Chapter 5

An evaluation study of AntHocNet

In this chapter, we present an evaluation study of the AntHocNet routing algorithm. We show results of an extensive set of simulation tests, in which we compare AntHocNet to a number of state-of-the-art AHWMN routing algorithms under a wide variety of different scenarios. We also present results of tests in which we introduce variations to the parameters and components of AntHocNet, in order to get a better understanding of the internal working of the algorithm. Since AntHocNet was in the first place developed for working in MANETs, we organized our simulation experiments according to common practice in the area of MANET routing research. In particular, we use open space scenarios in which nodes move according to typical MANET mobility patterns. This allows us to compare to existing algorithms on a fair basis. Later, in chapter 6, we present results of a different study, in which we use a WMN in an urban scenario.

The rest of this chapter is organized as follows. First, in section 5.1, we describe the setup of the evaluation study. Next, in section 5.2, we present the tests in which we compare AntHocNet to existing AHWMN routing algorithms. Finally, in section 5.3, we present results of the experimental investigation of the internal working of AntHocNet. The work presented in this chapter has been published in part in the earlier mentioned papers [73–75, 94, 95], as well as in [96] and in project deliverable [77].

5.1 Setup of the evaluation study

In this section, we describe the organization of the evaluation study presented in this chapter. First, we provide a general discussion regarding the use of simulation for the evaluation of AHWMN routing algorithms. Then, we talk about the simulator software we use. Next, we give specific details about the setup of the simulation studies we carry out. Then, we give a brief overview of

the routing algorithms we use for comparison. Finally, we discuss the measures we use to evaluate the results of the simulation studies.

5.1.1 On the use of simulation

For the evaluation of network algorithms, one can can consider two different options: an analytical study or an experimental one. For traditional telecommunication networks, analytical performance evaluation is a well studied topic [122]. Also for AHWMNs, there have been a number of interesting analytical studies, e.g. investigating physical properties such as the maximum possible throughput of a MANET [120], or the relationship between node density and connectivity [89] (see section 2.3 for descriptions of both studies). For AHWMN routing algorithms, analytical studies have been used for instance to compare load balancing properties of single path and multipath routing [109, 214] (see subsection 2.4.3), or to study the scalability of routing algorithms [231]. Such studies give interesting insights into some properties of algorithms, but are necessarily limited in scope. This is because AHWMNs are very complex environments. Mobility of network elements and interferences and loss of connectivity over wireless links cause constant changes in the network. Moreover, algorithms at different levels in the protocol stack are often quite complex (e.g. MAC protocols), precisely to deal with the dynamically changing environment, and can have unexpected interactions with each other. Analytical studies can therefore only be carried out under strict assumptions (e.g., no mobility, or perfect MAC mechanisms), which can have a strong impact on the obtained results.

Most of the research on AHWMNs therefore follows the second approach, in which new algorithms are evaluated through experiments. The basic idea in experimental research is to run the system under investigation and observe its behavior. When it is difficult to obtain sufficient observations from the real system, a possible alternative is to run the experiments in simulation. Generally speaking, simulation can be defined as the process of designing a model of a real system and conducting experiments with this model [136, 237]. Simulation is often used in computer network research, and has been particulary popular in AHWMN research. This is because it is expensive and technologically difficult to develop a real AHWMN testbed for research purposes, and because in simulation it is easier to carry out large and repeatable sets of tests. Nevertheless, in recent years, there has been a growing interest in real implementation tests as a way to validate and complement the results obtained in simulation. More about this will follow later in chapter 7.

An important issue when setting up experiments, be it in simulation or using a real implementation, is to choose the scenarios to be used in the study. Designing a scenario includes the definition of a number of variables, such as the size of the network, the movement patterns of the nodes, the data traffic between the nodes, the network protocols to be used, etc.. Each choice has an important impact on the network environment and on the measured performances, and it is therefore important that the scenarios are sufficiently representative for the applications that the network will eventually be used for. This is a problem in

AHWMN research, and especially in the field of MANETs. Until recently, the community remained rather vague about possible applications for this kind of networks, so that it was difficult to figure out what would be a realistic scenario. Hence, MANET algorithms have mainly been evaluated in experimental setups that make minimal assumptions about the environment and the use of the network: they use very simple scenarios, in which nodes move according to random patterns in a rectangular, open area and send data packets to each other at fixed rates. Only in the last few years there has been an increasing interest in experiments that use scenarios that are more complex and possibly more realistic. Especially urban scenarios are popular, as recently a number of real WMNs have been set up in urban environments, such as e.g. the public WMNs in San Francisco and Philadelphia. However, there are still considerably less studies available that use these different scenarios compared to those using simpler scenarios.

In the current chapter, our aim is to carry out a detailed evaluation study of AntHocNet, comparing it to existing state-of-the-art routing algorithms for MANETs and WMNs, and investigating its internal working. In order to obtain a better and more fair comparison with existing work in the research area, we decided to stick here with the common practice in the research community, and use simulation studies with simple scenarios that are similar to those used by other researchers. Later, in chapter 6, we will take a different approach and investigate the behavior of AntHocNet in a realistic urban scenario. Finally, in chapter 7, we go again a step further, and discuss the implementation of AntHocNet and other ACO routing algorithms in a real testbed.

5.1.2 The QualNet network simulator

When simulating a computer network, one needs to create models of the protocols and technologies that are used. Furthermore, in the case of AHWMNs, it is also necessary to develop models of the movement of the network nodes and of relevant physical phenomena, such as radio wave propagation and interference. Comparative tests have shown that differences in the used models can lead to significant differences in observed results (see [54]). It is therefore important that all the models are detailed and accurate. Such accurate models are provided in an integrated way in a number of different network simulator software packages that are available to the research community. These include ns-2 [262], OPNET [207], GloMoSim [263], SWAN [175] and QualNet [232].

For the work presented in this thesis, we used the QualNet simulator. This is a commercial simulation tool developed by Scalable Network Technologies as the follow up of the older GloMoSim simulator. QualNet offers a number of important advantages when compared to other simulators. First of all, it includes a wide range of models to support the simulation of both wired and wireless networks, and comes with an extensive library that is specifically related to MANETs and WMNs. Second, it uses a clear, modular organization, following the layered TCP/IP architecture. This makes it easy to understand and to plug in new protocols. Third, being a commercial product, it comes with

good documentation and support. Fourth, it is equipped with several graphical user interfaces, to support the design of new algorithms, the setup of simulation studies, etc.. Finally, QualNet has been specifically designed to simulate large AHWMNs, something that has traditionally been a problem in other network simulators such as ns-2.

5.1.3 Simulation scenarios

The aim of this chapter is to investigate the performance of AntHocNet in a range of different scenarios. In order to do tests in a controlled way, we define a common base scenario, from which all other scenarios are derived by varying relevant parameters such as the node speed, the data send rate, the network size, etc.. This base scenario was designed in line with the most commonly used scenarios in the research area, in order to allow a fair comparison with other algorithms. Here, we describe the properties of the base scenario. Later, in each of the experiments, we specify how the different applied scenarios were derived from it.

We consider a network of 100 nodes that move in a rectangular area of $2400 \times 800m^2$. It is an open area, in the sense that there are no obstacles that could limit node mobility or signal propagation. Node movements are defined according to the RWP mobility model (see subsection 2.3.1 and [140]). Under this model, each node starts from a randomly chosen initial position in the area, and independently chooses a random speed between a given minimum and maximum speed, and a random destination. Then, it moves at the chosen speed towards the chosen destination in a straight line. Upon arrival, it remains static for a fixed pause time, after which it chooses a new speed and destination. We use a minimum and maximum speed of respectively 0 and $10m/s$, and a pause time of $30s$. Each experiment has a duration of $900s$, and is repeated 20 times, using different random instances of the same scenario. Data traffic is generated by constant bit rate (CBR) sessions: 20 data sessions are run between randomly chosen source and destination nodes. Sessions start between 0 and $180s$ after the beginning of the simulation, and run till the end. Each session generates 4 packets of 64 bytes per second.

For the simulation of radio propagation, we use the two-ray signal propagation model, which is a common approach to model the propagation of wireless signals in open space [167]. The two-ray model assumes that a signal reaches a receiver over two different paths: one direct and one reflected over the ground. Compared to the signal that travels along the direct path, the one that travels along the reflected path arrives with a certain delay, which depends on the distance between the source of the signal and the receiver. Depending on this delay and the relative phase, the reflected signal can reinforce or disturb the direct signal. As a consequence, the two-ray model considers a different decay of the signal strength depending on the distance: it applies a decay of order R^2 (where R is the transmission distance) for short distance transmission and of order R^4 for long distance transmissions.

At the physical layer, we use the IEEE 802.11 protocol, with data transmis-

sion rate of $2Mbit/s$. The estimated radio range is $250m$. At the MAC layer, we use the IEEE 802.11 DCF protocol, which was described in subsection 2.3.3. Finally, at the transport layer, we use the UDP protocol, rather than TCP, as is common in AHWMN research. There are several reasons not to use TCP. First of all, TCP is known to behave badly in AHWMNs (see subsection 2.3.4). Second, TCP's various mechanisms to control the flow and to resend packets can influence the results in unforeseen ways so that it becomes difficult to evaluate the performance of the routing algorithms. Finally, UDP is the normal transport protocol to be used in combination with CBR applications.

5.1.4 Algorithms used for comparison

In the tests of section 5.2, we investigate how well AntHocNet performs in comparison to existing AHWMN routing algorithms. To this end, we have chosen a selection of algorithms that are representative for the wide class of available routing algorithms in the research area. The selection includes AODV, OLSR and ANSI. Here, we briefly discuss the choice for each one of them.

AODV [213] is a reactive routing algorithm. It is under investigation for standardization by the IETF MANET group [6], and has gained considerable status as the de facto standard routing algorithm for MANETs and WMNs. Most existing work in this research area uses AODV as a benchmark for comparisons, and we have followed this common practice. A short description of AODV can be found in subsection 2.4.2. Originally, we also included the DSR [140] routing algorithm in our comparative study. This is a different reactive algorithm that has also received a lot of attention in the community. However, the results obtained with DSR were quite bad and were therefore not included here.

OLSR [61] is a proactive routing algorithm. While proactive routing has often been considered a less good approach in AHWMNs [42], OLSR has received considerable attention since its publication in 2001. It is one of the most studied proactive algorithm, and it is together with AODV one of the prime candidates for standardization by the IETF MANET group. While it is less used than AODV as a benchmark in comparative studies, we consider it important to use also a proactive algorithm in our evaluation of AntHocNet, and therefore decided to include OLSR. A description of the OLSR algorithm has been given earlier in subsection 2.4.2.

ANSI [220] is, like AntHocNet, an ACO routing algorithm for AHWMNs. Due to this common source of inspiration, it has more similarities with AntHocNet than the previously mentioned algorithms. It uses full path sampling to gather routing information, sets up multiple routes, and applies to some extent a hybrid approach, where initial routes are set up reactively, and proactive sampling is used to keep this initial routing information up-to-date. The full algorithm, however, is quite different from AntHocNet. ANSI was chosen to make comparisons because we consider it important to also include an ACO routing algorithm. A brief description of ANSI has been given in subsection 3.2.6.

5.1.5 Evaluation measures

Here, we describe the measures that we use to evaluate the performance of the different routing algorithms in the experiments. We distinguish between measures of effectiveness and measures of efficiency. The measures we apply are all derived from recommendations made by the IETF MANET standardization group [62].

