Mobile Networks and Applications (MONET) manuscript No.
(will be inserted by the editor)

Simulation of Ad Hoc Routing Protocols using OMNeT++
A Case Study for the DYMO Protocol

Christoph Sommer · Isabel Dietrich ·
Falko Dressler

Authors' version. The original publication is available at www.springerlink.com.
http://dx.doi.org/10.1007/s11036-009-0174-5

Abstract Mobile Ad Hoc Networks (MANETs) have evolved in the last years into standards in the communication world. By definition, they do not need any network infrastructure to facilitate communication between participating nodes. Therefore, MANETs are dealing with new challenges in the context of ad hoc routing. Simulation techniques are one of the fundamental methodologies to support the protocol engineering process, especially in the early stages of ad hoc network protocol design. In this paper, we set out common criteria that may serve as guidelines for meaningful simulative evaluations of ad hoc routing protocols. We present typical and necessary measures for ad hoc routing in general and MANET routing in particular. As a case study, we demonstrate a comprehensive performance evaluation of the Dynamic MANET On Demand (DYMO) routing protocol using a model we implemented for the popular OMNeT++ discrete event simulation environment.

Keywords MANET · Protocol · Simulation · OMNeT++ · Ad hoc routing · Performance Evaluation

1 Introduction

Ad hoc routing is still a hot topic in the domain of mobile communication. This can be seen from the high number of papers describing new routing approaches as well as from the activities in the IETF Mobile Ad Hoc Network (MANET) working group. Several surveys critically discuss the advantages of those protocol approaches especially in terms of the routing efficiency [1,12]. Thus, common criteria are required for meaningful evaluations of ad hoc routing protocols. Basically, the entire protocol engineering process strongly relies on methods for protocol testing and performance evaluation. Simulation is one of such methods. Usually, simulation techniques are employed if the

C. Sommer, I. Dietrich and F. Dressler
Computer Networks and Communication Systems
Department of Computer Science, University of Erlangen, Germany
E-mail: {christoph.sommer,isabel.dietrich,dressler}@informatik.uni-erlangen.de

complexity of the protocol and/or the environment becomes too high for analytical solutions.

Simulation techniques allow analyzing the behavior of networking protocols even at a large scale – depending on the available computing power for running the simulation experiments. Usually, discrete event simulation is employed for analyzing networking protocols. There are a huge number of simulation tools available. In this case study, we focus on OMNeT++ [26]. Nevertheless, all the findings and observations for meaningful analysis of ad hoc routing protocols can be applied to any other discrete event based network simulator such as the well known ns-2.

OMNeT++ already provides the framework for comprehensive simulation setups. In the context of the *INET Framework* and the *mobility framework*, a number of ad hoc routing protocols have been implemented as described in Section 3.1. Furthermore, models for a great number of network protocols, including the complete TCP/IP stack are available.

In this paper, we draw some conclusions from a simulation study of the Dynamic MANET On Demand (DYMO) routing protocol, which we presented at the 1st International Workshop on OMNeT++ [22]. This study was performed using a simulation model of DYMO, which we developed for the OMNeT++ simulation environment. Furthermore, the paper presents a generic study of simulative performance analysis of ad hoc routing protocols. We describe the steps from developing the simulation model (Section 3.2) until the final simulation results can be evaluated (Section 4). Thus, this paper can be used as a guideline for evaluating further ad hoc routing approaches using OMNeT++ or any other event driven simulation framework. We specifically describe the capabilities of OMNeT++ for evaluating ad hoc routing protocols, using the DYMO routing protocol as a case study. Based on the implemented DYMO model, we performed a comprehensive performance evaluation to study the effects of DYMO (Section 4.6).

2 Dynamic MANET On Demand

Research on MANETs addresses many objectives such as scalability and energy efficiency of ad hoc routing techniques [15, 20] and a number of protocols have been proposed in the last decade [1, 12, 13]. These routing protocols build the basis for all communications in MANETs as well as for even more resource restricted networks such as Wireless Sensor Networks (WSNs).

In general, routing protocols can be differentiated according to their method of acquiring and distributing routing information [7]. In the following, we briefly outline the typical classes of ad hoc routing protocols and present the protocol DYMO in more detail.

2.1 Classification of ad hoc routing protocols

Basically, three techniques can be distinguished: proactive, reactive, and hybrid protocols [15]. The basic characteristics of these techniques are described in the following.

Proactive routing protocols rely on the periodic collection and exchange of topology information. All such protocols essentially always provide up-to-date routing information, which can be used to forward data packets through the network towards their destination. Periodic routing updates require a remarkable amount of network overhead for state (topology) maintenance – depending on the specific protocol and the degree of mobility. The best known example of a table-driven ad hoc routing protocol is the Destination Sequenced Distance Vector (DSDV) protocol [18].

Reactive routing protocols have been introduced to prevent the periodic routing information exchange, which may consume an essential amount of the available network resources. The idea is to only search for paths through the network if data packets need to be transmitted. The first protocol of this class was developed back in 1994: the Dynamic Source Routing (DSR) protocol [14]. More current approaches are Ad Hoc on Demand Distance Vector (AODV) [19], which is probably the best known protocol in the ad hoc networking community, and DYMO, the most recent development [4].

Hybrid routing protocols try to exploit the specific advantages of proactive routing protocols, i.e. the fast delivery of packets as routing information is always available, and of reactive routing protocols, i.e. the reduced costs in terms of network overhead for state maintenance. A typical example is the Zone Routing Protocol (ZRP) [11]. ZRP distinguishes between a local neighborhood called the "routing zone", for which routing information is maintained proactively, and all destinations beyond the routing zone, for which routes are acquired on demand.

2.2 Route discovery using DYMO

DYMO is a reactive (on-demand) routing protocol, which is currently developed in the scope of the MANET working group of the Internet Engineering Task Force (IETF). DYMO builds upon experience with previous approaches to reactive routing, especially with the routing protocol AODV. It aims at a somewhat simpler design, helping to lower the nodes' system requirements and simplify the protocol's implementation. DYMO retains proven mechanisms of previously explored routing protocols like the use of sequence numbers to enforce loop freedom. At the same time, DYMO provides enhanced features, such as covering possible MANET-Internet gatewaying scenarios and implementing path accumulation.

Besides route information about a requested target, a node will also receive information about all intermediate nodes of a newly discovered path. Therein lies a major difference between DYMO and AODV, the latter of which only generates route table entries for the destination node and the next hop node, while DYMO stores routes for each intermediate hop. This is illustrated in Figure 1. When using AODV, node A knows only the routes to B and D after the route request is satisfied. In DYMO, the node additionally knows a route to node C.

