Formal Development of Path Discovery in AODV Routing Protocol using Event-B

Arun Kumar Singh
Department of Electronics Engg.
IET, Lucknow, India

Divakar Yadav
Department of Computer Science & Engg.
IET, Lucknow, India

Vinod Kumar Singh
Department of Electronics Engg.
IET, Lucknow, India

ABSTRACT
An ad hoc network is collection of mobile nodes that do not have any fixed topology. In such a network, nodes are likely to join or leave the network in an arbitrary manner. Joining of any nodes is announced by itself to other neighboring nodes and nodes already present in the network learns about joining of the new node by this announcement. In such a scenario, discovery of path from one node to other node is an important task for the routing protocols used for the purpose. Adhoc on demand vector (AODV) is one routing protocol used in MANET environment. Being a reactive protocol, path discovery process in AODV is initiated by sender node when there is no routing information for an intended destination.

In this paper, we present a formal model for path discovery process of the AODV protocol using Event-B. The model have been developed and checked using the RODIN tool which provides an integrated framework for development of Event-B models. Event-B technique uses a notion of refinement to specify the mathematical models of distributed systems in an incremental manner. The specifications of the system have been checked for consistency and satisfy the behavioural properties of the system expressed as invariants. All the proof obligations were discharged automatically by the RODIN tool.

Keywords
MANET, AODV, Formal Method, Event-B.

1. INTRODUCTION
A mobile ad hoc network (MANET) may be defined as a collection of autonomous mobile hosts which are free to move arbitrarily. Every node is independent and may join or leave frequently the periphery of the existing network of cooperative and coordinating nodes. Thus a MANET environment may be visualized as a complex distributed environment where the number of nodes may vary and their topology keep on changing frequently during normal operation. Routing of messages in such an environment becomes difficult compared to the stationary environment, as the routing protocol to be used, need to be adaptive to changing topology of the environment.

In an ad hoc network, there is no way that the location of a host can be precisely determined. There is no centralizing server to administer the network. So the problem of communication, in a small area where an ad hoc network is set up, is still manageable if the movement of nodes is not very fast. In a situation like this, every node could be preloaded with route table initially and to routing table may keep on updating in an incremental manner whenever any node in the network broadcast route update information. If the network is very dynamic and spread widely, then a different approach is needed. In fact, in a decentralized environment where network topology keep on changing, establishment of a route between two nodes can be visualized as path finding in any dynamic graph.

Routing protocols proposed for MANET are categorized by the routing strategy followed by them. First, there are protocols that are *distance vector* typed. The typical distance vector algorithms such as Distributed Bellman Ford algorithm [1, 2], cannot be adapted for mobile networks because of their slow rate of convergence and *count-to-infinity* problem. Protocols such as Wireless Routing Protocol (WRP) [3], Destination Sequence Distance Vector (DSDV) routing protocol [4], are extensions of distance vector algorithm. Second, there are protocols that are based on link state [5] algorithms such as Global State Routing (GSR) [6], Fisheye State Routing (FSR) [7], Source Tree Adaptive Routing (STAR) [8], Optimized Link State Routing (OLSR) protocol [9], and Landmark Ad Hoc Routing (LANMAR) [10]. An important class of algorithms, termed as *on-demand* routing protocols [11, 12], are proposed exclusively for ad hoc networks only. On-demand routing protocols do not maintain route table of the network on a continual basis, rather, routes are established whenever there is a demand by the source. When a route is needed by the source, it floods a route request packet to construct a route. Upon receiving route requests, the destination selects the best route based on route selection algorithm. Lightweight Mobile Routing (LMR) [13], Dynamic Source Routing (DSR) [14, 15], Temporarily Ordered Routing Algorithm (TORA) [16], Ad-Hoc on demand distance vector (AODV) routing [17] are typical on-demand routing protocols.

Network protocols have been analysed using number of formal methods. Some of the analyses have been performed using model checking [18, 19]. The protocols have also been analysed using theorem prover [20] and technique of refinements [21, 22, 23, 24, 25]. An extensive study for routing in changing environment has been reported in [26].

