Towards a Rigorous Analysis of AODVv2 (DYMO)

Sarah Edenhofer
Universität Augsburg, Germany
Email: edenhofer.sarah@googlemail.com

Peter Höfner
NICTA, Australia
University of New South Wales, Australia
Email: peter.hoefner@nicta.com.au

***Abstract*—Dynamic MANET On-demand (AODVv2) routing, formerly known as DYMO, is a routing protocol especially designed for wireless, multi hop networks. AODVv2 determines routes in a network in an on-demand fashion.**

In this paper we present a formal model of AODVv2, using the process algebra AWN. The benefit of this is two-fold: (a) the given specification is definitely free of ambiguities; (b) a formal and rigorous analysis of the routing protocol is now feasible. To underpin the latter point we also present a first analysis of the AODVv2 routing protocol. On the one hand we show that some of the problems discovered in the AODV routing protocol, the predecessor of AODVv2, have been addressed and solved. On the other hand we show that other limitations still exist; an example is the establishment of non-optimal routes. Even worse, we locate shortcomings in the AODVv2 routing protocol that do not occur in AODV. This yields the conclusion that AODVv2 is not necessarily better than AODV.

I. Introduction

Wireless Mesh Networks (WMNs) are ad-hoc networks providing broadband communication without relying on a wired backhaul. They are self-organising, self-configuring and self-healing networks and have a wide spectrum of applications, ranging from emergency response over surveillance applications to smart grids. WMNs have the benefit of low-cost deployment, but have to cover crucial aspects of wireless communication, such as packet loss and node mobility. Since most current commercial wireless (mesh) networks use the same standard IEEE 802.11 radios, the key factor affecting the performance is the software, mainly the network protocols. Limitations, shortcomings or problems in the routing protocol immediately decrease the performance of the entire network.

"The Mobile Ad Hoc working group of the IETF developed, among other protocols, Ad hoc On-Demand Distance Vector (AODV) Routing [...]. This protocol captured the minds of some in the embedded devices industry, but experienced issues in wireless networks such as 802.15.4 and 802.11's Ad Hoc mode. As a result, it is in the process of being updated in the Dynamic MANET On-demand (DYMO) Routing protocol." [2]

Using approaches such as test-bed experiments, simulation or formal analysis, it has been shown that AODV has limitations [8], [20]. For example, Miskovic and Knightly have demonstrated that AODV establishes non-optimal routes, even if the given topology does not change [15]. Other shortcomings stem from ambiguities or contradictions in the informal standard, written in natural language; this may lead to different readings of the standard and hence to implementations that behave differently, although all are compliant to the standard. In [23] it is for example shown that AODV can yield routing loops if ambiguities occurring in the RFC are resolved "incorrectly".

Using words of Zave, there are "three conclusions about the description of protocols, based on extensive experience: (1) Informal methods are inadequate for widely used protocols. (2) Lightweight formal methods are easy and useful. (3) Informal natural language cannot be trusted, but natural-language paraphrases of formal language can be trusted for certain purposes." [24]

To limit the amount of shortcomings and problems in AODVv2 (formerly known as DYMO), we believe that formal modelling, formal analysis and formal reasoning should be applied already during the development process, i.e., *now*. It has been shown that process algebra, which can be complemented by model-checking, is a powerful tool for the analysis of ad hoc on-demand routing protocols for WMNs [9], [10]. In this paper, we use the same techniques to formalise and analyse AODVv2. In particular we present a process-algebraic model of AODVv2 that covers all core functionality, but abstracts from timing issues. Although timing is crucial for routing protocols, already our untimed model reveals limitations. From the algebraic model we generate a formal model that is accepted by the model checker Uppaal. The combination of these two models allow a quick, but thorough analysis of the routing protocol. As a proof of concept we present and discuss a number of shortcomings found in the AODVv2 routing protocol and propose some solutions. Since AODVv2 is currently an internet-draft open for discussions, we hope that some of our findings and suggestions will be considered for the next version of this routing protocol.