Measures of effectiveness are external measures of performance: they measure to what extent the algorithm manages to execute the task it was designed for. We use three different measures of effectiveness. The first one is the data delivery ratio. This is the fraction of correctly delivered data packets versus sent packets. This is an important measure in AHWMNs, as due to the constant changes in the topology it is difficult to deliver all data packets. As a second measure of effectiveness, we consider the end-to-end packet delay. This is the cumulative statistical measure of the delays experienced by packets traveling between their source and destination. Finally, as a third measure, we use the average delay jitter. This is the variation in the time interval between the arrivals of subsequent packets. It is calculated as shown in equation 5.1, where t_i is the time of arrival of the i^{th} packet, and n is the total number of packets received by a destination during a communication session. Delay jitter is an important measure for QoS applications, and also gives an indication of the algorithm's ability to respond smoothly to disruptive events in the network. In this sense, it is a measure of robustness and adaptivity.

$$jitter = \sum_{i=2}^{n} |(t_i - t_{i-1}) - (t_{i-1} - t_{i-2})| \tag{5.1}$$

Measures of efficiency are internal evaluation measures. They are concerned with the generated overhead. We consider two different measures of efficiency. The first is the overhead in number of packets. It is the total number of control packets transmitted by the nodes of the network versus data packets delivered at their destination. The second is the overhead in number of bytes. This is the total number of control bytes transmitted versus data bytes delivered. While both of these are closely related, the difference between the two measures is important in AHWMNs. As has been pointed out earlier in subsection 2.3.3, the limitations in available bandwidth in AHWMNs are for a large part due to MAC layer issues, rather than to intrinsic limitations in the possible data transmission rate. MAC layer overhead is incurred with the transmission of each packet, and the number of transmitted packets is therefore important when it comes to measuring efficiency. On the other hand, sending more bytes leads to longer channel occupancy, and also requires nodes to spend more energy. So also the overhead in number of bytes has its importance.

5.2 Comparisons to other routing algorithms

In this section, we present the results of a range of tests in which we compare AntHocNet to representative routing algorithms for MANETs and WMNs. For each of the tests, we derive scenarios from the above described common base scenario. We vary a different environmental property each time, in order to investigate its effect on the performance of the algorithms independently. We do tests changing the node mobility, the data traffic, the node density, and the network size. To change the node mobility, we run separate tests varying the maximum speed in the RWP mobility model, varying the pause time of the RWP model and using a different mobility model, namely the Gauss-Markov model (GM). To change the data traffic, we run tests varying the data send rate and the number of sessions. To change the node density, we vary the size of the area in which the nodes of the network move. Finally, to change the network size, we vary the number of nodes and the network area simultaneously.

5.2.1 Varying the maximum node speed for RWP mobility

In this first set of experiments, we vary the maximum node speed in the RWP mobility model, from $1m/s$ ($3.6km/h$, or the speed of a leisurely walk) up to $30m/s$ ($108km/h$, or the speed of a car on a highway), using as intermediate values 2, 5, 10 and $20m/s$. Varying the maximum speed in the RWP mobility model affects the node mobility directly in an obvious way: the higher the speed, the higher the mobility. Higher mobility leads to more frequent changes in the network environment, and therefore to more difficult scenarios. The results of the experiments are shown in figure 5.1, where we report (a) the delivery ratio, (b) the average end-to-end delay, (c) the average delay jitter, (d) the overhead ratio in number of packets, and (e) the overhead ratio in number of bytes.

The results for delivery ratio reflect the increasing level of difficulty of the scenarios: for all algorithms the delivery ratio decreases with increasing node speeds. The best results are obtained by AntHocNet, that even at the highest speeds is able to deliver almost 90% of all packets. This shows that AntHocNet is able to adapt well to the fast changes in the highly dynamic environment caused by high node mobility. AODV and ANSI give less good results, and with ANSI, the performance gap grows as the speed increases. The worst results are obtained by OLSR, that gives a delivery ratio of less than 40% for the highest speed scenario. This confirms the earlier mentioned observation that proactive routing algorithms have a hard time keeping up in highly dynamic environments (see subsection 2.4.1).

The results for average delay show similar trends. Like for delivery ratio, AntHocNet gives better results than AODV and ANSI. However, the gap in performance between AntHocNet and AODV decreases slightly for the highest delays. For ANSI, on the other hand, the performance gap increases, even stronger than when considering delivery ratio. One striking difference with the delivery ratio results is that for delay, OLSR gives good performances for the highest speed scenarios. For 20 and $30m/s$, OLSR even outperforms AODV

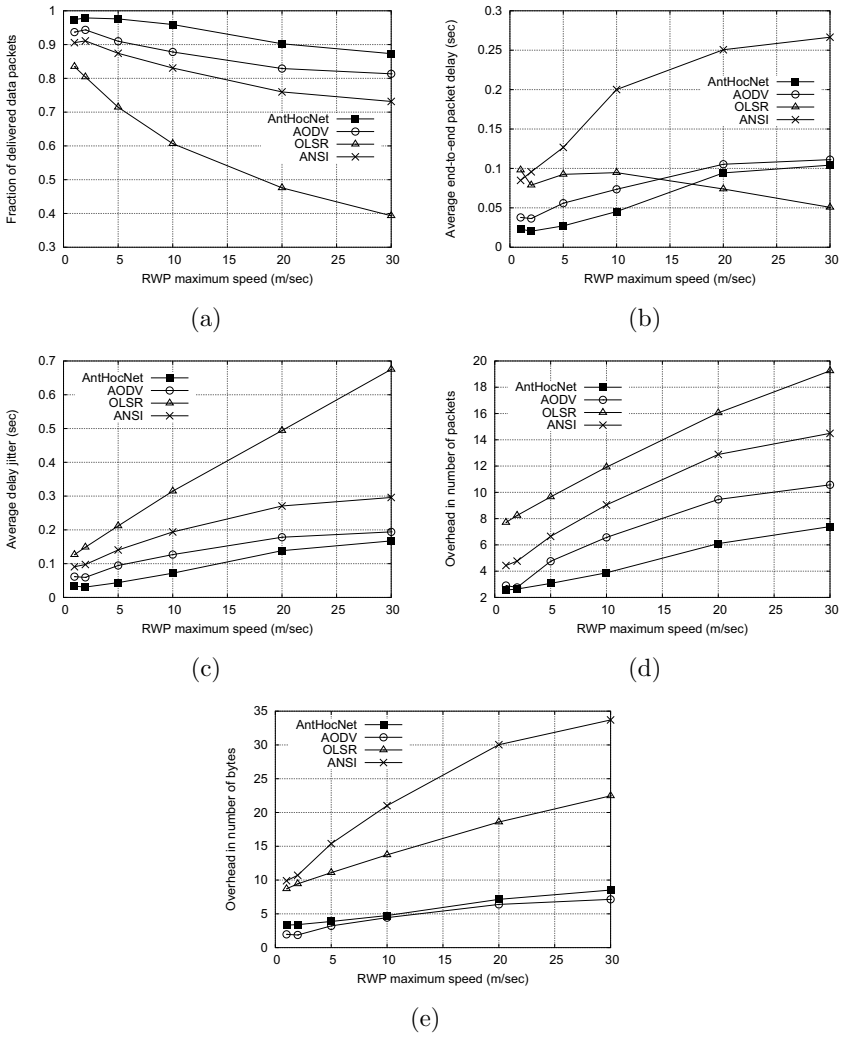

Figure 5.1: Results for AntHocNet, AODV, OLSR and ANSI using different values for the maximum speed in the RWP mobility model: (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

and AntHocNet. These results need to be read with some caution though. The good delay results are obtained in a situation where OLSR delivers only a very low percentage of the packets, and have therefore little value.

The results for average delay jitter follow the same trend as those for delivery

ratio, with AntHocNet performing better than AODV, ANSI, and OLSR, in that order. The delay jitter measures the variation in the time between arrivals of subsequent data packets. It is an indicator of robustness and adaptivity, as low jitter shows that the algorithm is able to limit the effect of disruptive events.

Finally, also in the results for the overhead measures, that reflect the efficiency of the algorithms, we can observe similar trends. There is a difference, however, between the overhead in number of packets and the overhead in number of bytes. When considering the number of packets, we obtain the same order as before, with AntHocNet giving the best performance in terms of efficiency, followed by AODV, ANSI and OLSR. When considering the number of bytes, the ACO algorithms AntHocNet and ANSI suffer a bit more, with AntHocNet becoming slightly worse than AODV, and ANSI becoming a lot worse than OLSR. This indicates that ANSI and AntHocNet use considerably larger control packets than AODV and OLSR. One reason for this is that ants gather information about the full path that they have followed. In the case of AntHocNet, an obvious other cause of large control packets is the piggybacking of routing information on top of hello messages. As mentioned before in subsection 5.1.5, both the overhead in terms of number of packets and in terms of number of bytes have their importance in AHWMNs. When we compare to the measures of effectiveness, however, the slightly worse results in terms of overhead in number of bytes does not seem to affect the performance of AntHocNet directly.

5.2.2 Varying the pause time for RWP mobility

Here, we vary the pause time of the RWP mobility model, from $0s$ up to $480s$, with as intermediate values 15, 30, 60, 120 and $240s$. The results of the experiments are shown in figure 5.2. Increasing the pause time has two different effects on the general properties of the scenario that are relevant for routing. The first of these is a decrease in node mobility: since nodes stay still for longer periods, they are less mobile, and the network becomes less dynamic. As a consequence, the scenario becomes less difficult. The second effect is a bit less straightforward, and has to do with the distribution of nodes over the network area when the RWP mobility model is used. It has been shown that under RWP, there tends to be a higher node density in the center of the network area than on the edges, especially when pause times are low [29]. To understand this, consider the square network area of figure 5.3, where we follow a single node moving according to RWP. The node starts from a randomly chosen start point (A). It chooses a random destination point (B), moves to it in a straight line according to a random speed, and then pauses for a while. After that, it repeats this same sequence of actions till the end of the simulation (in the figure, the node moves subsequently to C, D, E and F). The initial start point and all subsequent destination points are chosen according to a uniform distribution, and can therefore be anywhere in the network area. However, the straight line between any two random points has a higher probability of going through the center than of visiting edge or corner areas. As a consequence, nodes that are

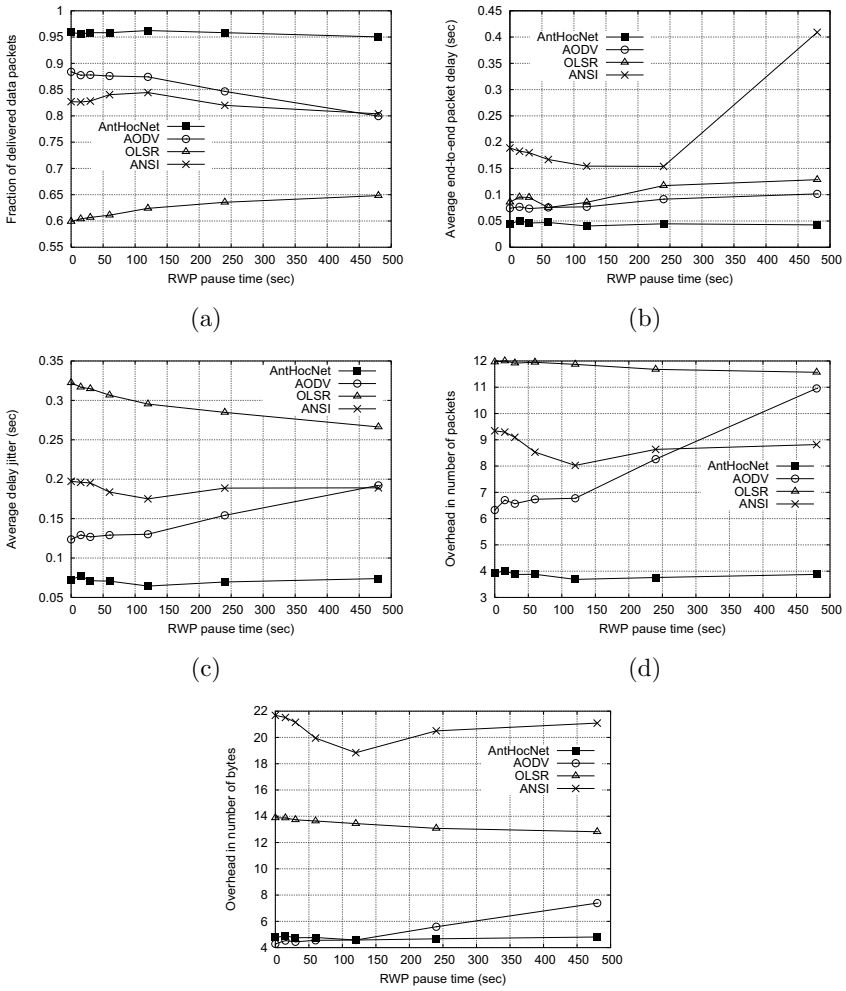

Figure 5.2: Results for AntHocNet, AODV, OLSR and ANSI using different values for the pause time in the RWP mobility model: (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

pausing in their destination points are uniformly spread over the network, while nodes that are on the move are more clustered in the center, giving a higher node density there. Concretely, this means that when pause times are increased, nodes are more spread out, giving a lower effective node density to the network.

