Quantitative Analysis of AODV and its Variants on Dynamic Topologies using Statistical Model Checking

Peter Höfner[1,4], Maryam Kamali[2,3]

[1] NICTA, Australia
[2] Turku Centre for Computer Science (TUCS), Finland
[3] Åbo Akademi University, Finland
[4] University of New South Wales, Australia

Abstract. Wireless Mesh Networks (WMNs) are self-organising ad-hoc networks that support broadband communication. Due to changes in the topology, route discovery and maintenance play a crucial role in the reliability and the performance of such networks. Formal analysis of WMNs using exhaustive model checking techniques is often not feasible: network size (up to hundreds of nodes) and topology changes yield state-space explosion. Statistical Model Checking, however, can overcome this problem and allows a quantitative analysis.
In this paper we illustrate this by a careful analysis of the Ad hoc On-demand Distance Vector (AODV) protocol. We show that some optional features of AODV are not useful, and that AODV shows unexpected behaviour—yielding a high probability of route discovery failure.

1 Introduction

Route finding and route maintenance are critical for the performance of networks. Efficient routing algorithms become even more important when mobility of network nodes lead to highly dynamic and unpredictable environments. The Ad hoc On-Demand Distance Vector (AODV) routing protocol [16] is such an algorithm. It is widely used and particularly designed for Wireless Mesh Networks (WMNs), self-organising ad-hoc networks that support broadband communication.

Formal analysis of routing protocols is one way to systematically analyse protocols for flaws and to present counterexamples to diagnose them. It has been used in locating problems in automatic route-finding protocols, e.g. [1,4]. These analyses are performed on tiny static networks (up to 5 nodes). However, formal validation of protocols for WMNs remains a challenging task: network size (usually dozens, sometimes even hundreds of nodes) and topology changes yield an explosion in the state space, which makes exhaustive model checking (MC) techniques infeasible. Another limitation of MC is that a quantitative analysis is often not possible: finding a shortcoming in a protocol is great but does not show how often the shortcoming actually occurs.

Statistical model checking (SMC) [20,19] is a complementary approach that can overcome these problems. It combines ideas of model checking and simulation with the aim of supporting quantitative analysis as well as addressing the size barrier. SMC trades certainty for approximation, using Monte Carlo style sampling, and hypothesis testing to interpret the results.

In this paper we demonstrate that SMC can be used for formal reasoning of routing protocols in WMNs. We perform a careful analysis of different versions of the AODV protocol. In particular, we analyse how dynamic topologies can affect the protocol behaviour. In other words, we analyse the performance of the protocol while the network topology evolves. We show that some optional features provided by AODV should be avoided since they affect the performance of the protocol. Moreover, we show that in some scenarios the behaviour of AODV is not as intended yielding a high probability of route discovery failure. When possible we suggest improvements of the protocol.

The paper is organised as follows: in Sect. 2 we give an overview of AODV, present optional features such as the resending of route requests, and sketch the encoding of AODV in SMC-Uppaal, the statistical extension of Uppaal. In Sect. 3 we describe the mobility model, which is used for our analysis of AODV. Sect. 4 discusses the experiments performed, the main contribution of this paper: (i) We show that a single mobile node can have a massive impact on the success of route discovery. Moreover we show that some options of AODV should not be used in combination, unless the protocol specification is adapted (changed). (ii) A second category of experiments reveals a surprising observation: adding "noise" (for example an additional data packet) to a network can increase the success of route discovery. (iii) The third category discusses the consequences of different speeds of mobile nodes. The paper closes with a discussion of related work in Sect. 5 and a short outlook in Sect. 6.

2 AODV, its Variants and their Uppaal Models

2.1 The Basic Model

The AODV routing protocol [18] is a widely used routing protocol, particularly tailored for WMNs. It is currently standardised by the IETF MANET working group and forms the basis of new WMN routing protocols, including HWMP in the upcoming IEEE 802.11s wireless mesh network standard [12].

AODV is a reactive protocol, meaning that a route discovery process is only initiated when a node S in the network has to send data to a destination D for which it does not have a valid entry in its own routing table. The route discovery process starts with node S broadcasting a route request (RREQ) message, which is received by all nodes within S's transmission range. If a node, which is different to the destination, receives a RREQ message and does not have a valid entry for the destination in its routing table, the request is forwarded by re-broadcasting the RREQ message. During this forwarding process, the intermediate node updates its routing table and adds a "reverse route" entry with destination S into its routing table, indicating via which next hop the node S can

be reached, and the distance in number of hops. To avoid unnecessary message sending each RREQ has a unique identifier which allows nodes to ignore RREQ messages that they have handled before.

As soon as the RREQ is received by the destination itself or by a node that knows a valid route to the destination, a route reply (RREP) is generated. In contrast to RREQ messages, a RREP message is unicast, i.e., it is only sent to a single node, not to all nodes within transmission range. The RREP message travels from its generator (either D or an intermediate node knowing a route to D) back along the established route towards S, the originator of the RREQ message. All intermediate nodes on the selected route will process the RREP message and, in most cases, forward it towards S. However, there are scenarios where RREP message are discarded (see below). By passing a RREP message towards S, a node adds a "forward route" entry to its routing table.