The remainder of the paper is organised as follows: In Sect. II, we give a brief overview about the main functionality of Dynamic MANET On-demand (AODVv2) routing. In Sect. III we present our formal model of AODVv2. The model, which is discussed in Sect. III-B, is based on a process-algebraic approach that we present in Sect. III-A. After that we perform a first analysis of the AODVv2 routing protocol in Sect. IV. In particular, we show that AODVv2 still establishes non-optimal routes and that control messages can be lost. Moreover, we argue that loop freedom cannot taken for granted as the internet-draft suggests; a detailed and rigorous analysis has to be carried out, which will be part of future work. Before we summarise our work in Sect. VI, we discuss related work in Sect. V.

II. Dynamic MANET On-demand Routing

AODVv2 [18] (aka. DYMO) is a reactive routing protocol designed for WMNs, i.e., routes are searched, calculated and discovered upon need, and every router only maintains the routes that currently work and are needed. In this paper, we model the latest version of AODVv2, the internet-draft 22, dated March 12, 2012, including the feature of an intermediate route reply (see below). The latter feature is crucial for shortening route-discovery times and described in the internet-draft 00, dated July 10, 2012 [19]. Due to its importance, we have decided to include this feature in our model, although it is not part of the main internet-draft any more—it was part of AODVv2 until the internet-draft 21.

The basic routine of AODVv2 is similar to the one of AODV: if a node S is required to send one or more data packets to a target node T, but does not have a route stored in its routing table, it buffers the data and initiates a route discovery process by broadcasting a route request (RREQ) message. The RREQ is typically forwarded by intermediate nodes—nodes that are not the target. An intermediate node that receives the RREQ message updates its routing table by creating entries for a route to the originator of the message (node S) and to all intermediate nodes that have forwarded the RREQ message (path accumulation). After that, the node typically re-broadcasts the request to its neighbours. This is repeated until the RREQ reaches the target node T that replies by unicasting a corresponding route reply (RREP) message back to the source S along the previous established path. To shorten route-discovery times, intermediate nodes are allowed to and should reply on behalf of the target if they know a route to T. An intermediate node that has this information updates its routing table as usual; after that it unicasts a RREP message back to the originator of the RREQ message (node S), but it also unicasts a RREP message to the target node T. By this, routes between S and T, and between T and S are established.

The RREP messages are forwarded by the intermediate nodes along previously established routes. When forwarding RREP messages, nodes create routing table entries for all nodes that have already forwarded that route, to the originator of the RREP message and, in case the originator is different to the target of the original RREQ, also to the target node T. After the RREP message has reached node S, a route from S to T is established and data packets can start to flow.

In case of link breaks, AODVv2 uses route error (RERR) messages to notify affected nodes, i.e., nodes that could potentially use this link: if a link break is detected by a node, it invalidates all routes stored in the node's own routing table that actually use the broken link. Then it broadcasts a RERR message containing the unreachable destinations to all (direct) neighbours using this route.

When updating routing table entries, nodes use *sequence numbers* to determine the freshness of an entry. The larger the sequence number, the fresher the information—a sequence number with value 0 indicates that the number is not known. To maintain the information about sequence numbers each node stores its own sequence number. It is a common belief that sequence numbers are sufficient to guarantee loop freedom if they are monotonically increased over time. Whenever a node initiates a route request or a route reply, it increments its own sequence number and transmits the incremented value as part of the message. Whenever a RREQ or a RREP message is forwarded, an intermediate node adds its sequence number together with its IP-address to the content of the message.

Full details of the protocol are given in [18] and [19].

$X(exp_1, \ldots, exp_n)$	process name with arguments
$P + Q$	choice between processes P and Q
$[\varphi]P$	conditional process; execute P only if condition φ holds
$[\![\texttt{var} := exp]\!]P$	assignment followed by process P
broadcast$(ms).P$	broadcast message *ms* followed by P
unicast$(dest, ms).P \blacktriangleright Q$	unicast *ms* to *dest*; if successful proceed with P; otherwise with Q
deliver$(data).P$	deliver data to application layer
receive$(\texttt{msg}).P$	receive a message

TABLE I
Main Constructs of AWN ([10], [12])

III. Formal Modelling

Like most of the Standards, RFCs and internet-drafts maintained by the Internet Engineering Task Force (IETF), the description of AODVv2 is given in English prose and follows the philosophy of "rough consensus and working code". It is well-known that those descriptions are not adequate for widely used protocols. Whoever uses the AODVv2 routing protocol or any other protocol should be able to use the specification as standard, unambiguous reference. Experience shows that this is often not the case (e.g. [24]).