In this paper, we attempt to delineate a path discovery model in MANET of AODV protocol applying Event-B. The model contains a *Send* event that models the sending of route request message. The route request messages from one node to other node is forwarded using event *Forward.* The events *Rev_Route_Entry*, *Rrep_to_Rreq*, *Rrep-Backward*, *Receive* models the event of information of traversed node, sending of reply message in response to route request message, passing of route reply message in backward direction and delivery of message at destination or target site respectively.

The rest of the paper is organized as follows. Section 2 describes framework for formal modeling using Event-B and RODIN. The informal description and assumptions for modeling is given in section 3. Section 4 presents formal model of path discovery of AODV followed by conclusion of the paper in section 5.

2. FORMAL MODELING USING EVENT-B AND RODIN PLATEFORM

Event-B [21, 23, 26] is a method for discrete-level modeling and analysis which is derived from the B-Method [27] by incorporating the ideas of action systems [28]. It is based on set theory and logic as modeling notations, refinement to represent systems at different abstraction levels and proof obligations to verify consistency between refinement levels. The purpose of the proof obligations is to show that a model is rigorous with respect to some behavioural semantics and to verify properties of the model. This purpose permits to use the same proof obligations for different modeling domains such as concurrent, reactive, and distributed systems. For developing mathematical models, set theory is used as modeling notation. These models consist of several components which are either a machine or a context.

The behavioural properties of an Event-B model [29, 30] are described by Machine which is dynamic and includes variables, invariants, theorems, and events. Variables refer to mathematical objects like functions, sets, binary relations, and numbers. These variables are constrained by invariants which are supposed to be fixed whenever value of variable changes. However, this must be proved through the discharge of proof obligation. The theorem of machine is used within the context and invariants. A machine also contains a number of events which have three components: an event name, guard(s) and action(s). The guard is necessary condition for the performance of event. The actions regulate the way through which the state variables are evolving during event. Moreover, machines are refined by other machines with the limitation that each machine can refine maximum one machine.

The static structure of an Event-B model is designated by Context which includes constants, axioms, carrier sets and theorems which are further used to describe the properties of sets and constants. Constants enlist the different constants introduced in the context, while axioms defines the main properties of the constants and any axiom can be marked as theorem if it is derived from previously declared axioms. The Figure1 given below explains the relationship between various components of a model.

The static structure of an Event-B model is designated by Context which includes constants, axioms, carrier sets and theorems which are further used to describe the properties of sets and constants. Constants enlist the different constants introduced in the context, while axioms defines the main properties of the constants and any axiom can be marked as theorem if it is derived from previously declared axioms. There also exists a relationship between components of model as shown in Figure 1.

The Rodin tool provides an extensible and configurable platform for specifying and refining Event-B machines and offers a seamless integration between modeling and proof obligations.

