Statistical Model Checking of Wireless Mesh Routing Protocols

Peter Höfner[1,3] and Annabelle McIver[2,1]

[1] NICTA
[2] Department of Computing, Macquarie University
[3] Computer Science and Engineering, University of New South Wales

Abstract. Several case studies indicate that model checking is limited in the analysis of mesh networks: state space explosion restricts applicability to at most 10 node networks, and quantitative reasoning, often sufficient for network evaluation, is not possible. Both deficiencies can be overcome to some extent by the use of statistical model checkers, such as SMC-Uppaal. In this paper we illustrate this by a quantitative analysis of two well-known routing protocols for wireless mesh networks, namely AODV and DYMO. Moreover, we push the limits and show that this technology is capable of analysing networks of up to 100 nodes.

1 Introduction

Wireless Mesh Networks (WMNs) are self-organising ad-hoc networks that support broadband communication without relying on a wired backhaul infrastructure. They have gained popularity through their flexibility which allows them to be used in a diverse range of applications, from emergency response to transportation systems. Automatic route-discovery, maintenance and repair play a fundamental role in reliability and performance of such networks where typical scenarios include dynamic topologies. The engineering challenge is to design protocols which facilitate good service in spite of these harsh operating conditions.

Traditional approaches to the analysis of WMN protocols are simulation and test-bed experiments. While these are important evaluation methods they are typically used for testing implementations rather than design specifications. Moreover, the analysis is restricted to global properties such as overall throughput or message delay. Formal analysis of specifications is one way to systematically screen protocols for flaws and to present counterexamples to diagnose them. It has been used in locating problems in automatic route-finding protocols [2, 9].

Unfortunately, current state-of-the art model checkers are unable to handle protocols of the complexity needed for WMN routing in realistic settings. In previous work [8] we used the model checker Uppaal to analyse basic qualitative properties of the Ad hoc On-Demand Distance Vector (AODV) routing protocol, one of four protocols currently standardised by the IETF MANET working group. We were able to analyse systematically *all* network topologies of up to five nodes. Although this provides a partial analysis, as does simulation, the

network sizes are far from realistic and quantitative information such as probabilities were not included. In this paper we investigate whether *statistical model checking* can combine the systematic methodology of "classical" model checking with the ability to analyse quantitative properties and realistic scenarios.

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 currently prevents useful analysis of large models. SMC trades certainty for approximation, using Monte Carlo style sampling, and hypothesis testing to interpret the results. We are interested in timed systems and so we use SMC-Uppaal, the Statistical extension of Uppaal (release 4.1.11) [4], which supports the composition of timed and/or probabilistic automata. The sampling is carried out according to the probability distribution defined by the 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 the theoretical Chernoff-Hoeffding bounds ($O\big(\frac{1}{\varepsilon^2}\ln\frac{2}{\alpha}\big)$), which crucially is independent of the size of the model. SMC-Uppaal generates an interval $[p-\varepsilon, p+\varepsilon]$ for estimating p, the probability of CTL-property ψ holding w.r.t. the underlying probability distribution.

In this paper we model two routing protocols for WMNs: AODV and DYMO (Dynamic MANET On-demand).[1] One aim is to understand the role of the different design choices via a number of performance and correctness measures. We analyse the performance, both over a complete set of topologies for small networks as well as for medium-to-large network sizes. Since the complexity and size of these protocols go far beyond what can be analysed with standard model checking, these case studies provide excellent test bases for demonstrating the power and capacity of the new statistical tools. We illustrate here the range and depth of the analysis which is achievable with statistical analysis, which we believe is currently not possible using traditional simulation alone.

In Sect. 2, we give an informal summary of routing, followed by a description of our Uppaal models, concentrating particularly on timing aspects. Four categories of experiments are discussed in Sect. 3. The first presents a timing analysis of AODV; the second and third provide a thorough comparison of AODV against DYMO both w.r.t. overall performance and quality of the routes discovered, where we find some surprising trends. Finally we demonstrate that this analysis is scalable, illustrated by redoing a selection of experiments for networks consisting of up to 100 nodes. In Sect. 4, we review related work and, in Sect. 5, we reflect on the challenges ahead for SMC.

2 Routing Protocols and their Architecture

On demand routing protocols such as AODV and DYMO are designed to establish routes only when needed, typically when a new data packet is injected by

[1] Since March 2012, DYMO is sometimes referred to as AODVv2.

a user (application layer). Each node maintains its own routing table thereby enabling it to act as its own router. Routing tables can be updated whenever new messages are handled, since incoming messages carry a wealth of information concerning network connectivity simply because they have just successfully travelled from somewhere. Nodes mine that information in different ways, which, as our analysis shows, yields different behavioural profiles.

The collective information in the nodes' routing tables is at best a partial representation of network connectivity as it was sometime in the past; in the most general scenarios mobility continually modifies that representation. Nodes following either AODV or DYMO store information about a route towards a possible destination `d` (if a route has been discovered) as follows. The total number of hops in the route (`hops`), the identity of the very next hop in the route (`nhop`), a "destination sequence number" (`dsn`) (a measure of the freshness of the entry), and a "validity flag" (`flag`),[2] which is unset whenever information arrives indicating that one of the downstream links in the route is broken. Whilst currently our analysis only looks at static topologies we nevertheless find that these protocols do not always perform as we would expect.