A. The specification language AWN

For our specification of AODVv2, we use AWN [10], a process algebra specifically designed for WMNs. Key motivation for the use of AWN is the possibility of writing protocol specifications in a simple, intuitive and unambiguious way that makes the specification easy to read, to understand and to use. Additionally, as all process algebras, AWN provides algebraic laws that allow formal reasoning. The algebra offers three necessary primitives for WMN routing: data structures, local broadcast, and conditional unicast. The former allows us to model routing tables, data packets and other structures needed by the protocol. As usual, local broadcast describes message sending to all connected neighbouring nodes in a network; and the conditional unicast operator models the message sending to exactly one node and chooses a continuation process dependent on whether the message transmission was succesful. These operations, as well as others, such as nondeterministic choice and assignment, are summarised in Table I.

Recently, AWN was used to model and reason about AODV [11]. Since AWN is a formal specification language, it does not allow any ambiguities and hence a specification is absolutely precise. Different readings of the original specification (due to ambiguities) yield different AWN-models, which then can be analysed and compared (e.g., [12]).

Process 1 RREQ Handling (Snippet)

```
RREQ(sip,hoplim,tip,tsn,oip,osn,odist,inodes , ip,sn,rt,store) ≝
 1. [ ip = oip ]      /* node is originator of the message */
 2.    DYMO(ip,sn,rt,store)
 3. + [ ip ≠ oip ]     /* node is not the originator */
 4.    (
 5.       [ ip ≠ oip ∧ rt = update(rt,oip,osn,sip,odist + 1,req) ]      /* info is stale, loop possible, disfavoured or equivalent */
 6.          ⟦inodes := distinc(inodes)⟧     /* increment distances to all intermediate nodes */
 7.          ⟦rt := updinter(rt,inodes,sip,req)⟧     /* update rt to intermediate nodes */
 8.          DYMO(ip,sn,rt,store)
 9.       + [ ip ≠ oip ∧ rt ≠ update(rt,oip,osn,sip,odist + 1,req) ]      /* route information is preferable (fresh enough) */
10.          ⟦rt := update(rt,oip,osn,sip,odist + 1,req)⟧
11.          ⟦inodes := distinc(inodes)⟧     /* increment distances to all intermediate nodes */
12.          ⟦rt := updinter(rt,inodes,sip,req)⟧     /* update rt to intermediate node */
13.          [ ip = tip ]      /* node is target node */
14.             ⟦sn := sn + 1⟧      /* increment node's own sequence number */
15.             /* generate rrep message */
16.             unicast(nhop(rt,oip),rrep(ip,10,oip,osn,ip,sn,0,∅)).DYMO(ip,sn,rt,store)
17.             ▶ /* if the transmission is unsuccessful, a RERR message is generated */
18.                ⟦unodes := {(rip, sqn(rt,rip)) | rip ∈ kD(rt) ∧ nhop(rt,rip) = nhop(rt,oip)}⟧
19.                ⟦rt := invalidate(rt,unodes)⟧
20.                broadcast(rerr(ip,10,unodes)).DYMO(ip,sn,rt,store)
21.          + [ ip ≠ tip ]      /* node is not target node */
22.             [ tip ∈ kD(rt) ∧ sqn(rt,tip) > tsn ]      /* intermediate node generates route reply */
23.                ⟦sn := sn + 1⟧      /* intermediate node increments its own sequence number */
24.                unicast(nhop(rt,oip),rrep(ip,10,oip,osn,ip,sn,0,{(tip,sqn(rt,tip),dist(rt,tip))})).      /* send RREP towards the originator of the request */
25.                   unicast(nhop(rt,tip),rrep(ip,10,tip,tsn,ip,sn,0,inodes ∪ {(oip, osn, odist+1)}))      /* send RREP towards the target of the request */
26.                   ▶ ...
27.    )
```

In this paper we present a formal model of AODVv2 using the process algebra AWN. So far we have not used the full power of AWN, such as formal reasoning; this will be part of future work. However, writing the formal model already revealed shortcomings (cf. Sect. IV) that are not easy to find when reading the draft.

