Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 37 (2014) 160 – 167

Procedia Computer Science

The 5th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2014)

Formal verification of a new version of AOMDV in ad hoc network

Djellouli Ahmed Amine[a,*], Abdi Mustapha Kamel[a], Kechar Bouabdellah[a]

[a] *Faculty of Exactes and Applied Sciences, Laboratory of Industrial Computing and Networking PO Box 1524 El M'naouar Oran University, Oran 31000, Algeria*

Abstract

In ad hoc networks like MANET the topology change frequently and interferences problems are inevitable in many cases, as a result link failures can arise. Unfortunately, traditional routing algorithms are no more suitable for this kind of networks especially in case of using a single path routing schemes. In order to overcome this problem, multipath routing approach is proposed where in some cases as an extension of the traditional routing algorithms. Our aim in this paper is to propose a formal study based on model checking to formally verify an enhancement version of AOMDV. In this new version we have added new functionalities in ROUTE DISCOVERY and ROUTE MAINTENANCE to achieve energy efficiency, packet overhead minimization and latency reduction.

Keywords: Ad-hoc networks; routing protocols; Formal verification; model checking;

1. Introduction

A mobile ad-hoc network or MANET is a collection of mobile nodes sharing a wireless channel without any centralized control. Each nodes act as both end systems and routers at the same time. In this kind of network with all nodes capable of movement or any other kind of network where the topology changes frequently, manage communication is very difficult especially on single path routing algorithm. We distingue tree types of routing algorithms: proactive protocol which continuously exchange routing information between the nodes; reactive

* Corresponding author. Tel.: +213-772-296145.
E-mail address: djellouli_a@ymail.com

doi:10.1016/j.procs.2014.08.025

protocol which built route on demand and hybrid protocol the combination of the two. The major drawback of proactive protocol is that the maintenance of unused paths may occupy an important part of the available bandwidth if the topology changes frequently[1]. Reactive routing protocols have some inherent limitations. First, since routes are only maintained while in use, it is usually required to perform a route discovery before packets can be exchanged between communication peers. This leads to a delay for the first packet to be transmitted. Second, even though route maintenance for reactive algorithms is restricted to the routes currently in use, it may still generate an important amount of network traffic when the topology of the network changes frequently. Finally, packets to the destination are likely to be lost if the route to the destination changes[1]. Several performance studies[2,3] of ad hoc network have shown that on demand protocols incur lower routing overheads compared to the proactive protocols. However, in dynamic network the performance will be reduced due to frequent route discovery (i.e. high route discovery latency and overhead). In order to overcome the limits of those protocols, multipath routing algorithm have been developed to overcome these limits by computing several path in a single route discovery attempt. In this case, whenever a route is broken the node will just skip to the alternative path without the need of a route discovery process, which is time intensive. An example of multipath routing algorithm AODVM[4] and AOMDV[5], both of these protocols are bases on the Ad hoc On demand Distance Vector AODV[6], which work on the principle of creating routes only if it is required between a source and destination. In spite of AOMDV which incurs more routing overhead and packet delay than AODV, many studies[7,8] has shown that AOMDV results is superior than AODV when there is mobility induced link break in distributed environment. The idea is to improve AOMDV in such a way to give better performances by reducing the routing overhead. To prove the good functioning of our new algorithm, we pass by a formal verification using the model checking[9]. This one has been successfully employed to detect ambiguities in the standard AODV and its implementations[10,11,12]. When model checking is applicable in large network protocol, such deep errors are found[13,14,15]. It consist first to build a model for the system then to verify it against specifications (expressed in a temporal logic), using a software tool called model checker. We use the tool UPPAAL instead others, due to its facilities to model the timed aspects[16,17] and especially the notion of broadcast communication that can be modeled in an easiest way[18]. Also, UPPAAL includes techniques to minimize and avoid falling into situations of explosion state[19].

The rest of this paper is organized as follows: in section 2 we review the AOMDV protocol and detailed the optimized version of AOMDV in section 3. Section 4 describes the modeling methodology under UPPAAL tool and presents the verification of model results compared to the properties. Finally, Section 6 concludes the paper.

