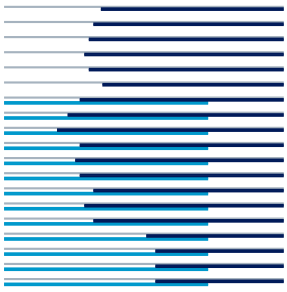
A New Approach for Integrating Proactive and Reactive Routing in Mobile Ad Hoc Networks

Frederick Ducatelle
Gianni Di Caro
Luca Maria Gambardella



Technical Report No. IDSIA-02-08

IDSIA / USI-SUPSI
Dalle Molle Institute for Artificial Intelligence
Galleria 2, 6928 Manno, Switzerland

A New Approach for Integrating Proactive and Reactive Routing in Mobile Ad Hoc Networks

Frederick Ducatelle
Gianni Di Caro
Luca Maria Gambardella

Abstract

In this article, we propose a new approach to integrate proactive and reactive routing in mobile ad hoc networks. Our work deploys a lightweight proactive algorithm that runs in the background offering a basic routing service, and a reactive algorithm that can be called on demand offering a connection-oriented routing service. The reactive algorithm relies for its working as much as possible on the routing information provided by the proactive algorithm, creating a synergy between the two parts of our system. The most interesting property of our system is that it allows the choice between proactive and reactive routing to be made for each session individually, by their source nodes. This allows to shift the load of data that is being routed proactively and reactively while the network is being deployed. Also, it gives network nodes a very fine-grained level of control over the routing process, and allows them to maximally exploit the complementary properties of proactive and reactive routing, for example by matching the choice of the routing approach to the needs of individual sessions. In a range of tests, we show the validity of our approach.

1 Introduction

In the last few years, a large number of routing algorithms have been proposed for ad hoc networks (see [2] for an overview). These algorithms are typically classified as being either proactive or reactive [20]. *Proactive algorithms* try to maintain correct routing information between all pairs of nodes in the network at all times. Examples include Destination-Sequenced Distance Vector routing (DSDV) [17] and Optimized Link State Routing (OLSR) [8]. *Reactive algorithms* take a different approach: they only gather and maintain routing information for nodes between which data sessions are going on (in this paper we will use the terms "session" and "flow" interchangeably). They are therefore also called *on-demand algorithms*. Examples of such algorithms include Dynamic Source Routing (DSR) [12] and Ad hoc On-Demand Distance Vector Routing (AODV) [18].

A large number of studies have been carried out to compare the performance of the various available routing algorithms, often focussing on the distinction between reactive and proactive algorithms (e.g., see [5] for an early work and [13] for a more recent one). While the results of these studies sometimes contradict each other, there seems to be a general consensus that reactive and proactive approaches both have their advantages, and that these depend on the deployment scenario of the network. Proactive algorithms usually deliver data packets with lower end-to-end delay, as routing information is always kept readily available for all destinations in the network, but might cause large amounts of overhead and/or incur significant packet losses when they are not able to keep up with all changes in the network. They typically work well in small networks with limited mobility. Reactive algorithms, on the contrary, often cause less overhead and obtain

higher packet delivery ratios, but can suffer from larger delays due to their on-demand nature. They usually outperform proactive algorithms in large and dynamic networks, especially when data traffic patterns are sparse.

The complementary nature of proactive and reactive routing strategies makes it difficult to choose a single algorithm for all possible ad hoc network applications. Hence, current standardization efforts in the Mobile Ad Hoc Networks (MANET) working group of the IETF [1] are focused on two different routing algorithms: a proactive one (OLSR) and a reactive one (DYnamic MANET On-Demand routing, DYMO, which is derived from AODV [7]). This way, an individual choice can be made for each ad hoc network deployment. Nevertheless, since the relative advantages of proactive and reactive algorithms are only understood in a qualitative, approximative way, it can still be difficult to make this choice, even with a specific ad hoc network application scenario in mind. Moreover, the situation might change during deployment, requiring to switch to a different routing algorithm. One possible solution to this problem is to combine proactive and reactive routing in a single network, as is done in hybrid and multi-mode routing algorithms (see section 2). In this paper, we follow this same solution, proposing a new approach to it. Also within the MANET working group, the possibility of taking a combined approach to routing is still kept open, "if significant commonality is observed" [1].

We present a novel approach for integrating proactive and reactive routing, called Integrated Routing Algorithm (IRA). An important property of our system is the fact that it allows nodes to make a choice between proactive and reactive routing for each data session individually. This allows for a very fine-grained and on-line tuning of the routing process. Such a detailed adaptive control over routing can be advantageous in a number of different scenarios. For example, nodes could adapt their routing behavior in order to implement a specific network wide policy, or in order to react to changes in the properties of the network, such as the data traffic load or the network mobility. Another appealing application is Quality-of-Service (QoS) provisioning, whereby nodes can adapt their routing choices based on the requirements of individual sessions. For example, a session that requires low delay but can support certain levels of packet loss can be routed proactively, while a session that requires low levels of packet loss can be routed reactively. In this way, the choice between reactive and proactive routing can be linked to the problem of QoS provisioning. In this article, we focus on the description of the mechanisms applied in IRA for integrating proactive and reactive routing on a per-session basis.

The working of our integrated routing service is as follows. We have on the one hand a proactive algorithm that runs continuously in the background and is bounded in terms of the control traffic it generates, and on the other hand a reactive algorithm that can be deployed on-demand. The proactive algorithm is kept lightweight: all proactive information is piggy-backed on top of periodic beacon messages, routing update messages are limited in size, and no extra messages are used in response to disruptive events. This means that the overhead created per node is constant with respect to the change rate of the network, the amount of data sent and the size of the network. The downside of this approach is that the performance of the proactive algorithm can degrade in certain scenarios, mainly leading to increased levels of data packet loss. Therefore, in accordance to QoS terminology, we say that the proactive routing algorithm offers a best effort service for packet delivery, characterized by a low end-to-end delay but possibly a high packet loss rate. If such a service is not opportune, the reactive routing algorithm can be used. The reactive algorithm offers a connection-oriented service: it executes a route setup process at the start of a data session and employs route improvement and repair mechanisms for as long as the session is going on. As a consequence, the reactive routing algorithm gives lower loss rates but also higher delay. An important aspect of our proposed integrated approach is that the reactive algorithm makes use of the routing information offered by the proactive algorithm whenever possible, such as for example to support its route improvement or repair attempts. This way there exists a synergy between both algorithms, and the proactive algorithm remains useful even when all data are sent reactively.

In earlier work on the AntHocNet routing algorithm [9], where a very simple proactive mechanism supports a process for route improvement of a mainly reactive algorithm, we have observed that such a strategy can be beneficial even when the proactive information is relatively inaccurate.

Our work gives a number of interesting contributions. First of all, it provides a new way of integrating proactive and reactive routing in ad hoc networks. Second, it allows the choice between either of these to be made individually and autonomously by the nodes, avoiding issues of network-wide coordination. Third, it gives the possibility to make a different choice for each individual session, rather than expecting a single decision per node or for the whole network together (as is common in other algorithms that combine proactive and reactive routing), which means that a much more fine-grained level of control over the routing process is possible. Finally, the possibility to link this choice to a task like QoS provisioning allows to exploit inherent differences between proactive and reactive routing, and to make informed decisions that serve the applications making use of the network.

