Generalizing Wireless Ad Hoc Routing for Future Edge Applications *

André Rosa Pedro Ákos Costa João Leitão
NOVA LINCS & DI/Nova School of Science and Technology, UNL, Portugal
{af.rosa,pah.costa}@campus.fct.unl, jc.leitao@fct.unl.pt

Abstract

Wireless ad hoc *networks are becoming increasingly relevant due to their suitability for Internet-of-Things (IoT) applications. These networks are comprised of devices that communicate directly with each other through the wireless medium. In applications deployed over a large area, each device is unable to directly contact all others, and thus they must cooperate to achieve multi-hop communication. The essential service for this is* Routing*, which is crucial for most applications and services in multi-hop* ad hoc *networks. Although many wireless routing protocols have been proposed, no single protocol is deemed the most suitable for all scenarios. Therefore, it is crucial to identify the key differences and similarities between protocols to better compare, combine, or dynamically elect which one to use in different settings and conditions. However, identifying such key similarities and distinctions is challenging due to highly heterogeneous specifications and assumptions. In this paper, we propose a conceptual framework for specifying routing protocols for wireless* ad hoc *networks, which abstracts their common elements and that can be parameterized to capture the behavior of particular instances of existing protocols. Furthermore, since many wireless* ad hoc *routing protocols lack systematic experimental evaluation on real networks, we leverage an implementation of our framework to conduct an experimental evaluation of several representative protocols using commodity devices.*

1 Introduction

In recent times, we have been witnessing the emergence of the *Internet-of-Things (IoT)*: ubiquitous networks of interconnected everyday objects (e.g., vehicles, buildings, household appliances) capable of performing computations and exchanging data with other devices [1, 31]. A vast amount of IoT applications depends on Cloud services, and their deployments rely on infrastructure-based wireless networks [41]. This architecture, however, is becoming unsuitable for several IoT scenarios due to its inherent limitations. On the one hand, the ever-increasing amounts of data produced and consumed by IoT devices are rendering the Cloud unable to collect, process, and reply promptly as well as increasing the operational costs [11]. On the other hand, while infrastructure-based wireless networks provide fairly reliable, high-speed, and high-bandwidth links, they also inhibit the flexibility of applications since they constrain the mobility of devices and require attention to their deployment, configuration, and relocation.

The demand to offload computations from the Cloud motivates a paradigm shift towards Edge Computing [25], which exploits the computational capabilities of peripheral network devices that are located near end-users. In this sense, *wireless ad hoc networks*, i.e., decentralized set of devices that communicate directly through the wireless medium without relying on any pre-existent infrastructure, emerge as a more flexible and robust platform than infrastructure-based wireless networks for materializing Edge Computing in the context of IoT. These networks are suitable for situations with inadequate, inexistent, unavailable, or debilitated network infrastructures [1, 31], such as: rescue/support on natural disasters; environmental monitoring; autonomous vehicles; and smart cities or homes. As such, IoT has been inducing the contemporary reemergence of wireless *ad hoc* networks.

*This work was partially supported by NOVA LINCS (FC&T grant UIDB/04516/2020) and NG-STORAGE (FC&T grant PTDC/CCI-INF/32038/2017). Fractions of this report has been published in EAI Mobiquitous'21.

On these networks, the devices, also called nodes, are typically scattered through a wide area, being unable to communicate directly with all the others, forming a multi-hop network. Consequently, they must cooperate, by retransmitting messages on behalf of other nodes, so that communication can be achieved among all devices. This essential service is named *Routing*, enabling point-to-point communication by message forwarding among nodes. A plethora of routing protocols for wireless *ad hoc* networks have been already proposed over the years, exploring and combining different techniques. Nonetheless, due to these networks' highly dynamic and heterogeneous nature, no single protocol is deemed the most suitable for all scenarios. Therefore, it is crucial to identify how the different protocols relate to each other to better compare, combine, or dynamically select them. However, uncovering the relations among them is rather challenging due to heterogeneous specifications and assumptions. This observation motivated us to devise a framework for specifying routing protocols for these networks, which abstracts their common elements while offering parameters to materialize particular instances.

In addition, the vast majority of routing protocols have only been evaluated through simulations [3,9,10], since they provide an accessible, inexpensive, and controlled evaluation environment. Nonetheless, even the most detailed simulations are unable to capture the particular characteristics of real wireless *ad hoc* environments [2,5], usually employing inaccurate models, not considering hardware limitations of wireless interfaces, or ignoring external sources of interference in the wireless medium. Although real testbeds have been employed in the past to evaluate some protocols [20, 22], they generally resort to grid topologies with equidistant nodes and without external interference, which is highly unrealistic; or consist of few nodes (less than 10), which are not enough to derive significant conclusions. Therefore, leveraging an implementation of our routing framework, we conducted an experimental evaluation of five representative routing protocols on a real wireless *ad hoc* network formed by commodity devices.

The remainder of this paper is structured as follows: Section 2 analyzes routing in wireless *ad hoc* networks; Section 3 delves into our framework; Section 4 presents the details of our experimental evaluation; Section 5 briefly discusses the related work; and Section 6 concludes the paper with some final remarks.

2 Routing in Wireless Ad Hoc

A plethora of routing protocols has been proposed throughout the years, exploring different techniques to increase the robustness and efficiency of network-wide communications. These protocols are categorized mainly by their route provision strategy as proactive [6,7,27], reactive [18,29,40], or hybrid [16,28,30]. However, in this paper, we make an effort to better characterize these protocols down to their fundamental operation, going beyond the employed route provision strategy. In this sense, the operation of routing protocols can be divided into two complementary parts, the *route computation* and the *message forwarding*.