Additional explanations and a full description of AWN can be found in [10], [11].

The process algebraic approach can easily be complemented by model checking. The advantage of this dual approach is that AWN can be used to prove essential properties, such as loop freedom or route discovery. These proofs are valid for all possible scenarios, i.e, they are independent of the network topology of the ordering of route discovery processes etc. However, during the modelling or design process, unexpected behaviour, limitations or shortcomings should be discovered quickly and, if possible, automatically. Here, model checking techniques can be used. In our setting we use Uppaal (e.g. [3]), a well established model checker that has been used for protocol verification before. The two synchronisation mechanisms provided by Uppaal, binary synchronisation and broadcast channels, match with the AWN-primitives **unicast** and **broadcast**. For our analysis of AODVv2 we use Uppaal in two ways: (a) when we found a shortcoming in the protocol during modelling, we used the simulation feature to quickly verify this limitation; (b) we run our model with hundreds of small topologies to detect limitations of the protocol. Of course the properties we checked have to be given manually; for our first analysis we took the properties used for AODV in [8] and [9].

B. The formal AODVv2 model

Our formalisation of AODVv2 carefully follows the IETF's internet-drafts [18], [19]. Since the drafts are written in English prose, they lack of clarity at several points and contains ambiguities. There are even some unspecified scenarios. Hence our formalisation is only one of many possible readings of the drafts. However, it is our belief that the presented model captures the intuition behind AODVv2.

So far our formalisation models the core functionalities of AODVv2, including intermediate route reply, but abstracts from timing aspects. Additionally, we model the injection, the forwarding and the delivery of data packets. Although this is not part of any routing protocol, it is crucial to trigger route discovery processes and to formalise protocol properties.

Our AWN-model consists of around 150 lines split up into six processes: (1) `DYMO` reads a message from the message queue and, depending on the type of the message, calls other processes. When there is no message handling going on, the process can initiate the transmission of queued data packets or generate a new route request; (2) `PKT` describes all actions performed by a node when a data packet is received; (3) `RREQ` models all events occurring when a route request is handled (see below); (4) `RREP` describes the reaction of the protocol to an incoming route reply; (5) `RERR` models the part of AODVv2 that handles error messages. In particular, it describes the modification and forwarding of error messages; and finally, (6) `QMSG` concerns message handling. Whenever a control message is received, it is stored in a message queue.

In this paper, we only present a snippet of our model (see Pro. 1); the full AWN-model including a description of the data structure, all definitions of functions used, as well as the corresponding Uppaal-model is available online at http://www.hoefner-online.de/wripe12/.

Process 1 shows our specification of the handling of a RREQ message received by a node `ip`. The node itself maintains, next to its IP address, its own sequence number `sn`,

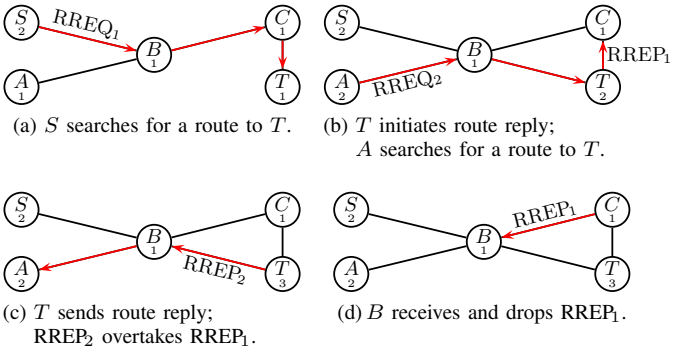
its routing table `rt` and a `store` for buffering data packets. First, the node checks if it is the originator (`oip`) of the RREQ message. If this is the case (Line 1), the message is ignored and the main process `DYMO` is called (Line 2). In case `ip` is not the originator, a test on the freshness of the incoming route information is performed. A message is considered fresh enough, if the routing table entry for the originator `oip` is updated, i.e., the function `update` changes the routing table.[1] In case the routing table entry for `oip` is not updated (Line 5), routes to intermediate nodes are updated or created (Line 7), but the message itself is still dropped. In case the received message contains fresh routing information for the originator `oip`, this information as well as all information about routes to intermediate nodes are inserted in the routing table (Lines 10–12). Lines 14–20 deal with the case where the node receiving the RREQ message is the intended target, i.e., `ip`$=$`tip` (Line 13). In this case, first `ip` increments its own sequence number. Then the node generates a RREP message, which is unicast to the next hop on the route to `oip` (Line 16). Parameters for the generation of the message are the sender node `ip`, a hop limit[2] (which is set to the value 10), the final destination of the message `oip` in combination with its sequence number `osn`, as well as information about the originator of the RREP message. The last parameter $\varnothing$ will be changed when the message is forwarded and will contain information about all intermediate nodes. In case the unicast fails, a route error message is generated and broadcast (Lines 18–20). The remaining lines shown in the snippet handle the case of intermediate route reply, i.e. `ip` is not the target node, but has information that can be used to generate a route reply. In that case two RREP messages are sent: the first one to `oip`, the originator of the RREQ message (Line 24), the second one to the target of the same message (Line 25).