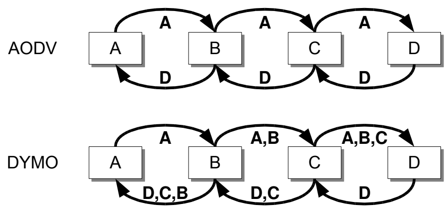


Fig. 1 Routing information dissemination in AODV and DYMO

DYMO is able to set up and maintain unicast routes in IPv4 and IPv6 network scenarios by using the following mechanism:

1. In order to discover a new route to a peer, a node transmits a route request message (RREQ) to all nodes in range. This can be achieved by sending the message to a dedicated link-local multicast address that is associated with all MANET routers. When an intermediate node receives such an RREQ, it takes note of previously appended information, deducing routes to all nodes the message previously passed through. The node then appends information about itself and passes the message on to all nearby nodes. This way, the RREQ is effectively flooded through the MANET and eventually reaches its destination.
2. The destination responds to the received RREQ by sending a route reply message (RREP) via unicast back to the node it received the RREQ from. As with the propagation of a RREQ, this node again appends information about itself and takes note of all routing information contained in the RREP. With the help of the routing information previously obtained while forwarding the corresponding RREQ, the intermediate node is able to send the RREP further back to the start of the chain, until it eventually reaches the originating node. This node will now know a route to the requested destination, as well as routes to all intermediate nodes, and vice versa.

An implementation may also choose to allow intermediate nodes the generation of RREPs on behalf of a RREQ's destination if a suitable route was already stored.

To efficiently deal with highly dynamic scenarios, links on known routes may be actively monitored, e.g. by using the MANET Neighborhood Discovery Protocol (NHDP) [5] or by examining feedback obtained from the data link layer. An implementation may also choose to not actively monitor links, but simply drop inactive routes. Detected link failures are made known to the MANET by sending a route error message (RERR) to all nodes in range, informing them of all routes that now became unavailable. Should this RERR in turn invalidate any routes known to these nodes, they will again inform all their neighbors by multicasting an RERR containing the routes concerned, thus effectively flooding information about a link breakage through the MANET.

DYMO is also designed with future enhancements in mind. It uses the Generalized MANET Packet/Message Format [6] and offers ways of dealing with unsupported elements in a sensible way. Furthermore, multipath capabilities have been investigated in order to improve the throughput in MANETs [10].

2.3 Available implementations

Currently, there are a few implementations of DYMO available, mainly for Linux-based systems. Most notable are the following DYMO implementations for the Linux kernel:

– *DYMOUM* (GPL)
– *NIST-DYMO* (public domain)
– *EK-DYMO* (closed source)
– *DYMO-AU* (GPL, written in LUA and C)

Furthermore, there is also an implementation available developed for sensor motes of the Mica series and the TinyOS operating system: *TYMO* (GPL, written in nesC).

In the world of network simulation, there exists an old implementation of *DYMO for OPNET* (proprietary license). Furthermore, the *DYMOUM* package has been ported to two of the most commonly used network simulators, ns-2 and OMNeT++. The *DYMOUM* package is based on DYMO draft version 05.

There have been several factors that motivated us to start a new implementation for OMNeT++. First, the dependency on Linux-based packages lead to some problems in a number of simulation settings. For example, evaluating DYMO in a sensor network scenario was almost impossible due to the need to always involve the complete TCP/IP stack. Furthermore, the *DYMOUM* package only considered quite an old version of the IETF draft.

Therefore, we started a new implementation of DYMO for OMNeT++ that is also used for the presented case study in our group. This OMNeT++ model is freely available for use and distribution under the terms of the GPL [22].[1] Our simulation model closely followed and supported the IETF's development of the DYMO protocol, currently reflecting the state of draft version 10. As a main advantage of this model, we see the applicability in a fully standard conform TCP/IP based environment; as well as the available simplified version for use without the complex IP stack, e.g. for evaluating of ad hoc routing based applications in sensor networks.

3 Simulation Model

In the following, we briefly introduce the OMNeT++ simulation environment and our implementation of DYMO, which we used for the performance evaluation study.

3.1 The OMNeT++ simulation environment

OMNeT++ is a simulation environment free for academic use [26]. The OMNeT++ engine runs discrete, event-driven simulations of communicating nodes on a wide variety of platforms and is becoming increasingly popular in the field of network simulation. The simulation kernel currently used in the OMNeT++ community and the one we performed all evaluations with is OMNeT++ 3.4b2. At present, the developers are working to release version 4.0 [27].

Scenarios in OMNeT++ are represented by a hierarchy of small, reusable modules written in C++. Behavioral modeling of modules using finite state machines is supported by OMNeT++ and communication within and between modules is primarily based on message passing, but the foundation in open source software written in C++ also allows for a rapid prototyping approach to module development. Modules' relationships and communication links are stored as plain-text *Network Description* (NED) files and can be modeled graphically. Simulations are either run interactively in a graphical environment or are executed as command-line applications.

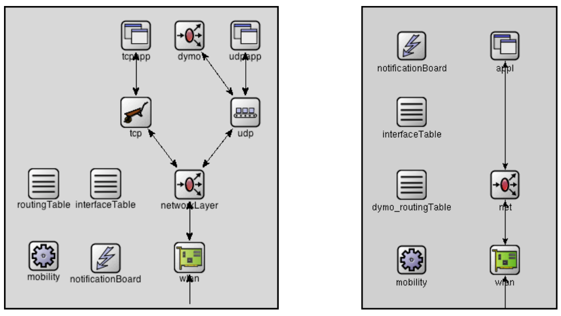
The *INET Framework* extension is a set of simulation modules released under the GPL. It provides OMNeT++ modules that represent various layers of the Internet protocol suite, e.g. the TCP, UDP, IPv4, and ARP protocols. The *INET Framework* also features models for wireless radio communication including radio distribution models and various MAC protocols such as IEEE 802.11. Mobility modeling was introduced into OMNeT++ with the *Mobility Framework* [8]. It was later incorporated into the *INET Framework* for modeling spatial relations of mobile nodes.

[1] The model and simulation setups are available online at `http://www7.informatik.uni-erlangen.de/~sommer/`

In conclusion, OMNeT++ and the *INET Framework* provide all the necessary components for simulating Internet protocols in general and MANET protocols in particular. Because of its modular architecture and its ability to directly access, monitor and alter all modules' internal states, OMNeT++ is very well suited for the implementation of complex protocols. Based on our implementation of DYMO, we will show the feasibility and the procedures of evaluating ad hoc routing protocols.

3.2 Implementation of DYMO

The DYMO routing protocol was implemented as an application layer module of the *INET Framework* module set, using Netfilter-style hooks in the network layer. Following the specification [4], it employs UDP to communicate with other instances of DYMO. The network layer hooks are used for two purposes. First, they facilitate the queuing of outbound packets before routing in the network layer occurs, so that a route can be set up by DYMO. The queue can then be signaled to release buffered packets for a given destination—either in order to have them routed to the first hop or to have them discarded by the network layer because no route could be found. Secondly, a hook is installed in the inbound packet path. It notifies DYMO of the arrival of packets. This way, routing table entries can be refreshed and route errors can be sent. Following the OMNeT++ component model, assembling DYMO and standard components of the *INET Framework* to form a simulated MANET node, illustrated in Figure 2(a), is a straightforward task.



(a) full-featured (b) minimal

Fig. 2 Screenshots of full-featured and minimal MANET nodes running DYMO

We outfitted our model of DYMO with the capability to queue payload messages received from the application layer, should no usable route be known at the time data is received. As mandated, our model will in this case repeatedly try to establish a route, then dequeue the messages for delivery to the destination or for destruction if no route could be found. Regarding route maintenance, we chose the simplest of the defined models for our implementation. Established routes are not actively monitored, but just time out if they are not used.