Earlier, in subsection 2.3.1, we have discussed how a lower node density makes a scenario more difficult to deal with, as the lower connectivity provides less routing alternatives. Hence, increasing the pause time can both decrease and increase the difficulty of the scenario for routing, depending on whether the used routing algorithm is more sensitive to high mobility or to low node density.

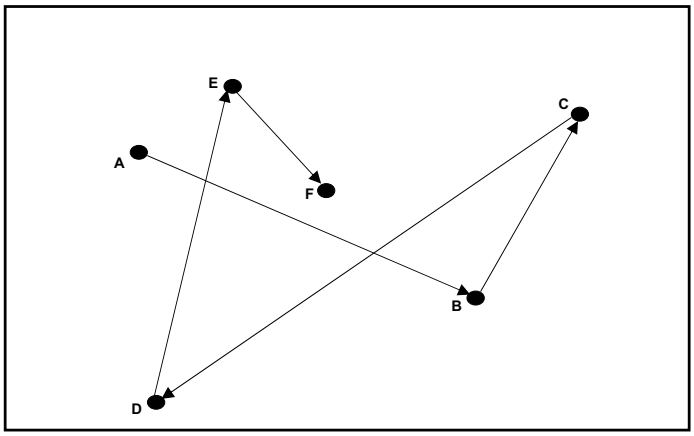

Figure 5.3: A node moving according to the RWP mobility model. The node starts in a randomly chosen initial point (A). It chooses a random destination point (B) and moves to it in a straight line. Then, it pauses for a fixed amount of time. After that, it repeats this sequence of actions till the end of the simulation (leading to the points C, D, E, and F).

When we consider the results for the delivery ratio, the ambiguity in the effect of increasing the pause time can be noted in the difference in performance between the algorithms. AntHocNet shows the best performance, and is rather insensitive to the increase in pause time. ANSI is also quite insensitive, but its level of performance is lower than that of AntHocNet. AODV shows a decrease in delivery ratio for higher pause times: from 88% for the lowest pause time down to 80% for the highest pause time. This is an indication that it is more sensitive to the decrease in connectivity than to the decrease in mobility. Finally, OLSR shows an opposite trend: its delivery ratio increases from 60% for the lowest pause time to almost 65% for the highest pause time.

For delay, the results are similar but slightly different. AntHocNet continues to show good results that are rather insensitive to the variance in pause time, AODV and OLSR show a slight drop in performance, and ANSI a strong one. For the other measures, jitter, overhead in terms of control packets and overhead in terms of bytes, we see the same trends as for delivery ratio: AntHocNet and ANSI show stable results, AODV displays a rather strongly deteriorating performance, and OLSR shows a slight improvement in performance.

We investigate in a bit more detail the difference between AntHocNet and

(a) Number of route setups per session							
Pause time (s)	0	15	30	60	120	240	480
AntHocNet	23.0	24.6	22.1	22.6	20.2	21.4	24.3
AODV	162.5	167.5	164.8	166.3	164.8	183.1	217.7
(b) Number of route retries per session							
Pause time (s)	0	15	30	60	120	240	480
AntHocNet	15.2	16.5	15.0	14.9	13.5	15.0	17.2
AODV	108.1	114.9	111.5	112.7	107.4	127.2	152.4
(c) Number of route repairs per session							
Pause time (s)	0	15	30	60	120	240	480
AntHocNet	24.2	25.9	23.8	24.0	22.2	22.1	21.4
AODV	16.1	17.9	17.5	18.7	21.0	25.1	31.8

Table 5.1: Different control packets used by AntHocNet and AODV in the experiments with increasing RWP pause times. We report the number of route setups, route retries and route repairs per session.

AODV. Here we refer to table 5.1, which reports on different types of control packets used by AntHocNet and AODV. In particular, the table gives the number of route setups, route retries (a new attempt at setting up a route, when an initial attempt has failed before) and route repairs used per session. Of these, the route setups and route retries involve the flooding of a RREQ (in the case of AODV) or a reactive forward ant (in the case of AntHocNet) over the network, and are therefore quite heavy. Route repairs involve a limited flooding and are less heavy. It is striking to see how AODV uses about 8 times as many route setups and route retries than AntHocNet. This is an indication that AntHocNet's strategy of constructing multiple paths proactively pays off. This is how AntHocNet manages to keep the overhead in number of packets low compared to AODV. When we consider the scenarios with high pause time, we see a large increase in the number of control packets needed by AODV, while AntHocNet remains quite stable.

5.2.3 Varying the speed for GM mobility

Here, we present results for tests using the GM mobility model. This is different from all other presented results, where we use the RWP mobility model. The reason for using a different model is that, while RWP is by far the most used model for the generation of node movement patterns in the literature, it has also received some criticism. This criticism concerns a number of different points. A first one is that RWP does not generate uniform node distributions [29]. This has been discussed in detail before in subsection 5.2.2. A second point of criticism is that the average node speed under RWP can be non-stationary and decreasing [287]. This is due to the fact that nodes are usually allowed to choose a random speed between 0 and a given maximum. Nodes that choose a speed that is very close to 0 may be traveling towards their next destination for a time that is longer than the duration of the simulation, and never choose a

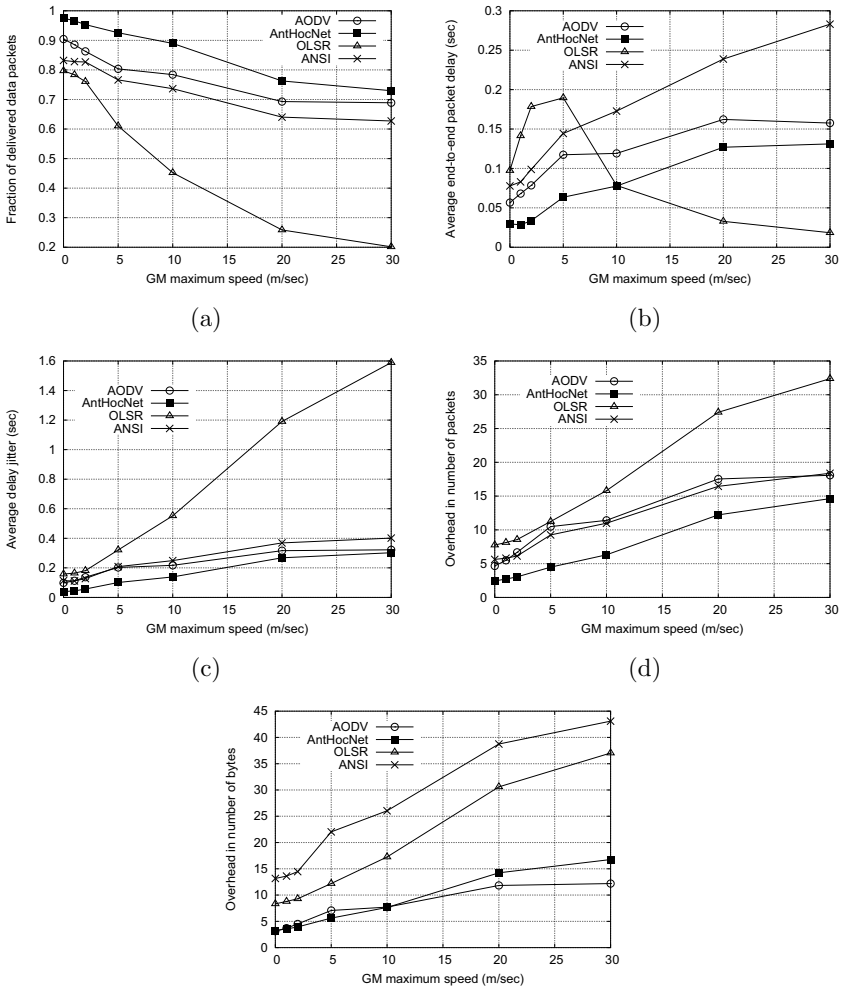

Figure 5.4: Results for AntHocNet, AODV, OLSR and ANSI using different values for the maximum speed using the GM mobility model: (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

new random speed: they are stuck in the low speed and bring down the average speed of the nodes in the network. A third point of criticism for RWP is that it does not represent human mobility very well. In particular, it leads to abrupt, uncorrelated movements after each new routing decision.

Average link duration (s)						
Maximums speed (m/s)	1	2	5	10	20	30
RWP	380	269	155	111	60	45
GM	345	207	102	55	29	20

Table 5.2: The average link duration for RWP and GM mobility over a 900 second scenario using increasing maximum speeds.

The GM mobility model was originally proposed to model node movement in infrastructure based wireless networks [172], but has also been applied in AHWMN research [46]. A number of different implementations of the model have been described in the literature. Here, we use the one provided in the BonnMotion mobility pattern generation tool [68]. Each node starts from a randomly chosen initial point in the network area, and moves according to a randomly chosen speed and direction. At fixed time intervals, the speed and direction of all of the nodes in the network are changed. For each node, a new speed value is chosen from a gaussian distribution in which the mean is the node's previous speed value, and the standard deviation is a fixed parameter value. A new direction value is chosen in the same way. Speed values are limited to a certain minimum and maximum value: a value that is chosen outside this allowed range is replaced by the closest value that falls inside the range. When a node moves outside of the network area, its next direction is adapted to be one that brings it back into the area. The GM mobility model offers some solutions to the earlier mentioned problems of RWP mobility: it produces movements that are more smooth than the sudden turns that appear under RWP, and it does not give rise to non-stationary node speeds. Saying something about the node density under GM mobility is difficult, as this has not been investigated in as much detail as for RWP.

In our simulation tests, we have used GM mobility with an update frequency of $2.5s$, a standard deviation for speed of 0.5, and a standard deviation for direction of 0.4. The minimum speed is $0m/s$ and we vary the maximum speed using the same values as in the speed value experiments with RWP of subsection 5.2.1: from $1m/s$ up to $30m/s$, with as intermediate values 2, 5, 10 and $20m/s$. One important difference between GM and RWP mobility is that under GM no pause time is used. This makes GM scenarios in general more mobile. This difference in mobility is illustrated in table 5.2, where we show the average link duration in a $900s$ scenario under both mobility models for the different maximum speed values that we use. The average link duration is the time that elapses on average between the moment a link appears and the moment it disappears again, and has been shown to be a good indicator of the mobility of an AHWMN [229]. The values in the table show that the GM scenarios are consistently more dynamic and therefore more difficult than the RWP scenarios.

The results of our experiments using GM mobility are shown in figure 5.4. As can be expected, they follow more or less the same trends as those of the speed experiments using RWP mobility: in general, AntHocNet has the best

performance, followed by AODV, ANSI and OLSR, and all algorithms show a decreasing performance as the maximum speed increases. The fact that node mobility is higher under GM compared to RWP is clearly visible: for all measures and all algorithms, the results are worse under GM than under RWP. This is especially true for the highest speed values, where, according to table 5.2, the relative difference in link duration between RWP and GM is largest. We can also see that in terms of delivery ratio, the difference in performance between AntHocNet and AODV first increases with increasing node speed, and then decreases again. The initial increase confirms what we observed earlier in the tests with RWP, namely that AntHocNet is better able to deal with the growing number of changes in the network. The eventual decrease in the difference between AntHocNet and AODV shows that there is a limit to the adaptivity of AntHocNet. At the highest levels of mobility, the proactive mechanisms of AntHocNet get more difficulties keeping up with the changes in the network, and are less able to make a difference. As a consequence, we can also see a decreasing advantage of AntHocNet in terms of jitter and an increasing advantage of AODV in terms of overhead in number of bytes. So, for very high levels of mobility, AntHocNet keeps performing well, but looses a bit of its advantage over competing algorithms. Finally, we note that the advantage of OLSR in terms of delay for the highest speed values is even stronger here than in the RWP experiments. Again, however, this good delay is only obtained when less than 50% of all packets are delivered and has therefore little relevance.

