Ambient Noise in Wireless Mesh Networks: Evaluation and Proposal of an Adaptive Algorithm to Mitigate Link Removal

Jesús Friginal[a], Juan-Carlos Ruiz[b], David de Andrés[b], Antonio Bustos[b]

jesus.friginal@laas.fr, {jcruizg, ddandres}@disca.upv.es, anbusrod@alumni.upv.es

[a]*LAAS-CNRS, 7 Avenue du Colonel Roche, 31077 Toulouse Cedex, France*
[b]*STF-ITACA Universitat Politècnica de València, Campus de Vera s/n, 46022, Spain*

Abstract

Ambient noise is one of the major problems in Wireless Mesh Networks (WMNs). It is responsible for adverse effects on communications such as packet dropping, which dramatically affects the behaviour of ad hoc routing protocols, a key element of these networks. This issue is of prime importance for WMNs since the loss of communication links experienced by nodes may strongly increase the convergence time of the network. Furthermore, the dynamic nature of this problem makes it difficult to address with traditional techniques. The contribution of this paper goes in the direction of (i) exploring this problem by assessing the behaviour of three state-of-the-art routing protocols in presence of *ambient noise* (OLSR, B.A.T.M.A.N and Babel) and (ii) improving the resilience capabilities of these protocols against *ambient noise* by proposing an algorithm for the link quality-based adaptive replication of packets, named LARK. The goal of LARK is to avoid the loss of communication links in the presence of high levels of *ambient noise*. The effectiveness of the proposal is experimentally assessed, thus establishing a new method to reduce the impact of *ambient noise* on WMNs.

Keywords: wireless mesh networks, ambient noise, adaptive mechanisms, experimental resilience evaluation.

1. Introduction

The progressive interest (and dependence) of our society on mobiquitous (mobile and ubiquitous) systems explains why incorporating new added-value services into modern devices is not an option but a demand requirement. Ad hoc networks enable the rapid and spontaneous creation of low-cost data networks without the need of any pre-existing communication infrastructure. Wireless Mesh Networks (WMN) follow this paradigm to offer self-managed networking solutions, which are currently exploited by (i) some city councils to democratise the free wireless Internet access to their citizens; and (ii) private companies to provide affordable Internet access to isolated populations (such as Meraki[1] and TerraNet[2]).

In mainstream WMNs, routing protocols are responsible for the efficient creation of multi-hop wireless communication routes among distant nodes. Most of the routing protocols used in WMNs are proactive, which means that they periodically exchange topology control messages with other nodes in order to maintain updated routes towards Internet gateway nodes. Unfortunately, signal propagation and environment saturation problems may affect the reception of control messages exchanged by routing protocols. This issue, known as *ambient noise*, might lead to excessive loss rates or packet delays [1].

In such scenarios, a simple recovery action that can be adopted by a proactive routing protocol is to force the two nodes concerning the affected link to switch to a different channel with less spectrum activity. This technique may eventually have a very negative impact over the network convergence time, since it may require a large portion of the rest of mesh nodes to switch channels so the network topology stays connected. If degradation of communication links induced by noise does not partition the network, an alternative approach adopted by most mesh routing protocols consists in repairing affected paths by automatically choosing other alternative links to maximise some quality link metrics of the protocol [2]. Today, most of existing WMN routing protocol implementations incorporate mechanisms to react against *ambient noise*. Overly agile protocol reactions may lead to route flapping [3] and must be avoided. In fact, they may increase the network overhead by flooding the route repair control messages without gaining much throughput.

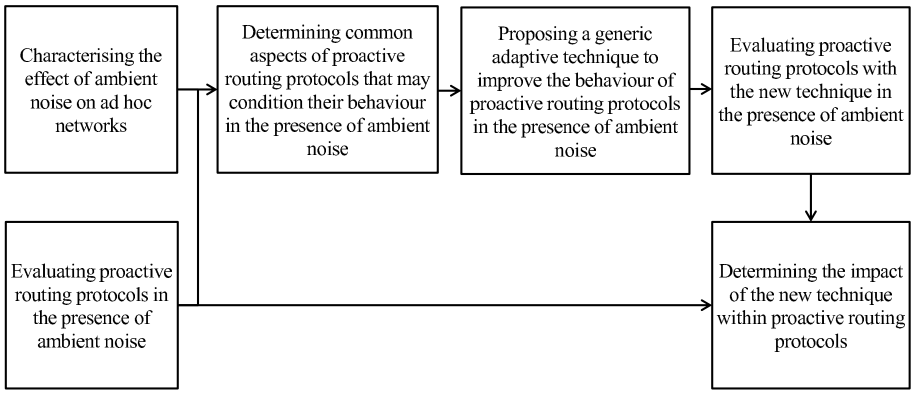
[1]http://meraki.com
[2]http://terranet.se

One possible way to counter this effect is to use flap damping. The goal is to limit the global impact of unstable routes by temporarily suppressing routes with rapid changes over short time periods. However this technique may cause persistent oscillation in the network due to the adverse interactions between flap damping and route convergence [4]. On the other hand, if the reaction of the protocol against noise is too slow it may entail the loss of existing communication links in the network. The main consequence is the activation of the self-configuration capabilities provided by the routing protocols to establish new communication routes among affected nodes. When many links result affected, the convergence time increases, which reduces the network steadiness and availability [5].

It seems thus reasonable to keep communication links alive as long as possible in case of being subjected to *ambient noise*. This can be done by simply tuning routing protocols in order to increase the lifetime of their links, thus reducing the issues related to the network converge time. What must be carefully considered is the overhead derived from such a tuning and the pertinence of the resulting configuration along the time. In [6], for instance, authors propose an automatic approach to manage link communication faults in WMNs by inferring suitable configurations from network model simulations. Facing dynamic perturbations, such as *ambient noise*, asks for more dynamic strategies to adapt at runtime the level of link protection against *ambient noise* in WMNs.

Following this trend, this paper proposes to explore the effect of *ambient noise* in well-known state-of-the-art proactive routing protocols named OLSR [7], Babel [8] and B.A.T.M.A.N [9], as a first step to propose an adaptive strategy to prevent nodes from losing their communication links in the presence of high rates of *ambient noise*. By introducing this algorithm within routing protocols we are able to reduce the convergence time of routes while improving their availability in a dynamic way.

Figure 1 can assist the reader to understand the motivation that leads us to carry out every step we take towards this goal. The paper is consequently structured in this fashion. So, firstly, Section 2 describes the innate capabilities of such protocols to *ambient noise* adaptation. Section 3 identifies the various parameters affecting the behaviour exhibited by the protocols, and experimentally assesses such a behaviour under similar experimental conditions. Results show that under similar conditions, differences between protocols mainly rely on the overhead they induce in the network rather than in their protection capabilities. However, it seems clear that it is nei-



Characterising the effect of ambient noise on ad hoc networks
Determining common aspects of proactive routing protocols that may condition their behaviour in the presence of ambient noise
Proposing a generic adaptive technique to improve the behaviour of proactive routing protocols in the presence of ambient noise
Evaluating proactive routing protocols with the new technique in the presence of ambient noise
Evaluating proactive routing protocols in the presence of ambient noise
Determining the impact of the new technique within proactive routing protocols

Figure 1: Paper's roadmap.

ther desirable nor affordable to suffer from such an overhead when noise does not exist or it is very low. The alternative that is introduced in Section 4, focuses on the feasibility of adapting the level of protection provided by existing link-quality-based mechanisms, and thus the level of induced overhead, to the level of noise experienced at each moment, in each network node, by each communication link. The idea is to keep links alive (i) by replicating control messages when the network is exposed to high levels of noise, and (ii) by dynamically incrementing or reducing the level of replication attending to the evolution of such a noise. The approach is deployed and assessed on the three considered routing protocols, thus showing the feasibility of the approach and its generality. Section 5 places this study with respect to related works. Finally, Section 6 concludes the paper.

2. Proactive Routing and Noise Adaptation

This section provides an overall view of what is a proactive routing protocol. Then, the notion of *ambient noise* and how it can impact on routing is introduced. Finally, the viability of link-quality-based techniques to face this problem is discussed.