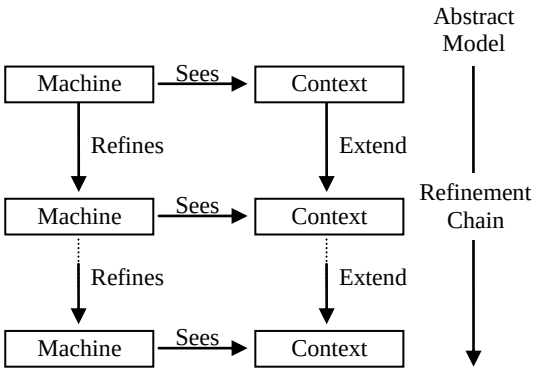


Fig 1: Relationship between components of model

3. INFORMAL DESCRIPTION OF AODV AND MODELING ASSUMPTIONS

3.1 Informal Description of AODV

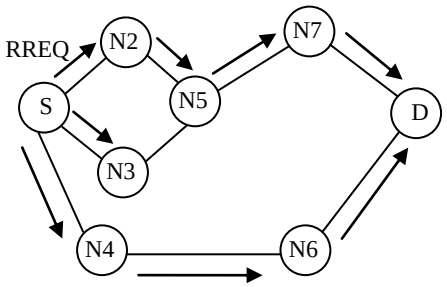
The AODV is a source initiated reactive routing protocol. It is adaptive to dynamic link conditions and determines unicast between sources and destinations. It uses destination sequence numbers in order to ensure that paths are free from loops. AODV do not suffer from counting to infinity problems associated with classical distance vector protocols. This is an on demand protocol which has the same functionality as in DSR during route discovery phase. In contrast to DSR, it is not a source based routing scheme. Rather, every hop of a route in AODV maintains the next hop information by its own. It employs the destination sequence number technique used in DSDV. Thus, AODV may be visualized as an on demand version of DSDV because it typically minimizes the number of required broadcasts by creating routes on an on-demand basis. In AODV, nodes that are not on a selected path do not maintain routing information or participate in routing table exchanges.

Path Discovery

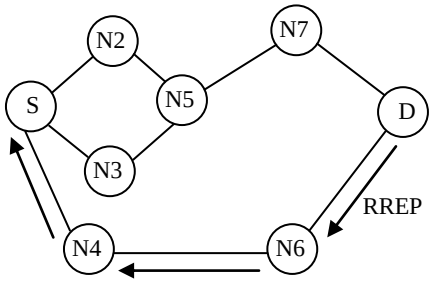
The process of path discovery starts initially by sending a *Hello* message by all nodes on its interface and receive *Hello* messages from its neighbors. This process is repeated periodically to determine connectivity to the neighbours. When a source node desires to send a message to some destination node, it checks for the valid route to that destination. If the same is not found, it initiates a *Path Discovery* process to locate the other node. For this purpose, it broadcasts a route request (RREQ) packet to its neighbors, which forward the request to their neighbors, and so on. The process is repeated until either the destination or an intermediate node with a relatively fresh route to the destination is located.

Figure 2 illustrates the path discovery process of AODV. AODV utilizes destination sequence numbers to ensure all routes are loop-free and contain the most recent route information. Each node maintains its own sequence number. The RREQ packet of the source node contains the most recent sequence number of the destination and its own sequence number. Intermediate nodes can reply to the RREQ request only if they have a route to the destination whose corresponding destination sequence number is greater than or equal to that contained in the RREQ. During the process of forwarding the RREQ, intermediate nodes record the address of the neighbor from which the first copy of the broadcast

packet is received. This helps in establishing a reverse path. If copies of the same RREQ are received later, the packets are discarded. Once the RREQ reaches the destination or an intermediate node with a fresh enough route, the destination intermediate node responds by a route reply (RREP) packet to the neighbor from where it received the RREQ first.



(a) Route Request (RREQ)



(b) Route Reply (RREP)

Fig 2: Path discovery in AODV

As the RREP is routed along the reverse path, nodes appearing in this path update forward route entries in their route tables. This entry points to the node from which the RREP was received. These forward route entries signify the active forward routes. When *Route Reply Message* reaches the source node of RREQ, the route is ready and is fresh. If any link on the forward path is broken, the intermediate node adjacent to the broken link sends new *Route Reply Message* to all the sources to inform them of the link failure using the *forward path.* Consequently, other nodes will send to their neighbors until all nodes that use *forward path* are informed. The source nodes can then initiate new RREQ procedures if route is desired to the destination.

3.2 Modeling Assumptions

In MANET environment the status of links changes arbitrarily, therefore, sometimes it is not possible that each node have the correct information of network topology and status of links. Also in real time the environments behavior may not be represented correctly by the protocol in use [30]. To handle the situation, we consider the limiting case, when the environment is reasonably stable in terms of topology and anticipate that the information maintained in local routing table will become consistent with states of the actual topology at the time of conclusion.

The major requirements of the system can be outlined as follows

-Message transition from source node to destination node must be successful in MANET.

-Every node has the correct status of the links between all nodes in the network assuming the topology stable for a reasonably long time.

Before developing the formal model of path discovery of AODV, we have following assumption on MANET environment:

- There are only finite no of nodes.

- There are directed links between some pairs of distinct nodes. Links may become active or inactive at any point of time.

- Nodes are communicating by broadcasting when they are directly connected. A new node announces its presence and listens for announcements broadcast by its neighbours.

-Messages are transmitted asynchronously.