5.2.4 Varying the data send rate

With the experiments described here and in the next subsection, we investigate the effect of the data load on the performance of the different routing algorithms. In the base scenario, 20 data sessions each sending 4 packets per second are run between randomly chosen start and destination nodes. Here, we investigate the effect of varying the data send rate, while in the next subsection, we investigates what happens when the number of sessions is changed. We do tests sending 1, 4, 8, 10 and 12.5 packets per second, or 1 packet every 1, 0.25, 0.125, 0.1 and 0.08 seconds. Increasing the data rate has as an effect that the network load gets higher, so that congestion and interference become more likely. The results for the tests with varying data rates are presented in figure 5.5.

In terms of delivery ratio, the effect of the increasing congestion is obvious: all algorithms have a monotonously decreasing performance. AntHocNet performs better than the competing algorithms, but also suffers starting from 8 packets per second, where the delivery ratio is down to 66%. It is remarkable to see how AODV's performance drops very suddenly: from 88% at 4 packets per second down to just 40% at 8 packets per second. To understand this behavior, it is important to realize that an AHWMN is a highly non-linear system where the interaction between different mechanisms can have dramatic consequences. In the current experiments, the increase of the data load augments the congestion, which leads to packet loss. AODV interprets this packet loss as an indication of a link failure, and reacts to it with a route repair or a new

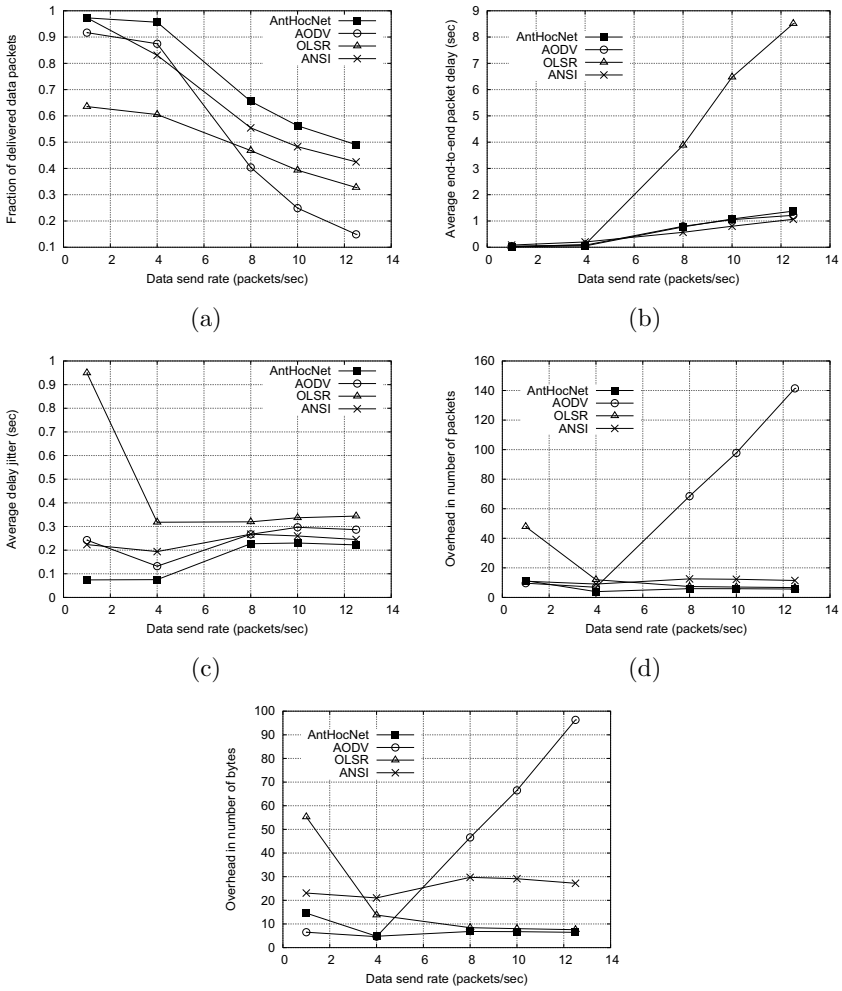

Figure 5.5: Results for AntHocNet, AODV, OLSR and ANSI using different values for the data send rate: (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

route setup. This reaction in turn strongly increases the load in the network (the flooding of a RREQ creates a large amount of extra overhead) and makes the situation worse. Hybrid algorithms such as AntHocNet and ANSI suffer much less from such problems because they do not rely purely on reactive mechanisms to deal with events; e.g. AntHocNet's proactive route maintenance process makes

multiple routes available, which can serve as backup and help to avoid the need to execute a route setup process (see also the earlier presented table 5.1, where we show the difference in number of route setup processes used by AntHocNet and AODV). Finally, proactive algorithms such as OLSR are even less sensitive to this kind of interactions, as they normally do not react to events.

For the delay measure, we can see the same trends as for the delivery ratio: all four algorithms have decreasing performance (increasing delay). AntHocNet has the best performance for the lowest data rates, but is outperformed by ANSI starting from 8 packets per second. For those data rates, however, the delivery ratios for both algorithms are quite low. Compared to the delivery ratio results, AODV shows less problematic behavior here, in the sense that for the few packets that it manages to deliver, it gets a reasonably low delay. OLSR, on the other hand, suffers strongly from the increase in data load.

In terms of jitter, we see a slightly different picture. Between 1 and 4 packets per second, all four algorithms improve to some extent their performance. This is because at 1 packet per second, the network changes a lot between every pair of subsequent data packets, so that there are wide variations in delay, leading to higher jitter. At 4 packets per second, subsequent data packets have more probability of being able to follow the same path and encountering the same network conditions. They experience more similar delays, so that the performance becomes more stable and a better jitter can be obtained. Once above 4 packets per second, the effect of the higher congestion can be felt, and all four algorithms experience a drop in performance. Of all algorithms, AntHocNet is best able to provide a low jitter.

For the remaining two measures, overhead in number of packets and overhead in number of bytes, the results bear resemblance to those for jitter. Also here, the fact that at high data rates subsequent packets are sent closer after each other has a positive effect on the performance. This is because the information gathered by the routing algorithms can get used for more packets before it gets out of date, so that the routing algorithm can work in a more efficient way. This effect is especially visible for the OLSR algorithm, that has a monotonously improving performance. This is because OLSR works in a proactive way and does not react to changes in the data rate: the amount of control packets or bytes it generates is quite stable, and increasing the data rate just means that more data packets are available to be delivered, so that the denominator of the overhead measures increases. Also the other algorithms profit to some extent from the possibility to work more efficiently when data packets are sent at a higher rate: AODV, ANSI and AntHocNet all have a decreasing amount of overhead for the lowest data rates. However, for the higher data rates, the effect of the increased congestion becomes stronger. Especially AODV suffers a lot: while it has the lowest overhead both in terms of packets and bytes for the lowest data rate, it increases rapidly and is outperformed by all other algorithms for higher data rates. The reason for this has been explained before, when we commented on the results for delivery ratio: AODV's purely reactive nature makes it extra sensitive to the arrival of disruptive events. For the highest data rates, AntHocNet has the lowest overhead.

5.2.5 Varying the number of data sessions

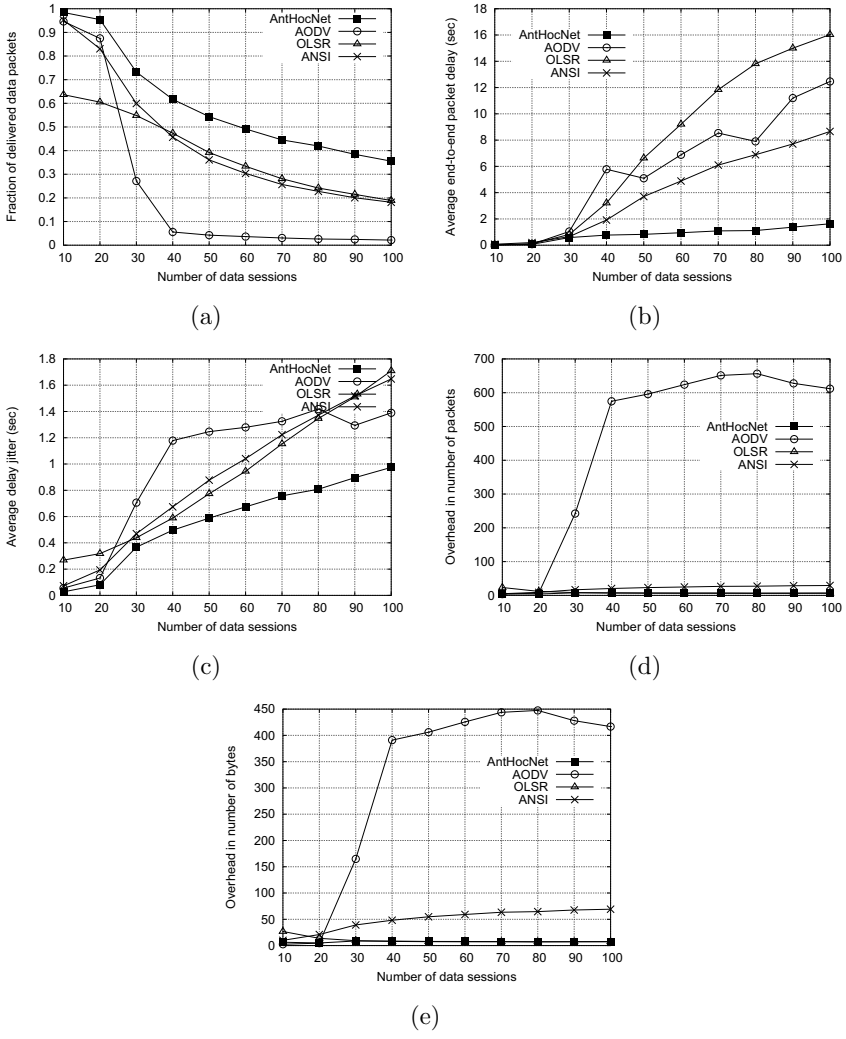

Figure 5.6: Results for AntHocNet, AODV, OLSR and ANSI using different values for the number of data sessions: (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

Here, we present results of tests with a varying number of data sessions: we use from 10 up to 100 sessions, with an increment of 10 each time. Each data session sends 4 packets per second, and source and destination nodes are

chosen randomly. When increasing the number of sessions, we increment the total data load in the network, just like we did when increasing the data send rate in the experiments of subsection 5.2.4. In particular, in terms of number of generated data packets, the scenarios with 40 sessions presented here are equivalent to those with 8 packets per second in the experiments of subsection 5.2.4, and the scenarios with 50 sessions to those with 10 packets per second (other approximate points of comparison between both sets of experiments are at 10 sessions, which corresponds to sending 2 packets per second, and at 60 sessions, which corresponds to sending 12 packets per second). Nevertheless, increasing the data load by augmenting the number of sessions can have a different effect than increasing it by sending at a higher rate. This is because the increased data load is spread over multiple sessions. When extra actions need to be performed on a per-session basis, as is the case for reactive routing algorithms, increasing the number of sessions can be more challenging. The results of the experiments with the number of sessions are shown in figure 5.8.

When considering the delivery ratio, we see a picture that is very similar to that for the data rate experiments: all four algorithms have a decreasing performance for increasing data load, AntHocNet shows the best results, and AODV has a more sudden drop in performance than the other algorithms. When comparing results directly, we can see that for AODV they are even lower here than in the data rate experiments (at 40 sessions, AODV delivers only 5% of all packets, compared to 25% when sending 8 packets per second in the data rate experiments). Apparently the higher number of route setups needed due to the presence of more sessions makes the algorithm even more sensitive to the increase in data load and congestion. Also ANSI suffers a bit more than in the data rate experiments: its delivery ratio even drops below that of OLSR when 40 or more data sessions are used. For AntHocNet, the difference with the data rate experiments is much smaller. Apparently its lower need for route setups (see table 5.1) protects its performances sufficiently. Finally, for OLSR there is practically no difference with the data rate experiments. This is because the algorithm works in a purely proactive way, so that increasing the number of sessions does not provoke higher overhead than increasing the data rate.

