ECA MODEL BASED QOS AODV ROUTING FOR MANETS

Raghavendra M.[1] and Pallapa Venkataram [2]

1 Department of Computer Science and Engineering, Siddaganga Institute of Technology, Tumkur 572103, India.
2 Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India.

ABSTRACT

Applications like banking, interactive multimedia, movie on demand, VOIP, etc., are delay sensitive by nature. The QoS given to users will be affected by network delay, which can be mitigated by employing QoS routing and efficient data transfer. To build routing table, normal AODV routing uses flooding technique, which will not consider QoS requirements. Hence QoS based routing which is stable for the entire application is essential, which understands the dynamic nature of the MANET and establishes the required route, in minimum possible time. We have proposed an intelligent routing protocol based ECA model and AODV for establishing QoS route.

The simulation results shows that the ECA model gives better results, while considering the local connectivity time, source to destination connectivity time, number of data packets successfully delivered to the destination, local and global error correction time, compared to AODV.

KEYWORDS

adhoc routing, adhoc wireless networks, Quality of Service (QoS), Event Condition Action (ECA)

1 INTRODUCTION

Mobile Adhoc Network (MANET) is established where no fixed infrastructure is available, and the nodes will communicate with each other on demand. These devices are useful in communications in areas where low human density such as desert, mountain or any isolated area where it is not possible to establish an infrastructure. The examples include human made disaster like war or environmental disaster like flood or earth quake where, we cannot rely on centralized monitoring because every node itself must perform all activities of sender, receiver and router. MANETs are inherently robust, because even if any one node losses the connectivity with its next hop node for a particular source-destination pair, the data can be routed using another nearby node. Higher error rate, obstacles in the electronic wave propagation, dynamic topology, hidden or exposed terminal problem, sending frequent control messages are some of the drawbacks of the MANET. The obstacles in the propagation may lead to transmitted packets becoming garbled causing the received packet in error. Depending on the density of the network, there is always more than one way to reach the desired node(destination), using re-routing process, when a particular node looses its connectivity. Nodes in the MANETs are communicating each other via wireless links and are in constant motion. Maintaining constant connectivity between neighboring nodes is a major issue because received signal strength will decrease when two nodes will move in opposite directions, and finally they will disconnect each other resulting in data loss. Due to dynamic positioning of the nodes, the topology of the network will change and this requires dynamic updation of routing tables and re-routing process is initialized. We need to consider

DOI: 10.5121/ijcnc.2017.9308

different QoS parameters such as bandwidth, delay, throughput and delivery probability in the routing decision. The communication algorithm must be designed such that they must preserve minimum required QoS values. In the process of route construction, we need to find out whether sufficient resources are available in the next forwarding node, which will satisfy the minimum QoS requirements. MANET's have been extensively used for transfer of data generated by different applications. These datas are time bound, security sensitive and needs enormous network resources. For example - Bulky data generated by movie on demand applications, for which enormous resources are needed. In this connection we propose to develop QoS based systems, which ensures reliability with timely delivery of data, maintaining its quality in a higher rate. AODV is an on demand reactive routing protocol, which will exchange routing information among neighboring nodes, reduces excessive memory requirements and excessive route discovery. When sending node wishes to send data packets to a destination, then only a route is needed to be established. In AODV, the rate of path convergence is slow compared to flooding technique employed in the link state routing. The proposed ECA based model is a reactive protocol, where before establishing the path between source and destination node, the required resources are checked and reserved in advance at the intermediate nodes. This advance resource allocation guarantees stable and efficient path. The proposed ECA model has 7 phases, which are presented in detail at Section 4.1. It provides high degree of flexibility to change in the interaction and patterns of mobile node behavior at run time. ECA rule enables the dynamic delivery and life cycle management of traffic at runtime.

The rest of the paper is organized as follows. In section 2, we review the related work. In section 3, different cases of ECA model and our modified model is presented. In section 4, we present ECA sequence diagram and with detailed explanation of the first two phases. In section 5, we present the simulation details and its results. We conclude the paper with conclusion followed by references. At appendix we present, the ECA sequence diagrams for the rest of the five phases.

2 RELATED WORK

Most of the adhoc routing algorithm considers hop count as a metric for route selection. But this is not the efficient technique to construct high quality path because, we are neglecting other QoS parameters such as bandwidth, buffer space and cpu resources available at nodes along the path from source to destination [1]. AODV is one such routing algorithm, which uses hop count or shortest path as the main metric for the path selection. It uses the routing table entries present at each and every node, one entry per destination [2]. This routing table has entries for forwarding the data packet to the destination and bringing the acknowledgement back to the source. Sequence number is needed to be maintained at source to avoid routing loops occur during routing process [7]. Since basic AODV algorithm will not take care of the QoS parameters, we need to add additional features into the algorithm [3]. Detailed description of the AODV RFC 3561 [4] is studied and QoS related algorithms are implemented. These QoS parameters will be affected by different constraints such as node mobility, limited resources of the nodes [5, 6, 8, 9], etc., Due to these reasons, designed application [10] must satisfy their QoS requirements in terms of end to end delay and bandwidth. Since in MANET, each and every node in the network needs to act in multiple roles of sender, receiver and router [11], and are mostly battery powered, energy depletion of the nodes will occur quickly resulting in node failure.

AODV is on demand routing protocol [13], that establishes route to destination node only when required. Frequent route breakage can be avoided, if we consider proper QoS requirements, while creating the route. Each mobile agent may have different QoS requirements, that needs to be considered before establishing the connection between source and destination. Also there is must be a path between the current node and next hop node[14]. When both of the nodes are operational, then only the agent will visit its next destination taking that path.

IEEE 802.11 is a wireless based standard [15], which is used to support multiple types of communication services such as data, voice, image and video with different QoS requirements. Due to Node mobility, a continuously changing communication topology is created, in which paths break and new one forms dynamically. The routing table of each router in an adhoc network must be kept up-to-date. This causes link failure in the channel during the communication, leads to the degradation of the QoS, as MANET applications requires strict QoS requirements [16]. To deal with this problem, we can check the availability of the resources before constructing the path.

Due to bandwidth constraint and dynamic topology of the mobile adhoc network [17], supporting QoS is a challenging task. Therefore the routing protocols in adhoc networks must be adaptive to face frequent topology changes because of node mobility to support QoS routing. The parameters such as delay, bandwidth, jitter, loss rate in the network should be available and manageable [18]. Bandwidth is a QoS routing parameter in a real world mobile network [12], which is sufficiently needed to deliver the packets to the destination node with minimum delay. In adhoc networks, certain QoS parameters like error rate, delay and packet loss are increased, while throughput and delivery rates are decreased in transport layer due to MAC issues. This is due to the impact of multiple wireless hops and node mobility in the network.

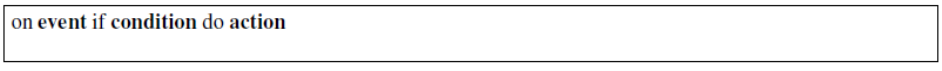
Adhoc networks are usually prone to transmission errors. To reduce such errors, transmission techniques which combines Hop-by-Hop (HBH) and End-to-End (E2E) retransmission schemes [19], are implemented, to ensure the reliability. In the HBH retransmission scheme, a lost packet in each hop is retransmitted by the intermediate node to ensure link level reliability. In E2E scheme, source will wait for ACK of data from destination node. If source is not received ACk with in global time out period, then it will retransmit the data once again.