The rest of this article is organized as follows. First, in section 2 we discuss the related work in this area. Then, in section 3 we discuss the IRA routing approach, in particular giving details about its proactive and reactive parts, and how these parts work together and interact. Next, in section 4, we present results of simulation studies in which we investigate the working of our approach in detail. Finally, in section 5, we discuss future work to be done.

2 Related work

The fact that proactive and reactive routing algorithms each have their own advantages and that these are to some extent complementary has been recognized since quite some time in the field of ad hoc networking [20]. There has therefore always been an interest in developing systems that let proactive and reactive routing work together in the same network.

Some early work in this area was focused on the development of hybrid routing algorithms. These are algorithms that combine elements from both proactive and reactive routing in order to get the "best of both worlds". One example of such an algorithm is the Zone Routing Protocol (ZRP) [10], in which nodes maintain proactive routing information for destinations in their immediate neighborhood (their proactive zone) and use reactive routing for destinations that are further away. Another example is the Sharp Hybrid Adaptive Routing Protocol (SHARP) [19], which is based on the same ideas, but is more adaptive as it gives nodes the possibility to autonomously change the size of their proactive zone according to their own needs.

A related line of research is multi-mode routing algorithms, where nodes switch between proactive and reactive routing based on properties of the network. In [21], the authors present a routing algorithm that uses proactive routing for a subset of all network nodes, namely those that are reachable over relatively stable links and those that are likely to be the destination of new data sessions, and uses reactive routing for all other nodes. More recently, in [11] a system is proposed in which nodes individually decide whether they will be involved in a proactive routing process or not. This decision relies on measured statistics about the network situation. Only data packets for which both the source and destination node participate in the proactive process can make use of proactive routing information. All other packets rely on reactive routing. Finally, in [15] the authors propose an approach in which all nodes of the network run the same algorithm, which is chosen from a set of proactive and reactive algorithms. The article is focused on the mechanisms used to let all nodes of the network switch between routing algorithms simultaneously.

Compared to the work we present here, none of the approaches described above offers the same fine-grained level of control for the decision between proactive and reactive routing: they allow decisions to be made on a per-node basis, or for the whole network together, while our approach supports decisions on a per-flow basis.

3 An Integrated Routing Algorithm

In this section, we describe our new integrated routing algorithm in detail. We start with an overview of the working of the entire IRA system. After that, we first give details about the working of the proactive part of the system and then about the reactive part. We end with a short discussion of how data forwarding is done.

3.1 Overview of the system

The working of IRA relies on a proactive algorithm that runs continuously in the background, and a reactive algorithm that can be deployed on demand.

The proactive algorithm disseminates its routing information via periodically locally broadcast update messages, which also serve as beacon messages. These messages are limited in size, and no extra messages are sent in response to disruptive events. As a consequence, the overhead created for the proactive algorithm by each node is constant with respect to the rate of change of the network, the amount of data sent, and the size of the network. The downside of this design is that the performance of the algorithm can degrade, especially in networks with challenging properties, such as very large or highly dynamic networks. The proactive algorithm can therefore only offer a best effort service for data delivery. Details about the proactive algorithm are given in subsection 3.2.

The reactive algorithm offers a connection-oriented service, primarily focused on limiting data packet loss rates. It can be used whenever the best effort service offered by the proactive routing information is not sufficient. The algorithm relies on a route setup process to gather initial routing information at the start of a new data session. Then, during the course of the session, it performs route improvement and, when necessary, route repair operations. Details about the reactive algorithm are provided in subsection 3.3.

An interesting aspect of the IRA system is that the reactive algorithm makes extensive use of the information provided by the proactive algorithm. It does so during route setup, route improvement and route repair. The use of the proactive information makes the reactive algorithm both more effective and more efficient, as is shown in the evaluation of the algorithm in section 4. This means that the proactive algorithm running in the background always gives a contribution to the system as a whole, even when all data is routed reactively.

3.2 The proactive algorithm

Our proactive algorithm implements a multipath version of Bellman-Ford routing [3]. An important aspect of the algorithm is that it needs to function as a lightweight background process in ad hoc networks, and hence is designed to work efficiently under all circumstances. In what follows, we first describe the general working of the proactive algorithm, then we discuss properties of the routing information update process, next we describe how we obtain loop avoidance, after that we talk about the routing metric, and finally we give some details about the composition of update messages and their size.

3.2.1 General working of the proactive algorithm

The basic working of the algorithm is as follows:

1. Each node i in the network maintains a routing table T_i, which has an entry t_{ij}^d for each known destination d and each available next hop j. t_{ij}^d contains an estimate c_{ij}^d of the cost of the route towards destination d going over next hop j (details about the applied cost metric are given later in this subsection).

2. At periodic intervals (set to $1s$), node i broadcasts an update message containing for maximum m of the destinations in its routing table the best routing estimate $\hat{c}_i^d = \min_{j \in N_i}(c_{ij}^d)$ taken over all the possible next hops j in its set of neighbors N_i (if it does not have any routing estimate to a certain destination, this is indicated with a cost of infinity). m is a constant that limits the size of the update messages. If there are less than m destinations in the network, all of them are included in each update message. Otherwise, update messages include subsets of m destinations in a round-robin fashion.

3. Each of the neighbors k of node i receives the update message, and uses it to calculate for each of the destinations d its own new estimate c_{ki}^d for the cost of the route over next hop i to d. It does so by adding the estimate $\hat{c}_i^d$ received from i to the estimated cost of going from k to i. If i indicated in the update message that it does not have any valid route to d, k sets c_{ki}^d to infinity.

3.2.2 The routing information update process

The main difference between the proactive mechanism described here and other Bellman-Ford routing algorithms, such as e.g. the DSDV [17] algorithm for ad hoc networks, lies in the way routing updates are spread. As indicated in step 2 above, we only use *periodic update messages* that are *limited in size*. This means that the routing overhead is not influenced by the occurrence of disruptive events, as no extra routing information is generated to deal with them (contrary to DSDV, where substantial changes are broadcast faster). It also means that routing overhead is not dependent on the size of the network: if the number of possible destinations exceeds the constant factor m, information about them is spread over multiple subsequent update messages.

In terms of implementation, routing update messages are piggy-backed on top of beacon messages. These are short messages that are broadcast periodically by the nodes of the network in order to inform neighboring nodes about their presence. Beacon messages are commonly used in ad hoc networks to support the monitoring of link availability. Including the update messages inside beacon messages has two important advantages. On the one hand, since one message serves two purposes, the total number of generated control packets can be reduced. This is important because ad hoc networks usually rely on contention based algorithms at the MAC layer, whereby channel access activities can get quite costly, and it is therefore better to send one large message than two smaller ones. On the other hand, the size of the beacon messages is increased with the addition of routing information, which can make them more reliable for link detection [6]. This is because small beacon messages have more chances of getting forwarded correctly over lossy wireless links than larger data packets, and can therefore indicate the presence of a connection while this connection can actually not be used for data.

