An any-sink energy-efficient routing protocol in multi-hop Wireless Sensor Networks for planetary exploration*

Guido Oddi, Antonio Pietrabissa, Francesco Liberati, Alessandro Di Giorgio, Raffaele Gambuti, Andrea Lanna, Vincenzo Suraci, Francesco Delli Priscoli
Department of Computer, Systems and Management Engineering (DIAG)
University of Rome "La Sapienza"
Via Ariosto 25, 00185, Rome, Italy
{oddi, pietrabissa, liberati, digiorgio, gambuti, lanna, suraci, dellipriscoli}@dis.uniroma1.it

Abstract—Wireless Sensor Networks (WSNs) are made of spatially distributed autonomous sensors, which cooperate to monitor physical or environmental parameters and to deliver such data to one or more central sinks. A promising field of application of WSNs is planet exploration, characterized by a continuous monitoring of the surface, to have clear notion of planet conditions and prepare for future manned missions. The potentially large size of the region to be monitored and the line-of-sight limitations (as, for instance, on the Moon, the case study discussed in the paper), hamper the possibility of having 1-hop sensor-sink communication. Therefore, the sensors must be able to create and maintain a *multi-hop ad-hoc network,* to allow sensed data to reach the sink(s). This paper proposes an ad-hoc routing algorithm applicable to WSNs for planetary exploration, aiming at (i) assuring any-cast communication with multiple data sinks, (ii) minimizing the control overhead for the routing paths maintenance, (iii) being light in terms of memory/computational requirements, to be installed into low-power and low-memory/processing devices, (iv) being rapid to reconfigure in presence of node failures and (v) optimizing the choice of the routes, to achieve energy balancing and saving. Extensive simulations demonstrate the efficiency of the proposed approach.

Keywords—Wireless Sensor Networks; planetary exploration; ad-hoc networks; routing; energy-awareness; low-overhead.

1 Introduction

Wireless Sensor Networks (WSNs) are made of spatially distributed autonomous sensors, which cooperate to monitor a certain physical or environmental condition and pass their data through a network to a central processing location. These networks, initially motivated by military applications, are used in the industry for process and machine health monitoring and appear to be a promising tool in several areas like catastrophe evaluation, fire monitoring, environmental experiments and in other distributed sensing applications.

A very promising field of application of WSNs is planet exploration [1]. In order to prepare for manned missions to other planets, it is necessary to permanently monitor the surface environment and have a clear notion of its conditions. The surface to be monitored could be very large in dimension and, thus, hundreds of small wireless sensors should be dropped onto such surface to assure a uniform and sufficient coverage. In this kind of large-scale scenario, standard routing protocols applied in WSNs (as for instance LEACH [2], PEGASIS [3], GAF [4] and others) are not applicable, since they require a direct communication between each sensor deployed over the surface and the data sink, which is in charge of gathering all monitored data and sending back,

*The research leading to these results has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 312826.

through a satellite link, to Earth. The large size of the region to be monitored and the line-of-sight limitations on remote planets (for instance the Moon, as studied in the SWIPE project [1]), impose constraints on the possibility to have 1-hop sensor-sink communication. Therefore, the autonomous sensors must create and maintain a *multi-hop ad hoc network* (i.e., nodes act also as relays for traffic generated by other nodes), instead of a star topology (or a 1-level clustered topology) as commonly happens in WSNs. Multi-hop protocols suitable for standard fixed networks ([5], [6], [7], [8]) or mobile wireless networks ([9], [10], [11], [12], [13], [14]), are, however, not optimized for the WSN segment, which has very strict constraints, especially regarding the scarcity of computational, bandwidth and energy resources. Moreover, differently from the protocols used in Mobile Ad-Hoc Networks (MANETs), WSNs normally consider fixed nodes. Finally, since the human intervention is limited, the network should be able to self-optimize, in particular to save and to balance the use of the sensors batteries. This is particularly important in environmental monitoring scenarios, in which sensors should remain alive as longer as possible, to continuously provide the precious monitored information. In this respect, as mentioned above, a multi-hop approach should be followed, taking ideas from the routing in MANETs, but the limited resources availability must be taken into account as a constraint.

In this paper, an ad-hoc routing algorithm applicable to WSNs for planetary exploration (such as the one of the SWIPE project) is proposed, with the aims of (i) assuring any-cast communication with multiple data sinks, deployed for redundancy purposes, (ii) minimizing the control overhead for the maintenance of the routing paths, (iii) being light in terms of memory and computational requirements, in order to be installed into low-power and low-memory/processing devices, (iv) being robust and rapid to reconfigure in presence of node failures and (v) optimizing the choice of the routes, to achieve energy balancing and saving.

This paper is organized as follows. Section 2 illustrates the planetary exploration scenario, including the challenges, the network view and the main objective of the routing in the considered WSN. Section 3 reports some state of the art of the routing applied in terrestrial ad-hoc networks. Section 4 describes the proposed approach and also reports the details about the control packets, internal data structures and the pseudo-code of the routing procedures, along with memory and computational complexity considerations. Section 5 shows the simulation results, proving the efficiency of the proposed approach. Finally, conclusions are drawn in Section 6.

2 Planetary exploration scenario – the SWIPE project

2.1 Mission scenario and challenges

This paper presents a routing algorithm to be applied in planetary exploration scenarios, such as the one considered in the SWIPE project [1]. The Space WIreless sensor networks for Planetary Exploration (SWIPE) project intends to bring satellite and WSN technologies to space. In order to prepare for manned missions to other planets, the surface environment must be permanently monitored in order to have a clear notion of its conditions. Hundreds or thousands of small wireless sensors, deployed onto the surface, would create their own ad-hoc network, while some of them, equipped with satellite communication capabilities, would establish a link between the WSN and the satellite. Data gathered from the sensors would be processed and sent to the satellite and later to Earth. The following figure shows the considered scenario.

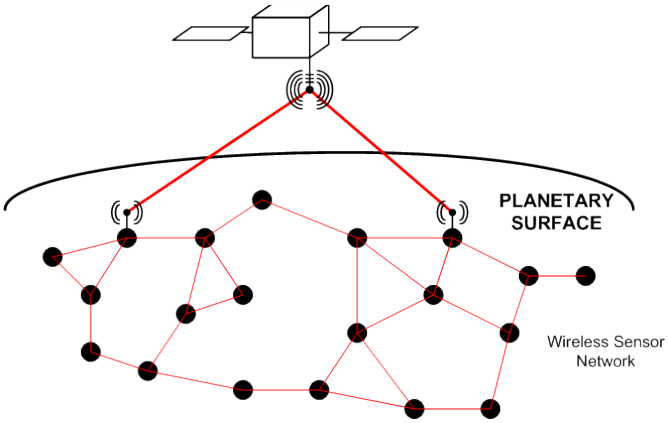


Figure 1 Planetary exploration scenario ([1]).

The mission scenario considered in SWIPE is the Moon. One of the most interesting local features on the Moon's surface are *swirls*. Swirls have a high albedo such as the Reiner Gamma swirl [15] and tend to be associated with magnetic anomalies [16]. Some scientists speculate that they are consequences of different space weathering conditions. The fact is that these anomalies have not been studied in detail yet and there is not much information available about them. The WSN presented in this paper is devoted at studying one of these anomalies. There are several different swirl locations on the Moon surface, divided between *nearside* and *farside*. The nearside swirls have been extensively studied with remote observations whereas farside swirls are more enigmatic. Actually, the entire farside is less known and understood, which from a scientific point of view makes it a more interesting location. By analysing different locations against several features of interest of the Moon, Mare Ingenii is the site that gathers the highest number of interesting elements, from a scientific perspective and was the selected landing site for the SWIPE mission ([17]).

A series of challenges and constraints arise from the considered scenario, dictated mainly by the mission requirements and by the harsh environment. In this paper we focus on the challenges related to the problem of optimizing the network management and control. In the first place, due to radio line-of-sight limitations, nodes cannot be positioned over 500m of distance one to each other. This distance has been calculated taking into account the rotundity of the Moon, the size of the nodes which could be deployed on the planet (see the results of [1] for further details) and margins because of the presence of boulders and craters. Thus, in order to guarantee that each node is connected with the data sink(s), the limit to the communication range imposes that nodes must support *multi-hop communication*. Furthermore, energy efficiency is perhaps the main challenge to be faced, given the *limited energy storage* capabilities of the nodes and the absence of primary recharging energy (especially during nights, when possible solar panels cannot provide energy to the batteries). Moreover, communication tasks are energy consuming, therefore energy wise and reconfigurable network protocols have to be developed in order to maximize WSN operational life and avoid that the energy depletion of few nodes puts in threats the overall mission. Furthermore, network protocols have to be compliant with the expected degree of change on the WSN topology, considering *node or link failures,* for instance due to the harsh environment or to complete energy depletions. Additionally, the network protocols must be able to handle hundreds or thousands of

nodes (e.g., in SWIPE largest scenario, up to 1200 nodes): i.e., they must be *scalable* with respect to the network size. The network topologies reported in Table 1 have been envisaged in the project (see [19]), differentiated with respect to coverage, resiliency and nodes positioning.

Finally, nodes periodically (with intervals of 300s or 600s) measure a set of parameters from the environment (i.e., radiation, illumination, temperature, dust, as explained in Section 2.3), compose packets using proper payload sizes (of the order of some hundreds of bits) and send them to the data sinks. For further details on more precise figures, please refer to [19].

2.2 Network system view and components

The network considered in this paper is characterized by up to hundreds or thousands of small wireless sensor nodes deployed onto the surface of a planet to assure a uniform and sufficient coverage. SWIPE defined three nodes, belonging to the WSN segment:

- *SWIPE regular nodes* (or simply SWIPE nodes): sensor nodes which are in charge of (i) sensing relevant data from the environment, (ii) fuse and aggregate sensed data, (iii) act as relays for other nodes, allowing transmitted data to be disseminated throughout the network, in a multi-hop fashion.
- *Data sinks*: SWIPE nodes which are also in charge of collecting and processing the data generated by the SWIPE nodes. Having more than one data sink increases the tolerance to possible failures.
- *Exit points*: SWIPE nodes that have satellite communication capability and are responsible for transmitting the data collected by the data sinks to the satellite.

Figure 2 shows the network flow involving the three above-mentioned components, in which multiple data sinks and multiple exit points are placed into the network. Scientific (and housekeeping) data, generated by the nodes, flow towards the data sinks following multi-hop paths (and passing through one or more relay nodes, e.g., other regular nodes).