2.1 Route Computation

Computing routes is the main concern of routing protocols and hence encompass a variety of essential components, which include *discover* a node's *neighbors* (i.e., nodes with whom the local node can directly exchange messages), *identify* the *cost* of direct communication, apply *distributed strategies* to actually *compute routes*, and *disseminate information* to inform other nodes of existing routes.

At the basis of any routing protocol is *neighbor discovery*, essential for computing routes as it provides each node with information about the other nodes which can be directly reachable by itself. However, routing protocols must ensure some Quality of Service (QoS), thus neighbor discovery must obtain properties of the wireless links between neighboring nodes. One of such properties is the bidirectionality of communication [6,7], i.e., both nodes can send and receive messages from each other, as it is often crucial to ensure two-way communication.

In addition, routing protocols require *cost* metrics to select the best routes, as in general there might be multiple available routes from each node to each destination. The cost of a route is a function of its constituent link's costs, usually the sum [6]. However, other functions can be employed [27, 40]. These metrics can be in their simplest form the number of hops towards the destination, or incorporate properties of the links, such as the link's expected number of transmissions to deliver a message (ETX) [19, 23], the link's expected transmission time (ETT) [12], the link's stability [13, 27, 40], the congestion of the nodes [24], or the energy spent using the link [37]. In this sense, routing protocols resort to a *cost function* that evaluates the local links and is used to qualify each route.

The process of actually computing routes requires the distributed cooperation of nodes and leverages each node's local neighborhood information. In this sense, there are three main *computation strategies* to distributively construct routes: *i*) *distance-vector* [6, 17, 18, 29], where each node announces the cost of its best route towards a given destination, allowing the other nodes to assign as next-hop the neighbor which provides the best route; *ii*) *link-state* [7, 15, 39], where nodes gather, through collaborative dissemination, the complete, or a connected sub-set, of the network topology, and locally compute the best routes to all reachable destinations; and *iii*) *link-reversal* [14, 28, 30], where the nodes distributively construct a directed acyclic graph (DAG) over the network topology for each destination, with each directed path in the DAG correspondig to a route to the destination. Note that these high-level strategies can be further specialized to better fit specific routing protocols, which we discuss in more detail in the next section. Since routes are computed in a distributed way, each node does not have to be aware of the complete routes, only their next-hops. This information is typically encoded by the routing strategies in a conceptual data structure, local to each node, called the *routing table* [7, 17]. In some protocols, these tables may contain additional information [18] or even not be used at all [21].

Finally, routing protocols need to propagate control messages throughout the network to enable the computation of routes through the use of some computation strategy. Across the literature, routing protocols employ several *dissemination strategies*, even within the same protocol, which can be grouped into specific communication patterns according to the nature and intended destinations of such messages into: *network-wide* [7, 27, 29] or *limited-hop broadcast* [6, 16], to inform all or a sub-set of the nodes; *bordercast* [16], to inform a specific sub-set of the nodes; or (*network-wide*) *unicast* [18, 29] to inform a single node.

2.2 Message Forwarding

Besides route computation, routing protocols are also responsible for leveraging the computed routes to forward applicational messages. To this end, routing protocols employ different *forwarding strategies* that provide different trade-offs across reliability and communication overhead.

The simplest strategy is to forward to the next-hop contained in the routing table. However, other strategies can be found in the literature. For instance, *multipath* protocols [26] leverage several routes to the same destination to increase the chances of delivering messages. Alternatively, *opportunistic* protocols [4, 35] employ coordination mechanisms to dynamically elect, from a set of candidate next-hops, the one which will proceed with the forwarding of each message. As another option, *geographic* protocols [21,32] use the nodes' coordinates to base their forwarding decisions. Furthermore, *source* routing protocols [18] do not require each route's intermediary nodes to maintain information regarding the route. Instead, the nodes which originate messages maintain the complete routes, which are then carried within the messages to allow the intermediary nodes to retrive their next-hop to whom they will forward it. Finally, some protocols [6, 29, 40] rely on the explicit or implicit *acknowledgment* and retransmission of messages to increase the reliability of their forwarding strategies.

3 Routing Framework for Wireless Ad Hoc Networks

In this section, we present our conceptual routing framework, which captures a broad spectrum of existing routing protocols for wireless *ad hoc* networks. The framework's design follows directly from the observations made in Section 2. In the following, we present an overview of the workflow of events in a generic routing protocol, which lies at the core of our framework. In addition, we also present the notation used to specify routing protocols using our framework and illustrate this using a representative set of existing routing protocols.

3.1 Overview

Figure 1 illustrates the execution flow captured by our framework, which is divided into two parts: the *control plane*, responsible for computing routes, and the *forwarding plane*, responsible for forwarding applicational messages.

3.1.1 Control Plane

The control plane is responsible for processing internal events to manage the routing strategy. There are three main entry points in this plane: a *neighborhood update*, that is processed by the ***cost function***; the set off of