Two mechanisms are used to limit the range and the frequency of RREQ flooding, respectively. In order to limit the range, an expanding ring search technique as used by AODV is used to find the target of RREQs, linearly increasing the TTL

Table 1 DYMO Module Parameters

Parameter	Value
`MIN_HOPLIMIT`	5 hops
`MAX_HOPLIMIT`	10 hops
`NET_TRAVERSAL_TIME`	1000 ms
`ROUTE_TIMEOUT`	5 s
`ROUTE_AGE_MIN_TIMEOUT`	1 s
`ROUTE_AGE_MAX_TIMEOUT`	60 s
`ROUTE_NEW_TIMEOUT`	5 s
`ROUTE_USED_TIMEOUT`	5 s
`ROUTE_DELETE_TIMEOUT`	10 s
`RREQ_RATE_LIMIT`	$10\,\mathrm{s}^{-1}$
`RREQ_BURST_LIMIT`	3 RREQs
`RREQ_WAIT_TIME`	2 s
`RREQ_TRIES`	3

from MIN_HOPLIMIT to MAX_HOPLIMIT with each new try. In order to limit the frequency of RREQs, a token bucket mechanism is used, with RREQ_RATE_LIMIT configuring an average rate and RREQ_BURST_LIMIT setting the maximum burst size.

For the purpose of evaluating the performance of the routing protocol only, as well as in order to prevent potential side effects introduced by a transport or network layer, we designed a variant of our DYMO model as a replacement for all intermediate layers and thus used it to not only forward RREQs and RREPs, but also to take care of delivering our application layer's payload data.

As shown in Figure 2(b), the simulated network nodes utilized in this evaluation thus contain only three modules for the handling of messages: Application layer data is sent and received by a traffic generator module, is routed through the DYMO module and exchanged with other nodes via an IEEE 802.11 module provided by the *INET Framework*. This simplified model of the DYMO protocol can also be used to evaluate ad hoc routing in environments that do not support the complex TCP/IP stack. The modules on the left hand side of Figure 2(b) do not deal with messages directly, but manage node connectivity (*notificationBoard*, *interfaceTable*) and node movement (*mobility*), as required by the *INET Framework*.

An investigation of effects introduced by the presence of the *INET Framework*'s transport and network layers has also been conducted, but is out of scope for the presented evaluation.

The simulation parameters used to configure a DYMO module correspond directly to the suggested parameter set. They are summarized in Table 1, together with the values used in our evaluation.

4 Performance Analysis of Ad Hoc Routing Protocols

The performance evaluation of ad hoc routing protocols requires more than just an implementation of the protocol in a given simulation environment. In the following subsections, we outline the necessary steps for successfully evaluating such a routing protocol. This case study can be used as a basis for further simulation experiments. We do not intend to provide a fully comprehensive protocol analysis – this is strongly depending on the intended application scenario. Instead, we outline the different performance evaluation criteria and steps based on a number of simulation experiments using our implementation of DYMO [22]. This shows the protocol evaluation in different MANET scenarios and under different traffic conditions.

The following issues will be discussed and evaluated:

– *Network topology and mobility issues* – typical network topologies, including measures like the node density, mobility pattern, and special cases such as a line scenario for ad hoc routing protocols
– *Parameter settings and traffic patterns* – configurable protocol parameters and the traffic behavior for the performance evaluation
– *Performance metrics* – relevant performance metrics to evaluate the efficiency of the protocol
– *Validation* – methods and techniques for model validation
– *Simulation control* – parameters such as the setup of simulation replications and the termination criteria for improved statistical relevance of obtained results
– *Analysis of simulation results* – a comprehensive analysis of the measurement results

4.1 Network topology and mobility issues

4.1.1 Topology

Ad hoc routing protocols must be able to cope with a wide variety of different *network topologies*. In spite of that, evaluations are often only performed in a random topology using a varying number of nodes. The main problem with random topologies is that they will most likely not cover the worst case setups for ad hoc networks. Therefore, we recommend evaluating routing protocols using several different topologies that help to outline possible weaknesses in the ad hoc routing process. The following topologies represent a subset of possibly helpful setups that are known to also stress aspects besides those covered by a random topology alone.

Random topology The nodes are placed randomly in a rectangular simulation playground. This is often the base scenario in protocol evaluations. It can be used to provide an indication of how the protocol may perform in the average case.

Grid topology The nodes are placed in a regular grid. The symmetry of the grid may cause problems for wireless communication.

Linear network The nodes are placed in a straight line forming a single possible path through the network that must be discovered by the protocol. This topology can be used to gain insight into how a protocol handles long routing paths.

4.1.2 Density

In addition to the different (initial) topologies, the *density* of the network can have a great influence on the routing performance. Thus, both sparse and dense networks need to be evaluated. Concerning density, it is not necessary to regard the absolute distances between nodes. What is relevant is rather the relation between inter-node distances and their communication ranges, i.e. how many neighbors there are within the communication range of a node.

4.1.3 Mobility

Finally, node *mobility* influences the outcome of simulations to a large degree. Various mobility models have been proposed for the examination of ad hoc routing protocols, depending on a scenario's abstraction level. Generally speaking mobility models define a separate research domain by themselves [2, 3] with the random waypoint (RWP) mobility model serving as the lowest common denominator of granularity. Specific mobility models then cover individual MANET domains, e.g. Vehicular Ad Hoc Networks (VANETs) [24].

4.1.4 Settings for the case study

Consistent with these considerations, we simulated a number of different setups for the DYMO study. As can be seen from the analysis of the simulation results in Section 4.6, all these setups are needed to unravel possible performance limitations of the ad hoc routing protocol.

For the first set of simulations, each setup used 100 nodes running our implementation of DYMO, of which 99 continuously generated packets addressed to one node located in a corner of the playground acting as a packet sink. We evaluated DYMO's performance for the following combinations of scenarios:

– Nodes were arranged either to form a *10x10 grid* or in a completely *random* manner.
– The playground size was adjusted so that the radio range of a node corresponded to either *one hop* or *three hops* (according to the grid scenario).
– Nodes were completely stationary. Additionally, a dynamic scenario was simulated by moving 10 % of the nodes according to the RWP model.

A screenshot for the *10x10 grid* scenario illustrating the *one hop* distance experiment is provided in Figure 3. All hosts named `host[xy]` participate in the application by generating data messages and transmitting them to the host named `sink`, located in the bottom right corner.

Additionally, we modeled a network consisting of 11 nodes arranged in a straight *line*, with the distance between neighboring nodes corresponding to *one hop*. This scenario is shown in Figure 4. Here, ten nodes are periodically generating payload messages and send them to a single dedicated sink node located at the end of the line, shown on the left.

4.2 Parameter settings and traffic patterns

For any protocol evaluation, it is necessary to precisely identify and document the possible ranges of the protocol parameters. These parameters must either be clearly defined by the application scenario or they need to be analyzed in different simulation runs. Based on this parameter set, the simulation becomes reproducible.

For DYMO, the protocol parameters are depicted in Table 1. This table also includes the values that have been used for the case study. As can be seen, we configured parameters that are suggested by the IETF. The same simulation setup could also be used to find optimal values for specific topology settings.