2.1. Proactive Routing Protocols

Proactive routing protocols provide facilities for discovering, establishing and maintaining communication links among 1-hop neighbour nodes. Depending on the strategy considered to compute such routes, proactive routing protocols can be link-state or distance vector. Link-state routing requires

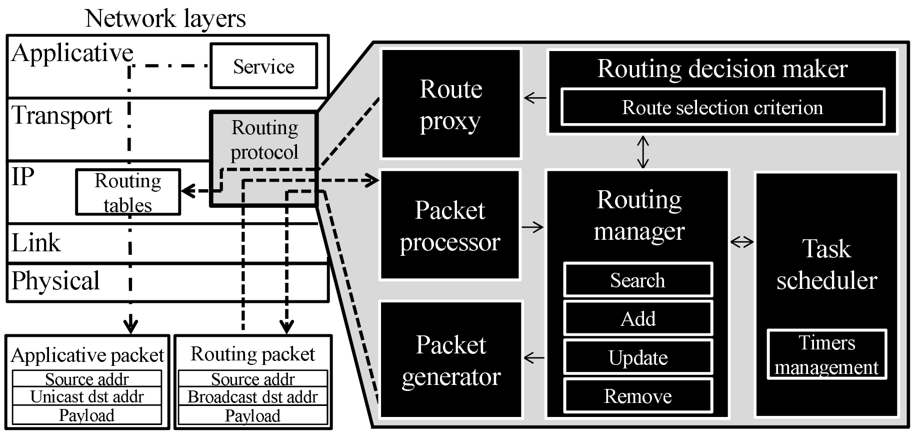


Figure 2: General architecture of a proactive routing protocol.

each node to maintain at least a partial topology map of the whole network. Conversely, in distance vector protocols, a given node only knows their 1-hop neighbours, but not the rest of the route. Despite such differences, the general architecture of a proactive routing protocol is depicted in Figure 2. As can be seen, the first important thing is to establish a clear difference between the packets generated at the applicative layer (*applicative packets*) from those that are generated by the routing protocol (*routing packets*) in order to keep the network nodes interconnected. Proactive routing protocols rely on a *packet generator* which is in charge of periodically creating and sending routing packets.

A core element of any proactive routing protocol is its *routing manager*. It is the responsible for handling (searching, adding, updating and removing) existing and new routes attending to the different situations of the network. When computing a route from a source to a given destination node, it is possible to distinguish three different situations. Obviously the worst case consists in finding no route, in such a case, the communication will not be possible. Other situation may be finding one single route, but in case one node in the route fails, the communication will be compromised. Finally, the best case consists in discovering different alternative routes. In this situation, the *routing decision maker* will rank the different alternative routes aiming at selecting which is the best option, and which ones remain as backup routes in case the best one fails. Dynamic factors of the environment like mobility or the *ambient noise* force the alternation of these three basic situations in the network. In order to adapt the protocol behaviour to these circumstances, routing packets can be sent or forwarded attending to network generated events (such as the reception of a routing packet asking for a particular type of action, or the timeout of an internal timer). The *task scheduler* manages all the internal protocol timers, which are used to periodically trigger the broadcasting of control routing messages to the network and the expiration of a link. Valid incoming routing packets containing routing information are processed and stored by the *packet processor* in the protocol internal routing information repository. The content of incoming packets and the information stored in the routing repository is different in distance vector and link state protocols. However, in both cases, when a change in the state of a link is discovered (i.e., a new link has been created, updated or removed), the *route proxy* reflects such a change in the *routing tables* of the node, which are located in the network stack (which is not part of the routing protocol).

From all the events taking place in the routing protocols, link removal is the most critical from the viewpoint of network connectivity. This event becomes potentially probable if the links maintained by the routing protocol stop being updated. This typically happens when there is any persistent problem affecting the reception of routing packets, thus causing the link expiration. Unfortunately, this effect may increase the risk of network partitioning specially if there is just one single route available between source and destination, consequently isolating certain regions of the network.

2.2. Ambient Noise and its Consequences

On the last decades, wireless communications have gained importance in our daily life. However, there are two main factors that determine their behaviour on practice: (i) the impact of the radio propagation environment and (ii) intra-system interferences. With respect to the first factor, radio propagation is affected by daily changes such as water vapour in the atmosphere due to the sun, or the presence of fix or mobile obstacles, such as walls or vehicles. With respect to the second factor, as the number of wireless networks increases, the environment that these networks share becomes increasingly saturated with signals from a wide variety of sources. More concretely, since the industrial, scientific and medical (ISM) radio bands are unlicensed, many different types of equipment (such as cordless phones, cell phones or radio-frequency-based remote controllers typically supporting e.g. the IEEE 802.11x and the IEEE 802.15.x standards) may use or generate noise in these frequencies, either accidentally or deliberately (e.g., microwave ovens and other electrical devices). As a result, an increasing number of transmissions interact and interfere one another and cause unforeseen problems for communication protocols. Such problems typically increase delays, bit error rates and packet loss probabilities on receivers. This phenomena is typically known in the bibliography as background noise or *ambient noise* [10].

The existence of *ambient noise*, and its negative effect in wireless mesh networks (WMNs), is something indisputable. In this paper we focus on the packet dropping originated by *ambient noise* as one of the main problems that might lead wireless networks in general, and WMNs in particular, to suffer from link removals.

2.3. Link Quality Metrics

In general, solving the problem of *ambient noise* is not an easy task. In wireless communications, this issue is tackled by introducing adaptive schemes on systems (link adaptation, adaptive coding schemes, TCP rate adaptations, etc.).

$$ETX(i) = \frac{1}{RPAR(i) \cdot NRPAR(i)} \tag{1}$$

In the particular domain of multi-hop wireless communications, the traditional Ethernet philosophy of selecting a given communication link towards a destination, among those available, with the criterion of minimising the number of remaining hops (*hop-count*) is a poor choice. Due to noise, the quality of the communication links between mesh nodes is not the same, which advices against the use of such a simplistic metric. Since mid-00s, the notion of link quality is basically computed by each node according to the amount of routing information received from its neighbourhood: the higher this reception rate the better. The *Expected Transmission Count* (ETX) is without any doubt, the most well-known metric for characterising the quality of a link [2]. It reflects the number of expected transmissions of a packet to be received without error at its destination. This number varies from one to infinity. An ETX of one indicates a perfect transmission medium, whereas an ETX of infinity represents a non-functional link. ETX is shown in Expression 1. Given a sampling window in link i, $RPAR(i)$ is the Routing Packets Arrival Rate seen by a node, and $NRPAR(i)$ is the $RPAR(i)$ seen by the neighbour node. An alternative quality metric is proposed in [9] and it is called *Transmission Quality* (TQ). This second metric shares the same principles established by ETX but it is computed in another way. In essence, receivers calculate the number of routing packets received against those expected following Expression 2. In this expression $RPAR(i)$ and $NRPAR(i)$ have the same meaning than in ETX, and MAX_LQ is a constant which bounds the ideal maximum quality. The term $p(i)$ refers to the penalty that is applied to unidirectional links, to promote those that are bidirectional. The higher the TQ, the better.

$$TQ(i) = \frac{RPAR(i) \cdot NRPAR(i) \cdot p(i)}{MAX\_LQ^2} \tag{2}$$

For the sake of simplicity, the rest of this paper will denote the quality of a link i as lq_i and it will be always interpreted as the higher the lq_i the better. Under such an assumption, lq_i should be viewed as $1/ETX$.

As a result, whenever two communication links towards a destination i and j are available, the protocol selects the one providing the better link quality. From this viewpoint, and since such link quality metrics are periodically recomputed, one can say that presented approaches adapt to variations of quality derived from *ambient noise*. However, as Section 3 will show, such a capacity is not enough to keep communication links alive whenever such an *ambient noise* persists or increases.

3. Case Study

This section introduces some representative proactive routing protocols. Then, the *ambient noise* model used in this paper is presented. This model will be used to subject targeted routing protocols to the effects of *ambient noise*. After that, targeted routing protocols will be thoroughly assessed to show to what extent their level of adaptiveness to resist the presence of *ambient noise* depends on a particular set of parameters. Such results will be finally analysed from the viewpoint of the overhead introduced to determine if the cost to pay is affordable or not.

3.1. Proactive Routing Protocols Under Study

Three state-of-the-art proactive routing protocols have been selected as targets of this case study. Their names are Optimised Link State Routing

(OLSR), the Better Approach To Mobile Ad hoc Networking (B.A.T.M.A.N) and Babel.

OLSR [7] is the most well-known link state protocol and one of the most widely-used proactive routing protocols nowadays. OLSR employs an optimised flooding mechanism, where only special nodes called Multi-Point Relay (MPR) are responsible for broadcasting the routing information along the network. Although the initial specification of OLSR (RFC 3626) established route computation using hop-count as metric, the current specification OLSRv2 promotes the use of link quality extensions, like ETX.

B.A.T.M.A.N [9] is a novel proactive distance-vector routing protocol. For each node, B.A.T.M.A.N periodically sends out broadcast messages to inform neighbours of its existence. This process is repeated until the routing information reaches all network nodes. For each link the routing packets arrival rate is advertised so that neighbour nodes can determine the link quality. B.A.T.M.A.N uses the TQ metric to estimate the quality of network links.