The downside of the proposed routing update mechanism is that routing information is spread rather slowly through the network. While this does not mean that the algorithm does not converge to correct routing information eventually (as is proven for Bellman-Ford algorithms [3]), it does entail that this convergence can be slow, so that some of the routing estimates maintained in the nodes can temporarily be out of date, indicating erroneous routes. Data packets following erroneous routes can find themselves in three different scenarios. The first is that packets are delivered over a sub-optimal path, the second is that they are dropped because they reach a dead end, i.e. a node where no more routing information with respect to their destination is available, and the third is that they go in a loop, i.e. they return to a node they have visited before. The first two scenarios lead to a degradation of the routing performance: longer delivery times and/or packet loss. This is related to the design choice we made for the development of the proactive algorithm: we want to keep the algorithm lightweight and efficient, and are prepared to accept that its performance degrades under certain conditions. This is why we say the proactive routing

algorithm can only provide a best effort service for data forwarding, and we need the reactive algorithm when better performance is required. The third scenario, of data packets going in loops, is more problematic, as this can lay a heavy burden on the limited bandwidth and processing resources of the ad hoc network. Therefore, we implemented a simple mechanism to avoid loop formation.

3.2.3 Loop avoidance

The loop avoidance mechanism is based on the use of sequence numbers, similar to the method proposed in the DSDV routing algorithm. However, since DSDV is a single path Bellman-Ford algorithm (it maintains only the best next hop for each destination) while ours is multi-path (we maintain routing information over each outgoing next hop), we need to use the sequence numbers in a different way. The main idea is to have all destination nodes issue sequence numbers and spread these together with the routing information, and to let data packets only follow routes of increasing sequence numbers or decreasing costs.

The general working of the mechanism is as follows:

1. Each node i maintains a local sequence number s^i. It also maintains in each routing table entry t^d_{ij} a sequence number s^d_{ij}, which is the last sequence number received for destination d over next hop j. Finally, it maintains for each destination d a sequence number s^d_i, which is the highest sequence number i has forwarded for d in its update messages, and a cost value c^d_i, which is the lowest cost forwarded for d related to this sequence number.

2. For each periodic routing update message, i increments the local sequence number s^i by 1 and includes it in the message. Then, it adds the best routing information for each destination d to the update message (as is described earlier), and adds to this the sequence number s^d_{ij} related to the best next hop j towards d. Finally, it updates s^d_i, its local record of the highest sequence number it has forwarded for destination d, and the associated lowest cost c^d_i as follows: if $s^d_i < s^d_{ij}$, s^d_i is set to s^d_{ij} and c^d_i is set to c^d_{ij}; otherwise, if $s^d_i = s^d_{ij}$ and $c^d_i > c^d_{ij}$, c^d_i is set to c^d_{ij}.

3. A node j receiving an update message from i stores the sequence number s^i in its routing table entry t^i_{ji} as s^i_{ji}, the last sequence number received for destination i over next hop i. Then, for each destination d mentioned in the update message, it sets s^d_{ji} to the received sequence number.

4. Data packets arriving in a node i for a destination d are only forwarded to a next hop j if $s^d_{ij} > s^d_i$ (the sequence number for d related to j in i is higher than the highest that i has forwarded for d), or if $s^d_{ij} = s^d_i$ and $c^d_{ij} \leq c^d_i$ (the sequence number for j is the same as the highest i has forwarded and the cost for j is equal or lower than the lowest i has forwarded for this same sequence number).

To understand how the above described mechanism ensures loop freedom, consider a data packet following a path $P = \{1, 2, \ldots, n\}$. We indicate by s^n_i and c^n_i the highest sequence number and lowest cost forwarded for destination n by a node i along the packet's path, and by s^n_{ii+1} and c^n_{ii+1} the sequence number and cost stored in i for destination n over the outgoing link to $i+1$, which is followed by the data packet. From step 2 we know that $s^n_{i-1i} < s^n_i$, or $s^n_{i-1i} = s^n_i$ and $c^n_{i-1i} > c^n_i$ (we use $>$ rather than $\geq$ because the update of c^n_{i-1i} in node $i-1$ involves adding the cost of going from $i-1$ to i, which is greater than 0). From step 4 we know that $s^n_i < s^n_{ii+1}$, or $s^n_i = s^n_{ii+1}$ and $c^n_i >= c^n_{ii+1}$. Now, suppose the hop from i to $i+1$ closes a loop, i.e. node $i+1$ is the same as a node h that comes before i. Then, we must have that

$s^n_{ii+1} \leq s^n_{i+1} = s^n_h \leq s^n_{ii+1}$. This is only possible if all of these sequence numbers are equal, but then we should have $c^n_{ii+1} > c^n_{i+1} = c^n_h > c^n_{ii+1}$, which is clearly impossible.

3.2.4 The routing metric

In principle, the proactive routing algorithm (and also the reactive algorithm described below) can be implemented using a variety of different routing metrics. Here, we use a metric based on the signal-to-noise ratio (SNR). SNR indicates the ratio between the signal strength over a link and the background noise. SNR is a crucial factor defining the success of a wireless reception. When SNR is high, reception has a high probability of being successful, whereas when it is low, reception is impossible. In between there is a range for which reception is possible with some probability. Here, we are not interested in fine variations in the SNR level, but rather in a coarse grained distinction between "good" and "bad" wireless links. Therefore, we apply a simple approach using a critical SNR value SNR_c as threshold: links with an SNR higher than SNR_c are given a cost 1, while other links get a cost c_{const}, which is a constant higher than 1. In simulation tests using IEEE 802.11b radios sending at 2Mbps, we empirically set SNR_c to 17dB and c_{const} to 3. The value of 17dB for SNR_c is in line with critical values of SNR found in empirical research on wireless LANs using the same radio technology [14]. The measurement of SNR levels are done locally by the nodes.

3.2.5 The structure and size of update messages

Here, we give some the details about the structure and size of the routing update messages. Each message contains a sequence number (2 bytes), a number indicating how many destinations the message reports about (1 byte), and an array of entries containing information about individual destinations. Each of these entries contains a node destination address (4 bytes, if we use IP addresses), a number indicating the routing cost estimate to this destination (2 bytes), and a sequence number (2 bytes) . The size of an entry is therefore 8 bytes. The total size of the update message is then $(3 + l * 8)$ bytes, whereby l is the number of destination entries, which is lower than or equal to m, the maximum number of destinations per update message. In our tests, we set m to 20, which results in a maximum message size of 163 bytes.

3.3 The reactive algorithm

The reactive algorithm provides a connection-oriented service for routing. At the start of a data session, it executes a route setup process to build an initial route. During the course of a session, it applies route improvement mechanisms to try to adapt the currently used route to changes in the ad hoc network. Finally, it has a number of mechanisms to deal with link failures, including warning messages and a route repair process. In all aspects of its working, the reactive algorithm tries as much as possible to make use of the information that is made available by the proactive algorithm, in order to get a synergy between both components of the integrated IRA system. In what follows, we discuss each aspect of the reactive algorithm in detail.

3.3.1 Route setup

The route setup process constructs an end-to-end route between the source and destination of a data session. It is used at the start of a new session, or whenever the source of an ongoing session falls without routing information. If a route setup process fails (i.e., no answer is received within a certain time), the process is restarted. Each route setup is given maximally three attempts.