ECA-PC (Event Condition Action - Post Condition) rules [20] helps to manage the events happening at the ubiquitous computing system. In ECA model, policy rule may tell when event occurs in a situation, where conditions is true, then action is executed. These rules are useful to define an algorithm in contextaware applications [21].

ECA rules can be written/implemented in several language such as c, xml [22], android, java [21, 23] etc.These models can be used in systems like database, sensor network, adhoc network, e.t.c.

3 ECA MODEL

ECA refers to the structure of active rules in the event driven architecture and active data base systems. These rules consists of 3 parts namely - event, condition and action. Event is triggered by signals when certain conditions tested to be true. It will execute certain actions, that will update or operate on data items. Programs stored in the mobile nodes are executed on event driven basis. ECA rule will define and implement application logic, which are used and implemented in these mobile nodes, in different combinations. There are different ways of working in the ECA model as shown in the Fig.1.



3.1 FUNCTIONS OF ECA

Case 1. In this model, if certain conditions are met in an event, an action is triggered. If at least one condition or set of conditions or certain combination of conditions are satisfied, then the statements in the actions are got executed.

Case 2. Event combined with satisfied condition, triggers particular action.

Case 3. Action will be triggered, when two incoming events will join together to execute an action as one or more events needs to be executed.
For the present work, model presented in the Fig.1 is considered to be more suitable, where condition is

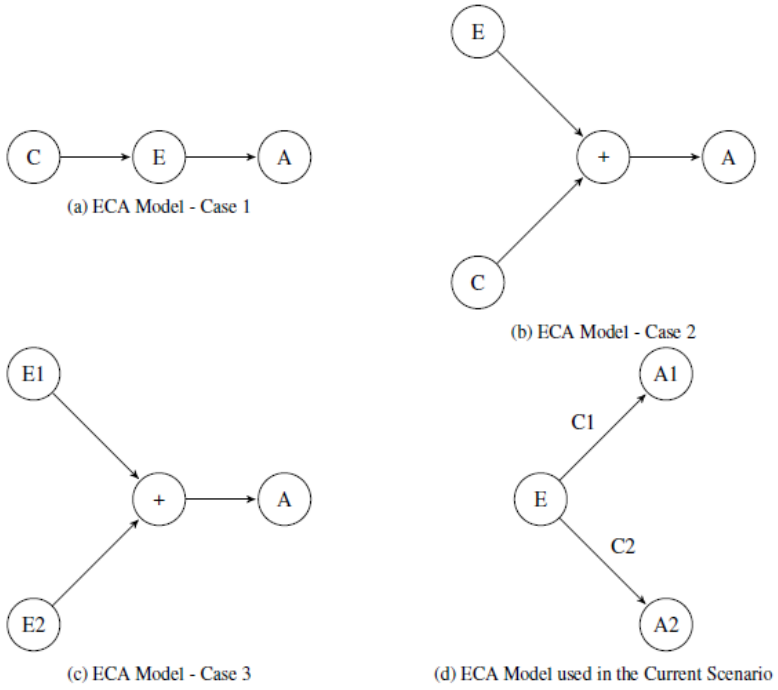


Fig. 1: Cases Involved in ECA Model

written as the label in the state diagram. Because in an event, if conditions are satisfied, then action will be executed. ie. If condition C1 is satisfied then action A1 is executed, otherwise if C2 is satisfied then action A2 is executed. As a general example, we will consider the task of monitoring the quantity on hand for each item in the production cycle. If the quantity is below certain threshold, then a reorder procedure has to be initiated automatically as shown in Fig.2. Thanks to [24] [25].

Event : Update_quantity_on_hand(item)
Condition: quantity_on_hand(item) + quantity_on_order(item) < threshold(item)
Action : reorder(item)

Fig. 2: ECA Model for Producing an Item

3.2 Scientific Significance Of The ECA Model

Since ECA model confirms the resource availability of the next forwarding node, the resulting route is always QoS aware. Due to this the probability of network disconnection is very low.

3.3 PROPOSED MODEL AND REAL TIME PRODUCT

The generated QoS path will guarantee lossless data transmission. Since it is a QoS aware path the data will able to reach the destination in first attempt. The number of retransmissions required will be less or nearly nil. This will reduce congestion in the network and the speed of the data transmission increases. This will results in increased network speed. So if the new protocol will implemented in the routers, the speed of the data transfer will increase and as a result the overall Internet speed will increase.

4 PROPOSED ECA BASED QOS ROUTING MODEL

Algorithm 1 Resource Check Procedure

Input: req_bs, req_bw, req_ts, req_ene { req:required , bs:buffer space, bw:bandwidth ts:cpu time slot, ene:energy }
Output: ACK , NACK
1: **if** req_bs < avail_bs And req_bw < avail_bw And req_cpu_ts < avail_cpu_ts And req_ene < avail_ene **then**
2: return ACK
3: **else**
4: return NACK
5: **end if**

4.1 ECA STATE DIAGRAM

The overall process of ECA is divided into 7 phases. 1. Data Generation (DG) 2. Route Request (RREQ) 3. Route Response (RSP) 4. Timeout/Error at Route Reply pair(ERR) 5. Data Transmission (DT) 6. Error Handling at Data Transmission (EDT) 7. Route Maintenance (RM)

Sequence diagram for data generation is shown in Fig.3. The route establishment sequence is shown in Fig.4. ECA sequence for these two phases are list at Table 1.

The remaining sequence diagrams and tables are provided at the appendix.

Phase 1 : Data Generation(DG) :

The invoked application program will generate data, if sufficient resources are available(DG-C1). The routing table present at the route cache is searched for the valid entry for the source-destination pair. If valid entry is detected (DG-C2), then packets will utilize the same path to take the data to the destination (DG-A2). If no valid entry is found(DG-C3) or if particular source-destination entry does not exists(DGC4), then new RREQ procedure (DG-A3) is invoked to search for a new route.

Phase 2 : RREQ Preparation (RREQ) :

The RREQ packet is prepared at the source node (RREQ-A1), when no route cache entry is detected during the event RREQ-E1, provided OS can reserve sufficient resources (RREQ-C1) to perform the task. After successfully creating the RREQ packet, the resources required to broadcast the packet (RREQ-A2) is checked (RREQ-C2). Event (RREQ-E3) will be triggered when, the RREQ packet is received, where destination node field is checked against the received node. If the RREQ packet is reached to destination (RREQ-C3), then it is processed (RREQ-A3) and route reply (RREQ-A4) packet is prepared. If the RREQ packet is at the intermediate node

(RREQ-C4), then it is further forwarded to the next node, which satisfies the required resource constraints (RREQ-C5) at the event (RREQ-E4).