2. The Ad hoc On Demand Multipath Distance Vector protocol

Ad-hoc On-demand Multipath Distance Vector Routing (AOMDV) protocol is an extension to the AODV protocol for computing multiple loop-free and link-disjoint paths[7]. AODV is an IP routing protocol using distance vectors (measured with hops). It consists of two procedures:

- ***ROUTE DISCOVERY process**:* The source broadcasts the RREQ (ROUTE REQUEST) packet and waits the reception of RREP packet (ROUTE REPLEY). When a node receives RREQ, it first checks if it is not a RREQ that has received earlier or an old one. In case where it's a new one, a reverse route is built to the previous node, update the fields in the RREQ and forward it, otherwise it deletes. The RREP packet is sent to the source either by an intermediate node who knows the route to the destination, or by the destination node itself.
- ***ROUTE MAINTENANCE process*:** This procedure allows a rollback to the source in case the route is broken in order to update it or to discover another. A node reports its status to the neighbors by sending a message called HELLO. In case where no HELLO message is received from a node, then a local route discovery is performed to discover an alternative path. If no route is found, an error message is diffused called RERR (ROUTE ERROR). All nodes receiving this packet invalid the route and the source node starts a new route discovery.

Before starting the description of AOMDV, we differ between node-disjoint and link-disjoint routes. As shown in Figure 1.a node-disjoint routes do not have any common nodes in two routes, but in link-disjoint two different routes may send over the same node in Figure 1.b.

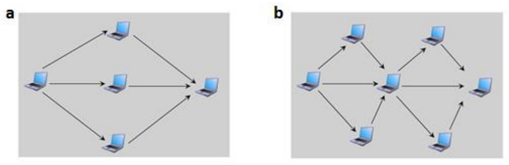
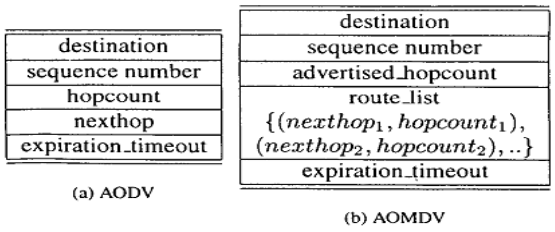


Fig. 1. (a) Node disjoints route; (b) link disjoints route.

2.1. AOMDV (Ad hoc On demand Multi-path Distance Vector)

Unlick in AODVM which is able to detect multiple node-disjoint paths between source and destination in one route discovery process, AOMDV is able to detect multiple link-disjoint paths. When receiving duplicate RREQ-packets, they are not systematically discarded. Instead, whenever an intermediate node receives a RREQ-packets it records the source who generated the RREQ, the destination for which the RREQ is intended, the neighbor who transmitted the RREQ, and some additional information (as shown in Fig. 2 (b)) in a table which is referred as RREQ table.



destination
sequence number
hopcount
nexthop
expiration_timeout

(a) AODV

destination
sequence number
advertised_hopcount
route_list $\{(nexthop_1, hopcount_1), (nexthop_2, hopcount_2), ..\}$
expiration_timeout

(b) AOMDV

Fig. 2. (a) Structure of the RREQ table entry in AODV; (b) Structure of the each routing table entry in AOMDV

AOMDV has two main components:

a) A route update rule to establish and maintain multiple loop-free paths at each node.

b) A distributed protocol to find link-disjoint paths.

The routing entries for each destination contain a list of the next-hops along with the corresponding hop counts. All the next hops have the same sequence number. This helps in keeping track of a route. Loop freedom is assured for a node by accepting alternate paths to destination if it has a less hop count than the advertised hop count for that destination, which is defined as the maximum hop count for all the paths and does not change for the same sequence number. Whenever a greater sequence number is received for a destination, the next-hop list and the advertised hop count are reinitialized.

To find multiple link-disjoint paths, AOMDV add a new field in the RREP-packets named “First Hop”, which indicate the first neighbor of the source who has received the packet. Also, each node maintain a list called “first hop-list” so that to keep a trace of the neighbors of the source which transmitted the RREQ-packet. Only one version of the packet is rebroadcasted, but keeps in memory the neighbors who send the RREQ-packets in case where the First Hop is different. This allows an intermediate node to know multiple node-disjoint paths to return to the source. The destination responds to k copies of RREQ-packets arriving via the same neighbor (independent from the First Hop) with RREP-packets in the corresponding reverse path.

Each intermediate node receiving this packet, choose one it neighbor from its routing table and transmit to him the RREP-packet. In a case where multiple RREP-packets are received by the same node, this one takes in charge to transmit each one of them to a different neighbor so that the RREP-packets follows path which are link-disjoint.

3. New optimized version of AOMDV

The idea is to improve AOMDV by minimizing the communication phases. What we try to do is to add the strengths of AODVM to AOMDV to create a hybrid version of the two containing their benefits.