To initiate a route setup process, the source node s creates a route setup message. The aim of this message is to find a path towards a destination d. At any node i (including s), the message

can be forwarded in two possible ways. The first way is by using the proactive routing information related to d, i.e. the message is forwarded to next hop $j = argmin_{j \in N_i}(c_{ij}^d)$. The second way is by broadcasting, i.e. the message is forwarded to all next hops $j \in N_i$ simultaneously. During the first attempt of a route setup, nodes use the proactive information whenever it is available, and broadcast otherwise. Forwarding over the routes indicated by the proactive information can improve efficiency, as it limits the spreading of the route setup message compared to broadcasting. However, when the proactive information is inaccurate, this can lead the route setup message in the wrong direction and prevent it from finding the destination. Therefore, in case the first route setup attempt fails and a second and third attempt are needed, only broadcasting is used for forwarding. Broadcasting produces multiple copies of the same route setup message. In order to limit the produced overhead, nodes only process the first copy they receive and discard all other ones.

On its way from s to d, the route setup message collects a list $P = [s, \ldots, i, \ldots, d]$ of all the nodes it has visited. After reaching d, it returns to s retracing P. On its way back, it gathers cost information for each of the links along P (measured locally by the nodes according to the metric described in subsection 3.2) and uses it to set up reactive routing information in each of the nodes $i \in P$ and in s. This routing information consists of a reactive next hop rn_i^d for destination d and a reactive cost rc_i^d related to it. Once the route setup message gets back to s, data forwarding over the new route can start.

3.3.2 Route improvement

Route improvement takes place during the whole duration of a reactively routed data session. Its aim is to try to find better routes for the session, in order to adapt to changing conditions in the ad hoc network.

The working of the route improvement process is as follows. Each node s that is the source of a reactively routed data session checks at regular intervals (set to $1s$) whether there is proactive routing information for the destination d of the session that is better than the currently available reactive route cost, i.e. $\min_{j \in N_i}(c_{ij}^d) < rc_i^d$. If this is the case, the proactive routing information gives an indication that a better route might be possible, and the route improvement process is started. s creates a route improvement message, which is routed towards d following the proactive routing information: in each node i, the message is forwarded to next hop $j = argmin_{j \in N_i}(c_{ij}^d)$. In contrast to route setup messages, route improvement messages are never broadcast; if they arrive in a node where no proactive routing information is available, they are discarded. This is in order to limit overhead. Once a route improvement message reaches d, it behaves like a route setup message: it traces its path back to s, measures link qualities on its way, and updates in each node i the reactive next hop rn_i^d and reactive cost rc_i^d.

3.3.3 Dealing with link failures

Link failures can be detected in two different ways. One is through link layer feedback, whereby the link layer informs the routing algorithm about the failed transmission of a unicast packet. The other is through the use of periodic beacon messages, whereby a node assumes the link to a neighboring node to have failed when it has not received any message from that node for a number of beacon send periods (we set this number to 2).

When a node i detects the failure of a link to a neighboring node j, it removes any entries related to next hop j from its proactive and reactive routing information. Then, if the link failure lead to the failure of any reactively maintained routes, it needs to take further actions, which depend on the specific situation. If i is the source of a failed reactive route, it starts a new route setup process (see earlier). If, on the other hand, i is an intermediate node for a reactive route,

it reacts with a route repair message if it currently has data messages to send to the destination of the failed route, and with a route failure message otherwise. A route repair message is similar to a route setup message, in the sense that on its way to the destination it follows proactive routing information whenever possible and is broadcast otherwise, and on its way back, it follows its original path back and updates routing information. The main differences between the two types of messages are that the maximum number of broadcasts for repair messages is limited (to 2), in order to reduce the generated overhead, and that only one attempt is possible. A route failure message basically contains a list of all the destinations that i lost a reactive route to, and is broadcast to all i's neighbors. A neighbor j of i receiving this message checks whether it had a route over i to any of the mentioned destinations, and, if this is the case, removes this route and creates its own route failure message. If j is the source of any of the lost routes, it starts a new route setup process.

A last mechanism to deal with link failures is the warning message. This is used whenever a node receives reactively routed data packets which it does not have any reactive routing information for. This can happen when an earlier sent route failure message did not get received properly by an upstream node on a route. In that case, it sends a warning message to the sender of the erroneously routed data packet, indicating that it does not have routing information for the requested destination.

3.4 Data forwarding

When a new data session is started, its source node decides whether it will be routed proactively or reactively. Every packet of the session is subsequently classified according to this decision, and all nodes in the network forward the packet either proactively or reactively based on this classification. In order to mark a packet's classification, we use the type of service (ToS) field of the IP header. Here, we foresee two possibilities: either we use the precedence bits of the ToS field, or we use one of the unused bits at the end of the field. When we use the precedence bits, we mark proactively routed packets with the precedence bit pattern 000 ("routine" traffic according to the IP precedence classification, or "best effort" traffic according to the DSCP classification [16]), while reactive routed packets are marked with the pattern 001 ("priority" traffic according to the IP precedence classification, or "AF1" traffic according to the DSCP classification). The precedence bits of the ToS field also influence queueing and drop priorities, so that the reactively routed packets get an improvement of their QoS experience compared to proactively routed packets. This approach is therefore useful when we want to use the IRA system in combination with QoS provisioning. If this is not desired, we can instead make use of the last bit of the ToS field, which is normally unused and therefore has no consequences for queueing and dropping priorities. We set this bit to 1 to indicate reactive routing, and to 0 to indicate proactive routing. In the tests described further, we normally use the first method, based on the precedence bits.

4 Experimental evaluation

In this section, we present the results of a number of experiments in which we evaluate the integrated IRA approach. All experiments are carried out in simulation, using QualNet 4.0 [22]. In what follows, we first describe the setup of our simulation studies, and then discuss experimental results.

4.1 Setup of the experiments

In our experimental setup, a mobile ad hoc network of 100 nodes is deployed in a rectangular, open area of $2400 \times 800m^2$. The nodes move according to the random waypoint (RWP) mobility model [12]. We vary the maximum speed from 1 up to $20m/s$, in order to get scenarios of increasing difficulty. We use a minimum speed of $1m/s$ (here we follow the recommendation given in [23] in order to avoid problems of decreasing average speeds in RWP) and a pause time of $30s$. Each experiment has a duration of $1800s$, whereby during the first $900s$ no communication takes place, so that the node distribution can get to a steady state (for an extensive discussion on the node distribution under RWP mobility, see [4]). Each experiment is repeated 10 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 900 and $1000s$ 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, as is common for open space scenarios. At the physical layer, we use the IEEE 802.11 protocol operating with a data transmission rate of $2Mbit/s$. The estimated radio range is $250m$. At the MAC layer, we use the IEEE 802.11 DCF protocol. Finally, at the transport layer, we use the UDP protocol.

4.2 Experimental results

In what follows, we first investigate the performance of the proactive and reactive modes of operation of our algorithm, and how this is related to per-session service differentiation. Then, we study the relative competitiveness of our algorithm by comparing it to existing reference routing algorithms. Next, we focus on other performance measures: we investigate the average delay jitter, which is an important metric in the field of QoS provisioning, and the created overhead, which concerns the efficiency of the system. Finally, we investigate in how far we obtain the desired synergy between proactive and reactive routing, i.e., in how far the reactive routing algorithm is able to profit from the availability of the proactive routing information.

