Mobile Netw Appl (2012) 17:281–297
DOI 10.1007/s11036-011-0335-1

Energy- and Delay-Efficient Routing in Mobile Ad Hoc Networks

Nicola Costagliola · Pedro Garçia López · Francesco Oliviero · Simon Pietro Romano

Published online: 23 July 2011

Abstract In this paper we discuss how we improved the *MChannel* group communication middleware for Mobile Ad-hoc Networks (MANETs) in order to let it become both delay- and energy-aware. MChannel makes use of the Optimized Link State Routing (OLSR) protocol, which is natively based on a simple hop-count metric for the route selection process. Based on such metric, OLSR exploits Dijkstra's algorithm to find optimal paths across the network. We added a new module to MChannel, enabling unicast routing based on two alternative metrics, namely end-to-end delay and overall network lifetime. With such new module, we prove that network lifetime and average end-to-end delay improve, compared to the original OLSR protocol implementation included in the mentioned middleware. Thanks to MChannel's approach, which implements routing in the user's space, the improvements achieved in the unicast jOLSR routing protocol are transparently applied to the upstanding MChannel overlay multicast OMOLSR protocol. We also discuss how the proposed new module actually represents a general framework which can be used by programmers to introduce in MChannel novel metrics and path selection algorithms.

Keywords green networking · mobile Ad-hoc Networks · delay-and energy-aware routing · context-aware optimization

1 Introduction

Mobile Ad hoc Networks (MANETs) are the subject of many research works in the field of computer networks. Such an interest is due to the fact that they enable the creation of infrastructure-less wireless networks, e. g. in environments in which the infrastructure itself is either unavailable from the beginning or damaged because of calamities. On the other side of the coin, ad hoc networks introduce new issues, like auto-configuration and auto-adaptation to changes, while emphasizing well-known problems of other kinds of wireless networks, like dynamism, communication interferences and limitation of resources. Other open challenges comprise scalability, minimum consumption of power resources, Quality of Service (QoS) and security provisioning [1]. To allow these networks to work, the different layers of the IP stack and the way in which they exchange information between each other, have to be modified. With regard to the latter point, it has been recognized that cross layer optimization represents a mechanism which can bring several benefits in terms of performance [2]. Cross-layer optimization is an approach consisting in breaking the inter-layer communication approach of

N. Costagliola · F. Oliviero · S. P. Romano (✉)
Computer Science Department, University of Napoli "Federico II", Via Claudio 21, 80125 Napoli, Italy
e-mail: spromano@unina.it

N. Costagliola
e-mail: n.costagliola@studenti.unina.it

F. Oliviero
e-mail: folivier@unina.it

P. G. López
Department of Computer Engineering and Maths, Universitat Rovira i Virgili, Av. Paisos Catalans 26, 43007 Tarragona, Spain
e-mail: pedro.garcia@urv.cat

the Open System Interconnection (OSI) model, and allowing protocols at the different levels to share information. The strict layering approach of the OSI model allows information exchanging just between two adjacent levels. Most researchers recognize that strict layering enables controlled interaction among layers, because each of them is developed and maintained independently. On the other hand, cross-layer solutions generate highly coupled code that is impossible to maintain effectively. A cross-layer solution enabling user-space applications to access network layer (and typically kernel space) information, can definitely improve the performance of a MANET. Though, with such approach we lose the aforementioned advantages of strict layering, to the detriment of applications portability. MChannel [3] is a group communication middleware for ad hoc networks which overcomes this obstacle by moving the routing protocol to the user space. With such approach, middleware and applications can benefit from the knowledge of the underlying network topology, at the same time avoiding cross-layer interactions with the network layer. The aim of this paper is to improve the routing protocol implemented in MChannel, in order to consider both battery level of the nodes and delay of the links in the path selection process. MChannel makes use of the Optimized Link State Routing (OLSR) protocol, currently based on a simple hop-count metric and relying on an implementation of the Dijkstra's algorithm to find the best path based on such metric. The mentioned approach is not sufficient in case we want to take into account specific performance parameters which are specific to a MANET. In this work we discuss how we added a new module to the MChannel middleware, enabling unicast routing based on two alternative metrics, namely end-to-end delay and overall energy efficiency of the MANET. We show that network lifetime and average end-to-end delay improve, compared to the OLSR protocol implementation natively included in MChannel. Thanks to MChannel's user-space routing, all the routing improvements achieved in the unicast MANET protocol are transparently applied to the overlay multicast protocol. We also discuss how the new module can be considered as a general framework made available to programmers willing to experiment with new metrics and path selection algorithms. The paper is organized as follows. Section 2 provides information about the paper's rationale and motivation, by presenting a brief survey on MANET routing protocols, as well as discussing the MChannel approach as opposed to cross-layer optimization. Section 3 is devoted to delay- and energy-efficient routing and presents state-of-the-art proposals which can be found in the literature, with special regard to variants of the OLSR protocol. The same sections analyzes multi-objective routing, in which the routing protocol looks for paths which jointly improve packet latency and network lifetime. Section 4 presents our proposal for a new routing module based on both energy and delay metrics, which we integrated in the MChannel middleware. Experimental results are presented in Section 5, which shows the performance improvements achievable thanks to the exploitation of the mentioned new metrics. The same section also introduces a couple of simulation scenarios in which a balance has to be struck between minimizing delay and maximizing the network's lifetime. The work ends with Section 6, which provides conclusions and highlights some directions of future work.

2 Background and motivation

Mobile ad hoc networks are built on top of the IP stack of traditional networks. Though, because of their peculiarities, each level of such stack has to be properly modified in order to optimize the performance of a MANET. This certainly holds for the network layer, and to routing in particular, whose criticality has since long attracted the attention of many researchers. Among the numerous proposals which can be found in the literature, cross-layer optimization has gained significant momentum [4]. Such approach brings in a critical switch of perspective in the networking communication paradigm, since it envisages that information held by a certain layer of the IP stack can be shared with another layer, even if not adjacent. Many cross-layer solutions have been actually characterized by an interaction between the application layer and the network layer. As an example, the direct knowledge of the underlying network topology has allowed applications to become location-aware (or even more generally, *context-aware*). On the other side of the coin, it stands clear that cross-layer interactions violate the general principle of separation of concerns, which is the key success factor of the strict layering approach and provides it with desirable functionality like the intrinsic ease of maintenance. Based on these assumptions, the work presented in [3] tries to take the best out of the two mentioned approaches, by leveraging network-layer information at the application level, at the same time avoiding to violate the principle of separation of concerns. Similarly to cross-layer approaches, in fact, the MChannel middleware enables an application to be aware of the network topology. Though, unlike cross-layer solutions, MChannel does not interact with kernel components running at layer three of the stack,

since it moves network layer functionality, namely the routing protocol, to the user space. The implemented network layer includes the Optimized Link State Routing (OLSR) protocol, which enables each node in a MANET to build the topology of the whole network. In this section we briefly recall routing approaches in MANETs. Then, we delve into the most relevant details of the MChannel solution, which paves the ground to the main contribution of the paper concerning the introduction of energy- and delay-aware criteria in MChannel's routing process.

2.1 Routing in MANETs

At network level, the routing protocol has to guarantee that a node can be reached from any other node in the network. This objective is difficult to achieve because of the presence of both wireless links and mobile nodes, which call for dynamic reconfiguration of the the routing strategy as soon as network connectivity changes. The classical link-state and distance vector routing protocols are not suitable in such case, since they have not been designed for mobile devices with limited resources and which communicate through wireless links. For this reason, a number of routing protocols specifically devised for MANETs have been proposed in the last years, and some of them have been standardized by the Internet Engineering Task Force (IETF). Such protocols can be roughly classified in three categories [5]: *proactive*, *reactive* and *hybrid*.

With proactive protocols (also named table-based) each node maintains information enabling it to decide how to route messages towards any other node in the network. Such information is usually stored in a certain number of tables (updated over time) providing each node with a view of the network topology. Differences among these protocols reside in the way the topology information is detected and updated, as well as in the type of information that is stored in each such table. Protocols falling in this category do not work efficiently when the topology changes quickly and the number of nodes is high. In fact, network changes require time to be spread among the nodes, and the amount of information to store and update grows linearly with the size of the network. On the other hand, proactive protocols guarantee to find the forwarding path in a very short time, because all the necessary information is already available when data have to be transmitted. Moreover, they allow to find, in a simple way, a path based on specific QoS requirements. Destination-Sequenced Distance Vector (DSDV) and Optimized Link State Routing (OLSR) are probably the most well known proactive protocols available nowadays.

With reactive protocols (also named *on demand* protocols) a path discovery process is started from a source which wants to transmit a packet towards a specific destination. The name 'on demand' is due to the fact that the search of a suitable forwarding path takes place only when data transmission is needed. Once a node determines a route, it will maintain this route for the entire duration of the transmission. In the discovery process of a route towards a destination, the source sends route request messages through flooding. Nodes which know how to reach the required destination, send back route reply messages; this message exchange phase goes on until the entire route is defined. The basic principle of reactive protocols enables a smaller overhead, because nodes only maintain information about active routes, instead of keeping in memory an updated view of the overall network. For this reason, they are suitable for highly dynamic networks. Their major drawback clearly resides in the transmission delay incurred when new data have to be transmitted. Ad-hoc On-demand Distance Vector routing (AODV) and Dynamic Source Routing (DSR) currently represent the most widely spread reactive protocols available for MANETs.