The route discovery process is completed when the RREP reaches node S; an end-to-end route from S to D has been established, and data packets can start to flow. If any link breaks down (e.g. by a node moving out of transmission range), the node that detects the break broadcasts a route error (RERR) message.[1] All notified nodes invalidate their routing table entries that use the broken link and forward the RERR message if necessary.

Full details can be found in RFC 3561 [16], the de facto standard of AODV.

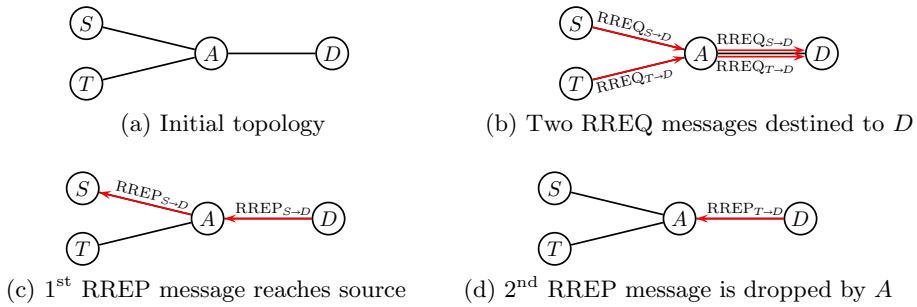
2.2 Variants of AODV

The specification of AODV [16] offers optional features, which yield different variants of the routing protocol. One aim of this paper is to compare versions of AODV with different features turned on.

Destination Only (D) Flag. Each RREQ message contains a field called *destination only flag*. If the value of this Boolean flag is `true`, it indicates that only the destination node is allowed to respond to this RREQ. That means that the RREQ travels through the entire network until it reaches the destination. Only then a reply is sent back. By this a *bi-directional link* between the source and the destination is (usually) established.

Resending a Route Request. The basic version of AODV, as presented in the previous section, suffers the problem that some routes, although they do exist, are not discovered. Reasons for route discovery failure can be message transmission failures (the receiver of a unicast message has moved out of transmission range) or the dropping of RREP messages, that should be forwarded. With respect to the latter, the problem is that a node only forwards a RREP message if it is not the originator node, *and* it has created or updated a routing table entry to the destination node described in the RREP message. [16]

[1] Following the RFC, a node uses precursor lists to store those nodes that are interested in some particular routes—when sending an RERR message only those neighbours are informed. However, precursor lists do not contain all neighbours that are interested in a particular route (e.g. [8]); that is why we model an improved version of AODV where RERR messages are broadcast.



(a) Initial topology (b) Two RREQ messages destined to D

(c) 1st RREP message reaches source (d) 2nd RREP message is dropped by A

Fig. 1. Route Discovery Failure

An example for route discovery failure, taken from [10], is sketched in Fig. 1.[2] On the 4-node topology depicted in Part (a) nodes S and T, resp., initiate a route discovery process to search for a route to D. The messages travel through the network and reach the destination D (Part (b)). We assume that $\text{RREQ}_{S\to D}$, the request stemming from S, reaches node D first. In Part (c), D handles $\text{RREQ}_{S\to D}$, creates an entry for S in its routing table[3] and unicasts a RREP message to A. Node A updates its routing table (creates an entry for D) and forwards the message to the source S. In Part (d), D handles $\text{RREQ}_{T\to D}$, creates an entry for T in its routing table and unicasts a RREP message to A. Since $\text{RREP}_{T\to D}$ does not contain new information for A (a route to D is already known), node A does not update its routing table and, according to the specification, will not forward the RREP message to the source T. This leads to an unsuccessful route discovery process for node T.

The solution proposed by the RFC is to initiate a new route discovery process, if no route has been established 2 seconds after the first request was sent; the number of retries is flexible, but the specification recommends one retry only. In the example node T would initiate another route request; node A, which receives the RREQ message, will immediately unicast a RREP message back to T.

Local Repair. In case of a link break, the node upstream of that break can choose to repair the link locally if the destination was no farther away than a predefined number of hops (the number is specified by the user and often depends on the network size). When a node receives a RREP message or a data packet destined for a node for which it does not have a valid route, the node buffers the message and initiates a new route discovery process. As soon as a route has been re-established, the buffered message is sent.

2.3 Modelling AODV and its Variants in Uppaal

Table 1 lists all variants of AODV that are modelled, analysed and compared in this paper. The analysis is performed by SMC-Uppaal, the statistical extension

[2] A similar example has been published at the IETF mailing list in 2004; `http://www.ietf.org/mail-archive/web/manet/current/msg05702.html`.

[3] Routing tables are not presented in the figure.

name	**optional features**	**remark**
basic	none	follows description of Sect. 2.1
resend	resending RREQ	"standard" configuration of AODV
dflag	D-flag	the flag is set for all route discovery processes
dflag_res	D-flag and resending	this configuration has a flaw (see below)
dflag_res'	D-flag and resending	not following the RFC literally, but flaw fixed
repair	local repair	use local repair

Table 1. Different Variants of AODV

of Uppaal [3]. The modelling language for SMC-Uppaal is the same as for "standard" Uppaal, namely networks of guarded, timed and probabilistic automata.