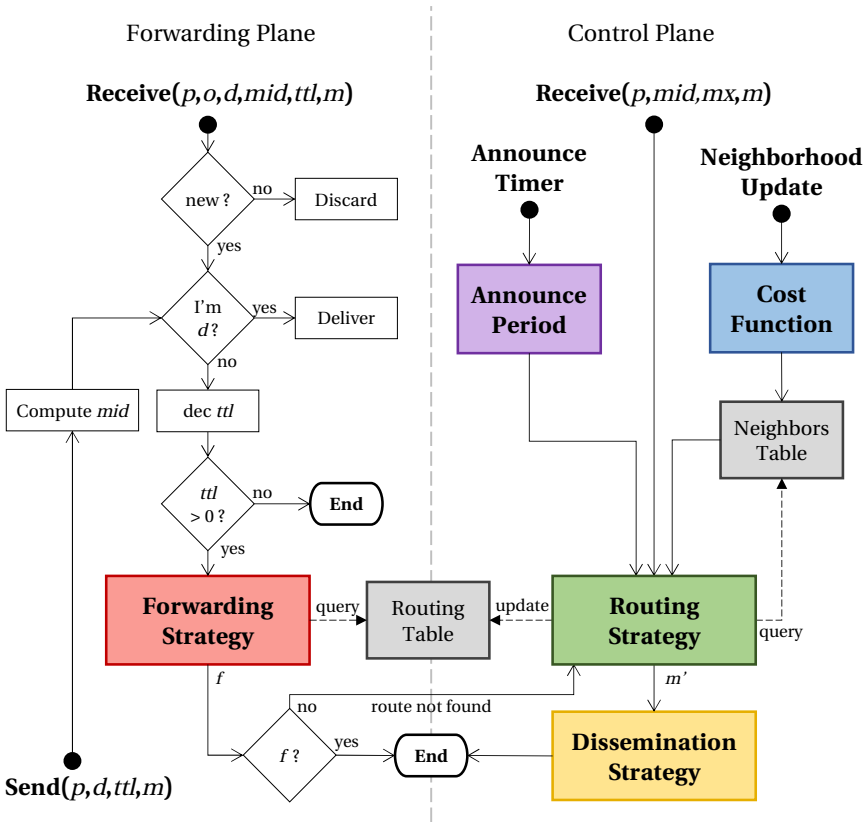


Figure 1: Routing Framework Execution Flow.

an *announce timer*, that is processed by the ***announce period***; and the reception of a *control message*, that is directly processed by the ***routing strategy***.

Neighborhood updates are assumed to be provided by an external discovery protocol that is outside the scope of this paper. However, a neighborhood update must encode either the discovery, suspicion of failure, or an update to the state of a communication link with a neighboring node. This update is processed by the ***cost function***, which assigns a cost metric to that neighbor. Next, the framework updates its internal neighborhood table containing essential information for each neighbor, such as the link cost and bidirectionality and the neighbor's address, which is leveraged by the ***routing strategy*** to compute routes.

The announce timer is a periodic timer that informs the ***routing strategy*** to disseminate a new control message, and is employed by protocols following a proactive or hybrid routing strategy. When the announce timer is triggered, it is first processed by the ***announce period*** which is responsible to reset it. This enables the usage of dynamic periods for announcements [33].

The reception of a control message is immediately processed by the ***routing strategy***. A control message is composed of four parts: (p, mid, mx, m), respectively, the identifier of the node that generated the control message, the message's unique identifier, metadata obtained from the message's propagation and which is associated with a specific dissemination strategy, and the message payload that encodes data specific to the routing strategy.

These three events flow into the main component of the control plane: the ***routing strategy***. The routing strategy evaluates these events, which may lead into an update on the routing table and/or the dissemination of a new control message. On either case, the routing strategy may query the framework's internal neighborhood table to obtain cost metrics and bidirectionality information to compute or update a new or existing route. Finally, in the case a new control message is to be disseminated, the framework delivers the message to the ***dissemination strategy*** that is responsible to send it to all intended destinations.

3.1.2 Forwarding Plane

The forwarding plane is responsible for handling the flow of applicational messages and applying a ***forwarding strategy***, which can be triggered by two events: a *request to send* a message to an arbitrary destination, or the *reception* of a forwarded message from a neighboring node.

To request the forwarding of a message, the application must provide the following parameters: (p, d, ttl, m), respectively, the local node's identifier, the identifier of the destination node, a time-to-live, and the message payload. Upon receiving such request, the framework first generates a message identifier (mid) to uniquely identify the message in the network. Then, the message enters the flow of received messages in the forwarding plane.

Upon the reception of a message, the framework verifies if the message was already processed, discarding it if so. Next, if the destination of the message is the local node, it is delivered to the application, continuing otherwise to the next processing stage, where the ttl is decremented and verified. If the ttl has expired, the forwarding of the message ends, otherwise the message is delegated to the ***forwarding strategy*** which will obtain the next-hop, potentially consulting the framework's routing table, and send the message to it. If no next-hop was found or the message could not be successfully forwarded, the routing strategy in the control plane is notified, possibly requesting the dissemination of a new control message, such as a route request in reactive protocols [18, 29]. Otherwise, the workflow for that message ends.

3.2 Framework Parameters

Our framework represents a generic (or meta) routing protocol that can be parameterized to express a multitude of different protocols with different properties and strategies. To specify a routing protocol in our framework, one only has to define five parameters: (FS, AP, CF, RS, DS), where FS is the forwarding strategy, AP is the announce period, CF the cost function, RS the routing strategy, and DS is the dissemination strategy. In the following we discuss some alternatives of possible values for these parameters, being that they can also assume a value of $\perp$ to encode not employing a specific parameter.

Forwarding Stategies are responsible for selecting the next-hop, and forwarding to it, any applicational message. These strategies can be Simple, where it simply retrieves the first next-hop contained in the routing table; Multipath, where instead multiple next-hops are retrieved from the routing table and one is selected according to some criteria; Source, where the complete route is retrieved and piggybacked in the message, allowing intermediary nodes to become aware of their next-hops; Acked(s), that extends a strategy s with explicit acknowledgments and retransmissions of forwarded messages; Opportunistic, where the next-hop is dynamically selected; and Geographic where the next-hop is chosen as the neighbor geographically closer to the destination.