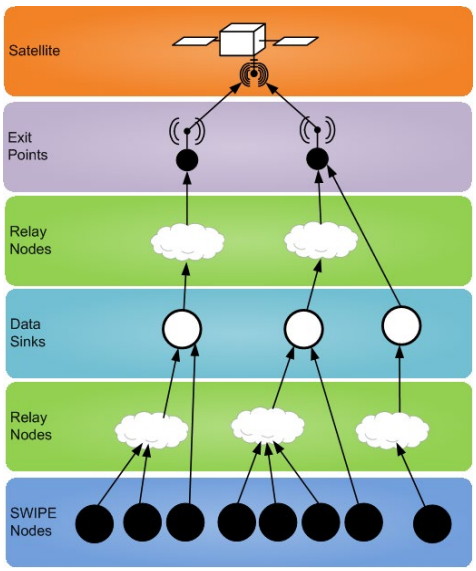
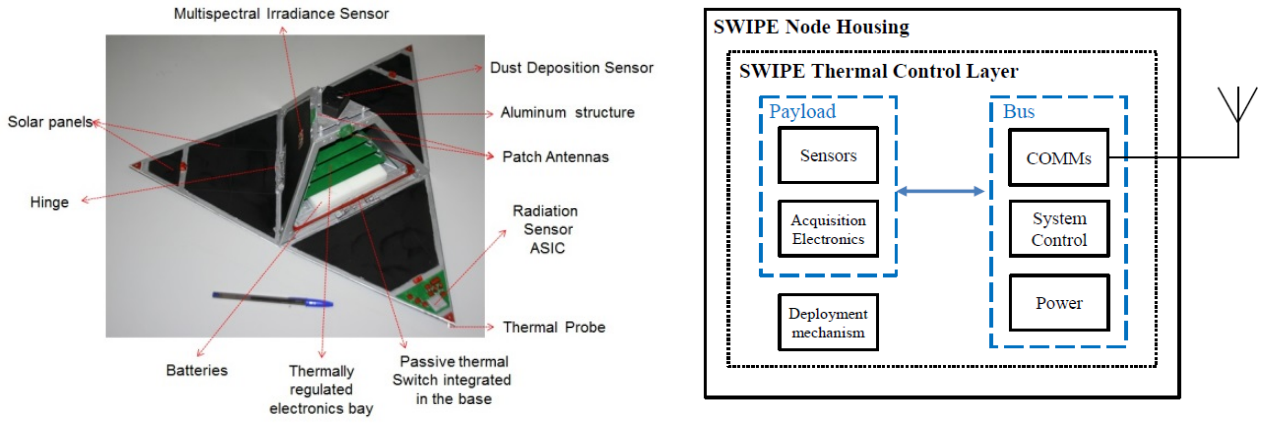


Figure 2 Network flow of the SWIPE logical nodes (*taken from* [21]).

The information stored in the data sinks are periodically sent to the exit points which are in charge of delivering the data using the satellite segment. Exit points communicate with the satellite every 8449s. This is the duration, in fact, of the period in which the orbiting satellite is not visible from the WSN (the so called non-line-of-sight (NLoS) period, see [20]).

2.3 The SWIPE node

In this section, we briefly summarize the node architecture, principally to highlight the types of measurements which will be taken by the SWIPE nodes as well as the internal logical structure and the constraints it could pose. For further details, please refer to [21]. The SWIPE node is a tetrahedron highly optimized both in mass (max 2000g) and volume (200mm×200mm of base and 200mm of height). A light aluminium structure, forming the tetrahedron and providing stiffness to the node, holds all the elements together. The SWIPE node is equipped with deployable solar panels and a passive thermal switch to regulate the temperature of the electronics bay in function of the external temperature. The following figure shows a mock-up of the proposed SWIPE node (see [21] for further details):



(a) (b)

Figure 3 (a) Mock-up of the proposed SWIPE node; (b) SWIPE node logical structure.

The sensors of the SWIPE nodes are the following: (i) *radiation sensor*, which is an ASIC to monitor total ionizing dose and single effect upsets radiation; (ii) three *surface thermal sensors* situated at the end of the node walls (in contact with the ground once the walls are deployed); (iii) three *multispectral sensors* (VIS, IR and UV); (iv) *dust deposition sensor* to measure the dust deposited over a horizontal surface during a certain exposition time. From a logical point of view, as depicted in Figure 3(b), the SWIPE node is composed by a *payload* (which contains the sensors and the acquisition electronics) and a bus, including the *system control module*, a *power module* and a *communication module*. The power module is in charge of handling the batteries. The system control module is an on-board computer (OBC), which handles the data interfaces with the payload and the communications module, manages the power distribution inside the node, is responsible for running the network algorithms that enable the creation of the WSN and the data processing/fusion algorithms. The communication module is in charge of the external communications and is based on Software-Defined Radio (SDR) technology. The compact nature of the SWIPE node imposes constraints on the battery, computation and memory capacity. The network algorithms must take

into account the scarce availability of such kind of resources. Moreover, fusion and aggregation techniques must take into consideration the nature of the measured data (temperature, radiation, illumination, etc.) to perform accurate and optimize data fusion.

2.4 Objectives of the WSN routing protocol in the proposed scenario

According to what defined in the network requirements of the SWIPE project ([20]), several objectives should be achieved by an optimized routing protocol and are reported in the following list:

- To compute feasible paths, in a *multi-hop fashion*, among the nodes and the data sinks.
- To allow the communication with *multiple data sinks*. The decision of having more than one data sinks has two main motivations: (i) to be robust of possible data sinks failure; (i) nodes close to the data sink (and the data sink itself) consume their energy faster than farther nodes. This implies that, if nodes close to the sink discharge, the data sink is not able to receive any data and gets disconnected from the network. A solution to this problem is to deploy several sinks: in this way, in fact, traffic is balanced among the multiple data sinks and the network lifetime can be significantly increased (see, for instance, [22]).
- To *minimize the control overhead* for the maintenance of the routing paths, due to the large scale of the WSN.
- To be *light in terms of memory and computational requirements*, in order to be installed into low-power and low-memory/processing devices.
- To be *robust and rapid to reconfigure in presence of node failures*, in the sense that the network connectivity and functionalities will be guaranteed also in case of node failures (unless the failure partitions the network in strongly connected components, obviously).
- To optimize the use of the feasible paths, to balance the traffic among the nodes of the network and maximizing the *network lifetime*, avoiding battery discharges, especially during the nights (in which no solar energy can be exploited). The energy awareness can be achieved by considering the residual battery of the nodes as an input feedback for the routing algorithm. Moreover, paths characterized by a short distance between hops should be preferred, principally if the radio equipment is able to modulate the transmission power in function of the distance with the next-hop of the path.

3 Routing in ad-hoc networks

Routing algorithms for multi-hop networks could be grouped in three families: *proactive*, *reactive* (or on-demand), or *hybrid* protocols. Proactive protocols are characterized by the fact that each node maintains routing information towards every destination of the network, regardless of the effective immediate necessity of sending data towards such destination. One main example in routing for ad-hoc networks is OLSR [23]. Reactive protocols envisage that a specific source-destination path is computed only when a specific data packet must be delivered from the source towards the destination. One main example in routing for ad-hoc networks is AODV [24]. Hybrid protocols consist of a combination of proactive and reactive features, for instance by differentiating specific destinations. One example in routing for ad-hoc networks is HWMP [25].

Another classification of routing algorithms for multi-hop networks is done according to the method for the paths computation. The two main methods, in this sense, are the *link state* and the

distance vector approaches. In link state routing algorithms, every node builds a logical view of the network connectivity, in the form of a *graph.* Each node independently computes the route towards every possible destination in the network. The name comes from the fact that a proper state is associated to each link (for instance the link availability, the estimated delay, etc.): this state is used for the best path computation. An example of such method in routing for ad-hoc networks is the above-mentioned OLSR. In distance vector algorithms each node has not the knowledge of the whole topology (as in the link state approach), but associates, to each destination, a value determining the distance (or the cost) as well as the next-hop to reach such destination. An example of such method in routing for ad-hoc network is the above-mentioned AODV.

In the following we will show the main characteristics of the two main protocols used in ad-hoc networks (OLSR and AODV) as well as their extensions available in literature, proposed to optimize their performances in specific application scenarios. A particular emphasis will be done to the state of the art in energy-aware routing algorithms. Before illustrating some energy-aware approaches, it is important to introduce two principal metrics which evaluate the goodness of a routing algorithm in terms of energy-efficiency: (i) the *minimum node lifetime*, which could be defined as the duration of time until the first node in network fails because of battery depletion; (ii) the *network disconnection time*, which is the quantity of time before the network is disconnected (i.e., there is at least a couple of nodes for which there not exist any route connecting the two nodes) Moreover, an example of hybrid protocol available in literature is briefly described. The analysis of the state of the art has been an important basis to develop the proposed routing algorithm, as will be clear in next Sections.

3.1 OSLR and its extensions

The Optimized Link State Routing (OLSR) protocol [23] is a largely-used proactive protocol for MANETs, based on the link state approach. To generate the network topology inside each network node, a control messages flooding is needed. This implies that a very high number of messages are potentially exchanged among nodes, due to the flooding. The larger is the network, the higher is the control overhead to maintain the topology view in each network node. In order to minimize the number of exchanged messages, OLSR uses the concept of *Multipoint Relay* (MPR). Each node selects a subset of its neighbours as Multipoint Relays. Only the nodes selected as MPR are responsible for the dissemination of the topology information (the *Topology Control* messages – TC) throughout the network: in this way, the overhead is reduced compared to the traditional link state approach. Once the network topology is built up, each node computes the path towards each destination, using, for instance, the famous Dijkstra algorithm [26]. Since the protocol is proactive, the flooding of TC messages (and the broadcasting of Hello messages among neighbours) is done on a regular basis. The reactivity of the protocol to possible network variations could be increased by decreasing the interval of the broadcasting of Hello and TC messages. Finally, in OLSR nodes can announce their *willingness* to act as MPR for their neighbours.

The standard OLSR algorithm has been largely studied and extended in literature ([27], [28]). A main extension of the OLSR protocol is the Fish Eye extension [29]. The principle of Fish Eye is that the topology information could be refreshed more frequently for close nodes than for far nodes. This could be achieved by properly setting the time-to-live (TTL) value of the TC message. In [29] the authors studied the OLSR protocol scalability with fish eye extension. They affirmed that OLSR

with fish eye extension is capable to manage a higher amount of nodes than the standard one, by reducing the control overhead. Typical extensions of the OLSR algorithm regard the introduction of Quality of Service (QoS) metrics and requirements into the routes computation process. In the QOLSR protocol [30], the TC messages contain other information, such as the estimated delay and available data rate for each link of the network. Once built the topology with each link associated to a state in terms of delay and data rate, each node solves a Delay and Bandwidth Constrained Least Hop path problem. Such problem is an optimization problem of the following form:

$$\min_{p \in P(s,d)} hop(p)$$
$$del(p) \leq \Delta_{del}, r(p) \geq \Delta_r$$

in which, $P(s,d)$ is the set of feasible paths from a source node s to a destination node d, Δ_{del} is the maximum end to end delay to be guaranteed to the application flow, Δ_r is the minimum data rate to be assured to the flow, $del(p)$ is the estimated delay of the path p and $r(p)$ is the data rate estimated along the path p. The main issue of QOLSR is that such optimization problem is computationally expensive. This is usually solved through approximation, finding sub-optimal solutions. Another approach which considers QoS metrics is the OLSR with the Expected Transmission Rate (ETX) path metric (see, for instance, [31]). This simple routing path metric favours high-capacity and reliable links. The ETX is computed on the basis of the fraction of successful packets that are received from a neighbour within a window period. The path goodness takes into account, therefore, the reliability of all the links of the path. As far as energy awareness is concerned, several extensions have been proposed. [32] reports an extension to OLSR protocol with the aim of selecting an appropriate set of MPRs, taking into account the link transmission power. In [33] the authors proposed an energy-efficient routing algorithm (Energy Efficient OLSR, EE-OLSR) based on a heuristic to set the willingness value of OLSR. Each node decides to attribute a low, a default or a high willingness variable depending on the node's residual energy and predicted lifetime. [34] extends this last approach by proposing an adaptive energy-aware routing protocol for MANETs using fuzzy logic for adjusting such willingness parameter. In [35] OLSR is modified to select the paths according to the residual energy level of intermediate nodes. In particular, the node energy consumption is predicted using ARIMA model. [36] proposed an energy-aware clustering for the OLSR protocol, based on the residual energy of nodes. A set of criteria for the cluster head election has been provided and simulations shown the increase of performances, at different nodes' speed (it results to be useless in our context, since nodes are considered fixed). In [37] the authors proposed an energy-aware multipath routing algorithm based on OLSR. This algorithm, named MBA-OLSR, assigns to each link a cost calculated on the basis of the residual battery of both nodes of the link. The residual battery of each node is contained into the OLSR Hello and TC messages. The higher the residual battery of a link, the lower is the cost associated to the link. Results have shown that network lifetime is increased compared with no energy aware multipath routing approaches.