In terms of delay, we can notice the same trends as for the delivery ratio. Also here, all four algorithms show decreasing performance, and AntHocNet has the best results. Moreover, like for the delivery ratio, the results for OLSR and AntHocNet are very similar to those obtained in the experiments with increasing data rates. For AODV and ANSI on the other hand, the performance is worse than in the data rate experiments, with a larger difference for the purely reactive AODV algorithm and a smaller difference for the hybrid ANSI algorithm.

In terms of jitter, all four algorithms have decreasing performance, with AntHocNet showing the best results. The overall trends are considerably different from those obtained in the tests with increasing data rates, where all algorithms first showed an improvement in performance, and then a slow deterioration. In section 5.2.4, we explained that the performance improvement was due to the fact that at higher data rates, subsequent packets come closer after each other and therefore experience more similar conditions, leading to lower variations in

interarrival times. When increasing the number of sessions, this positive effect is not present.

Considering the last two measures, the overhead in number of packets and in number of bytes, we can see that the performance of OLSR improves with the number of sessions, while that of AntHocNet and ANSI is relatively stable and that of AODV shows a dramatic deterioration. Overall, AntHocNet shows the best performance. The improvement in overhead results of OLSR is again due to the fact that it is a proactive algorithm and therefore does not use extra control packets when more sessions are started; therefore, more data packets are delivered correctly while using the same number of control packets. Different from the data rate experiments, the other three algorithms do not show an improvement in overhead at the low end of the range of the data load. In the data rate experiments, such an improvement was possible because the higher send frequency of data packets allowed to use reactively obtained routing information for more data packets, thus improving efficiency. When increasing the number of sessions, this effect does not exist.

5.2.6 Varying the network area size

In this subsection, we present results of tests in which we vary the size of the area in which the nodes move. The sizes we use are $1200 \times 400m^2$, $1500 \times 500m^2$, $1800 \times 600m^2$, $2100 \times 700m^2$, $2400 \times 800m^2$, $2700 \times 900m^2$, $3000 \times 1000m^2$, $3300 \times 1100m^2$, and $3600 \times 1200m^2$. Increasing the size of the network area increases the average path length between nodes and decreases the node density. Both make the scenario more difficult. The importance of the node density in AHWMNs has been discussed before in subsection 2.3.1. Sparser networks form a more difficult environment because they are less well connected. In the best case, this means that there are few routing alternatives between the source and destination nodes of a session so that link failures are difficult to repair. In the worst case, there is just no connectivity, and data packets cannot be sent. The results for the network area tests are shown in figure 5.7.

When considering delivery ratio, we can see that for the scenarios with smallest network area, AntHocNet and AODV perform equally well, while the result is slightly worse for ANSI and a lot worse for OLSR. The similarity in performance between AntHocNet and AODV is to be expected. In a dense scenario on a small surface area, paths are short and many alternatives are available, and it is therefore relatively easy to rebuild routes after a link failure. As a result, the proactive route maintenance and the reactive route repair processes of AntHocNet, which are aimed at improving routes and avoiding the need for new route setups, create extra overhead without adding much value. Moreover, the large hello messages sent out by all nodes in AntHocNet can cause more interference than in sparse scenarios, since each node has many neighbors. For increasing area sizes, the scenarios become more sparse, and all algorithms show decreasing performance. AntHocNet is better able to deal with the difficulties of sparse scenarios than the other three algorithms, because here its different mechanisms do pay off. Especially with AODV there is a growing difference in

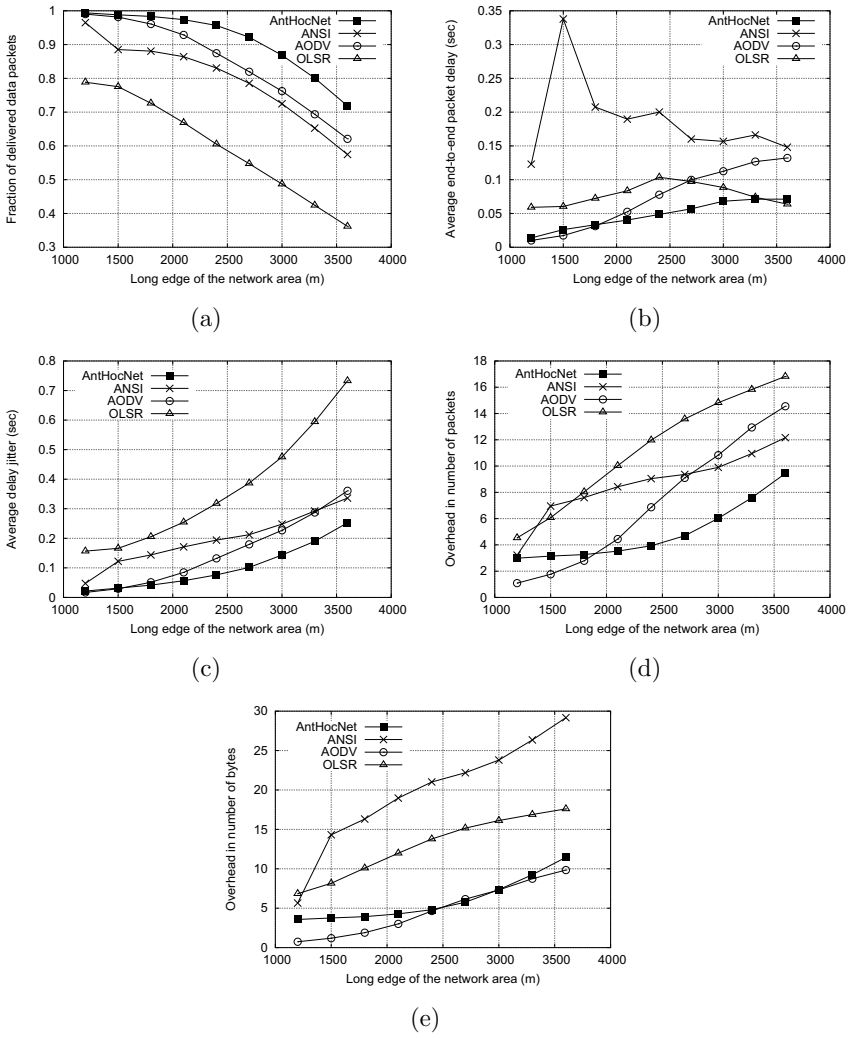

Figure 5.7: Results for AntHocNet, AODV, OLSR and ANSI using different sizes for the network area surface. The length of the long edge in meters is given on the x-axis, while the length of the short edge is always one third of this. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

performance.

When considering delay, the image is similar. Now, for the smallest network areas, AODV performs slightly better than AntHocNet. But again, as the area

increases in size, AntHocNet becomes the better algorithm. For OLSR, the results are a bit ambiguous, with first an increase of the delay and then a decrease. This behavior is similar to that shown in the speed experiments of subsections 5.2.1 and 5.2.3. Also there the delay gets low for OLSR in the most difficult scenarios, when the delivery ratio is already very low. For ANSI the delay results are quite bad and rather unstable.

Also for jitter, the results are similar to those for delivery ratio. For the smallest area size, AODV has similar or even better performance than AntHocNet. Then, as the size increases, AODV's jitter deteriorates faster, and AntHocNet becomes better. OLSR and ANSI both perform worse than AntHocNet, with OLSR giving the worst performance.

Finally, also for the overhead measures we see the same kind of patterns. Here, the advantage of AODV in the scenarios with smallest area is more pronounced. This confirms what we mentioned earlier, that in the scenarios with high density and short paths AntHocNet's mechanisms produce extra overhead without improving performance. AODV purely reactive approach is then better. However, for the scenarios with larger areas, the overhead in number of bytes is comparable for AODV and AntHocNet, while the overhead in number of packets of AntHocNet is much lower than that of AODV. ANSI and OLSR have worse results for both overhead measures.

5.2.7 Varying the number of nodes

In this subsection we investigate the scalability of our routing algorithm. We present the results of a set of tests with increasing network sizes: we increment the number of nodes from 100 up to 800 nodes in steps of 100. We increment the network area size proportionally, from $2400 \times 800m^2$ for the 100 node network up to $6800 \times 2250m^2$ for the 800 node network (with as intermediate steps $3400 \times 1130m^2$, $4150 \times 1390m^2$, $4800 \times 1600m^2$, $5370 \times 1790m^2$, $5800 \times 2000m^2$ and $6350 \times 2100m^2$), in order to keep the node density constant. The results of the experiments are shown in figure 5.8.

When we consider the results for delivery ratio, we can see that AntHocNet is able to deliver more packets correctly than the other three algorithms over the wide range of different network sizes. Moreover, the difference in performance grows with increasing network sizes. For the highest network sizes, AntHocNet still delivers more than 70% of all data. AODV and ANSI, on the other hand, fall below 50%. For OLSR, we did not run tests for more than 500 nodes, as the results were too low (and simulation times became very large). These results show that AntHocNet scales well. Its various mechanisms for proactive route maintenance and reactive route repair allow it to deal better with the longer paths in large networks, and help it avoid the need for new route setups. The latter is very important as a route setup involves the flooding of a reactive forward ant to all nodes in the network. The bad results of OLSR confirm that proactive routing is more difficult when the number of nodes gets higher, as it gets impossible to keep correct routing information for all possible destinations in all nodes.

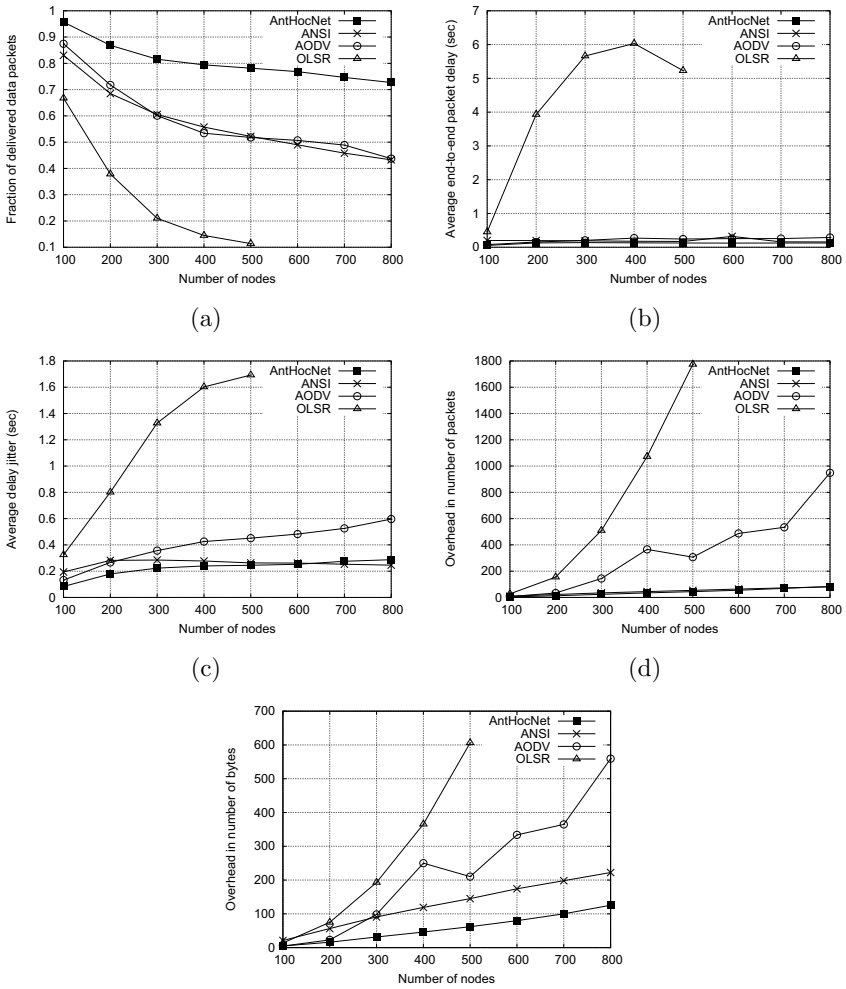

Figure 5.8: Results for AntHocNet, AODV, OLSR and ANSI using different network sizes. The number of nodes in the network is indicated on the x-axis. The network area size is incremented proportionally so that the node density remains the same as in the base scenario. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

When considering delay and jitter, the results are slightly different, but similar. For delay, AntHocNet shows the best results. However, ANSI is now not

much worse. For jitter, ANSI is slightly better than AntHocNet for the largest networks. For both measures, AODV lags more behind, and the difference grows with increasing networks sizes. OLSR, finally, performs really badly. Its delay goes down slightly for the largest networks, where it is only delivering the easiest data packets (a similar effect was also visible in the speed experiments of subsections 5.2.1 and 5.2.3 and the density experiments of subsection 5.2.6).

