A Routing Protocol Suitable for Backhaul Access in Wireless Mesh Networks

Miguel Elias M. Campista[*], Luís Henrique M. K. Costa, Otto Carlos M. B. Duarte

Grupo de Teleinformática e Automação
PEE/COPPE - DEL/POLI
Universidade Federal do Rio de Janeiro
P.O. Box 68504 - 21945-970, Rio de Janeiro, RJ, Brazil
Tel.: +55 21 2562-8635 Fax: +55 21 2562-8628

Abstract

This work proposes the Wireless-mesh-network Proactive Routing (WPR) protocol for wireless mesh networks, which are typically employed to provide backhaul access. WPR computes routes based on link states and, unlike current routing protocols, it uses two algorithms to improve communications in wireless mesh networks taking advantage of traffic concentration on links close to the network gateways. WPR introduces a controlled-flooding algorithm to reduce routing control overhead by considering the network topology similar to a tree. The main goal is to improve overall efficiency by saving network resources and avoiding network bottlenecks. In addition, WPR avoids redundant messages by selecting a subset of one-hop neighbors, the AMPR

[*]Corresponding author.
Email addresses: `miguel@gta.ufrj.br` (Miguel Elias M. Campista), `luish@gta.ufrj.br` (Luís Henrique M. K. Costa), `otto@gta.ufrj.br` (Otto Carlos M. B. Duarte)

(Adapted MultiPoint Relay), needed to reach all two-hop ones. We first analyze the proposed algorithms compared with the algorithms used by OLSR for the same tasks in terms of running time, optimality, and number of routing messages. Results show that the algorithms proposed by WPR are more efficient than the algorithms used by OLSR in running time and number of routing messages. In addition, we also perform simulations to evaluate the performance of WPR. Results reveal that the aggregated throughput of WPR outperforms OLSR by up to 27% using a combination of web and backbone internal traffic despite our design assumption of traffic convergence toward gateways.

Keywords:
Wireless mesh networks, routing protocol, routing algorithm.

1. Introduction

Wireless mesh networks (WMNs) are a cost-effective solution for Internet access given their low installation cost, ease of deployment, and mobility support [1]. The main goal of these networks is to improve connectivity and to provide backhaul access, to nodes not within range of gateways, to the wired infrastructure [2, 3]. Wireless mesh networks implement a backbone of stationary wireless routers, which communicate via multiple hops. As a consequence, routing plays a major role in these networks.

Most wireless networking advantages are related to the broadcast nature of radiofrequency transmissions. Nevertheless, communication protocols must cope with challenges of radiofrequency transmissions as fast link-quality changes, high attenuation, and interference [4, 5]. In addition, bandwidth constraints of wireless channels and shared medium impose limits on the maximum data rate achievable. Protocols, as a consequence, introduce control packets to improve communication reliability and also to provide current network conditions to the routing layer. This additional overhead, however, can interfere in data communications and consume a significant amount of network bandwidth. This side effect is considered by other protocols of wireless networks. For instance, in ad hoc networks, routing protocols generally adopt one of the three strategies for route construction: proactive, on-demand, or hybrid. Proactive protocols require nodes to periodically flood the network with control messages to maintain a valid route to every other node in the network [6, 7]. In the reactive strategy, nodes broadcast a route request to a given destination only when they have data to send [8, 9] to avoid periodic control overhead. This strategy, however, incurs in route discovery latency and cannot guarantee that overhead is indeed lower than in the proactive case. If route failures often happen, then overhead becomes high because route discovery procedures are repeatedly triggered [8, 9, 10, 11].

Routing protocols for wireless mesh networks are also designed to avoid overhead [12]. In these networks, the impact of control messages is more severe than in the ad hoc case because wireless mesh networks are usually employed to provide backhaul access [1, 2, 3]. Hence, more bandwidth available is required to forward data traffic to and from users. Despite the stationary backbone, the wireless medium is unstable and the data traffic typically converges toward a gateway to the wired network. This results in network bottlenecks, which increases the importance of saving network resources. Protocols such as Link Quality Source Routing (LQSR) [13] and