2.1 Basic Architecture for Ad Hoc Routing

AODV and DYMO follow the same basic architecture. Each node maintains a message queue to store incoming messages and a processor for handling messages. Whilst the queue is always enabled to receive messages, message handling can take time and so communication between queue and handler occurs only when the handler has successfully processed a message. The workflow of the handler is as follows: first, the next (oldest) message is loaded from its message queue. Depending on the type of message (see below) the routing table is updated and, if necessary, a new message is created, and either broadcast or unicast.

The AODV Architecture. Each node maintains its own destination sequence number, routing table and keeps a record of the messages it has already received (or initiated). It also manages a queue to store data packets waiting to be delivered. Messages are handled appropriately according to their type:

`PKT` Messages containing data packets play no part in route-finding. In the case that a node has a valid route for the `PKT`'s destination, the packet is forwarded to `nhop`, the next hop on the route. In the case that the data packet is injected by the application layer and no (valid) route is known, the packet is placed on the node's packet queue, and a route discovery process is initiated by broadcasting an appropriate `RREQ` message.

`RREQ` Route requests are messages, broadcast to every node within transmission range. They contain information about the originator of the route discovery process, the neighbour that most recently sent it, and the number of nodes through which the request travelled. All of this information is available for updating routing tables. The same request can be received via

[2] AODV calls it *Valid Destination Sequence Number flag*; DYMO *Route.Broken*.

different routes and so nodes maintain a record of those that have already been handled so that duplicates can be discarded. For new requests, the following actions are taken. (a) If the node is either the destination or has a valid route to the destination stored in its routing table, a route reply (`RREP`) message is generated, which is unicast back to most recent sender. (b) If the node is neither the destination nor has any information about the destination, it increments the hop count and broadcasts it on.

`RREP` Replies are "logically" matched up with the corresponding request that gave rise to it so that a route for the requested destination can be established. The routing table is updated for that destination, by recording `nhop` as the neighbour from which the `RREP` was received and similarly taking `hops` and `dsn` from the `RREP`. Only if the routing table was changed during the update, the hop count of the `RREP` is increased and then (in the case that the node was not the original initiator) forwarded to the neighbour from which the corresponding request was received.

`RERR` Error messages are generated whenever link breaks are detected by some nodes. Often this occurs when a message (`RREP` or `PKT`) fails to be sent. In these cases an error message is sent to all neighbours. If an `RERR` message is received the routing tables are updated—in particular routes are marked as *invalid*, and the error message is forwarded to all neighbours.

This informal introduction to AODV should be sufficient to understand the experiments described below. A detailed description can be found in [14].

The DYMO Architecture. DYMO [15] follows the same basic workflow as AODV. In this section we only highlight the major design differences.[3]

(a) DYMO's mechanism for managing duplicate requests is no longer based on a queue of handled `RREQ` messages. Instead DYMO uses sequence numbers to judge whether information contained in a message should be forwarded. While this modification saves some memory, it has been shown that the change can lead to loss of route requests [6].

(b) On the other hand AODV can lose route replies since `RREP` messages are only forwarded if the routing table of an intermediate node is updated (changed).[4] To avoid this, a node generating a route reply increments the sequence number for the destination, thereby guaranting that the routing table of nodes receiving the `RREP` message will be updated, and the `RREP` forwarded.

(c) DYMO establishes *bidirectional* routes between originator and destination. When an intermediate node initiates a route reply, it unicasts a message back to the originator of the request (as AODV does), but at the same time it forwards a route reply to the intended destination of the route request.

(d) DYMO uses the concept of *path accumulation*: whenever a control message (`RREQ`, `RREP`, `RERR`) travels via more than one node, information about *all* intermediate nodes is stored in the message. In this way, a node receiving a message establishes routes to *all other intermediate nodes*. In AODV nodes only establish routes to a the initiator and to the sender of a message.

[3] Our model is based on DYMO's internet draft version 22.

[4] `http://www.ietf.org/mail-archive/web/manet/current/msg05702.html`

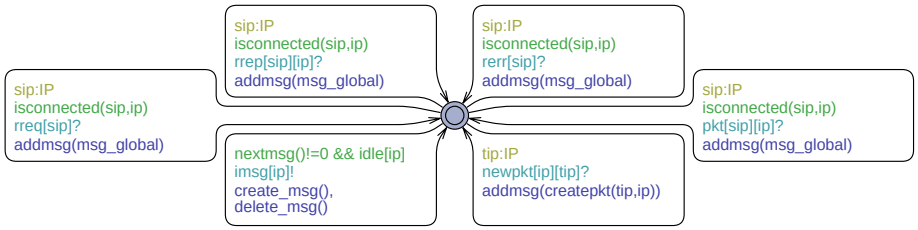


Fig. 1: Automaton modelling the `Queue`.