3.1. New ROUTE DISCOVERY process

```
Algorithm 1 send_DATA

1:  if route exist then send the packet DATA
2:  else
3:    Broadcast the packet RREQ
4:    i = 0
5:    repeat
6:      if NTT is elapsed then NTT=NTT x 2 end if
7:      i = i + 1
8:    until i >= 3 and no RREP received
9:    if RREP is received then
10:     Update the routing table
11:     Send DATA to destination
12:   else cannot reach the destination
13:   end if
14: end if
```

```
Algorithm 2 receive_RREQ

1:  if source.RREQ_ID < node.RREQ_ID then drop the packet RREQ
2:  else
3:    update the reverse route to the source if necessary
4:    if node know a path to destination then send the packet RREP
5:    else
6:      Save the PREVIOUS_IP into the RREQ_LIST
7:      if source.RREQ_ID > node.RREQ_ID then
8:        Update RREQ
9:        Update node.RREQ_ID
10:       Broadcast the packet RREQ
11:     end if
12:   end if
13: end if
```

```
Algorithm 3 receive_RREP

1:  delete from the RREQ_LIST the node whose address is equal to RREP_PREVIOUS_IP
2:  send the packet RREP only to the nodes whose addresses are stored in RREQ_LIST
```

Fig. 3. Algorithm route discovery process of the new AOMDV

In AODV each node has its own routing table, IP address and RREQ_ID. This list has been increased with a new variable named RREQ_LIST which is a table containing the addresses of the last nodes that have broadcasted the RREQ. If a node desires to know a route to a destination, it broadcasts a RREQ. Immediately upon receipt of the packet, it will be either accepted or ignored according to the value of RREQ_ID and SOURCE_IP (same value or greater). The RREQ will be rebroadcasted only if it is the first one received (RREQ_ID greater than the value stored before), for the others only the value PREVIOUS_IP will be saved in RREQ_LIST. Those steps are executed and re-executed until that the RREQ reaches the destination or a node that knows a path to destination. This node will respond with a RREP. Each one receiving the RREP (different from the source node) retransmits it not on unicast way but on multicast to all the nodes whose address was saved in the RREQ_LIST. This allows saving multiple paths to the destination in case where it will receive several RREP. When the source receives the RREP, it starts the transmission of data packets following the shortest path. To avoid a loop, each node who receives a RREP packet from another whose address is stored in the RREQ_LIST will delete this entry from the table.

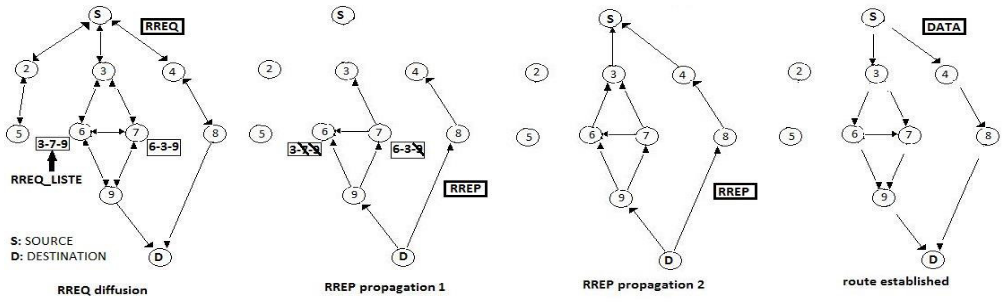


Fig. 4. Optimized version of AOMDV

3.2. New route maintenance

It is almost the same steps as AOMDV, except in case where a broken link is detected and there's no other route stored, a RERR packet will be sent only to the nodes whose address were stored in RREQ_LIST in multicast way (to avoid disturbing nodes not concerned) and the DATA packet will be sent back to the last one who have transmitted it. This last node will choose another route in its table (if it exists), otherwise the DATA will be sent to the earlier one. If the DATA reach the source node, this one will wait a while before restarting a new ROUTE DISCOVERY.

4. Modeling the protocol using model checking

Formal verification is a combination of techniques that allow verifying rigorously computer programs or electronic equipment using mathematical logic in order to demonstrate their validity. Their aim is to establish system correctness with mathematical rigor. Model checking is an automated technique that, given a finite state model of a system and a formal property, systematically checks whether this property is true for all the states of the system. In this way, we can show that a model system really satisfies a certain property. Same errors that were not discovered using test and simulation can be revealed using model checking.