Announce Period is a natural number t that represents the interval between periodic announcements of control messages. This value can be the result of a function when the protocol employs dynamic periods.

Cost Functions assign cost metrics to links to qualify the routes, and include: HOPS, which is always 1 so that the routes' cost is their number of hops; ETX, which estimates the expected number of retransmissions for a successful forwarding; ETT, which estimates the expected time for a successful forwarding; AGE, where the time elapsed since the neighbor was detected is used to estimate the cost, with older neighbors representing better links (i.e., more stable); DIST, which uses the geographic distance between the local node and the neighbor, with higher costs representing better links (i.e., closer to the destination); and MCX, where the number of control messages received from different sources and neighbors is considered, with higher counts representing better links.

Routing Strategies are responsible for computing routes, and include: LinkState, that periodically disseminates a small sub-set of the known topology, allowing all nodes to gather the global topology which is used to locally compute routes to all reachable destinations. MultiDistVec, which disseminates a portion of the local routing table containing all known destinations and associated costs, allowing each node to select the best routes to each destination. SingleDistVec(m), which disseminates the local node's identify across the network, with the m parameter controling if this dissemination is proactive (pro) or reactive (re), and uses information regarding the path taken by the control messages to calculate routes to the origin node. LinkReversal(m), that distributively constructs a DAG directed to the local node, with the m parameter encoding if the DAG's construction is triggered proactively (pro) or reactively (re). And Zone(i, o, r), where a proactive routing strategy i is employed within routing zones with a limited scope of r hops, and a reactive strategy o is employed to compute routes towards nodes outside of these zones.

Label	Ref	FS	AP	CF	RS	DS
OLSR	[7]	Simple	5	ETX	LinkState	Bcast(∞)
FSR	[15]	Simple	5	ETX	LinkState	Bcast(1)
BABEL	[6]	Simple	5	ETX	MultiDistVec	Bcast(1) $\cup$ Bcast(∞)
BATMAN	[27]	Simple	5	MCX	SingleDistVec(pro)	Bcast(∞)
JOKER	[35]	Opportunistic	5	MCX	SingleDistVec(pro)	Bcast(∞)
AODV	[29]	Simple	$\perp$	ETX	SingleDistVec(re)	Bcast(∞) $\cup$ Ucast
DSR	[18]	Source	$\perp$	ETX	SingleDistVec(re)	Bcast(∞) $\cup$ Ucast
ABR	[40]	Simple	$\perp$	AGE	SingleDistVec(re)	Bcast(∞) $\cup$ Ucast
ZRP	[16]	Simple	5	ETX	Zone(i, o, r)	Bcast(r) $\cup$ Bordercast $\cup$ Ucast
TORA	[28]	Simple	$\perp$	$\perp$	LinkReversal(m)	Bcast(∞) $\cup$ Bcast(1)
GPSR	[21]	Geographic	$\perp$	DIST	$\perp$	$\perp$

Table 1: Specification of Routing Protocols.

Dissemination Strategies are responsible for disseminating control messages to their intended destinations. These can be: Bcast(h), where control messages are broadcast throughout the entire network if h is ∞, or up to limited number of hops h, otherwise. Bordercast, where messages are disseminated to the nodes at the border of routing zones. And Ucast, where messages are sent to a single destination, leveraging previously discovered routes.

With these parameters, we can define a large number of protocols found in the literature. Table 1 contains an illustrative set of examples. The values of the AP column are in seconds. In the next section, we present our experimental evaluation resorting to some of these protocols.

4 Experimental Evaluation

In this section, we present our evaluation work resorting to an experimental assessment of representative routing protocols found in the literature, and implemented using a prototype of our proposed framework, that follows the execution flow previously presented in Figure 1. In the following we detail our experimental setting, followed by the presentation of the experimental results.

4.1 Experimental Setting

The framework, all its modules, and the companion discovery and broadcast protocols were implemented in the C programming language resorting to the Yggdrasil framework [8]. Our framework operates over WiFi (802.11b/g/n standard at 2.4 GHz), without any MAC or PHY changes. We selected the five most well-known representative routing protocols to evaluate: OLSR, BABEL, BATMAN, AODV, and DSR, which were configured as indicated in Table 1. The first three protocols are proactive, employing different routing strategies, and the other two are reactive, employing distinct forwarding strategies. Due to lack of space, we omit further descriptions of these protocols. The interval of the periodic announcements of the companion discovery protocol were configured with a value of 5 seconds, to minimize the contention and collisions in the wireless medium.

The experimental evaluation was conducted in a wireless *ad hoc* network composed of 17 Raspberry Pi 3 - model B, that were dispersed through the rooms and hallways (with approximately 30 meters) of our department building across two floors, as schematically illustrated in Figure 2.

Each node executed one of the routing protocols, a companion discovery protocol, a companion broadcast protocol, and a simple ping application for a period of 10 minutes, including grace periods of 2 minutes at the beginning and end. The ping application, at every second, requests to the routing protocol to send a message to a randomly selected destination (other than the local node), which upon the reception replies with the same message to the source node. This behavior allows to evaluate the selected routes in both directions.