These changes imply (as intended) quite different behaviour: for example (c) and (d) might mean that DYMO establishes many more routes in the network as a whole than does AODV. On the other hand (a) could imply that some routes might not be discovered at all. We investigate some of these differences below.

2.2 AODV and DYMO in Uppaal

In previous work [8], an untimed Uppaal model of AODV was developed and used to analyse some basic qualitative properties. In this paper we extend that analysis to quantitative properties combining time and probability. As a consequence the models needed a significant redesign to include timing constraints on sending messages between nodes, as well as redesigning communication between nodes so that the unicast behaviour of DYMO and AODV was correctly rendered using SMC-Uppaal's (only) broadcast mechanism.

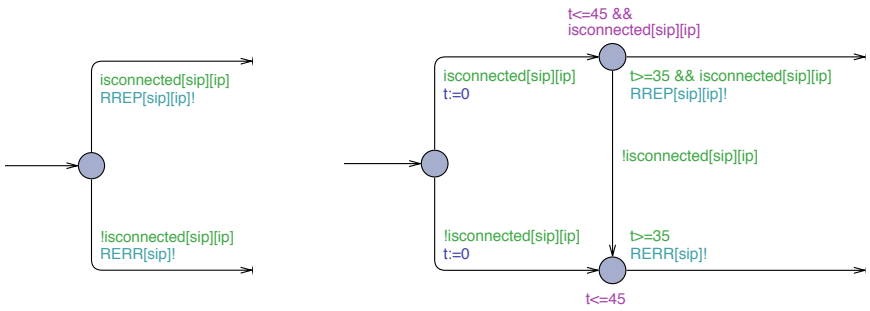
We model AODV and DYMO as a parallel composition between node processes, where each process is a parallel composition of two timed automata, the `Handler` and the `Queue`. Communication between nodes `i` and `j` is only feasible if they are in transmission range of each other. This is modelled by predicates of the form `isconnected[i][j]`, which is true if and only if `i` and `j` can communicate. Communication between different nodes `i`, `j` are on channels named according to the type of message being delivered (`rerr`, `rrep`, `rreq`).

The `Queue` of a node `ip` is depicted in Fig. 1. Messages (arriving from other nodes) are stored in a queue, by using the function `addmsg`. Our model guarantees that messages sent by nodes within transmission range are received.

The `Handler`, modelling the message-handling protocol, is far more complicated and has around 20 locations. It is busy while sending messages, and can only accept a new message from the `Queue` once it has completely finished handling a message. Whenever it is not processing a message and there are messages stored in the `Queue`, the `Queue` and the `Handler` synchronise via channel `imsg[ip]`, transferring the relevant message data from the `Queue` to the `Handler`. The `Handler` then follows the workflow sketched in Sect. 2.1. Due to lack of space, we cannot present the full timed automaton modelling the `Handler`, but it is available in full online[5]. Here, we concentrate on our treatment of time.

According to the specification of AODV [14], the most time consuming activity is the communication between nodes, which takes on average 40 milliseconds.

[5] `http://www.hoefner-online.de/nfm2013/`



(a) untimed model (b) timed model

Fig. 2: Part of `Handler`—unicast a message.

In Fig. 2 we compare the extract of a model without time (as used in [8]) with the corresponding extract including time. For the untimed model we simply guard the communication with `isconnected` so that the message (here a `rrep`) is sent whenever the nodes are connected, and an error message is generated otherwise. In the timed model, we use a clock variable `t`, set to `0` before transmission, and then we use an intermediate location which has the effect of selecting a delay of at least 35 milliseconds and no more than 45 milliseconds uniformly at random. In the case that the nodes are still connected at the time of sending then the `rrep` message is successfully transmitted, and otherwise an error is reported.[6]

3 Experiments

The experiments split into four categories: a timing analysis of AODV (Sect. 3.1); a comparison between AODV and DYMO (Sect. 3.2); a quantitative analysis of the two protocols (Sect. 3.3); and a feasibility study of networks of realistic size (Sect. 3.4). The experiments of the first three categories use the following setup: 3.1 GHz Intel Pentium 5 CPU, with 16 GB memory, running the Mac OS X 10.7 operating system. The final category needs 128 GB memory (3.3 GHz). For all experiments we use SMC-Uppaal 4.1.11 (June 2012). In the first three categories, the parameters of false negatives (α) and probabilistic uncertainty (ε) are both set to 0.01—yielding a confidence level of 99% and SMC-Uppaal checks 26492 runs (cf. Chernoff-Hoeffding bound). The last category uses, due to its calculational complexity, only 738 runs and a confidence level of 95% ($\alpha = \varepsilon = 0.05$).

3.1 A Timing Analysis of AODV

The first category extends experiments performed for the untimed model for AODV [8], exploring in more depth the surprising result that AODV might fail to discover an existing route in 47% of all network topologies with up to 5 nodes.

[6] This complexity needs to be inserted because a change in connectivity could result in nodes being connected at the start of transmission, but become disconnected before the transmission is completed.