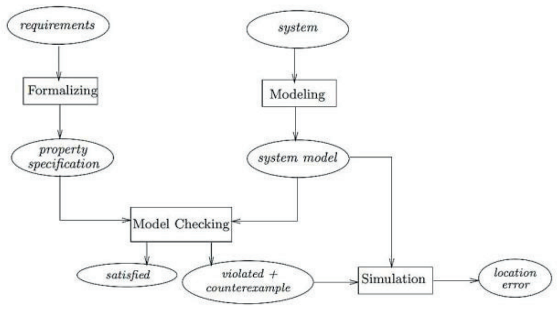
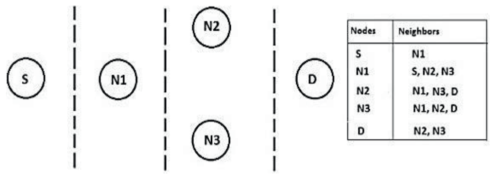


Fig. 5. Schematic view of the model-checking approach[19]

4.1. Modeling methodology

Each node acts either as a transmitter, receiver or an intermediate node. We propose a model of n-node, where each one acts as a specific role and to reduce the model, we will ignore HELLO message and the fields ‘previous’ and ‘next sequence number’. The IP addresses are represented with integer numbers from 0 to N-1, where the node “0” is the source and “N-1” the destination, the rest is intermediates nodes which are dispersed according to the topology showed in fig 6.



Nodes	Neighbors
S	N1
N1	S, N2, N3
N2	N1, N3, D
N3	N1, N2, D
D	N2, N3

Fig. 6. Topology of the network

Also we have taken into consideration the case of packet losses. The functions of each node are described below:

- ***The source node***: generate and send RREQ packets, receive RREP packets and transmit the DATA.
- ***The intermediate node***: receive the RREQ, RREP, RERR and DATA packets and resend it.
- ***The destination node***: receive the RREQ, RERR and DATA packets, also generate and send RREP.

4.2. UPPAAL models

A. Link failure

It simulates a link failure in the middle of data transmission, between N1 and N2 or N1 and N3 according to the next hop of the DATA (data[2]) as shown in fig 7.

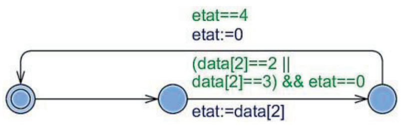
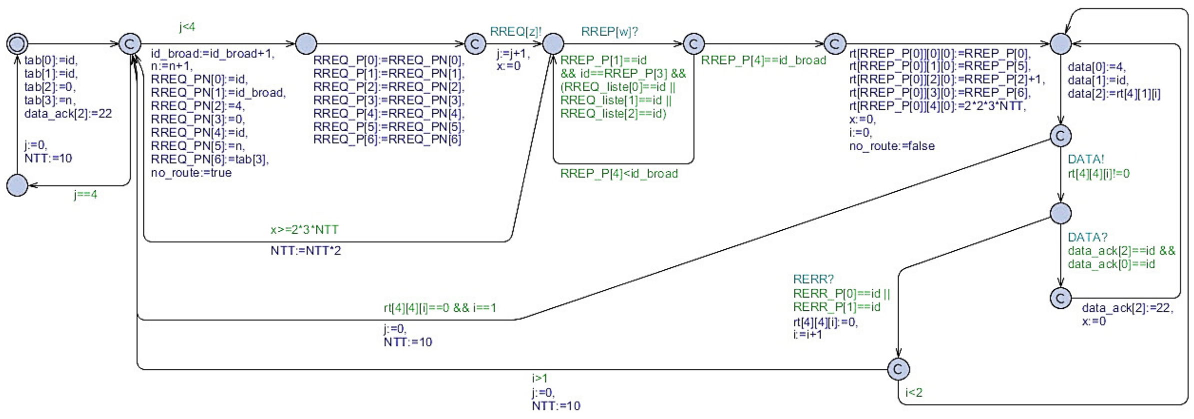


Fig. 7. Template of link failure

B. Optimized version of AOMDV: the source node

First, the node starts with an initialization phase of its routing table, increments its id_broadcast, prepares the RREQ packet and broadcast it. If no RREP is received, the NTT is doubled and another RREQ is rebroadcasted with id_broadcast incremented, otherwise the routing table is updated and we begin the transmission of the DATA. While sending the DATA, the node can choose another path to destination (if exist) if the first one is broken. If no other route is known it discovers a new path.