In conclusion, it has been assessed ([38]) that the scenario in which OLSR performs well is a dense network: in this kind of scenario, a low number of MPRs is able to reach a high number of nodes, minimizing the control overhead needed for the flooding of TC messages. In this case the algorithm is also scalable. The main shortcoming of this approach is that, in sparse or regular

networks, the overhead is inacceptable (due to the data flooding) and OLSR does not scale to large networks. Moreover, the link state approach needs to maintain a data structure modelling the whole network topology and to compute computationally expensive optimization algorithms to compute the best paths. This constrain the nodes to be equipped with potentially high processing and memory resources, not compatible with the memory and processing constrained nodes such as the considered SWIPE nodes.

3.2 AODV and its extensions

Differently from the OLSR, the Ad-hoc Distance Vector (AODV) routing protocol [24] is a reactive protocol based on the concept of distance vector. This protocol belongs to the reactive family, thus, the routing tables are updated on-demand, i.e., when a flow requires to be routed within the network. AODV foresees three different control messages: *Route Request* (RREQ), *Route Reply* (RREP) and *Route Error* (RERR). When a source node wishes to communicate to a destination node, it starts a route search throughout the network by flooding a RREQ control packet. The path is computed when a RREQ message reaches the destination (or, alternatively, an intermediate node knows a "recent" path to reach the destination) and a RREP control message is delivered back to the source. When passing intermediate nodes, they update the routing tables, accordingly. In order to verify that a path is "stale", a comparison between *Destination Sequence Numbers* (DSNs) is carried out. DSNs are used also to avoid the formation of loops, due to link failures.

The standard AODV algorithm has been largely studied and extended in literature ([39], [40], [41]). Also when considering AODV protocol, QoS has been primarily introduced into the algorithm to take into account flow quality measurements and requirements. An example is the Radio-Metric AODV (RM-AODV) [25]. In this algorithm, the RREQ packet contains information on the QoS required by the application flow (for instance, minimal data rate, maximum delay, etc.). Moreover, in the RREQ message is inserted a value of accumulated metric (for instance the estimated delay of the path). Among all the RREQ packets reaching the destination (which surely satisfy the QoS constraints), the destination node chooses the path with the lowest accumulated metric (e.g., the lowest delay). [42] proposed an extension of AODV which aimed at re-routing around nodes low on battery power as far as possible, in order to prolong the network lifetime. The nodes are classified according to three different regions: Normal zone, Warning zone and Danger zone. To each zone is associated a fixed cost which is considered in the distributed path computation. Regarding protocol modification, [42] introduced the concept of Route Warning (RWARN) packet (if the residual battery sensibly changes and switches to a different zone) which is sent back to the source to update the routes according to the new level of battery. [43] proposed an adaptive energy-aware routing protocol for MANETs using Reinforcement Learning (RL), which was built over AODV. The RL is used to make each node learn the best rate of RREQ forwarding, on the basis of its residual battery lifetime. Such rate reflects the willingness of a node to be part of routes towards the destinations. Reinforcement Learning and distance vector-like approaches are also used in [9]. Energy-aware and fault tolerant metrics were used by the authors and aggregated in a multiplicative manner into a RL framework.

In conclusion, AODV is quite good in networks in which the traffic is concentrated in a small sub-set of nodes and the mobility is low, principally as far as control overhead is concerned. In this

case, in fact, a low amount of RREQ and RERR are disseminated throughout the network, since few nodes require to transmit data and few paths fail. For this reason, the AODV protocol (and its extensions) is not directly applicable in WSNs in which all nodes generate traffic towards the data sink(s) (traffic is not concentrated).

3.3 The Hybrid Wireless Mesh Protocol (HWMP)

The Hybrid Wireless Mesh Protocol (HWMP) [25] is a hybrid routing protocol for mesh networks which combines the reactive (on-demand) routing with the proactive maintenance of a tree, based on distance vector. A tree is proactively maintained to guarantee the communication with a single important node, named the *root* (for instance a gateway in a sub-network). This is particularly useful when the majority of the traffic is directed outside the sub-network, for instance to the internet. As explained in [25], there are two methods for the proactive tree building and maintenance. The first method uses a proactive *Route Request* (RREQ) message (as happens in AODV) sent by the root with destination set to a broadcast address. Using this flooding the routes between the root and all MPs in the network are periodically setup. The second method uses a *Root Announcement* (*RANN*) message which is flood throughout the network to announce to nodes the presence of the root. The actual routes to the root are built on-demand by the nodes by sending, in unicast, specific RREQ control message to the root. This method leads to a higher amount of overhead, since there is the need of a RREQ for each node of the network. As far as internal communications are concerned, the HWMP envisages the use of the RM-AODV (extension of AODV considering Quality of Service requirements, as illustrated in the previous Section) on-demand protocol.

It is clearly understandable that a tree-based approach could be very interesting in applications (e.g., see [46]) in which the majority of traffic is constantly directed to a specific node or a set of specific nodes (the on-demand feature is not suitable to connect nodes with the root) and there is not the need of maintaining routing information to allow communication between every couple of network nodes (as happens in link state approach such as OLSR), which require a lot of memory and computation resources in the nodes. Due to its hierarchical nature (with one or more roots representing the data sinks), the considered environmental monitoring scenario belongs to such category of applications. Concerning the communications with the data sinks, a proactive tree-based approach could be applied, enforced with reactive features to rapidly re-configure the tree(s) in case of node failure. For occasional communications among nodes, an on-demand approach is envisaged.

4 Proposed approach

As investigated in the previous Section, the considered planetary exploration scenario does not allow to directly utilize OLSR or AODV algorithms (and their extensions). In the first case, the link state approach needs to maintain a data structure modelling the whole network topology and to compute computationally expensive optimization algorithms to compute the best paths. This constrains the nodes to be equipped with potentially high processing and memory resources, not compatible with the memory and processing constrained nodes such as the considered low-mass and low-volume nodes. In the second case, the distribution of the application flows over all the

network nodes makes the on-demand approach inapplicable, generating too much unneeded overhead, even over a reduced topology structured in a clustered fashion.

The proposed routing approach extends the tree-based feature of the HWMP protocol, as follows:

- *Multiple distance vector trees* are setup and dynamically maintained. Each data sink is a root of a distinct tree. This tree multiplicity, gives the nodes the possibility of rapidly switching from a data sink to another data sink in case of a neighbour node failure (*fault tolerance*) and to better distribute the traffic load among data sinks (*any-sink approach*).
- A combined *energy-aware and distance-based link cost* is introduced to maximize the minimum node lifetime and the network disconnection time. The cost of the link must be judiciously designed to choose the most charged paths but, at the same time, to avoid that paths long in distance are chosen (even if the nodes involved in the path are really charged) because the more nodes are involved in one packet transmission the higher is the energy consumed. This could generate an opposite effect which could reduce the network lifetime.

4.1 Setup and maintenance of multiple distance vector trees for any-sink communication

The physical network can be modelled as a graph $G(V, E)$ in which V is the set of nodes and E is the set of links of the network. Over such physical network, a clustered virtual topology may be built, especially to handle large scale networks [44]. The network, thus, can be structured in a set of clusters (each of which is led by a special node called Cluster Head – CH), connected through a virtual backbone, which can be viewed as a sub-graph of the original one. Since such sub-graph has not a star or a centralized structure, a routing algorithm is needed to allow communication between any couple of network nodes. In case of large scale networks, routing should be performed directly over the virtual backbone (in small networks it is not needed to setup a clustered topology). This paper does not focus on the problem of setting up the clustered topology which is assumed to be given.

Due to the nature of the considered application scenario (i.e., periodic scientific and housekeeping delivery to specific network nodes named data sinks), it is not needed that paths connecting every couple of nodes are constantly kept available to the application. A specific subset of nodes is considered crucial for the considered application scenario: those nodes are the data sinks. Another subset of nodes which are crucial to report data back to the Earth are the exit points. The communication with those nodes is, however, less frequent and the routing approach governing their connectivity could be an on-demand approach such as the one used in AODV. A tree-based approach is used in to connect each node to a data sink. Multiple trees are setup to guarantee the connection of each node to the multiple data sinks available in the WSN. The following figure shows an example of such concept.

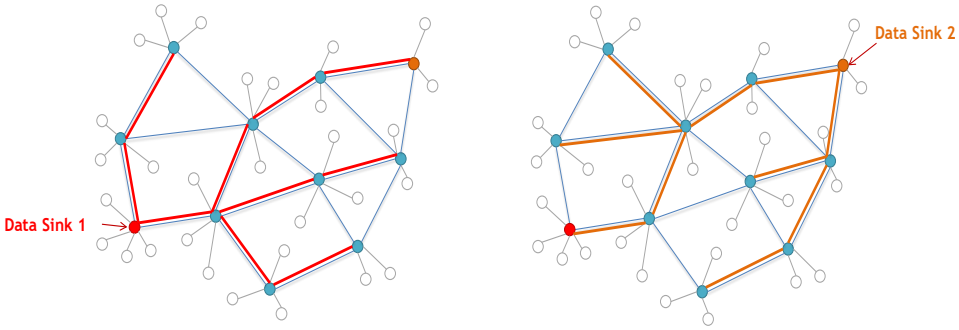
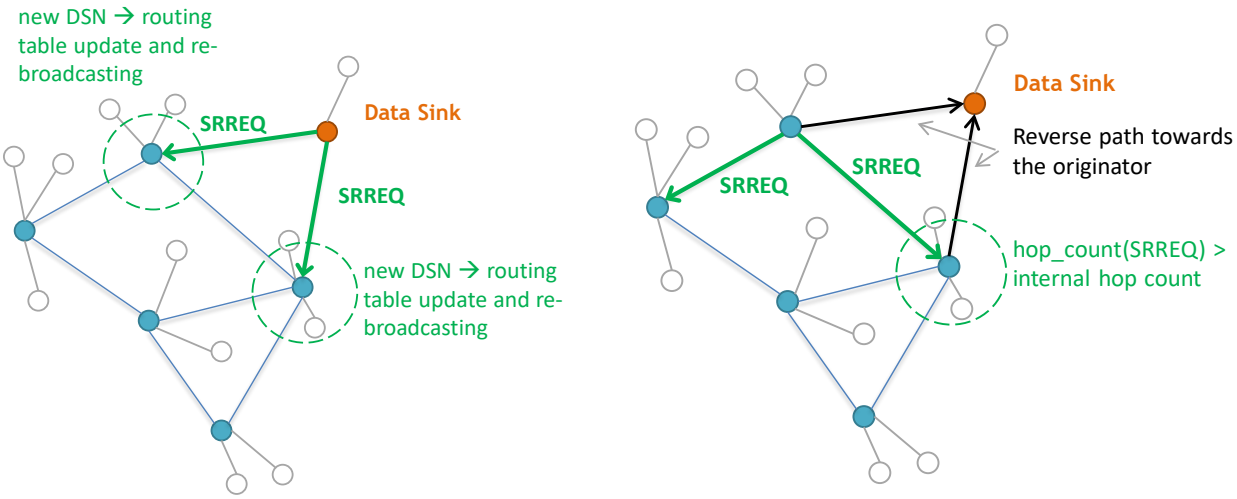


Figure 4 Multiple trees formation over a clustered topology.