Hybrid protocols try to jointly take advantage of both the proactive and the reactive approach. Close-by nodes communicate with each other proactively, while nodes which are deemed to be too far away from the source are reached through routes discovered using a reactive approach. To give an idea of this hybrid approach, we briefly introduce the so-called Zone Routing Protocol (ZRP). ZRP defines, for each node, a zone that is the set of nodes which are r hops away from it, where r (radius) is a parameter of the protocol. A node computes its routing table by means of a proactive protocol, if the destination node belongs to its zone; a reactive protocol is otherwise exploited. In the latter case, when a node has to send data to a destination node external to the zone to which it belongs, it sends a route request to its peripheral nodes (nodes reachable in exactly r hops). If these latter nodes belong to a zone including the destination, they will route the message to it, otherwise to their peripheral nodes. The process iterates until the destination is discovered. Protocols like ZRP reduce both the typical delays of the reactive protocols and the communication overhead introduced by the proactive protocols. Though, fine tuning of the radius r becomes critical in such case.

2.2 The MChannel approach

MChannel [3] is a middleware designed to support collaborative applications for MANETs. The basic idea

of this middleware is to make applications aware of the underlying topology, by moving the network protocol to the user space. MChannel is built on top of a modified version of JGroups.[1] JGroups is a set of software libraries allowing an easy development of reliable multicast applications. It basically consists of a channel, an abstraction of a multicast communication channel, and a stack of protocols which carry out actual communication through the network. MChannel modifies the JGroups channel and protocol stack, in order to make them usable from mobile ad hoc network applications. Two new protocols have been added to it, namely jOLSR and OMOLSR. The former is an implementation of the OLSR unicast protocol; the latter is a new multicast protocol which uses the underlying jOLSR protocol to improve group communication efficiency.

In addition to jOLSR and OMOLSR, there are two further protocols (without considering UDP). One is an adaptation of the JGroups Flow Control (FC) protocol, which offers flow control thanks to a credit system based on the knowledge of the topology obtained by jOLSR. Notice that flow control is not provided by the native UDP transport-layer protocol, usually used both in multicast applications and in MANETs. The other protocol (Reliability) looks after reliable communication, by means of an acknowledgment scheme. Finally, the MChannel API is a Java Interface which enables the transmission of messages either to a specific multicast group or to one of its members. N instances of MChannel can be created to communicate with N groups.

The contribution of this paper stems from the mentioned work done with MChannel and resides in the introduction of extended QoS metrics in the path selection mechanism of the OLSR protocol. In particular, we herein focus on two metrics to be alternatively chosen for the routing table computation phase, namely *end-to-end delay* and *network lifetime*. The latter is the time that can still elapse before exhaustion of the nodes' battery. At the moment, only a simple hop-count metric is considered in MChannel. Starting from the graph provided by OLSR, the Dijkstra's algorithm computes the shortest path by solving a Shortest Path Problem (SPP) characterized by all equal arc costs. In the routing table, the next-hop to reach a specific destination will be the downstream node along the computed shortest path towards it. In the case of the delay metric, the problem to solve is again the SPP, but this time each arc cost will depend on the delay experienced by packets while crossing the arc itself. On the other hand, in the case of the energy metric, the problem becomes a Max-Min Problem (MMP) and costs are associated with the nodes instead of the arcs. In both cases, two new algorithms are used for the resolution phase. We will show in the paper how the solutions we devised can be easily generalized. What we realized is indeed a quite general framework for the MChannel middleware, enabling the formulation (and subsequent validation through actual experimentation) of new optimization problems for the routing table computation, based on predefined QoS metrics dependent on ad-hoc computed weights associated with either nodes or arcs. The presented implementations of routing strategies aimed at optimizing end-to-end delay or network lifetime actually represent two specific use cases of such general framework. We will show in the paper that thanks to the application of the novel routing strategies proposed, average end-to-end delay and network lifetime improve compared to the basic jOLSR implementation.

[1] http://www.jgroups.org/

3 Energy- and delay-efficient routing for MANETs

One of the most challenging issues related to MANETs is by no doubt represented by QoS support [6]. Two important QoS parameters for a MANET are network lifetime and average end-to-end delay. With regard to network lifetime, bear in mind that most nodes in a MANET are battery powered. For this reason, it is important to maximize the time before the nodes fail because of battery exhaustion, in order to avoid network partitioning and consequent loss of communication. With regard to end-to-delay, many applications (e.g. real-time) are characterized by time-sensitive data. Routing table computation performed by a routing protocol based on energy or delay measures can obviously improve the two aforementioned parameters. In this regard, a great amount of energy- and delay-aware routing protocols for ad hoc networks have been proposed in the last years [7–15, 21, 22].

In this section we present some of the latter works, by analyzing their main features and drawbacks. Since this work is specifically concerned with the OLSR protocol, we cover in more detail extensions of this protocol designed to optimize either delay or nodes lifetime.

3.1 Energy-aware routing protocols

Power consumption in MANETs is an important issue because all or most of the nodes are battery supplied, and the communication infrastructure is composed of the same nodes which are using it. In such context,

optimizing energy consumption also means maximizing the overall usability of the network. Power is required for both processing (e.g. protocols operations and applications execution) and communication (e.g. control and data messages transmission). To reduce the amount of required power, we can adopt techniques at the several layers of the protocols stack, paying attention to the fact that protocol layers are closely coupled from the power consumption perspective [7]. Research works about these techniques involve, in particular, physical, data link and network layers. At the physical layer, a mechanism for the auto-adjustment of transmission power is an example [8]. At the network level, since a significant amount of energy is spent by a node to transmit packets, the routing algorithm's path selection criteria can affect a MANET's lifetime. The most relevant energy-aware routing protocols proposed in the literature can be distinguished by the number of paths used for the transmissions: there are multi-path and single-path energy-aware protocols. The solutions are further classified based on the objective to achieve. They can try to: (i) minimize the total power needed to transmit packets; (ii) maximize the lifetime of every single node; (iii) minimize the total power needed to transmit packets at the same time maximizing the lifetime of every single node. Some interesting energy-efficient route selection schemes, falling in one of the previous categories, are presented in [7] and briefly described in the following.

Minimum Total Transmission Power Routing (MTPR) is a routing protocol aimed at minimizing overall power consumption in MANETs. Given a source s and a destination d, we denote with P_r the total transmission power for a generic route r from s to d. P_r is the sum of the power consumed for the transmission between each pair of adjacent nodes belonging to r. MTPR selects the route r^* such that $r^* = \min_{r \in R} P_r$, where R is the set containing all possible routes from s to d. A simple shortest path algorithm can be used to find this route. A drawback of this schema is that a route with a great number of hops can be selected, with a consequent increase in both delay and path instability (the latter, due to the dynamic nature of the MANETs). A more significant drawback is that, while the total transmission power is reduced, residual energy of every node is not considered and the nodes can fail quickly.

Minimum Battery Cost Routing (MBCR) associates each node n_i in the network with a weight $f_i(c_i(t)) = 1/c_i(t)$, where $c_i(t)$ is the battery capacity level of n_i at time t. Given a source s and a destination d, if we say E_r the sum of the nodes weights of a generic route r from s to d, MBCR selects the route r^* such that $r^* = \min_{r \in R} E_r$, where R is the set containing all possible routes from s to d. Such a scheme will always choose routes with maximum total residual energy. Nevertheless, this metric does not consider the residual energy of a single node. For instance, if a route includes a node characterized by a very low energy together with others with high energy, such route might be chosen. Indeed, in this case it would be better to choose a path in which all the nodes have comparable energy levels, even though not so high.

With *Min-Max Battery Cost Routing* (MMBCR), starting from the above definition of $f_i(c_i(t))$, for each route r from a source s to a destination d, a cost is defined as $C_r(t) = \max_{i \in r} f_i(c_i(t))$. The chosen route r^* verifies the relation $C_{r*}(t) = \min_{r \in R} C_r(t)$. MMBCR safeguards nodes with low energy level because it selects the route in which the node with minimum energy has more energy, compared to the nodes with minimum energies of the other routes. Nevertheless, it does not take into account explicitly the transmission power consumption, hence resulting in a possible reduction of the overall network lifetime.

Conditional Max-Min Battery Capacity Routing (CMMBCR) proposes an approach based on both MTPR and MMBCR. Let us consider the node of a generic route r from a source s to a destination d, with lowest energy. Let also $m_r(t)$ be its energy, and R the set of all the routes from s to d. If some paths with $m_r(t)$ over a specific threshold exist in R, one of these will be chosen using the MTPR scheme. Otherwise, the route r^* satisfying the relation $m_{r*}(t) = \max_{r \in R} m_r(t)$ will be selected. This scheme suffers from an unfair increment of the forwarding traffic towards nodes with more energy [9].