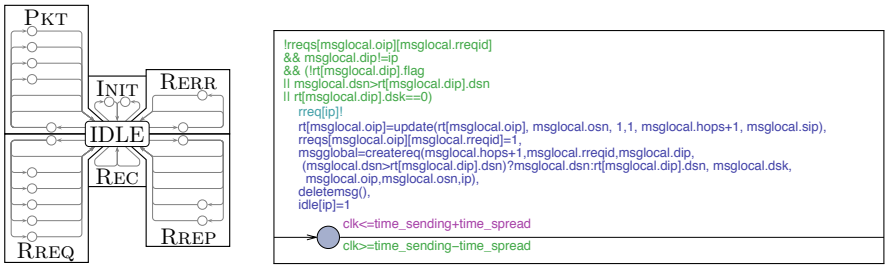
Our 6 models of (all variants of) AODV are based on a single untimed Uppaal model that was used to analyse some basic qualitative properties [7].[4] Since we are interested in a quantitative analysis of the protocol, the model had to be equipped with time and probability. The latter is needed to model dynamic topologies and mobile nodes. Hence, the (untimed) model was significantly redesigned and extended to include timing constraints on sending messages between nodes. Both the untimed and the timed model were systematically derived from an unambiguous process-algebraic model that models the intention of the RFC and does not contain contradictions. Communication between nodes had to be modelled so that the unicast behaviour of AODV was correctly rendered using SMC-Uppaal's (only) broadcast mechanism.

Each node of a network is modelled by two timed automata: the first models a message queue that buffers received messages, the other models the AODV routine. This main routine consists of ~ 20 locations, 1 clock measuring the sending time, and a complicated data structure with approx. 10 variables. The latter includes an array `rt` of length N modelling the routing table, where N is the number of nodes in the network. The overall structure of the main automaton is depicted in Fig. 2(a), it consists of 7 regions. If the automaton is in the region Idle, which consists of one location only, then AODV does not perform any action in the moment and the automaton is ready to receive messages. This happens in Rec if there is at least one message buffered. The regions Rreq, Rrep, Rerr and Pkt perform actions depending on the type of the received message. Rreq for example handles route request messages. Init initiates the transmission of data injected by the user as soon as the route is established.

Message handling often contains actions for updating the internal data (such as routing tables) and sending of a message. Fig. 2(b) gives an impression of such an update by showing a snippet of the automaton modelling the forwarding of a RREQ message.

Message sending is the only action that takes time: according to the specification of AODV [16], the most time consuming activity is the communication between nodes, which takes on average 40 milliseconds; all other times are marginal and assumed to be 0.

[4] Our models can be found at `http://www.hoefner-online.de/formats2013/`.



(a) Structure (b) Detail: updating data and forwarding a message

Fig. 2. Overall structure of the SMC-Uppaal model of AODV

Our models cover all core components of AODV. However, we encoded one main assumption: whenever a message is sent and the receiver of the message is within transmission range, the message will be received. In reality message loss during transmission happens regularly, for example due to communication failures or packet collisions. This loss could easily be modelled using Uppaal's broadcast mechanism in combination with probabilistic automata. However, this abstraction enables us to interpret a failure of guaranteed message delivery as an imperfection in the protocol, rather than as a result of a chosen formalism not allowing guaranteed delivery. Due to lack of space we cannot give more details about the modelling; more details about the model *basic* can be found in [11].

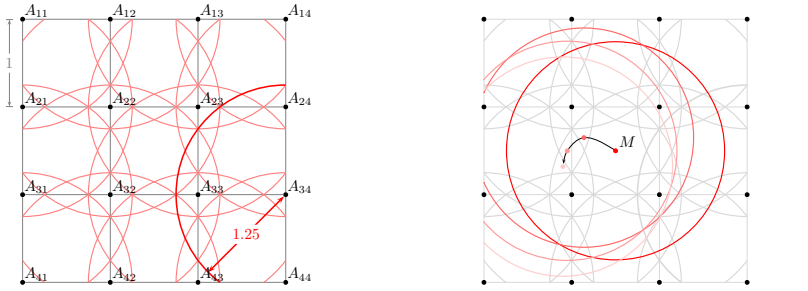
Next to the automata modelling the behaviour of the node, two additional automata are needed: the first is a scenario generator initiating the route discovery process, i.e., it forces one of the nodes to generate and broadcast a RREQ message. The second automaton models the mobility within the network.

3 Modelling Dynamic Topologies

To analyse quantitative properties of AODV and to compare different variants in a dynamic network, we use a *topology-based mobility model* [9]. It reflects the impact of mobility on the network topology and distinguishes static and mobile nodes; only connections to and from mobile nodes can change. Each movement is characterised either by adding a new link to or by removing an existing link from the *connectivity graph*. Whenever a mobile node M enters the transmission range of a node A, a new link is established between M and A. If M leaves the transmission range, the link between these two nodes is removed.

To decrease the number of possible topology changes due to a large number of mobile nodes, we set up the topology as follows: the network consists of 16 static and one mobile node.[5] The static nodes form a 2-dimensional rectangular grid with grid size 1, i.e. the smallest distance between two nodes is 1 unit (cf. Fig. 3(a)); the transmission range is set to 1.25. In reality, 1 unit might correspond to 100 metres, the transmission range to 125 m, a realistic value.