In terms of both overhead measures, finally, we again see similar results. AntHocNet and ANSI have the best performance, with a small difference when considering number of packets, and a larger one in the advantage of AntHocNet when considering number of bytes. AODV has worse performance, and the difference grows with increasing network sizes, indicating that it is less scalable than the two ACO routing algorithms. Finally, the proactive OLSR algorithm turns out to be highly inefficient for large network sizes.

5.2.8 Summary

In the results presented in this section, we have compared AntHocNet to a number of representative routing algorithms. We have varied many different environmental parameters, in order to investigate how each of these affects the performance of the algorithms in absolute and relative terms. In general, we could observe that AntHocNet shows very good behavior over the wide range of scenarios, and often outperforms the other three algorithms.

When considering the mobility experiments, we can see that AntHocNet can deal better than the other algorithms with increasing mobility. It is better able to deal with the network changes induced by mobility. In the RWP tests with increasing maximum node speed, we can see that as the network gets more dynamic, AntHocNet's advantage over the other routing algorithms grows. In the pause time tests, results are rather ambiguous, due to the various conflicting trends that are caused by changes in the pause time. In the test with increasing speed under GM mobility, we can see similar trends as under RWP mobility, but we can also see that there is a limit to AntHocNet's adaptivity: for the highest speed values AntHocNet's advantage over AODV becomes smaller. This is because mobility under GM gets higher than under RWP, and under the extreme mobility of these scenarios, it becomes hard for AntHocNet's proactive mechanisms to keep up and make a difference.

When considering the data load experiments, we can see that none of the considered algorithms are really able to deal with high data send rates or high numbers of sessions. The AHWMN capacity is just too limited. Nevertheless, we can see that AntHocNet keeps better up with the increasingly challenging environment. Only for the delay results in the data rate experiments, it is slightly worse than ANSI. So we can say that the performance of AntHocNet scales well with increasing data load. On the other hand, the purely reactive approach of AODV turns out to be quite sensitive to changes in the data load, since it creates too much overhead in reaction to disruptive events.

When considering the experiments with varying node density, we can observe that all algorithms suffer from the longer path lengths and lower connectivity as

scenarios get sparser. However, AntHocNet deals better with these challenges, and especially compared to AODV there is a growing gap in performance as node density decreases. On the downside, we can observe that AntHocNet has more difficulties in the densest scenarios. Its proactive route maintenance and reactive route repair mechanisms are rather useless there, as in those scenarios reactively rebuilding a route like AODV does can be more efficient than continuously trying to extend, improve and repair routes. This is most visible in the overhead results, where AODV clearly outperforms AntHocNet when the network is small and dense.

Finally, when we consider the experiments with increasing network sizes, we see similar patterns. Again, all algorithms suffer from the increasing scale, and especially OLSR turns out to be unable to cope with large AHWMNs. In terms of delivery ratio, we see again the same trend as before, with an increasing performance gap between AntHocNet on the one hand and AODV and ANSI on the other hand, showing that AntHocNet is better able to deal with the difficulties that arise in larger networks. In terms of the other performance measures, the same growing performance gap between AntHocNet and AODV remains visible, but ANSI's performance is more similar to that of AntHocNet. In general, the results of the tests with increasing network sizes show that AntHocNet is able to maintain its good performance as the network size increases, thereby showing its good scalability.

5.3 Analysis of AntHocNet's internal working

In this section, we present a number of tests in which we try to get a better understanding of the working of AntHocNet. To this aim, we make variations in the parameters and components used by the algorithm and observe the effect of these changes. In particular, we do tests switching off the proactive components and the local repair mechanism, using different routing metrics, varying the send frequency of proactive forward ants, varying the number of entries in the pheromone diffusion messages, varying the routing exponent of proactive forward ants, and varying the routing exponent of data packets. All tests are again carried out in the earlier described base scenario and adaptations of it. We use adaptations that are relevant for the analysis at hand. We use as evaluation measures the delivery ratio, end-to-end-delay, delay jitter and overhead in number of packets. For some of the experiments, we also include the number of hops taken by successfully delivered data packets. This is a measure of efficiency, as it indicates how many transmissions were needed to bring each of the data packets to its destination. The overhead in number of bytes was not included here, due to general similarity with the results for the other overhead measure.

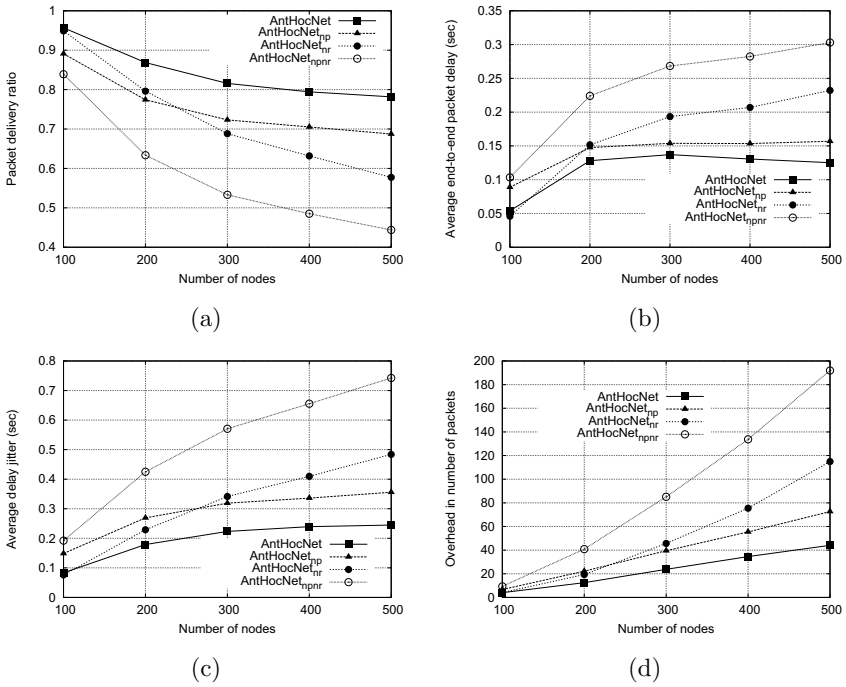

Figure 5.9: Results for AntHocNet, AntHocNet without proactive route maintenance process ("AntHocNet$_{np}$"), AntHocNet without local route repair ("AntHocNet$_{nr}$") and AntHocNet with neither proactive route maintenance nor route repair ("AntHocNet$_{npnr}$"). The tests are carried out in scenarios with increasing number of nodes, as in subsection 5.2.7. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

5.3.1 Switching off proactive actions and local repair

In the experiments presented in this subsection, we try to figure out what the individual effect is of different components of AntHocNet. In particular, we investigate the relevance of the proactive route maintenance process and the route repair process, as these are two components that we found to be defining for the algorithm's behavior. We compare the performance of the full AntHocNet algorithm with the performance of the algorithm without proactive route maintenance (which we refer to as AntHocNet$_{np}$), the algorithm without local route repair (which we refer to as AntHocNet$_{nr}$), and the algorithm with neither proactive route maintenance nor local route repair (which we refer to as AntHocNet$_{npnr}$). The tests scenarios that we use are the ones of subsection 5.2.7, where we increase the number of nodes and the network area simultaneously. The maximum number of nodes here is 500. The results are

presented in figure 5.9.

When we first consider the smallest network size (100 nodes), we can see that AntHocNet$_{nr}$ performs equally well as, or even better than, the full AntHocNet algorithm. This shows that the local repair component adds little or no value at this scale. On the other hand, the proactive route maintenance process does add a lot of value: AntHocNet$_{np}$ performs considerably worse than the full AntHocNet algorithm for all evaluation measures. It is also interesting to see that proactive route maintenance and local repair can substitute each other up to a certain extent. This can be concluded from the fact that AntHocNet$_{npnr}$ performs considerably worse than AntHocNet$_{np}$, while AntHocNet$_{nr}$ does not perform worse than the full AntHocNet algorithm: it seems that the local repair mechanism has more value in AntHocNet$_{np}$, where there is no proactive route maintenance, than in the full AntHocNet algorithm.

When we consider larger network sizes, we can see that the performance gap between the full AntHocNet algorithm and AntHocNet$_{np}$ grows steadily for all evaluation measures, indicating the continued importance of the proactive route maintenance component. On the other hand, the gap between AntHocNet and AntHocNet$_{nr}$ grows fast, indicating that in large network sizes, the local repair mechanism does become an important mechanism in order to maintain good performance.

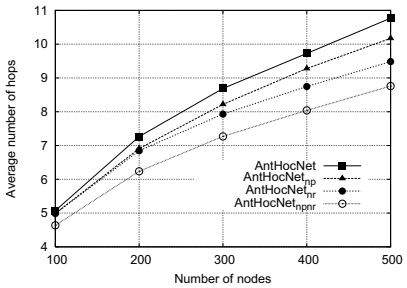

Figure 5.10: The average number of hops for different versions of AntHocNet in scenarios with increasing number of nodes.

Finally, we also present results for the average number of hops taken by successfully delivered data packets. The number of hops is an indication of how efficiently algorithms manage to bring data packets to their destination. The results are shown in figure 5.10. It is striking to see that the relative performances for this measure of efficiency go directly against the performances for all other measures. The full AntHocNet algorithm, which has the best results for delivery ratio, delay, jitter and overhead, uses the longest paths to deliver its data. On the other hand, AntHocNet$_{npnr}$, which has the worst results for the other measures, uses the shortest paths. An explanation for the shorter path lengths used by AntHocNet$_{npnr}$ is that due to the lack of proactive maintenance or repair, routes have to be rebuild from scratch after each link failure. When

building a new route, a reactive forward ant is flooded over the network, and the first copy of it to reach the destination is sent back to the source. This approach assures that a new short route is set up each time. On the other hand, when routes are repaired, or replaced by backup routes, there is no possibility to restart from scratch, so that longer routes are often used. Nevertheless, it is clear from the results that using shorter paths does not necessarily lead to good results. This has also been described earlier in subsection 2.4.3, and we come back to this issue in the next subsection, when we discuss the use of different metrics.