In order to evaluate the different levels of performance offered by IRA to proactively and reactively routed sessions, we evaluate the packet delivery ratio (the fraction of correctly delivered data packets versus the total number of sent packets, equal to one minus the packet loss ratio) and the average end-to-end delay in a number of different scenarios. We distinguish between scenarios where all 20 sessions are sent reactively, scenarios where all 20 session are sent proactively, and mixed scenarios where 10 sessions are sent reactively and the other 10 are sent proactively. The results are shown in figure 1, whereby the results for the mixed scenarios are split up: we show separate performance graphs for reactively and proactively routed sessions. From the figure, it is clear that the algorithm's ability to choose between proactive and reactive routing for each individual session results in a differentiation of the service provided to the sessions: reactively routed data sessions receive a higher delivery ratio (lower loss rate) than proactively routed sessions, but also a higher delay. The difference in terms of delivery ratio remains stable as the network gets more dynamic, while the difference in terms of delay grows. When we compare the performances between the mixed scenarios and the other ones, we can see that reactively routed sessions get better performance when less than 100% of the traffic is sent reactively, while proactively routed sessions get worse performance when less than 100% of the traffic is sent proactively. This is because the reactively sent data packets get priority in queueing and forwarding (see subsection 3.4), and therefore any increase in the number of reactively routed data sessions increases the competition for resources in the network.

In order to get an idea of the relative competitiveness of our approach, we compare to DYMO and OLSR (version 2). These are two important references in the field: DYMO is the follow-up

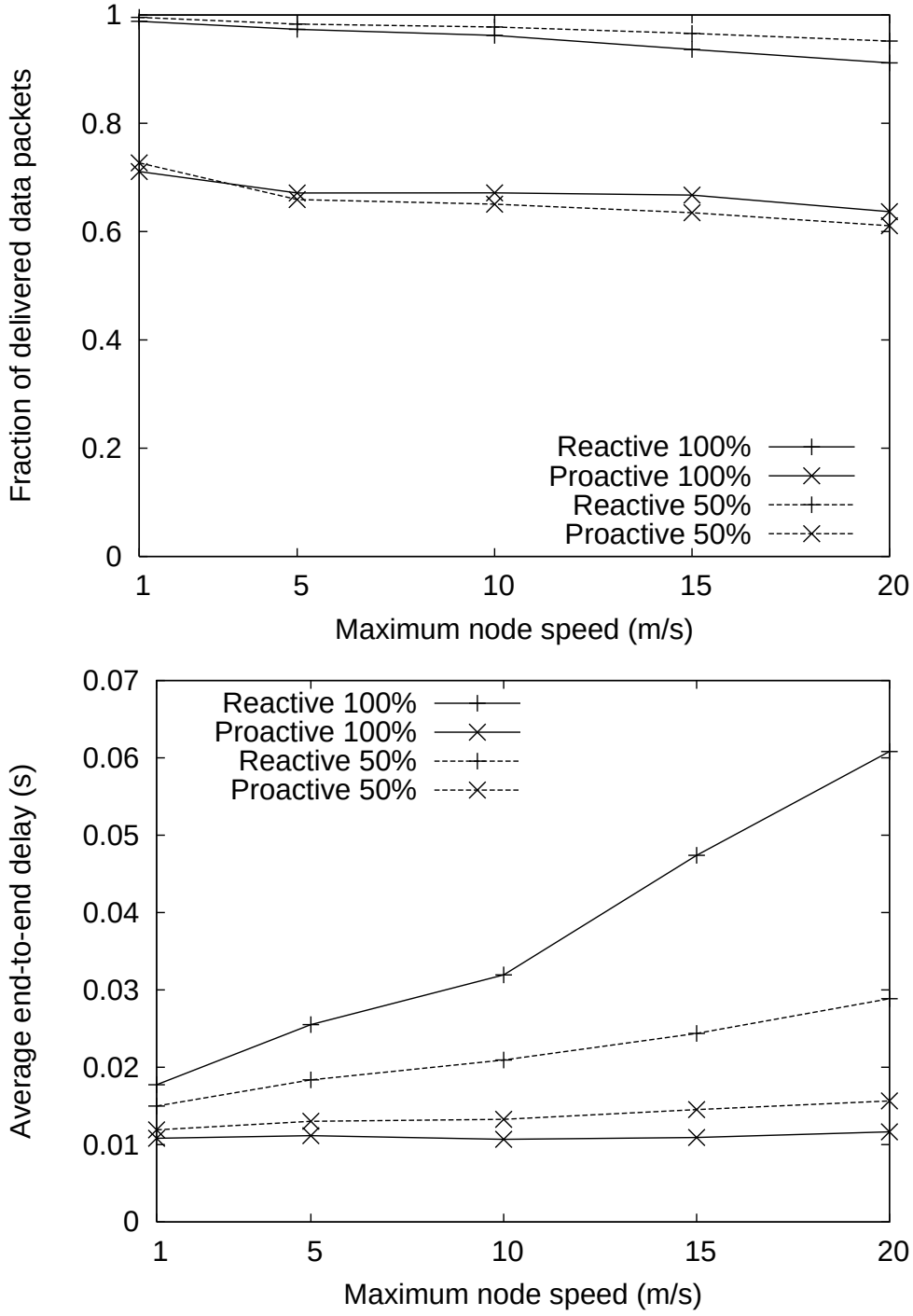


Figure 1: The delivery ratio and average end-to-end delay for the case where all sessions are sent reactively ("Reactive 100%"), the case where all sessions are sent proactively ("Proactive 100%") and the mixed case where half of the sessions are sent reactively and the other half are sent proactively (the performance for the reactively sent sessions is indicated by "Reactive 50%" and for the proactively sent sessions by "Proactive 50%").

of the AODV routing protocol, the most cited algorithm for routing in ad hoc networks, and is currently the candidate for standardization by the IETF as reactive ad hoc routing protocol. OLSR is the candidate for standardization as proactive ad hoc routing protocol. The implementations of DYMO and OLSR are provided in QualNet 4.0. The results of our comparisons are shown in figure 2. Here, we use a single graph for the performance of the reactive and proactive sessions in the mixed test case. As can be seen from the figure, the performance of the integrated IRA approach is quite good compared to these reference algorithms. In terms of delivery ratio, IRA's reactive algorithm outperforms DYMO, while its proactive algorithm outperforms OLSR. In terms of average end-to-end delay, both DYMO and OLSR score less well than the different modes of operation of IRA. Moreover, the performance of DYMO degrades faster than for IRA's reactive algorithm. The interesting aspect of these results is that IRA functioning in reactive mode gives better performance than the reference reactive algorithm, and IRA functioning in proactive mode gives better performance than the reference proactive algorithm. Moreover, different from these reference algorithms, IRA also gives the possibility to switch between reactive and proactive routing.

Now, we investigate some other performance measures. The first of these is average delay jitter. This is an important metric in the area of QoS provisioning. It concerns the variation in the time interval between the arrivals of subsequent packets of the same session. It is calculated as $\sum_{i=2}^{n} |(t_i - t_{i-1}) - (t_{i-1} - t_{i-2})|$, 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 for one session. We again do tests with 100% of the traffic routed reactively, tests with 100% routed proactively, and mixed tests of 50% routed reactively and 50% proactively. Also, we again include results for DYMO and OLSR for comparison. As can be seen in figure 3, the results for jitter follow those for the delivery ratio, with reactively routed sessions getting a better service than proactively routed sessions. This shows that the connection-oriented service offered by the reactive algorithm provides more stability for data packet delivery than the best-effort service of the proactive algorithm. Compared to DYMO and OLSR, we can see that our algorithm gives again superior results.