[5] We also performed experiments with more than one mobile node; but these experiments do not show new (odd) results.



(a) grid with 16 nodes (static topology)

(b) node moving between zones (transmission ranges are indicated)

Fig. 3. Topology-based mobility model

The transmission ranges of the nodes A_{ij} $(1 \leq i, j \leq 4)$ split the grid into 102 different zones. The different zones are shown in Fig. 3(a). When a mobile node M moves within a zone, the exact position of the node does not matter, since it does not enter or leave the transmission range of any node—the connectivity graph stays the same. For example any node that is within the central zone is connected to nodes A_{22}, A_{23}, A_{32} and A_{33} (cf. Fig. 3(b)) When M transits the border of a zone, it triggers a network topology change. Only the change of the connectivity graph is considered, other details such as the exact direction and angle of transmitting are not needed for characterising the dynamic network. In the example given in Fig. 3(b), M moves to the left and enters the transmission range of A_{21}. Next, in fainter colours, the node enters transmission range of A_{31} and leaves the range of A_{23}.

The topology-based model captures the topology changes as a Markovian transition function $\texttt{prob}(T_1, T_2)$, that assigns to two topologies T_1 and T_2 a transition probability. The probability of moving from one zone to a neighbouring zone is based on the ratio of the length the two zones share compared to the overall border length of the zone in which the node is in. For instance, the probability of transiting from the central segment of the grid to any adjacent zone is $\frac{1}{8}$, due to equal segment lengths.

Our model sets the speed of the mobile node in such a way that the node has to change zones every 35–45 time units, where the probability to leave the zone at time t is equally distributed in the interval.

The zones can be grouped by their shapes; each shape forms an equivalence class. The Uppaal model reflects this observation. Each mobile node is modelled by a separate timed and probabilistic automaton; each location of the automaton characterises exactly one equivalence class. (See [9] for details.)

4 Experiments

Our experiments analyse the impact of mobile nodes and dynamic topologies on AODV; they are grouped into several categories: the first category analyses the probability of route establishment for a single route discovery process, i.e., an

originator node `oip` is searching for a route to `dip`; the second category analyses the likelihood of route establishment between `oip` and `dip` when additional route discovery processes occur; the last category changes the speed of the mobile node.

Before discussing the experiments, we briefly describe some foundations of statistical model checking. SMC [20,19] combines ideas of model checking and simulation with the aim of supporting quantitative analysis as well as addressing the size barrier that prevents useful analysis of large models. By trading certainty for approximation, it uses Monte Carlo style sampling, and hypothesis testing to interpret the results. The sampling follows the probability distribution defined by the non-deterministic and probabilistic automata. Parameters setting thresholds on the probability of false negatives (α) and on probabilistic uncertainty (ε) can be used to specify the statistical confidence on the result. SMC-Uppaal computes the number of simulation runs needed by using Chernoff-Hoeffding bounds, which is independent of the size of the model; it generates an interval $[p-\varepsilon, p+\varepsilon]$ for estimating p, the probability of CTL-property ψ holding w.r.t. the underlying probability distribution.

For most of our experiments we use "only" a confidence level of 95% and allow a large probabilistic interval of 10%—this is the default setting of SMC-Uppaal and means that both α and ε are set to 5%. When using this set up, SMC-Uppaal simulates 738 runs to determine the probability of a property.

Experiments with $\alpha = \varepsilon = 1\%$ (26492 runs) are also feasible with a standard desktop machine, but require much more time. While an experiment using a confidence level of 95% takes only a couple of minutes; an experiment using a level of 99% takes more than 3 hours. We illustrate this by our first experiment.

4.1 Single Route Discovery Process

Our first experiment is based on 17 nodes; 16 forming a grid (Fig. 3(a)) and one mobile node M which is located in the middle of the grid at the beginning of the experiment. After a delay between 140 and 160 time units (the time that the mobile node needs to perform four movements) the first RREQ message is broadcast. By this delay, the location of M is random at the point the route discovery process is initiated.

In the experiment A_{11} searches for a route to A_{44}, that means it initiates a route discovery process. We are interested whether (and at which time) A_{11} establishes a route to A_{44}. In Uppaal syntax this reachability property is

```
Pr[<=2000](<>A11.rt[A44].nhop!=0) .        (1)
```

Checking this query determines the probability (`Pr`) satisfying the CTL-path expression `<>(A11.rt[A44].nhop!=0)` with a time bound of 2000 time units; we choose this bound as a conservative upper bound to ensure that the analyser explores paths to a depth where the protocol is guaranteed to have terminated. The term `ip.rt[dip]` refers to a route to `dip` stored inside the routing table of node `ip`. Whenever the next hop `nhop` is set ($\neq 0$), a route has been established.