- When any messages sent on a link which is not received are treated as lost and link is treated as inactive.

In following section, we present the formal mmodel of the system using Event-B.

4. SYSTEM MODEL

We start with our abstract model of DSDV routing protocol presented in [24]. The context part contains NODE and MSG as carrier sets which represents nodes and messages respectively. The variables *sent*, *got* and *lost* are defined as a subset of *MSG*. The variable *ALinks* defined as: *ALinks* ∈ NODE↔NODE represents the set of active links between nodes. The variables and invariants are given in figure 3.

MACHINE
RoutingM1
VARIABLES
chstore, dseq, dseqm, sseqm, nodemetric, neighbour, nodemetricm, rreq, receivedreq, messagetype, travel_node, pathnode, pathsize, travel_node, chback
INVARIANTS
inv1 : $chstore \in NODE \leftrightarrow MSG$
inv2 : $dseq \in (NODE \leftrightarrow NODE) \rightarrow \mathbb{N}$
inv3 : $dseqm \in (MSG \leftrightarrow NODE) \rightarrow \mathbb{N}$
inv4 : $sseqm \in (MSG \leftrightarrow NODE) \rightarrow \mathbb{N}$
inv5 : $nodemetric \in NODE \rightarrow (NODE \rightarrow \mathbb{N})$
inv6 : $neighbour \in NODE \leftrightarrow NODE$
inv7 : $nodemetricm \in MSG \rightarrow (NODE \rightarrow \mathbb{N})$
inv8 : $rreq \in NODE \rightarrow MSG$
inv9 : $messagetype \in MSG \rightarrow MSGTYPE$
inv10 : $travel\_node \in TNODE$
inv11 : $pathnode \subseteq NODE$
inv12 : $pathsize \in \mathbb{N}$
inv13 : $travel\_node \in 1 \cdot \cdot pathsize \rightarrowtail\!\!\!\!\!\rightarrow pathnode$
inv14 : $chback \in NODE \leftrightarrow MSG$

Fig. 3. Variables and Invariants of Machine

Other important variables are detailed below.

(i) The variable chstore is a relation which contains messages which are in communication channel. A mapping $n \mapsto m : chstore$ represents that message m is sent by node n and it is present in channel.

(ii) Every site maintains the information about sequence number of destination site. The variable *dseq* represents sequence number of destination site. It is represented as $dseq \in NODE \rightarrow (NODE \rightarrow \mathbb{N})$.A mapping $\{s \mapsto (\{t \mapsto n1\})\} \in dseq$ indicates that routing table of site *s* has entry that *n1* is the sequence number of destination site t.

(iii) The variable sseqm specifies sequence number of source assigned to source field of message. The domain of this variable is a relation between message set *MSG* and node set *NODE*. A mapping of form $(\{datamsg \mapsto s\}) \mapsto n1 \in sseqm$ indicates that *n1* is sequence number assigned to source node *s* of message *datamsg*. Similarly, variable *dseqm* specifies sequence number of destination assigned to destination field of message.

(iv) The variable *neighbour* gives the information about neighbour of any node.

(v) The variable *nodemetric* contains the information about the total hop count from a particular node.

(vi) The variable rreq presents route request message send by any site. A mapping $x \mapsto mm \in rreq$ indicates that node x has sent route request message mm.

(vii) The variable *travel_node* is declared as:

$$travel\_node \in 1 \cdots pathsize \rightarrow\!\!\!\!\!\rightarrow pathnode$$

This variable sequentially maintains entry of each traversed node. It is defined as a total bijective function. The total bijective function is a one-to-one and onto relation which maps all elements of the domain. Each time when a route request message passes from one node to other this variable makes the entry of traversed node sequentially

(viii)The variable *chback* presents the information about node from which route reply message is received.