We also investigate the overhead ratio, both in terms of packets and bytes. The overhead ratio is calculated as the amount of control packets/bytes forwarded divided by the number of data packets/bytes generated. The results are given in figure 4. They show that our algorithm is relatively efficient. The overhead created by the proactive algorithm can be observed by looking at the test case where all sessions are routed proactively (since there the reactive algorithm is never called). We can see that it is constant with respect to node mobility, which was a design objective of the system. The overhead created by the reactive algorithm can be observed in the case where all sessions are routed reactively, by subtracting the overhead created by the proactive algorithm (since even when all sessions are routed reactively, the proactive routing algorithm runs in the background). We can see that this overhead is relatively small, and increases slowly with node mobility. OLSR produces more overhead than our approach in all its different modes of operation, and for both overhead measures. DYMO produces more overhead than IRA in terms of packets, and this overhead grows faster. In terms of bytes, the overhead produced by DYMO is lower, but it grows faster with node mobility. As pointed out earlier in subsection 3.2, limiting the overhead in number of packets is important in ad hoc networks when a contention based MAC layer is used.

Finally, we investigate to what extent the reactive algorithm is able to profit from the availability of proactive routing information. We consider the case where all sessions are routed reactively, and compare the performance of the full IRA system with that of IRA without the proactive algorithm running in the background (we obtain this by setting parameter m, the number of entries per update message, to 0). In this case, the reactive algorithm cannot rely on the guidance by proactive information for route setup, route improvement and route repair. In figure 5, we show results for delivery ratio and end-to-end delay. For both measures, the reactive algorithm does much better when it has the support of proactive information. Also, it is interesting to see that

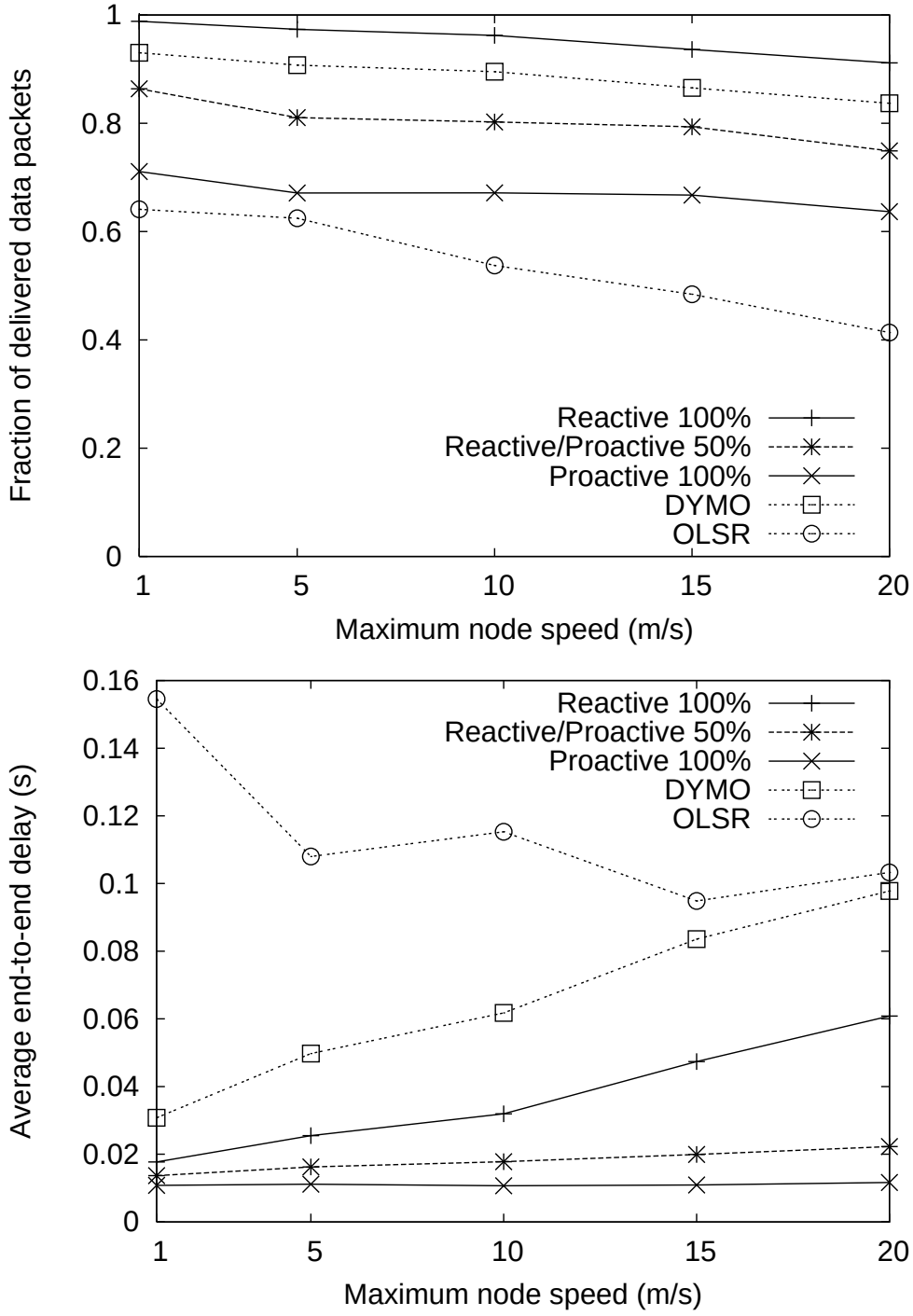


Figure 2: The delivery ratio and average end-to-end delay for the case where all sessions are sent reactively ("Reactive 100%"), the case where all sessions are sent proactively ("Proactive 100%") and the mixed case where half of the sessions are sent reactively while the other half are sent proactively ("Reactive/Proactive 50%"), as well as for the reference algorithms DYMO and OLSR.