One main control packet is foreseen to allow the setup of one of the trees. This control packet is called *Sink Route Request* (SRREQ). Periodically, a data sink floods a SRREQ throughout the WSN, indicating its presence. Once a node received a SRREQ from a neighbour, setups a *reverse path* towards the originator node (i.e., the data sink), passing through the neighbour. The reverse path is based on the distance vector approach: this means that, for a specific data sink, only the indication of the next-hop as well as the distance to reach the sink must be saved in the internal memory of the node. The reverse path is updated only if the hop count of the path is less than the current one or if the SRREQ contains a higher Destination Sequence Number). The Destination Sequence Number is used as a logical timestamp to force the route to update (e.g., in case of proactive re-construction or response to a node/link failure, as will be clear hereafter). The following figure shows the behaviour of the tree formation:



(a) (b)

Figure 5 SRREQ flooding and reverse path formation.

The SRREQ generated by the data sink is received by its neighbours. Since the SRREQ contains a new Destination Sequence Number (DSN), then the neighbours are forced to update the routing table and re-broadcast the SRREQ with an incremented (by one) value of the hop count. This

process is shown in Figure 5(a). If a node received for the second time a SRREQ (with the same DSN value), it updates its internal routing table and re-broadcasts the SRREQ with the updated value of the hop count, only if the hop count in the SRREQ incremented by one is lesser than the current one saved into the routing table. Thus, in the example of Figure 5(b), the circled node does not update the routing table and does not forward an updated SRREQ (the internal hop count is 1 but the one tagged into the SRREQ incremented by one is 2). Another point to highlight is that there is not the need, for nodes receiving a SRREQ, to send back to the originator node a reply (e.g., such as the RREP in the AODV protocol), since the standard communication needed to achieve the reporting of scientific/housekeeping data to the data sink(s) is from the nodes to the data sink(s) (and not vice versa). This is another feature that leads to an overhead reduction compared to the AODV approach, very important feature in the considered WSN.

The setup of the trees based on distance vector allows to guarantee a communication among data sinks and regular nodes. Moreover, it is important to point out that the tree multiplicity (one per sink) gives the nodes the possibility of rapidly switching from a data sink to *another data sink* in case of a neighbour node failure. This feature increases the *fault tolerance* of the algorithm, avoiding that a sudden node failure produces a loss of important measured information from a subset of the network nodes. If there is not any path that allows to reach any of the data sinks, it is needed to guarantee that the network is *reconfigurable* in case of node/link failures. In this case, the trees must be re-built in case sudden events verify and could affect the connectivity of the whole network. If the routing table does not modify, according to the failed nodes/links, then loops can be generated and/or a high amount of data packets can be lost (very bad consequence, especially considering that nodes cannot be manually reconfigured by operators, due to the extremely remote location). The protocol proposed in this paper foresees the introduction of a second control packet: the control packet is named *Route to Sink Error* (RSERR). A RSERR is sent in flooding as soon as a node becomes aware that a neighbour node, that permitted (in a multi-hop fashion) to reach a specific data sink, is no longer available. The RSERR includes, in its fields, the indication of the data sink currently no longer reachable. As the data sink receives the RSERR from a node, it will re-start a process for the setup of the tree for which it is the root. To guarantee the correct update of the routes inside the network nodes, the data sink increments by one the Destination Sequence Number and tags it into the new SRREQ. In the tree discovery process, therefore, nodes are forced to update their routing table and a new tree is built up and the network is reconfigured. As far as the time needed for the network reconfiguration is concerned, an evaluation through simulations of the latency for the reconfiguration is reported in Section 5.4, especially for large scale clustered networks. The details about the control packets as well as the pseudo-code of all the routing procedures are reported in Section 4.3.

4.2 Combined energy-aware and distance-based link cost

The routing process described in the previous Section is demonstrated (e.g., by the authors of [24] or [25]) to converge to a tree in which each node is connected to the root through the shortest path (i.e., the path with the minimum number of hops). The main issue of such approach is that the all the traffic generated by the regular nodes flows along fixed trees: it means that those “unlucky” nodes will discharge earlier than the other nodes. This implies that precious information will not be available at the data sinks (for instance no measurements from specific regions of the sensor field

will be reported to the data sinks and, later, to Earth). On the light of the above consideration, there is the need to maximize the duration of the nodes, especially for those that have an active role in the forwarding of data generated by other nodes. There are several possible ways to achieve this objective, including the following ones: (i) by improving the efficiency of the node in terms of power consumption (for instance designing energy-efficient internal device components, such the radio equipment, the CPU, etc.); (ii) by aggregating the payload of the data packets during their traveling throughout the tree. This can be achieved through proper data aggregation techniques; (iii) by *balancing* the data forwarding effort among different dynamic trees, in order to uniformly distribute such load over the network; (iv) by choosing nodes in the path which are "*close" in terms of distance* each other. If the radio equipment is capable to tune the transmission power according to the distance of the destination neighbour node, a large amount of energy can be saved in this fashion. The proposed routing algorithm described in this Section focuses on the points (iii) and (iv) of the previous list.

As far as energy balancing is concerned, one approach to achieve that objective could be to periodically switch to a new tree which excludes the nodes that, in the previous time interval, belonged to the tree. This switching could be done in a round robin fashion. This approach, however, is affected by two main issues: (i) there is the need to compute and maintain the set of trees to be alternated during the time, which could be onerous in terms of computation and memory; (ii) it does not take into account the fact that the batteries of the nodes could have different levels, both due to a heterogeneity of the batteries and, in particular, because they have the capability of being recharged thanks to the solar energy. This recharge is usually not uniform among all the nodes (for instance because the nodes are deployed in different regions of the remote surface, the ground differently absorbs/reflect the solar energy, etc.). Taking into account what said, the proposed approach considers the introduction of the *residual battery* into the paths computation process. The idea is to periodically (with a period of T_{TREE} seconds) invoke the generation of the new trees according to a path cost different from the hop count. The main concept is the following. At each link (i, j) is assigned a *cost* $c_{ij}(t)$, variable during time, that is the higher the lower is the residual battery $e_j(t)$ (in percentage with respect to the full battery level) of the destination node j of the link. The metric shall be able to balance between the shortest path (short and less nodes involved in the data forwarding) and the most charged path (nodes very charged but more nodes involved in the data forwarding). In this paper we propose the following link cost:

$$c_{ij}(t) = 1 + \log^2 e_j\,(t) \tag{1}$$

The term $+1$ is introduced to give a unit cost for the fact that a new hop is introduced in the path. The term $+\log^2 e_j\,(t)$ is the higher the lower is the residual battery of node j. This term assumes high values only when the residual battery is sensibly low: in this case longer paths will be used. The behaviour of this metric in function of the residual battery $e_j(t)$ is depicted in Figure 6.

Another important aspect when talking about energy consumption is that the energy spent in transmission strictly depends on the distance of the next-hop node in the route (at least it depends on the square, as the first order radio model we use in the simulations). Thus, if the radio equipment has the capability to modulate the transmission power in function of the distance with the next-hop, a lot of energy can be saved. In this respect, it could be very intelligent to choose the path in which

the distances between couples of nodes are minimized. To do so, we propose to substitute the factor +1 with a function of the distance between the nodes i and j of the link (i,j). In this case, we propose a quadratic function as follows:

$$c_{ij}(t) = k_d \left(\frac{d_{ij}}{r}\right)^2 + k_e \log^2 e_j\,(t) \tag{2}$$

where d_{ij} is the distance (in meters) between node i and node j, r is the maximum transmission range and k_d, k_e are constant identifying the weight of each term in the cost. This quadratic function has sense since the energy consumption is at least quadratic with respect to the distance (for example the first order energy model used in the simulations of this paper, as we will define later). If all the nodes are at the same distance (or if the transmission power is fixed and cannot dynamically be modulated), thus, the cost is equal for any link the metric does not give any contribution. Both the metrics are illustrated in Figure 6 ($k_d = 5$, $k_e = 1$). Until the destination node in the link is sufficiently charged, the distance-based cost assumes a relevance. As soon as the node starts to discharge (in the figure example when the battery is the 40% of its full level), the energy-based term starts to be preponderant. This behaviour is good since it is not convenient to balance the relaying effort too early, since longer (with charged nodes) paths could be chosen leading to an energy waste.

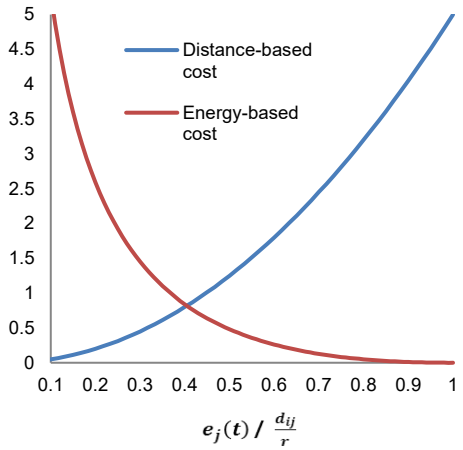


Figure 6 Link cost terms behaviour.

To be applicable, obviously, a node should be capable to compute the approximated distance with its neighbours using ranging techniques. In the simulations, for simplicity, we will give equal weight to the two cost terms (i.e., $k_d = k_e = 1$).

4.3 Algorithm details and pseudo-code

This Section reports the low level algorithm details (including the control messages data structures) and the pseudo-code of all the triggered routing procedures. The nodes identifier length is set to 16 bits, allowing to address tens of thousands of nodes (a 32 bit address would increase too much the control overhead).

4.3.1 Description of control packets for routing

Control packets for routing are detailed in Table 2.

4.3.2 Routing data structures

Routing data structures are detailed in Table 3.

4.3.3 Routing events

The relevant events in the routing process are detailed in Table 4.

In addition to the presented pseudo-code, the "*Neighbour n unreachable*" event is triggered when no Hello messages are received by a neighbour for a certain amount of time. Moreover, the list_of_waiting_RSERRs data structure is reset after a timeout to avoid that a loss of a RSERR packet could affect the network connectivity.

4.4 Memory and computational complexity

This section reports an analysis of the required memory and the computational complexity of the proposed routing algorithm, which is also important for the selection of suitable hardware for the node (e.g., choice of the memory banks, CPU, storage, etc.). The following subsections report:

- The size of the control packets exchanged for the trees maintenance (in bits). Those figures are useful for the quantification of the amount of memory required to store and analyse the packets.
- The size of the internal data structures (routing table, etc.) to be stored in the memory locations.
- The complexity of the various procedures of routes maintenance and route selection.