If sufficient energy is present at the next node (RREQ-C6), then (positive) ACK(RREQ-A7) otherwise NACK(RREQ-A8) is returned. If ACK is received at the intermediate node (RREQ-E5), and if (RREQ-C7 and RREQ-C8) conditions are satisfied, then route cache table entry needs to be updated (RREQ-A9),123 followed by releasing the resources (RREQ-A10). If ACK is received(RREQ-E5), with in local time out period, then n2n connectivity is successfully established. If NACK is received at the intermediate node(RREQ-E6), then another node is queried regarding the resource availability (RREQ-A5), and the process is repeated by the action (RREQ-A13). This involves sending RREQ packet to the node(RREQ-A12), using random strategy for next hop node selection (RREQ-A12) and RREQ local timer is started to wait for the ACK (RREQ-A13).

RREPLY timer is already started and it is waiting at the source node during RREQ packet is broad casted. Event RREQ-E7 is triggered when RREPLY is not received with in RREPLY s2d global time out period. Now (RREQ-A14) action routine is invoked in order to check RREQ re-transmission count, and this will trigger event (RREQ-E8) at the source node.

If the RREQ attempt count at the source node exceeds the maximum permissible value (RREQ-C10), then no more RREQ attempt is made and the searching process for the route is halted (RREQ-A15).

If the RREQ attempt count at the source node is within permissible limit (!RREQ-C10), then RREQ process is re-initiated by the action (RREQ-A16). The same action will be executed if the global Route Error signal is received at the source node within RREPLY s2d timeout period.

If the maximum RREQ attempts exceeds at the intermediate node (RREQ-C10), at the event RREQ-E9, then the local broadcast error is sent to all the neighboring nodes within the radio range and route error signal is sent to the source node to recalculate the fresh route. RREQ process is re-initiated by the action RREQ-A16 at the source node.

Local RREQ timeout will occur at the intermediate node that will trigger, the event (RREQ-E9). The number of RREQ attempts to establish the route to the next hop node is checked and if it exceeds (RREQC11), then local broadcast error message is generated (RREQ-A17), and route error signal is sent to all neighboring nodes with in the radio range to update their Routing Table entries. Route error signal is also sent back to the source node(RREQ-A18), requesting to recalculate the fresh route.

If the RREQ attempt count is within the permissible limit(!RREQ-C11) at the intermediate node, then id of the next node to send the RREQ packet is generated on random basis and it is transmitted to the next node during the action (RREQ-A19).

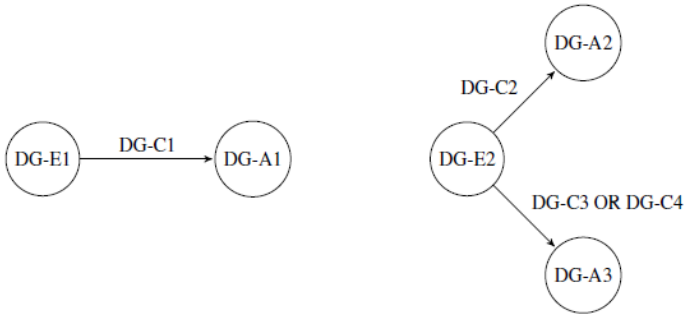


Fig. 3: ECA Sequence For Data Generation and Transfer

The Alg.1, describes the resource check procedure, where memory space, bandwidth, cpu time slot and residual node energy are inspected to process the RREQ. The algorithm checks the resource availability at the potential receiver. If sufficient resources are available at the next hop node with in the radio range, then ACK is returned otherwise NACK is returned.

ECA model tries to enquire, the availability of resources in advance before transmitting RREQ packet. If the minimum required resource constraints are met, then only the RREQ packet is forwarded and a QoS satisfied path will be constructed which will make efficient resource utilization. Since the QoS requirements

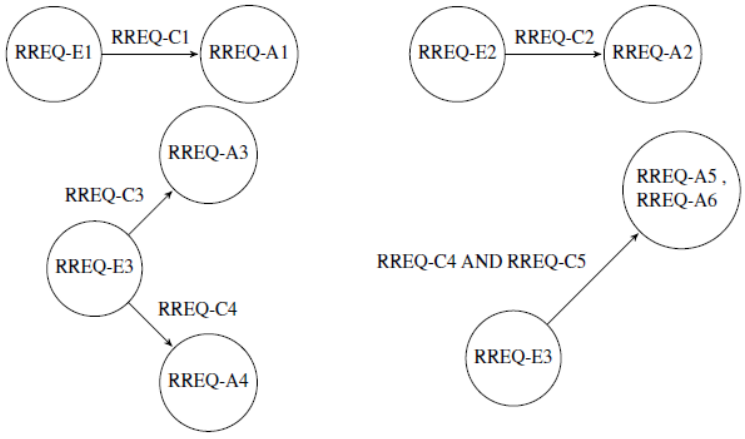


Fig. 4: ECA Sequence For Route Establishment

are considered while determining the path, the path becomes more stable compared to AODV, where next hop node is decided using minimum hop count towards destination node and no other QoS parameters are considered.

Phase 3: Route Response (RSP):

When RREQ packet reaches destination, RREPLY packet will be constructed, which will use the route just established, but in reverse direction, towards the source node.

Phase 4: Error Handling at Route-Reply pair (ERR) :

The source node is waiting for RREPLY after it transmits RREQ packet. At RREPLY timeout, if the route response packet is not received, then the source node needs to retransmit RREQ packet once again (Request is re-initiatized), till $MAX\_RREQ\_ATTEMPTS$ count is reached. When the number of retransmission count was crossed, the "Destination UnReachable" error message is sent to the application at the source node.

Phase 5: Data Transmission (DT) :

This phase makes use of forward routing table, in which there is unique best QoS entry for the next hop is present. From this table next hop information is collected and appropriate timers are initialized in this phase.

Phase 6 : Error Handling for Data Transfer(EDT) :

The algorithm maintains local timeout procedure (RTS, CTS) for node to node (intermediate) and global RERR mechanism for end to end (source to destination) connectivity. Timeout: The connectivity is checked using RTS, CTS signals. If CTS is not received within timeout, then RERR message needs to be generated. Packets will be placed at the buffer and appropriate Route Maintenance actions needs to be taken to re-establish the connection. However unique sequence number to the original packet ensures the rejection of duplicate packets received at the receiver(if any).

Phase 7 : Route Maintenance (RMN):

This phase will handle disconnection occur due to either node movement or energy depletion at the battery. In Adhoc networks, since the nodes are moving, disconnection can occur frequently, causing the termination of the connection between source and destination nodes. In both cases, the route needs to be re-established quickly, without much data loss.

Local correction can be done if the condition, $s2d\_RTT > Repair\_Time$ is satisfied. Otherwise if the time taken for local repair is longer than $s2d\_RTT$, then RERR message needs to be sent back to source which results in global route correction, as this RERR message will be informing to source, and all the nodes in the path regarding the disconnection. Now the source node needs to generate fresh RREQ, which establishes new route to the destination.