Babel (RFC 6126) [8] is the most recent protocol under consideration. This is a distance-vector routing protocol that has two main characteristics to optimise its relay mechanism. On one hand, it uses history-sensitive route selection to minimise the impact of route flaps. In such a way, the route selection favours the previously established path rather than alternating between two routes. On the other hand, it forces a request for routing information each time it detects a link failure from one of its preferred neighbours. This is a best-effort mechanism to reduce the convergence time of the network. Like OLSR, Babel uses ETX as a metric of quality for network links.

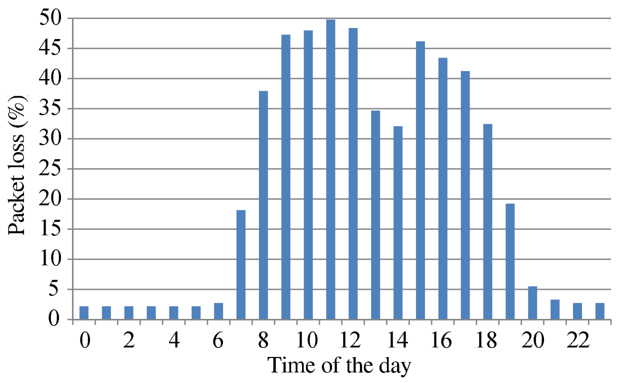


Figure 3: Packet loss conditions in the considered indoors scenario.

Well-known implementations of OLSR (*olsrd*[3]), Babel (*babeld*[4]) and B.A.T.M.A.N (*batmand*[5]) were taken into account for the purpose of experimentation.

3.2. Fault Model & Faultload

In the development of multi-hop routing protocols, the presence of external sources of transmission is generally considered as a source of errors. Generally, these evaluations are performed in simulators that rely on noise models more or less realistic depending on the assumptions of the environment made [11, 12]. However, as stated in [13], the noise levels in real-world contexts are much higher than those used in current simulators. In essence, this is one of the main reasons why the case study considered in this paper focuses on obtaining results from real-world experimentation. Consequently, characterising the levels of *ambient noise* in the environment where the network will be finally deployed is very important to determine which is the routing protocol that best fits the scenario conditions. The level of *ambient noise* of the laboratory where the experimentation of this paper takes place, was consequently assessed. Accordingly, a simple communication between two conventional Linksys routers in the same radio range (thus not requiring the aid of routing protocols) was deployed in our department for a period of 24 hours. This communication consisted in the establishment of a simple UDP data flow of 200 Kbps throughout the *iperf* tool [14]. As *ambient noise* is eventually manifested in a reduction of the packets correctly delivered [15], the results of the experiment, in Figure 3, show the percentage of packet loss caused by the *ambient noise* along the day. Thanks to this experimentation, we could distinguish three different levels of *ambient noise*: a high *ambient noise* along the workday involving a packet loss ranging from 35% to 50%, a moderate one matching with the lunch breaks from 5% to 35% and a low one from 0% to 5% at night. Despite not being so frequent, the range from 50% to 100% typically represents additional external perturbations like malicious attacks from signal inhibitors or accidental faults like interferences from microwave ovens.

Once the fault model defined, it is necessary to specify how to recreate the presence of *ambient noise* in practice. The emulation of *ambient noise* through packet loss is a possible solution to experience the exact conditions of

[3]v.0.6.0 from www.olsrd.org

[4]v.1.1.1 from www.pps.jussieu.fr/jch/software/babel/

[5]v.0.3.2 from www.open-mesh.org

Table 1: Script to introduce a given packet loss $LOSS for those packets received in port $PORT.

```
...
tc qdisc add dev $DEV root handle 1:  prio
tc qdisc add dev $DEV parent 1:3 handle 30:  netem loss ${LOSS}%
tc filter add dev $DEV protocol ip parent 1:0 prio 3 u32 match ip dport $PORT 0xffff flowid 1:3
...
```

noise in a given experimental environment. Thus, it is possible to enhance the repeatability and reproducibility in the experimentation, and the accuracy in the results provided. So, the use of tools such as *netem* [16] for the injection of a given rate of packet loss in the network has been required. It enables experimenters to tune the intensity of packet loss for a given type of traffic. So, in order to clearly determine the impact of *ambient noise* in the routing level, only the routing packets addressed at default ports 698 (for *olsrd*), 4305 for *batmand* and 6696 (for *babeld*) was targeted for packet loss. In this way, it is possible to conclude which is the unavailability introduced at the applicative layer due to the *ambient noise* at the routing level. Table 1 shows a script to force the packet loss (from 0% to 100%) to a given level of noise.

3.3. Measures

As previous works indicate [13], the Effective Packet Delivery Ratio (EPDR) is one of the measures that better represents the impact of *ambient noise* in the network behaviour. The EPDR for a given route r is expressed as the percentage of applicative packets received from the total sent. It can be theoretically computed according to Formula 3, where the route availability (RA) is typically expressed in percentage and accounts for the average time the route between source and destination was ready to be used, and the packet loss ratio (PLR) caused by *ambient noise*. Obviously, the RA ends up conditioning the EPDR in the applicative layer because, if any link fails, all the communications supported by such a link will be irremediable affected. The higher RA and the lower PLR, the better.

$$EPDR(r) = RA(r) \cdot (1 - PLR(r)) \tag{3}$$

3.4. Experimental Setup

To carry out our experiments and assess the resilience of routing protocols against *ambient noise*, a new network deployment was considered this time where both regular and tiny devices were considered for their execution. Regular nodes were implemented by 7 HP 530 laptops with a processor of 1.6GHz and 512MB of RAM running Ubuntu 7.10 OS. Tiny devices consisted of 10 Linksys WRT54GL routers with a processor of 200 MHz and 16MB of RAM running OpenWRT White Russian OS. Considered nodes were equipped with both a wired Ethernet and a wireless IEEE 802.11b/g interface. The fact of selecting two different types of devices helps us to show the portability of targeted routing protocols.

Concretely, a WMN implementing the topology depicted in Figure 4 recreates the map of our department. Routes of 4-hops distance (A-B and A-C) have been considered as representative in our study. This experimental setup takes into consideration the two basic types of routes already explained in Section 2.1, route A-B with alternative paths and route A-C without alternative paths. In the second case, routing protocols typically change the old best route by the new best one from the set of available alternative routes in case of updates in their links' quality. The study of this case has been simplified by considering just two alternative routes, which is the minimum number of routes to perform a route switch. Thus, route A-B can be established through nodes x_i (x_1, x_2 and x_3) and y_i (y_1, y_2 and y_3) respectively. Given the unavoidable presence of physical obstacles, the studied routing protocols usually find the best route from A to B (and vice-versa) through x_i nodes most of the times, rather than through y_i nodes.

According to the previous setup, the goal will be to study the impact of *ambient noise* in the target routes. The faultload in charge of easing this task will be deployed as follows. On one hand, route A-C will be subjected to the presence of a gradually increasing *ambient noise*. As far as there are no alternative routes, this experimentation will be useful to study the effects of route partitioning. The presence of *ambient noise* along the whole route will be emulated by injecting an *ambient noise* equivalent to a given packet corruption, thus creating a homogeneous *ambient noise*. On the other hand, the experimentation on route A-B will be oriented to analyse the effects of an heterogeneous presence of *ambient noise* by considering two different areas of noise in the network. One zone, delimited by all the nodes in route A-B but x_3, will be subjected to the same gradually increasing presence of packet corruption. Conversely, remaining node (x_3, typically taking part in

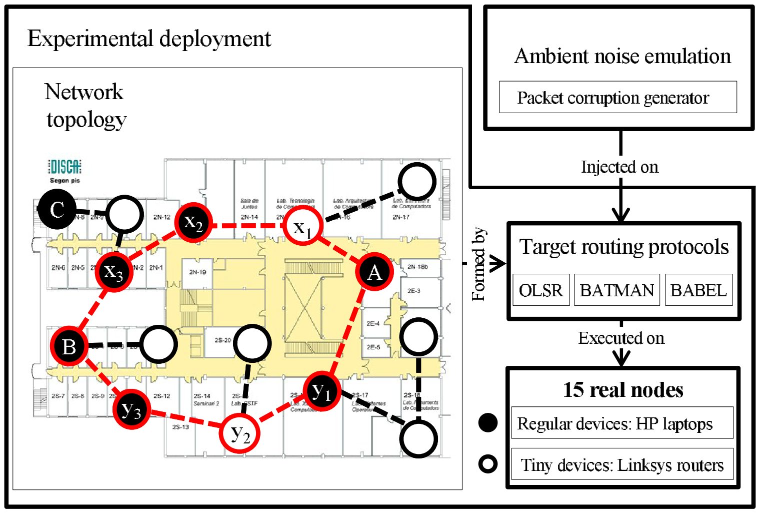


Figure 4: Experimental WMN deployment.

the active route by default) will be exposed to a constant extreme presence of *ambient noise* of 95% packet corruption. Forcing this worst-case situation, we ambition to study the effects of route switching in route A-C from x_i to y_i nodes, and consequently analysing its impact on the reconfiguration time of the protocol to offer a new alternative route. To animate routes, a workload consisting on UDP data flows of 200 Kbps is established to compute EPDR. In order to limit the influence of real *ambient noise* in results, experimentation was carried out at night, assuming an acceptable intrusiveness of 0% to 5% of real (non-emulated) packet corruption for wireless networks. In summary, 600 experiments of 300s each divided in two experimental campaigns (one per route type) were executed.