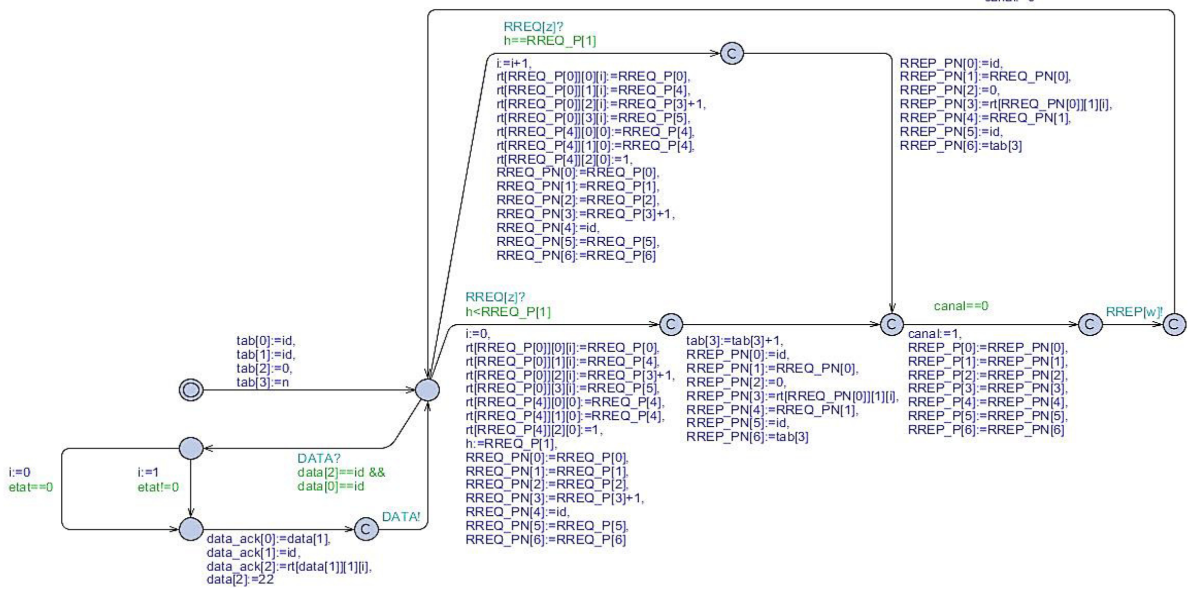


j<4
tab[0]:=id, tab[1]:=id, tab[2]:=0, tab[3]:=n, data_ack[2]:=22
j:=0, NTT:=10
j==4
id_broad:=id_broad+1, n:=n+1, RREQ_PN[0]:=id, RREQ_PN[1]:=id_broad, RREQ_PN[2]:=4, RREQ_PN[3]:=0, RREQ_PN[4]:=id, RREQ_PN[5]:=n, RREQ_PN[6]:=tab[3], no_route:=true
RREQ_P[0]:=RREQ_PN[0], RREQ_P[1]:=RREQ_PN[1], RREQ_P[2]:=RREQ_PN[2], RREQ_P[3]:=RREQ_PN[3], RREQ_P[4]:=RREQ_PN[4], RREQ_P[5]:=RREQ_PN[5], RREQ_P[6]:=RREQ_PN[6]
RREQ[z]!
j:=j+1, x:=0
RREP[w]?
RREP_P[1]==id && id==RREP_P[3] && (RREQ_liste[0]==id || RREQ_liste[1]==id || RREQ_liste[2]==id)
RREP_P[4]<id_broad
RREP_P[4]==id_broad
rt[RREP_P[0]][0][0]:=RREP_P[0], rt[RREP_P[0]][1][0]:=RREP_P[5], rt[RREP_P[0]][2][0]:=RREP_P[2]+1, rt[RREP_P[0]][3][0]:=RREP_P[6], rt[RREP_P[0]][4][0]:=2*2*3*NTT, x:=0, i:=0, no_route:=false
data[0]:=4, data[1]:=id, data[2]:=rt[4][1][i]
DATA! rt[4][4][i]!=0
DATA? data_ack[2]==id && data_ack[0]==id
data_ack[2]:=22, x:=0
x>=2*3*NTT
NTT:=NTT*2
rt[4][4][i]==0 && i==1
j:=0, NTT:=10
RERR? RERR_P[0]==id || RERR_P[1]==id
rt[4][4][i]:=0, i:=i+1
i>1 j:=0, NTT:=10
i<2

Fig. 8. Source node template of the new version of AOMDV

C. Optimized version of AOMDV: the destination node

The destination node accepts the RREQ packet (if the conditions are met), updates its routing table and generates the corresponding RREP. It also receives DATA and responds with the ACK corresponding and also may choose an alternative path to the source whenever the selected route is broken.



