure 5.14: User Interface for Application

Graphical User Interface has following facilities:

- To change the parameters of the AODV protocol.
- To initialize the model.
- To set the various parameters of AODV and generate random topology.
- To view node topology and responses of various performance metrics like PDR, EED and RANGE.
- To run various models like simple AODV, Fuzzy AODV and ANFIS AODV through menu bar.
- To view results of workspace in listbox like tout, x-position, y-position, route, threshold, power, received data, etc.
- Help menu to operate the designed simulator.
- Close menu to close the entire opened window related to simulator.

Figure 5.15 shows the GUI which is design to give password protection to above AODV simulator.

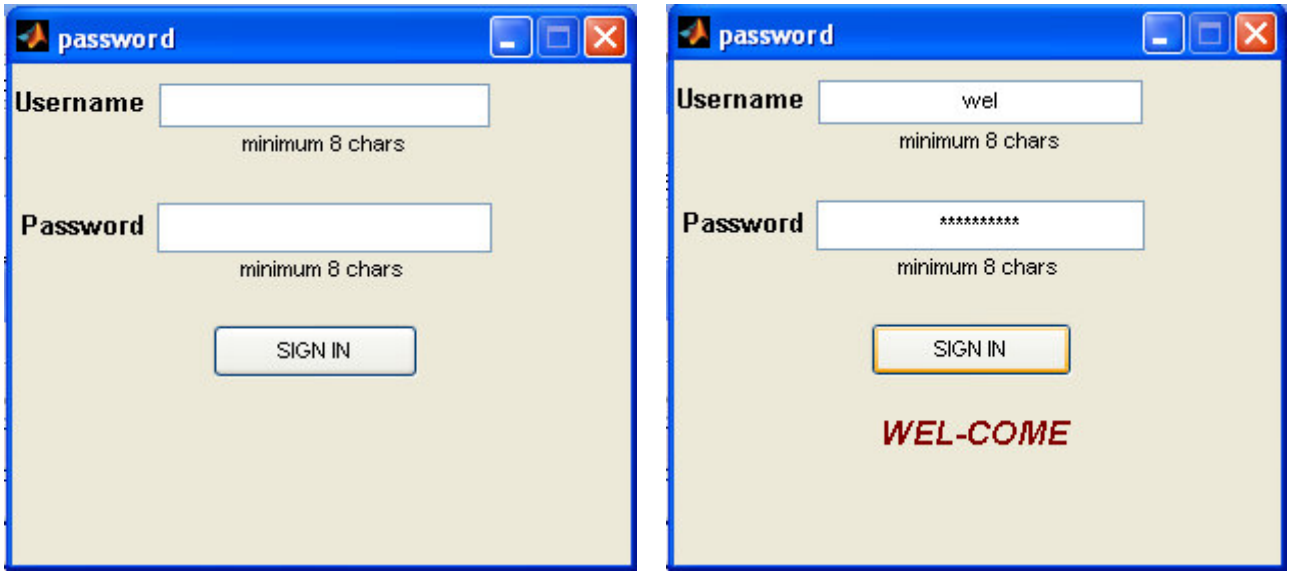


Figure 5.15: GUI for password

The GUI for simulator opens only if correct username and password are entered. The well-come displays if correct user name and password are available to GUI otherwise it will give error of wrong password or username which is shown in **Figure 5.15.**

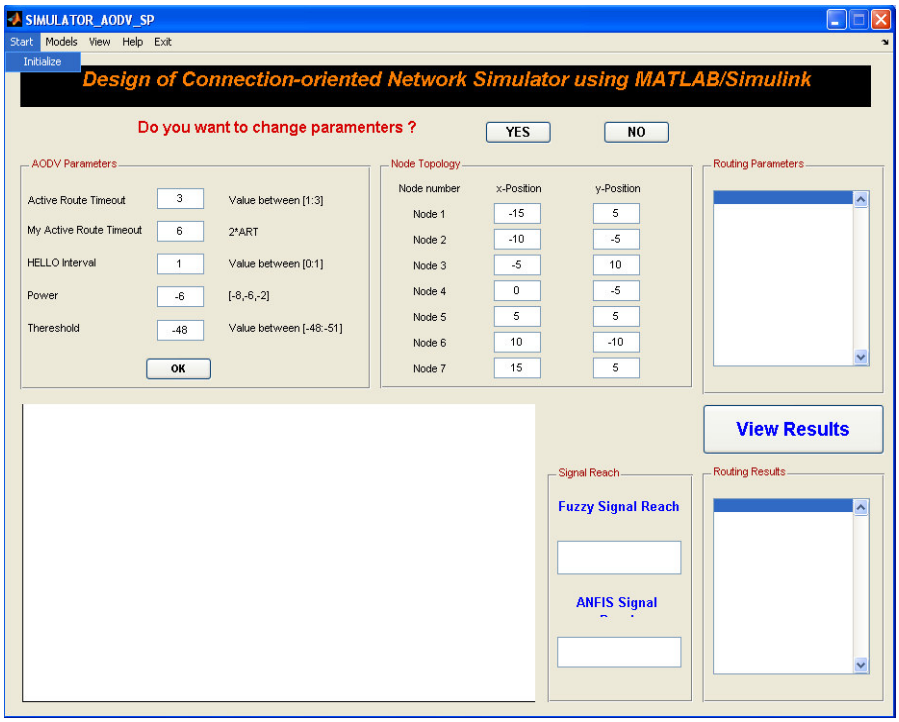


Figure 5.16: GUI showing initialization

Figure 5.16 shows the initialization of user interface can be done from start menu in AODV simulator. The initialization helps to initialize with initial values of model.