4.4.1 Size of control packets and routing internal data structures

The size of the control packets and of the routing internal data structures is reported, respectively, in Table 5 and Table 6.

n_{sink} is the number of data sinks in the network and d_{MAX} is the maximum degree of the network graph. According to the table, the total size of the internal routing data structures is determined to be not more than $178n_{sink} + 24d_{MAX} + 72$. For instance if the $d_{MAX} = 8$ and $n_{sink} = 4$, it is required an amount of memory for internal routing data structures of 976 bits. It is remarkable to highlight that the size of the internal routing data structures does not depend on the total number of nodes, but only on the maximum degree of the network which is considerably smaller. This allows to achieve scalability if the number of nodes in the network becomes very high.

4.4.2 Complexity of routing procedures

A characterization of the routing algorithm in terms of number of executions (in the worst case) of the routing procedures and basic operations (read, write, transmission, sum, etc.) is reported in Table 7 and Table 8.

Similarly to what stated for the memory, also the computational complexity does not depend on the number of nodes but only (in a linear fashion) on the maximum network degree d_{MAX} and on the number of data sinks n_{sink}, achieving scalability with an increasing number of nodes.

5 Simulations

This Section reports the simulations proving the higher performances of the proposed any-sink energy-aware routing algorithm. The algorithm has been implemented in MATLAB ([18]) in an event-based fashion. This means that all the routing procedures, detailed in the previous section, have been implemented and the sending/receiving of packets simulated accordingly. Thus, the convergence and loop freedom of the algorithm have been verified. This Section is organized as follows. Section 5.1 illustrates the network and energy models used in the simulations to prove the feasibility and the performances of the proposed routing algorithm. Section 5.2 proved the capability of a balanced and efficient use of the battery capabilities of the nodes. Grid, clustered and uniform random network have been simulated. Section 5.3 provides an analysis of the performance both if the radio equipment is able or not to modulate the power in function of the distance with the next-hop node. It will be shown that modulating the power provides higher performances. Nevertheless, the proposed approach is also applicable in presence of fixed transmission ranges (the distance-based term of the cost link is useless). Section 5.4 provides an analysis of the time needed for the network re-configuration. Section 5.5 provides an analysis of the control overhead, to evaluate if it is modest and, thus, makes the proposed approach applicable in energy-constrained networks such the considered WSN.

The following common parameters are used in all the simulations of the any-sink energy-aware routing algorithm. Nodes periodically (every 600s) send to the data sinks packets with payload size of 692bits (compliant to what proposed in the SWIPE project). Those packets include housekeeping, scientific data without any data fusion applied, data packaging and miscellaneous. The data sinks are supposed fixed and placed in a uniform spatial fashion in the sensor field. This differs from approaches in which data sinks are mobile (e.g., see [47]), although it is applicable also in this scenario (as a data sink migrates, a new tree must be rapidly instantiated). The tree reconfiguration timeout is set to 2 hours for all the data sinks. The consistency (when enabled) among data sinks is guaranteed through the exchange of the stored data (properly fused and grouped in packets of 1500 bytes) every 30 minutes. We hypothesize that each data sink has a specific data fusion ratio, applied to the collected data, that is defined as the size of the original data over the size of the fused data. A data fusion ratio of 3 means that the fused data is three times less than the original one. The routing control packets have the sizes defined in Table 2. A network header of 128 bits is added to all data and control packets. No data aggregation is performed in the intermediate nodes of the paths. The data rate of the wireless links is set to 1Mbps. In this simulation section, the data sinks (besides the exit point) are equipped with larger battery capacities, than the regular nodes, to ease the analysis of the proposed routing approach. Nodes are considered as discharged if their residual battery is under the 1% of their initial battery capacity. Finally, every NLoS period (8449s), the exit point queries the data sinks to deliver the stored data to it, in order to be transmitted to the satellite.

Three main parameters are evaluated: (i) the *minimum node lifetime* (expressed in days), (ii) the *network disconnection time* (expressed in days) and (iii) the *total amount of information delivered to the data sink(s)* (expressed in MB). Moreover, we will measure the *control overhead* (in terms of required bits per seconds) and the *reconfiguration time*, in milliseconds, to evaluate the impact of the routing algorithm in the network and its self-adaptation capability.

5.1 Simulation network and energy models

The simulations reported in this section are characterized by network sizes in line with the network topologies reported in Table 1. We simulate grid, random and clustered topologies. As far as clustered topologies are concerned, Figure 7 shows the three clustered topologies used to test the routing and data aggregation algorithms.

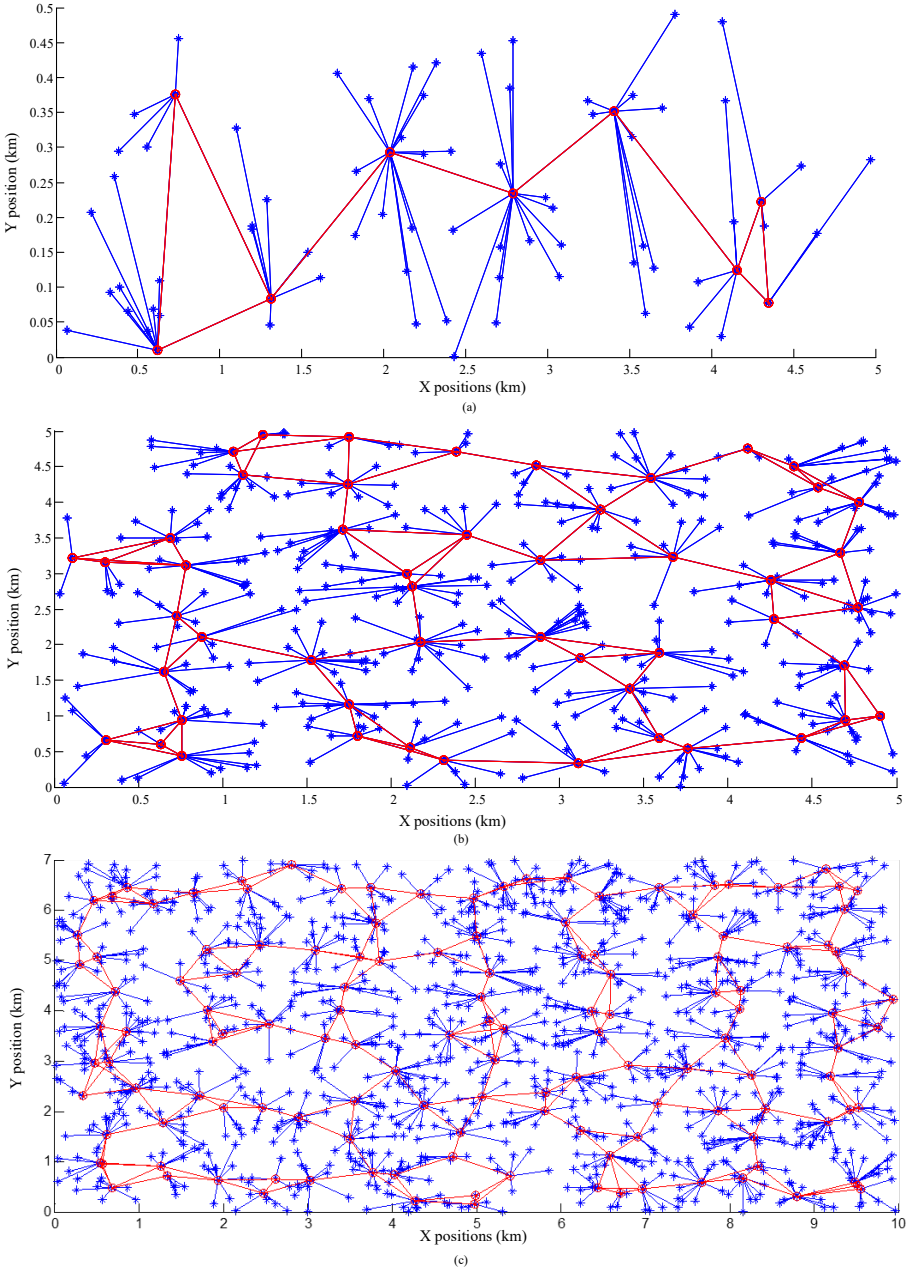


Figure 7 Clustered network topologies. (a) Minimal coverage; (b) Preferable coverage; (c) Extended coverage.

There are 80, 400, and 1200 nodes in each network topology respectively, and the maximum transmission radius is set to 0.8 km. In each graphic, the symbols '*' and 'o' denote the RNs and CHs respectively. The CHs are connected through a virtual backbone, which is highlighted in red

on the figure. The data sink node(s) and the exit point are selected among the network nodes as described in the various simulations sections (e.g., in a spatial uniform fashion).

A first order radio model (see [45]), is adopted in in the energy analysis, for simplicity. In this model, the radio dissipates $E_{elec} = 50$ *nJ/bit* to run the transmitter or receiver circuitry and $\varepsilon_{amp} = 100$ *pJ/bit/m*^{2} for the transmitter amplifier. The equations used to calculate transmission costs and receiving costs for a L-bit message and a distance d are shown below:

$$E_{TX(L,d)} = L \cdot E_{elec} + L \cdot \varepsilon_{amp} \cdot d^2 \quad (3)$$

$$E_{RX}(L) = L \cdot E_{elec} \quad (4)$$

5.2 Energy balancing and efficiency

5.2.1 Grid-based networks

In this Section, we simulate the three *low resiliency* and *accurate positioning* scenarios (see Table 1). Since the number of nodes is low, to cover all the sensor field it is convenient to deploy the nodes in a *grid* topology. The nodes are placed at 500m of distance in each row and column of the grid. The range determining the neighbourhood of each node is put to 600m. The simulations are done considering the presence of one sink in the centre of the grid and an exit point in the north of the network. No aggregation is performed at the intermediate nodes travelled by packets towards the data sink. The initial battery of all nodes (except for the data sink and the exit point, which are equipped with a higher capacity battery) is 2.5kJ. Figure 8 shows the trend of the minimum residual battery, expressed in kJ, among the nodes of the network, in relation to the preferable coverage scenario. Such scenario envisages a network of 100 nodes, placed over a sensor field of 5km × 5km (25km^2). A comparison between the shortest path algorithm and the proposed energy-aware routing algorithm has been performed. If the shortest path algorithm is used, we notice that the minimum residual battery decreases in a linear fashion with a high pendency. This happens because a subset of the nodes (the ones of the shortest paths) is always used until they discharge. The proposed energy-aware algorithm starts to balance the paths from the beginning: the pendency is lower in this case. Moreover, when the residual battery becomes very low, the strength of the proposed approach is higher: in fact, the $\log^2 e_j(t)$ term of the cost expression of equation (2) produces high cost values for the links with low-charged nodes, and, thus, they are more judiciously chosen. The pendency, thus, becomes very low and allows to make the first node be discharged almost in correspondence of the network disconnection time. Since all nodes are placed at the same distance (considering the range of 600m), the part of the metric taking into account the distances among nodes does not give contribution.

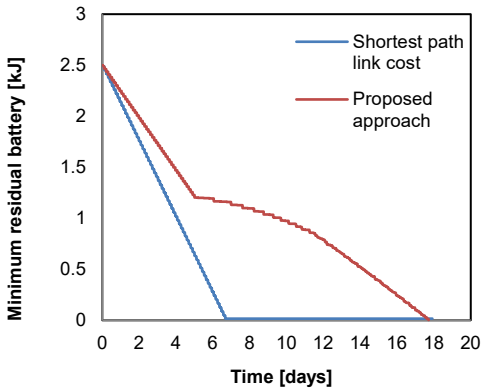


Figure 8 Min. residual battery (in kJ) of the preferable coverage scenario (low resiliency and accurate positioning).