3.5. Impact of ambient noise

Figure 5a illustrates the results of the experiments obtained from route A-C, measuring the packet delivery ratio through x_i nodes. As observed when *batmand* is in charge of routing, the packet delivery ratio decreases proportionally with the amount of *ambient noise* introduced. *babeld* also presents very similar results, while *olsrd* starts degrading from 30% *ambient noise*. If analysing these results, the case representing the ideal packet delivery ratio loss should involve an identical decrement with respect to the *ambient noise* introduced. Any case where the packet delivery ratio decreases faster than

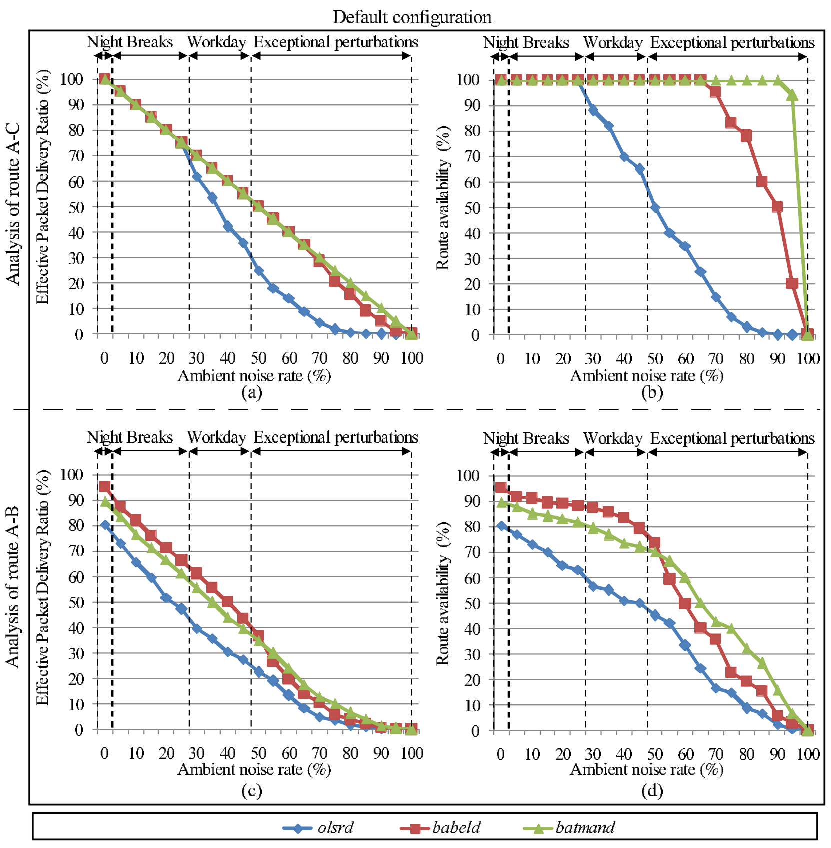


Figure 5: Experimental results obtained applying the default configuration.

the *ambient noise* introduced, necessarily involves the presence of any additional effect impacting on the packet delivery ratio. To study this effect in more detail, Figure 5b shows the route availability of the different routing protocols considered. In this graphic, it is possible to appreciate the robustness of the route exhibited by each routing protocol against *ambient noise*. As can be deduced, any route availability below 100% impacts negatively on the global packet delivery ratio of the route. Consequently, the longer the protocol can maintain the route available, the better for the packet delivery ratio. In this sense, *batmand* deserves being considered the best routing protocol since their route A-C strongly resists and does not create network partitioning until introducing 95% of *ambient noise* in the system. *babeld*, starts degrading the route availability with 70% of *ambient noise*. Finally,

Table 2: Reconfiguration time in route A-B.

Protocol version	**Reconfiguration time (s) per packet loss (%)**									
	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
olsrd v.0.6.0	58	81	106	130	147	165	200	250	274	293
babeld v.0.9.6	13	27	33	38	50	80	151	193	242	283
batmand v.0.3.2	28	45	51	62	87	90	120	172	203	253

olsrd starts suffering the impact of *ambient noise* (around 30%) within the daily levels of *ambient noise* (breaks and workday).

Now let us focus on the results obtained for route A-B. Figure 5c shows the packet delivery ratio when the route traverses a zone of extreme *ambient noise* (through node x_3) and the route A-B must be dynamically re-established through y_i nodes. In this case, the major difference introduced in Figure 5c with respect to Figure 5a is that packet delivery ratio never achieves 100% even in absence of *ambient noise*. Figure 5d illustrates the route availability for this type of experiment to explain this result. From this graphic it is possible to deduce that the application of the new route is not for free and involves a cost, even in absence of *ambient noise*. For the time required to apply the new route when the old one is no longer available (reconfiguration time or convergence time), the communication between A and B is not possible. Consequently, the longer this time, the major impact on the route availability. Concretely, *babeld* is the protocol presenting the best behaviour up to considering an *ambient noise* of 50%, but beyond this rate, *batmand* tolerates the presence of *ambient noise* slightly better. In any case, *olsrd* is the worst option. These results are consistent with the reconfiguration time measured in Table 2, and show how this time is one of the main reasons of route unavailability in WMNs.

The considerable differences in the behaviour of protocols lead us to analyse their source code to understand why protocols behave in that way. After analysing the code in detail throughout debugging tools like *gdb*, we observe that, the previous ranking established is not casual. Behind the name of different variables and constants (always consistent with their respective specification), there are three common parameters characterising the behaviour of proactive routing protocols against *ambient noise*. Table 3 identifies them and states their default values. T, is the default period to send a routing packet advertising a given link, and $Twindow$ is the validity time determining the temporal window after which the protocol decides whether discarding a link or not. Basically, the use of these two time-related parameters is located within the task scheduler module already presented in Figure 2. The Minimum Quality Threshold (MQT) defines the minimum acceptable quality before removing a link. This quality-based parameter is used within the routing manager to decide about the routing capacity of a given link (see Section 2.1). After analysing the target routing protocols, the conditions that must be satisfied to remove a link have been ordered from the most reactive to the most conservative as follows: *olsrd* and *babeld* require (i) the expiration of the $Twindow$ or (ii) exceeding the MQT. Conversely, *batmand* only requires the expiration of $Twindow$ before removing the link. Surprisingly, the notion of MQT is never taken into consideration in *batmand*. Unlike *babeld* and *olsrd*, the link quality is only updated in *batmand* when getting new information through incoming routing packets. If focusing on the configurations of T (shown in Table 3) to analyse how many opportunities has a protocol to refresh their link quality within a $Twindow$, it is easy to estimate that *olsrd* can only send 6 packets to update the quality of the link, whereas *babeld* admits up to 15 packets, and *batmand* has 200 new opportunities. Obviously, the more opportunities to update the link quality, the quicker the routing protocol can react against unexpected changes.

3.6. Tuning the routing protocol configuration

Essentially, as can be deduced, the success of *batmand* in our results is likely not due to its conservative policy, but to the configuration of parameters considered, which provides *batmand* a major frequency to send packets and consequently more opportunities to update the quality of their links than the rest of routing protocols considered. To make the comparison between the routing protocols considered fairer, let us apply the parameters configuration used in *batmand* to the rest of protocols. Additional experimentation was required to achieve this goal. The results finally obtained are shown in Figure 6. Figures 6a and 6b show the results of packet delivery with re-

Table 3: Critical parameters for the route availability in WMNs.

Protocol implementation	Twindow (s)	T (s)	MQT (%)
olsrd v.0.6.0	30	5	10
batmand v.0.3.2	200	1	none
babeld v.1.1.1	60	4	0

spect to the *ambient noise* rate. One can observe above all a considerable improvement in *olsrd* in particular for route A-C. According to route A-B, the enhancement on the packet delivery ratio concerns both *olsrd* and *babeld*. As far as the packet delivery ratio depends on the route availability, let us analyse deeply the results from that viewpoint.

