# A Simulation Model of DYMO for Ad Hoc Routing in OMNeT++

Christoph Sommer Isabel Dietrich Falko Dressler

Computer Networks and Communication Systems
Department of Computer Science, University of Erlangen, Germany
{christoph.sommer,isabel.dietrich,dressler}@informatik.uni-erlangen.de

## 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, they are dealing with new challenges in the context of ad hoc routing. In this paper, we present our new implementation of the ad hoc routing protocol Dynamic MANET On Demand (DYMO) for the popular discrete event simulation environment OMNeT++. DYMO offers adaptation to changing network topology and determines unicast routes between nodes within the network on demand. Based on the developed model, we performed several simulation experiments to analyze the performance of DYMO with respect to changing network size and traffic conditions. The implementation of DYMO and the results of the simulation experiments are described in this paper.

## 1. INTRODUCTION

Research on Mobile Ad Hoc Networks (MANETs) addresses many objectives such as scalability and energy efficiency of ad hoc routing techniques. A number of protocols have been proposed in the last decade [1, 9, 10]. 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).

The developments of protocols and solutions are covering multiple problem domains. It turned out that in most MANET scenarios, reactive routing protocols outperform proactive approaches to a certain extent. The probably best known protocol in this context is Ad Hoc on Demand Distance Vector (AODV) [3, 11, 12]. This protocol searches routes through the network on demand when data needs to be transmitted. AODV has been intensively studied in the last years [6]. Based on all these findings, a new protocol has been developed, the Dynamic MANET On Demand (DYMO) routing protocol [2].

In this paper, we describe a simulation model of DYMO, which we developed for the OMNeT++ simulation environment[1]. OMNeT++ [16] already provides the framework for comprehensive simulation setups. A great number of network protocols used in the context of MANETs are available as simulation models, including the complete TCP/IP stack and an AODV implementation. Based on the implemented DYMO model, we performed a comprehensive performance evaluation to study the effects of DYMO. The results of this analysis are also provided in this report. Due to space restrictions, we cannot provide all the results in this paper. In particular, we have to refer to an earlier technical report that outlines the performance measures for an older version of DYMO [7].


*OMNeT++ 2008* March 3, 2008, Marseille, France


## 2. DYNAMIC MANET ON DEMAND

DYMO is the most recent 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.

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 as-

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

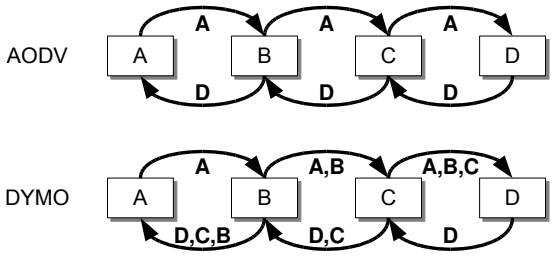



**Figure 1: Routing information dissemination in AODV and DYMO**

sociated 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 an 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 chose to allow intermediate nodes the generation of RREPs on behalf of an 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 [4] 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 a 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 [5] and offers ways of dealing with unsupported elements in a sensible way.

Regarding related work, several DYMO implementations in different languages and for different systems exist, most notably *DYMOUM*, a GPL implementation for use in both the Linux kernel and ns-2. Also developed for use in the Linux kernel are *NIST-DYMO* (public domain), *EK-DYMO* (closed source) and *DYMO-AU* (GPL, written in LUA and C). Finally, there exist *TYMO* for TinyOS (GPL, written in nesC) and an old implementation of *DYMO for OPNET* (proprietary license).

# 3. THE DYMO SIMULATION MODEL

In this section, we outline the implementation of our simulation model of DYMO. The model is freely available for use and distribution under the terms of the GPL.

## 3.1 Simulation environment – OMNeT++

We modeled DYMO for use as a simulation model in the OMNeT++ 3.4b2 [16] tool, a simulation environment free for academic use, and its *INET Framework* 20061020 extension, a set of simulation modules released under the GPL. The OMNeT++ engine runs discrete, event-driven simulations of communicating nodes on a wide variety of platforms and is getting increasingly popular in the field of network simulation. It is also part of the SPEC CPU2006[2] benchmark suite released in August 2006.

Scenarios in OMNeT++ are represented by a hierarchy of reusable modules written in C++. Modules' relationships and communication links are stored as *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* provides a set of OMNeT++ modules that represent various layers of the Internet protocol suite, e.g. the TCP, UDP, IPv4, and ARP protocols. It also provides modules that allow the modeling of spatial relations of mobile nodes and IEEE 802.11 transmissions between them.