Srcr [14] use a hybrid strategy to avoid overhead. Although they maintain information about link state, they use route discovery procedures to update the metrics of the most frequently used links. These protocols assume that periodic flooding is inefficient because most communications include a wired-network gateway. Link-state proactive protocols, on the other hand, use controlled-flooding techniques to reduce overhead. The Optimized Link-State Routing (OLSR) protocol is a well-known example. OLSR defines a subset of neighbors in charge of forwarding routing messages. This subset, called MPR (MultiPoint Relay), is composed of one-hop neighbors which are enough to reach all two-hop ones. By using the MPR set, OLSR avoids redundant control messages because it minimizes the probability of two-hop neighbors receive the same message more than once. The Fisheye State Routing (FSR) [15] adjusts the time-to-live (TTL) field of routing messages to concentrate route updates in the vicinity. The authors argue that it is more important to maintain information of nearby links than of distant ones. The claim is that a packet finds increasingly accurate routing information as it approaches its destination. The Optimized Fisheye Link-State Routing (OFLSR) [16] protocol combines OLSR and FSR controlled-flooding techniques to further reduce control overhead.

In this work, we propose a proactive link-state routing protocol called Wireless-mesh-network Proactive Routing (WPR) protocol[1]. Using a proactive strategy, we maintain a route to every other node in the network and we avoid initial latency during route discovery procedures. WPR also introduces

[1]A preliminary version of this paper appeared in [17].

a controlled-flooding algorithm tailored to wireless access networks. Unlike OLSR and FSR, which reduce routing overhead considering that communications between any pair of nodes have the same probability, WPR updates more frequently the route metrics of the most-used links. These are the links within the shortest paths connecting backbone nodes to the wired-network gateway and vice-versa [18]. WPR, therefore, considers the network topology a tree, in which the wired-network gateway is the root. Additionally, WPR introduces the AMPR (Adapted MultiPoint Relay) set, which aims at reducing redundant control messages received by nodes.The proposed AMPR set algorithm computes the minimum possible subset of one-hop neighbors needed to reach all two-hop ones taking as input the resultant tree topology used by the controlled-flooding algorithm. We analyze the proposed algorithms by comparing them with their OLSR counterparts in terms of running time, optimality, and number of control messages. In addition, we analyze the performance of WPR also compared with OLSR via simulations. Results show that WPR throughput is up to 80% better than the throughput achieved by OLSR when all nodes generate traffic only towards the wired-network gateway. In addition, simulations using traffic between internal nodes show that the aggregated throughput achieved by WPR is up to 27% better than OLSR even though WPR algorithms take into account that most communications involve the wired-network gateway.

This work is organized as follows. Section 2 describes definitions and notations used throughout this work. Section 3 presents the proposed Wireless-mesh-network routing (WPR) protocol. The algorithms used by WPR are analyzed in Section 4. Section 5 describes the simulation setup. Simulation results are presented in Section 6. Finally, Section 7 concludes this work and identifies future directions.

2. Definitions and Notations

In this work, we assume that applications usually require Internet access [13, 19]. Therefore, wireless mesh networks provides backhaul access to users connected to the wireless backbone. Nevertheless, applications in which the source-destination pair is inside the wireless mesh are also possible (e.g. peer-to-peer applications where some peers are inside the mesh). We also assume that the network topology known by different backbone nodes is similar. This characteristic is required by routing protocols to avoid errors such as routing loops.

2.1. Network Model

We model a wireless mesh network as a weighted connected graph $G = (V, E)$ where V is the vertex set and E is the edge set. Vertices represent network nodes and edges represent links connecting pairs of nodes. Considering v_i and v_j in V, if there is a link from v_i to v_j, then $(v_i, v_j) \in E$. In our model, there is a weight $w(v_i, v_j) \in \Re_+$ associated with each link in the network, which represents the routing metric of the link in routing context. The graph is also directed because routing metrics may be different from v_i to v_j and vice-versa.