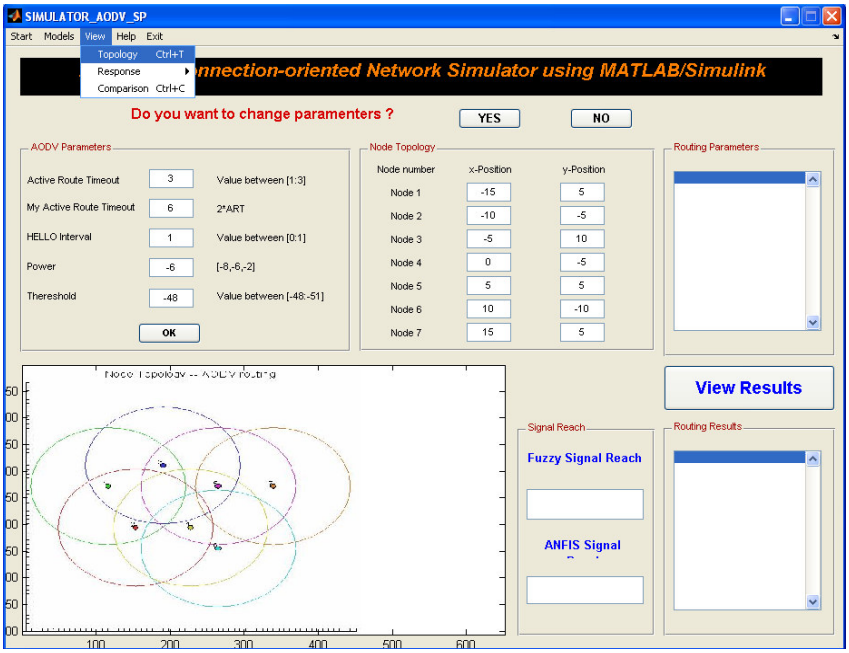


Figure 5.17: GUI showing node topology

The node topology and comparison between various models can be seen through view menu. The node topology is displayed on axes of user interface. This is shown in **Figure 5.17.**

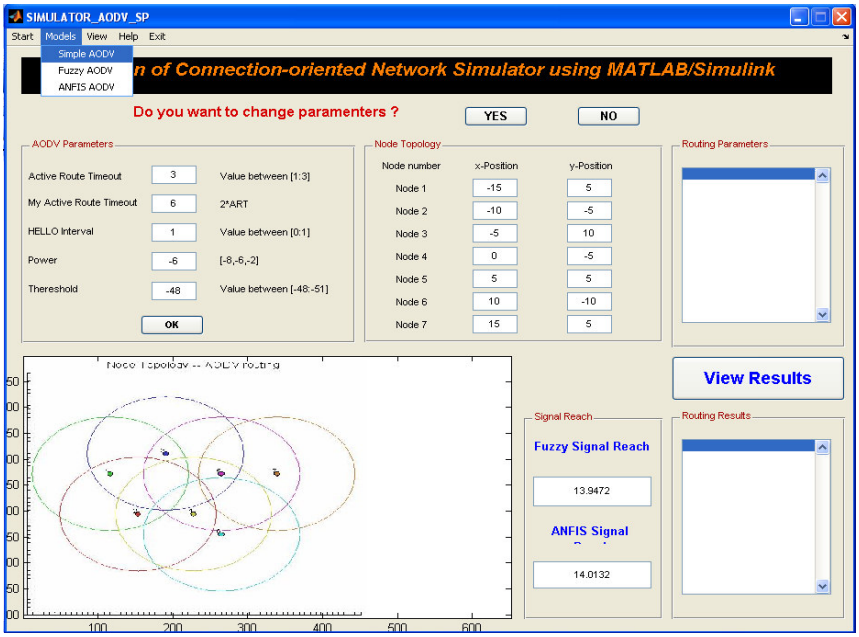


Figure 5.18: Fuzzy AODV and ANFIS AODV

Selection of various models can be done through models menu as shown in **Figure 5.18**.