Aside from varying the network topology as introduced in Section 4.1, we also ran simulations according to three different load models. We used a traffic generator module

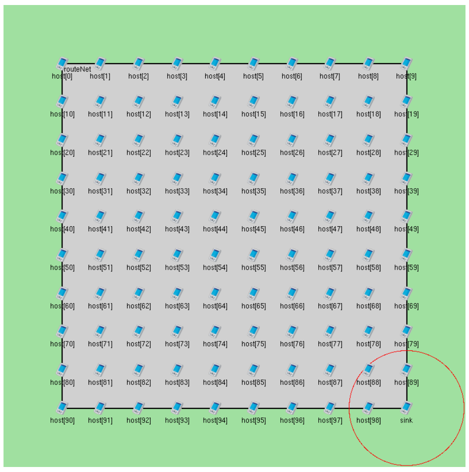


Fig. 3 Screenshot of a *10x10 grid*, *one hop* setup. All nodes send packets to a common sink, drawn in the lower right hand corner together with its communication range.

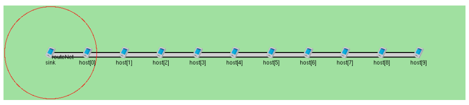


Fig. 4 Screenshot of a *line* setup. All nodes send packets to a common sink, drawn at the left edge together with its communication range.

that is available for OMNeT++. All the application layer packets were generated by this module, which also plays the role of a packet collector consuming these packets at sink nodes. Using a flexible XML-based parameter structure, it can be configured to generate various types of traffic, including the usual constant bit rate (CBR), on-off, and exponentially distributed traffic. For our case study, we did not vary parameters such as the packet size that might be relevant for evaluating routing protocols in low-resource networks such as WSNs. Instead, we set it to a fixed value of 50 Bytes to keep its impact on network load small, but noticeable. Regarding traffic patterns, we evaluated the following configurations:

– all the source nodes send new packets following an exponential distribution with a mean value of 1 s
– as above, but using a mean value of 10 s
– packets are sent following a random, *bursty* pattern, which consists of waiting between zero and five minutes, then sending ten packets spaced 0.49 s apart

The first two configurations have been selected according to the route timeout of 5 s. It is valuable to analyze the behavior of the protocol in a scenario in which the number of data packets ensure that the routes are almost never timed out; as well as a scenario in which the probability for such timeouts, i.e. the need for new route setups, is quite high. The third setup allows evaluating the routing behavior in the average case.

4.3 Relevant performance metrics

A number of performance metrics are relevant for the adequate analysis of network protocols. Our model is capable of recording a number of statistics, such as the number of packets sent and received, or message latencies. The following list names the set of recorded statistical measures that we used to evaluate the overall behavior of the DYMO routing protocol in our case study. Some of them are especially relevant in the domain of wireless ad hoc routing. Whereas this list is perhaps not complete, it covers all the relevant aspects that are influenced by the configured setups w.r.t. different topologies, network densities, mobility configurations, traffic patterns, etc.

Collisions on the MAC Layer In order to make sure that none of the effects discussed in later evaluations occurred due to message loss resulting from collisions in the overloaded shared medium, the first measures that we evaluated were the number of collisions and the number of successfully received transmissions, as observed by the MAC layer.

Loss on the MAC and Physical Layers To determine the amount of messages lost on either the MAC or the physical layer, e.g. because of node movement, we recorded the total number of application-layer messages passed down by the network layer of all nodes to lower layers. We then compared it with the number of messages received by the network layer of the sink and thus obtained the ratio of application messages lost per message generated.

Frequency of Route Setups The traffic overhead induced by DYMO was estimated by relating the number of route request messages to the number of application-layer messages sent by DYMO via established routes.

Data Packets dropped by DYMO As a measure for DYMO's aptitude for finding routes, we compared the amount of data packets dropped at the network layer with the number of packets requested to be transmitted to a particular destination.

Route Discovery Delay Another immediate performance measure we evaluated was the delay between a message being queued for delivery by DYMO and its removal from the queue when a route was established. It should be noted that this measure does not reflect cases where a message was not queued because a route was already known; neither does it reflect cases where a message was discarded because no route could be discovered in the time interval set.

End-to-end Delay From a user point of view, one of the most important measures is the delay of messages as perceived by the application layer. Observations of this value are directly related to all previously mentioned performance measures and are discussed in Section 4.6.6.

Protocol specific metrics For a more detailed analysis of the spatial load distribution, we also examined the following measures in dependence of the recording node's position:

– the number of generated RREQs per second, i.e. not including RREQs forwarded on behalf of other nodes
– the number of sent DYMO messages per second, i.e. the rate of RREQs, RREPs and RERRs generated or forwarded on behalf of other nodes
– the number of sent payload messages per second, i.e. including messages forwarded on behalf of other nodes

4.4 Validation

To gain confidence that the implemented simulation program does indeed represent the system which was intended to be modeled with sufficient accuracy, the simulation has to be validated. In the vast majority of cases, a proof that the simulation model works as intended is too costly to acquire. Instead, evaluations and tests are performed to increase the confidence in the model's correctness. These evaluations should be continued until a sufficient level of confidence has been achieved.

A number of different aspects of a simulation model should be subject to validation. First, it needs to be ensured that the data used to build the simulation model and the data used as input to the simulation experiments are valid. This means that a sufficient amount of data has to be made available, and that the data have to be appropriate and accurate. Second, to validate the conceptual model, it needs to be verified that the underlying assumptions and the representation of the system structure and its internal relationships are correct. Third, to verify the correctness of the model's computer implementation, testing methods from software engineering can be applied. Finally, and most importantly, it needs to be validated how the model reacts to its inputs, i.e. the model's output behavior.

Several validation methods are discussed in the literature (see, for example, [21]). In the following, we list some of the methods which are especially applicable to simulation models of network protocols.

– *Animation* of a simulation run can yield information about the types of messages in the network, their sources and destinations, and their contents.
– *Testing extreme conditions and degenerate cases* should be a part of any software development process. An example in a routing protocol study could be to verify that route request messages are only sent initially if the route timeout is infinite.
– *Face validity* can be ensured by asking system experts whether the model itself and/or its output behavior seems reasonable.
– *Operational graphics* display the values of performance measures in a time series (either during or after a simulation run). This can be used to evaluate which performance measures give a good indication when the system reaches a steady state. This can be used to select performance measures for simulation control as described in Section 4.5.
– *Parameter variability and sensitivity analysis* consists of systematically varying the simulation's input parameters and analyzing how the output reacts to the variations.

Whichever methods are used to validate a simulation model, the most important part is to document the entire validation process. This documentation is crucial to convince others of the model's correctness. The documentation should include information on which methods were applied, the data and parameters used, and the results of each validation step.

For the model presented in this paper, we showcase a part of our validation documentation in Section 4.6. For that part, we combined parameter variability analysis with face validation to judge if the model's reactions to varying input parameters meet the expectations of an expert.

4.5 Simulation control

Two aspects of simulation control need to be considered in the context of automated simulation execution: How to run simulations and when to terminate each run.

The simplest approach to running simulations would be to execute all simulations sequentially on one computer. However, simulations may take a long time to complete. It is therefore desirable to distribute the simulation execution on several computers.