IV. A First Analysis

Based on our formal model presented in the previous section, we perform an analysis of AODVv2. In particular we discuss which problems that already occured in AODV have been solved in AODVv2 and which have not. All findings are formally verified with Uppaal.

A. Message Losses

It is well known that AODV may lose route replies. This is due to the fact that messages are only forwarded if the routing table of an intermediate node is updated (changed). That means, if the received control message does not contain new content, it is ignored. An example, taken from the IETF mailing list[3], uses a linear topology of four nodes: S—A—B—T. First, A establishes a route to T, using a standard RREQ-RREP cycle. If intermediate node reply is deactivated and S searches for a route to T, the RREQ message is forwarded to T. Node T generates a RREP message. Node B will not forward the reply since the message does not contain



(a) S searches for a route to T.
(b) T initiates route reply; A searches for a route to T.
(c) T sends route reply; RREP_2 overtakes RREP_1.
(d) B receives and drops RREP_1.

Fig. 1. Lost Route Reply

new information about T. Similar scenarios can happen even if intermediate nodes are allowed to generate route replies on behalf of the target, see [12].

This shortcoming has been solved and addressed in AODVv2: just before a node generates a message, it increments its own sequence number (Lines 14 and 23 of Pro. 1). By this, the information about the originator of the message is guaranteed to be "fresher" than any information stored in any routing table. However, in AODVv2 nodes still forward messages only if a route (to the originator of the message) is updated (or created) (cf. Lines 5 and 9). Following this strategy messages can be lost when messages "overtake" each other. An example is given in Fig. 1. The numbers shown in the nodes indicate the nodes' sequence numbers. In Fig. 1(a), node S initiates a route discovery process destined for T. The message is forwarded to T via B and C. The link between B and T either does not exist or the message travelling along that link is lost. After that, in Fig. 1(b), T unicasts a reply back to C. At the same time it receives another route request, this time initiated by A. While the first reply is travelling via C,[4] RREP_2 overtakes it and is received by node B first. B updates its routing table and forwards the RREP message to A. When the first reply finally arrives at B it is ignored, since it carries older information about T.

Most likely S will find a route, when re-sending the request. However, broadcasting requests always yields many messages in the network; it would be more reasonable to forward *all* RREP messages. Unfortunately, the situation is even worse. In AODV only RREP message were lost; in AODVv2 RREQ messages can be dropped as well. Since the freshness of RREQ message are determined only by the use of sequence numbers, a similar example can be constructed.[5] Both examples were found with the help of Uppaal.

B. Non-Optimal Routes

In [15] it is shown that current routing protocols such as AODV often fail to select the best (shortest) routes. There, it is shown that "poorly selected paths can have significantly higher routing-metric costs, and their duration can extend to

[1] `update` returns the unchanged routing table if the information is stale.
[2] The remaining number of hops this message is allowed to traverse.
[3] http://www.ietf.org/mail-archive/web/manet/current/msg05702.html

[4] In reality there might be several nodes on the paths from T to C and from C to B. We just keep the example small.
[5] In contrast to this, AODV keeps track of all RREQ messages previously handled and avoids such scenarios.