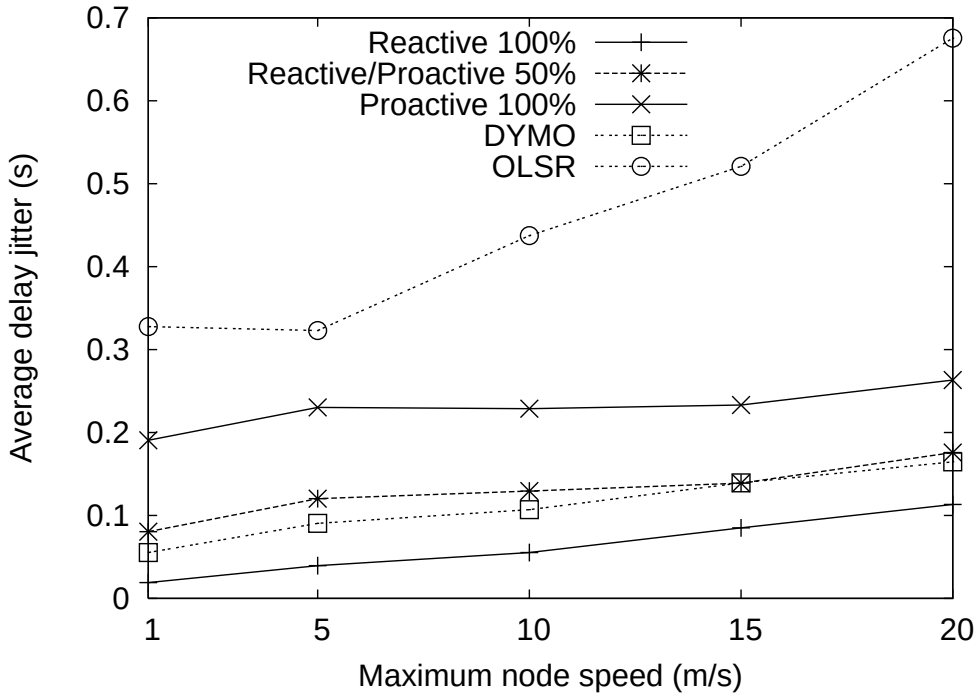


Figure 3: The average delay jitter for the case where all sessions are sent reactively ("Reactive 100%"), the case where all sessions are sent proactively ("Proactive 100%") and the mixed case where half of the sessions are sent reactively while the other half are sent proactively ("Reactive/Proactive 50%"), as well as for the reference algorithms DYMO and OLSR.

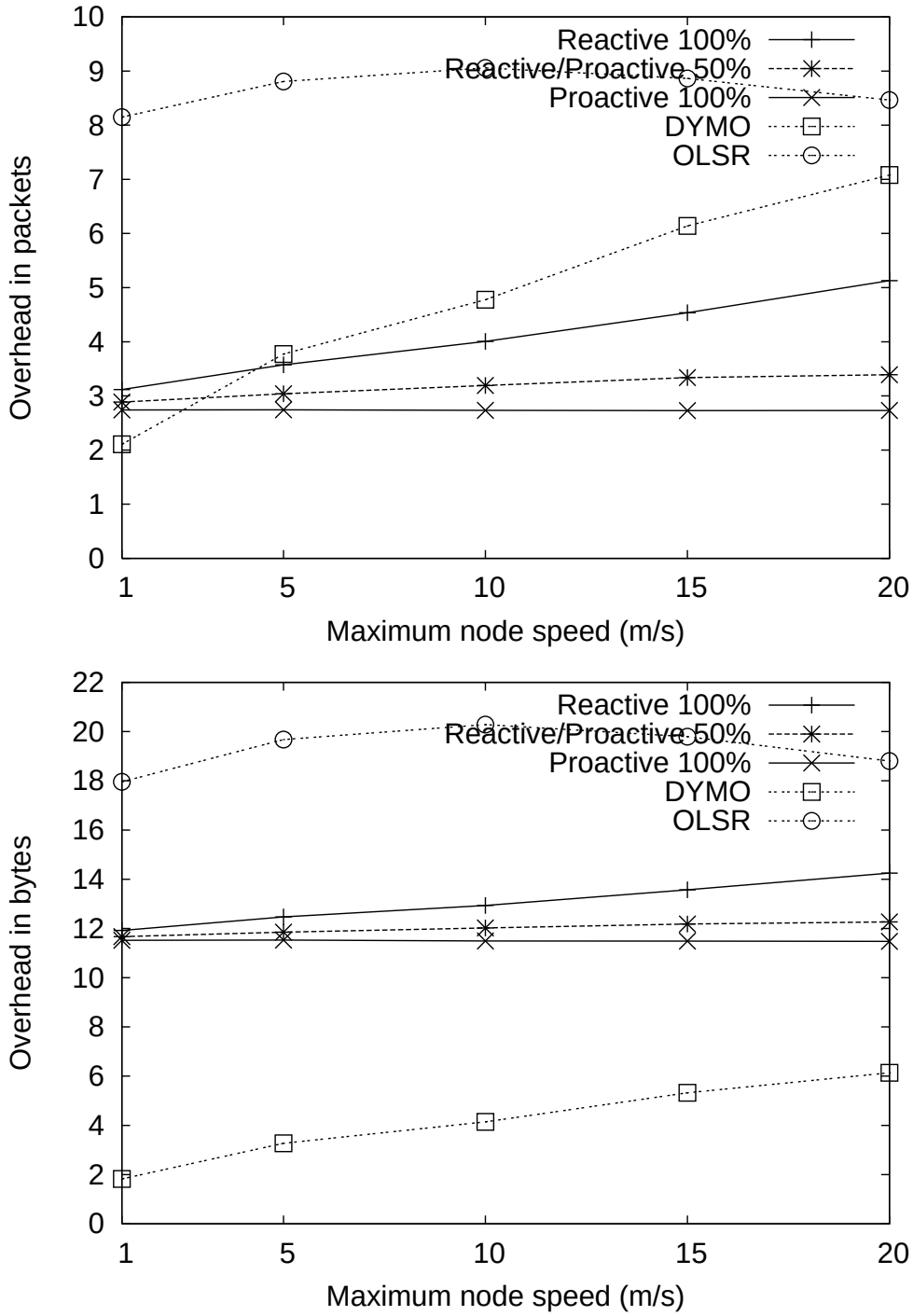


Figure 4: The overhead ratio, in packets and in bytes, for the case where all sessions are sent reactively ("Reactive 100%"), the case where all sessions are sent proactively ("Proactive 100%") and the mixed case where half of the sessions are sent reactively while the other half are sent proactively ("Reactive/Proactive 50%"), as well as for the reference algorithms DYMO and OLSR.

the difference in performance grows for increasing levels of node mobility, even though the delivery ratio of the proactive algorithm shows a decrease in this case (see figure 1).

5 Conclusions and future work

We have described a new approach to integrate proactive and reactive routing. The proactive system is kept lightweight, with constant overhead per node, and runs in the background offering a basic, best-effort routing service. The reactive algorithm can be called on demand, and offers a connection-oriented service including route setup, route improvement and route repair operations. A synergy between the two algorithms is created since the reactive algorithm makes use of the information gathered by the proactive algorithm whenever possible. The choice for using proactive or reactive routing is made autonomously by the nodes of the network, and can be different for each individual session. This allows nodes to involve the requirements of individual sessions in this choice. In a range of tests, we show that our algorithm gives good performance in its different modes of operation, and that it can give different types of service to individual sessions according to the chosen routing approach. We also show that the algorithm is efficient, and that the synergy between reactive and proactive routing pays off.

Future work will be focused in two different directions. On the one hand, we will extend the current integrated routing approach to incorporate some new features, and on the other hand, we will integrate it with a framework for QoS provisioning, as was suggested in the introduction to this paper. A first new feature we want to introduce is a monitoring module, which would give the system direct feedback about the performance delivered by the proactive and reactive routing components. A second feature is the adaptivity of the proactive system: in case there are many nodes in the network, one could decide to include only a subset of these as possible destinations in the proactive process, in order to reduce the complexity of the task performed by the proactive system. Such adaptation could be done online and in a distributed way. A last feature would be to add the possibility to use other routing algorithms for the integration; in particular, we could use derivations of OLSR and DYMO for the proactive and reactive part, in order to stay closer to current standardization efforts of the IETF MANET group. In terms of QoS provisioning, we would like to integrate our system with a QoS framework for ad hoc networks. This would allow applications to express QoS requirements for their data sessions, which could then directly influence the tuning of the routing process. Our preference goes to a DiffServ style framework, where QoS is provided according to a number of forwarding classes, which could be linked to different approaches to routing. Related to this, we would extend the routing process to include more different metrics, so that a wider range of QoS requirements can be served.