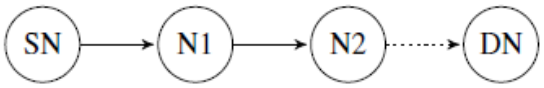


Fig. 5: Source to Destination Communication

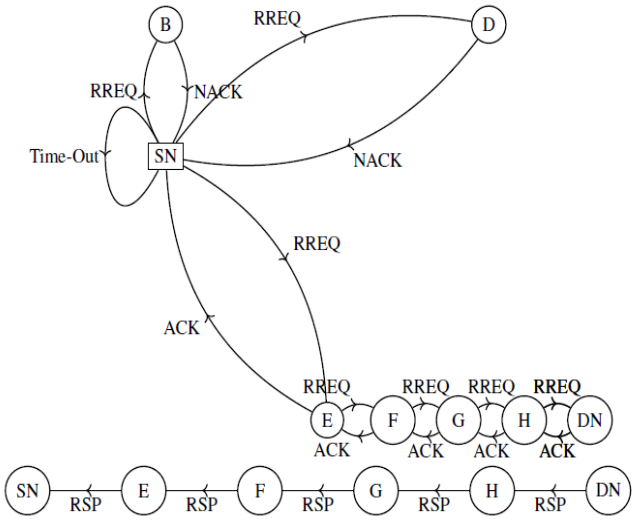


Fig. 6: ECA - Route Establishment State Diagram

In the ECA model, the source node (SN), will broadcast RREQ's to all the nodes in the radio range as shown in the Fig.6, and it is received by the nodes B,D and E. The nodes B and D sends NACK, because it will not have sufficient resources to handle the current traffic. But node E sends ACK and the RREQ further propagates to E, then F and so on till it reaches to the destination node(DN). All these intermediate nodes already received ACK in the process and finally DN will generate RSP packet and it will travel back to the SN using the already created path.

The time taken to establish local (n2n) connectivity can be expressed using the equation,

$$n2n\_con\_tm(SN,E) = RREQ\_tm(SN,D) + NACK\_tm(D,SN) + RREQ\_tm(SN,B) + NACK\_tm(B,SN) + RREQ\_tm(SN,E) + ACK\_tm(E,SN) \quad (1)$$

where $n2n\_con\_tm(SN,E)$ is the time delay to receive positive ACK after RREQ packet is broad-casted. $RREQ\_tm(SN,D)$ is sending RREQ packet to the neighboring node E. $NACK\_tm(D,SN)$ is returning NACK for insufficient resources. Finally at node E, the required resources are available and it will return $ACK\_tm(E,SN)$.

The communication between source node (SN) to destination node (DN) is multi hop as the packet required to travel through one or more intermediate nodes (N1,N2, ...), till the destination node is reached, as shown in the Fig5. The time spent in the process can be mathematically expressed as,

$$s2di\_tm(SN,DN) = tm(SN,E) + tm(E,F) + tm(F,G) + tm(G,H) + tm(H,DN) \quad (2)$$

where s2di tm(SN,DN) is the time taken by the RREQ packet to cross all the intermediate nodes in the path.

tm is the time taken by the RREQ packet to establish intermediate connection between current node and next hop node.

Event	Condition	Action
1. Application Generates data (DG)		
DG-E1: Application_triggered(SN)	DG-C1: check_resources_for_data();	DG-A1: Generate_data();
DG-E2: Data_Pkt_Ready(SN)	DG-C2: route_cache(s,d).path == valid	DG-A2: send_data();
DG-E2	DG-C3: route_cache(s,d).path == In_valid	DG-A3: NRCED();
DG-E2	DG-C4: route_cache(s,d).path == not_exists	DG-A3();
2. Route Request (RREQ)		
RREQ-E1: NRCED(SN)	RREQ-C1: check_resoures_for_RREQ_prep();	RREQ-A1: prepare_RREQ_pkt();
RREQ-E2: RREQ_pkt_Ready(SN)	RREQ-C2: check_resources_for_RREQ_bc();	RREQ-A2: broadcast_RREQ();
RREQ-E3: RREQ_received(ND)	RREQ-C3: node.id == destn	RREQ-A3: process_at_destn();, RREQ-A4: prepare_RREPLY();
	RREQ-C4: node.id == IN AND RREQ-C5: check_resources_to_recv_RREQ();	RREQ-A5: process_RREQ();, NN=generate_NNid(); RREQ-A6: forward_RREQ(NN);
RREQ-E4: Resource_check_req(ND)	RREQ-C6: node.energy > 0	RREQ-A7: reply_ACK();
RREQ-E4	!RREQ-C6	RREQ-A8: reply_NACK();
RREQ-E5: ACK_recvd(IN)	RREQ-C7: dis(sender,next_hop) ≤ radio_range AND RREQ-C8: time(RREQ,ACK) < LTO	RREQ-A9: MURCE();, RREQ-A10: RRUR();
RREQ-E6: NACK_recvd(IN)	RREQ-C7() AND RREQ-C9: time(RREQ,NACK) < LTO	RREQ-A5(); RREQ-A11: process_NACK();, RREQ-A6();, RREQ-A13: Wait_for_ACK();
RREQ-E7: Timeout_RREPLY(SN)		RREQ-A14: check_RTX_count();
RREQ-E8: check_RTX_count(SN)	RREQ-C10: exceeds_MRRTXAS !RREQ-C10()	RREQ-A15: halt(); RREQ-A16: re_initiate_RREQ();
RREQ-E9: Timeout_ACK/NACK(IN)	RREQ-C11: exceeds_MRRTXAIN	RREQ-A17 LBC_ERR();, RREQ-A18: send_RERR(SN);
	!RREQ-C11()	RREQ-A19: RTX_RREQ(NN);

Table 1: ECA AODV Description

The Table 1, shows the ECA sequence for 1st two phases, where LTO is the local timeout period, as it is

calculated by the equation,

$$LTO = 2 * local\_rtt \tag{3a}$$

$$local\_rtt = (ts + tp + tr) * 2 \tag{3b}$$

where $ts = tr = data\_pkt\_bits/10Mbps$ and

$tp = radio\_range/(2 * 10e8)m/sec$

$local\_rtt$ is the time taken by the packet to reach the receiver and come back where the receiver is within radio range. If the ACK/NACK is not received in this stipulated time (ie local timeout occurs), then RREQ needs to be retransmitted. ts is the time taken to send the packet to the wireless media (sending time), tr is the time spend to receive the packet in the next hop node (receive time) and tp is the time taken to propagate the packet to the next hop node (propagation time) through wireless media.

5 SIMULATION AND RESULTS

Parmeter	Value
Area Of deployment	1Km * 1Km
Max Speed of Node Mobility	10 m/s
Max Simulation Time	10 Sec
RADIO_FREQ	2.4GHZ
DATA_RATE	10Mbps
SPEED	2*10e8 mps
RT_SRCH_TM	5*10e-6 sec
CPU_SPEED	3*10e6 bps
MAX_HP_CT	16
RREQ_ACK_SIZE	64 bits
RREPLY_PKT_SIZE	160 bits
DATA_PKT_SIZE	1024 bits
RERR_PKT_SIZE	160 bits
INIT_ENERGY	10.0 J
RADIO_RANGE	250 m

Table 2: ECA-AODV Simulation Parameters

5.1 SIMULATION ENVIRONMENT

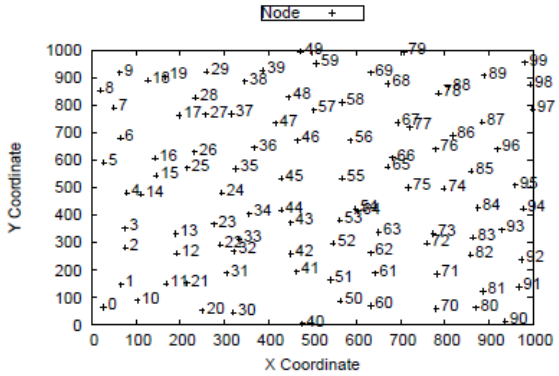


Fig. 7: Initial Node Deployment

MANET having 100 nodes deployed in random basis over a region of 1Km * 1Km area as shown in the Fig.7. The nodes are allowed to move arbitrarily with maximum speed of 10 m/s in different directions.