Minimum Drain Rate (MDR [10]) proposes a mechanism which takes into account node energy dissipation rate, thus avoiding the above problem. MDR defines for each node n_i a weight $C_i = RBP_i/DR_i$, where RBP_i is the residual battery power and DR_i the drain rate of n_i. Intuitively, DR_i represents the consumed energy per second in a specified time interval. To confer more precision to this energy dissipation rate estimation, this parameter is computed by each node every T seconds as: $DR_i = DR_{i_{\text{old}}} + (1-\alpha) DR_{i_{\text{sample}}}$. $DR_{i_{\text{old}}}$ is the previous computed value of DR_i and $DR_{i_{\text{sample}}}$ the new one. The parameter α reflects the relative importance to be given to the past with respect to the current values of DR_i. Obviously, T has to be tuned appropriately in order to avoid frequent updates. Now, let C_r be the minimum weight of a generic route r from a source s to a destination d. MDR selects the route r^* such that $C_{r*} = \max_{r \in R} C_r$. In this way, residual energy level, as well as the energy consumption rate due to the

incoming traffic to be forwarded, are jointly taken into account.

3.2 Delay-aware routing protocols

A routing strategy which allows to reduce the average end-to-end delay is desirable for all real-time applications (e.g. video/audio streaming, multi-party games, real-time control systems), whose utilization is becoming widely spread, even in wireless scenarios. End-to-end delay is the time elapsing from message generation, at the source, until message reception, at the destination. It is composed of the delays experienced between each pair of adjacent nodes along the path from source to destination and actually comprises several different contributions: processing delay (at each stack layer), queuing delay, transmission delay and propagation delay. In the literature, there are not so many routing protocols designed to optimize the end-to-end delay, compared to energy-aware routing protocols like those mentioned in the previous section. In the following, we introduce some of them, by just focusing on proposed extensions to reactive protocols. A separate section is then devoted to the proposed extensions to OLSR, which is of special interest for us, since the solution we propose is built on top of it.

Seminal work on QoS-enabled routing in MANETs can be found in [11]. The *Quality of Service for Ad hoc On-Demand Distance Vector Routing* protocol is an extension of AODV specifically conceived for QoS. Authors of [12] have proposed the *Split Multipath Routing* (SMR) protocol. With such protocol, the source node floods an RREQ message in order to find a route to the destination node. All the nodes involved in the discovery process insert in the RREQ the address of the node from which they have received the message (like in DSR). Since these nodes do not discard the duplicated RREQ, the destination node will learn several alternative routes, so it can select multiple disjoint routes and send an RREP to the source node along them. In the mentioned work, the authors have decided to consider the case in which the destination node selects only two disjoint paths. The first chosen path is the one discovered first, i.e. the path with the potentially shortest end-to-end delay; the second path is the path maximally disjoint from the first one. Once the first route is available for the source, data transmission via this route will start. When the second route is also known, both paths will be used by using a per-packet allocation algorithm. Clearly, the usage of two (or more) paths can reduce the overhead caused by the discovery process of a new route, in case of route failures.

3.3 Power and delay in the OLSR protocol

Several works have focused on extending the OLSR protocol in order to offer QoS guarantees. This is mainly due to the fact that OLSR is a link state protocol, and at the same time it introduces a limited overhead thanks to a controlled flooding. Link state protocols have the advantage of offering a complete view of the network topology, and potentially of its quality level. In fact, the control messages used to carry topology information might also carry information like links stability, links delays, nodes energy level, etc.. A recent work about the introduction of QoS features in OLSR is described in [13]. The same work also provides a critical overview of the most well known extensions of OLSR for QoS provisioning. In the following, we focus on the mechanisms devised to take into account in OLSR two QoS parameters, namely energy and delay.

Quality of Service-OLSR (QOLSR) represents one of the first projects aimed to provide OLSR with QoS capabilities. In QOLSR the *HELLO* and *TC* messages carry, for each advertised link, a set of related QoS metrics (by default, bandwidth and point-to-point delay estimations made by each node). In this way, each node can build a weighted graph where link weights reflect the QoS metrics associated with each link. Based on this graph, different strategies can be used for path computation. In [14], the point-to-point delay between two neighbors is computed as follows. The *HELLO* message contains its creation time. When a node receives it, it subtracts the creation time from the current time in order to obtain a delay estimation. To take into account also the previous estimations, the actual point-to-point delay is computed as: $averageDelay = averageDelay + (1 - \alpha) \cdot measuredDelay$. This simple mechanism assumes that a synchronized network is available, so that all the nodes can refer to a global clock. The route from source s to destination d is selected by a modified version of the Dijkstra's algorithm, which finds the path with maximum bandwidth (where the bandwidth of a path is equal to the minimum bandwidth of a link in the path). If the algorithm determines more paths with maximum bandwidth, the path with minimum delay is selected. In [15], the mechanism for evaluating delay becomes more complex. The end-to-end delay referred to a single hop is expressed as a function of parameters specific of the IEEE 802.11 Medium Access Control (MAC) protocol, combined with interference measures.

The *Constrained Minimum Drain Rate* (CMDR) [9] protocol computes routing paths which reduce energy consumption, thus increasing the nodes lifetime. Let us

consider the node of a generic route r from a source s to a destination d, with lowest energy. Let $m_r(t)$ be its energy, and R the set of all the routes from s to d. If some paths with $m_r(t)$ over a threshold exist in R, a route among them will be chosen using the Maximum Transmission Power Routing (MTPR) scheme we already mentioned in the paper. Otherwise, CMDR selects the route r^* by means of the Minimum Drain Rate scheme, also mentioned above. With this approach, both energy consumption and residual energy of the nodes are considered when choosing the next-hop. In fact, as far as the nodes in the network have sufficient energy, we can optimize the power consumption caused by the transmission of a message. Instead, when most of the nodes in the several paths towards the destination have a low battery level, MDR optimizes their energy expenditure.

The work in [13] has been specifically conceived to improve the performance of voice applications. Network information is associated with all the network links and consists of estimations of point-to-point delay, delivery probability and data-rate, computed between two adjacent nodes. With regard to point-to-point delay, this is estimated by means of a Global Positioning System (GPS) module (which offers a sort of a global clock to all the nodes in the network), in conjunction with send-time information included in *HELLO* messages. In practice, this approach is similar to the one described in [14], with the difference that in this case a GPS module is used for nodes synchronization. Each node computes the three aforementioned link parameters for all its outgoing/incoming links, and it spreads such information across the network through the OLSR control messages.

Minet et al. [16] propose some alternative solutions to make OLSR energy-aware with respect to the routing table computation. In one of them, the route selection algorithm chooses as next-hop the node belonging to the path which causes the lowest energy consumption. To identify such node, the Dijkstra's algorithm is applied to the network graph of the forwarder. All the links $i \rightarrow j$ in this graph have a cost equal to the energy spent to transmit a packet across them. This energy is given by: $e_{ij} = e_{\text{trans}} + n \cdot e_{\text{recv}}$, where e_{trans} and e_{recv} represent the energy for a single transmission and reception respectively, and n is the number of non-sleeping nodes placed in the interference zone of the transmitter i. A different solution uses the multipath source routing: the source determines two different paths, switching from one to the other for each packet to transmit. The chosen path is also included in the data packet, so that each forwarder selects the next hop according to the path computed by the source. The source computes the paths in two different ways:

1. one path by Dijkstra's algorithm, in order to minimize the energy consumption resultant from packet transmission (as with the first mechanism described); the other path is determined in the same way, after deleting all the links composing the previous path;
2. one path is computed by Dijkstra's algorithm, as above; the other path is determined in the same way, after deleting all the nodes belonging to the previously selected path.

Simulations show that the solution which adopts a single path source routing performs the best, both with regard to the network lifetime and with regard to the percentage of user data delivered.

3.4 Multi-objective routing in Ad Hoc Networks

An original way to solve the path selection issue, at the same time offering QoS guarantees, is to consider it as a multi-objective (MO) optimization problem. Given a source and a destination, we move the focus from the research of the path that minimizes a scalar function (e.g. the number of hops) to the research of the path that optimizes a vectorial function. Each component of this function represents a performance facet, like throughput, delay, energy, robustness, etc. The different objectives take into account the several application requirements. The characteristics of the environment are included both in the expression of the objective function and in the mathematical relations representing the constraints of the optimization problem. In this kind of problems we have to find a Pareto solution, in the so-called Pareto set. For this purpose, a lot of techniques exist, some of which are exact while others based on heuristics. Since MO modeling is usually applied to complex systems, heuristics are preferred. Among them, Evolutionary Algorithms (EA), and in particular Genetic Algorithms (GA) play a major role. Several works have proposed MO approaches to routing optimization in MANETs. In [17], a probabilistic network model comprising link and node probabilities is defined. The optimization problem's variables are the throughputs of the network nodes, on which the value of a node presence probability depends. The latter value is used to discriminate whether or not a node belongs to the routing path. A Pareto solution is found through exhaustive search because of the limited size of the network used for the simulations, but authors propose also meta-heuristic algorithms for

the model they have created. The parameters of the model are related to the different layers of the protocol stack (e.g. path loss attenuation factor for the physical layer) and they characterize both the network and the communication pattern. Delay, robustness and energy are the performance figures to be optimized. The authors of [18] have proposed to solve the route selection problem by means of the so called *Evolutionary Ad hoc On-demand Fuzzy Routing* (E-AOFR). Such module has been implemented on top of a reactive routing protocol, but it can also be adapted to a proactive protocol. The cost of a route is a function of four metrics: (i) remaining battery capacity at a node; (ii) buffer length at an intermediate node; (iii) link stability between two intermediate nodes along the route; (iv) number of intermediate hops along the route. Each node, during path discovery, computes a part of this cost by using fuzzy logic, and the destination node selects the path with minimum cost. The selected route should minimize the end-to-end delay, while maximizing both packet delivery and lifetime of the batteries.