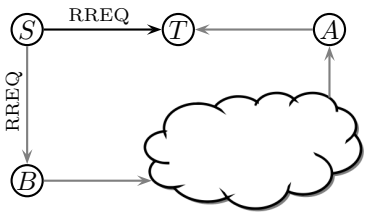


Fig. 2. Node A accidentally creates a non-optimal route to S.

minute time scales". In AODV the only nodes that generally discover optimal routes to the source and destination nodes are those lying on the selected route between the source and the target (or the intermediate node) generating the reply. Other nodes may accidentally create non-optimal routes. The same situation occurs in AODVv2; an example is sketched in Fig. 2.

Assume a "ring topology" with at least five nodes, where node S searches for a route to T. The node S initiates a route discovery process by broadcasting a RREQ message, which is received by nodes B and T. Following the specification, T generates a route reply, but does not forward the RREQ message (cf. Pro. 1, Line 16). On the other hand, since B is not the target node and does not have information about the target, it continues to forward the received RREQ message. Eventually, this message will be received by A. Although node A is only two hops away from S, it will establish a non-optimal route to S via B. In AODV the route to S is the only non-optimal one. Since AODVv2 uses path accumulation, i.e. every intermediate node adds information about itself, A may establish many more non-optimal routes.

In case of AODV, it has been shown in [12] that this shortcoming can be solved by allowing the target node to forward the RREQ message. The same method solves the problem in AODVv2. This would allow node A to establish its optimal route to S. In addition, the forwarded RREQ message from the destination node could be modified to include a flag that indicates a RREP message has already been generated and sent in response to the former message. This would prevent other nodes from generating RREP messages.

C. Loop Freedom & Self-Entries

It is a common belief that sequence numbers are sufficient to guarantee loop freedom if they are monotonically increased over time. In the last but one internet draft of AODVv2 (DYMO) [17] it is written that "DYMO uses sequence numbers to ensure loop freedom [Perkins99]". Here, [Perkins99] refers back to AODV [16].[6] However, it has been shown recently that AODV is not a priori loop free and that some of the open-source implementations can yield routing loops [23]. In fact, loop freedom depends on non-evident assumptions to be made when interpreting the English specification. Following this result, loop freedom of AODVv2 cannot be taken for granted.

Using our formal model in combination with tests performed with the Uppaal model checker on several hundreds topologies with up to 5 nodes we were not able to find a loop.

[6]In the current draft, this reference has been removed.

This strengthens the conjecture that (our model of) AODVv2 is loop-free. However, while looking at loop freedom, we found out that nearly every node establishes self-entries, i.e., stores routing table entries to itself. The reason for this is that nodes establish routes to *all* intermediate nodes. An example is given in Fig. 3. Here, node B receives a RREQ

Fig. 3. Node A establish a self-entry.

message that was sent by node A. In case that B forwards the message, it adds information about itself to the content of the message. The broadcast message is received by A, which updates (creates) routing table entries to all nodes that have handled this message before. This includes the node A itself. In practice, when a node sends data to itself, it will not use any routing protocol. Instead the packet will be delivered to the corresponding application directly. However, the information stored in the routing table is sent to neighbouring nodes by the protocol. Broadcasting information about self-entries does not make sense and, in case of AODV, might even yield routing loops [23]. Due to this, self-entries should be forbidden in AODVv2. A simple if-statement could achieve this.

V. Related Work

The main tools for the evaluation of network routing protocols are still provided by test-bed experiments and simulation. In case of AODVv2, there are a bunch of performance evaluations [1], [13], [14], [21], [22]. The first one for example shows that the performance of AODVv2 is not as good as that of AODV. As performance measurement the authors use, among others, at the packet delivery ratio (PDR). However, none of the papers list reasons for the problematic performance.

Looking at formal models, there is only one other approach we are aware of, namely coloured Petri nets [5], [7]. Both papers model AODVv2 (different versions). The models presented differ in the size and in the capability of modelling different topologies. Finally [5], [7] perform test runs to "prove" that the models behave as expected. Again, an analysis is not provided. Moreover it is difficult to see that the model captures the main functionality of AODVv2, since Petri nets do not correspond nicely to the description given in the draft nor to a programming language. This is in contrast to our approach using AWN, which should be easy to understand.