canal:=0
RREQ[z]? h==RREQ_P[1]
i:=i+1, rt[RREQ_P[0]][0][i]:=RREQ_P[0], rt[RREQ_P[0]][1][i]:=RREQ_P[4], rt[RREQ_P[0]][2][i]:=RREQ_P[3]+1, rt[RREQ_P[0]][3][i]:=RREQ_P[5], rt[RREQ_P[4]][0][0]:=RREQ_P[4], rt[RREQ_P[4]][1][0]:=RREQ_P[4], rt[RREQ_P[4]][2][0]:=1, RREQ_PN[0]:=RREQ_P[0], RREQ_PN[1]:=RREQ_P[1], RREQ_PN[2]:=RREQ_P[2], RREQ_PN[3]:=RREQ_P[3]+1, RREQ_PN[4]:=id, RREQ_PN[5]:=RREQ_P[5], RREQ_PN[6]:=RREQ_P[6]
RREP_PN[0]:=id, RREP_PN[1]:=RREQ_PN[0], RREP_PN[2]:=0, RREP_PN[3]:=rt[RREQ_PN[0]][1][i], RREP_PN[4]:=RREQ_PN[1], RREP_PN[5]:=id, RREP_PN[6]:=tab[3]
RREQ[z]? h<RREQ_P[1]
canal==0
RREP[w]!
tab[0]:=id, tab[1]:=id, tab[2]:=0, tab[3]:=n
i:=0, rt[RREQ_P[0]][0][i]:=RREQ_P[0], rt[RREQ_P[0]][1][i]:=RREQ_P[4], rt[RREQ_P[0]][2][i]:=RREQ_P[3]+1, rt[RREQ_P[0]][3][i]:=RREQ_P[5], rt[RREQ_P[4]][0][0]:=RREQ_P[4], rt[RREQ_P[4]][1][0]:=RREQ_P[4], rt[RREQ_P[4]][2][0]:=1, h:=RREQ_P[1], RREQ_PN[0]:=RREQ_P[0], RREQ_PN[1]:=RREQ_P[1], RREQ_PN[2]:=RREQ_P[2], RREQ_PN[3]:=RREQ_P[3]+1, RREQ_PN[4]:=id, RREQ_PN[5]:=RREQ_P[5], RREQ_PN[6]:=RREQ_P[6]
tab[3]:=tab[3]+1, RREP_PN[0]:=id, RREP_PN[1]:=RREQ_PN[0], RREP_PN[2]:=0, RREP_PN[3]:=rt[RREQ_PN[0]][1][i], RREP_PN[4]:=RREQ_PN[1], RREP_PN[5]:=id, RREP_PN[6]:=tab[3]
canal:=1, RREP_P[0]:=RREP_PN[0], RREP_P[1]:=RREP_PN[1], RREP_P[2]:=RREP_PN[2], RREP_P[3]:=RREP_PN[3], RREP_P[4]:=RREP_PN[4], RREP_P[5]:=RREP_PN[5], RREP_P[6]:=RREP_PN[6]
i:=0 etat==0
i:=1 etat!=0
DATA? data[2]==id && data[0]==id
DATA!
data_ack[0]:=data[1], data_ack[1]:=id, data_ack[2]:=rt[data[1]][1][i], data[2]:=22

Fig. 9. Destination node template of the new version of AOMDV

D. Optimized version of AOMDV: the intermediate node

The node initializes its routing table and waits to receive different packets (RREQ, RREP and DATA). This time several packets can be accepted even if they have the same RREQ_ID. Also, if the link used to transmit the DATA is broken, the node will choose directly an alternative path to destination. If there is no other path, a RERR packet is sent only the nodes whose addresses were stored in the RREQ_LIST and the DATA is sent back.