4.1 Send Event

This event models the sending of route request message (fig. 4). The *s* and *t* are two different nodes are specified through the guard *grd1*, *grd2* and *grd3*. The message *datamsg* has not been sent is specified through guard *grd5*. The node *s* and node *t* are the source and destination or target of message *datamsg* is specified through guard *grd6* and *grd7*. The guard *grd8* specifies that type of *datamsg* is *RREQ* i.e; route request message. The guard *grd9* ensures that message *datamsg* has not been sent. This event adds the *datamsg* in to channel *(act2).* The action *act3* assigns the sequence number of source *s* to the source field of message *datamsg*. Similarly, action *act4* assigns the sequence number of destination *d* to the destination field of message *datamsg.* The action *act5* assigns *nodemetric* of source *s* to message *datamsg*. The action *act6* add the request message *datamsg* in to request set *rreq*. The action *act7* add the source node *s* as starting node because node information is added at 0^{th} position.

4.2 Forward Event

This event forwards the route request message from one node to other node (fig. 5). The guard *grd6* ensures that message *datamsg* is a route request message. The message has been sent but it is neither received nor lost is ensured through guard *grd1*. The guard *grd3* ensures that channel contains the route request message sent by site *x*. The guard *grd2* ensures that node *x* and node *y* are connected with each other. The guard *grd4* specifies that *y* is a neighbour of node *x*. The guard *grd8* ensures that route request message has travelled the node *x*. This event forwards route request message from node *x* to node *y*. The action *act1* add route request message in *rreq* set. The action *act2* formalize the forwarding of route request message. It adds the entry for node *y* and remove entry of node *x*.

Send ≙
ANY *s, t, datamsg*
WHERE
grd1 : $s \in NODE$
grd2 : $t \in NODE$
grd3 : $s \neq t$
grd4 : $datamsg \in MSG$
grd5 : $datamsg \notin sent$
grd6 : $source(datamsg) = s$
grd7 : $target(datamsg) = t$
grd8 : $messagetype(datamsg) = RREQ$
grd9 : $s \mapsto datamsg \notin chstore$
THEN
act1 : $sent := sent \cup \{datamsg\}$
act2 : $chstore := chstore \cup \{s \mapsto datamsg\}$
act3 : $sseqm(\{datamsg \mapsto s\}) := dseq(\{s \mapsto s\})$
act4 : $dseqm(\{datamsg \mapsto t\}) := dseq(\{s \mapsto t\})$
act5 : $nodemetricm(datamsg) := nodemetric(s)$
act6 : $rreq := rreq \cup \{s \mapsto datamsg\}$
act7 : $travel\_node(pathsize+1) := s$
END

Fig. 4. Send Event

Forward ≙
ANY datamsg, x,y
WHERE
grd1 : $datamsg \in sent \setminus (got \cup lost)$
grd2 : $x \mapsto y \in ALinks$
grd3 : $x \mapsto datamsg \in chstore$
grd4 : $y \in neighbour[\{x\}]$
grd5 : $y \mapsto datamsg \notin chstore$
grd6 : $messagetype(datamsg) = RREQ$
grd7 : $datamsg \in ran(rreq)$
grd8 : $x \in ran(travel\_node)$
THEN
act1 : $rreq := rreq \cup \{y \mapsto datamsg\}$
act2 : $chstore := (chstore \setminus \{x \mapsto datamsg\}) \cup \{y \mapsto datamsg\}$
END

Fig. 5. Forward Event

4.3 Capturing Traversed Node Information (Rev_Route_Entry Event)

This event formalizes the information of traversed node (fig. 6). The message *datamsg* is route request message is ensured through guard *grd1*. The guard *grd4* ensures that route request message *datamsg* is currently at node *x*. The target of route request message is node *t*. The guard *grd7* ensures that message is updated because destination sequence number in route request message *datamsg* is greater than sequence

number of destination at node *x*. The action *act1* makes the entry of traversed node node *x*.