5.3.2 Using different routing metrics

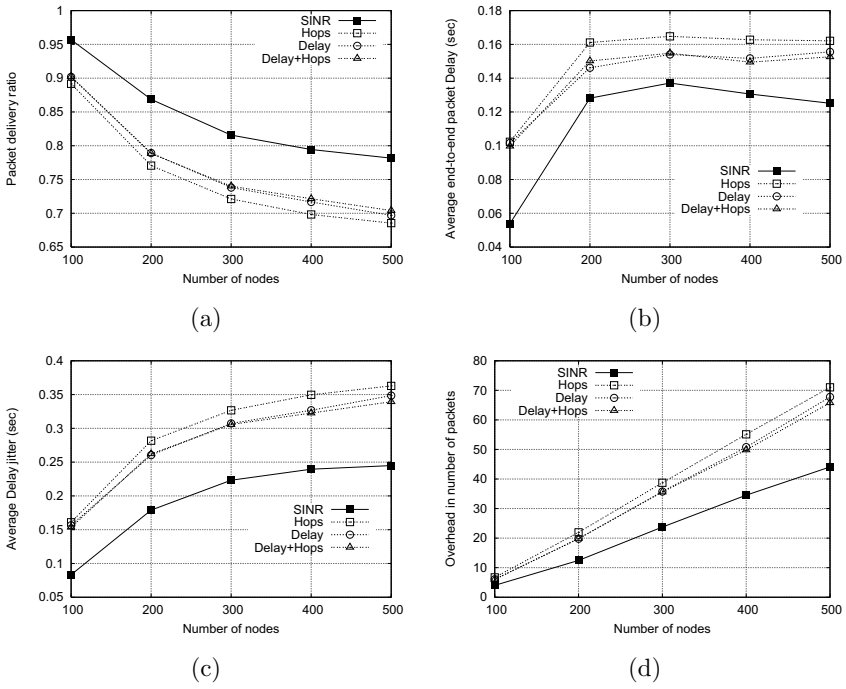

Figure 5.11: Results for AntHocNet using different routing metrics: signal-to-interference-and-noise ratio ("SINR"), number of hops ("Hops"), delay ("Delay"), and the combination of hops and delay ("Delay+Hops"). The tests are carried out in scenarios with increasing number of nodes, as in subsection 5.2.7. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

In this subsection, we present results of tests in which we use different routing metrics. The routing metric is the criterium used by the algorithm to compare

and choose routes. The different metrics we use here have been described in subsection 4.2.6. They are the number of hops, the end-to-end delay, the combination of hops and delay, and a metric based on the signal-to-interference-and-noise ratio (SINR), which penalizes the use of low quality links. The tests presented here were carried out in networks of increasing sizes, with a maximum of 500 nodes, as before. The results are presented in figure 5.11.

From the presented graphs, we can see that using the metric based on SINR gives by far the best results for all considered evaluation measures. So, it is clearly advantageous to be able to detect bad links and avoid them. The delay metric and the metric that combines delay and number of hops give worse results. The worst results, finally, are obtained when using the number of hops metric. This is despite the fact that this metric leads to the discovery and use of shortest paths, as is indicated in figure 5.12, where we plot the average number of hops used for each successfully delivered data packet. Choosing the shortest paths is a common practice in the AHWMN literature. The reason why this is not a good idea was pointed out in [67]: paths with a low number of hops usually consist of long hops, which can be of low quality and break easily as a consequence of node movement, and tend to go through the center of the AHWMN area, where congestion and wireless channel contention is higher.

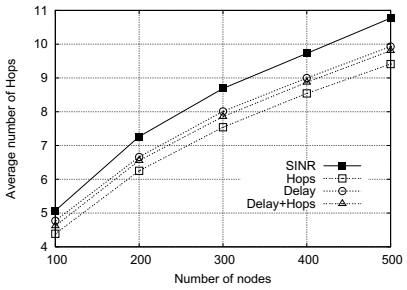

Figure 5.12: The average number of hops for AntHocNet using different routing metrics in scenarios with increasing number of nodes.

5.3.3 Varying the proactive ant send interval

Here, we present results of tests in which we vary the proactive ant send interval. This is the time between the launching of successive proactive ants in the proactive route maintenance process. It defines how often the algorithm looks for path improvements, and therefore how quickly it can adapt to new routing opportunities. We did tests with send intervals of 0.5, 1, 2, 5, 10, 20, and 50s. We use two groups of scenarios. In the first group, we use variations of the base scenario with increasing mobility: we apply RWP with maximum node speeds of 2, 5, 10 and 20m/s. We use these scenarios in order to investigate the interaction between the rate of change of the scenario and the adaptivity rate of

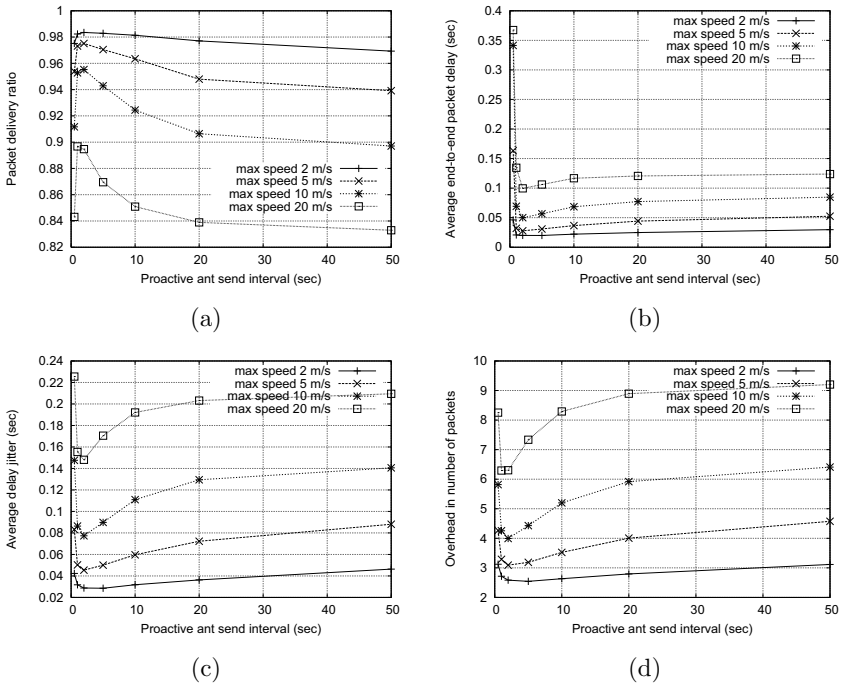

Figure 5.13: Results for AntHocNet using different send intervals for the proactive ants. We send 1 ant every 0.5, 1, 2, 5, 10, 20, and 50s. This is indicated on the x-axis. We use scenarios with varying mobility: we apply RWP with maximum speeds of 2, 5, 10 and 20m/s. The results for different speed values are represented with different curves. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

the algorithm. In the second group, we use variations of the base scenario with increasing data send rate: we have data sessions sending at 1, 4 and 8 packets per second. We use these scenarios in order to investigate how the send rate of ants interacts with the send rate of data. The results of the experiments varying the node speed are given in figure 5.13, and those varying the data send rate in figure 5.14.

When considering both figures 5.13 and 5.14, we can observe a constant pattern for all different scenarios and evaluation measures. First, at very low ant send intervals, the algorithm shows bad performance. This is because too many ants get injected into the network, so that they cause congestion. Then, there is an optimum value at around 1 to 2s. After that, the performance decays because the algorithm is not sending enough ants to keep up with the changes in the network. When we focus specifically on the results using varying levels

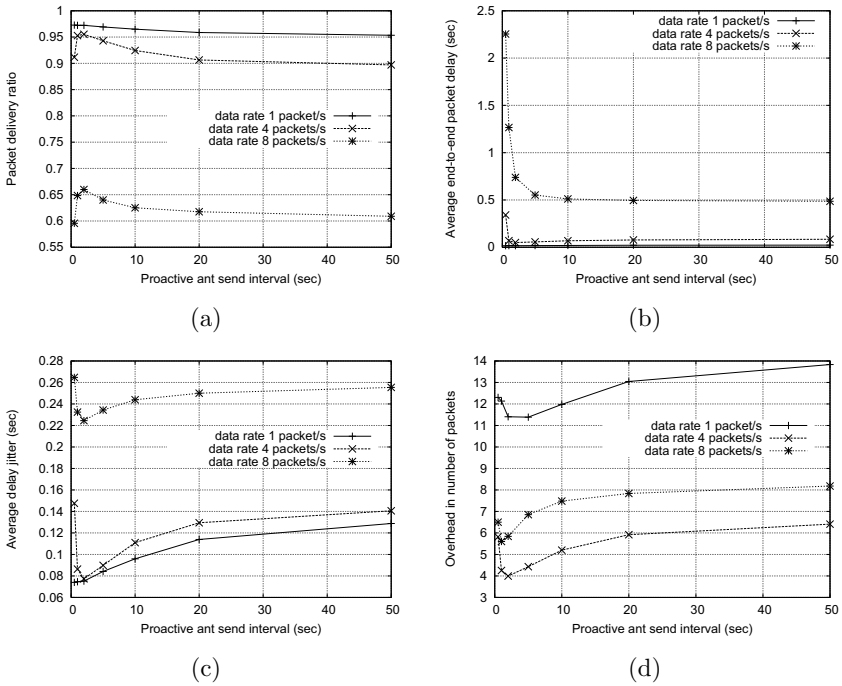

Figure 5.14: Results for AntHocNet using different send intervals for the proactive ants. We send 1 ant every 0.5, 1, 2, 5, 10, 20, and 50s. This is indicated on the x-axis. We use scenarios with varying data load: we use data sessions sending 1, 4 and 8 packets per second. The results for different data send rates are represented with different curves. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

of mobility, we can see that the above pattern is less clearly visible for the low speed scenarios than for the high speed ones. This is because when the network changes slowly, it is less crucial to adapt quickly. When we zoom in on the results with varying data load, we can see that the pattern is best visible at the intermediate send rate of 4 data packets per second. When sending less data, paths often need to be rebuilt for each data packet (see also subsection 5.2.4), so that adaptivity is less able to make a difference. For high data rates, the performance generally deteriorates strongly due to high levels of congestion, and again it is more difficult to make a difference using proactive adaptivity. One interesting observation when comparing the results over all different scenarios is that the optimal ant send rate is relatively stable and independent from the node mobility or data send rate. Sending one ant every 2s almost always gives the best performance.

5.3.4 Varying the number of entries in the pheromone diffusion messages

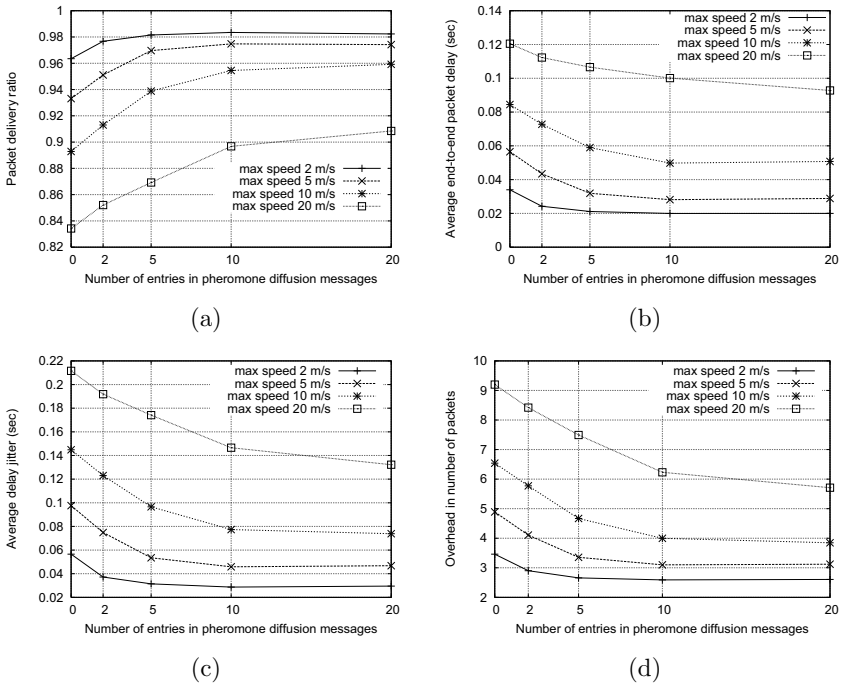

Figure 5.15: Results for AntHocNet using different number of entries in the pheromone diffusion messages. The number of entries used are 0, 2, 5, 10 and 20, and are indicated on the x-axis. We do experiments with varying node mobility: we apply RWP with maximum speeds of 2, 5, 10 and 20m/s. The results for different speed values are represented with different curves. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

Here we present the results of tests in which we vary the maximum number of entries used in the pheromone diffusion messages (the hello messages). This number of entries defines how much information is sent out in each of the messages, and therefore how quickly information can spread over the network (see also subsection 4.2.3). We made tests using 0, 2, 5, 10 and 20 entries. 0 entries is the extreme case in which no pheromone diffusion takes place. In that case, no virtual pheromone is available in the network, so that proactive ants cannot find new routes and the proactive route maintenance process is effectively switched off. As scenarios, we again use different levels of mobility, applying RWP with maximum speeds of 2, 5, 10 and 20m/s. The results of our experiments are

shown in figure 5.15.