For each routing protocol, we measured its *reliability*, as the ratio of messages that were successfully received back; its *latency*, as the average round-trip-time (RTT) of each message; and its *communication overhead*, as the

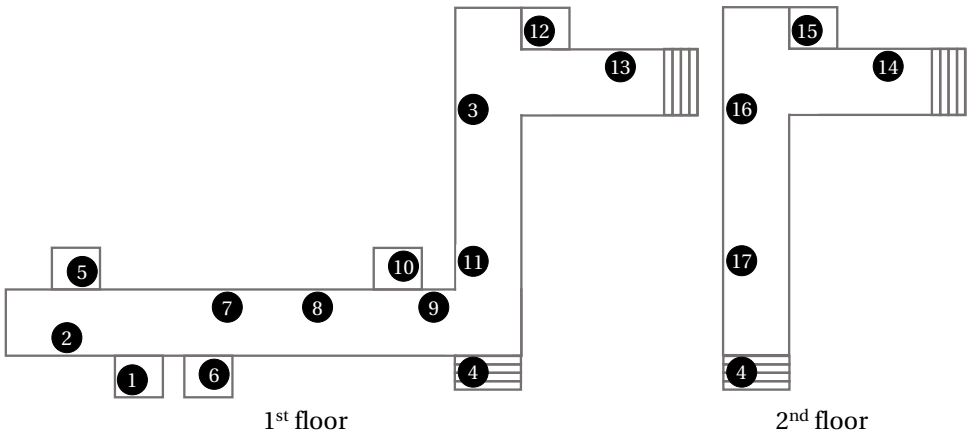


Figure 2: Network Deployment.

total number of transmissions incurred by the dissemination of control messages by the routing protocol and all companion protocols.

We evaluated each protocol in four scenarios: one without node faults and three with deterministic node faults of the first two nodes, five nodes, and nine nodes from the sequence 3, 12, 7, 9, 11, 2, 5, 10, 14. In the experiments with faults, these were introduced simultaneously at the middle point of the experiment (5 minutes). Furthermore, the nodes configured to fail were never selected to be the destination of messages as to not affect reliability measurements. Each experiment was executed three times, in a random order, and the results show the average of all runs. Next, we present and discuss the experimental results.

4.2 Experimental Results

Figure 3 presents in each plot the results for the reliability, represented in the y axis, discriminated by node and on average (the last set of columns) represented in the x axis. Overall, all protocols achieved a reliability above 85% in all scenarios, with the proactive protocols (OLSR, BABEL, and BATMAN) achieving higher reliability than the reactive ones. This is explained by the nodes dropping requested messages while route computation is being performed and routes being constantly broken and re-computed due to unstable neighbors, whose impact is mitigated in proactive solutions since routes are continuously updated. BABEL was the protocol that achieved higher reliability on average, in all scenarios. The reason for this behavior is that, among the proactive protocols, BABEL was the one with the lowest overhead (discussed further ahead) and, as such, this lead to less interferences in the wireless medium causing less message losses.

Among the reactive protocols, DSR achieved a slightly higher reliability than AODV in all scenarios. We suspect this behavior was caused by unstable neighborhood relations that induced the routes to break, leading the intermediary nodes in AODV to remove the routes from their routing tables. This instability impacted DSR less since the full routes are carried within the messages.

We note that, as the number of failures increases so does the reliability of the protocols. This is due to the fact that the resulting network after the faults had more stable paths, had less unstable redundant paths, and less interference between the nodes. Furthermore, in the scenario with two faults (Figure 3b), we note that BATMAN had significantly lower reliability when compared to the other scenarios. This was caused by the emergence of a high number of short-lived routing loops. These loops emerge since BATMAN's routing strategy has no loop prevention mechanism and the combination of BATMAN's cost function and dissemination strategy, allied with unstable neighborhoods, cause the nodes to frequently change their selected next-hops.

Figure 4 presents in each plot the average latency in milliseconds (ms) in the y axis, across all nodes and on average (the last set of columns), represented in the x axis. Overall, all protocols achieved a latency below 35 ms in all scenarios, with approximately the same average latency per scenario. The reason behind these results is that all the protocols converged to the same routes being selected (with approximately 2.1 hops on average) since almost all protocols used the same cost metric and the diversity of available routes to compute in our network deployment was small. The exception was BATMAN, consistently being the protocol with the highest latency,