Of course, formal methods such as process calculi and model checking have been used in the past to analyse different protocols. For example, [25] 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. Other approaches use the model checkers Spin and Uppaal, respectively, to analyse AODV, the predecessor of AODVv2 [4], [6], [9]. For example, it was shown with the help of Spin that an early draft of AODV could create routing loops [4].

VI. Conclusion and Future Work

In this paper we have used the process algebra AWN to model the Dynamic MANET On-demand (AODVv2) routing protocol. The formal model is based on the latest draft and captures the main functionalities of the protocol, but abstracts from timing aspects. Moreover, we have derived a model for Uppaal. Both (isomorphic) models have been used to do a first analysis of the routing protocol. Doing this, we have revealed shortcomings of AODVv2 and have sketched possible solutions. To the best of our knowledge, we are the first who found limitations in AODVv2 (DYMO) in general and in the current draft of AODVv2 in particular. The latter might not be surprising since the latest draft we built on ([19]) was published only two weeks ago, but shows nicely that a formal analysis does not need much time.

Surely, all the presented problems could theoretically be found by test-bed experiments, simulations or just by "staring" at the draft. However, experience with test-beds and simulations show that these approaches need some amount of time to be set up and it takes usually a long time to find shortcomings.

So far we have done only a first analysis of the protocol. A more rigorous analysis using formal proving techniques of AWN and model checking is part of future work. In particular, we plan to provide a formal proof of loop freedom of AODVv2. There are good reasons to believe that the protocol under consideration is loop-free, but contrary to common belief this cannot be taken for granted. Examples of other protocols that were thought to be loop free and were not are known (e.g. [4], [23]). In parallel we want to evaluate the limitations found. In particular we want to use experiments to find out how often they occur in real network scenarios. Hence, mainly simulation techniques will be used. More further work will be the careful analysis and the implementation of the improvements suggested for AODVv2.

References

[1] M. Amin, M. Abrar, Z. U. Khan, Andusalam, and S. Rizwan, "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] F. Baker and D. Meyer, "Internet protocols for the smart grid," RFC 6272 (Informational), June 2011, request for Comments. [Online]. Available: ftp://ietf.org/rfc/rfc6272

[3] G. Behrmann, A. David, and K. Larsen, "A Tutorial on Uppaal," in *Formal Methods for the Design of Real-Time Systems*, ser. LNCS, M. Bernardo and F. Corradini, Eds. Springer, 2004, vol. 3185, pp. 200–236, http://dx.doi.org/10.1007/978-3-540-30080-9_7.

[4] K. Bhargavan, D. Obradovic, and C. A. Gunter, "Formal verification of standards for distance vector routing protocols," *J. ACM*, vol. 49, no. 4, pp. 538–576, 2002.

[5] J. Billington and C. Yuan, "On modelling and analysing the dynamic MANET on-demand (DYMO) routing protocol," in *Transactions on Petri Nets and Other Models of Concurrency III (ToPNoC)*, ser. LNCS, K. Jensen, J. Billington, and M. Koutny, Eds. Springer, 2009, pp. 98–126. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04856-2_5

[6] S. Chiyangwa and M. Kwiatkowska, "A timing analysis of AODV," in *Formal Methods for Open Object-based Distributed Systems (FMOODS'05)*, ser. LNCS, vol. 3535. Springer, 2005, pp. 306–321.

[7] K. Espensen, M. Kjeldsen, and L. Kristensen, "Modelling and initial validation of the DYMO routing protocol for mobile ad-hoc networks," in *Applications and Theory of Petri Nets (PETRI NETS'08)*, ser. LNCS, K. M. van Hee and R. Valk, Eds., vol. 5062. Springer, 2008, pp. 152–170. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-68746-7_13

[8] A. Fehnker, R. J. van Glabbeek, P. Höfner, A. McIver, M. Portmann, and W. L. Tan, "Modelling and analysis of AODV in UPPAAL," in *Workshop on Rigorous Protocol Engineering (W-RiPE'11)*, 2011.