Rev_Route_Entry ≙
ANY *datamsg, x, t*
WHERE
grd1 : $messagetype(datamsg)=RREQ$
grd2 : $x \in NODE$
grd3 : $t \in NODE$
grd4 : $x \mapsto datamsg \in chstore$
grd5 : $x \mapsto datamsg \in rreq$
grd6 : $target(datamsg) = t$
grd7 : $dseqm(\{datamsg \mapsto t\}) > dseq(\{x \mapsto t\})$
THEN
act1 : $travel\_node(pathsize+1) := x$
act2 : $pathsize := pathsize+1$
END

Fig. 6. Rev_Route_Entry Event

Rrep_to_Rreq ≙
ANY *t, datamsg1, y, datamsg2, n,s*
WHERE
grd1 : $n = pathsize$
grd2 : $datamsg1 \in MSG$
grd3 : $messagetype(datamsg1)=RREQ$
grd4 : $datamsg2 \in MSG$
grd5 : $messagetype(datamsg2)=RREP$
grd6 : $target(datamsg1) = t$
grd7 : $source(datamsg1)=s$
grd8 : $y \mapsto datamsg1 \in chstore$
grd9 : $(n \mapsto y) \in travel\_node$
grd10 : $dseqm(\{datamsg1 \mapsto t\}) < dseq(\{y \mapsto t\})$
grd11: $target(datamsg2)=s$
THEN
act1 : $sent := sent \cup \{datamsg2\}$
act2 : $chback := chback \cup \{y \mapsto datamsg2\}$
END

Fig. 7. Rrep_to_Rreq Event

4.4 Sending of Reply Message (Rrep_to_Rreq Event)

This event models the sending of reply message in response to route request message (fig. 7). The message *datamsg1* is route request message is ensured through *grd2* and *grd3*. The message *datamsg2* is route reply message is ensured through *grd4* and *grd5*. The target and source of route request message *datamsg1* are node *t* and node *s* respectively (*grd6* and *grd7)*. The guard *grd8* ensures that route request message *datamsg1* is currently present at node *y*. The guard *grd9* specifies that node *y* is traversed node. The guard *grd10* ensures that node *y* has more updated route information since sequence number of destination is greater than destination sequence number present in route request message *datamsg1*. This event triggers the sending of route reply message *datamsg2* to the source of route request message *datamsg1*. The action *act2* adds the information that *y* has sent route reply message *datamsg2*.

Rrep-Backward ≙
ANY *x, y, datamsg*
WHERE
grd1 : $y=travel\_node(pathsize)$
grd2 : $y \mapsto datamsg \in chback$
grd3 : $x=travel\_node(pathsize-1)$
grd4 : $messagetype(datamsg)=RREP$
grd5 : $datamsg \in sent \setminus (got \cup lost)$
grd6 : $y \mapsto x \in ALinks$
THEN
act1 : $chback := (chback \setminus \{y \mapsto datamsg\}) \cup \{x \mapsto datamsg\}$
act2 : $pathsize := pathsize-1$
END

Fig. 8. Rrep_Backward Event

4.5 Passing Reply Message in Backward Direction (Rrep_Backward Event)

This event models the passing of route reply message in backward direction (fig. 8). The guard *grd1* and *grd2* specifies that route reply message is currently present at node *y*. The *grd3* ensures that next node to which route reply message to be forwarded is node *x*. specifically, node *x* is that node from where node *y* has received route request message. The guard *grd4* ensures that type of message *datamsg* is route reply message. The message has already been sent is ensured through guard *grd5*. The guard *grd6* specifies that node *y* and node *x* are connected node. The action *act1* forwarded route reply message from node *y* to node *x*. The action *act2* reduces path size by one.

Receive ≙
ANY *s, t, datamsg*
WHERE
grd1 : $s \in NODE$
grd2 : $t \in NODE$
grd3 : $datamsg \in MSG$
grd4 : $source(datamsg) = s$
grd5 : $target(datamsg) = t$
grd6 : $datamsg \in sent \setminus (got \cup lost)$
grd7 : $t \mapsto datamsg \in chstore \vee t \mapsto datamsg \in chback$
THEN
act1 : $got := got \cup \{datamsg\}$
END

Fig. 9. Receive Event

4.6 Receive Event

This event models the delivery of message at destination or target site (fig. 9). The guard *grd4* and *grd5* specify that source and target of message datamsg are node *s* and node *t* respectively. The guard *grd6* ensures that message has been sent but not received by the target node. The guard *grd7* specifies that message is forwarded to target node *t*. The delivery of message may be done either in forward direction for route request message or in backward direction for route reply message.The action *act1* specifies that receiving of message.