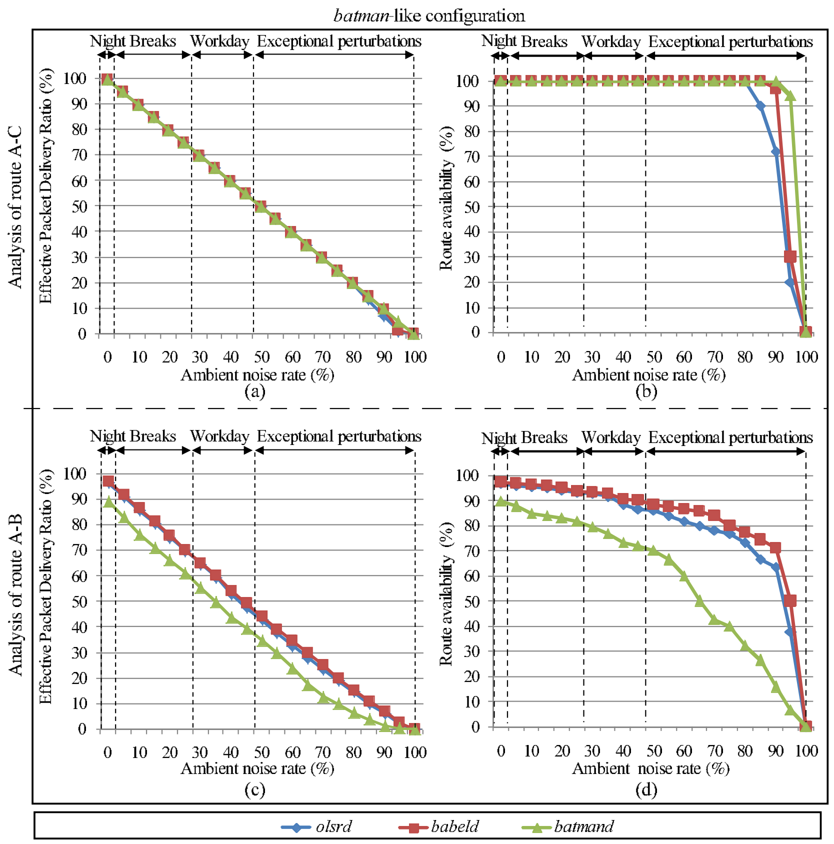


Figure 6: Experimental results obtained applying the *batmand*-like configuration.

If focusing on Figure 6b, it is worth mentioning how *olsrd* (above all) and *babeld* improve the robustness of the route A-C. With the *batmand*-like configuration, the route degradation in *olsrd* starts around 80% of *ambient noise* rate, enhancing 50 percentage points (pp) with respect to its default configuration. In the case of *babeld*, the improvement is around 20 pp. *batmand* keeps on being the best protocol in this scenario, but the differences with the other protocols have been strongly reduced.

If considering now route A-B (through node x_3), an interesting result can be observed from Figure 6d. In this case, the result is striking not so because *olsrd* and *babeld* increase their route availability (as expected if reducing the period T between packets sent), but because they behave better than *batmand*, which was considered the best routing protocol for route A-C. Evidently, something else apart from the parameterisation must be influencing these results. The conservative policy of *batmand* seems to be its drawback when facing dynamic changes of routes caused by an extreme *ambient noise*. As the network topology in Figure 4 states, when x_3 dramatically starts losing packets, the route A-B through x_i nodes is no longer available. However, node A has no indication that the route through nodes y_i is better than the offered through x_i until the next routing packets from B to A arrive through y_i and enhance the route quality of x_i. Obviously, the route reconfiguration time is intimately related to the *ambient noise* in the network. The harder the *ambient noise* conditions, the longer the reconfiguration time. Conversely to *batmand*, *olsrd* and *babeld* implement instruments like the MQT which promote the protocol reaction to minimise the reconfiguration time, which, as shown in Figure 6d, have been proven useful. In this sense, the protocol in node A is able to react earlier not only because of receiving packets from y_i, but because packets from x_i announce a broken link with x_3 once the MQT exceeded. As seen, the selection of a suitable parameterisation can improve the robustness of routing protocols against *ambient noise*. However, it is necessary to analyse the cost to pay in terms of the routing overhead introduced in the network before taking any decision.

Figure 7a and Figure 7b, study the average routing overhead introduced by each node when applying the default and the *batmand*-like parameterisation respectively. If analysing Figure 7a, *batmand* is the protocol with the highest routing overhead in terms of both packets sent and received when applying the default configuration (50% more than *olsrd* and *babeld* in the case of routing packets sent, and 127% and 78% in the case of *olsrd* and *babeld* respectively for the routing packets received. However, the trend changes when applying the *batmand*-like parameterisation. In this case, *olsrd* obtains the highest routing overhead in terms of packets sent (400% more than *batmand* and 58% more with respect to *babeld*) while *olsrd* and *babeld* increase the received routing overhead 161%. Since the considered routing protocols send packets with the same period T, these differences can be explained due to the average size of the routing packets sent by each routing protocol (380B in *olsrd*, 220B in *babeld* and 78B in *batmand*). In this case, the higher size of

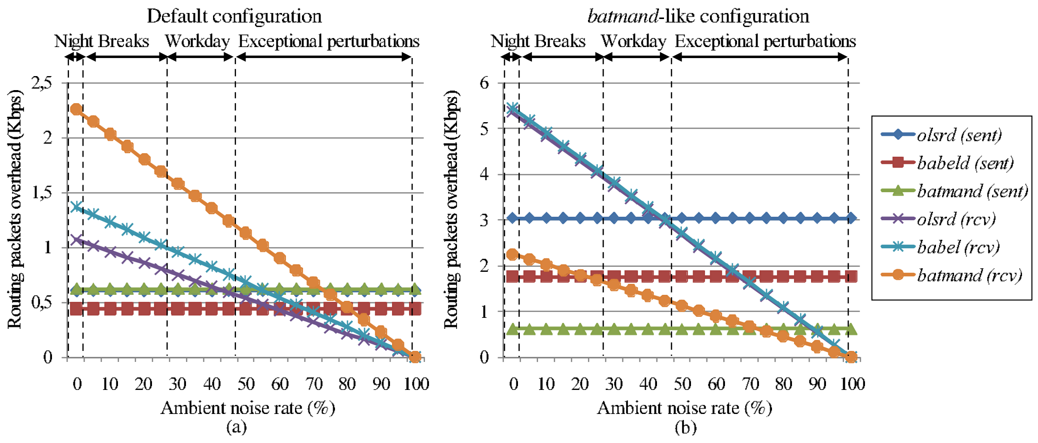


Figure 7: Routing overhead induced by routing packets sent and received.

olsrd packets penalises its routing overhead. However, it is worth noting the lack of mechanisms to prevent the flooding of routing packets in *babeld* and *batmand*. This means that the cost of such a protocol in terms of packets sent is quadratic and depends on the number of nodes $O(n^2)$. This fact could benefit *olsrd* if performing more experiments in a network including a wider amount of nodes, given the optimisation mechanism based on multi-point relay which provides *olsrd* a cost $O(n)$.

As graphics show, regardless the configuration used, the routing information rate sent is always constant because routing protocols periodically send the same (or almost the same) amount of information. This routing overhead is characterised for being ambient-noise independent. Conversely, the routing information received highly depends on the presence of *ambient noise* and its reception is directly proportional to the amount of packet corruption induced by the *ambient noise*. This fact makes that the ratio between the routing packets sent and received is quite disproportional as the *ambient noise* increases. Indeed, there are situations where the routing protocol could afford sending less routing packets to keep on maintaining the routes alive in presence of little *ambient noise*. However, in situations with a severe presence of *ambient noise*, the amount of routing packets received, decreases to the extent of provoking the mentioned problems of network partitioning and long convergence (or reconfiguration) times. One could think on parameterising the routing protocol according to the level of *ambient noise* in the network, however any offline configuration of these protocols stops being valid when the conditions of the environment, and specially of the *ambient noise*, vary over time. Given these situations, the provision of adaptive strategies to balance the routing overhead could result very useful to introduce the necessary routing overhead in the network to keep the links alive.

4. LARK: A Link-quality-based Adaptive Replication of Packets

This section faces the problem of *ambient noise* in proactive routing protocols proposing a generic adaptive strategy which enables the routing protocol to increase the routing overhead only when required. Replication is a well-known technique in the domain of fault tolerance that can be used for this purpose. The use of packet replication in this solution is devoted to ensure the reception of the routing information even in the presence of a high level of *ambient noise* that disturbs the communication between nodes. So, this approach, denominated Link-quality-based Adaptive Replication of Packets (LARK), can be useful in environments affected by *ambient noise* when links run the risk of disappearing or it is necessary to speed up the reconfiguration time.

4.1. Analytical overview of the technique

LARK is based on the principles of T, $Twindow$ and MQT previously identified in Section 3.5. As stated, nodes (re-)compute the quality of each one of their links each T. A link is lost whenever (i) its quality is lower than the MQT accepted by the protocol or (ii) no routing message is received for a period $Twindow$, despite its link quality.