Acknowledgments

The authors would like to thank Liliana Carrillo for useful input and fruitful discussions.

References

[1] The internet engineering task force mobile ad-hoc networking page (MANET). Available from: `http://www.ietf.org/html.charters/manet-charter.html`.

[2] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks*, 2:1–22, 2004.

[3] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

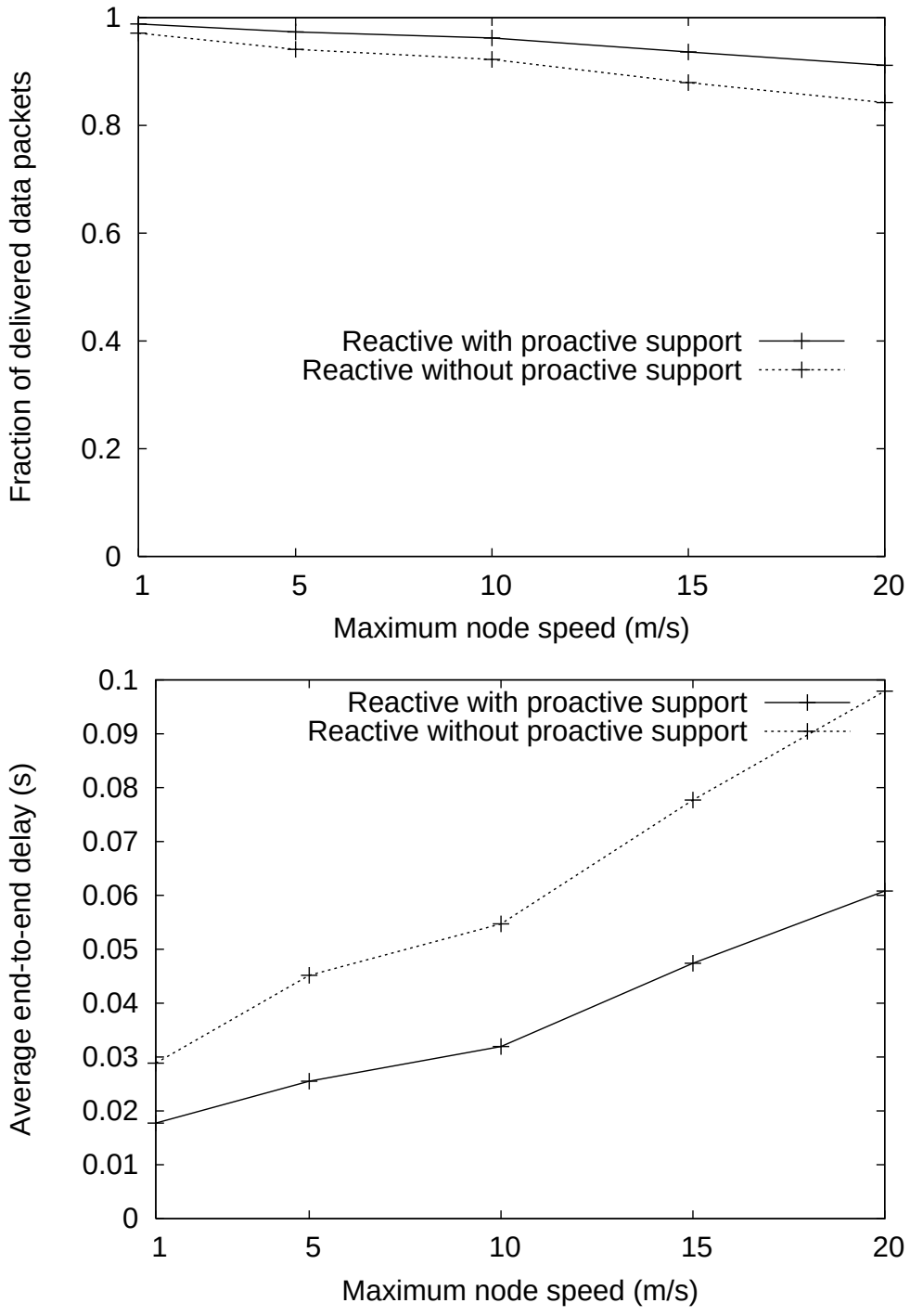


Figure 5: The delivery ratio and average end-to-end delay for the full system ("Reactive with proactive support"), and for the system with the proactive part switched off ("Reactive without proactive support"). In these tests, all sessions are sent reactively.

[4] C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(3):257–269, 2003.

[5] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1998.

[6] I. D. Chakeres and E. M. Belding-Royer. The utility of hello messages for determining link connectivity. In *Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Honolulu, HI, USA, October 2002.

[7] I. D. Chakeres and C. E. Perkins. *Dynamic MANET On-demand Routing Protocol*. IETF, February 2008. Internet Draft.

[8] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *Proceedings of IEEE INMIC*, 2001.

[9] F. Ducatelle, G. Di Caro, and L. M. Gambardella. Using ant agents to combine reactive and proactive strategies for routing in mobile ad hoc networks. *International Journal of Computational Intelligence and Applications (IJCIA)*, 5(2), 2005.

[10] Z. J. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of the IEEE International Conference on Universal Personal Communications*, 1997.

[11] J. Hoebeke, I. Moerman, B. Dhoedt, and P. Demeester. Adaptive multi-mode routing in mobile ad hoc networks. In *Proceedings of the 9th International Conference on Personal Wireless Communications (PWC)*, 2004.

[12] D. B. Johnson and D. A. Maltz. *Mobile Computing*, chapter Dynamic Source Routing in Ad Hoc Wireless Networks. Kluwer, 1996.

[13] C. Mbarushimana and A. Shahrabi. Comparative study of reactive and proactive routing protocols performance in mobile ad hoc networks. In *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW)*, 2007.

[14] C. Na, J. K. Chen, and T. S. Rappaport. Measured traffic statistics and throughput of IEEE 802.11b public WLAN hotspots with three different applications. *IEEE Transactions on Wireless Communications*, 5(11), November 2006.

[15] S. Nanda, Z. Jiang, and D. Kotz. A combined routing method for wireless ad hoc networks. Technical Report TR2007-588, Dartmouth College, 2007. Submitted to the 17th International Conference on Computer Communications and Networks (ICCCN '08).

[16] K. Nichols, S. Blake, F. Baker, and D. Black. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*. IETF, December 1998. RFC 2474.

[17] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[18] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[19] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer. Sharp: A hybrid adaptive routing protocol for mobile ad hoc networks. In *Proceedings of The Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[20] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 1999.

[21] C. Santivanez and I. Stavrakakis. A framework for a multimode routing protocol for manet networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, September 1999.

[22] Scalable Network Technologies, Inc. *QualNet Simulator, Version 4.0*, 2006. Available from: `http://www.scalable-networks.com`.

[23] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proceedings of IEEE INFOCOM*, 2003.