MO analysis is widely used to improve the performance of MANETs, and not only for the computation of the best route. For example, in [19] a methodological approach to the selection of the parameters of a generic routing protocol is proposed. The set of parameters is seen as a solution of a MO problem (whose objectives are the minimization of both dropped packets and transmission delay) which is solved through a genetic algorithm. In the following, we mention some works which do not consider explicitly the MO analysis, but nonetheless define a routing mechanism which finds paths considering more metrics at a time. With regard to delay and energy guarantees two distinct approaches can be found in the literature: (i) routing based on delay and energy constraints (typically through extensions to reactive protocols [21, 22]); (ii) routing based on the discovery of paths which jointly optimize average end-to-end delay and energy depletion. An interesting example of the latter approach is represented by the *Optimized Energy-Delay Routing Protocol* (OEDR) [20], which is a variant of OLSR considering energy and delay both in the MPRs selection and the routing table computation phases. Every time a node receives a *HELLO* message, it estimates end-end-delay and transmission energy towards all its neighbors. Since the way to estimate the delay is similar to [14], OEDR assumes the presence of a synchronized network. Thanks to *HELLO* and *TC* messages, this information is spread through the network so that all nodes can build a weighted graph with link costs equal to the mathematical product between energy and point-to-point delay. Unlike OLSR, OEDR determines a minimum spanning tree for routing packets, instead of a shortest path for each destination.

4 Introducing energy- and delay-efficient routing in MChannel

After analyzing both the background and the related work, we eventually arrive at the core contribution of this paper. We herein propose a new module we designed and implemented in the MChannel middleware so as to enable it to perform either delay-aware or energy-aware routing. The former case requires estimations of point-to-point delays, considered as costs associated with network links; the latter, calls for adequate estimations of nodes' energy levels, which are used as costs associated with network nodes. Such costs represent the input of the algorithm which computes the routing table. In this section, we first define the requirements of the new module; then, we describe the design and the implementation of our solution.

4.1 Problem statement

The routing algorithm's goal is to define a next-hop for data transmission from a source node s to a specific destination node d, by finding a path which meets a requirement related to a certain metric. In the simplest case, the number of hops of the path has to be minimum between s and d; the native OLSR is an example. In general, metrics different from the hop-count or also more QoS metrics at the same time, can be considered. We can formulate the routing problem in this way. Let P_{sd} be the set of all the possible paths between s and d. The path p^* to choose, has to optimize a vector function of the following type:

$$\overrightarrow{z}(p) = (z_1(p), z_2(p), \ldots, z_n(p)) \tag{1}$$

where p is a generic element of P_{sd} and $\{z_i(p)\}_{i=1,\ldots,n}$, the set of considered metrics. In practice, $\{z_i(p)\}_{i=1,\ldots,n}$ are cost functions which, for a given path, return a real number. Equation 1 defines the objective function of a multi-objective problem. The solution we usually have to find for this kind of problems is a Pareto solution. For this purpose, a lot of techniques exist, some of which are exact while others heuristic. In most of the cases, heuristics are preferred because they find a *good enough* solution in a reasonable time.

The first issue we had to face was the design and implementation in MChannel of a general framework enabling programmers to implement a specific path selection strategy based on the routing problem defined above. In other words, jOLSR should be able to route

requests based also on metrics different from the simple hop-count. The values $\{z_i\}_{i=1,\ldots,n}$ to be associated with a generic path p, depend on a set of weights (or costs) related to both the nodes and the arcs belonging to p. The weights to consider, the function in Eq. 1 and the resolution algorithm employed are all decided and implemented by the programmer.

Once done with this phase, we also focused on the design of an example of use of the framework. We chose to consider two different weights, associated, respectively, with arcs and nodes. Arc weights depend on the point-to-point packet latency of a link; node weights represent values depending on each node's residual energy. An application programmer can choose a path selection strategy (in jOLSR), which either minimizes the average end-to-end delay or maximizes the overall network lifetime. To the purpose, we consider two different kinds of functions, named z_e and z_d, respectively.

Let us consider the following notations, in addition to those used above:

- e_i: node's weight, depending on the node's battery level;
- e_p: weight of a generic path $p \in P_{sd}$, depending on e_i;
- d_{ij}: arc's weight, depending on the point-to-point packet latency;
- d_p: weight of a generic path $p \in P_{sd}$, depending on d_{ij}.

The functions z_e and z_d to be optimized are:

$$z_e(p) = e_p \tag{2}$$

$$z_d(p) = d_p \tag{3}$$

This second problem requires to: (i) define the above costs and find a way to compute them; (ii) define the relations in Eqs. 2 and 3; (iii) choose two suitable algorithms to solve the corresponding problems; (iv) implement the cost computation mechanisms, as well as the identified algorithms in MChannel, by exploiting the novel framework.

4.2 Design

Currently, routing in MChannel is executed by jOLSR. The routing table is computed based on the well-known data structures employed in the standard OLSR protocol: Link Set (LS), Topology Set (TS), Neighbor Set (NS) and 2-hop Neighbor Set (NNS). In general, a node updates these tables upon arrival of either a *HELLO* or a *TC* message (which indicate changes concerning either the network topology or the status of a specified link). As soon as a change occurs in one of the aforementioned tables, the routing table is recomputed by means of a route computation process based on Dijkstra's algorithm.

We have introduced in jOLSR a new table called *QoS Costs Set* (QCS) and we have changed the format of the above mentioned control messages, as well as their respective elaboration. The QCS table stores the costs associated with both links and nodes. The HELLO message has been modified in order to let it carry, for each neighbor it includes, the estimated costs related to the link between the advertised neighbor and the message sender, as well as the node cost of the advertised neighbor. The TC message includes the same additional information, but this time referred to the advertised destinations.

In Fig. 1 we depict the structure of the QCS for a generic node n (in this case we report a single cost for each node and a single cost for each link).

When a node n receives a *HELLO* message, it determines the costs of each link towards all its neighbors. Such costs are then inserted in the *HELLO* messages

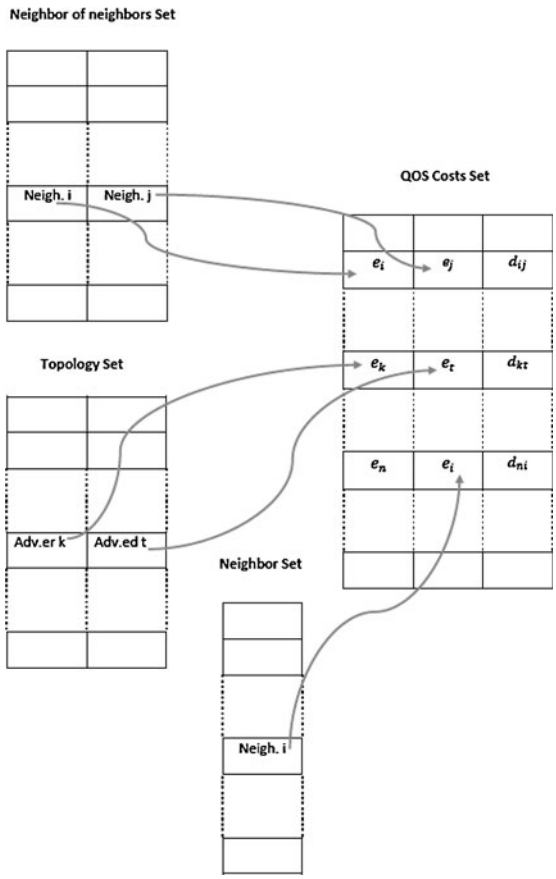

Fig. 1 The QoS costs set table

sent by node n. In this way, each neighbor of n can learn the costs of the links between n and its neighbors (the so-called 2-hop neighbors). Since the advertised nodes in a *TC* message are neighbors (MPR Selector Set), such information can be also spread in the network, so that each node knows the costs of all the links. The node costs are computed by each node and spread through the network together with the node addresses, included in the *TC* and *HELLO* messages. Node and link costs are stored in the QoS Costs Set (QCS) and used to compute the routing table. Notice that the QCS is updated more frequently than the OLSR tables, because the costs change more frequently than the topology. Since a modification of one of the tables causes the re-computation of the routing table, we can adopt two solutions to reduce the overhead. One is to define a re-computation period N in terms of number of QCS table changes. Alternatively, the re-computation can be done when the differences between old costs, stored in the QCS, and new costs are relevant, e.g. over a certain threshold T. In these cases, N or T represent parameters that have to be opportunely tuned.