Far from tuning their value, the algorithm proposed in this section is applied to the default configuration of the routing protocol, (but it may be applied to any other). Thus, LARK estimates an evolution factor m from the current link quality lq_i and the previous one lq_{i-1}. The key to compute m can be easily understood through the graphic in Figure 8.

If applying basic algebraic notions, given two points A (x_2, y_2) and B (x_1, y_1) in Cartesian axis, it is possible to determine the equation of the linear function for any point C (x, y), as Formula 4 shows.

$$y - y_1 = m(x - x_1) \quad | \quad m = \frac{y_2 - y_1}{x_2 - x_1} \tag{4}$$

If replacing points A and B in Formula 4 by (t_i, lq_i) and (t_{i-1}, lq_{i-1}) respectively where t_i represents the current time (T), lq_i is its respective link

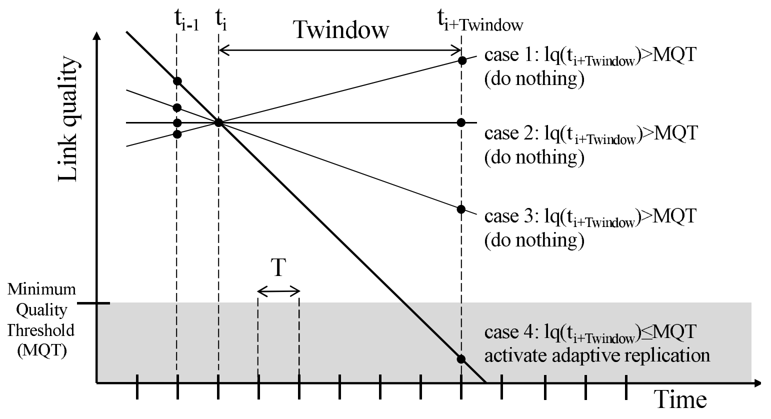

t_{i-1}
t_i
Twindow
t_{i+Twindow}
case 1: lq(t_{i+Twindow})>MQT (do nothing)
case 2: lq(t_{i+Twindow})>MQT (do nothing)
case 3: lq(t_{i+Twindow})>MQT (do nothing)
T
case 4: lq(t_{i+Twindow})≤MQT activate adaptive replication
Link quality
Minimum Quality Threshold (MQT)
Time

Figure 8: Link-quality-based adaptive packet replication technique.

quality, t_{i-1} and lq_{i-1} represent that information for last T, and point C is replaced by ($lq_{i+Twindow}$, $t_{i+Twindow}$) we can obtain Formula 5. According to Formula 5, it is possible to forecast the link quality $lq_{i+Twindow}$ for a given time $t_{i+Twindow}$ according to the trend pointed by m. Then, if the estimated link quality after $Twindow$ time $t_{i+Twindow}$ is lower than the Minimum Quality Threshold (MQT), the replication level R, which counts the number of routing packets to be replicated, increases in a factor δ to keep the link alive regardless the effect of current *ambient noise*. Else if R is greater than 0, it is decreased in δ packets to consider the situation when the network has overcome the risk to remove the link (either because the *ambient noise* has been reduced, or because of the effect of the adaptive packet replication). δ represents the number of packets to be added or subtracted to R. The idea of LARK is reacting (in time) against a possible link removal.

$$lq_{i+Twindow} = m \cdot Twindow + lq_i \quad | \quad m = \frac{lq_i - lq_{i-1}}{T} \tag{5}$$

4.2. Implementation of LARK

LARK is included within the routing manager module of the routing protocol (see Figure 2). Table 4 shows the pseudo-code that has been implemented in C language for each routing protocol considered in this section. The real conditions of the network in practice impose limiting the amount of replicas to N_{max} and *delta* (δ). If considering a very severe *ambient noise*, the fact of sending more and more replicas will only contribute to increase, even more the effect of *ambient noise*. The value of N_{max} has been empirically computed for our deployment to 10 packets in order not to exceed the routing overhead obtained when applying the *batmand*-like configuration beyond 150%. In the same line, *delta* was limited to 1 packet. The fine tuning of these parameters falls out of the scope of this section. However, the approach followed in [17] could be used as a reference to carry out this task.

Table 4: Link-Quality-based Adaptive Packet Replication (LARK).

```
#DEFINE Nmax
#DEFINE delta
float current_lqi, previous_lqi;
int R = 0;
/*every time link quality is computed for the current link*/
for each T
/*forecast link quality*/
lqi_in_Twindow = ((current_lqi - previous_lqi)/T * Twindow) + current_lqi
/*determine the number of replicas to send*/
  if (lqi_in_Twindow <= MQT) then
    if (R < Nmax) then R += delta;
  else
    if (R > 0) then R -= delta;
  /*send the replicas required [0, Nmax]*/
  send_broadcast(R, routingPacket);
  /*save the variables for the next iteration*/
  previous_lqi = current_lqi;
```

As previously stated, *batmand* presents certain limitations like the absence of a MQT. However, given the genericity of LARK, nothing impairs assigning a $MQT = 0$ to *batmand* in LARK, or to any other proactive routing protocol which does not consider its use. LARK is applied before sending a routing packet every time T. Then, the algorithm proposed must obtain the value of lq_i and lq_{i-1}. The value of lq_i can be easily obtained from the current state of the routing manager module. However, not all the protocols consider storing the previous state. Accordingly, the algorithm must store lq_i to provide lq_{i-1} in next iteration of T. This cost is negligible in terms of memory footprint even for the tiny devices considered in our experimentation.

The next step involves computing $lq_{i+Twindow}$ through the expression in Formula 5. In case this value is underneath MQT, the value of R indicating the number of replicated packets that will be sent in T, is progressively increased only if its current value is lower than N_{max}. Otherwise, in case the

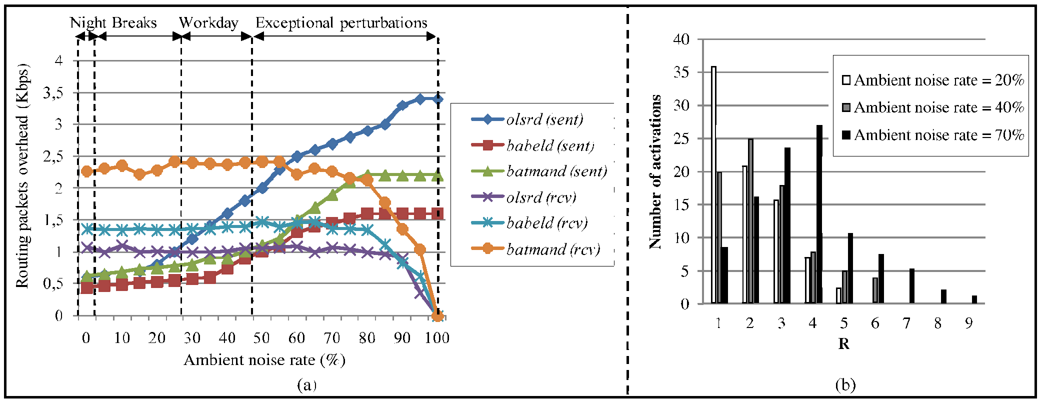


Figure 9: Routing overhead after applying the link-quality-based packet replication (a); and histogram of the replication level (R) for different rates of *ambient noise* in *olsrd* (b).

link has overcome the risk of disappearing, the number of replicas is progressively reduced up to 0, thus restoring the default behaviour of the protocol. The value of R is also stored to increase or decrease it in the following iteration, depending on the state of the link. In any case, all the replicated packets send the same information, so, any packet already received will be discarded. Our goal thus is not sending new packets with further information, but increasing the probability of broadcasting the same information at least once. As all the protocols natively implement the mechanism to discard replicas, no additional strategy has been required to be introduced in LARK in the reception of packets.

Given the simplicity of the operations considered, and the time elapsed between the iterations, the overhead introduced in the protocol in terms of CPU is also negligible (less than 1%).