[9] ——, "Automated analysis of AODV using UPPAAL," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*, ser. LNCS, C. Flanagan and B. König, Eds., vol. 7214. Springer, 2012, pp. 173–187.

[10] ——, "A process algebra for wireless mesh networks," in *European Symposium on Programming (ESOP'12)*, ser. LNCS, H. Seidl, Ed., vol. 7211. Springer, 2012, pp. 295–315.

[11] ——, "A process algebra for wireless mesh networks used for modelling, verifying and analysing AODV," NICTA, Tech. Rep. 5513, 2012, http://www.nicta.com.au/pub?id=5513.

[12] P. Höfner, R. J. van Glabbeek, W. L. Tan, M. Portmann, A. McIver, and A. Fehnker, "A rigorous analysis of AODV and its variants," in *Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM'12)*. ACM Press, 2012.

[13] D. Johnson and A. Lysko, "Comparison of manet routing protocols using a scaled indoor wireless grid," *Mob. Netw. Appl.*, vol. 13, no. 1-2, pp. 82–96, 2008. [Online]. Available: http://dx.doi.org/10.1007/s11036-008-0048-2

[14] D.-W. Kum, J.-S. Park, Y.-Z. Cho, and B.-Y. Cheon, "Performance evaluation of AODV and DYMO routing protocols in MANET," in *Consumer Communications and Networking Conference (CCNC'10)*. IEEE, 2010, pp. 1046–1047. [Online]. Available: http://dl.acm.org/citation.cfm?id=1834217.1834455

[15] S. Miskovic and E. W. Knightly, "Routing primitives for wireless mesh networks: Design, analysis and experiments," in *Conference on Information Communications (INFOCOM'10)*. IEEE, 2010, pp. 2793–2801.

[16] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Mobile Computing Systems and Applications (WMCSA'99)*, 1999, pp. 90–100. [Online]. Available: http://dx.doi.org/10.1109/MCSA.1999.749281

[17] C. Perkins and I. Chakeres, "Dynamic MANET on-demand (DYMO) routing," IETF Internet Draft, January 2011, (Work in Progress). [Online]. Available: http://tools.ietf.org/html/draft-ietf-manet-dymo-21

[18] ——, "Dynamic MANET on-demand (AODVv2) routing," IETF Internet Draft, March 2012, (Work in Progress). [Online]. Available: http://tools.ietf.org/html/draft-ietf-manet-dymo-22

[19] ——, "Intermediate RREP for dynamic MANET on-demand (AODVv2) routing," IETF Internet Draft, July 2012, (Work in Progress). [Online]. Available: http://tools.ietf.org/html/draft-perkins-irrep-00

[20] M. Saksena, O. Wibling, and B. Jonsson, "Graph grammar modeling and verification of ad hoc routing protocols," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08)*, ser. LNCS, C. R. Ramakrishnan and J. Rehof, Eds., vol. 4963. Springer, 2008, pp. 18–32.

[21] M. Saleem, S. A. Khayam, and M. Farooq, "On performance modeling of ad hoc routing protocols," *EURASIP J. Wirel. Commun. Netw.*, vol. 2010, pp. 31:1–31:13, 2010. [Online]. Available: http://dx.doi.org/10.1155/2010/373759

[22] E. Spaho, L. Barolli, G. Mino, F. Xhafa, and V. Kolici, "Goodput evaluation of AODV, OLSR and DYMO protocols for vehicular networks using CAVENET," in *Network-Based Information Systems (NBIS '11)*. IEEE, 2011, pp. 118–125. [Online]. Available: http://dx.doi.org/10.1109/NBiS.2011.27

[23] R. J. van Glabbeek, P. Höfner, W. L. Tan, and M. Portmann, "Sequence numbers do not guarantee loop freedom—AODV can yield routing loops," (submitted), 2012, http://rvg.web.cse.unsw.edu.au/pub/AODVloop.pdf.

[24] P. Zave, "Experiences with protocol description," in *Workshop on Rigorous Protocol Engineering (W-RiPE'11)*, 2011.

[25] ——, "Using lightweight modeling to understand CHORD," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 2, pp. 49–57, 2012. [Online]. Available: http://doi.acm.org/10.1145/2185376.2185383