The results are summarised in Table 2. From an experimental point of view, the table shows that a confidence level of 99% does not yield much better results

	model	conf. level	probability route discovery	time route discovery	Uppaal running time
1.	*basic*	95%	[55.4336,65.4336]	595.67	$4m\,18s$
2.	*basic*	99%	[59.7806,61.7806]	597.52	$157m\,44s$
3.	*resend*	95%	[95.00,100.00]	847.04	$6m\,03s$
4.	*resend*	99%	[98.9623,100.00]	836.87	$209m\,43s$
5.	*dflag*	95%	[54.4851,64.4851]	597.50	$4m\,52s$
6.	*dflag*	99%	[59.5655,61.5655]	597.63	$164m\,21s$
7.	*dflag_res*	95%	[64.5122,74.5122]	698.70	$7m\,47s$
8.	*dflag_res*	99%	[68.2133,70.2133]	688.79	$249m\,12s$
9.	*dflag_res'*	95%	[81.3144,91.3144]	822.89	$7m\,08s$
10.	*dflag_res'*	99%	[83.4104,85.4104]	807.43	$230m\,57s$
11.	*repair*	95%	[59.7696,69.7696]	607.86	$7m\,31s$
12.	*repair*	99%	[63.5742,65.5742]	606.53	$165m\,11s$

Table 2. Single Route Discovery Ratio (confidence level 95% and 99%)[6]

than a confidence level of 95%; but the running times of Uppaal (last column) are much higher (in average by a factor of 33.6).

Next to the running times of Uppaal the table lists the model (first column), the probability of a successful route discovery (third column) and the average time needed to establish a route between A_{11} and A_{44} (fourth column). It is no surprise that the models *basic* and *dflag* yield the same results—in this setting they behave identically. Furthermore, it is obvious that the probability for successful route discovery increases when using the *resend* option, while at the same time the discovery time increases as well. However, the experiments reveal three surprising and unexpected observations concerning AODV.

Observation 1 A single mobile node can already have a massive impact on the success of route discovery. In our setting the probability of route discovery can decrease by about 40%.

Using the same setting without mobility (e.g., the mobile node does not exist or keeps sitting in the centre of the grid), the probability of route discovery success is 100%. The success rate in our experiment using AODV *basic*, is only $60.78 \pm 1\%$ (Row 2 of Table 2). The setting of the experiment guarantees that the RREQ reaches the destination A_{44} and that A_{44} will generate a route reply. It means that the route reply, which is unicast back via a previously established path gets lost. Since the experiment consists of a single

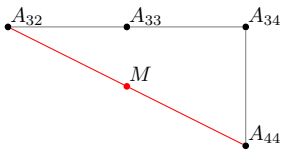


Fig. 4. Mobile node shortens distance

[6] We use a standard computer equipped with a 3.1 GHz Intel Pentium 5 CPU, 16 GB memory, running a Mac OS operating system. As SMC-tool, we use SMC-Uppaal, the Statistical extension of Uppaal (release 4.1.11) [3], which supports both timed and probabilistic systems. Timing aspects are heavily needed to model AODV (cf. Sect. 2); the topology-based mobility model relies on probabilistic choices to determine the movement of the node.

request only, RREP messages are not dropped and situations as the one sketched in Fig. 1 cannot occur. As a consequence, failure in route discovery means that a RREP message could not be unicast, which means that the established route from A_{44} to A_{11} uses the mobile node.

At first glance it seems to be impossible that 40% of all established routes use the mobile node as intermediate hop. But a closer analysis on time interval when a route for A_{11} is discovered shows that this is in fact the case since a route via a mobile node can shorten the distance between originator and destination. In general, AODV prefers shorter routes, hence it would choose the route via the mobile node M. Fig. 4 illustrates how a mobile node decreases the distance between A_{32} and A_{44} from 3 hops to 2. The lesson learned is that static nodes should be set up in a way that it is unlikely for a mobile node to shorten the distance, or static and mobile nodes should be distinguished and routes via static nodes only should be preferred, even if they are longer.

Observation 2 The model *dflag_res* does not yield much improvement w.r.t. route discovery compared to *basic* and is much worse than using *resend* alone.

The chance that a route is established by the first route discovery process is around 60% (cf. *basic*). In case no route is established (chance $\sim 40\%$), a new request is issued; the chance that this second request yields a route establishment between A_{11} and A_{44} is again 60%. Putting these numbers together the success rate for *dflag_res* should be $0.6 + 0.4 \cdot 0.6 \approx 0.84 = 84\%$. Surprisingly, the probability determined by our experiments is only around 70% in case of *dflag_res* (Row 7 and 8 of Table 2). That means that many RREP messages are lost (using the same reasoning as before, no RREQ message is lost). The explanation lies in the RREP-forwarding mechanism of AODV. As explained in Sect. 2, RREP messages are not forwarded if they do not contain new information. Let us now assume that the first RREQ reaches the destination A_{44}, which unicasts a RREP message to the next hop on the route back to A_{11}, say to node A_{34}. This reply gets lost afterwards. Since the resend-option is set, the originator issues another request, which also reaches A_{44}. In case the route to A_{11} is not changed in A_{44}'s routing table, A_{44} sends another RREP message to A_{34}. This message does not contain new information and is dropped by the intermediate node.

To repair this flaw, we change the RREP-generation procedure. Whenever a RREP message is generated, a counter (the sequence number), which indicates the freshness of the message is incremented.[7] This change is implemented in *dflag_res'*; the evaluation results for this model are now as expected.