The deployed nodes will be made to run m number of delay sensitive applications at any given point of time. So in this environment, we develop a quality of service routing by using proposed protocol for the given application.

Both algorithms are simulated in the above deployment for 10 seconds and the trans-receiver circuitry works at operating radio frequency of 2.4GHz. The wireless interface circuit transmits and receives data at the rate of 10Mbps. The data signal will travel in the wireless media (free air) at the speed of 2*10e8 meter per second (mps). The routing table search time (RT_SRCH_TM) is assumed to be 5*10e-6 sec. The input link bandwidth ($IN\_LINK\_BW\_10$) and output link bandwidth ($OUT\_LINK\_BW\_10$) at the interface has bandwidth 10*10e6 bps. The cpu processes the data at the speed of 3*10e6 bps. The IO and CPU buffer can store up to 20 packets at any instance of the simulation. The packet sizes of RREQ, RREPLY, DATA and RERR packets sizes are defined as per AODV RFC. All the nodes are initialized with residual energy ($INIT\_ENERGY$) of 10 Joules(watt-sec). The radio range of all nodes are uniform and is 250m. Each packet in the simulation can cross (MAX_HP_CT) 16 hops. The simulation parameters are listed in the Table 2.

We have simuilated ECA based AODV for creating a path between source and destination node with several types of applications. These built path varies from three node path to nine node path and results obtained for all the paths. Similar experiments have been done by simulating the original AODV protocol for the comparision.

The simulation results shows that ECA is performingwell compared to originalAODV. In the ECAmodel, before constructing the path the senders will confirm the availability of sufficient resources at the receiver node. This creates resource aware path between source and destination node makes efficient resource utilization.

The delay involved is minimum in ECA compared to original AODV where the simulation uses an iterative method to select the next hop in the route creation process.

In the ECA model, the next node in the route creation process is selected in random basis, with a probability p. Due to this the convergence of connectivity, in most of the cases, occurs at the early stages of the loop, resulting in minimum delay. But in case of AODV, which uses serial loop in the searching process which may select the same next hop node very frequently, will drain the battery of few nodes in the network, causing the connectivity failure (disconnection).

5.2 RESULTS

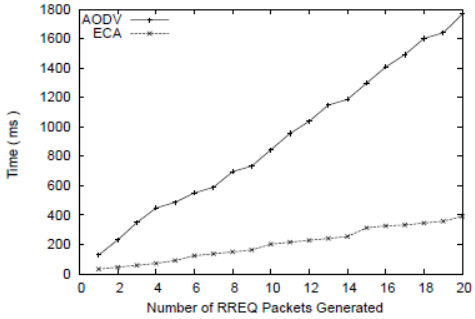


Fig. 8: Time Spent For Generation of RREQ

ECA model gives better performance in path building without any overheads.

Fig.8 analyzes how the delay is effected by increasing the size of RREQ load in local (n2n) connection establishment. Graph is plotted for Number of route requests generated from source node along the Xaxis against the time consumed to generate RREQ packets along the Y-axis. The size of the RREQ traffic is incremented in every consecutive data transmission cycle will help to

study the behavior of both algorithms, how increase in traffic load will affect the performance of the protocols.

Initially source node sends one RREQ and time required to receive positive ACK is calculated. In the next iteration, the RREQ traffic will be incremented by 1 ie, 2 RREQ's are sent from two different source nodes and time taken to receive positive ACK is measured. Both readings are added and this process is repeated for 20 simultaneous RREQ's sent from different source nodes. The experiment is repeated for AODV model where the source node will wait till suitable next hop node is determined, based on minimum hop count towards destination node without considering QoS requirements.

As the number of RREQ's generated at the source node increases, AODV slows down in its response, ranging from 100 msec to 1800 msec while ECA responds very fast to the increasing load, ranging from 15ms to 400ms, when the number of iterations in the simulations are considered.

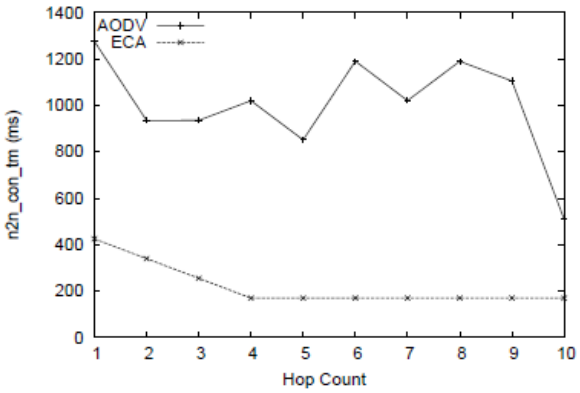


Fig. 9: Node to Node connectivity Time (n2n-con)

In the Fig.9, graph is plotted by taking hop-count from source towards destination node along X-axis,which is labeled as 1 for source(0) to 1st hop node, labeled as 2 for 1st hop node to 2nd hop node and so on till the destination(10) is reached, which is labeled as 10, against the time consumed in establishing connection along the Y-axis.

Node to Node connectivity time (n2n con tm) is given by the formula:

$$n2n\_con\_tm = (n-1) * (req\_tm + nack\_tm) + req\_tm + ack\_tm \quad (4)$$

where req_tm, is the time required by the route request packet to reach receiver, $nack\_tm$ is the time spent in generation and transmission of negative ACK, indicating that the node does not have sufficient resources to handle the current traffic. ack_tm is the time spent in generation and transmission of ACK packet. The node having sufficient resources are obtained in the search process so that the next hop node is determined.

At the source node, the RREQ packet needs to be created due to which the graph shows initial higher value. In the intermediate nodes, the time in establishing the connection remains almost constant for ECA, since it will select the node to be transmitted in random basis in the loop iteration. So in most of the cases, the convergence of connectivity occurs at the early stages of the loop. In case of AODV, which searches sequentially the neighboring nodes to select the next best hop node, which increases connectivity delay.