One method to distribute simulations is to divide up the simulation model and run each part of it on dedicated hardware. Parallel Discrete Event Simulation (PDES) is a common approach to increase the speed and thus reduce the runtime of simulations to a manageable level. OMNeT++ offers support for PDES by automatically mapping different parts of a model to a number of logical processes that communicate via files, named pipes or a Message Passing Interface (MPI). The user manual, however, lists severe constraints which impede the use of PDES in OMNeT++. Unless, for example, two modules are mapped to the same logical process, data between them may be exchanged only via messages and those messages may not be sent directly to a submodule. No global variables may be used and all link delays must be previously known. The restriction that has the largest impact of PDES on ad hoc networks, however, is that the topology of a network must be previously known and must remain static throughout the simulation.

A different approach to parallelization can be taken based on the assumption that for a simulation to yield meaningful results, a single simulation run has to be repeated several times using independent random numbers for each replication [16]. When executing simulations with Multiple Replications In Parallel (MRIP) [17], all replications of a simulation run are executed independent from each other, e.g. on separate machines, and the individual instances run non-parallelized. This means that simulation models do not suffer from the restrictions PDES incurs.

The second aspect of simulation control that needs to be addressed is determining when enough data has been gathered and simulations can be terminated. This is especially difficult when multiple simulation runs are executed in parallel. The Akaroa2 simulation manager [9, 25] allows MRIP execution and exploits the fact that in steady-state simulations, results from multiple runs that are being executed in parallel can be correlated on the fly using automated sequential analysis.

In Akaroa-controlled simulations, a single manager process controls simulation engines running on standard hardware, communicating with them via a TCP connection. Periodically, simulation engines report the state of selected variables back to the manager process, which correlates these observations and evaluates the mean value of variables, the confidence, and the precision, taking care to discard observations made before a run reached its steady state. All simulation runs are terminated when a sufficient number of observations have been made.

The question remains how the metrics reported to Akaroa should be chosen. As explained above, the purpose of reporting metrics to Akaroa is to enable to terminate a simulation run after collecting enough observations. Therefore, any metrics used for this purpose should somehow reflect that a simulation has reached a steady state. In other words, the variations in any suitable metric should diminish as the system approaches a steady state.

There are several types of metrics that are generally not fit for this purpose. Summary statistics, like packet or collision counts, are not appropriate because they are typically reported just once. Rare events that only occur very few times during a simulation run are also not appropriate because the sample size to calculate variations is too small. Any metric that averages a certain event type over time is not appropriate because of the smoothing that results from averaging.

So, in general, we are looking for a metric that is recorded or updated frequently, is not averaged over time, and reflects a system's reaching of a steady state. In ad hoc routing, message latencies are excellent candidates for this. The other metrics introduced in Section 4.3 fail to meet at least one of these conditions.

There are two plausible choices for message latencies: data message delays and route management message delays. In DYMO, route management messages consist of route requests, route replies and route errors. The problem is that route management messages may become scarce in some networks once the nodes have populated their routing tables. Using route management message latencies for simulation control is therefore problematic. Data messages do not have this problem. Data messages occur frequently during the entire course of the simulation, and their latencies can be recorded per packet and are therefore not averaged. Still, in a multi-hop network, it is problematic to directly use the data message latencies between the originating node and its destination because the latencies will vary greatly depending on the hop count the message needs to travel to its destination. This can, however, easily be overcome by recording the data message latency for every single hop the message uses.

Therefore, for the presented evaluations, Akaroa2 was configured to estimate the *data delay per hop* with confidence and precision levels of 95 % and 5 %, respectively.

4.6 Analysis of simulation results

In this section we illustrate how details of the examined routing protocol's behavior can easily be inferred from the results of simulations. The parameter space of the simulations was configured according to the principles outlined earlier in this section.

Results for each of the metrics discussed in Section 4.3 are drawn as boxplots: For each data set, a box is drawn from the first quartile to the third quartile, and the median is marked with a thick line. Additional whiskers extend from the edges of the box towards the minimum and maximum of the data set, but no further than 1.5 times the interquartile range. Data points outside the range of box and whiskers are considered outliers and drawn separately, as small circles. Alternatively, the mean value of observations is depicted in the form of one small square.

4.6.1 Collisions on MAC Layer

Figure 5 shows the ratio of MAC collisions per link-layer packet sent for a scenario of non-moving, randomly deployed nodes. As can be seen, only in the case of high node

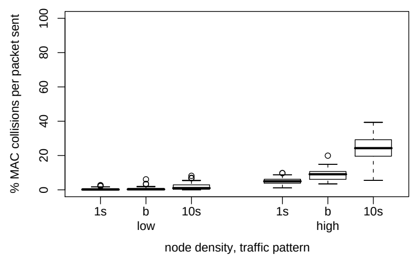


Fig. 5 Collisions on MAC layer. Scenario: random deployment, no mobile nodes

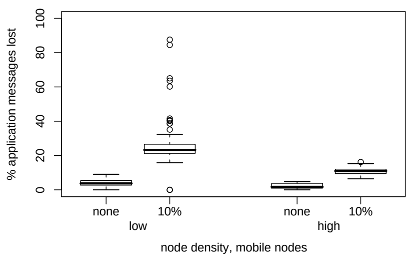


Fig. 6 Loss on MAC and physical layer. Scenario: random deployment

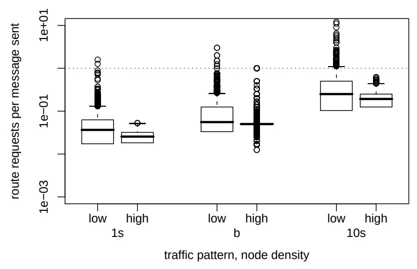


Fig. 7 Frequency of route setups. Scenario: random deployment, no mobile nodes

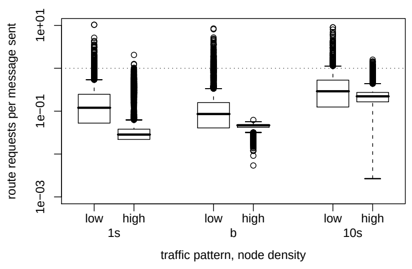


Fig. 8 Frequency of route setups. Scenario: random deployment, 10 % mobile nodes

density and a mean interval between two application-layer messages of 10 s exceeds the ratio 20 %. Thus, observations made on higher protocol levels in these scenarios need to be disregarded in further evaluations. MAC collisions for other scenarios were much more seldom, reaching only insignificant ratios. Similar results were obtained if some nodes moved according to a random waypoint model and/or if nodes were arranged in a grid.

4.6.2 Loss on MAC and physical layer

As shown in Figure 6, insignificant message loss with ratios of well below 10 % can be observed in scenarios where the nodes' positions remained static. Only in dynamic scenarios, as simulated by moving 10 % of the nodes according to a random mobility model, the percentage of packets lost rises noticeably.

Both in this setup, as well as when nodes were arranged in a grid, we observed that in scenarios with high node density, node mobility had less impact on packet loss. Here, packet loss on the MAC and physical layer was approximately cut in half compared to the low density scenarios.