## 3.2 Implementation

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.

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*. 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 this document.

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 the 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

[2] `http://www.spec.org/cpu2006/`

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 | $10\,s^{-1}$ |
| RREQ_BURST_LIMIT | 3 RREQs |
| RREQ_WAIT_TIME | 2 s |
| RREQ_TRIES | 3 |

TTL 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.

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

# 4. PERFORMANCE ANALYSIS

We performed a number of simulation experiments in order to check our implementation of DYMO, as well as to evaluate the protocol in different MANET scenarios and under different traffic conditions. In the following subsections, we outline the simulation setup and the selected performance metrics. Finally, we discuss the obtained performance measures.

## 4.1 Simulation setup

For the performance analysis, we simulated a number of different setups, each consisting of 100 nodes running our implementation of DYMO, 99 of which 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:

1. Nodes were arranged either to form a *10x10 grid* or in a completely *random* manner.
2. The playground size was adjusted so that the average distance between neighboring nodes corresponded to either *one hop* or *three hops* (according to the grid scenario).
3. Nodes sent a new packet either following an exponential distribution with a mean value of *one second* or a mean value of *ten seconds*, or following a random, *bursty* pattern, which consisted of waiting between zero and five minutes, then sending ten packets spaced 0.49 s apart.

A screenshot for the *10x10 grid* scenario illustrating the *one hop* distance experiment is provided in Figure 2. 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.

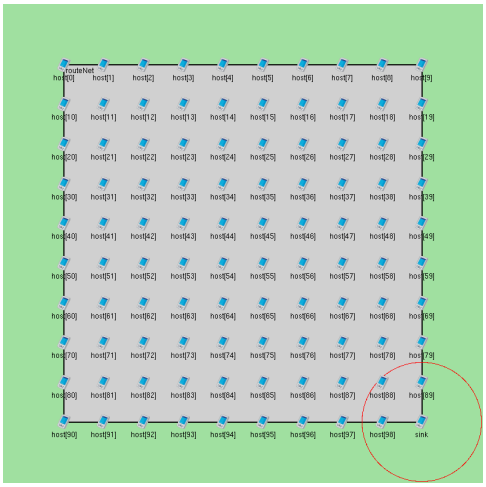


**Figure 2: Screenshot of a *10x10 grid*, *one hop* setup**

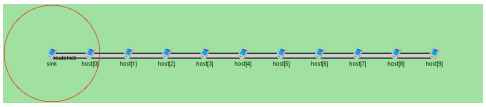


**Figure 3: Screenshot of a *line* setup**

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 3. 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. The time between two packets is exponentially distributed with mean values of *one second* and *ten seconds*, respectively.

## 4.2 Analyzed performance metrics and simulation control

Our model is capable of recording a number of statistics, such as the number of packets sent and received, or message latencies. Overall behavior of the DYMO routing protocol was then examined by recording the following statistical measures:

- Collisions on MAC Layer
- Loss on MAC and Physical Layer
- Frequency of Route Setups
- Data Packets dropped by DYMO
- Route Discovery Delay