The Fig.10 shows path establishment time for various intermediate nodes. The graph is plotted for Number of intermediate nodes Vs Path establishment Time. These are the intermediate nodes the packet needs to cross in order to reach the destination. The path establishment time for source to destination is given by

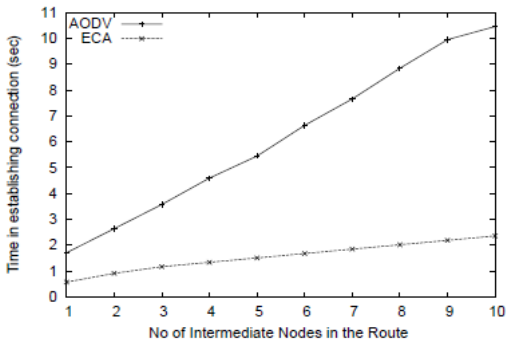


Fig. 10: End to End Path Establishment Time (s2d)

the formula, where we have 9 intermediate nodes.

$$route\_tm(s2d) = tm(n0, n1) + tm(n1, n2) + ... + tm(n9, n10) \quad (5)$$

where $route\_tm(s2d)$ is the time taken to establish the connectivity between source node $(n0)$ and destination node $(n10)$, while $n1$ ... $n9$ are intermediate nodes in the route. tm is the time taken by the RREQ to establish node to node connection.

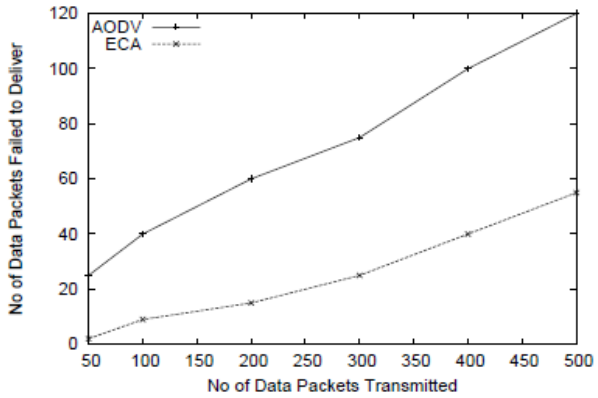


Fig. 11: Data packet drop in the network

The graph in Fig.11 is plotted for the number of data packets created and transmitted along X-axis against, the number of packets dropped due to node/route failure along the Y-axis. The size of the data traffic is increased after every consecutive data transmission cycle. The nodes along the route will consume energy and their levels are continuously dropped in the transmission and receiving process. Finally they will loose all their energy resulting in route/node failure. Also due to node movement, any two nodes in the route will move far away from each other and finally they will disconnect. In both of the above cases, corresponding route between source-destination pair is said to be broken, and packets will not be able to reach the destination node and data packets traveling in the path is said to be dropped.

Fig.12 is plotted by taking number of local route correction requestmessage along X-axis against the time taken to re-construct the path along the Y-axis. The local correction method is used to establish the route

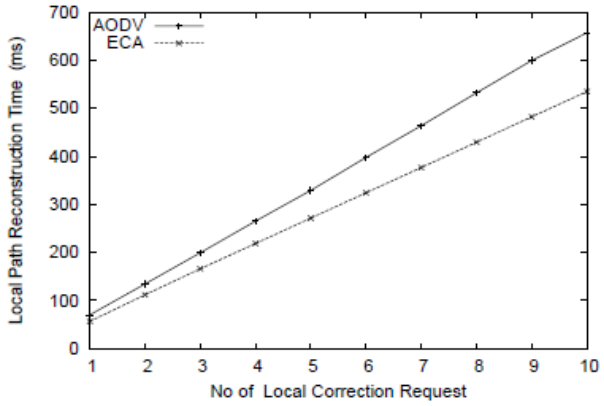


Fig. 12: Local Path Reconstruction/correction Time

when the broken link can be corrected within $s2d\_RTT$. This involves the time taken for broadcasting the RERR message to all the nodes within the radio range which will inform local node failure and will update their respective routing tables to reflect the change in the network topology, and searching for the best next hop node to re-establish end to end connection.

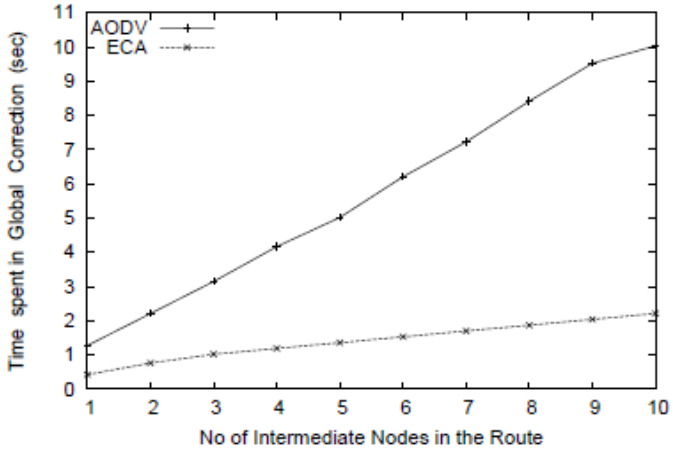


Fig. 13: Time taken in Global Route Error correction

Fig.13 is the analysis of global route reconstruction technique, which is invoked when the path cannot be corrected in $s2d\_RTT$. The graph is plotted by taking number of intermediate nodes in the route along the X-axis against the time taken to re-establish the path along the Y-axis. The task involves the time required to send Route Error message back to the source node from the damaged link/node followed by creating and sending new RREQ to the destination node and the process is represented by the equation,

$$gl\_route\_cor\_tm(s2d) = RERR\_tm(BR\_link, SN) + RREQ\_tm(SN, DN) \quad (6)$$

where BR_link denotes node where the link is broken.

The time consumed in the global route reconstruction process by the summation of RERR message propagation time ($RERR\_tm(BR\_link, SN)$) and the RREQ packet propagation time ($RREQ\_tm(SN, DN)$). In the graph, the readings are considered for single global route error correction requirement by varying the number of intermediate nodes between source node and destination node.

6 CONCLUSION

ECA QoS aware model checks the QoS status of the neighboring nodes before selecting the next node to forward the RREQ packet, which will guarantee the establishment of reliable route to the destination. The resulting route, increases the reliability of the data transfer since it satisfies QoS requirements, which constructs a more stable path as compared to original AODV, as it uses normal flooding of RREQ packets without considering any QoS requirements. The simulation results shows that the ECA model gives better results, while considering the local connectivity time, source to destination connectivity time, number of data packets successfully delivered to the destination, local and global error correction time, compared to AODV. In the future work, we will analyze, how the motion of the nodes, will effect the QoS parameters.We can also add proper buffer management strategies to improve the simulation accuracy and to compare memory utilization of the nodes. We would like to provide mathematical analysis for the ECA model.