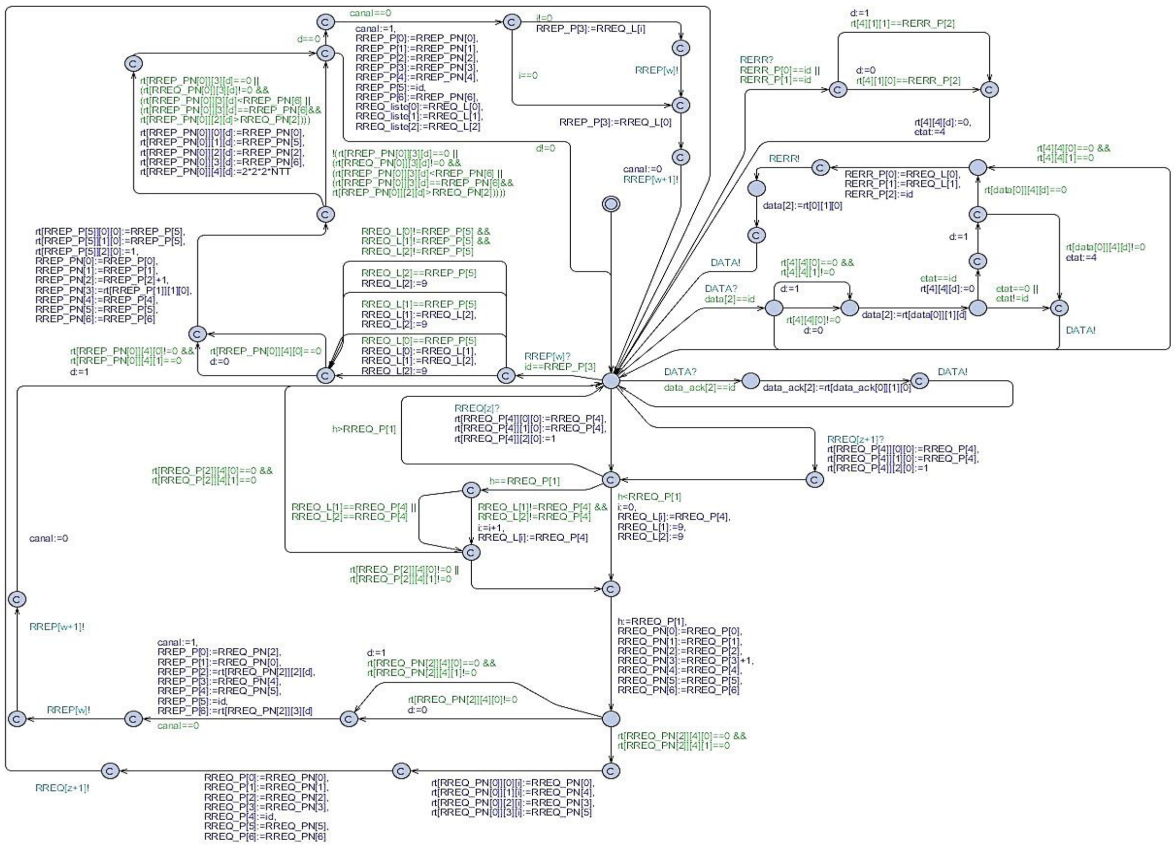


Fig. 10. Intermediate node template of the new version of AOMDV

4.3. Properties analysis

A. Property 1: If a route exists then it will be found (reachability property) **A[] no_route==TRUE**

In the initial phase **no_route** will have the value TRUE, which means "no path is known between source and destination". This property means that the value of **no_route** will never change. If we can manage to find at least one counterexample to it, the protocol is able to reach to destination (it can also not be able to find a route to destination because we have taken into consideration the packet losses). The property is unsatisfied; this means that there is a scenario where the source is able to discover a route to destination by receiving a RREP packet.

B. Property 4: The protocol will never fall into a loop (safety property) **A[] n1.rt[4][2][0]<=4**

"rt" references to the routing table associated to the node 1. The property is satisfied, thanks to RREQ_LIST which deletes each entry associated to a node sending RREP. This new version of AOMDV also avoids falling in a route discovery failure[11], where RREQ_LIST requires each node to forward the RREP in multicast way to each neighboring nodes whose address has been saved in this table.

C. Property 3: The protocol will never fall into a deadlock state (safety property) **A[] not deadlock**

The property is satisfied. This means that in any situation, thin or thick network, the protocol will be functional and never falls into a deadlock state.

D. Property 2: Behavior of the protocols during a link failure (reachability property) **E<> route_broke==false**

The template *Link Failure* simulates a broken link in the middle of transmitting DATA using the variable **etat** which indicates the state of the chosen path. It will have the value "2" or "3" (reference to the link between "destination" and "N2", "N3"). The variable **route_broke** take the value false only if "N1" choose an alternative path to the destination after receiving the RERR. The property is satisfied; "N2"/"N3" will send the RERR only to nodes whose addresses are stored in the RREQ_LIST instead of broadcasting it; after that "N1" choose directly the second route to "D" stocked in its routing table. If there is no other path, then a RERR packet is sent only to "S".

5. Conclusion

In this paper, an enhancement version of the routing protocol AOMDV was proposed, where the main objective is to minimize the energy consumption and to discover multiple paths to a destination. In this new version, our focus was mainly on reducing the number of packets transmitted while minimizing the number of conditions and tests required compared to the old version. To validate it, we proposed a formal verification using a powerful model checking approach. As a future remaining work, we plan to make a deep performance evaluation by simulation to show the effectiveness of the new version of AOMDV over the old one.