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

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

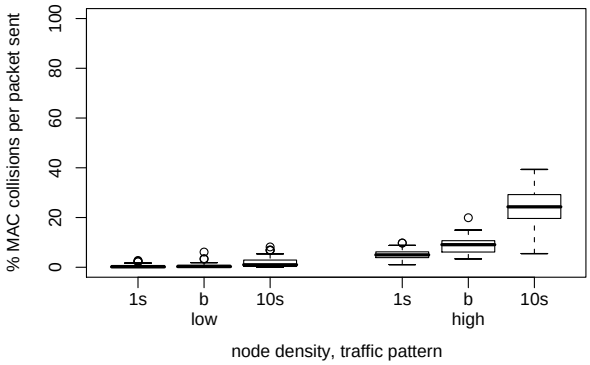



**Figure 4: Collisions on MAC layer. Scenario: random deployment, no mobile nodes**

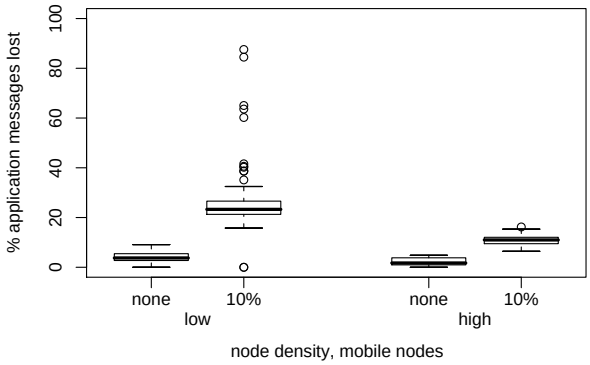



**Figure 5: Loss on MAC and physical layer. Scenario: random deployment**

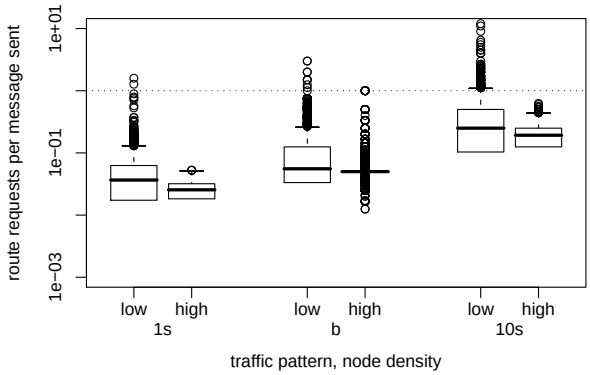



**Figure 6: Frequency of route setups. Scenario: random deployment, no mobile nodes**

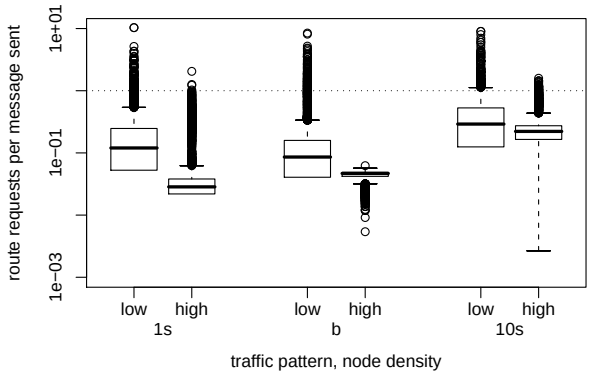



**Figure 7: Frequency of route setups. Scenario: random deployment, 10 % mobile nodes**

- number of sent payload messages per second, i.e. including messages forwarded on behalf of other nodes

All simulations were performed under the control of the Akaroa2 simulation manager [8, 15], a tool "aimed at improving the credibility of results from quantitative stochastic simulation using automated sequential analysis". For each simulation setup, multiple simulation runs were conducted in parallel until the measured *data delay per hop* could be determined with Akaroa's confidence and precision exceeding 95 % and 5 %, respectively.

## 4.3 Collisions on MAC Layer

In order to make sure that none of the effects discussed in later sections 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.

Figure 4 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 density and a mean interval between two application-layer messages of 10 s exceeds the ratio 20 %. 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.

## 4.4 Loss on MAC and physical layer

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.

As shown in Figure 5, 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.

We observed that in scenarios with high node density, node mobility had less impact on packet loss, where packet loss on the MAC and physical layer was approximately cut in half compared to the low density scenarios. Similar results were obtained if nodes were arranged in a grid.