7 APPENDIX

Event	Condition	Action
3. Route Response (RSP)		
RSP-E1:RREPLY_PKT_ready(DN)	RSP-C1: RREQ_received AND RSP-C2: resources_for_RREPLY();	RSP-A1: send_RREPLY();
RSP-E2: RREPLY_PKT_received(IN)	RSP-C2();	RSP-A2: process_RREPLY();, RSP-A3: SPNidRTC();, RSP-A4: send_RREPLY(PN);
4. Error Handling at Route Request - Response pair (ERR)		
ERR-E1: recv_RREQ_local_ACK (ND)	ERR-C1: time(RREQ,ACK) < LTO	ERR-A1: CRCE ();, ERR-A2: select_NN();
ERR-E2: recv_RREQ_local_NACK (ND)	ERR-C2: time(RREQ,NACK) < LTO	ERR-A3: retransmit(NN);
ERR-E3: local_RREQ_Timeout(ND)	ERR-C3: time(RREQ,RES) > LTO	ERR-A4: process_local_RREQ_timeout() { compute_NNid();, send_RREQ(NN); }
ERR-E4: No_RREPLY_At_ Timeout(SN)	ERR-C4: time > NTT	ERR-A5: process_RREPLY_timeout() { check_RReq_ReTx_attemps(); }
ERR-E5: RREQ_lost (ND)	ERR-C5: nack before timeout	ERR-A6: retransmit_local_RREQ(NN);
ERR-E6: RREQ_corrupted (ND)	ERR-C6: cal_CHKS(RREQ) ! = CHKS (header)	ERR-A5();
ERR-E7: RREPLY_corrupted (ND)	ERR-C6();	ERR-A5();
ERR-E8: RREPLY_lost(SN)	ERR-C7: RREPLY_timer_expired()	ERR-A8: send_RERRor();
ERR-E9: exceeds_MRRTXA(IN)		ERR-A9: report_ERR (source_APP, "DESTN UNREACHABLE");
ERR-E10: RT_entry_Timeout(ND)		ERR-A10: invalidate_RT_entry();
ERR-E11: ERT(IN)	ERR-C8: S2D_RTT > repair_time	ERR-A11: bc_RERR_local_repair();
	!ERR-C8();	ERR-A12: send_RERR(SN) ;
ERR-E12: recvd_RERR(SN)		ERR-A13: ReTx_RReq();
ERR-E13: recvd_RERR_BC(IN)		ERR-A14: local_route_correction();
5. Data Transmission - send and recv (DT)		
DT-E1: data_PKT_ready (SN)		DT-A1: schedule_transmit();, DT-A2: check_out_channel_status();, DT-A3: prepare_RTS();
DT-E2: RTS_PKT_Ready (IN)		DT-A4: send_RTS(NN);
DT-E3: RTS_recvd(IN)		DT-A5: check_resources();
DT-E4: resources_available(IN)	DT-C1: node.energy > 0	DT-A6: send_CTS(PN);

DT-E5: CTS_recvd(IN)		DT-A7: send_data_PKT(NN)
DT-E6: RTS_timeout(IN)	DT-C2: recvd(CTS)	DT-A8: prepare_For_NS();
	!DT-C2	DT-A9: check_RTS_count();
DT-E7: data_recvd(IN)	DT-C3: PKT.DN == DN !DT-C3	DT-A10: process_recvd_data(DN); DT-A11: send_data(NN) ;
6. Error Handling for Data Transmission (EDT)		
EDT-E1: RCME(IN)		EDT-A1: prepare_RERR(); EDT-A2: LBC_RERR(); EDT-A3: rebuild_new_Route(); EDT-A4: update_RT_cache_entry();
EDT-E2: RCEND(IN)		EDT-A5:send_RCME_ERR(SN);
EDT-E2: RCME_ERR_ Report_Recvd(SN)		EDT-A6: prepare_RREQ(SN); EDT-A7: construct_New_Route(SN);
7. Route Maintenance (RMN)		
RMN-E1: EBTH(RN)	RMN-C1: recv[node].energy < 0 AND RMN-C2: S2D_RTT > Repair_Time	RMN-A1: local_route_correction(); { RMN-A2: LBC_RERR(); }
	RMN-C1 And !RMN-C2	RMN-A2();, RMN-A3: send_RERR(SN);, RMN-A4: global_route_ correction(SD,DN);
RMN-E2: RERR_recvd(SN)		RMN-A5 : initiate_RREQ();
RMN-E3: relocate(SN)	RMN-C3: dis(prev_posn,cur_posn) > radio_range	 RMN-A5();
RMN-E4: relocate(IN)	RMN-C2 AND RMN-C3	RMN-A1();
	!RMN-C2 AND RMN-C3	RMN-A2();, RMN-A3();, RMN-A4();
RMN-E5: RCRRT(IN)	RMN-C2()	RMN-A6: BC_RERR_local_repair();
	!RMN-C2	RMN-A7: send_RERR(SN);
RMN-E6: recv_RERR(SN)		RMN-A8: RTX_RREQ();
RMN-E7: recv_RERR_BC(IN)		RMN-A1();

Table 3: ECA AODV Description for remaining phases

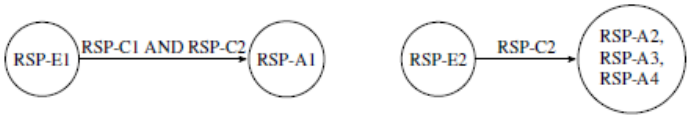


Fig. 14: ECA Sequence For Route Response

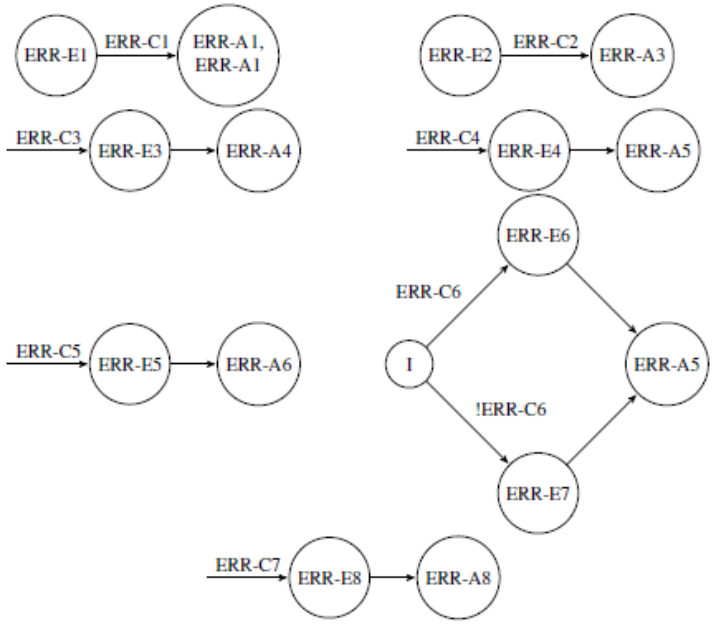


Fig. 15: ECA Sequence For Error Handling in Route Request - Response Pair

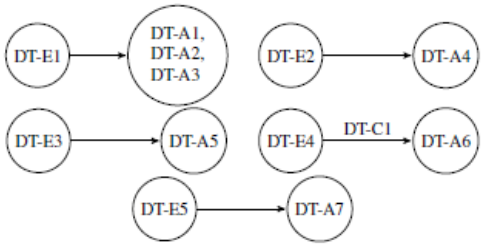


Fig. 16: ECA Sequence For Data Transfer

REFERENCES

[1] Aarti Bairagi, Shweta Yadav, "A Proposed Route Selection Method in AODV Routing Protocol for MANET", International Journal of Computer Technology and Electronics Engi neering (IJCTEE) Volume 3, Issue 2, April 2013