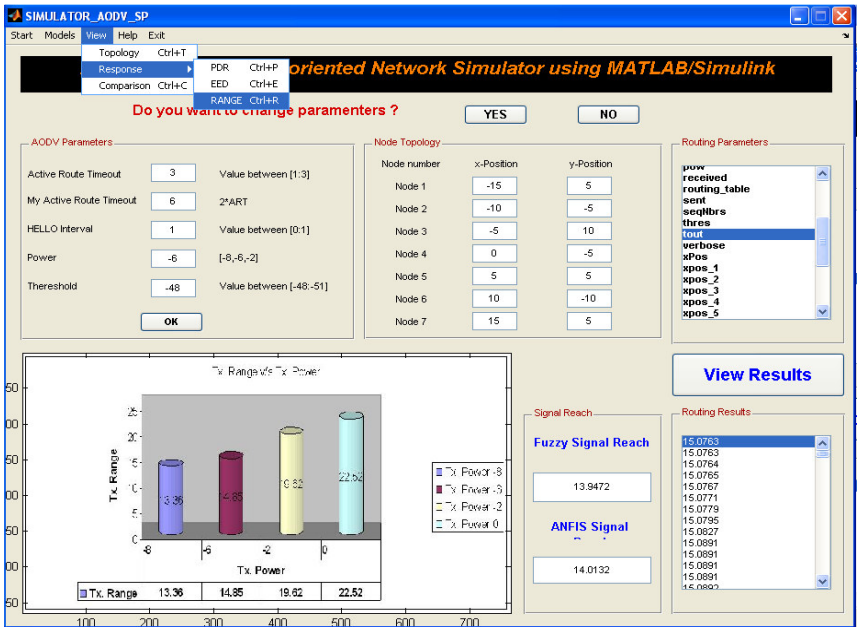


Figure 5.19: Various response for view menu

The response of various performance metrics can be displayed from view menu. The same is shown in **Figure 5.19**. The results from workspace can be displayed by clicking on View Results.

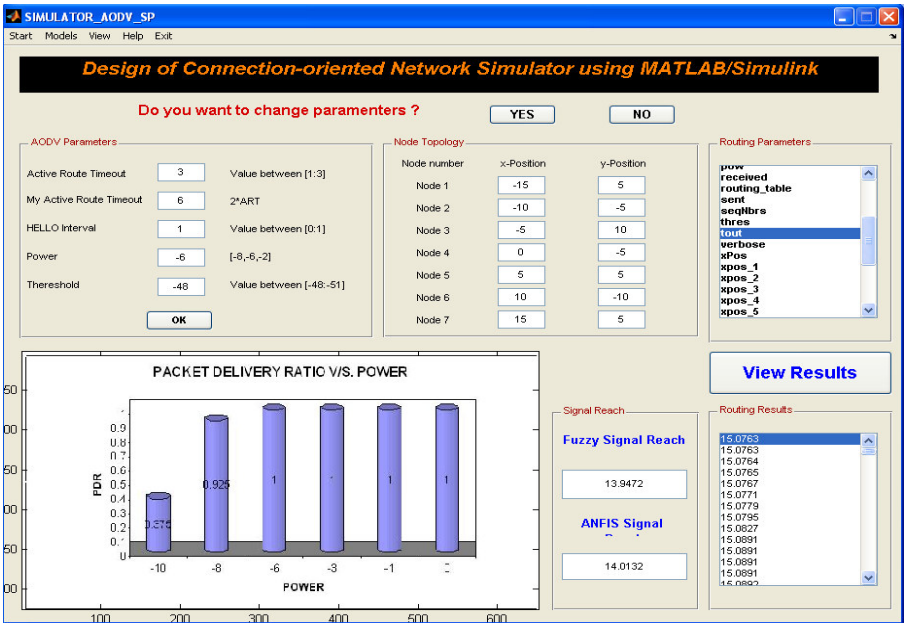


Figure 5.20: Response of PDR and results of workspace

The graph of PDR, results of Fuzzy and ANFIS signal reach, workspace results are shown in **Figure 5.20**.

5.8.3 Files linked with User Interface

Sr. No.	Name	Purpose	Type
1.	Initsim	For initialization of AODV parameters and node positions	m-file
2.	Node_init	For Hello interval	m-file
3.	Moteanimation	For models like Fuzzy and ANFIS	m-file
3.	Response	For various Performance Metrics	Excel file
4.	AODV	For simulation of model	Model file

Table 5.6: Files linked with GUI

Summary

Study of behavior of Ad hoc On Demand Distance Vector routing protocol using simulation on MATLAB has been presented in the chapter. A toolbox TRUETIME to be used with MATLAB is used for simulation of wireless network. Since for the AODV routing protocol signal reach of each node depends on the transmission power and threshold of the received signal strength, effect of change in transmission power of the nodes present in the wireless network is studied by measuring the signal reach of the each node. Algorithm is proposed which uses adaptive value for signal reach based on the transmitted signal power and the threshold value of the received signal strength employing soft computing technique viz. Fuzzy Inference System and ANFIS. The comparison of Performance of routing protocol done between the classical AODV, Fuzzy based AODV and ANFIS based AODV is carried out using Truetime. AODV Simulator is designed to study AODV routing protocol and comparison of the results for it.