4.2.1 Energy-aware routing in jOLSR

To enable jOLSR to execute an energy-aware routing, the cost to assign to the nodes (e_i) is computed as follows [10]: $e_i = RBP_i/DR_i$, where RBP_i is the residual battery energy and DR_i the drain rate of n_i. DR_i is evaluated every T seconds as an average of both new and past values: $DR_i = DR_{i_{\text{old}}} + (1-\alpha)DR_{i_{\text{sample}}}$, where $DR_{i_{\text{old}}}$ is the previous computed value of DR_i, and $DR_{i_{sample}}$ the new one.

If e_p is the minimum node's weight of a generic path $p \in P_{sd}$, we have to find a path p^* such that:

$$z_e(p^*) = \max_{p \in P_{sd}} e_p \tag{4}$$

Thanks to this mechanism, a good compromise between preserving the nodes with a lower energy and fairly exploiting the other nodes, is achieved. Equation 4 defines a max-min problem, which means that the energy-based metric is concave [14]. To solve such problem, we have chosen a variant of the well known Bellman-Ford algorithm, originally proposed in [23].

4.2.2 Delay-aware routing in jOLSR

To enable jOLSR to execute a delay-aware routing, we define as link cost d_{ij} the point-to-point packet transmission latency of the link (i, j). To compute such cost, we have designed a novel technique, which avoids the need of synchronization of the nodes in the network, by smartly exploiting the periodicity of the control messages typical of OLSR. The designed scheme envisages the use of OLSR acknowledgment messages (*ACKs*) and control messages. Here is how it works. We know that a node i sends periodically a *HELLO* to its neighbors. After the reception of N *HELLO* messages, each neighbor j responds with an *ACK*. Node i can estimate the delay of link (i, j) by subtracting the sending time of the N^{th} *HELLO* from the reception time of the *ACK*, hence dividing the result by 2. N is a parameter to properly choose (a typical value of N might be, for example, 4). Once a node has made such a delay estimation, it spreads this information through the network exactly as described previously for a generic set of link costs. As stated in [13], when we estimate the values of the point-to-point delay on the links, we do not need to obtain their exact values but rather to ensure that they meet the actual ordering relation among the real delays. Given two estimations on the links (i, j) and (i, k), let say them d_{ij} and d_{ik}, and the respective real delays d_{ij}^{real} and d_{ik}^{real}, if $d_{ij}^{real} < d_{ik}^{real}$ then $d_{ij} < d_{ik}$. If $d_p = \sum_{(i,j)\in p} e_{ij}$ is the path delay (sum of the link delays of the path), we have to find a path p^* such that:

$$z_d(p^*) = \min_{p \in P_{sd}} d_p \tag{5}$$

Equation 5 defines a shortest path problem, since the delay metric is additive. To solve this problem, we have chosen an implementation of the Dijkstra's algorithm which is completely different from the implementation of the same algorithm provided by both MChannel and OLSR, since these last ones can be used only when the simple hop-count metric is adopted. Our version can instead be used to compute routes using any type of link costs.

4.3 Implementation

The new module we herein describe, named QoS-routing (QoSR) (see Fig. 2), implements the general framework we designed to provide routing mechanisms based on specific QoS metrics.

It is organized in three packages:

- `urv.olsr.qosrouting.data`, containing all the data structures used by QoSR;
- `urv.olsr.qosrouting.routescomputation`, providing the algorithms used for the computation of the paths fulfilling the given QoS constraints;
- `urv.olsr.qosrouting.util`, containing all the classes required for the computation of the estimations used in the optimized routing.

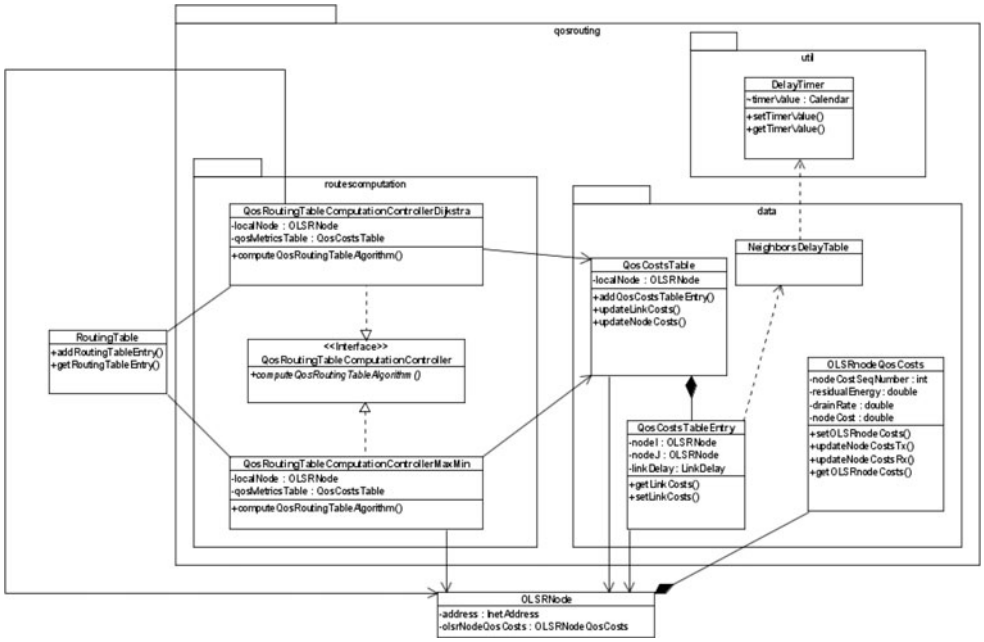

Fig. 2 The QoS-routing module

The class `OLSRnodeQosCosts` represents the set of costs associated with a node. `QosCostsTable` and `QosCostsTable– Entry` implement the QoS Costs Set (QCS). The interface `QosRoutingTableComputation Controller` defines the algorithm used to compute the routing table. With regards to the example of use of this framework, that is the development of the above described routing mechanisms based on energy and delay metrics respectively, the framework has been used as follows. The classes `OLSRnodeQosCosts`, `QosCostsTable`, `QosCostsTableEntry` include a cost for each link and one for each node. `QosRouting–TableComputationControllerDijkstra` and `QosRoutingTableComputationControllerMaxMin` are implementations of the interface `QosRoutingTable–ComputationController` used respectively for delay- and energy-aware routing in jOLSR. Actually, these classes implement generic algorithms usable for the computation of routing tables based on the resolution of max-min and shortest path problems. Additionally, we have developed other classes and modified some existing classes of MChannel. The new classes are `DelayTimer`, which is the timer needed for delay estimations, and `NeighborsDelayTable`, an auxiliary data-structure used for the same purpose. The main classes of MChannel we have modified are `OLSRNode`, which is the representation of a node running OLSR, and the classes `TcMessageHandler` and `HelloMessageHandler`. The modification of the first class has been made because we have implemented the costs of a node as a new attribute of the same class. The second modification was needed to enable handlers to update the QoS Costs Set upon reception of a control message. An application programmer can now choose if the selection path strategy in jOLSR has to optimize the number of hops, the nodes lifetime or the average end-to-end delay, by setting a configuration parameter of MChannel.

5 Trials and experimentations

In this section, in order to qualitatively appreciate the correct behavior of the new module introduced, we first of all provide some trivial examples showing how the optimized routing works. A first example shows how the module set up for an energy-aware routing avoids to select the path in which there is a node with low energy. A second example will show how the module set up for the delay-aware routing chooses the path with minimum end-to-end delay.

5.1 Qualitative evaluations

In the example of Fig. 3, node 1 has to send a data message to node 5. The figure shows the routing table of 1 computed by MChannel compared to the one computed by the version of MChannel making use of the new module. The former defines the next-hop by using the hop-count metric. In this case, all the routes

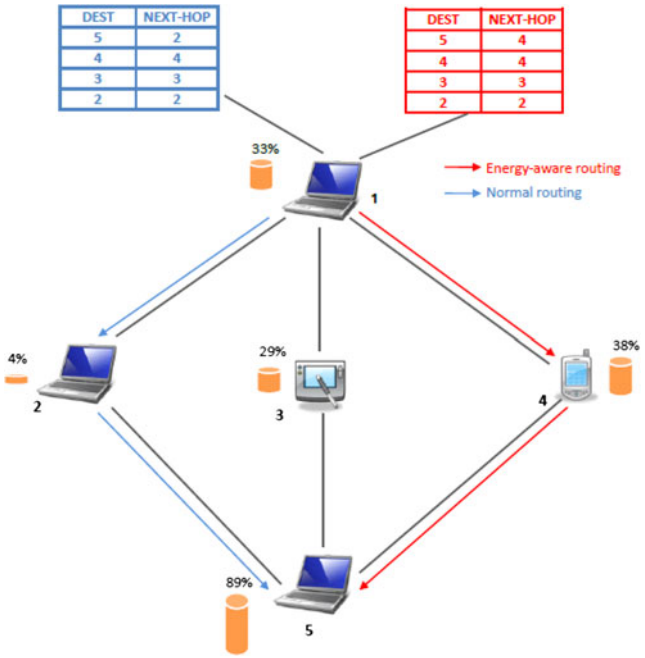

Fig. 3 A simple example of energy-aware routing

to 5 have a cost of two hops, so the choice of the next-hop is random and node 2 is selected. The latter chooses a route in which the minimum energy of a node is maximum, compared with the minimum energy of the other routes. In this case the next-hop is node 4, which is characterized by a higher residual energy.