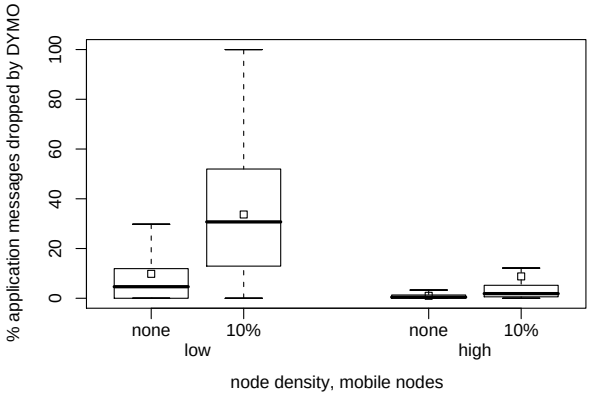



**Figure 8: Data packets dropped by DYMO. Scenario: random deployment, 1 s mean inter-packet spacing**

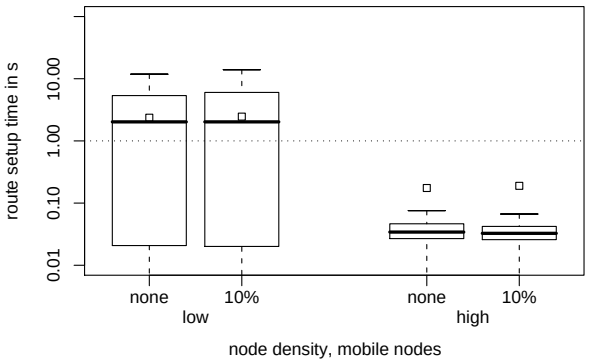



**Figure 9: Route discovery delay. Scenario: random deployment.**

## 4.5 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.

As depicted in Figure 6, the number of route request messages exchanged per generated application-layer message decreased slightly if packets were sent at an interval that could keep established routes from expiring. In these static scenarios, node density played a minor role insofar as it improved node connectivity, shortening and stabilizing routes and thus slightly reducing the frequency of route setups.

However, in the dynamic scenarios shown in Figure 7, 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. Similar results were obtained if nodes were arranged in a grid.

## 4.6 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.

Figure 8 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.

## 4.7 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.

As shown in Figure 9, 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.

## 4.8 End-to-end delay

From a user point of view, one of the most important measures is the delay of messages as observed by the application. Figures 10 and 11 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.

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 10 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

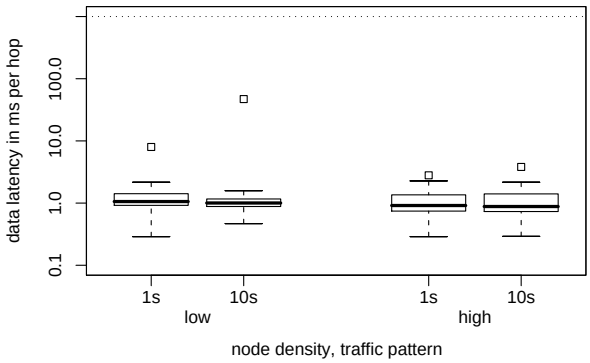



**Figure 10: End-to-end delay. Scenario: random deployment, no mobile nodes**

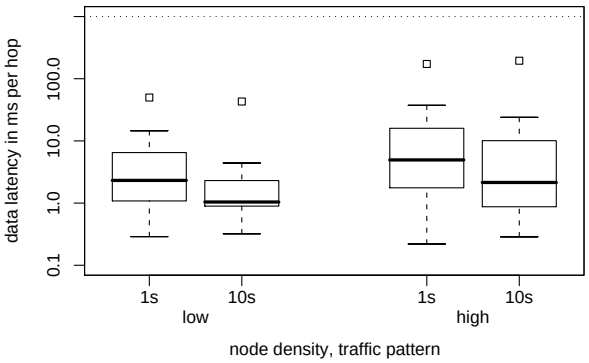



**Figure 11: End-to-end delay. Scenario: random deployment, 10 % mobile nodes**

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 4.5, 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.

## 4.9 Spatial load distribution in the "100 nodes 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 Figure 12, the spatial load distributions in *low density* and *high density* scenarios are depicted for the *one second* and *bursty* 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 can be seen in Figure 12(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 – 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 12(d) and 12(j), but with additional random variations introduced by less deterministic behavior. An interesting effect can be observed in the *high density* scenarios shown in Figures 12(g) and 12(j): 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.

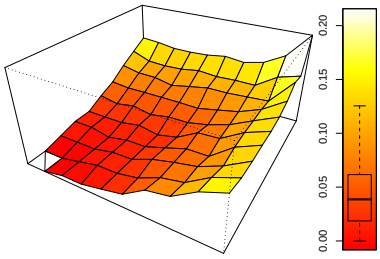
Plotted in Figure 12(b) is the nodes' number of transmitted DYMO messages per second. As can be seen, Intermediate DYMO Router RREP Creation helps to confine RREQs, preventing flooding both near the sink, where nodes were frequently able to answer on behalf of it, as well as flooding RREQs towards the far edges of the network. The effects of this mechanism are much less pronounced in a *high density* scenario, as depicted in Figure 12(h). 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 12(e) and 12(k).

Lastly, Figure 12(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 12(i), 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 12(f) and 12(l), respectively.