For the experiments we generate all topologies of up to 5 nodes, where for each topology we consider three distinct nodes A, B and C; each with particular originator/destination roles as per scenario described below. Up to symmetry this yields 444 topologies. For each scenario we analyse three properties; in total this requires approximately 4000 experiments for this category.

Initially, for each scenario no routes are known. Then, with a time gap of 35–45 milliseconds, two of the distinct nodes receive a data packet and have to find routes to the packets' destinations. The scenarios assign roles as follows:
(i) A is the only originator sending a packet first to B and afterwards to C;
(ii) B and C are originators both sending to A;
(iii) A is sending to B first and then B is also an originator sending to C;
(iv) B is an originator sending to C followed by A sending to B.
For each scenario we analyse two properties and their combination. The first property examines the time taken for the protocol to complete, i.e., until all messages have been handled, which encoded in Uppaal's syntax as

```
Pr[<=10000](<> (tester.final && emptybuffers()))
```
(1)

This query asks for the probability estimate (`Pr`) satisfying the CTL-path expression `<>(tester.final && emptybuffers())` within `10000` time units (milliseconds); 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. `tester` refers to a process which injects the data packets to the originators (`tester.final` means that all data packets have been injected), and the function `emptybuffers()` checks whether the nodes' message queues are empty.

The second property examines the time for requested routes to be established. This differs from (1) since routes are usually found before all buffers are emptied.

```
Pr[<=10000](<> (OIP1.rt[DIP1].nhop!=0 && OIP2.rt[DIP2].nhop!=0))
```
(2)

Here, `o.rt[d].nhop` is the next hop in `o`'s routing table entry for destination `d`. As soon as this value is set (is different to `0`), a route to `d` has been established.

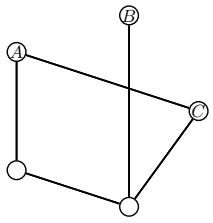
The third property combines the first two and analyses the time which is needed to finish the protocol and to establish the routes; this estimates the proportion of runs which end without ever finding a route.

```
Pr[<=10000](<> (tester.final && emptybuffers() &&
 OIP1.rt[DIP1].nhop!=0 && OIP2.rt[DIP2].nhop!=0))
```
(3)

For every scenario, SMC-Uppaal evaluates the property under consideration for 26492 runs and returns a probability interval $[p-0.01, p+0.01]$, where p is the averaged probability over all runs. Probability theory implies that with a likelihood of 99% the "real" value is inside this interval.

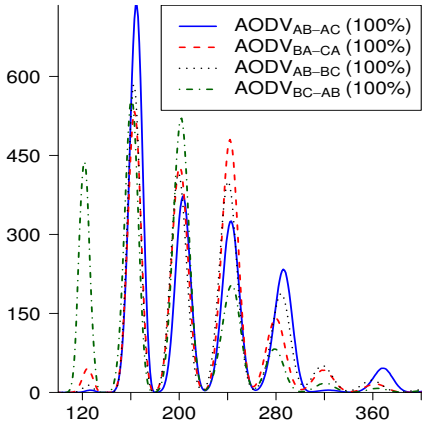
Fig. 3 displays the results for all 5-node networks.[7] The x-axis represents the time (in milliseconds) for the property to be satisfied; the y-axis represents the average number of simulation runs per topology for the property to be satisfied.

[7] The graphs for network sizes 3 and 4 look similar and can be found at `http://www.hoefner-online.de/nfm2013/`

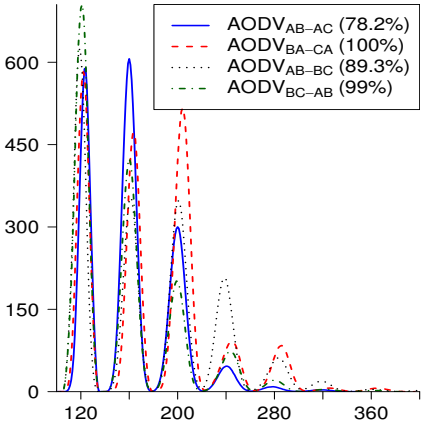


Four scenarios:
(i) A searches for B, then A for C;
(ii) B searches for A, then C for A;
(iii) A searches for B, then B for C;
(iv) B searches for C, then A for B

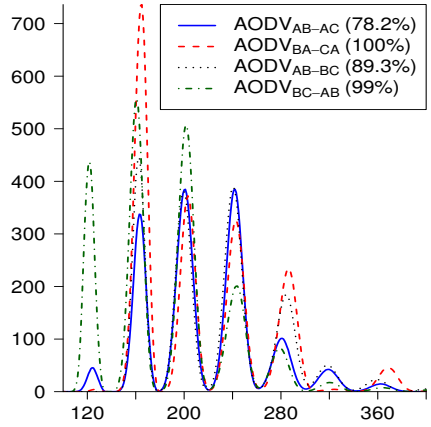
(a) example topology and scenarios



(b) running time of AODV



(c) route establishment



(d) combined properties

Fig. 3: A timed analysis of AODV (5-node topologies).[9]

For example the highest peak in (b) shows that 720 simulation runs (out of 26492) need 164 ms to finish the protocol. Figs. (b–d) refer to Properties (1), (2) and (3). Each graph depicts the results for each scenario. For example the solid graph corresponds to the the first scenario (A is the originator and B and C are the destinations). The overall probability that the property is satisfied is indicated by the percentage given in the legend.[8]

Analysis of the results. All of the experiments yield a periodic behaviour of roughly 40 milliseconds corresponding to the average time for sending a message. Surprising is that the performance of AODV is fairly stable across scenarios.