4.7 Critical Invariant Of Model

During route discovery process, when any intermediate node receives a route request message it sends the RREP provided it has recent information about destination node. The following invariant tells that a route reply message *datamsg2* will present in channel *(y↦ datamsg2)∈chback)* when an intermediate node *y* has fresh route information i.e; (sequence number of destination node present in the route request message *datamsg1* is less than sequence number of destination node present in intermediate node *y* i.e; *dseqm({datamsg1↦t})< dseq({y↦t})*).

∀datamsg1,datamsg2,t,y·(t∈NODE∧ y∈NODE ∧ messagetype (datamsg1)=RREQ∧ target(datamsg1)=t ∧ messagetype (datamsg1) =RREP ∧ (y↦ datamsg2)∈ chback ⇒ dseqm({datamsg1↦t})< dseq({y↦t}))

5. CONCLUSION

The Path discovery process is crucial element of any routing protocol in MANET environment. Path discovery in AODV routing protocol is based on a route request/route reply query messages. It is initiated to locate other nodes in the network when a message has to be sent to some destination node and there do not exist any valid path to that destination. The process starts by broadcasting RREQ packet to its neighbours, the neighbours receiving this packet forward the request to their neighbours. This process is repeated till a fresh route to the destination is located with the help of RREP received from the intermediate nodes. In this work, formal specifications of Path discovery process of AODV routing protocol have been developed using Event-B. RODIN platform has been used for writing the Event-B specifications and discharging proof obligations generated by system. The behavioural properties of the system have been expressed in terms of number of Invariants. The specifications have been model checked for consistency. The specifications ensure that any source node initiating the path discovery will eventually have the fresh route information and will get to know about the route to other node, if it exists. The specifications satisfy all the invariants signifying that correctness of the system in terms of the behavioural properties of the system.

Total 71 proof obligations were generated by system out of which 66 were discharged automatically while 5 of them were proved interactively.

6. REFERENCES

[1] Shree Murthy, J.J. Garcia-Luna-Aveces, *Distributed Bellman-Ford routing protocol (DBF), a routing protocol for packet radio networks,* In Proc. ACM International Conference on Mobile Computing and Networking, pp. 86-95, Nov, 1995.

[2] M. S. Corson and A. Ephremides, *A distributed routing algorithm for mobile wireless networks,* In ACM Journal of Wireless Networks, vol. 1, no. 1, pp. 61-81, 1995.

[3] S. Murthy and J.J. Garcia-Luna-Aceves, *An Efficient Routing Protocol for Wireless Networks,* In ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks, pp. 183-97 Oct 1996.

[4] C. Perkins and P. Bhagwat, *Highly dynamic destination-sequenced distance- vector routing (DSDV) for mobile computers,* In ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications, London, UK,pp. 234-244.Aug 1994.

[5] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani. and R. D. Gitlin. *AIRMAIL: A link-layer protocol for wireless networks.* In Wireless Networks, 1(1) pp. 47-60,Feb 1995.

[6] T.Chen and M. Gerla, *Global State Routing: A new routing scheme for ad- hoc wireless networks,* In Proc. of IEEE International Conference on Computers and Communications (ICC '98),Atlanta,GA,June 1998.

[7] Mario Gerla, Guangyu Pei, Xiaoyan Hong, Tsu-Wei Chen, *Fisheye State Routing Protocol (FSR) for Ad Hoc Networks* , Internet Draft, draft-ietfm anet-fsr-00.txt, work in progress, June 2001.

[8] J. Garcia-Luna, M. Spohn. *Source Tree Adaptive Routing Internet* Draft, draft-ietf-manet-star-00.txt, work in progress, October 1999.

[9] T. Clausen, P. Jacquet, P. Muhlethaler, A. Laouiti, A. Qayyum, and L. Viennot, *Optimized link state routing protocol,* In IEEE INMIC'Ol, Lahore, Pakistan, Dec 2001.

[10] Mario Gerla, Xiaoyan Hong, Li Ma, Guangya Pei, *Landmark routing protocol (LANM4R),* Internet Draft, draft-ietf-manet-lanmar-0 I .txt, work in progress, June 2001.