The values of the network disconnection time and the minimum node lifetime are reported in Table 9. The proposed energy-aware algorithm outperforms the shortest path, in terms of minimum node lifetime, of about the 160%.

Table 10 reports the values of the evaluated parameters in the cases of the three low resiliency and accurate positioning scenarios. As before, a data sink is placed in the centre of the network and the exit point is placed in the most northern row in the centre. Using the proposed algorithm, we obtain an increase in the minimum node lifetime of about 90% in the minimal coverage scenario, and more than 160% in the other two scenarios.

We proceed now to evaluate what is the behaviour of the algorithm in presence of 4 data sinks uniformly distributed over the sensor field. We suppose that consistency must be guaranteed among the 4 data sinks, and use data fusion ratios of 1 (no data fusion in the data sinks) and of 2. No data aggregation is foreseen in intermediate nodes during the paths from the sources to the data sinks. Table 11 reports the corresponding minimum node lifetime values (expressed in days).

If we do not have fusion of data in the data sinks, in the minimal coverage scenario it is not convenient to deploy four data sinks since we obtain a lower minimum node lifetime (as well as the network disconnection time not shown in the table) than as in the one data sink case. In the other two scenarios, even without data fusion the minimum node lifetime is increased: this is due to the fact that with a quite small network diameter (which characterizes the low resiliency and accurate positioning scenarios), the data exchanged for consistency traverse short paths (in terms of number of hops) and few nodes are interested.

5.2.2 Clustered network

The previous Section has shown the energy balancing capabilities in the low resiliency and accurate positioning scenarios. In those scenarios, especially the minimal coverage one, two close failures could affect the network connectivity. In order to extend the tolerance to node failures and increase the resiliency of the network, high resiliency scenarios have been conceived in the project, as reported in Table 1. Moreover, the increased number of nodes can also increase the robustness in case of erratic positioning of network nodes. Since the density and the scale of the network are increased, it is convenient [44] to organize the network in a *clustered* form, through the setup of a

virtual backbone). The formation of the virtual backbone is not the scope of this paper. Hence, in this paper, we evaluate the energy balancing features of the any-sink energy-aware routing algorithm over a clustered topology, in the preferable scenario, characterized by high resiliency and erratic positioning of nodes. The initial battery of each node is set to 12kJ and the transmission range of the radio equipment is set to 800m. Four data sinks are deployed in a uniform way over the sensor field and the exit point is situated in the north of the network. No data fusion is performed in the data sinks.

We consider the clustered topology provided in Figure 7. The results of the simulations carried out on this clustered scenario are illustrated in the histograms of Figure 9. As expected, the shortest path algorithm is the worst, in terms of all the three evaluated parameters. In particular it fails in providing sufficient node duration with the considered batteries (the first node discharges in less than 7 days). Very better results are obtained introducing the cost link which is function of residual battery of the nodes. In this case, the minimum node lifetime, the disconnection time and the received data are sensibly increased. The best results are given by the aggregated cost function (which takes also the distances between couples of nodes in consideration). It is worth highlighting that, with the proposed approach, the first node discharge in about 16 earth-days, which is higher than the duration of the night on the Moon on equator (about 14.25 earth-days). During the day, in fact, nodes are capable to recharge and the energy becomes a more available resource.

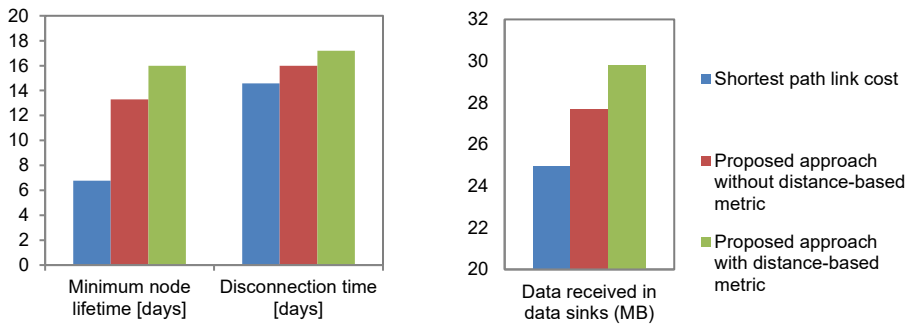


Figure 9 Simulation comparison in the preferable clustered scenario (high resiliency and erratic positioning of nodes).

5.2.3 Uniform random networks

In this Section, the behaviour of the proposed routing algorithm is evaluated in *uniform random* networks. $n = 200$ nodes are placed randomly (with a uniform distribution) in a squared scenario of 25km^2 (analogous to the preferable coverage scenario with high resiliency and accurate nodes positioning). Four data sinks are uniformly deployed in the network and the exit point is situated in the north of the scenario. We evaluate the shortest path routing, the energy-aware routing without considering the distance of nodes (equation (1)) and the energy-aware routing with the aggregated metric (equation (2)). 10 simulation runs have been carried out and the results averaged over the simulation set. The initial battery of the nodes is set to 2.5kJ (except for the data sinks and the exit point). The results are illustrated in the histogram of the next figure.

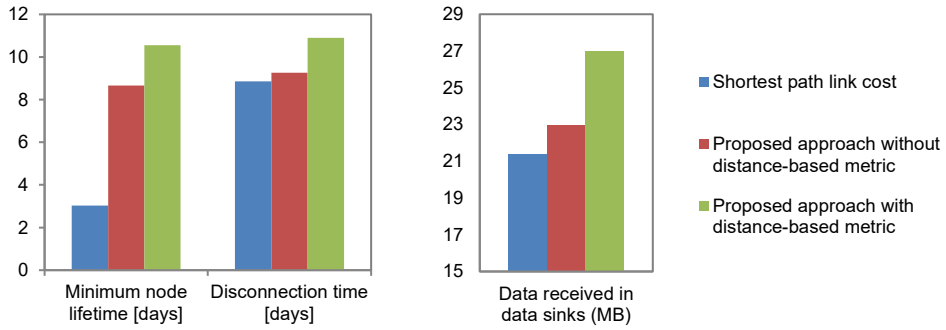


Figure 10 Simulation comparison in random networks.

The trend of the results of Figure 10 is quite similar to the ones presented in the previous Section, related to the clustered scenario and same considerations hold also in this case. The most important point to highlight is that the algorithm can handle any kind of topology (even completely random) and, thus, be adaptive if drastic changes of topology occur (e.g., several nodes fail because of external events derived from the harsh environment).

5.3 Fixed transmission power vs. variable transmission power

In the previous sections, we have shown how high are the potentialities of the proposed approach in scenarios in which the radio equipment of the nodes is capable to modulate the transmission power according to the distance to be covered (i.e., the distance to reach the next-hop in the path towards the destination). In case the nodes are not capable to modulate the transmission power, it is immediately clear that the results in terms of energy efficiency and balancing cannot be as good as the ones presented so far. To demonstrate this fact, we have simulated random networks exactly as the ones reported in the previous Section. The results are shown in the next histogram.

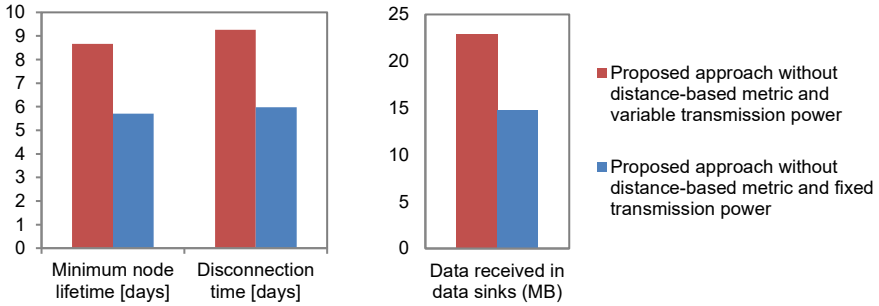


Figure 11 Simulation comparison with fixed and variable transmission power in random networks.

All the three evaluated parameters result to be increased if the radio equipment is capable to modulate the transmission power according to the distance from the next-hop node. Thus, modulating the power allows to save a lot of energy in comparison with a fixed transmission power. In terms of percentages, the three parameters increase of more than the 50%.

5.4 Analysis of the time for the network re-configuration

This Section reports an analysis of the time needed for the network re-configuration. For each link, we compute the transmission delay component of the total link delay. We neglect the

propagation delay, since it is negligible (the propagation delay is generally of the order of the microseconds whereas the transmission delay is of the order of the milliseconds). The bandwidth of all links is put to 1Mbps. Randomly, a node of the network fails and we compute the time the network takes to reconfigure, i.e., from the time of the node failure to the termination of the SRREQs flooding. The three high resiliency and erratic positioning of nodes scenarios are considered in this Section. A run of 10 simulations is carried out for each combination scenario-number of data sinks. Results are depicted in Figure 12.

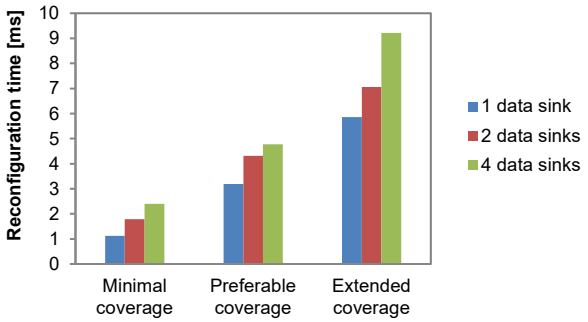


Figure 12 Reconfiguration time analysis in the three scenarios with high resiliency and erratic nodes positioning. The networks are clustered.

As expected, the reconfiguration time increases with the scale of the network and also with the number of data sinks deployed in the network (in fact, more trees must be maintained and re-setup). The reconfiguration time figures are very small (less than 10ms).

5.5 Analysis of the control overhead

This Section reports an analysis of the control overhead of the proposed routing algorithm, in terms of bps, considering a tree reconfiguration timeout of 2 hours for all the data sinks. The three high resiliency and erratic positioning of nodes scenarios are considered in this Section. The trees are re-established periodically with a period of 2 hours. The overhead results (in terms of bps) are reported in Figure 13, considering 1, 2 or 4 data sinks uniformly distributed in the network.

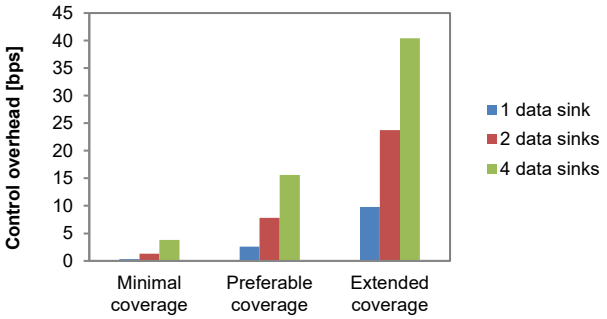


Figure 13 Control overhead analysis in the three scenarios with high resiliency and erratic nodes positioning. The networks are clustered.

As highlighted in the histogram, the control overhead of the proposed routing approach is very low and makes it applicable in energy-constrained networks such as the considered WSN. In fact,

this approach allows neither wasting the precious available bandwidth nor affecting the energy performances (as also proved in the previous sections).