[2] D. Deepthi Veronica, D.B.Jagannadha Rao , "Performance Analysis of AODV and DSR in MANETS Using NS2 Simulation", International Journal of Innovative Research in Computer and Communication
Engineering Vol. 1, Issue 4, June 2013

[3] Pushpavalli M, Dr.A.M.Natarajan , Annitha N, " A QUALITY OF SERVICE BASED AODV WITH QoS-AWARE ROUTING ALGORITHMS FOR MANETs", International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 2, Issue 2, February 2013

[4] Perkins, et. al. , "Ad hoc On-Demand Distance Vector (AODV) Routing - Experimental AODV Routing", RFC 3561, July 2003, http://www.rfc-base.org/txt/rfc-3561.txt

[5] G. Ilanchezhiapandian, P. Sheik Abdul Khade, "Quality of Service (QoS) Routing in Mobile Ad-hoc Network (MANET) using AODV protocol: Cross-Layer Approach", IJCA Proceedings on National Conference on Future Computing 2013, 2013 by IJCA Journal , NCFC - Number 1 , Year of Publication: 2013

[6] Pratik Gite, Manish Sharma, "Performance Evaluation of ad-hoc Network Routing Protocols using ns2 Simulation", ACEEE Int. J. on Network Security , Vol. 03, No. 01, Jan 2012

[7] Dr. K.D.Kulat V.K.Taksande, "An extensive simulation analysis of AODV protocol for chain topology in MANET", 2011, International Journal of Internet Computing Journal, Volume 1, Issue 1, Publisher-
Interscience

[8] Maamar Sedrati , Azeddine Bilami and Mohamed Benmohamed, "M-AODV: AODV variant to Improve Quality of Service in MANETs", IJCSI International Journal of Computer Science Issues, Vol.
8, Issue 1, January 2011, ISSN (Online): 1694-0814, www.IJCSI.org

[9] Nisarg Gandhewar, Rahila Patel, "Performance Evaluation of AODV protocol in MANET using NS2 Simulator", 2nd National Conference on Information and Communication Technology (NCICT) 2011,
Proceedings published in International Journal of Computer Applications (IJCA)

[10] Bouchama, Nadir ; Nouali-Taboudjemat, Nadia ; Assani, Djamil ; Djellab, Natalia ; Maouchi, Houari, "Extending the AODV Protocol to Provide Quality of Service in Mobile Ad Hoc Networks" , 1st International Conference on Information Systems and Technologies (ICIST-2011)

[11] Atef Abdrabou andWeihua Zhuang, "Statistical QoS Routing for IEEE 802.11Multihop Ad Hoc Networks", IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, VOL. 8, NO. 3, MARCH
2009

[12] Chunhung Richard Lin and Jain-Shing Liu , "QoS Routing in Ad Hoc Wireless Networks", IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 17, NO. 8, AUGUST 1999

[13] Ali khosrozadeh abolfazl akbari, Maryam Bagheri and Neda Beikmahdavi, "A new algorithm aodv routing protocol in mobile adhoc networks", vol 2 no 3 september 2011, Int. J Latest Trends Computing.

[14] Chandreyee Chowdhury, Sarmistha Neogy, "Reliability Estimate of Mobile Agent System for QoS MANET Applications", 2011 , IEEE

[15] Dr Chandra Shekar Reddy Putta , Dr K.Bhanu Prasad , Dilli Ravilla, Murali Nath R.S ,M.L.Ravi Chandra , Performance of Ad hoc Network Routing Protocols in IEEE 802.11", International Conf. on Computer & Communication Technology,

[16] A K Daniel, R Singh, Zubair Khan, "Position Based Multicast Routing Protocol for AD-hocWireless Network Using Backpressure Restoration" , 2010 IEEE , Volume 2, 2010, 2nd International Conference
on Computer Engineering and Technology V2-459

[17] Kannan .T , M.Pushpavalli , Dr.A.M.Natarajan , "Fortification of QoS Routing in MANETs using Proactive Protocol", zcto IEEE , ICWCSC 20lOX

[18] J.Premalatha, P.Balasubramanie, "Enhancing Quality of Service in MANETS by Effective Routing", Wireless Communication and Sensor Computing, ICWCSC-2010, International Conference Chennai , 2-4 Jan, 2010

[19] Javad Vazifehdan, R. Venkatesha Prasad, and Ignas Niemegeers, "Energy-Efficient Reliable Routing Considering Residual Energy inWireless Ad Hoc Networks" , IEEE TRANSACTIONS ONMOBILE COMPUTING, VOL. 13, NO. 2, FEBRUARY 2014

[20] Chetan Shiva Shankar, Anand Ranganathan and Roy Campbell , "An ECA-P policy-based framework for managing ubiquitous computing environments", Proceedings of the Second Annual International Conference onMobile and Ubiquitous Sy stems: Networking and Services (MobiQuitous-05) 0-7695-2375-7/05 2005 IEEE

[21] T. Nakagawa, C. Doi, K. Ohta, and H. Inamura, "Customizable Context Detection for ECA rule-based Context-aware Applications" 2012 by Information Processing Society of Japan ICMU 2012

[22] Thomas Heimrich and Gnther Specht " Enhancing ECA Rules for Distributed Active Database Systems A.B. Chaudhri et al. (Eds.): Web Databases and Web Services 2002, LNCS 2593, pp. 199205,
2003. Springer-Verlag Berlin Heidelberg 2003

[23] James Bailey , Alexandra Poulovassilis , Peter T. Wood , "An Event-Condition-Action Language for XML" , WWW-2002, May 7-11, 2001, Honolulu, Hawaii, USA., ACM 1-58113-449-5/02/0005

[24] Integrated Manufacturing Systems, Volume 5, Issue 4 (2006-09-19)

[25] Umeshwara Dayal, "Active Data Base Management System" , p-150, Proceedings of the Third International Conference on Data and Knowledge Basis, Jerusalem Issue, June 28-30, 1988

AUTHORS

Raghavendra M is working as Assistant Professor in the of Computer Science and Engineering at Siddaganga Institute of Technology, Tumkur, India. He received his Bachelors degree in Computer Science-Engineering from HMSIT, Tumkur, India and Master of Engineering in Computer Science-Engineering from University Visvesvaraya College of Engineering (UVCE), Bangalore University, Bangalore, India. He has more than 10 years of teaching experience. His research interest is in the area ofWireless Sensor Networks and Adhoc networks.

Pallapa Venkataram received his Ph.D. Degree in Information Sciences from the University of Sheffield, England, in 1986. He is currently Professorin the Department of ECE, IISc, Bangalore, India. Dr. Pallapas research interests are in the areas of Wireless Ubiquitous Networks, Communication Protocols, Computation Intelligence applications in Communication Networks and Multimedia Systems. Dr. Pallapa is the holder of a Distinguished Visitor Diploma from the Orrego University, Trujillo, PERU. He has published over 200 papers in International/national Journals/conferences. He has received best paper awards at GLOBECOM93 and INM95 and also CDIL (Communication Devices India Ltd) for a paper published in IETE Journal. He is a Fellow of IEE (England), Fellow of IETE(India) and a Senior member of IEEE Computer Society.