Observation 3 AODV's option of intermediate route reply should be used.

Let us have a look at the models *resend* and *dflag_res'*. The difference between the two models is that in the former model intermediate nodes are allowed to reply. Looking at the results, we notice a dramatic difference in the likelihood of route discovery. In the model *resend* the second request is followed by the

[7] In fact AODVv2 and LOADng, the successor protocols of AODV (still under development), implement exactly this variant.

generation of more than one RREP message. In fact, each node that established a route to A_{44} during the first RREQ-RREP-cycle (before the reply was lost), will generate a RREP message. Due to this, route establishment is guaranteed. In contrast, there is only one RREP message for each and every request in *dflag_res'*. This observation clearly indicates that intermediate route reply is a useful feature. Interestingly, there seems to be the tendency of preferring protocols without this feature: the two successors of AODV, AODVv2 [17] and LOADng [6] follow this philosophy and set the D-flag as default—if at all, they allow intermediate route reply as an optional feature.

Other Originator Nodes.

The first set of experiments considered a route discovery from A_{11} to A_{44}, the largest distance a packet can travel in our set up. We expected to see the clearest results by using this distance. However, we also performed experiments with all other pairs of nodes. Fig. 5 summarises some results. It illustrates the probability of route-discovery (y-axis) depending on the distance between originator and destination (x-axis). Of course, the larger the distance between originator and destination, the smaller the chance of route establishment. Interestingly, there is a clear drop down at a distance of four nodes. It seems that from this point on resending guarantees the success. Moreover, the graph illustrates that exhaustive MC cannot help: MC is usually limited to topologies of up to 6 nodes, distances of 5 hops and more are not possible if one considers a non-linear topology.

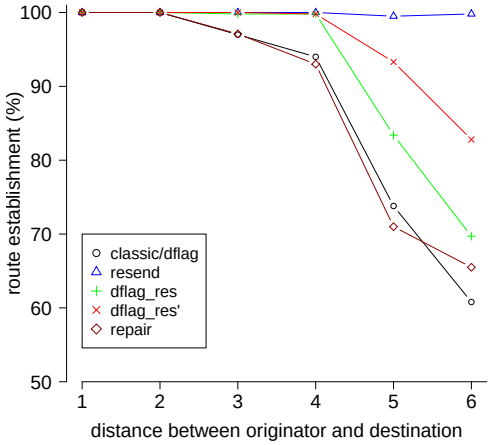


Fig. 5. Probability of route-establishment

4.2 Two Independent Route Discovery Processes

In order to evaluate the performance of variants of AODV under different network (traffic) load, we check the probability of route discovery when two route discovery processes are performed in parallel. For this second set of experiments, we are again interested in a route from A_{11} to A_{44}. However, shortly (35-45 milliseconds) after the packet is handed over to A_{11}, a second data packet is injected at another node, destined for some destination; in fact we did experiments for all destinations, but present only two observations—due to lack of space.

Observation 4 RREP messages are dropped more often than expected.

We consider the scenario where the second request is sent from A_{22} to A_{44}. Since some nodes drop RREP messages (cf. Sect. 2.2), the probability of route

distance between orig.	orginator of 2^{nd} request	class	probability (avg.)
1	$\{A_{12}, A_{21}\}$	nodes at border	43.36%
2	$\{A_{13}, A_{31}\}$	nodes at border	17, 75%
	$\{A_{22}\}$	inner node	13.28%
3	$\{A_{14}, A_{41}\}$	nodes at border	43, 10%
	$\{A_{23}, A_{32}\}$	inner nodes	29.74%
4	$\{A_{24}, A_{42}\}$	nodes at border	71, 61%
	$\{A_{33}\}$	inner node	47.29%
5	$\{A_{34}, A_{43}\}$	nodes at border	80.42%

Table 3. Two route discovery processes looking for the same destination A_{44}

establishment between A_{11} and A_{44} should decrease (compared to the 60% of Table 2). However, SMC-Uppaal shows that the probability of A_{11} finding a route to A_{44} is in the probability interval $[8.27913, 18.2791]$, i.e., a route discovery is unlikely. More results for the *basic* model are summarised in Table 3, grouped by the distance between the two originators. The table lists only the originator of the second route request; both the originator (A_{11}) of the first request and the destination (A_{44}) of both requests are fixed.

There is a correspondence between the success of route discovery and the distance between the two originators; moreover inner nodes have more influence on route discovery than nodes lying on the border of the network. This shows that the example of Fig. 1 occurs regularly. However, if the second originator $\texttt{oip}_2$ is far away from the first originator A_{11} no RREP message is dropped, since a route between $\texttt{oip}_2$ and A_{44} is established before the RREQ from A_{11} reaches $\texttt{oip}_2$. In the case of $\texttt{oip}_2 \in \{A_{24}, A_{42}, A_{34}, A_{43}\}$, the probability even increases. This is in line with Observation 3: when intermediate route reply is enabled, more RREP message are generated and the probability of route discovery success grows.

Observation 5 "Busy" mobile nodes increase the chance of route discovery.