Let $\mathcal{N}_1(v_i)$ denote the set of one-hop neighbors of node v_i and $\mathcal{N}_2(v_i)$ the set of two-hop neighbors of v_i, i.e. node v_i directly communicates to nodes in $\mathcal{N}_1(v_i)$, but needs a relay node in $\mathcal{N}_1(v_i)$ to communicate to a node in $\mathcal{N}_2(v_i)$. Thus,

$$\mathcal{N}_2(v_i) = \bigcup_{v_j \in \mathcal{N}_1(v_i)} (\mathcal{N}_1(v_j) - \mathcal{N}_1(v_i)). \tag{1}$$

In addition, we denote the MPR set of node v_i by $\mathcal{M}(v_i)$, where $\mathcal{M}(v_i) \subseteq \mathcal{N}_1(v_i)$. The MPR set is a subset of $\mathcal{N}_1(v_i)$ such that all nodes in $\mathcal{N}_2(v_i)$ are reachable through a node in $\mathcal{M}(v_i)$. The problem of finding an MPR set with the minimum number of nodes is modeled such as the set-covering problem. Because the optimal solution to this problem is NP-complete [20], the OLSR protocol uses a greedy algorithm (Appendix A) to compute MPR sets in polynomial time. Nevertheless, the resultant sets may not represent the optimal solution.

A path $p(.)$ from the source node v_1 to the destination node v_l is a sequence of distinct nodes in which any consecutive pair is connected by a link. Therefore, $p(v_1, v_l) = \langle v_1, v_2, ..., v_{l-1}, v_l \rangle$, where $\{v_1, v_2, ..., v_{l-1}, v_l\} \subseteq V$, $\{(v_1, v_2), ..., (v_{l-1}, v_l)\} \subseteq E$ and l is the length of the path. The cost of path $p(v_1, v_l)$ is defined as a composition function $f\colon w(v_1, v_2), ..., w(v_{l-1}, v_l) \rightarrow \Re_+$ that maps the weights of the links in the path to a positive real number. In addition, the shortest path from v_1 to v_l, denote by $p^*(v_1, v_l)$, is the path with the minimum possible cost ($\delta(v_1, v_l)$). We consider the shortest path $p^*(v_1, v_l)$ a subgraph of G. Hence, $p^*(v_1, v_l) = (V^*, E^*)$, where $V^* \subseteq V$ and $E^* \subseteq E$.

Hop-by-hop routing protocols, such as WPR and OLSR, build tables with the next hop to each reachable destination. The next hop to v_l, for instance, is the neighbor of v_1 within the shortest path $p^*(v_1, v_l)$. It is worth mentioning that the shortest path from a source to any destination is prone to changes according to medium conditions. Therefore, v_i always computes its shortest path to each other node upon receiving routing message updates from other network nodes. We denote these control messages by m_c. As WPR is based on link state, m_c is a link-state update. Similarly to OLSR, WPR runs the Dijkstra algorithm to find shortest paths using additive metrics. Alternative approaches can be found in Passos et al. [21]. In this work, we use the term control messages and routing messages interchangeably.

2.2. Ascendent, Descendent, and AMPR Sets

We define the *ascendent set* of node v_i, $\mathcal{A}(v_i)$, to be the set of nodes in the shortest path from v_i to the wired-network gateway (g), except v_i. Therefore, considering that $p^*(v_{i+1}, g)$ is a subgraph of G and V^* is the vertex set of $p^*(v_{i+1}, g)$, we have that:

$$\mathcal{A}(v_i) = \{v_j \in V^* \ | p^*(v_{i+1}, g) \subseteq G,\ p^*(v_{i+1}, g) = (V^*, E^*)\}. \tag{2}$$

We also define the *descendent set* of node v_i, $\mathcal{D}(v_i)$, to be the set of nodes which have v_i in their best path to the gateway. Hence,

$$\mathcal{D}(v_i) = \{v_j \in V \mid v_i \in \mathcal{A}(v_j)\}. \tag{3}$$

Figure 1 illustrates the ascendent and descendent sets of v_i, where $\mathcal{A}(v_i) = \{a_1, a_2, ..., g\}$ and $\mathcal{D}(v_i) = \{d_1, d_2, ...\}$, respectively. Set $\mathcal{T}(v_i)$ represents the set of all nodes in $\mathcal{A}(v_i)$, $\mathcal{D}(v_i)$, and $\{v_i\}$. Hence, $\mathcal{T}(v_i) = \mathcal{A}(v_i) \cup \mathcal{D}(v_i) \cup \{v_i\}$. The resultant topology considering the shortest paths from all nodes in $\mathcal{T}(v_i)$ to the gateway is a tree, depicted in Figure 1.