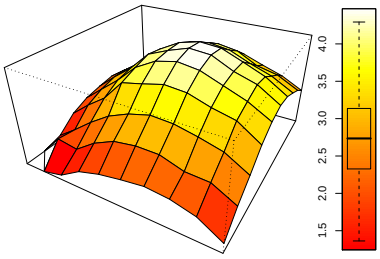
## 4.10 Spatial load distribution in the "11 nodes in a line" scenario

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

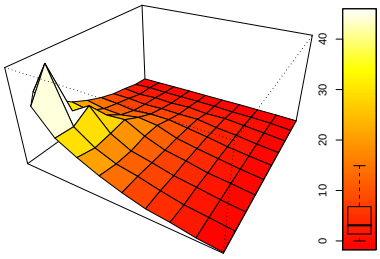
The measurement results are summarized in Figure 13. Not displayed are the results of simulations with a mean inter-packet spacing of 10 s, which yielded identical results. Plotted in Figure 13(a) is the participating nodes' number of generated RREQs per second, i.e. the number of necessary RREQs to 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



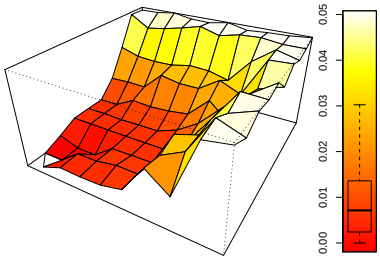
(a) RREQ sent, 1s, low



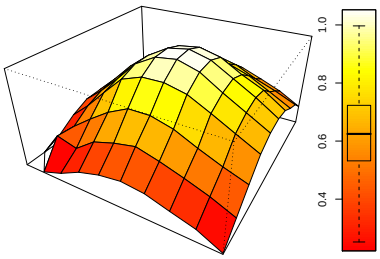
(b) Rxxx sent + fwd, 1s, low



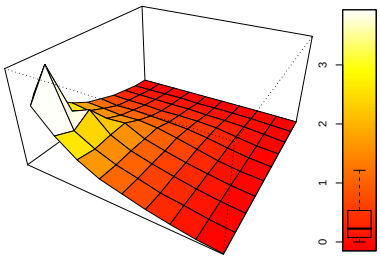
(c) msgs sent + fwd, 1s, low



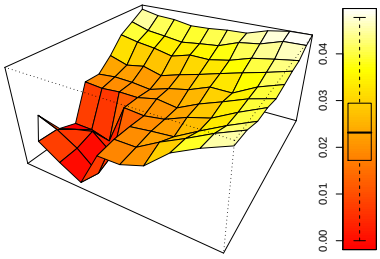
(d) RREQ sent, bursty, low



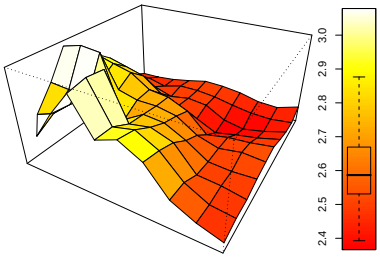
(e) Rxxx sent + fwd, bursty, low



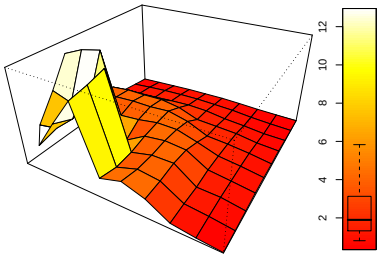
(f) msgs sent + fwd, bursty, low



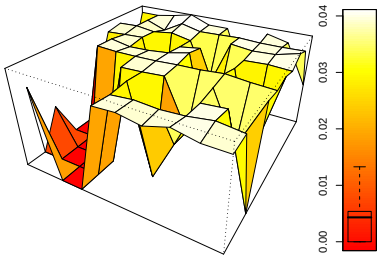
(g) RREQ sent, 1s, high



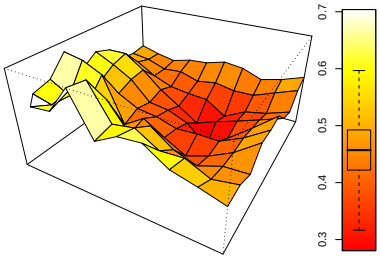
(h) Rxxx sent + fwd, 1s, high



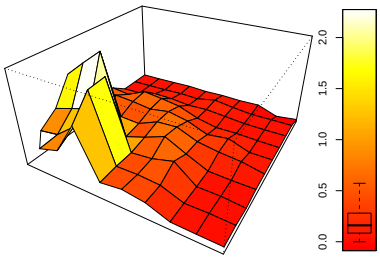
(i) msgs sent + fwd, 1s, high



(j) RREQ sent, bursty, high

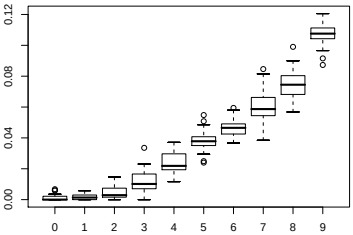


(k) Rxxx sent + fwd, bursty, high

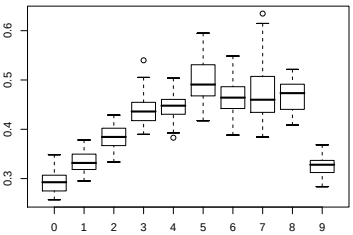


(l) msgs sent + fwd, bursty, high

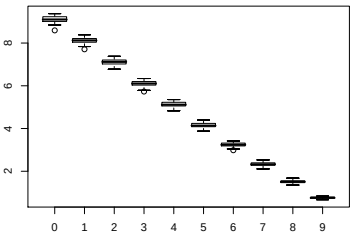
**Figure 12: Spatial load distribution in the 10x10 grid by packet spacing (1s, bursty) and density (low, high)**



(a) RREQ sent, 1s spacing



(b) Rxxx sent + fwd, 1s spacing



(c) msgs sent + fwd, 1s spacing

**Figure 13: Spatial load distribution in the line scenario**

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.

Figure 13(b) shows 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.

Figure 13(c) shows 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 AND FUTURE WORK

DYMO is one of the most recent IETF standardization efforts in the MANET community. During the last years, the standard has been updated and changed several times in order to improve the protocol and to make it better configurable depending on the particular use case. This requires intensive simulation efforts to study the effects of the resulting protocol behavior.