[8] More precisely, the probability shown is the average of all medians of the probability intervals returned by Uppaal.

[9] Figures 3, 4 and 7 have been produced using the tool `R` [16].

Fig. 3(b) shows that AODV always terminates and presents the running times for termination. Fig. 3(c) shows that in general route establishment occurs much earlier; the results also show that AODV cannot always establish routes: in the case of 3-node topologies routes are not established in 11.7% of all cases; for networks with 4 nodes in 10.85% of all cases; and in case of 5 nodes in approximately 10% of all cases.

Fig. 3(d) confirms that the quantitative analysis gives significantly more insight than an untimed analysis such as reported in [8]. There, we considered a similar property and found that in 47.3% of all 5-node topologies there is the possibility of route-discovery failure—a quantitative analysis was not possible. Our quantitative analysis shows that failure to find a route can now be estimated at around 10%. There are two reasons for this dramatic difference. First, the inclusion of time ruled out some scenarios where route failure was due to messages overtaking each other. Second, and more significant, the new analysis determines the number of runs (not the number of topologies), where route discovery fails and indicates that discovery failures are rare: whereas in half of the topologies route failure is possible [8], in only ∼10% of all runs failure actually happens.

3.2 AODV versus DYMO

In Sect. 2 we have outlined the design differences between AODV and DYMO. Moreover, we have speculated on what those differences might imply w.r.t. overall performance. We now run exactly the same experiments as described in Sect. 3.1, this time for DYMO. The results averaged over all 4-node networks and all scenarios for both routing protocols are presented in Fig. 4; in these diagrams we also indicate the average times by vertical bars.

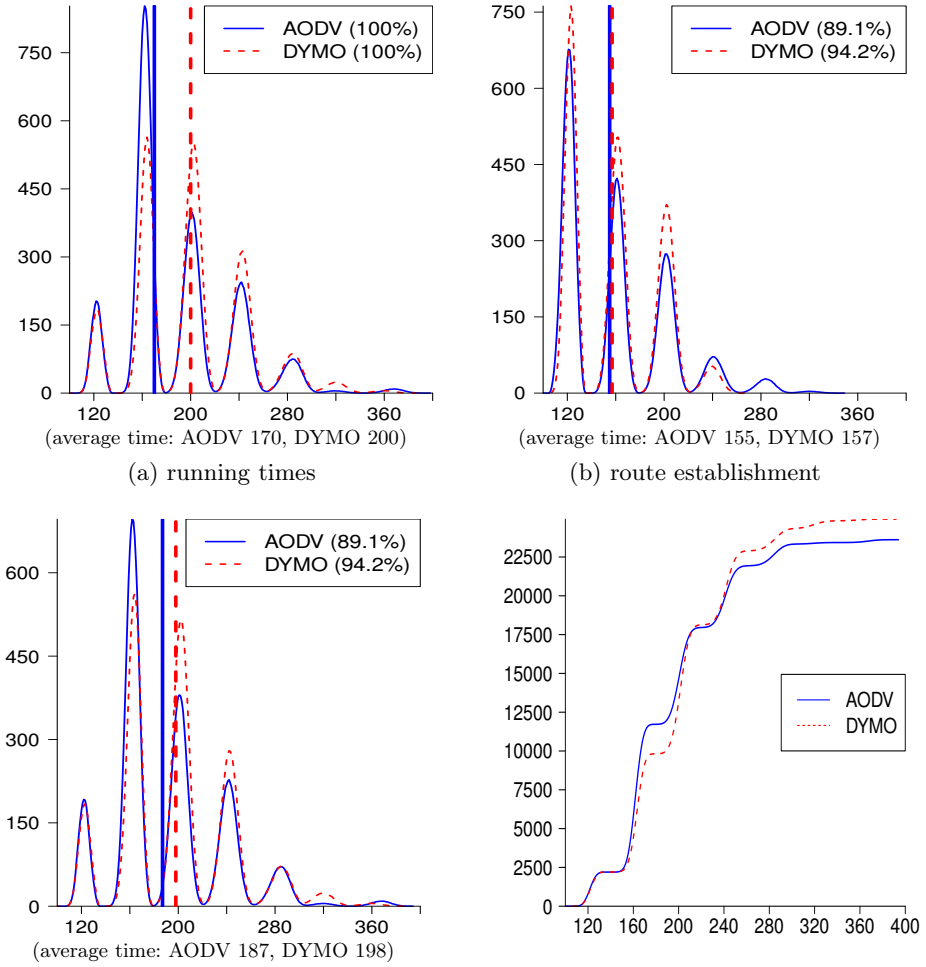
To our surprise, the variation in performance between the two is marginal: DYMO appears to be more reliable in that it can establish more routes than does AODV in some cases (Fig. 4(b,c)). DYMO takes on average longer to complete (Fig. 4(a)) but the average time to find routes is almost exactly the same as for AODV (Fig. 4(b)).