In general, the results for all evaluation measures show the same trend, with the performance monotonically improving with higher numbers of hello entries. This stresses the importance of having a quickly adapting proactive route maintenance process. Like for the results of subsection 5.3.3, the observed trend is more pronounced when using higher mobility, indicating that the importance of the effectiveness of the proactive adaptivity increases with increasing network change rates.

5.3.5 Varying the routing coefficient for ants

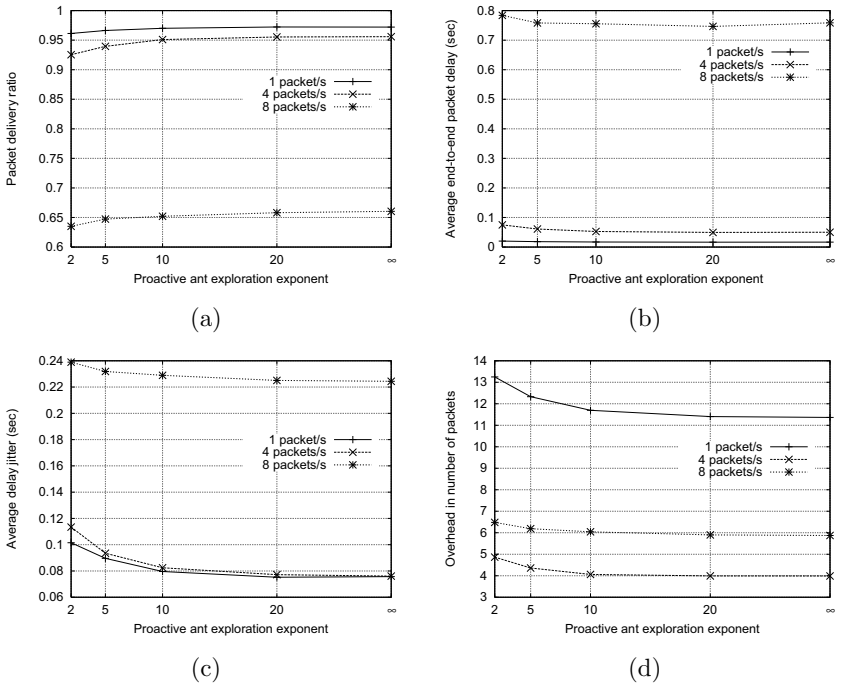

Figure 5.16: Results for AntHocNet using different values for the proactive ant routing exponent. The exponent values that we use are 2, 5, 10, 20 and ∞ (the latter represents deterministic forwarding of proactive forward ants along the best path). The tests are carried out in scenarios with varying data load: we use data sessions sending 1, 4 and 8 packets per second. The results for different data send rates are represented with different curves. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

In the experiments presented here, we vary the routing coefficient used by the

proactive forward ants, the parameter β_2 of formula 4.5. This parameter defines the amount of exploration the ants are allowed to do when they are constructing a path towards their destination. When β_2 is high, the ants are concentrated on the paths with the best pheromone values, so that they limit their exploration to paths that have been indicated to be good either by previous ants or by the pheromone diffusion process. On the other hand, when β_2 is low, the ants can also follow paths with low pheromone. This way, paths that are better than what their pheromone values indicate (e.g. because of changes in the network or erroneous previous estimates) can be discovered. A similar parameter β_1 exists for reactive forward ants, and a parameter β_3 for data packets. β_1 is always kept high because when constructing an initial path with reactive forward ants, we want to get a route as quickly as possible and do not want to risk loosing time exploring different possibilities. The effect of β_1 is therefore not investigated. β_3 defines to what extent data packets can be spread over multiple paths and is investigated in the next subsection.

The results of the current experiments are presented in figure 5.16. We use β_2 values of 2, 5, 10 and 20, and also consider the possibility of deterministically following the best pheromone, which corresponds to a β_2 value of infinity. We use scenarios with increasing data load as before, in which sessions send at 1, 4 and 8 packets per second.

From the results, it is evident that the scenarios with 8 packets per second are much more challenging than the ones with 1 and 4 packets per second. Nevertheless, a constant pattern with respect to the β_2 parameter can be observed for all three sets of experiments. As β_2 increases, and the amount of exploration by the ants decreases, the performance improves for all evaluation measures. This is in contrast with other ACO algorithms, such as the AntNet algorithm for wired networks (see subsection 3.2.3 and [71]), where explorative behavior of the forward ants is an essential part of the algorithm. The reason is that in AntHocNet's proactive route maintenance process, the task of exploring new good paths is performed by the pheromone diffusion process (while in AntNet, ants are the only available mechanism to do exploration). This process indicates the good routes it finds through the virtual pheromone, and the role of the ants is mainly to control whether this indicated information is correct. When proactive forward ants are requested to do more exploration, the algorithm is less fast to adopt the best routes indicated by pheromone diffusion, and therefore slower to adapt to new network situations. Therefore, we always use a high value for β_2. In a sense, AntHocNet uses a clear separation of tasks compared to other ACO routing algorithms: the pheromone diffusion process executes exploration and the discovery of new routes, while the ants continuously control the provided routing information and set up routes based on it.

5.3.6 Varying the routing coefficient for data

In this subsection, we present results of tests in which we vary the data routing exponent, parameter β_3 of equation 4.6. This parameter controls the stochastic forwarding of data packets. It defines how strong the preference of data packets

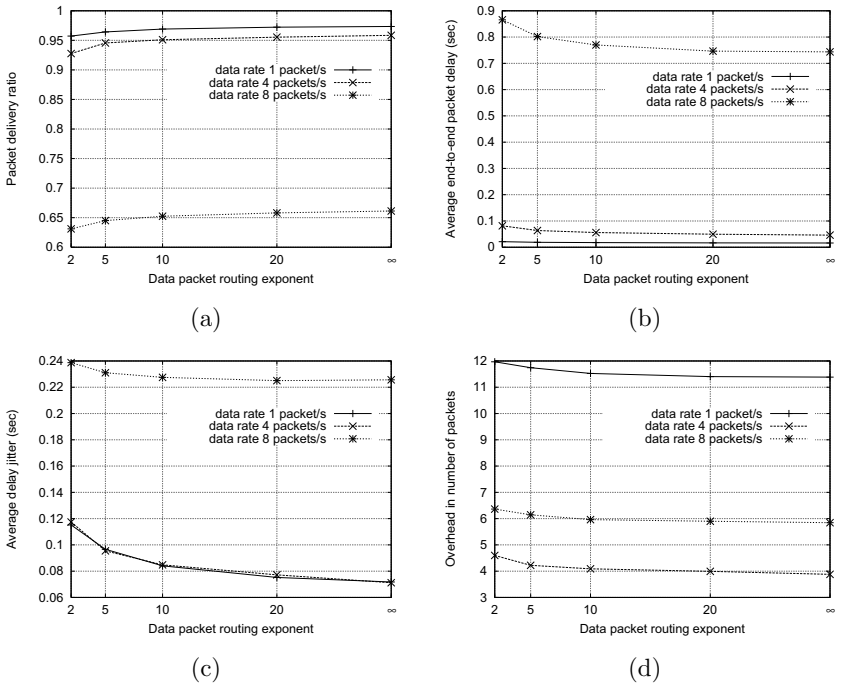

Figure 5.17: Results for AntHocNet using different values for the data routing exponent. The exponent values that we use are 2, 5, 10, 20 and ∞ (the latter represents deterministic forwarding of data along the best path). The tests are carried out in scenarios with varying data load: we use data sessions sending 1, 4 and 8 packets per second. The results for different data send rates are represented with different curves. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

for paths with high pheromone is. When β_3 is low, this preference is weak, and data can therefore be spread out over a range of multiple paths. On the other hand, when β_3 is high, data packets are only sent over the best paths. In the limit, where β_3 reaches infinity, data is forwarded deterministically over the best path. In our experiments, we use β_3 values of 2, 5, 10, 20, and infinity. We again use scenarios with increasing data load, in which sessions send at 1, 4 and 8 packets per second. The results are presented in figure 5.17.

The graphs show a similar trend as those for the tests with β_2. Also here, performance improves with increasing values for the routing coefficient under all scenarios and for all evaluation measures. So, it turns out that the feature of stochastically spreading data packets over multiple paths is not beneficial, but, on the contrary, deteriorates results. This is again in contrast with other ACO routing algorithms, where stochastic data forwarding allows to improve

performance by spreading the data load over multiple paths and making better use of the full network resources. An explanation of why this is not working in the case of AntHocNet can be found in the fact that in AHWMNs different paths between source and destination nodes are not very well separated: due to radio interference, data packets traveling over parallel paths can hinder each other, so that it is difficult to obtain any advantages. This issue can possibly be dealt with if specific mechanisms are used during the construction of the different paths, e.g. choosing paths that go over nodes that are outside each other's transmission range (see also subsection 2.4.3). However, such mechanisms would be quite complex, especially if we take into account the fact that nodes move, and paths that were originally sufficiently apart could move into each other's transmission range. In AntHocNet, we did not use such an approach. Instead, we keep the data routing coefficient high, and only use the very best paths. This way, we maximally exploit the best routes available.

5.3.7 Summary

In this section we have presented results of tests in which we switched on and off different components of the AntHocNet routing algorithm and varied its internal parameters. The aim was to investigate AntHocNet's internal working.

First, we have looked at the individual contribution to the algorithm's performance of the proactive route maintenance process and the reactive route repair mechanism. We have shown that the proactive route maintenance process has a strong positive influence on the performance over a range of different scenarios. On the other hand, the reactive route repair mechanism turned out to have a large positive contribution in large networks but very little or no contribution in small networks.

Next, we have investigated the use of different routing metrics. From the results, it was clear that the metric using the SINR was superior. The metric using delay and the one using a combination of delay and hops gave worse results. The worst results were for the number of hops metric, despite the fact that this metric lead to the use of the shortest paths and is very often used for routing in AHWMNs.

Then, we have investigated two parameters that are related to the speed of working of the proactive route maintenance process: we have done tests varying the send rate of proactive forward ants and varying the number of entries in the pheromone diffusion messages. Both tests indicated that a faster working proactive route maintenance process is better: sending ants more regularly and spreading out more information in each message during pheromone diffusion gave better results. Limits are present only when too much overhead is created. For example, sending more ants than one per second lead to rather bad results.

Finally, we have done tests varying the routing coefficient of proactive forward ants and data packets. In both cases, it turned out that increasing the coefficient lead to monotonically improving results. In the case of proactive forward ants, this shows that it is better to leave the task of exploring new paths to the pheromone diffusion process, and let the ants focus on the task of

controlling the obtained information and turn it into routes that can be used for data. In the case of data packets, it shows that it is difficult to get throughput advantage from spreading data over multiple paths, and that instead it is better to fully exploit the best routes found by the ants.

5.4 Conclusion

In this chapter we have provided an evaluation study of the AntHocNet routing algorithm. Tests were carried out in scenarios that are similar to those commonly used in the literature on MANET routing.

In a first set of tests, we compared AntHocNet to existing state-of-the-art routing algorithms. These included AODV, a reference reactive routing algorithm, OLSR, an important proactive routing algorithm, and ANSI, which is a representative of the class of ACO routing algorithms. Results showed that AntHocNet could outperform the other three algorithms over a wide range of different scenarios. Specifically, AntHocNet turned out to perform well in tests with increasing levels of mobility, to deal better than the other algorithms with different levels of data load, to cope well with sparse network situations, and to scale well to networks of increasing sizes. On the downside, AntHocNet's advantage was decreased when mobility got very high, and the algorithm was outperformed by AODV when very dense scenarios were used.

In a second set of tests, we investigated the internal working of AntHocNet. We switched on and off the use of different components of the algorithm, and varied various internal parameters. We found that the proactive maintenance process has a high contribution to the algorithm's performance over a range of scenarios, while the local repair mechanism did not have a positive effect in small networks, but was increasingly valuable in large ones. We also found that it is important that the proactive maintenance process works as fast as possible, as long as it does not produce excessive overhead. Among the possible routing metrics, we discovered that the SINR based metric lead to the best results. Finally, we saw that exploration in the proactive route maintenance process should be left to the pheromone diffusion process rather than to the ants, and that data packets should be forwarded over the best paths, with minimal data load spreading.