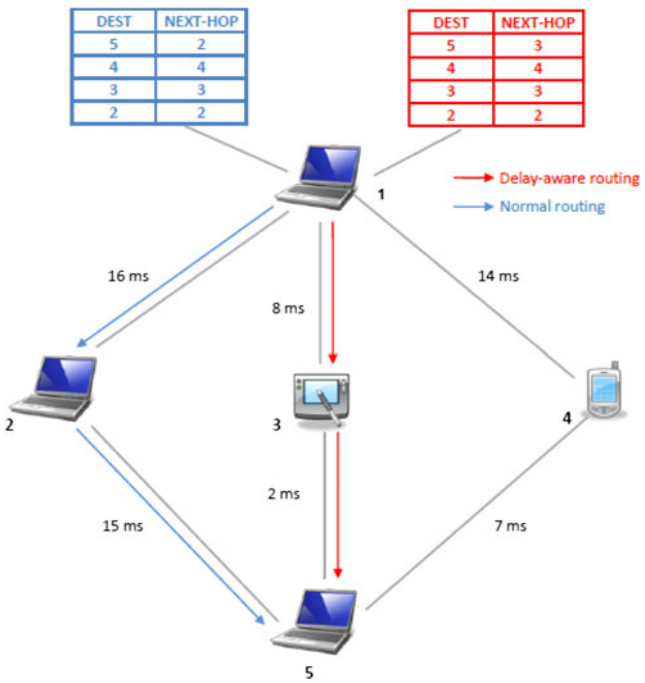

Fig. 4 A simple example of delay-aware routing

In Fig. 4, MChannel works as usual (simple hop count metric); hence, it does not take into account the delays of the several links. On the contrary, the improved version of MChannel selects the path with minimum end-to-end delay.

5.2 Quantitative evaluations

We now describe some experimental results related to our proposed extension of MChannel, which show the achieved improvement with respect to both *network lifetime* (*Energy-aware MChannel*—EMC) and *average end-to-end delay* (*Delay-aware MChannel*—DMC). Network lifetime can be defined in several ways [8]: (i) the time before the first node fails, (ii) the time before a MANET application cannot work, because a sufficient number of nodes is not available and (iii) the time before communication between each pair of nodes in the network is no longer possible (network partitioning). On the other hand, the average end-to-end delay is the end-to-end delay averaged considering all the possible source-destination pairs in the network. We report also some experiments executed with scenarios in which the requirements of lifetime and delay are not compatible. In fact, if we analyze the two implemented mechanisms, we can deduce that they might achieve similar results in terms of both average delay and network lifetime. Nevertheless, we have found scenarios characterized by a different behavior of the energy and delay aware routing protocols. Mchannel provides an emulator (see Fig. 5) which can be used to test applications.

Inside the emulator, the UDP level has been replaced with a virtual one: instead of delivering messages through sockets, local queues of messages are used. With such an approach, sending a message to a neighbor through a socket actually translates into inserting

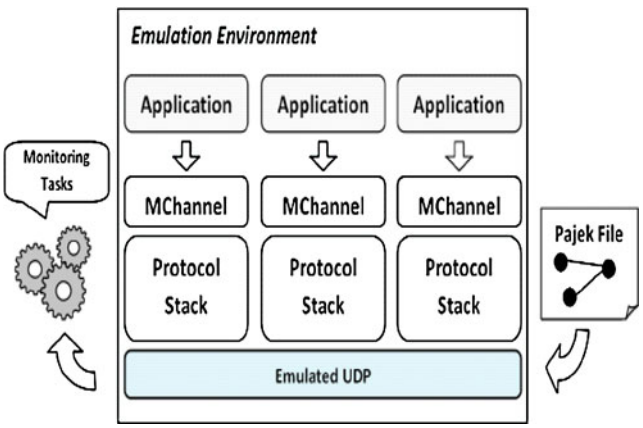

Fig. 5 MChannel's emulator

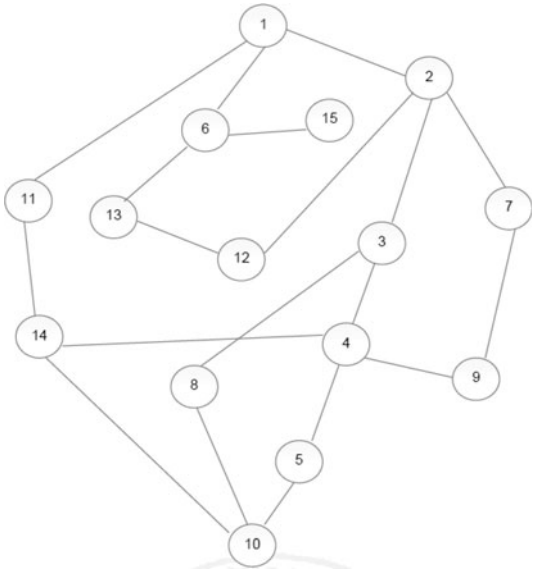

Fig. 6 Network topology used in the trials

it in the neighbor's message queue. The employed emulated network is provided to the emulator by means of a Pajek file.[2] The advantage of such an instrument is that no modification to an application is required when it comes time to deploy it on an actual network.

The scenario we have used for the emulations includes 15 static nodes (the network topology is reported in Fig. 6). We have no trial involving mobile nodes. This is due to two main reasons. First of all, we have not designed a brand new protocol; hence, there was nothing new we might have discovered in case of mobility. Furthermore, the employed metrics are associated with properties of the network nodes which have no strict relation with their mobility. The communication model, the initial node energy levels, as well as the delay values used in the emulations are different in each of the tests, and thus provide a reliable basis for the performance considerations we are going to derive from the trials.

5.2.1 Network lifetime

Since we have used emulations to perform tests, we had to develop a model for energy consumption. An implementation of such model has been integrated in each emulated node, so as to simulate the energy decrease due to message transmission and reception. The mentioned model is similar to the one used in [10] and [9]. We assume that all the nodes are equipped with a network interface with a 54 Mbps transmission rate (as in IEEE 802.11g). The energy necessary to transmit a packet p from a node is computed as: $E(p) = v \cdot i \cdot t(p)$ Joules, where v is the voltage, i the current and $t(p)$ the time needed to transmit the packet. Voltage has been set to 5 V, transmission current is 280 mA, reception current is 240 mA. The time $t(p)$ is computed as $(size(p))/(54 \cdot 10^6)$. We do not take into account overhearing (i.e. node energy consumption caused by the reception of traffic not directed to the node, but inevitably heard).

In the first test, nodes 6, 9 and 10 belong to the same multicast group. Node 6 sends 200 application messages to the group. All the others, except the three nodes in the group, have a very low energy. Figure 7 reports the number of alive nodes as a function of the number of transmitted messages. This figure shows how the routing algorithm influences the network lifetime during a communication session, both in the case of MChannel in its plain version and in the case of MChannel optimized in order to work with energy-aware path selection ('MChannel opt' in the picture).

We see that a higher number of nodes fail in MChannel, compared to its energy-optimized version. The reason is that the native version uses the hop-count metric, without taking into account the residual energy of the nodes in the network. With the energy-optimized version, instead, each node in the network selects paths which exclude low energy nodes. Control messages spread information about each node's status through the network. Figure 8 reports the distribution of residual energy of the nodes in the network, in a scenario where node 6 sends 500 messages to its multicast group and all the nodes have maximum energy. We notice that, in case of energy-aware routing, nodes are exploited for the data forwarding in an almost uniform way. In the other case, some nodes are used more than

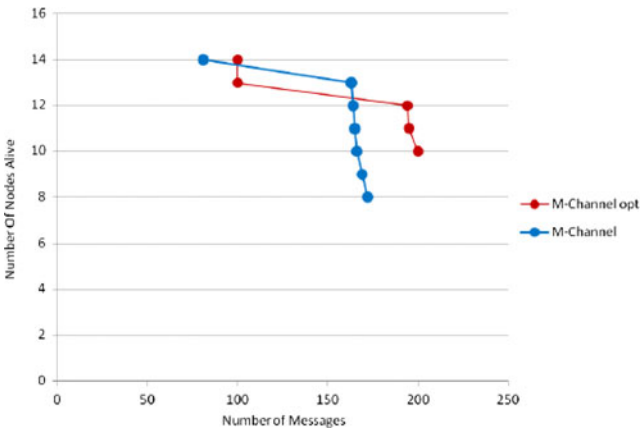

Fig. 7 Network lifetime

[2] http://vlado.fmf.uni-lj.si/pub/networks/pajek

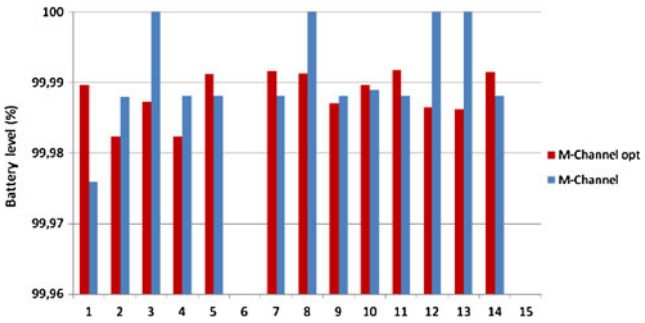

Fig. 8 Nodes battery level distribution

others and this reduces the network lifetime. For instance, node 6 exploits node 1 quite intensively because this is the best next-hop for the data transmission (when using the hop-count metric). In this case, the energy of node 1 decreases quickly. In the first case, however, the transmission overhead is distributed between nodes 1 and 13. In general, the optimized routing results in a uniform distribution of energy expenditure among all the nodes in the network, and it increases the time which elapses before network partitioning. We do not report the values of nodes 6 (the sender) and 15, because they are not involved in packet routing.