References

1. H.D.Trung, W.Benjapolakul, P.M.Duc, "Performance evaluation and comparison of different ad hoc routing protocols", Computer Communications, Volume 30, Issues 11–12, 10 September 2007, Pages 2478–2496.
2. Nilesh P. Bobade, Nitiket N. Mhala, "Performance evaluation of AODV and DSR on-demand routing protocols with varying MANET size", International Journal of Wireless & Mobile Networks (IJWMN) Vol. 4, No. 1, February 2012.
3. Pranav M. Pawar, Smita Shukla, Pranav Kulkarni and Adishri Pujari, "Simulation and Proportional Evaluation of AODV and DSR in Different Environment of WSN", BVICAM's International Journal of Information Technology, accepted November 2010, January – June, 2011; Vol. 3 No. 1; ISSN 0973 – 5658
4. Zhenqiang Ye, Srikanth V. Krishnamurthy, Satish K. Tripathi, "A Framework for Reliable Routing in Mobile Ad Hoc Networks"
5. Mahesh K.Marina, Samir R.Das, "On-demand Multipath Distance Vector Routing in Ad hoc network", Proceedings of the International Conference for Network Protocols (ICNP), pp. 14–23, Nov. 2001.
6. C. E. Perkins, E. B. Royer, and S. R. Das, "Ad Hoc On- Demand Distance Vector (AODV) Routing", RFC 3561, 2003.
7. S. R. Biradar, Koushik Majumder, Subir Kumar Sarkar, Puttamadappa C, "Performance Evaluation and Comparison of AODV and AOMDV", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 02, 2010, 373-377.
8. K.Vanaja, Dr. R. Umarani, "An Analysis of Single Path AODV Vs Multipath AOMDV on Link Break Using ns-2", International Journal of Electronics and Computer Science Engineering ISSN- 2277-1956
9. J.Edmund M.Clarke, O.Grumbdge, and D.A.Peledm "Model checking", MIT Press, Cambridge, MA, USA, 1999.
10. Peter Höfner, Wee Lum Tan, Annabelle McIver, Rob van Glabbeek, Marius Portmann and Ansgar Fehnker, " A Rigorous Analysis of AODV and its Variants", MSWiM '12 Proceedings of the 15th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems, Pages 203-212, 2012.
11. Peter Hofner and Maryam Kamali, "Quantitative Analysis of AODV and its Variants on Dynamic Topologies using Statistical Model Checking", Formal Modeling and Analysis of Timed Systems, Volume 8053, page 121-136, 2013.
12. M.Musuvathi, Dawson R.Engler "Model Checking Large Network Protocol implementations", NSDI'04 Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1, 2004.
13. K.L.McMillan, J.Schwalbe. "Formal verification of the gigamax cache consistency protocol", In Proceedings of the International Symposium on Share d Memory Multiprocessing, pages 242-51, Tokyo, Japan Inf. Process. Soc., 1991.
14. G. Nelson, "Techniques for program verification", Stanford University, 1981.
15. U. Stern, D.L. Dill, "Automatic verification of the SCI cache coherence protocol", In Correct Hardware Design and Verification Methods: IFIP WG10.5 Advanced Research Working Conference Proceedings, 1995.
16. Sibusisiwe Chiyangwa, Marta Kwiatkowaska, "Analysing Timed Properties of AODV with UPPAAL", FMOODS'05 Proceedings of the 7th IFIP WG 6.1 international conference on Formal Methods for Open Object-Based Distributed Systems, pages 306-321, 2005.
17. Sibusisiwe Chiyangwa, Marta Kwiatkowska, "Modelling Ad hoc On-demand Distance Vector (AODV) Protocol with Timed Automata", In Proceedings of the Thirds Work6shop on Automated Verification of Critical Systems (AVoCS'03), Southampton, UK, April 2003.
18. Henrik Ejersbo Jensen Kim G. Larsen Arne Skou "Modelling and Analysis of a Collision Avoidance Protocol using SPIN and UPPAAL", BRICS RS-96-24, ISSN 0909-0878, July 1996.
19. J.Bengtsson, K.G.Larsen, F.Larsson, P.Petersson and W.YI. UPPAAL – a tool suite for symbolic and compositional verification of real-time systems. In proceeding of the first workshop on tools and algorithms for the constructions and analysis of systems, volume 1019 of lecture notes in computer science. Springer-verlag, May 1995.
19. Christel Baier and Joost-Pieter Katoen, "Principles of Model Checking", The MIT Press, Cambridge, Massachusetts, London England, 2008.