4.3. Assessing the adaptive replication of packets

Additional experimental campaigns were required to show the effectiveness of the algorithm proposed. Concretely, let us first analyse the Figure 9a, which represents the routing overhead introduced by the routing protocols implementing the algorithm proposed. Basically, when applying LARK, all the routing protocols balance their routing overhead to adapt their behaviour in a context-aware way. In terms of routing information sent, this balance goes from the regular behaviour of the protocol (see Figure 7a) to a behaviour similar to the experienced when applying the *batmand*-like configuration (see Figure 7b). In the first case, no additional routing packet is sent, so the intrusiveness introduced in the network is null. Conversely, when the *ambient noise* increases and the routing protocol requires a major effort to maintain their routes, it is allowed to increment the amount of routing information sent. This behaviour can be observed in the histogram plotted in Figure 9b. This Figure shows the progressive increment of R in LARK as the *ambient noise* rate increments. Such a trend, exemplified in Figure 9b just for *olsrd* given space limitations, is perceived through the most frequent values of R demanded by LARK when its activation was necessary: 1 replica for an *ambient noise* rate of 20%, 2 replicas for an *ambient noise* rate of 40%, and 4 replicas for an *ambient noise* rate of 70%. Trends are similar for the rest of routing protocols. Unlike the regular behaviour of the routing protocols considered, what is constant using LARK is not the rate of routing packets sent, but the rate of received ones. The goal thus, is maintaining the routing capability as longer as possible, even with a severe amount of *ambient noise*. If comparing the new results with the previous routing overhead involving the packets sent (shown in Figure 7b), all the protocols reduce the amount of packets sent up to around 70% packet loss caused by the packet corruption of *ambient noise*. In this condition of extreme necessity for the links survival, the routing protocols using LARK are forced even to increase the routing overhead introduced with respect to the *batmand*-like configuration. Indeed, *olsrd*, *babeld* and *batmand* increment, in average, 13, 15 and 150 percentage points respectively in this aspect. However, the major difference is that now, the route availability increases in these conditions. Beyond this *ambient noise* rate, the routing packets received decrease given the practical bound imposed by N_{max} to limit the packet replication indefinitely.

If taking these results in mind when comparing the route availability obtained when applying LARK, with those provided previously in Figure 5c and Figure 5d (for the default configuration) and Figure 6c and Figure 5d (for the *batmand*-like configuration), the benefits of LARK can be observed. Concretely, the regular behaviour of the targeted routes A-C and A-B is absolutely improved regarding the default configuration in all the protocols (see Figure 10b and Figure 10d respectively). In the case of the *batmand*-like configuration (see Figure 10c), results are very similar for *olsrd* and *babeld* (less than 3% of difference), but taking into account that the routing overhead introduced has been widely reduced for the ranges of breaks and workday (more than 150% in all the cases), where the protocols will operate most of the time. Additionally, it is worth noting that thanks to this technique *batmand* speeds up its reconfiguration time, and consequently increments its

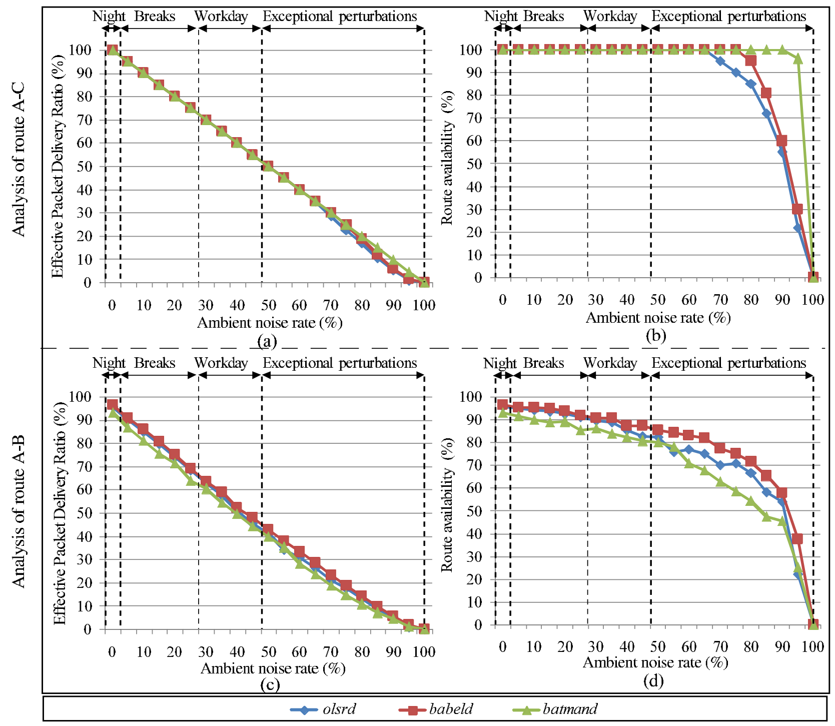


Figure 10: Experimental results after applying the link-quality-based packet replication.

route availability with respect to its default configuration from 5% to 10%.

As we have seen in this Section, dynamically tuning (with an accuracy of a period T) the number of replicated routing packets sent depending on the *ambient noise* rate, leads our technique to optimise the amount of routing packets exchanged (routing overhead). Consequently it is not necessary to perform additional offline studies to determine the proper routing overhead in a given moment or for a given routing protocol.

All these improvements are observed in terms of the packet delivery ratio in Figure 10a and Figure 10c, thus enhancing the general behaviour of the WMN with respect to the regular behaviour of the protocol.

5. Related work

As stated in the introduction of the paper, proactive routing protocols in WMNs are dynamic by nature, which means that they adapt their behaviour to the network circumstances. Such an adaption may take into account criteria ranging from energy saving, as in the case of [18, 19], to the presence of attacks [20]. Yet, in this paper we have focused on *ambient noise* as a key criterion due to its critical influence in mainstream WMNs networks.

Concerning *ambient noise*, one can find different solutions such as [21], [22] or [23]. In these works, the authors propose a multipath routing protocol to calculate channel saturation, thus balancing the load and improving the throughput. However, to monitor and report the required QoS parameters, the nodes implementing these protocols require multiple antennas, which involves an expensive exchange of information between parties as well as an investment in extra hardware. Furthermore, the conclusions released in all these papers are obtained from simulation results.

Alternatively, in the work presented in [24], the authors propose a Session-Oriented Adaptive Routing (SOAR) protocol, which recognises data sessions and prevents path switching while the data session is in progress. To achieve this goal, the protocol does not switch the path in use unless the potential throughput gain of a new path is considered better than a predetermined threshold. Moreover, the new path with lower link cost is also stored and used when a new data session is initiated. Similarly, in [25], an opportunistic-based routing protocol is provided to efficiently detect and retransmit lost packets, and determine the appropriate sending rate according to the current network conditions. However, since the implementation of these routing protocols is not available to the public, they have not been validated by the community as other well-known routing protocols, such as OLSR, Babel and B.A.T.M.A.N.

To face this challenge, some authors have proposed several generic (non-protocol-dependent) noise-aware techniques that can be applied to well-known routing protocols. This is the case of ALARM [26], a generic technique that uses a metric based on the number of packets queued in the wireless interface. However, although this computed value may offer an accurate perception of *ambient noise*, no counter-measure is proposed to improve the network throughput beyond a mere route change. In addition, this technique has been only evaluated on one routing protocol.

In this context, the work presented in this paper contributes to fill the existing gap in the state of the art. The adaptive technique we propose through LARK does not only identifies dynamically the variations of *ambient noise* rate, but it reacts by mitigating the loss of quality in the network links through a simple packet replication. The heuristic on which we base this counter-measure relies on linear prediction of link quality. Despite being simple, it is relatively effective since predictions have a short-term validity (indeed every period T a new estimation is provided). However, in a near future, we ambition to evaluate other alternatives based on regression models [27] or interpolation methods. The interest of our proposal does not lie in determining thresholds or window-based decision making. Instead, it is on (i) discovering that, despite the heterogeneous nature of proactive routing protocols, all of them share the same key characteristics based on thresholds and window-based decision making; and (ii) using these parameters to create generic mechanisms that can automate their tuning in a dynamic way, thus reducing the intrusiveness in the original source code of routing protocols. Figure 11 synthesises this idea. Such a genericity has been proven through three different (and real implementations of) proactive routing protocols such as OLSR, Babel and B.A.T.M.A.N.

In particular, this manuscript extends our study initiated in [28] to introduce improvements with respect to the notion of *ambient noise*, the algorithmic technique developed and the experiments to validate our results.

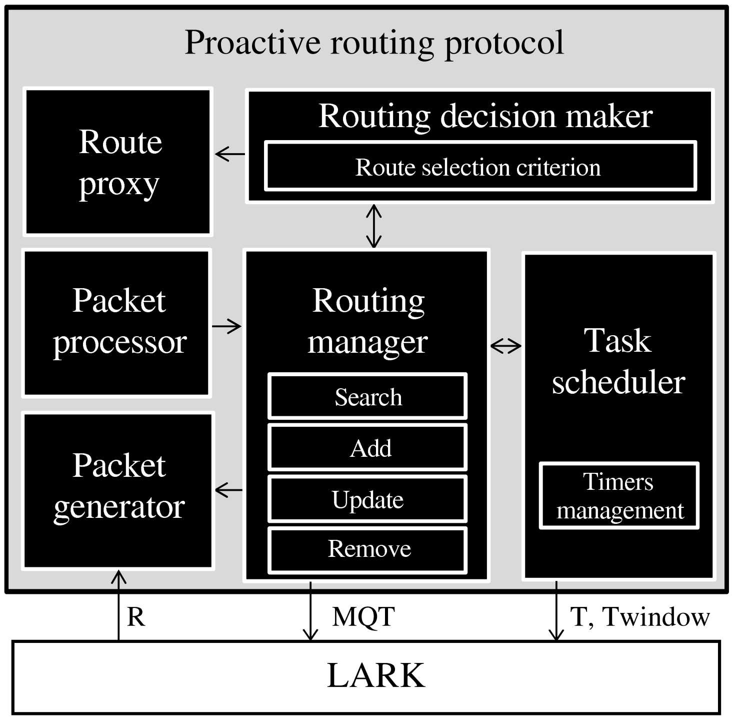


Figure 11: Relation between proactive routing protocols and LARK.