One could rephrase this observation to "adding noise sometimes increases performance". At first glance it seems that adding additional network traffic—here a second route discovery processes—should not increase performance. But, let us look the scenario where the first data packet needs to be send from A_{11} to A_{44} (as before); the second packet is sent from A_{31} to the mobile node M. While handling the second RREQ most of the nodes will not learn about A_{44} and A_{11}. However it turns out that in the *basic* model, the probability of route discovery increases from around 60% to 72%. One reason is that the mobile node handles the request and generates a RREP message. While doing this it cannot handle the first RREQ; in case the first RREQ is sent to M and it is handling a different messages (is busy), the message is buffered. If the message is buffered for a while, the chance that the RREQ from A_{11} reaches A_{44} via a path without M as an intermediate hop, increases. Hence not the shortest, but the "fastest" route is established from A_{44} to A_{11}; this route is then used to send the RREP, since it does not use the mobile node as intermediate hop, the RREP is not lost.

model	**probability**$_{\text{fast}}$	**probability**$_{\text{moderate}}$	**probability**$_{\text{slow}}$
basic	[48.5230,58.5230]	[55.4336,65.4336]	[61.9377,71.9377]
resend	[94.8645,100.00]	[95.00,100.00]	[94.3225,100.00]
dflag	[50.0136,60.0136]	[54.4851,64.4851]	[63.2927,73.2927]
dflag_res	[60.1762,70.1762]	[64.5122,74.5122]	[70.6098,80.6098]
dflag_res'	[75.8943,85.8943]	[81.3144,91.3144]	[85.5149,95.5149]
repair	[54.0786,64.0786]	[57.6016,67.6016]	[65.5962,75.5962]

Table 4. different mobile node speed and impact on AODV variants

4.3 Influence of Speed of Mobile Nodes

In our experiments the topology changes within a time frame of 35 to 45 milliseconds; This also determines the speed of the mobile node. One might argue that the speed of the mobile node affects our analysis and that other speeds could yield different behaviour. As shown in Table 4, this is not the case—the probabilities slightly change, but stay in the same ball park. Moreover the relationship between the different variants stays the same; a variant that is more reliable with a fast mobile node, is also more reliable with a slower node. For this category of experiments we enforce a topology change within the interval $[25, 35]$ (fast), $[35, 45]$ (moderate), and $[95, 105]$ (slow), respectively.

5 Related Work

Model checking has been used to analyse routing protocols in general and AODV in particular. For example, Bhargavan et al. [1] were amongst the first to use model checking—they used the SPIN model checker—on a draft of AODV, demonstrating the feasibility and value of automated verification of routing protocols. Musuvathi et al. [15] introduced the CMC model checker primarily to search for coding errors in implementations of protocols written in C. They used AODV as an example and, as well as discovering a number of errors, they also found a problem with the specification itself, which has since been corrected. Chiyangwa and Kwiatkowska [4] used the timing features of UPPAAL to study the relationship between the timing parameters and the performance of route discovery. None of these studies performed a quantitative analysis of AODV.

Statistical model checking techniques [20,19] are rather new. So far they have been used in a couple of case studies. Bulychey et al. [2] for example apply the SMC-Uppaal to an analysis of an instance of the Lightweight Media access Control (LMAC) protocol; by this they are able to analyse ring topologies of up to 10 nodes.[8] Applications of SMC within biological systems are discussed in [5,13]. To the best of our knowledge, SMC was not used for the analysis of routing protocols so far—except in [11], where SMC-Uppaal is used to compare AODV and DYMO and to illustrate that even large topologies (up to 100 nodes) can be analysed by SMC. Our experiments are in line with this. However, it is unique in the sense that we carefully study variants of AODV.

[8] Other case studies include firewire, bluetooth, and a train gate (see `http://people.cs.aau.dk/~adavid/smc/cases.html` for an overview).

6 Conclusion and Future Work

The aim of this paper has been a careful (quantitative) analysis of AODV and its variants using statistical model checking techniques. By this, we have made surprising observations on the behaviour of AODV. We have shown for example that some optional features (D-flag) should not be combined with others (resending). Another result shows that a well-known shortcoming occurs more often than expected and has a tremendous effect on the success of route establishment. One challenge we faced while performing our experiments has been the interpreting the data.

The results were often surprising and hard to interpret, particularly when they indicate odd behaviour. Unfortunately SMC-Uppaal does not store traces during analysis, thus it is difficult to recover counterexamples to explain the observations. At the moment counter examples are constructed "by hand" by formulating more probing queries beyond looking at overall performance. This suggests that more powerful statistical analysis such as "rare event simulation" in combination with multiple queries could be used to compile better evidence.

Next to this careful analysis, we also showed that SMC is a suitable tool for analysing WMNs. In this setting classical MC was limited to topologies with up to 6 nodes and therefore having a realistic mobility model was not possible.

Future work will be a continuation of our case study. In particular we want to look at topologies of up to 100 nodes—it has been shown that an analysis of such networks is possible [11]. However, choosing the right scenario is crucial here: Since one cannot analyse all scenarios, one has to pick the right topologies and the right mobility model(s); but in some sense finding the correct setting becomes a "stab in the dark". We hope that our previous experience helps to set the experiments right.