6 Conclusions

This paper presented an ad-hoc routing algorithm applicable to WSNs for planetary exploration (such as the one of the SWIPE project), which is able to assure any-cast communication with multiple data sinks, deployed for redundancy purposes, to minimize the control overhead for the maintenance of the routing paths, to be light in terms of memory and computational requirements, in order to be installed into low-power and low-memory/processing devices, to be robust and rapid to reconfigure in presence of node failures and to optimize the choice of the routes, to achieve energy balancing and saving. A multi-sink proactive tree-based approach has been selected, together with a combined link cost, depending on the residual battery of the nodes and their mutual distances. Extensive simulations have demonstrated the effectiveness and the efficiency of the proposed approach, in terms of network lifetime, overhead and reconfiguration time.

The proposed approach suggests interesting directions for further work. One idea could be to explicitly take into account possible interferences in the packets transmissions, due to the harsh environment, for instance including them into the link cost. Moreover, a simultaneous clustering and routing algorithm could be designed, to increase the overall network performances, also in terms of network throughput.

7 Acknowledgment

The authors wish to thank Pedro Rodrigues, Francisco Alvarez, Tanya Vladimirova, Xiaojun Zhai, Micheal Crosnier and all the team of the SWIPE project for their valuable contribution to the work presented in this paper.

8 References

[1] EU FP7-SPA SWIPE project, Grant Agreement n° 312826, web site: http://swipe.tekever.com, accessed on November 2014.

[2] Heinzelman, W, Chandrakasan A. and Balakrishnan H., “Energy-efficient communication protocol for wireless microsensor networks”, Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000.

[3] Lindsey S., Raghavendra C., “PEGASIS: Power-efficient gathering in sensor information systems”, IEEE Aerospace conference proceedings, Vol. 3, 2002.

[4] Xu Y., Heidemann J. and Estrin D., “Geography informed Energy Conservation for Ad-hoc Routing”, Proc. 7th Annual ACM/IEEE Int’l. Conf. Mobile Comp. and Net., pp. 70–84, 2001.

[5] Boyan J. A., Littman M. L., (1994). “Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach”. In Advances in Neural Information Processing Systems, Vol. 6, pp. 671-678.

[6] Bruni C., Delli Priscoli F., Koch G., Pietrabissa A., Pimpinella L., "Network Decomposition and Optimal Multipath Routing Control Problem for Load Balancing", Transactions on Emerging Telecommunications Technologies (ETT), John Wiley & Sons, Inc., USA, Vol. 24, Issue: 2, March 2013, pp. 154-165, doi: 10.1002/ett.2536, 2012.

[7] Conforto P., Delli Priscoli F., "Routing and Dynamic Resource Assignment Joint Game: A Non-cooperative Model for QoS Routing", International Journal of Control, Taylor & Francis (London), Vol. 77, No. 16, November 2004, pp. 1408-1425.

[8] Conforto P., Delli Priscoli F., Facchinei F., "Existence and Uniqueness of the Nash Equilibrium in the Non-Cooperative QoS Routing", International Journal of Control, Taylor & Francis (London), Vol. 83, N. 4,, April 2010, pp. 776-788.

[9] Macone D., Oddi G., Pietrabissa A., "MQ-Routing: Mobility-, GPS- and energy-aware routing protocol in MANETs for disaster relief scenarios, Ad Hoc Networks", Volume 11, Issue 3, May 2013, Pages 861-878, ISSN 1570-8705, http://dx.doi.org/10.1016/j.adhoc.2012.09.008.

[10] Oddi G., Macone D., Pietrabissa A., Liberati F., "A proactive link-failure resilient routing protocol for MANETs based on reinforcement learning", 20th Mediterranean Conference on Control & Automation (MED) 2012, 3-6 July 2012, doi: 10.1109/MED.2012.6265812.

[11] Hinds A. et al, "A Review of Routing Protocols for Mobile Ad-Hoc NETworks (MANET)", International Journal of Information and Education Technology 3.1, 2013.

[12] Oddi G., Pietrabissa A., "A distributed multi-path algorithm for wireless ad-hoc networks based on Wardrop routing," 21st Mediterranean Conference on Control & Automation (MED), 2013 , doi: 10.1109/MED.2013.6608833.

[13] Macone D., Oddi G., Palo A., Suraci V., "A Dynamic Load Balancing Algorithm for Quality of Service and Mobility Management in Next Generation Home Networks", Telecommun Syst (Springer), 2013, DOI 10.1007/s11235-013-9697-y.

[14] Oddi G., Pietrabissa A., Delli Priscoli F., Suraci V., "A decentralized load balancing algorithm for heterogeneous wireless access networks", accepted for publication to the World Telecommunication Congress (WTC) 2014.

[15] Hood L. et al., "Initial mapping and interpretation of lunar crustal magnetic anomalies using Lunar Prospector magnetometer data", Journal of Geophysical Research: Planets, vol.106, no.E11, pp.27825–27839, 2001.

[16] Halekas J., Mitchell D., Lin R., "Mapping of crustal magnetic anomalies on the lunar near side by the Lunar Prospector electron reflectometer", Journal of Geophysical Research: Planets (1991–2012), vol.106, no.E11, pp.27841-27852, 2001.

[17] Rodrigues P., Oliveira A., Mendes R., Cunha S., Alvarez F., Crosnier M. et al., "Wireless sensor networks for moon exploration," presented at the 64th International Astronautical Congress, Beijing, China, 2013.

[18] "MATLAB R2008a", The MathWorks Inc., Natick, MA, 2008.

[19] Crosnier M. et al. "D2.1 – Mission Design Report". European Commission Seventh Framework Programme - Space Theme, 2013, available on http://swipe.tekever.com, accessed on November 2014.

[20] Rodrigues P. et al. "D2.2 – System Requirements Document". European Commission Seventh Framework Programme - Space Theme, 2013, available on http://swipe.tekever.com, accessed on November 2014.

[21] Rodrigues, P.; Oliveira, A.; Alvarez, F.; Cabas, R.; Oddi, G.; Liberati, F.; Vladimirova, T.; Zhai X.; Jing H.; Crosnier, M.. Space Wireless Sensor Networks for planetary exploration: Node and network architectures. Adaptive Hardware and Systems (AHS), 2014 NASA/ESA Conference on , vol., no., pp.180,187, 14-17 July 2014.

[22] El Rachkidy N., Guitton A. and Misson M (2014).Joint Sink and Route Selection for Anycast Communications in Wireless Sensor Networks. Available at the URL http://iconceptpress.com/download/paper/11092122164739.pdf , retrieved on October 2014.

[23] Clausen T., Jacquet P. Optimized Link State Routing Protocol (OLSR) – RFC 3626. October 2003.

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

[25] Joshi A. et al. HWMP specification. Available at the following URL https://mentor.ieee.org/802.11/public/06/11-06-1778-01-000s-hwmp-specification.doc, retrieved on October 2014.

[26] Dijkstra E. W. . A note on two problems in connexion with graphs. Numerische mathematik 1.1: 269-271, 1959.

[27] Kunz T. Energy-efficient variations of OLSR."IEEE International Wireless Communications and Mobile Computing Conference, IWCMC'08, 2008.

[28] Frikha M. Ad Hoc Networks: Routing, QoS and Optimization. John Wiley & Sons, 2013.

[29] Nguyen D. and Minet P. Scalability of the OLSR protocol with the Fish Eye extension. IEEE Sixth International Conference on Networking, 2007.

[30] Badis H and Al Agha K. Quality of Service for Ad hoc Optimized Link State Routing Protocol (QOLSR). draft-badis-manet-qolsr-04.txt. October 2006.

[31] Pandey A. and Baliyan M. Performance Analysis of OSLR and Modified Version of OLSR-ETX/MD/ML in Mesh Networks. International Journal of Computer Science and Communication Networks, Vol2 (2), 2012.

[32] Benslimane A., El Khoury R., El Azouzi R., Pierre S. Energy power aware routing in OLSR protocol. Proceedings of the First Mobile Computing and Wireless Communication International Conference (MCWC), 2006.

[33] De Rango F., Fotino M., and Marano S. EE-OLSR: Energy Efficient OLSR Routing Protocol for Mobile Ad Hoc Networks. Military Communication Conference (MILCOM), pp. 1-7, 2008.

[34] Chettibi S. and Chikhi S. FEA-OLSR: An Adaptive Energy Aware Routing Protocol for MANETs Using Zero-Order Sugeno Fuzzy System. IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 2, No 1, March 2013.

[35] Guo Z. and Malakooti B. Energy aware proactive MANET routing with prediction on energy consumption. International Conference on Wireless Algorithms, Systems and Applications, pp. 287–293, 2007.

[36] Loutfi A. and Elkoutbi M. A New Efficient and Energy-Aware Clustering Algorithm for The OLSR Protocol. International Journal of Computer Science and Network Security, vol.14, No. 7, 2014.

[37] Jabbar W.A., Ismail M. and Nordin R. Performance Evaluation of MBA-OLSR Routing Protocol for MANETs. Journal of Computer Networks and Communications, 2014.

[38] Huhtonen A.. Comparing AODV and OLSR routing protocols. Telecommunications Software and Multimedia (2004): 1-9.

[39] Hinds A., Ngulube M., Zhu S. and Al-Aqrabi H. A Review of Routing Protocols for Mobile Ad-Hoc NETworks (MANET). International Journal of Information and Education Technology, Vol. 3, No. 1, February 2013.

[40] Vasan D. et al. A Survey On Routing Protocols Performance Simulated In Different Scenarios With Different Simulators. International Journal of Wireless Communication Networks (IJWCN 2013).

[41] Saida M., Alsanabani M. and Alahdal T. Comparison Study of Routing Protocols in MANET. International Journal of Ad Hoc, Vehicular and Sensor Networks 1.1 (2014): 1-9.

[42] Gupta N. and Das S.R. Energy-aware on-demand routing for mobile ad hoc networks. Distributed Computing. Springer Berlin Heidelberg, 2002. 164-173, 2002.

[43] Chettibi S. and Chikhi S. An Adaptive Energy-Aware Routing Protocol for MANETs Using The SARSA Reinforcement Learning Algorithm. IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), 2012.

[44] Li C., Zhang H., Hao B. and Li J. (2011). A Survey on Routing Protocols for Large-Scale Wireless Sensor Networks. Sensors 2011, 11, 3498-3526; doi:10.3390/s110403498

[45] Heinzelman, W. R., Chandrakasan, A., & Balakrishnan, H. Energy-efficient communication protocol for wireless microsensor networks. Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000.

[46] Krishna M. B. and Doja M. N. Analysis of tree-based multicast routing in wireless sensor networks with varying network metrics. International Journal of Communication Systems, Volume 26, Issue 10, pages 1327–1340, October 2013.

[47] Shi L., Yao Z., Zhang B. Li C. and Ma J. An efficient distributed routing protocol for wireless sensor networks with mobile sinks. International Journal of Communication Systems, Article first published online: 13 APR 2014, DOI: 10.1002/dac.2785.

9 Tables

	Minimal coverage		**Preferable coverage**		**Extended coverage**	
	Low resiliency	**High resiliency**	**Low resiliency**	**High resiliency**	**Low resiliency**	**High resiliency**
Surface (km^2)	2.5 (5 × 0.5)		25 (5 × 5)		70 (10 × 7)	
Nb of nodes (accurate positioning)	20	40	100	200	280	560
Nb of nodes (erratic positioning)	(1)	80	(1)	400	(1)	1200

(1) Erratic deployment is based on a high resiliency of nodes to prevent topologic failure owing to relief and roughness.