In this work, we presented and discussed our implementation of the DYMO ad hoc routing protocol for OMNeT++ and the *INET Framework*, and we commented on design choices we made in the process. The implementation is freely available for use and distribution under the terms of the GPL. We moved on to demonstrate how our model can be used to evaluate DYMO's performance in a number of different simulation setups and presented the results, which we obtained in these simulations. Compared to earlier measures using an older definition of DYMO, it can be seen that the protocol behavior has been improved (for more details, please refer to the corresponding report [7]). The reason is the consequent incorporation of advantageous principles from other approaches such as AODV.

We are currently using this implementation of DYMO, as well as a version which uses the full stack of the Internet protocol family, to evaluate the applicability and performance of DYMO in MANET and especially in Vehicular Ad Hoc Network (VANET) scenarios [13, 14]. 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.

# 6. 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] 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.

[3] I. Chakeres and E. M. Royer. AODV Routing Protocol Implementation Design. In *International Workshop on Wireless Ad Hoc Networking (WWAN)*, Tokyo, Japan, March 2004.

[4] 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.

[5] 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.

[6] S. R. Das, C. E. Perkins, and E. M. Royer. Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. In *19th IEEE Conference on Computer Communications (IEEE INFOCOM 2000)*, pages 3–12, Tel Aviv, Israel, March 2000.

[7] I. Dietrich, C. Sommer, and F. Dressler. Simulating DYMO in OMNeT++. Technical Report 01/07, University of Erlangen, Dept. of Computer Science 7, April 2007.

[8] 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.

[9] X. Hong, K. Xu, and M. Gerla. Scalable Routing Protocols for Mobile Ad Hoc Networks. *IEEE Network*, 16:11–21, July/August 2002.

[10] 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.

[11] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, July 2003.

[12] 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.

[13] C. Sommer, I. Dietrich, and F. Dressler. Realistic Simulation of Network Protocols in VANET Scenarios. In *26th IEEE Conference on Computer Communications (IEEE INFOCOM 2007): Mobile Networking for Vehicular Environments (MOVE 2007), Poster Session*, pages 139–143, Anchorage, Alaska, USA, May 2007. IEEE.

[14] 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.

[15] S. Sroka and H. Karl. Using Akaroa2 with OMNeT++. In *2nd International OMNeT++ Workshop*, Berlin, Germany, January 2002.

[16] A. Varga. The OMNeT++ Discrete Event Simulation System. In *European Simulation Multiconference (ESM 2001)*, Prague, Czech Republic, 2001.