As a further remark, we show the advantage resulting from the use of a node cost depending on the energy reduction rate of the node. In this scenario, node 10 has a data connection with node 2. All the nodes have a low energy, except nodes 8 and 3 which have a maximum battery level. If a node's cost depends on its residual energy (without considering the drain rate), the route selection scheme always chooses route $8-3-2$, because nodes 8 and 3 have maximum energy. If the drain rate component is considered, the transmission overhead is distributed among more paths, resulting in a reduction of energy expenditure in nodes 8 and 3. Figure 9 shows the trend of the residual energy in node 8.

As it comes out from the picture, the drain rate causes a fair exploitation of the nodes, but it also prevents the incoming links of nodes with higher energy (e.g. node 8) from becoming congested. If we take into account both residual energy and energy consumption rate, we strike a balance between preservation of nodes with low energy and overhead reduction of nodes with higher energy.

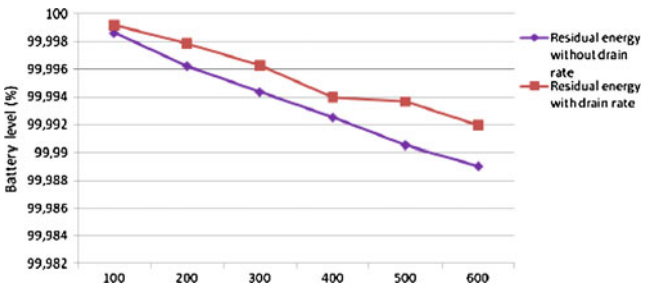

Fig. 9 Residual energy in node 8

5.2.2 Average end-to-end delay

In order to evaluate the average end-to-end delay, we set up a communication scenario which is the same as the one used for the energy consumption experiments, with the only difference that in this case energy values have been randomly set since they are not the subject of our investigations. We compare native MChannel with the new version configured so as to perform a delay-aware routing. The synthetic delays introduced in the links are random. We consider two cases: the former in which the link delay values are characterized by a low variance (i.e. they differ only slightly between each other); the latter in which they show, on the contrary, a large variance. Figures 10 and 11 compare the average end-to-end delay of the network in the two cases above.

First of all, the figures indicate a reduction of the average end-to-end delay in both cases. We further remark that, when link delays are comparable, the resulting improvement is smaller with respect to the alternative case. In fact, when the differences among link delays are not relevant, the optimal path chosen by the native hop-count metric might match the minimum delay path identified by the optimized version of the middleware. In the opposite case, it is more likely that we eventually find a path in the network with more than the minimum number of hops, but nonetheless characterized by a shorter end-to-end delay. In this

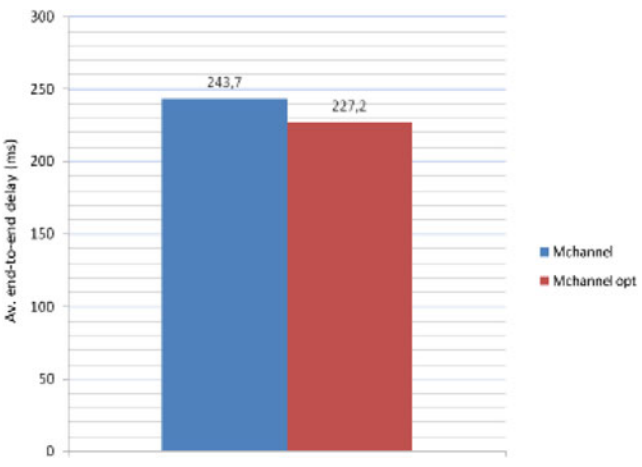

Fig. 10 Average end-to-end delay: low variance scenario

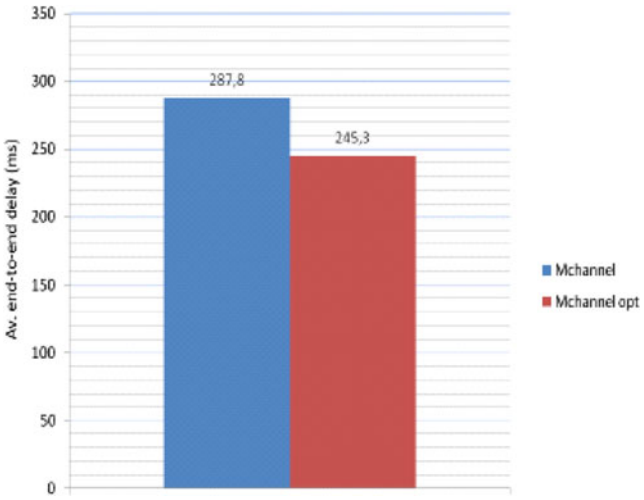

Fig. 11 Average end-to-end delay: high variance scenario

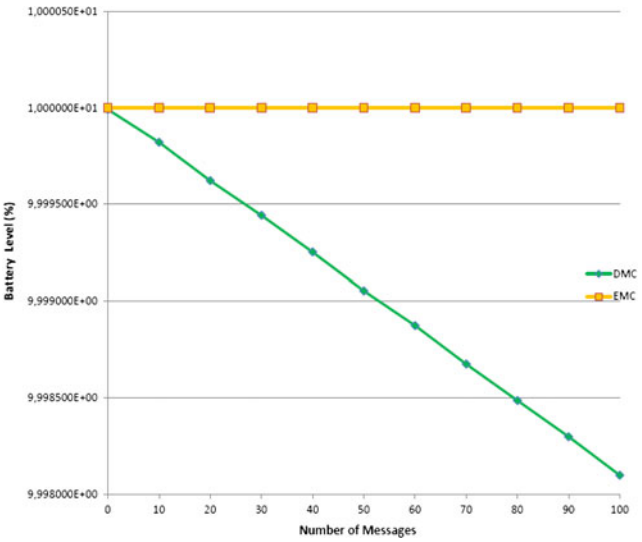

Fig. 12 Scenario 1

case, Dijkstra's algorithm based on the hop-count does not select the same path as the delay-aware selection scheme.

5.2.3 Energy vs delay metric

In the following test we consider two different scenarios. With the former, we will show that the adoption of delay-aware routing can negatively affect energy performance. In the latter, we instead demonstrate how energy-efficient routing can result in a higher end-to-end delay. In both cases, we use the novel module with either the energy-aware or the delay-aware algorithm alternatively activated. The tests show two cases in which delay-aware and energy-aware routing behave differently. In fact, we can easily understand that there are cases in which the paths selected are the same. For example, if we find a path with the minimum delay, the same path might also be selected by using the energy metric. The reason is that the energy cost of a node includes an estimation of its energy dissipation rate, which increases or decreases during the transmission. Now, if this rate is high, i.e. the node is forwarding many packets, the delay of its incoming links might also be high. In this case, both Dijkstra and the max-min algorithm might not choose that node as an ideal forwarder. The section concludes with a further comparison between one of the two advanced metrics and the native hop count approach.

In the first scenario, node 10 sends 100 unicast application messages to node 2. All the nodes have a high battery level, except node 8 (whose level is 10%). We observe the trend of node 8's battery level as a function of the number of messages sent, both in the case of DMC and in the case of EMC (Fig. 12). To better notice the contrast between the two considered routing strategies in this scenario, we consider only the decrease of energy caused by data traffic.

As expected, energy-aware routing works better because, unlike its delay-aware competitor, it takes into account the low battery level of node 8. With EMC, energy remains constant because the routing table of node 10 does not indicate node 8 as next-hop towards the destination 2. Hence, node 8 does not forward traffic to node 2. Instead, DMC selects the path based only on the link delays, and more often than not it happens to select node 8 as a next-hop.

In the second scenario, node 10 sends unicast messages to node 2; all the nodes have the same battery level and the delays on the links are random. Figure 13 compares average end-to-end delay of EMC and DMC.