[11] C. E. Perkins and E. M. Royer, *Ad-hoc On-Demand Distance Vector Routing, In* Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, pp. 90-100. New Orleans, LA, Feb 1999.

[12] Y.C. Hu and D. B. Johnson, *Caching -strategies in on-demand routing protocols for wireless ad hoc network,* In Proc. 6th Annual ACM/IEEE International Conf. on Mobile Computing and Networking (ACMMobiCom '00), pp 231-242, Aug 2000.

[13] M.S. Ccorson and A. Ephremides, *Lightweight mobile routing protocol (LMR) ,A distributed routing algorithm for mobile wireless networks,* Wireless Networks 1 *(1995).*

[14] D. B. Johnson, D. A. Maltz, and J. Broch, *DSR: The dynamic source routing protocol for multihop wireless ad hoc netivorks,* In Ad Hoc Networking. Addison-Wesley, 2001, ch. 5, pp. 139-172.

[15] Josh Broch, David Johnson, and David Maltz, *The dynamic source routing protocol for mobile ad hoc networks,* http://www.ietf.orglintemet-drafts/ draft-ietfm anet-dsr-03.txt, Oct 1999. IETF Internet Draft

[16] 'Joa-Ng and I.-T. Lu, *A Peer-to-Peer zone-based two-level link state routing for mobile Ad hoc Networks,* In IEEE Journal on Selected Areas in Communications, Special Issue on ad-hoc networks, Aug1999, pp.1415-25.

[17] C. E. Perkins and E. M. Royer, *Ad-hoc On-Demand Distance Vector Routing, In* Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, pp. 90-100. New Orleans, LA, Feb 1999.

[18] Christel Baier and Joost-Pieter Katoen: Principles of Model Checking. The MIT Press, 2008.

[19] Gerhard J. Holzmann: The Spin Model Checker: Primer and Reference Manual. Addison–Wesley, 2003.

[20] Marco Devillers, David Griffioen, Judi Romijn, and Frits Vaandrager: Verification of a leader election protocol: Formal methods applied to ieee 1394. Form. Methods Syst. Des.,16(3):307–320, 2000.

[21] Abrial, J.R., J.R., Cansell, D.,M´ery, D.:A Mechanically Proved and Incremental Development of IEEE 1394 Tree Identify Protocol. Formal Asp. Comput. 14(3), 215–227, 2003.

[22] A Udaya Shankar and Simon S Lam: A stepwise refinement heuristic for protocol construction.ACM Transactions on Programming Languages and Systems, 14(3):417–461,1992.

[23] Dominique M´ery and Neeraj Kumar Singh:Analysis of DSR Protocol in Event-B 13thInternational Symposium on Stabilization, Safety, and Security of Distributed Systems (2011)401-415.

[24] Arun Kumar Singh, Divakar Yadav and Vinod Kumar Singh , 2014. MODELING OF DSDV ROUTING PROTOCOL FOR AD HOC NETWORKS USING EVENT-B. International Journal of Computer Engineering & Technology (IJCET) Volume:5, Issue :2, Pages:108-116.

[25] Singh, A.K and Singh, V.K.: Formal Modeling of Distance Vector Routing Protocol using Event-B in Advance in Electronic and Electric Engineering ISSN 2231-1297, Volume 3,Number 1 (2013), pp. 91-98.

[26] Hoang, T.S., Kuruma, H., Basin, D.A., Abrial, J.R.: Developing Topology Discovery in Event-B. In: Leuschel, M., Wehrheim, H. (eds.) IFM 2009. LNCS, vol. 5423, pp. 1–19.Springer, Heidelberg (2009).

[27] Jean-Raymond Abrial. The B-book: assigning programs to meanings. CambridgeUniversity Press, 1996.

[28] Ralph-Johan Back and Reino Kurki-Suonio. Decentralization of process nets with centralized control. Distributed Computing, 3(2):73–87, 1989.

[29] Abrial, J.R.:Modeling in Event-B: System and Software Design. Cambridge University Press, 2010.

[30] Abrial, J.R.: Extending B without Changing it (for developing distributed systems). Proc. of the 1st Conf. on the B method, H. Habrias (editor), France, pages 169–190, 1996.