Acknowledgement: We thank Ansgar Fehnker, Rob van Glabbeek and Franck Cassez for fruitful discussions and their help.
NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Bhargavan, K., Obradovic, D., Gunter, C.A.: Formal verification of standards for distance vector routing protocols. J. ACM 49(4), 538–576 (2002)
2. Bulychev, P., David, A., Larsen, K., Legay, A., Mikučionis, M., Bøgsted Poulson, D.: Checking and distributing statistical model checking. In: Goodloe, A., Person, S. (eds.) NASA Formal Methods. LNCS, vol. 7226, pp. 449–464. Springer (2012)
3. Bulychev, P., David, A., Larsen, K., Mikučionis, M., Bøgsted Poulson, D., Legay, A., Wang, Z.: UPPAAL-SMC: Statistical model checking for priced timed automata. In: Wiklicky, H., Massink, M. (eds.) Quantitative Aspects of Programming Languages. EPTCS, vol. 85, pp. 1–16. Open Publishing Association (2012)
4. Chiyangwa, S., Kwiatkowska, M.: A timing analysis of AODV. In: Formal Methods for Open Object-based Distributed Systems (FMOODS'05). LNCS, vol. 3535, pp. 306–321. Springer (2005)

5. Clarke, E.M., Faeder, J., Langmead, C., Harris, L., Jha, S., Legay, A.: Statistical model checking in biolab: Applications to the automated analysis of t-cell receptor signaling pathway. In: Heiner, M., Uhrmacher, A. (eds.) Computational Methods in Systems Biology (CMSB'08). pp. 231–250. No. 5307 in LNCS, Springer (2008)
6. Clausen, T., Colin de Verdiére, A., Yi, J., Niktash, A., Igarashi, Y., Satoh, H., Herberg, U., Lavenu, C., Lys, T., Perkins, C., Dean, J.: The lightweight on-demand ad hoc distance-vector routing protocol - next generation (LOADng). Internet Draft (Standards Track) (2013), http://tools.ietf.org/html/draft-clausen-lln-loadng-08
7. Fehnker, A., van Glabbeek, R.J., Höfner, P., McIver, A., Portmann, M., Tan, W.L.: Automated analysis of AODV using UPPAAL. In: Flanagan, C., König, B. (eds.) Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12). LNCS, vol. 7214, pp. 173–187. Springer (2012)
8. Fehnker, A., van Glabbeek, R.J., Höfner, P., McIver, A., Portmann, M., Tan, W.L.: A process algebra for wireless mesh networks used for modelling, verifying and analysing AODV. Tech. Rep. 5513, NICTA (2013), http://www.nicta.com.au/pub?id=5513
9. Fehnker, A., Höfner, P., Kamali, M., Mehta, V.: Topology-based mobility models for wireless networks, (submitted) http://www.hoefner-online.de/formats2013/topology.pdf
10. Höfner, P., van Glabbeek, R., Tan, W., Portmann, M., McIver, A., Fehnker, A.: A rigorous analysis of AODV and its variants. In: Modeling, analysis and simulation of wireless and mobile systems. pp. 203–212. MSWiM '12, ACM (2012)
11. Höfner, P., McIver, A.: Statistical model checking of wireless mesh routing protocols. In: NASA Formal Methodsds. LNCS, vol. 7871, pp. 322–336. Springer (2013)
12. IEEE P802.11s: IEEE draft standard for information technology—telecommunications and information exchange between systems—local and metropolitan area networks—specific requirements—part 11: Wireless LAN Medium Access Control (MAC) and physical layer (PHY) specifications-amendment 10: Mesh networking (July 2010)
13. Jha, S., Clarke, E., Langmead, C., Legay, A., Platzer, A., Zuliani, P.: A bayesian approach to model checking biological systems. In: Computational Methods in Systems Biology (CMSB'09). pp. 218–234. No. 5688 in LNCS, Springer (2009)
14. Maler, O., Larsen, K.G., Krogh, B.H.: On zone-based analysis of duration probabilistic automata. In: Chen, Y.F., Rezine, A. (eds.) Verification of Infinite-State Systems. EPTCS, vol. 39, pp. 33–46. Open Publishing Association (2010)
15. Musuvathi, M., Park, D.Y.W., Chou, A., Engler, D.R., Dill, D.L.: CMC: a pragmatic approach to model checking real code. In: Operating Systems Design and Implementation (OSDI'02) (2002)
16. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-demand distance vector (AODV) routing. RFC 3561 (Experimental) (2003), http://www.ietf.org/rfc/rfc3561
17. Perkins, C., Chakeres, I.: Dynamic MANET on-demand (AODVv2) routing. Internet Draft (Standards Track) (2013), http://tools.ietf.org/html/draft-ietf-manet-dymo-25
18. Perkins, C., Royer, E.: Ad-hoc On-Demand Distance Vector Routing. In: 2nd IEEE Workshop on Mobile Computing Systems and Applications. pp. 90–100 (1999)
19. Sen, K., Viswanathan, M., Agha, G.A.: Vesta: A statistical model-checker and analyzer for probabilistic systems. In: Quantitative Evaluaiton of Systems (QEST'05). pp. 251–252. IEEE (2005)
20. Younes, H.: Verification and Planning for Stochastic Processes with Asynchronous Events. Ph.D. thesis, Carnegie Mellon University (2004)