6. Conclusion

The contribution in this paper is twofold. On the one hand, after analysing three state-of-the-art protocols (OLSR, Babel and B.A.T.M.A.N), this paper concludes that regardless of the protocol type, acting on the configuration parameters in charge of links management can improve the network robustness against *ambient noise*. It has been shown that, under similar configurations, the differences between considered protocols mainly lie on the overhead they induce in the network rather than in their protection capabilities. However, as far as their configuration must be statically tuned offline, adapting this level of overhead to the changing conditions of *ambient noise* in the network is not easy. Indeed, a high level of overhead may not be desirable, and it may be seen as unaffordable, when *ambient noise* does not exist or it is very low.

So, the second contribution provided in this paper goes in the direction of improving the network converge time resulting from a high level of *ambient noise*, while reducing the cost of routing overhead. Although results have been obtained from OLSR, Babel and B.A.T.M.A.N, a number of conclusions can be generalised and applied to any type of proactive routing protocol incorporating similar link-quality-based parameters. From such ideas, a novel strategy to fight against *ambient noise* is proposed as a complement to existing solutions. Concretely, its novelty is on promoting the dynamic adaptiveness of the routing protocol to the network environment to determine the optimum amount of routing information that must be exchanged among nodes in a given moment.

Given its genericity, this approach could be of interest not only to WMNs in particular, but also to any type of ad hoc network (sensor networks, mobile ad hoc networks or vehicular ad hoc networks) in a wide range of contexts of use. Reengineering the proposed link-quality-based replication algorithm as an aspect has the potential of improving its portability to other different proactive routing protocols. Aspect-Oriented Programming (AOP) has already showed its value in other contexts of use of ad hoc networks [29]. However, the complexity of redeploying LARK using AOP is something that goes beyond the scope of this paper.

References

[1] M. Raya, J.-P. Hubaux, I. Aad, Domino: a system to detect greedy behavior in ieee 802.11 hotspots, in: Proceedings of the 2nd interna-

tional conference on Mobile systems, applications, and services (MobiSys), 2004, pp. 84–97.

[2] X. Ni, K.-c. Lan, R. Malaney, On the performance of expected transmission count (etx) for wireless mesh networks, in: Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools (ValueTools), 2008, pp. 77:1–77:10.

[3] K. Ramachandran, I. Sheriff, E. Belding, K. Almeroth, Routing stability in static wireless mesh networks, in: Proceedings of the 8th international conference on Passive and active network measurement (PAM), 2007, pp. 73–83.

[4] J. Geibig, D. Bradler, Self-organized aggregation in irregular wireless networks, in: Wireless Days, 2010, pp. 1–7.

[5] G. Feng, F. Long, Y. Zhang, Hop-by-hop congestion control for wireless mesh networks with multi-channel mac, in: Proceedings of the 28th IEEE conference on Global telecommunications (GLOBECOM), 2009, pp. 242–246.

[6] N. Tcholtchev, R. Chaparadza, Autonomic Fault-Management and resilience from the perspective of the network operation personnel, 2010.

[7] T. Clausen and P. Jacquet, Optimized Link State Routing Protocol(OLSR), RFC 3626 (2003).

[8] J. Chroboczek, BABEL, [Online]. Available: http://www.pps.jussieu.fr/ jch/software/babel/, 2011.

[9] OpenMesh, Open Mesh, Better Approach To Mobile Ad hoc Networking (B.A.T.M.A.N.), [Online]. Available: http://www.open-mesh.net/, 2011.

[10] Y. He, R. Yuan, X. Ma, J. Li, Analysis of the impact of background traffic on the performance of 802.11 power saving mechanism, IEEE Communications Letters 13 (2009) 164 –166.

[11] H. Lee, A. Cerpa, P. Levis, Improving wireless simulation through noise modeling, in: Proceedings of the 6th international conference on Information processing in sensor networks (IPSN), 2007, pp. 21–30.

[12] B. Fu, G. Bernath, B. Steichen, S. Weber, Wireless background noise in the wi-fi spectrum, in: 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), 2008, pp. 1 –7.

[13] X. Su, R. V. Boppana, On the impact of noise on mobile ad hoc networks, in: Proceedings of the 2007 international conference on Wireless communications and mobile computing (IWCMC), 2007, pp. 208–213.

[14] Iperf, Iperf tool, [Online]. Available: http://sourceforge.net/projects/iperf/, 2014.

[15] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, M. Singh, Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols, in: IEEE Wireless Communications and Networking, volume 3, 2005, pp. 1664 – 1669.

[16] Netem, Network Emulator (Netem), [Online]. Available: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem, 2014.

[17] J. Friginal, D. de Andres, J.-C. Ruiz, P. Gil, Resilience-driven parameterisation of ad hoc routing protocols: olsrd as a case study, in: Proceedings of the 2011 IEEE 30th International Symposium on Reliable Distributed Systems, SRDS '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 85–90.

[18] A. Akhtar, M. Nakhai, A. Hamid Aghvami, Energy-efficient adaptive routing in wireless ad hoc and mesh networks, IET Networks 1 (2012) 249–256.

[19] T. T. Vinh, T. N. Quynh, M. B. T. Quynh, Emrp: Energy-aware mesh routing protocol for wireless sensor networks, in: 2012 International Conference on Advanced Technologies for Communications (ATC), 2012, pp. 78–82.

[20] H. Silva, R. Holanda, M. Nogueira, A. Santos, A cross-layer and adaptive scheme for balancing performance and security on wmn data routing, in: 2010 IEEE Global Telecommunications Conference (GLOBECOM 2010), 2010, pp. 1–5.

[21] D. Narayan, R. Nivedita, S. Kiran, M. Uma, Congestion adaptive multipath routing protocol for multi-radio wireless mesh networks, in: 2012 International Conference on Radar, Communication and Computing (ICRCC), 2012, pp. 72–76.

[22] B. Shin, S. Y. Han, D. Lee, Dynamic link quality aware routing protocol for multi-radio wireless mesh networks, in: 2012 IEEE 26th International Conference on Advanced Information Networking and Applications (AINA), 2012, pp. 44–50.

[23] Y. Ding, K. Pongaliur, L. Xiao, Channel allocation and routing in hybrid multichannel multiradio wireless mesh networks, IEEE Transactions on Mobile Computing 12 (2013) 206–218.

[24] Y.-J. Son, S.-H. Lee, Y.-B. Ko, S.-J. Park, Y.-S. Song, Session-oriented adaptive routing for improving path stability in wlan based mesh networks, in: 2011 13th International Conference on Advanced Communication Technology (ICACT), 2011, pp. 123–126.

[25] E. Rozner, J. Seshadri, Y. Mehta, L. Qiu, Soar: Simple opportunistic adaptive routing protocol for wireless mesh networks, IEEE Transactions on Mobile Computing 8 (2009) 1622–1635.

[26] A. A. Pirzada, R. Wishart, M. Portmann, J. Indulska, Alarm: An adaptive load-aware routing metric for hybrid wireless mesh networks, in: Proceedings of the Thirty-Second Australasian Conference on Computer Science - Volume 91, ACSC '09, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 2009, pp. 37–46.

[27] S. Priya, Adaptive control of routing protocol in mobile ad-hoc network using regression model, in: 2012 International Conference on Emerging Trends in Science, Engineering and Technology (INCOSET), 2012, pp. 509–514.

[28] J. Friginal, J.-C. Ruiz, D. de Andres, A. Bustos, Mitigating the impact of ambient noise on wireless mesh networks using adaptive link-quality-based packet replication, in: 2012 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2012, pp. 1–8.

[29] A. Bustos, J. Friginal, D. de Andrés, J.-C. Ruiz, An aspect-oriented approach to face neighbour saturation issues in proactive ad hoc routing protocols: olsrd as a case study, in: Proceedings of the 1st European Workshop on AppRoaches to MObiquiTous Resilience, ARMOR '12, ACM, New York, NY, USA, 2012, pp. 3:1–3:6.