The *Adapted MultiPoint Relay* (AMPR) set of v_i, denoted by $\mathcal{R}(v_i)$, is the union of two subsets. The subset of one-hop neighbors of v_i in $\mathcal{T}(v_i)$ able

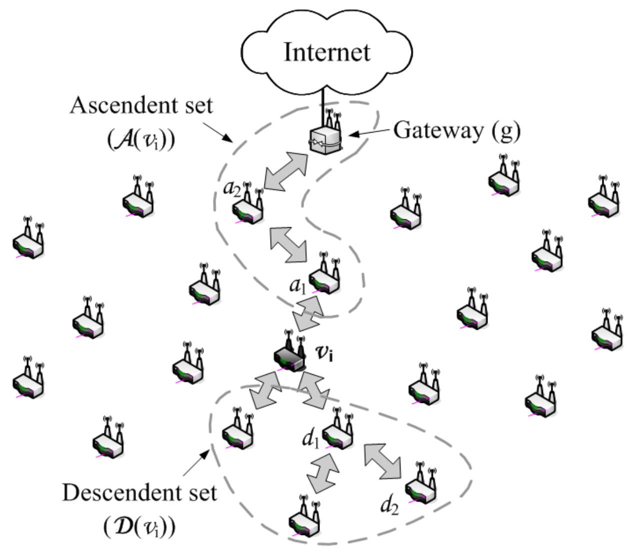


Figure 1: Ascendent and descendent sets of node v_i.

to reach all two-hop neighbors also of v_i in $\mathcal{T}(v_i)$ and the subset of one-hop neighbors of v_i in $\overline{\mathcal{T}(v_i)}$ able to reach all two-hop neighbors also of v_i in $\overline{\mathcal{T}(v_i)}$, where $\overline{\mathcal{T}(v_i)} = V - \mathcal{T}(v_i)$. Therefore, $\mathcal{R}(v_i) \subseteq \mathcal{N}_1(v_i)$. We can rewrite $\mathcal{R}(v_i)$ as following:

$$\mathcal{R}(v_i) = \mathcal{R}(v_i)' \cup \mathcal{R}(v_i)'', \tag{4}$$

where $\mathcal{R}(v_i)' \subseteq \{\mathcal{N}_1(v_i) \cap \mathcal{T}(v_i)\}$ and $\mathcal{R}(v_i)'' \subseteq \{\mathcal{N}_1(v_i) \cap \overline{\mathcal{T}(v_i)}\}$.

The next section explains how ascendent, descendent, and AMPR sets are used in WPR to reduce routing overhead.