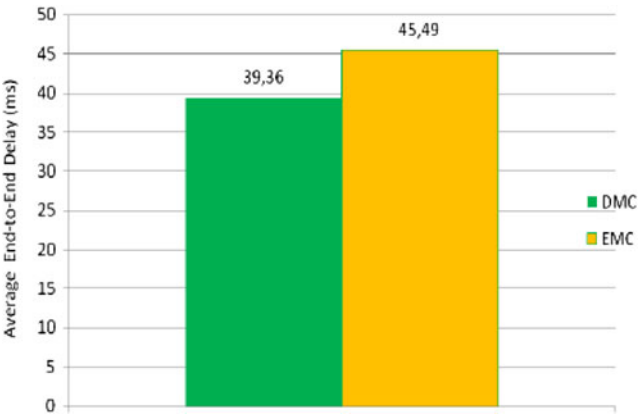

Fig. 13 Scenario 2

We have explained above a similarity between the delay and the energy-based metrics. In this scenario, since all the nodes have the same battery level, the energy rate component has a heavier impact on routing decisions. Nevertheless, in the presence of random delays on the links, EMC uniformly distributes among the network nodes the overhead associated with message forwarding; though, it does not consider the random delay introduced by the links. For this reason, it works worse than DMC.

As a last experiment, we compare the average number of hops in the cases of native MChannel, EMC and DMC. It is obvious that in this case the native MChannel outperforms its two extensions, because it optimizes the hop-count metric. DMC, in turn, performs slightly better than EMC, because delay minimization can often bring to the selection of paths with fewer hops (Fig. 14).

For the benefit of the readers, we finally describe (at a very high level) a typical scenario where the EMC is preferable to the DMC. If the nodes in the network run an intensive computing application characterized by a rare communication, the main cause of energy consumption would not be due to the transmission and reception of messages but rather to the frequent computations. In this case, a high energy drain rate on a node is not due to a notable traffic on the incoming/outgoing links. Hence, while the energy-aware routing would not consider such node as a good candidate for a potential forwarding path, the delay-aware routing might do so. In this case, the adoption of DMC would most probably result in a shorter network lifetime.

Indeed, the choice of one or the other routing strategy is strictly related to both the specific application considered and the network conditions (especially in terms of nodes battery level). For instance, real time applications might work better with DMC than with EMC. Nevertheless, if some nodes have low energy and there is the risk of network partitioning, EMC is definitely more indicated. The combined adoption of both strategies represents a further option. Based on the application requirements, a node can start with DMC, and once one or more nodes in the network reach a specified threshold of residual energy, it can switch to EMC in order to best adapt to the new situation.

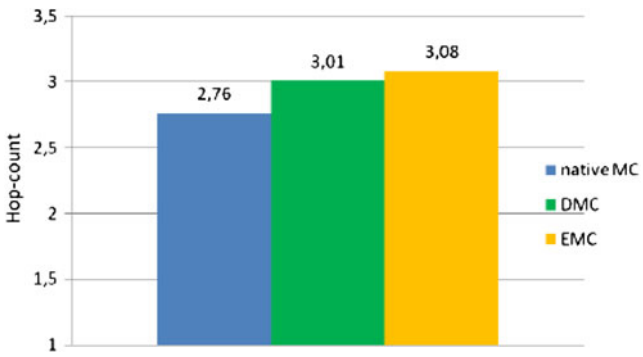

Fig. 14 Scenario 3

6 Conclusions and future work

This work has analyzed ways to optimize both delay and energy metrics in Mobile Ad-hoc Networks. We have proposed and evaluated two extensions of the OLSR protocol aimed at considering the mentioned metrics. Experimental results have demonstrated that such extensions overcome MChannel with regard to the above metrics.

In fact, our tests have shown that the new routing module we added to the MChannel middleware has a positive impact both on network lifetime and end-to-end delay. In particular, energy-aware routing allows for a prolonged duration of the network nodes, when compared to the native OLSR implementation based on shortest path routing. We guarantee a uniform utilization of the nodes involved in the data forwarding phase, which results in extended lifetime of the nodes themselves, as well as improved network availability. Similarly, delay-aware routing entails an improvement in terms of average end-to-end delay. Such improvement is greater if the link delays differ significantly between each other. We have also discussed the mutual impact of the two metrics we introduced, by showing that, even though in the most general case they find similar paths, scenarios exist in which the choice of a particular metric negatively affects the performance of the other.

Different extensions can be considered as future work. The most natural one is to find a way to consider both energy and delay, in the path selection policy. This requires defining a multi-objective model and subsequently designing a proper heuristic to solve it in a feasible way.

Other additional work can definitely be done to improve the energy-aware routing. At the moment, the Minimum Drain Rate (MDR) mechanism is used, which is known to not guarantee a low total transmission power consumption. The Constrained MDR (CMDR) might hence introduce an improvement in this sense, and thus represents one of the alternatives we are willing to analyze.

Future work which requires a greater effort concerns the study and implementation of strategies consider-

ing brand new metrics associated with network nodes and arcs. This can be done by exploiting the inherent flexibility of the framework we provided.

References

1. Anathan R, Redi J (2002) A brief overview of ad hoc networks: challenges and directions. IEEE Communications Magazine, 50th Anniversary Commemorative issue, pp 20–22
2. Conti M, Giordano S, Maselli G, Turi G (2004) Cross-layering in mobile ad hoc network design. IEEE Computer, Special Issue on Ad Hoc Networks
3. Lopez PG, Tinedoa RG, Alsina JMB (2010) Moving routing protocols to the user space in MANET middleware. J Netw Comput Appl 33(5):588–602 (Middleware Trends for Network Applications)
4. Lin X, Shroff NB, Member S, Srikant R (2006) A tutorial on cross-layer optimization in wireless networks. IEEE J Sel Areas Commun 24:1452–1463
5. Meghanathan N (2009) Survey and taxonomy of unicast routing protocols for mobile ad hoc, vol 1(1). The international journal on applications of Graph Theory in Wireless Ad hoc Networks and Sensor Networks (GRAPHHOC). Jackson State University, Jackson, MS, USA
6. Mahapatra P, Li J, Gui C (2003) QoS in mobile ad hoc networks. IEEE Wirel Commun 10(3):44–52
7. Toh C-K (2001) Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks. IEEE Commun Mag 39(6):138–147
8. Mahfoudh S, Minet P (2008) Survey of energy efficient strategies in wireless ad hoc and sensor networks. In: IEEE international conference on networking. Cancun, Mexico, pp 1–7
9. De Rango F, Fortino M (2009) Energy efficient OLSR performance evaluation under energy aware metrics. In: Symposium on performance evaluation of computer and telecommunication systems, pp 193–198
10. Kim D, Garçia Luna Aceves JJ, Obraczka K, Cano J, Manzoni P (2002) Power-aware routing based on the energy drain rate for mobile ad hoc networks. In: Proceedings of IEEE 11th international conference on computer communications and networks, pp 562–569
11. Perkins CE, Belding-Royer E (2001) Quality of service for ad hoc on-demand distance vector routing. Mobile Ad Hoc Networking Working Group, Internet Draft (expired on January 2001)
12. Lee S, Gerla M (2001) Split multipath routing with maximally disjoint paths in ad hoc networks. In: Proceedings of IEEE ICC
13. Sondi P, Gantsou D (2009) Voice communication over mobile ad hoc networks: evaluation of a QoS extension of OLSR using OPNET. In: Proceedings of AINTEC'09, Bangkok
14. Badis H, Munaretto A, Agha KA, Pujolle G (2003) Qos for ad hoc networking based on multiple-metric: Bandwidth and delay. In: IFIP/IEEE international conference on mobile and wireless communications networks
15. Badis H, Al Agha K (2005) QOLSR, QoS routing for ad hoc wireless networks using OLSR. Eur Trans Telecommun 16:427–442. doi:10.1002/ett.1067
16. Mahfoudh S, Minet P (2008) An energy efficient routing based on OLSR in wireless ad hoc and sensor networks. In: Proceedings of the 22nd international conference on advanced information networking and applications—workshops (AINAW '08). IEEE Computer Society, Washington, DC, pp 1253–1259. doi:10.1109/WAINA.2008.60
17. Jaffrès-Runser K, Schurgot M, Comaniciu C, Gorce JM (2010) A multiobjective performance evaluation framework for routing in wireless ad hoc networks. In: IEEE 8th international symposium on modeling and optimization in mobile, ad hoc, and wireless networks (WiOpt)
18. Marwaha S, Srinivasan D, Tham CK, Vasilakos A (2004) Evolutionary fuzzy multi-objective routing for wireless mobile ad hoc networks. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC2004), vol 2, pp 1964–1971
19. Montana D, Redi J (2005) Optimizing parameters of a mobile ad hoc network protocol with a genetic algorithm. In: Proceedings of the 2005 conference on Genetic and Evolutionary Computation (GECCO '05), pp 1993–1998
20. Sarangapani J (2007) Wireless ad hoc and sensor networks: protocols, performance, and control. Control Engineering Series
21. Asokan R, Natarajan AM (2008) An approach for reducing the end to end delay and increasing the network lifetime in mobile ad hoc networks. Int J Inf Technol 4(2):121–127
22. Asokan R, Natarajan AM (2008) Performance evaluation of energy and delay aware Quality of Service (QoS) routing protocols in mobile adhoc networks. IJBDCN 4(1):52–63. doi:10.4018/jbdcn.2008010104
23. Li Q, Aslam J, Rus D (2001) Online power-aware routing in wireless ad-hoc networks. In: Proceedings of the 7th annual international conference on mobile computing and networking (MobiCom '01). ACM, New York, pp 97–107. doi:10.1145/381677.381687