A first analysis of the circumstances behind the observed non-establishment of routes in DYMO is presented in [6], indicating that problems occur when messages can overtake others. The reason why DYMO needs longer running times is the additional `RREP`-message sent to the destination of a route request (cf. Page 4).

3.3 Quantitative Measurements

So far we have looked at running times and route discovery. In this section we illustrate how to the use Value-Estimation-Feature (`E`) of SMC-Uppaal to explore the quality and quantity of the routes established by AODV and DYMO.

One side effect of broadcasting route requests is that intermediate nodes, which handle those requests, are able to establish routes to the originator. Whilst this certainly represents an increase in "knowledge" across the network, there is no guarantee that the routes established are optimal. In [13] it is shown that non-optimal paths can impact overall performance of packet delivery dramatically.



(average time: AODV 170, DYMO 200)

(a) running times

(average time: AODV 155, DYMO 157)

(b) route establishment

(average time: AODV 187, DYMO 198)

(c) combined properties

(d) combined Property (cumulative)

Fig. 4: AODV vs. DYMO (4-node topologies)

We examine two properties: the total number of routes established over all routing tables, averaged over all network topologies for up to 5 nodes; and the average difference between the length of the route established and the length of the optimal route.

Route Quantity. Routing tables are updated whenever control messages are received. In case of `RREQ` and `RREP` messages, AODV does so only for the originator/destination and for the sender of the message, whereas DYMO uses path accumulation (cf. Page 4). This difference in design implies that DYMO could potentially establish more routes than AODV. We check whether this is indeed the case for all topologies and all scenarios described earlier using the property

```
E[<=10000;26492](max:total_knowledge())
```
(4)

Here, the function `total_knowledge()` counts the number of entries in all routing tables along a run (path); `max` takes the largest of these values. Since value estimation does not determine the number of runs, we set it to the same number as determined previously (26492); the time bound is again set to 10000.

	3 nodes	4 nodes	5 nodes
AODV	5.28	8.83	13.99
DYMO	5.25	7.87	11.94
max	6	12	20

Table 1: Average number of routes found

Table 1 presents the results, grouped by network size. Note that the last row shows the maximal number of possible routing table entries: this is $n \cdot (n-1)$ since each node can hold $n-1$ entries in an n-node network.

To our surprise, DYMO establishes fewer routes on average than does AODV. (Although it does establish more of the *requested* routes Fig. 4.) A possible explanation is the following: when DYMO floods the network with the first `RREQ`, many nodes establish many routes (more than with AODV), due to path accumulation. When the second `RREQ` is sent, the chance of an intermediate route reply is now greater (than for AODV)—an intermediate route reply means that the `RREQ` is not forwarded, thus additional opportunities to create routes in receiving nodes are suppressed.[10]

Route Quality. In almost all routing protocols based on `RREQ`-broadcast, non-optimal routes can be established [13]. This can happen when the destination does not forward the `RREQ` message, as the example in Fig. 5 shows. The scenario depicts node S searching for a route to node T. As soon as T receives the `RREQ` message, it generates a route reply, and suppresses the `RREQ`. Node A receives the same `RREQ` via B and establishes a non-optimal path to S via B.

In our second experiment we check the extent of establishing non-optimal routes. We use the query `E[<=10000;26492](max:quality())`, which is similar to (4), but instead uses a function `quality` that compares the length of established routes with the length of the corresponding optimal routes.[11]

The results in Table 2 show that the average deviation from the optimal length (in ‰) is small; which is to be expected in small networks. More interesting is that again DYMO performs less well than AODV. Again a potential explanation for this is the implication of path accumulation in DYMO. In the example, node A establishes a (non-optimal) route to S, but because of path accumulation node A will also establish a non-optimal path to B (as well as all the other nodes on this non-optimal path).

3.4 Networks of Realistic Size

In complex protocols used for routing, analysis by "classical" model checking is limited to around 8 nodes. WMNs usually consist of more than 50 nodes placing them far beyond the capabilities of systematic logical analysis. In this section we explore the scalability of SMC for such networks.

[10] An example is found at the website—it requires detailed knowledge of the protocols.
[11] The length of optimal routes can be calculated from the static network topology.

	3 nodes	4 nodes	5 nodes
AODV	0.00‰	0.50‰	2.31‰
DYMO	0.00‰	2.00‰	9.68‰

Table 2: Average deviation from the optimal

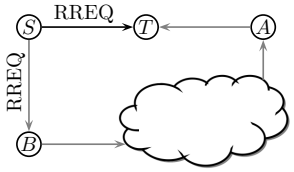


Fig. 5: Node A "accidentally" creates a non-optimal route to S.