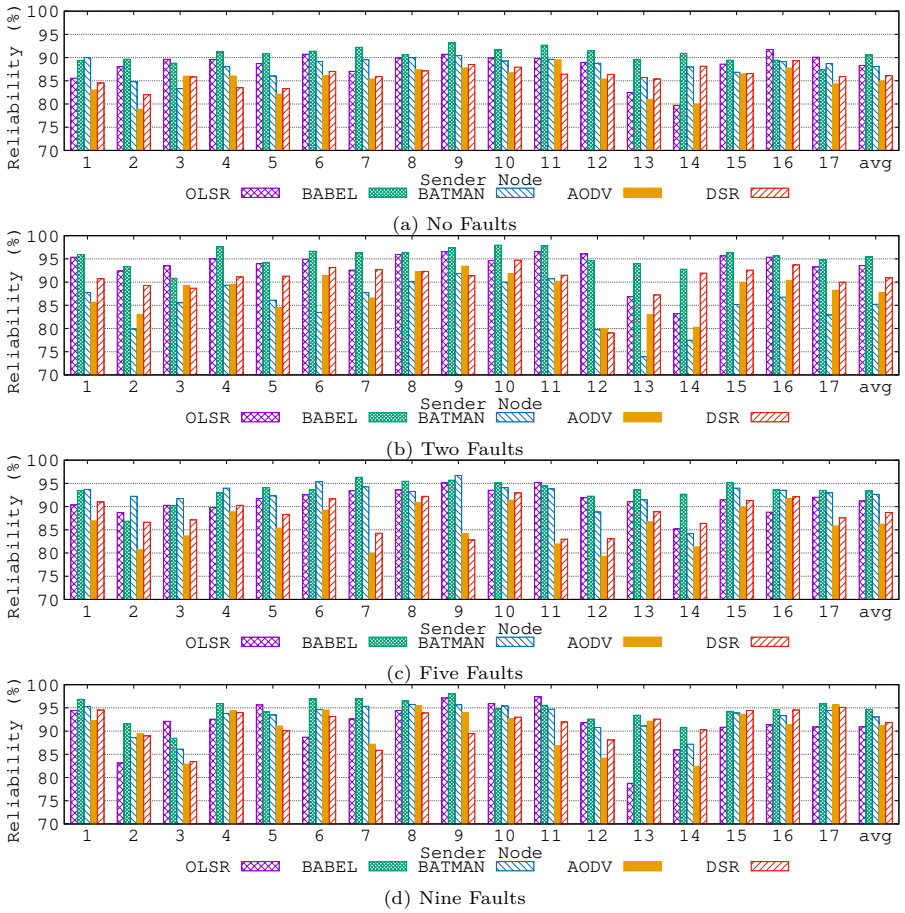


Figure 3: Reliability of Routing Protocols.

due to the formation of short-lived routing loops that were observed during the experiment across all scenarios.

Figure 5 presents in each plot the results of the total communication overhead, represented in the y axis, for each protocol, represented in the x axis. The overhead is discriminated into three types: the *discovery overhead*, as the number of transmissions incurred by the discovery protocol, the *broadcast overhead*, as the number of transmissions incurred by the broadcast protocol, and the *routing overhead*, as the number of transmissions incurred to disseminate control messages with unicast. We begin to note that, as the number of failures increases, the overhead decreases as fewer nodes disseminate control messages. Overall, OLSR presented the highest overhead since its routing strategy triggered the dissemination of unscheduled control messages whenever the selected sub-set of the topology to disseminate (with broadcast) changed, which frequently happened due to unstable neighborhood relations.

The reactive routing protocols, AODV and DSR, presented the lowest overhead across all scenarios. This is the result of caching eavesdropped routes destined to other nodes which allows less route requests to be disseminated.

The BATMAN protocol has the second highest overhead, which is fundamentally caused by the constant periodic broadcasting of a node's identity. In addition, BATMAN's neighbor discovery process is merged with the dissemination of such control messages, allowing to have no additional discovery overhead.

5 Related Work

In this section, we discuss the related work on systematizing routing protocols for wireless *ad hoc* networks. Throughout the literature, not many authors have attempted to performe such task. Nonetheless, a few exceptions can be found.

The Independent Zone Routing (IZR) [34] framework enables the hybridization of proactive and reactive pro-

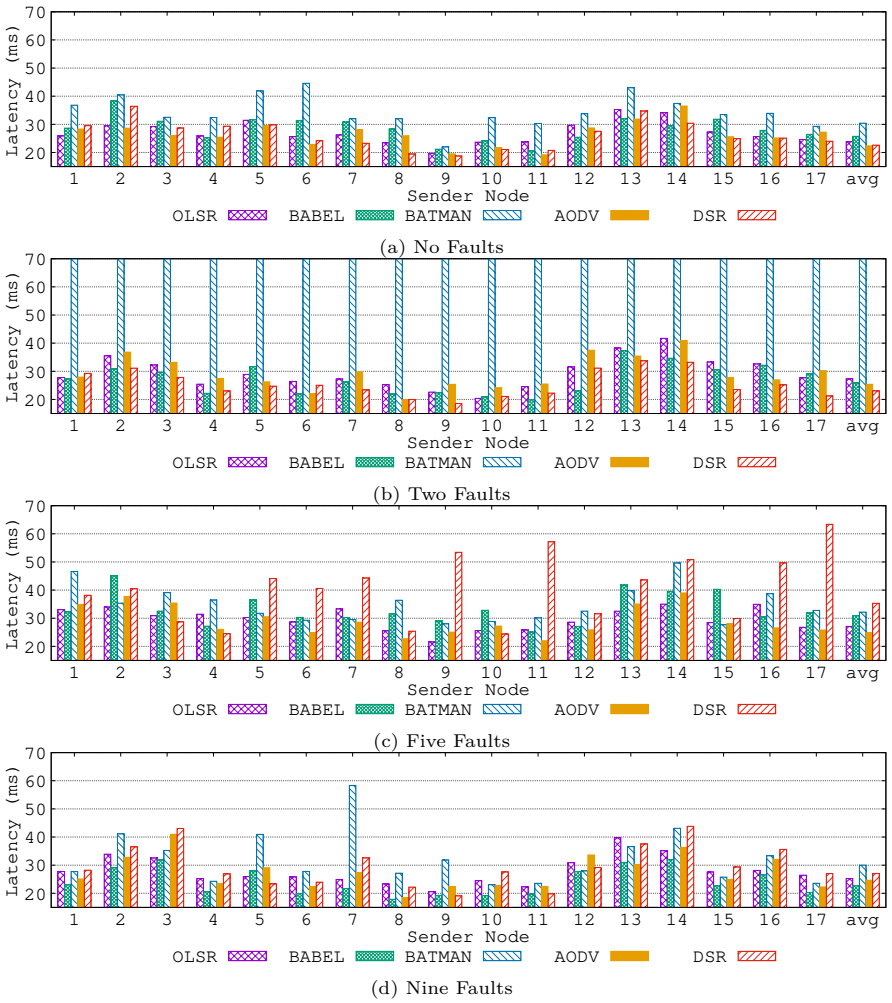


Figure 4: Average Latency of Routing Protocols.