Table 1 Number of nodes necessary to fulfil the mission goals depending on the coverage and resiliency.

Name	*Description*	*Fields (number of bits)*
Sink Route Request (*SRREQ*)	Control packet used to setup a tree with a specific data sink as root node.	- *SRREQ_ID* (32): the identifier of the SRREQ. Used to discriminate requests. - *sink_ID* (16): the global identifier of the data sink (i.e., the network address). - *sink_DSN* (32): the Destination Sequence Number of the data sink. - *path_cost* (16): the current cost to reach the data sink (extension of the hop count concept).
Route to Sink Error (*RSERR*)	Control packet used to trigger a new tree generation process, for a specific data sink. It is used to handle possible node failures.	- *RSERR_ID* (32): the identifier of the RSERR ID. Used to discriminate if a node has already broadcasted the RSERR of neighbour (to avoid indefinite flooding). - *source_ID* (16): the global identifier of the node which requests the formation of a new tree. - *sink_ID* (16): the global identifier of the data sink in question.
Hello	Control packet, periodically broadcasted and used to compute the list of neighbours in the network (and their residual battery) of each node.	- *node_ID* (16): the global identifier of the node generating the Hello packet. - *residual_battery* (8): the residual battery level (in percentage) of the node generating the Hello packet.

Table 2 Description of control packets for routing.

Data structure	*Description*	*Sub-fields (number of bits)*
routing_table	Table used to find a next-hop to deliver a data packet towards one of the available data sinks.	Data structure with variable size. Each entry contains: - *destination* (16) - *DSN* (32) - *active_route_flag* (1) - *valid_DSN_flag* (1) - *path_cost* (16) - *next_hop_ID* (16)
internal_SRREQ_ID	Value (32 bits) contained in the data sinks used to discriminate SRREQs each other.	-
list_of_RSERR_IDs	This list is used to avoid an indefinite flooding of a RSERR related to a specific data sink.	Data structure with variable size. Each entry contains: - *sink_ID* (16) - *RSERR_ID* (32)

internal_DSN	Value (32 bits) of the Destination Sequence Number contained into the data sinks to keep trace of the logical timestamp of routes towards it.	-
list_of_waiting_RSERRs	Structure to keep trace if the node has already sent a RSERR for a specific data sink.	Data structure with variable size. Each entry contains: - *sink_ID* (16) - *RSERR_ID* (32)
residual_battery	The value (8 bits) of the internal power level of the node (in percentage).	-
list_of_neighbours	List of the neighbours of the node, along with their residual battery levels.	Data structure with variable size. Each entry contains: - *neighbour_ID* (16) - *neighbour_residual_battery* (8)

Table 3 Routing node internal data structures.

Event/routing procedure in each node with ID id	*Pseudo-code*

Initialization
(event triggered at initial instant t_0.)
	begin
	list_of_neighbours = ∅;
	pending_data_packets_buffer = ∅;
	list_of_waiting_RSERRs = ∅;
	routing_table = ∅;
	list_of_RSERR_IDs = ∅;
	internal_DSN = 0;
	internal_SRREQ_ID = 0;
	end

Tree reconfiguration timeout
(event triggered every T_{TREE} seconds in a data sink with ID *id*)
	begin
	internal_SRREQ_ID = internal_SRREQ_ID + 1;
	internal_DSN = internal_DSN + 1;
	create a SRREQ packet *p*;
	p.SRREQ_ID = internal_SRREQ_ID;
	p.sink_ID = *id*;
	p.DSN = internal_DSN;
	p.path_cost = 0;
	send *p* in broadcast;
	end

Hello timeout
(event triggered every T_{HELLO} seconds)
	begin
	create a Hello packet *p*;
	p.node_ID = *id*;
	p.residual_battery = internal_residual_battery;
	send *p* in broadcast;
	end

Handle Hello *p* from network	**begin**
	if list_of_neighbours already contains *p*.node_ID,
	update the residual battery of the entry;
	else
	list_of_neighbours = list_of_neighbours ∪ {*p*.node_ID, *p*.residual_battery};
	end
	end

Neighbour *n* unreachable	**begin** remove the entry related to *n* from list_of_neighbours; **for** each sink not present in list_of_waiting_RSERRs, with ID *s_id,* find the next-hop *m* to reach *s_id* in the routing table; **if** *n* == *m,*

```
                                    set active_route_flag = 0 for the entry in question;
                                    list_of_RSERR_IDs[s_id] = list_of_RSERR_IDs[s_id] + 1;
                                    create a SRERR p;
                                    p.RSERR_ID = list_of_RSERR_IDs[s_id];
                                    p.source_ID = id;
                                    p.sink_ID = s_id;
                                    list_of_waiting_RSERRs = list_of_waiting_RSERRs ∪ {s_id, …
                                    list_of_RSERR_IDs[s_id]};
                                    send p in broadcast;
                            end
                    end
            end
```

Handle data packet from the application

```
1   begin
2       find the sink with minimal path cost (with ID min_sink_id);
3       retrieve the next-hop n to reach destination min_sink_id;
4       perform eventual data aggregation;
5       send the packet to node n;
6   end
```

Handle data packet from network

```
1   begin
2       retrieve the next-hop n from the routing table;
3       perform eventual data aggregation;
4       forward the packet to node n;
5   end
```

Handle SRREQ *p* from neighbour *n*

```
1   begin
2       if n != p.sink_ID,
3           Let rte be the routing table towards neighbour n (create it if does not exist);
4           rte.destination = n;
5           rte.next_hop_ID = n;
6           if rte created now or rte.valid_DSN_flag == 0,
7               rte. valid_DSN_flag = 0;
8               rte.DSN = -1;
9           end
10          rte.path_cost = compute_cij(distance(id,n), list_of_neighbours[n].residual_battery);
11          rte.active_route_flag = 1;
12      else
13          cost = p.path_cost + compute_cij(distance(id,n),
            list_of_neighbours[n].residual_battery);
14          retrieve routing table entry with destination == p.sink_ID (let rte be this entry);
15          if (rte == null) or (rte.active_route_flag == 0) or (rte.valid_DSN_flag == 0) or
            (p.sink_DSN > rte.DSN) or ((p.sink_DSN == rte.DSN ) and (cost < rte.path_cost)),
16              rte.destination = p.sink_ID;
17              rte.next_hop_ID = n;
18              rte.path_cost = cost;
19              rte.active_route_flag = 1;
20              rte.valid_DSN_flag = 1;
21              rte.DSN = p.sink_DSN;
22              list_of_RSERR_IDs[p.sink_ID] = p.SRREQ_ID;
23              remove the entry with p.sink_ID from list_of_waiting_RSERRs;
24              send an updated SRREQ in broadcast, with the increased path cost (set equal
                to cost);
25          end
26      end
27  end
```

Handle RSERR *p* from neighbour

```
1   begin
2       if id == p.sink_ID,
3           if p.RSERR_ID >= internal_SRREQ_ID,
4               internal_SRREQ_ID = p.RSERR_ID;
5               internal_SRREQ_ID = internal_SRREQ_ID + 1;
6               internal_DSN = internal_DSN + 1;
7               create a SRREQ packet q;
8               q.SRREQ_ID = internal_SRREQ_ID;
9               q.sink_ID = p.sink_ID;
10              q.DSN = internal_DSN;
11              q.path_cost = 0;
```

```
12                          send q in broadcast;
13                  end
14          else
15                  if list_of_RSERR_IDs[p.sink_ID] < p.RSERR_ID,
16                          list_of_RSERR_IDs[p.sink_ID] = p.RSERR_ID;
17                          send p in broadcast;
18                  end
19          end
20  end
```

Table 4 Pseudo-code of the routing procedures.

Name	*Size* (*bits*)
Sink Route Request (*SRREQ*)	96
Route to Sink Error (*RSERR*)	64
Hello	24
	184

Table 5 Size of control packets.

Data structure	*Maximum size* (*bits*)
routing_table	$82n_{sink}$
internal_SRREQ_ID	32
list_of_RSERR_IDs	$48n_{sink}$
internal_DSN	32
list_of_waiting_RSERRs	$48n_{sink}$
residual_battery	8
list_of_neighbours	$24d_{MAX}$

Table 6 Size of routing internal data structures.

Routing procedure	*Complexity of operations for each routing procedure (in the worst case)*
Initialization	$O(1)$ writes
Tree reconfiguration timeout	$O(1)$ reads + $O(1)$ writes + $O(1)$ sums + $O(1)$ message transmissions (of SRREQ packet)
Hello timeout	$O(1)$ reads + $O(1)$ message transmissions (of Hello packet)
Handle Hello p from network	$O(d_{MAX})$ reads + $O(1)$ writes
Neighbour n unreachable	$O(d_{MAX})$ reads + $O(n_{sink})$ reads/writes/sums/message transmissions (of RSERR packets)
Handle data packet from the application	$O(n_{sink})$ reads + $O(1)$ message transmissions (of data packet)
Handle data packet from network	$O(n_{sink})$ reads + $O(1)$ message transmissions (of data packet)
Handle SRREQ p from neighbour n	$O(d_{MAX} + n_{sink})$ reads + $O(1)$ sums + $O(1)$ writes + $O(1)$ message re-transmissions (of SRREQ packet) + $O(1)$ link cost computation.
Handle RSERR p from neighbour	$O(n_{sink})$ reads + $O(1)$ writes + $O(1)$ message re-transmissions (of RSERR packet)

Table 7 Number of execution (in the worst case) of routing procedures.

Operation	*Number of executions (in the worst case)*
Read	$O(n_{sink} + d_{MAX})$
Write	$O(n_{sink})$
Sum	$O(n_{sink})$
Link cost computation	$O(1)$
Message transmissions	$O(n_{sink})$

Table 8 – Complexity of a single routing procedure in the worst case.

Algorithm	Minimum node lifetime [days]	Network disconnection time [days]	Amount of data received at the data sink [MB]
Shortest path algorithm	6.67	17.92	21.83
Proposed energy-aware algorithm	17.50	17.65	21.78

Table 9 Minimum node lifetime (in days), network disconnection time (in days) and data received at the data sink (in MB) in the preferable coverage scenario (low resiliency and accurate positioning).

Algorithm	Minimal coverage			Preferable coverage			Extended coverage		
	Min. Node. Lifetime	Disc. Time	Rec. Data	Min. Node. Lifetime	Disc. Time	Rec. Data	Min. Node. Lifetime	Disc. Time	Rec. Data
Shortest path algorithm	32.2	59.06	13.65	6.67	17.92	21.83	2.29	6.41	21.90
Proposed algorithm	61.79	61.87	14.64	17.50	17.65	21.78	6.15	6.31	21.93

Table 10 Minimum node lifetime (in days) in the three low resiliency and accurate positioning scenarios.

Algorithm	Data fusion ratio	Minimal coverage	Preferable coverage	Extended coverage
1 sink in the centre	1	61.79	17.5	6.15
4 sinks uniformly distributed	1	42.33	21.56	7.29
	2	78.48	37.40	13.02

Table 11 Minimum node lifetime (in days) in the low resiliency and accurate positioning scenarios, in function of the number of sinks. The proposed link cost is evaluated.