4.6.3 Frequency of route setups

As depicted in Figure 7, representative for deployment of nodes both in a random and grid-like fashion, the number of route request messages exchanged per generated application-layer message decreased slightly if packets were sent at an interval that

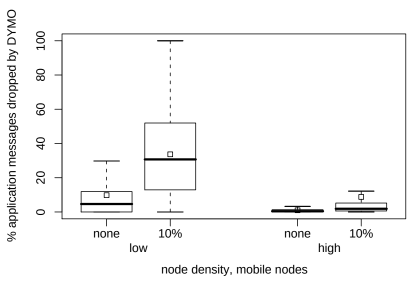


Fig. 9 Data packets dropped by DYMO. Scenario: random deployment, 1 s mean inter-packet spacing

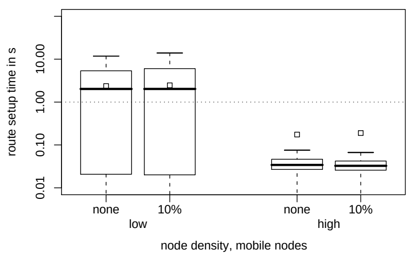


Fig. 10 Route discovery delay. Scenario: random deployment.

could keep established routes from expiring. In these static scenarios, a higher node density slightly reduced the frequency of route setups because it improved node connectivity, thus shortening and stabilizing routes.

However, in the dynamic scenarios shown in Figure 8, where 10 % of all nodes moved according to a random waypoint model, the higher node density played a key role in reducing the number of route requests. Here, a larger number of potential routes to the sink meant a higher probability that the chosen route included more static nodes, thus reducing the probability of this route breaking if one of the involved nodes moved out of communication range.

4.6.4 Data packets dropped by DYMO

Figure 9 shows the results of this comparison. Starting with this figure, mean delays are shown as squares; outliers are not plotted.

In the stationary scenario, almost no packets got lost due to problems at the network layer. The few outliers result from particular nodes being in principle unable to establish a path towards the sink in the random deployment. In the mobile scenarios, the probability to successfully set up a path *and* to transmit messages essentially relies on the probability of route failures. Obviously, the *low density* scenario tends to show a higher number of route failures compared to the *high density* example. Again, similar results were obtained for other inter-packet spacings and/or if nodes were arranged in a grid.

4.6.5 Route discovery delay

As shown in Figure 10, this delay mostly depends on the length of the path, i.e. in the low-density scenarios much higher delays can be observed compared to the high-density scenarios. Similar results were obtained if nodes were arranged in a grid.

4.6.6 End-to-end delay

Figures 11 and 12 show the results of our simulation experiments. In order to produce comparable results, both figures depict the mean delay per hop, i.e. the end-to-end latency divided by the number of hops for this particular transmission.

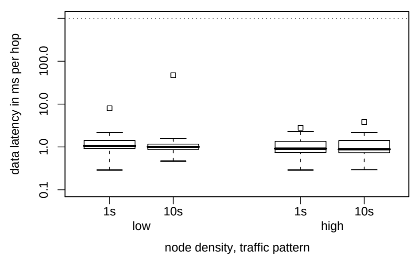


Fig. 11 End-to-end delay. Scenario: random deployment, no mobile nodes

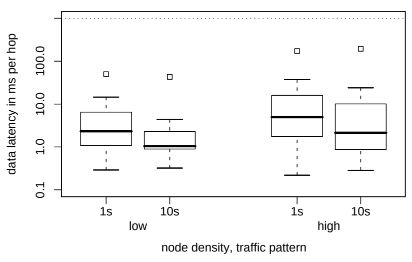


Fig. 12 End-to-end delay. Scenario: random deployment, 10 % mobile nodes

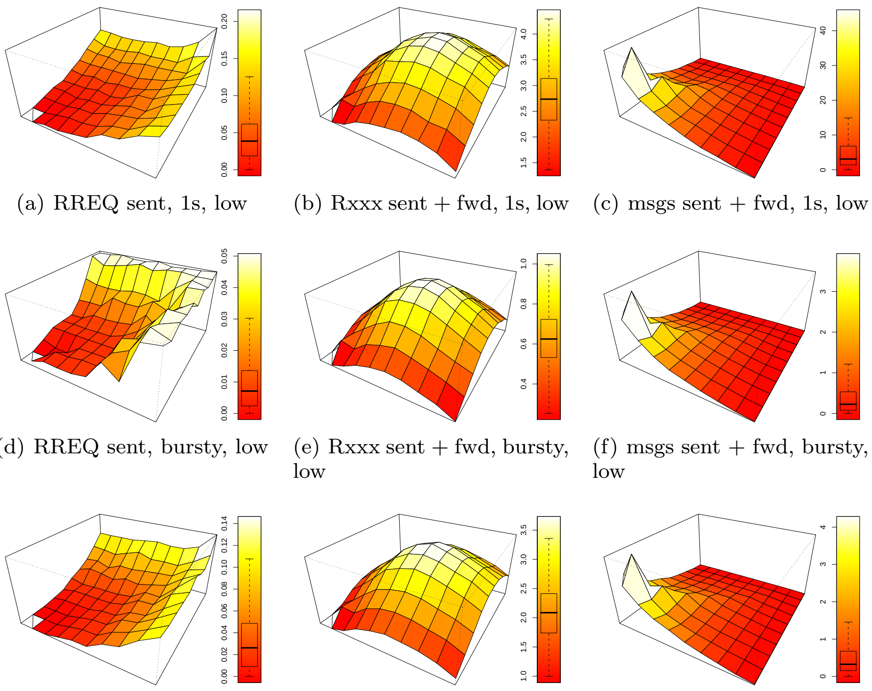
Without looking at particular numbers, which mainly depend on the specific network scenario, we need to discuss a number of effects that became visible in these figures. Considering the non-mobile case shown in Figure 11 first and comparing the traffic scenarios with one and ten seconds inter-packet time, we see that the average per-hop delay increases. This effect can be explained by the route timeouts used by DYMO in our experiment. In the *ten seconds* example, DYMO has to set up a route for almost each packet because the available routes have timed out. Thus, each time an additional route setup delay adds to the packet transmission delay. Thanks to the route response messages created on behalf of the destination, the median is equal in both cases.

Comparing the results for the mobile scenario, the effect of node mobility seems to be partly reversed, i.e. the measured delays for the *ten seconds* case are often smaller compared to the *one second* case. Again, this can be explained by the route timeouts. In the *one second* example, DYMO will try to re-use already discovered routes more often, but this time frequently fail because one or more of the involved nodes have moved out of communication range. Analogous to the effect discussed in Section 4.6.3, this reversed effect is more pronounced if node densities are lower, as this means a higher probability of a mobile node participating in a route.

4.6.7 Spatial load distribution in a grid scenario

Aside from the standard performance metrics discussed above, we analyzed the spatial load distribution in the network in order to get more information about the working behavior of DYMO and to identify possible bottlenecks. In Figures 13 and 14, the spatial load distributions in *low density* and *high density* scenarios are depicted for the *one second*, *bursty*, and *ten second* traffic patterns. Shown in the figure are the following three measures: the number of generated RREQs per second, the number of sent DYMO messages per second, i.e. the number of RREQ, RREP, and RERR messages, and the number of sent payload messages per second. As could be expected, results of simulations using the *one second* traffic pattern yielded comparable results to those using the *ten second* pattern. For this reason results from the latter are not discussed in detail.