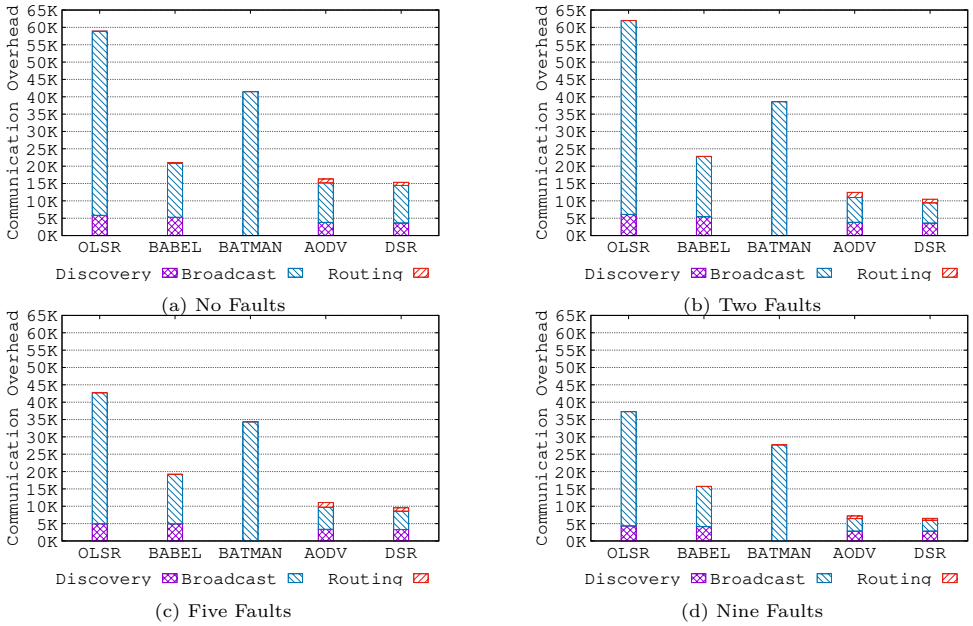
tocols while allowing to dynamically adapt the amount of proactive and reactive behavior. However, although IZR allows combining practically any proactive and reactive solutions, it considers them as "black boxes" and does not attempt to decompose them into their fundamental constituents to properly analyze each routing solution, as we do in this paper.

The Relay Node Set (RNS) [38] in contrast, is an analytical framework for comparing the communication overhead of routing protocols. RNS views each protocol as a handler of sets of nodes that retransmit control messages, being that each protocol may manage more than one of these sets at a time. In this sense, this framework dissects each protocol from an evaluation standpoint and not according to their internal operation, as our framework does.

Finally, the Multi-Mode Routing Protocol (MMRP) [36] framework independently selects the most suitable protocol for a given region of the network according to its local characteristics, allowing the coexistence of multiple protocols within the same network (called multi-mode routing). However, MMRP is not flexible enough to specify the majority of the existing protocols since it heavily relies on a single and specific architectural pattern, only suitable for a restricted set of solutions, unlike our framework which is much more generic.

6 Final Remarks

In this paper, we presented a conceptual framework to specify routing protocols for wireless *ad hoc* networks that abstracts the protocols' common aspects, a task that is not trivial due to their nature, and exposes parameters



(a) No Faults (b) Two Faults

(c) Five Faults (d) Nine Faults

Figure 5: Total Communication Overhead per Routing Protocol.

that capture the behavior of particular solutions. Leveraging a prototype of our framework, we implemented a representative set of existing routing protocols and evaluted their performance in a real wireless *ad hoc* network formed by commodity devices. The results showed interesting observations that have not been explored and discussed before in the context of routing protocols in wireless *ad hoc* networks. BATMAN, which is considered in the literature as one of the best protocols, was not only never the best regarding the reliability in any scenario but also was the worst regarding the latency in all scenarios. Furthermore, reactive protocols presented similar reliability to proactive ones despite the unstable neighborhoods and node faults, due to the usage of route caching, while having much lower overhead. However, these results cannot be extrapolated to other topologies, and more exhaustive evaluations should be carried out.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 17(4):2347–2376, 2015.

[2] T. R. Andel and A. Yasinsac. On the credibility of manet simulations. *Computer*, 39(7):48–54, 2006.

[3] S. Baraković and J. Baraković. Comparative performance evaluation of mobile ad hoc routing protocols. In *The 33rd International Convention MIPRO*, pages 518–523, 2010.

[4] A. Boukerche and A. Darehshoorzadeh. Opportunistic routing in wireless networks: Models, algorithms, and classifications. *ACM Comput. Surv.*, 47(2), Nov. 2014.

[5] D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of manet simulators. In *Proceedings of the Second ACM International Workshop on Principles of Mobile Computing*, POMC '02, pages 38–43. Association for Computing Machinery, 2002.

[6] J. Chroboczek and D. Schinazi. The Babel Routing Protocol. Technical report, Jan. 2021.

[7] T. H. Clausen, C. Dearlove, P. Jacquet, and U. Herberg. The Optimized Link State Routing Protocol Version 2. Technical report, Apr. 2014.

[8] P. A. Costa, A. Rosa, and J. a. Leitão. Enabling wireless ad hoc edge systems with yggdrasil. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, pages 2129–2136, New York, NY, USA, 2020. Association for Computing Machinery.

[9] S. R. Das, R. Castaneda, Jiangtao Yan, and R. Sengupta. Comparative performance evaluation of routing protocols for mobile, ad hoc networks. In *Proceedings 7th International Conference on Computer Communications and Networks (Cat. No.98EX226)*, pages 153–161, 1998.