Our first task is to generate a sample realistic topology. We use the *Node Placement Algorithm for Realistic Topologies* (NPART) [12]. This tool allows the specification of arbitrary-sized topologies and transmission ranges, and it has been shown that generated topologies have graph characteristics similar to realistic wireless multihop ones.

We analyse NPART topologies consisting of 25, 50, 75 and 100 nodes. Fig. 6 depicts the 100-node topology used for our analysis. The links between nodes are determined by the distance between nodes; rather than displaying the actual 201 links, we instead indicate the link distance by scale. The node labelled A is the originator of two packets with destinations B and C, both of which are connected to A albeit at several hops distance. We check Property (3), which confirms that both routes are found and that the protocols terminates. More significant are the resources required to perform the experiments for large networks which we report next.

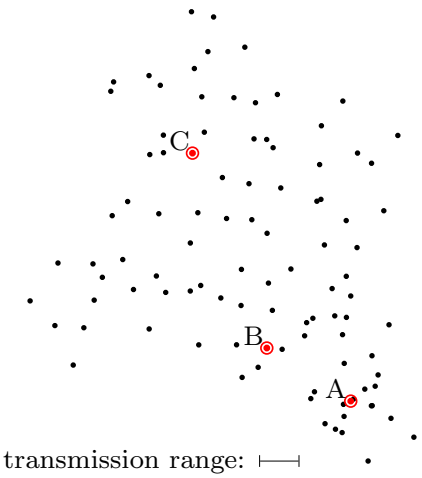


Fig. 6: A topology with 100 nodes

A network with 25 nodes is easily checked with a standard desktop machine in less than half an hour with a confidence level of 95% (which means 738 runs). However the memory consumption grows with the number of nodes. A summary of our observations is given in Table 3.

#nodes	50	75	100
memory (Gb)	14	30	80
run time (m)	270	328	1777

Table 3: Memory Consumption[12]

Fig. 7 shows that the protocol finishes on average within $24 \times 40\,ms$[13] for the given scenario. This also suggests that there is little interference between the

[13] The average time for sending a single message.

two requests for B and C, since the number of hops between A and C is roughly 10, and so at least 20 messages are required to establish that route alone.

4 Related work

Traditionally, protocols for WMNs are evaluated using test-bed experiments and simulation, e.g., [10]. Test-bed experiments evaluate protocols under realistic circumstances, whereas simulation is performed on a single machine, thus is closely related to our work. Simulation-based studies show that AODV performs better than DYMO in some scenarios and vice versa in others [1, 11]. Under packet delivery ratio (PDR) as measure Saleem et al. [17] imply that DYMO compares unfavourably to AODV (consistent with our results), but this analysis does not help to diagnose the reasons for this conclusion.

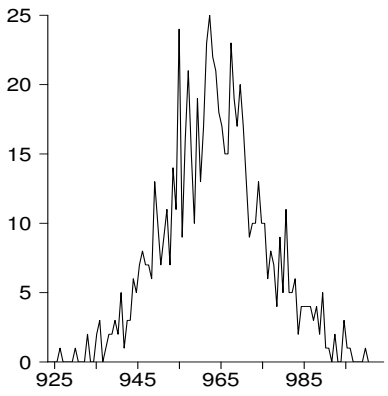


Fig. 7: Property (3) with 100 nodes

More recently formal analysis has been used to investigate the behaviour of complex protocols [2, 5]. Although formal analysis is often more detailed than test bed analysis, with the result that only small samples can be investigated, the outcome is often a more penetrating understanding of protocol behaviour. For example a study using the Spin model checker showed that an early draft of AODV could create routing loops [2]; Zave [21] uses the Alloy analyser in combination with the Spin model checker to show that no published version of the Chord ring-maintenance protocol is correct, and Schuts et al. establish an impossibility result for clock synchronisation in the Chess gMAC WSN protocol [18]. Other specific formal analyses of AODV include that of Chiyangwa and Kwiatkowska [5] who investigate the relation between protocol parameters and performance, such as time outs in AODV, and Espensen et al. [7] use coloured Petri nets to perform test runs to confirm specified behaviour.

5 Conclusion and Outlook

Our aim in this study was twofold: (i) We developed timed models for AODV and DYMO in order to carry out a systematic analysis across all small networks. In comparison to simulation and test bed studies, our analysis based on quality and quantity enabled us to examine reasons for observed differences in performance between AODV and DYMO, which was an open question before (cf. studies in [1, 11]). (ii) We examined the feasibility of SMC w.r.t. scalability. None of the formal studies above analysed routing protocols for networks containing more than 10 nodes, whereas our results imply that networks of realistic size can be analysed. Finally we draw some general conclusions about SMC critical analysis.

5.1 Statistical Model Checking: Lessons Learned

Resourcing. One of the main bottlenecks in the analysis was time—to analyse a 100 node network takes about 30 hours. One of the next steps is to determine whether the most recent distributed release [3] is able to reduce that overhead.