As can be seen in Figure 13(a), the nodes' number of newly generated RREQs per second (and, hence, the number of necessary RREQs per payload message sent) decreases towards the sink. This behavior results from a key property of DYMO –

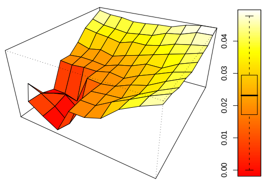


(a) RREQ sent, 1s, low
(b) Rxxx sent + fwd, 1s, low
(c) msgs sent + fwd, 1s, low
(d) RREQ sent, bursty, low
(e) Rxxx sent + fwd, bursty, low
(f) msgs sent + fwd, bursty, low
(g) RREQ sent, 10s, low
(h) Rxxx sent + fwd, 10s, low
(i) msgs sent + fwd, 10s, low

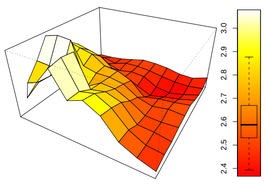
Fig. 13 Spatial load distribution in the 10x10 grid for varying packet spacing (1s, bursty, 10s). Depicted is the low density scenario

to learn routes from received RREQs and RREPs. Therefore, the probability that a node already stored a suitable route prior to initiating a data transfer increases for nodes close to the sink. The same effect can also be observed in the *bursty* scenarios depicted in Figures 13(d) and 14(d), but with additional random variations introduced by less deterministic behavior. An interesting effect can be observed in the *high density* scenarios shown in Figures 14(a) and 14(d): For nodes closer to the sink than its maximum communication radius, the number of sent RREQs per second is actually increasing, as all RREPs of the sink are sent via unicast and, hence, routes to the sink time out in nodes frequently "skipped" in the chain of RREPs back to source nodes.

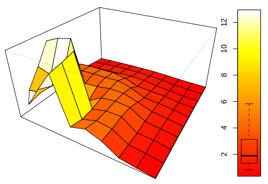
Plotted in Figure 13(b) is the nodes' number of transmitted DYMO messages per second. As can be seen, Intermediate DYMO Router RREP Creation helps to confine RREQs, preventing flooding both near the sink, where nodes were frequently able to answer on behalf of it, as well as flooding RREQs towards the far edges of the network. The effects of this mechanism are much less pronounced in a *high density* scenario, as depicted in Figure 14(b). Just like we observed in a *low density* scenario with Intermediate DYMO Router RREP Creation turned off, DYMO load here is almost uniformly distributed with only the relaying of RREPs contributing to a small rise of load towards the sink. Similar, but much less pronounced effects can be seen in the *bursty* scenarios shown in Figures 13(e) and 14(e).



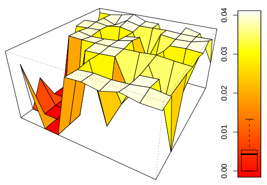
(a) RREQ sent, 1s, high



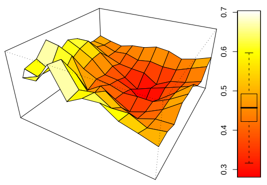
(b) Rxxx sent + fwd, 1s, high



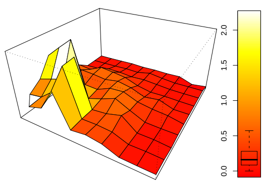
(c) msgs sent + fwd, 1s, high



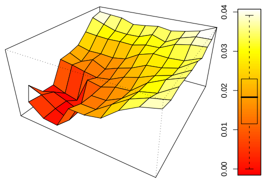
(d) RREQ sent, bursty, high



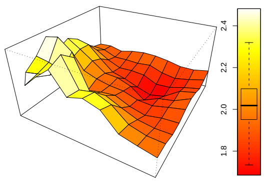
(e) Rxxx sent + fwd, bursty, high



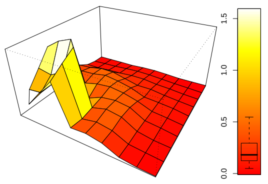
(f) msgs sent + fwd, bursty, high



(g) RREQ sent, 10s spacing



(h) Rxxx sent + fwd, 10s spacing



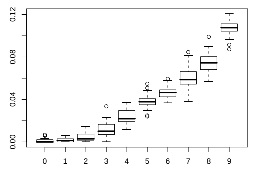
(i) msgs sent + fwd, 10s spacing

Fig. 14 Spatial load distribution in the 10x10 grid for varying packet spacing (1s, bursty, 10s). Depicted is the high density scenario

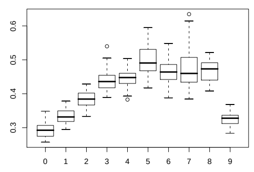
Lastly, Figure 13(c) shows the load distribution of payload data transmission. As all nodes uniformly contribute to the generation of messages and forward these messages in a directed way towards the sink node, a permanent increase of the load towards the sink can be seen. In the *high density* scenario shown in Figure 14(c), an interesting effect can be observed: Nodes with a distance to the sink of exactly its maximum communication radius relay much more payload messages than those nearer to it, because they are exactly one hop away and thus far more probable to be part of the shortest route between the sink and an arbitrary source, reinforced by the fact that responses of these nodes on behalf of the sink will always contribute to an optimal route. Identical effects can be observed in the *bursty* scenarios shown in Figures 13(f) and 14(f), respectively.

4.6.8 Spatial load distribution in a line scenario

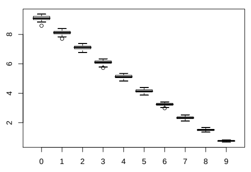
In order to verify the results, we created a dedicated scenario, which is commonly used to evaluate the performance of protocols in wireless ad hoc networks. In this scenario, 11 nodes are placed in a straight line. Ten of these nodes generate data using the same traffic characteristics as used in the grid scenarios. The 11th node is used as a sink to



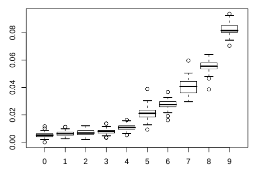
(a) RREQ sent, 1s spacing



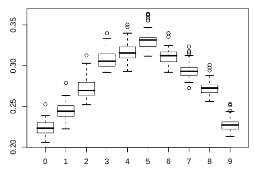
(b) Rxxx sent + fwd, 1s spacing



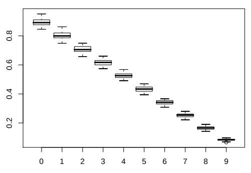
(c) msgs sent + fwd, 1s spacing



(d) RREQ sent, 10s spacing



(e) Rxxx sent + fwd, 10s spacing



(f) msgs sent + fwd, 10s spacing

Fig. 15 Spatial load distribution in the line scenario

transmit all data messages to. In the presented figures, the sink node is located on the left.