[10] S. R. Das, R. Castañeda, and J. Yan. Simulation-based performance evaluation of routing protocols for mobile ad hoc networks. *Mobile Networks and Applications*, 5(3):179–189, Sept. 2000.

[11] T. Dillon, C. Wu, and E. Chang. Cloud computing: Issues and challenges. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 27–33, 2010.

[12] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, MobiCom '04, pages 114–128. Association for Computing Machinery, 2004.

[13] R. Dube, C. D. Rais, Kuang-Yeh Wang, and S. K. Tripathi. Signal stability-based adaptive routing (ssa) for ad hoc mobile networks. *IEEE Personal Communications*, 4(1):36–45, 1997.

[14] E. Gafni and D. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, 29(1):11–18, 1981.

[15] M. Gerla. Fisheye State Routing Protocol (FSR) for Ad Hoc Networks. Internet-Draft draft-ietf-manet-fsr-03, Internet Engineering Task Force, June 2002.

[16] Z. J. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of ICUPC 97 - 6th International Conference on Universal Personal Communications*, volume 2, pages 562–566 vol.2, Oct. 1997.

[17] G. He. Destination-sequenced distance vector (dsdv) protocol. *Networking Laboratory, Helsinki University of Technology*, pages 1–9, 2002.

[18] Y.-C. Hu, D. A. Maltz, and D. B. Johnson. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. Technical Report 4728, Feb. 2007.

[19] N. Javaid, A. Javaid, I. A. Khan, and K. Djouani. Performance study of etx based wireless routing metrics. In *2009 2nd International Conference on Computer, Control and Communication*, pages 1–7, 2009.

[20] D. Johnson and G. Hancke. Comparison of two routing metrics in olsr on a grid based mesh network. *Ad Hoc Networks*, 7(2):374–387, 2009.

[21] B. Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pages 243–254. Association for Computing Machinery, 2000.

[22] W. Kiess and M. Mauve. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Networks*, 5(3):324–339, 2007.

[23] K.-H. Kim and K. G. Shin. On accurate measurement of link quality in multi-hop wireless mesh networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, MobiCom '06, pages 38–49. Association for Computing Machinery, 2006.

[24] S. . Lee and M. Gerla. Dynamic load-aware routing in ad hoc networks. In *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, volume 10, pages 3206–3210 vol.10, 2001.

[25] J. Leitão, P. Á. Costa, M. C. Gomes, and N. M. Preguiça. Towards enabling novel edge-enabled applications. Technical report, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2018.

[26] S. Mueller, R. P. Tsang, and D. Ghosal. Multipath routing in mobile ad hoc networks: Issues and challenges. In M. C. Calzarossa and E. Gelenbe, editors, *Performance Tools and Applications to Networked Systems*, pages 209–234, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[27] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich. Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.). Internet-Draft draft-wunderlich-openmesh-manet-routing-00, Internet Engineering Task Force, Apr. 2008.

[28] V. D. Park and D. S. M. Corson. Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification. Internet-Draft draft-ietf-manet-tora-spec-04, Internet Engineering Task Force, July 2001.

[29] C. E. Perkins, S. Ratliff, J. Dowdell, L. Steenbrink, and V. Pritchard. Ad Hoc On-demand Distance Vector Version 2 (AODVv2) Routing. Internet-Draft draft-perkins-manet-aodvv2-03, Internet Engineering Task Force, Feb. 2019.

[30] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer. Sharp: A hybrid adaptive routing protocol for mobile ad hoc networks. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, MobiHoc '03, pages 303–314. Association for Computing Machinery, 2003.

[31] D. G. Reina, S. L. Toral, F. Barrero, N. Bessis, and E. Asimakopoulou. *The Role of Ad Hoc Networks in the Internet of Things: A Case Scenario for Smart Environments*, pages 89–113. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[32] S. Ruehrup. Theory and practice of geographic routing. In X. C. Hai Liu, Yiu-Wing Leung, editor, *Ad hoc and sensor wireless networks: architectures, algorithms and protocols*, volume 69, chapter 5, pages 69–88. Bentham Science, 2009.

[33] P. Samar and Z. Haas. Strategies for broadcasting updates by proactive routing protocols in mobile ad hoc networks. In *MILCOM 2002. Proceedings*, volume 2, pages 873–878 vol.2, 2002.

[34] P. Samar, M. R. Pearlman, and Z. J. Haas. Independent zone routing: an adaptive hybrid routing framework for ad hoc wireless networks. *IEEE/ACM Transactions on Networking*, 12(4):595–608, 2004.

[35] R. Sanchez-Iborra and M. Cano. Joker: A novel opportunistic routing protocol. *IEEE Journal on Selected Areas in Communications*, 34(5):1690–1703, May 2016.

[36] C. A. Santivanez and I. Stavrakakis. Towards adaptable ad hoc networks: The routing experience. In M. Smirnov, editor, *Autonomic Communication*, pages 229–244, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[37] R. C. Shah and J. M. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *2002 IEEE Wireless Communications and Networking Conference Record. WCNC 2002 (Cat. No.02TH8609)*, volume 1, pages 350–355 vol.1, 2002.

[38] Tao Lin, S. F. Midkiff, and J. S. Park. A framework for wireless ad hoc routing protocols. In *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, volume 2, pages 1162–1167 vol.2, 2003.

[39] F. L. Templin, R. Ogier, and M. S. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). Technical report, Feb. 2004.

[40] C.-K. Toh. Long-lived Ad Hoc Routing based on the Concept of Associativity. Internet-Draft draft-ietf-manet-longlived-adhoc-routing-00, Internet Engineering Task Force, Mar. 1999.

[41] L. D. Xu, W. He, and S. Li. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, 2014.