Choosing the right scenario. For small networks it is possible to analyse all topologies for given scenarios. This gives a good overall view of the performance and behaviour in any situation. For large networks this is not feasible, and so the selection of topologies in combination with the right scenarios becomes something of a "stab in the dark".For our study we used the comparison of AODV and DYMO to observe that odd behaviour occur in the setting of two requests, thus we chose that scenario for our large networks. In general, a systematic analysis of small networks can be used as a preliminary phase for selecting the most informative scenarios.

Interpreting the results. The results are frequently hard to interpret, particularly when they indicate odd behaviour. Unfortunately SMC-Uppaal does not store traces during analysis, thus it is not possible to recover counterexamples to explain the observations. We tried to diagnose odd observations 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.

5.2 Future work

The models for AODV and DYMO are general enough to allow for the study of more complex scenarios, in particular mobility. In future work we will develop a number of mobility models for understanding the behaviour of these and other routing protocols.

Acknowledgement. We are grateful to David Jansen and Frits Vaandrager for helpful discussions, and to the NWO grant 040.11.302 for financial support. 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. Amin, M., Abrar, M., Khan, Z.U., Andusalam, Rizwan, S.: Comparison of OLSR & DYMO routing protocols on the basis of different performance metrics in mobile ad-hoc networks. American Journal of Scientific Research (2011)
2. Bhargavan, K., Obradovic, D., Gunter, C.: Formal verification of standards for distance vector routing protocols. J. ACM 49(4), 538–576 (2002)
3. Bulychev, P., David, A., Larsen, K., Legay, A., Mikučionis, M., Bøgsted Poulsen, D.: Checking and distributing statistical model checking. In: Goodloe, A., Person, S. (eds.) NASA Formal Methods (NFM'12). LNCS, vol. 7226, pp. 449–464. Springer (2012)

4. 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)
5. Chiyangwa, S., Kwiatkowska, M.: A timing analysis of AODV. In: Formal Methods for Open Object-based Distributed Systems (FMOODS'05). LNCS, vol. 3535, pp. 306–322. Springer (2005)
6. Edenhofer, S., Höfner, P.: Towards a rigorous analysis of AODVv2 (DYMO). In: Rigorous Protocol Engineering (W-RiPE'12). IEEE Press (2012)
7. Espensen, K., Kjeldsen, M., Kristensen, L.: Modelling and initial validation of the DYMO routing protocol for mobile ad-hoc networks. In: van Hee, K.M., Valk, R. (eds.) Applications and Theory of Petri Nets (PETRI NETS'08). LNCS, vol. 5062, pp. 152–170. Springer (2008)
8. 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)
9. van Glabbeek, R.J., Höfner, P., Tan, W.L., Portmann, M.: Sequence numbers do not guarantee loop freedom—AODV can yield routing loops (2012), http://rvg.web.cse.unsw.edu.au/pub/AODVloop.pdf
10. Johnson, D., Lysko, A.: Comparison of MANET routing protocols using a scaled indoor wireless grid. Mob. Netw. Appl. 13(1-2), 82–96 (2008)
11. Kum, D.W., Park, J.S., Cho, Y.Z., Cheon, B.Y.: Performance evaluation of AODV and DYMO routing protocols in MANET. In: Consumer Communications and Networking Conference (CCNC'10). pp. 1046–1047. IEEE Press (2010)
12. Milic, B., Malek, M.: NPART—node placement algorithm for realistic topologies in wireless multihop network simulation. In: Simulation Tools and Techniques (Simutools '09). pp. 9:1–9:10. ICST (2009)
13. Miskovic, S., Knightly, E.: Routing primitives for wireless mesh networks: Design, analysis and experiments. In: Conference on Information communications (INFOCOM'10). pp. 2793–2801. IEEE Press (2010)
14. 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`
15. Perkins, C., Chakeres, I.: Dynamic MANET on-demand (AODVv2) routing. IETF Internet Draft (Work in Progress) (March 2012), `http://tools.ietf.org/html/draft-ietf-manet-dymo-22`
16. R Core Team: R: A Language and Environment for Statistical Computing (2012), `http://www.R-project.org`
17. Saleem, M., Khayam, S.A., Farooq, M.: On performance modeling of ad hoc routing protocols. EURASIP J. Wirel. Commun. Netw. 2010, 31:1–31:13 (2010)
18. Schuts, M., Zhu, F., Heidarian, F., Vaandrager, F.: Modelling clock synchronization in the Chess gMAC WSN protocol. In: Andova, S., McIver, A., D'Argenio, P., Cuijpers, P., Markovski, J., Morgan, C., Núñez, M. (eds.) Quantitative Formal Methods: Theory and Applications (QFM'09). EPTCS, vol. 13, pp. 41–54 (2009)
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 Press (2005)
20. Younes, H.: Verification and Planning for Stochastic Processes with Asynchronous Events. Ph.D. thesis, Carnegie Mellon University (2004)
21. Zave, P.: Using lightweight modeling to understand CHORD. SIGCOMM Comput. Commun. Rev. 42(2), 49–57 (2012)