The measurement results are summarized in Figure 15, which shows the results for inter-packet spacings of 1 s and 10 s. As expected both configurations yielded comparable results. Plotted in Figures 15(a) and 15(d) is the participating nodes' number of generated RREQs per second, i.e. the number of necessary RREQs to set up routes towards the sink. With a mean inter-packet spacing of 1 s this measure directly corresponds to the average number of necessary RREQs per payload message sent. It can be seen that the number of RREQs increases with the distance to the sink. This validates the results from the grid scenario. Depending on the data rate, routes may time out and additional RREQs are necessary if the inter-packet time is greater than the route timeouts used by DYMO.

Figures 15(b) and 15(e) show the number of all DYMO messages transmitted, i.e. the number of RREQs, RREPs and RERRs either generated or forwarded on behalf of other nodes. As in the *10x10 grid* scenarios, Intermediate DYMO Router RREP Creation helped confine the flooding of RREQs and thus greatly reduced the number of relayed DYMO messages near the sink and at the far edge of the network. Again, when Intermediate DYMO Router RREP Creation was turned off, an almost uniform distribution of relayed DYMO messages was observed.

Figures 15(c) and 15(f) show the number of payload messages generated and forwarded by all DYMO nodes in the network. As expected, this number linearly increases towards the sink node. Again, the results from the grid scenario, where numbers increased quadratically, are supported by this measure.

5 Conclusion

Based on a case study, we presented and discussed necessary steps for the comprehensive evaluation of ad hoc routing protocols in OMNeT++. In particular, we used our protocol implementation of the DYMO routing protocol to go through each step of the process of simulation-based performance evaluation. We outlined a number of typical and necessary measures that are relevant for evaluating the performance of ad hoc routing protocols in general and for MANET routing protocols in particular. We further presented a set of common criteria that may serve as guidelines for meaningful simulative evaluations of ad hoc routing protocols and concluded the case study by demonstrating how certain peculiarities of the DYMO routing protocol can be discovered and explained using this approach.

We are currently using the presented implementations of DYMO, both the variant that reduces the protocol stack to only the network layer, as well as the full-featured one, to evaluate the applicability and performance of DYMO in MANETs, focusing on VANET scenarios [23, 24]. Especially for application in VANETs, the demands on the path setup performance and the adaptivity to dynamic topology changes proved challenging for ad hoc routing techniques such as DYMO.

References

1. K. Akkaya and M. Younis. A Survey of Routing Protocols in Wireless Sensor Networks. *Elsevier Ad Hoc Networks*, 3(3):325–349, 2005.
2. F. Bai and A. Helmy. A Survey of Mobility Models in Wireless Adhoc Networks. In *Wireless Ad Hoc and Sensor Networks*. Kluwer Academic Publishers, 2004.
3. T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications and Mobile Computing, Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.
4. I. Chakeres and C. Perkins. Dynamic MANET On-Demand (DYMO) Routing. Internet-Draft (work in progress) draft-ietf-manet-dymo-11.txt, IETF, November 2007.
5. T. H. Clausen, C. Dearlove, and J. W. Dean. MANET Neighborhood Discovery Protocol (NHDP). Internet-Draft (work in progress) draft-ietf-manet-nhdp-05.txt, IETF, December 2007.
6. T. H. Clausen, C. M. Dearlove, J. W. Dean, and C. Adjih. Generalized MANET Packet/Message Format. Internet-Draft (work in progress) draft-ietf-manet-packetbb-11.txt, IETF, November 2007.
7. F. Dressler. *Self-Organization in Sensor and Actor Networks.* John Wiley & Sons, December 2007.
8. W. Drytkiewicz, S. Sroka, V. Handziski, A. Köpke, and H. Karl. A Mobility Framework for OMNeT++. In *3rd International OMNeT++ Workshop, at Budapest University of Technology and Economics, Department of Telecommunications*, Budapest, Hungary, January 2003.
9. G. Ewing, K. Pawlikowski, and D. McNickle. Akaroa2: Exploiting Network Computing by Distributed Stochastic Simulation. In *European Simulation Multiconference (ESM 1999)*, pages 175–181, Warsaw, Poland, 1999.
10. J. J. Galvez and P. M. Ruiz. Design and Performance Evaluation of Multipath Extensions for the DYMO Protocol. In *32nd IEEE Conference on Local Computer Networks*, pages 885–892, Dublin, Ireland, 2007.
11. Z. J. Haas and M. R. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol. *IEEE/ACM Transactions on Networking (TON)*, 9:427–438, August 2001.
12. X. Hong, K. Xu, and M. Gerla. Scalable Routing Protocols for Mobile Ad Hoc Networks. *IEEE Network*, 16:11–21, July/August 2002.
13. A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen. Scalable Routing Strategies for Ad Hoc Wireless Networks. *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, 17(8):1369–1379, August 1999.
14. D. B. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. In *Workshop on Mobile Computing Systems and Applications*, pages 158–163, Santa Cruz, CA, December 1994. IEEE,.
15. C. S. R. Murthy and B. S. Manoj. *Ad Hoc Wireless Networks.* Prentice Hall PTR, 2004.

16. K. Pawlikowski, H.-D. Jeong, and J.-S. R. Lee. On Credibility of Simulation Studies Of Telecommunication Networks. *IEEE Communications Magazine*, pages 132–139, 2002.
17. K. Pawlikowski, V. W. C. Yau, and D. McNickle. Distributed stochastic discrete-event simulation in parallel time streams. In *26th Winter Simulation Conference (WSC '94)*, pages 723–730, San Diego, CA, USA, 1994. Society for Computer Simulation International.
18. C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *Computer Communications Review*, pages 234–244, 1994.
19. C. E. Perkins and E. M. Royer. Ad hoc On-Demand Distance Vector Routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.
20. R. Rajaraman. Topology control and routing in ad hoc networks: a survey. *ACM SIGACT News*, 33(2):60–73, June 2002.
21. R. G. Sargent. Verification and validation of simulation models. In *39th Winter Simulation Conference (WSC 2007)*, pages 124–137, Washington, D.C., 2007. IEEE.
22. C. Sommer, I. Dietrich, and F. Dressler. A Simulation Model of DYMO for Ad Hoc Routing in OMNeT++. In *1st ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008): 1st ACM/ICST International Workshop on OMNeT++ (OMNeT++ 2008)*, Marseille, France, March 2008. ACM.
23. C. Sommer and F. Dressler. The DYMO Routing Protocol in VANET Scenarios. In *66th IEEE Vehicular Technology Conference (VTC2007-Fall)*, pages 16–20, Baltimore, Maryland, USA, September/October 2007. IEEE.
24. C. Sommer, Z. Yao, R. German, and F. Dressler. On the Need for Bidirectional Coupling of Road Traffic Microsimulation and Network Simulation. In *9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM Mobihoc 2008): 1st ACM International Workshop on Mobility Models for Networking Research (Mobility-Models'08)*, pages 41–48, Hong Kong, China, May 2008. ACM.
25. S. Sroka and H. Karl. Using Akaroa2 with OMNeT++. In *2nd International OMNeT++ Workshop*, Berlin, Germany, January 2002.
26. A. Varga. The OMNeT++ Discrete Event Simulation System. In *European Simulation Multiconference (ESM 2001)*, Prague, Czech Republic, June 2001.
27. A. Varga and R. Hornig. An overview of the OMNeT++ simulation environment. In *1st ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008)*, Marseille, France, March 2008. ACM.