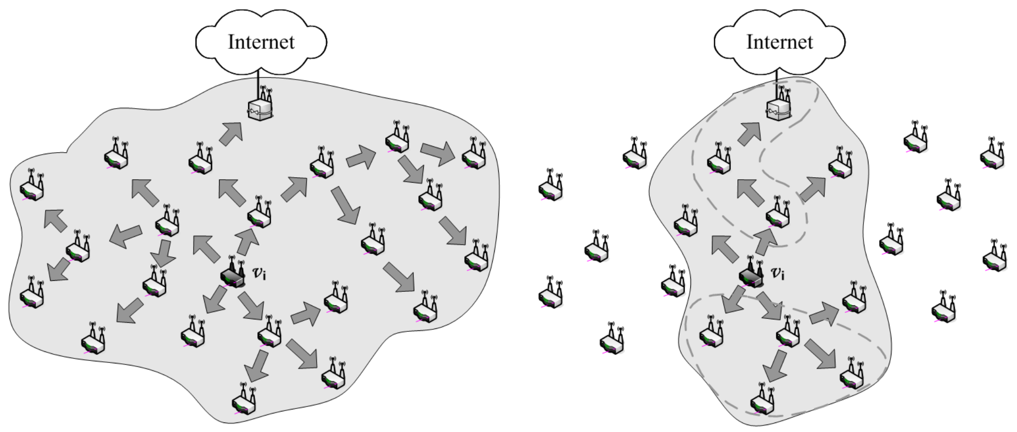
3. WPR Protocol

We propose a proactive link-state-based protocol called WPR (Wireless-mesh-network Proactive Routing) that uses two main algorithms to reduce routing overhead in wireless mesh networks: the controlled-flooding algorithm and the AMPR set computation algorithm. The key idea of the controlled-flooding algorithm is to reduce the overhead in wireless mesh networks by taking into account the traffic pattern of backhaul access networks. These networks typically concentrate traffic on links connecting the backbone nodes to the wired-network gateway. In addition, WPR avoids redundant routing messages using the AMPR set. The function of the AMPR set is similar to the MPR set of OLSR. The AMPR set, however, also considers the tree topology of WPR controlled-flooding algorithm.

3.1. The Controlled-flooding Algorithm of WPR

Controlled-flooding algorithms aim at reducing the transmission frequency of link-state updates from seldom used or redundant links [12]. Our controlled-flooding algorithm assumes that communications converge toward gateways. Hence, the topology can be modeled as a tree, where the gateway is the root. Based on the tree topology, the controlled-flooding algorithm of WPR considers as the most-used links by a node v_i, the nodes belonging to two different sets: the ascendent set ($\mathcal{A}(v_i)$) and the descendent set ($\mathcal{D}(v_i)$), as defined in Section 2.2.

WPR reduces routing overhead by restricting the number of nodes that forward a routing message originated at node v_i. Messages originated at v_i are only forwarded by nodes in $\mathcal{A}(v_i)$ and $\mathcal{D}(v_i)$. In the first case, messages are forwarded because v_i is a descendent whereas in the second case messages are forwarded because v_i is an ascendent. Forward routing messages from ascendent nodes allows descendents to compute their path to the gateway, whereas forward routing messages from descendent nodes allows as-



(a) Flooding range of the OLSR protocol. (b) Flooding range of the WPR protocol.

Figure 2: OLSR and WPR controlled-flooding range.

cendents to compute their path in the reverse direction. Figure 2 illustrates the controlled-flooding algorithms of OLSR and of WPR. Figure 2(a) shows the maximum range of an OLSR flooding. In OLSR, the MPR set eliminates redundant messages, but all nodes still receive a routing message from v_i. Figure 2(b) shows the range of a controlled-flooding message sent by node v_i using WPR. Node v_i has its routing messages forwarded only by nodes in $\mathcal{T}(v_i)$ set.

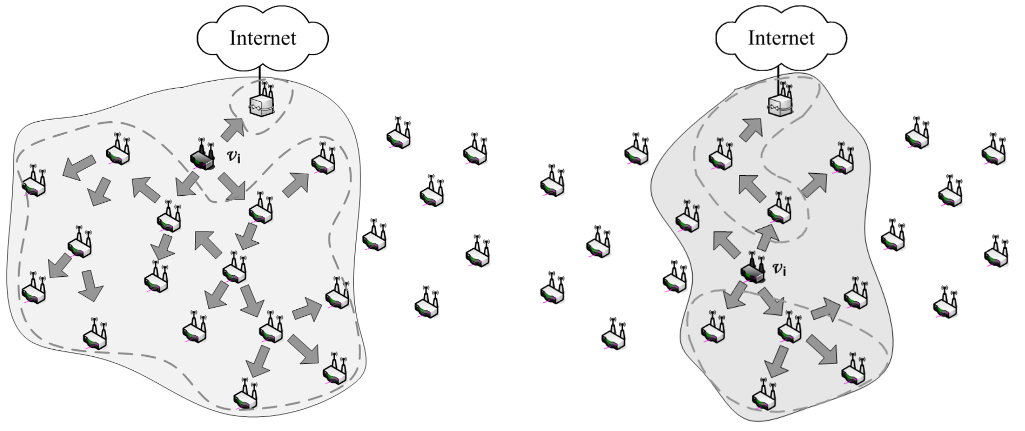
The $\mathcal{A}(v_i)$ set is known as soon as node v_i computes its path to the gateway (g). Therefore, a slight modification to typical routing protocols is needed to maintain complete paths instead of only the tuple composed of relay and destination nodes. This change requires more information per destination, but it does not add additional processing to compute routes. The $\mathcal{D}(v_i)$ set is known due to `HELLO` messages. In these messages, every node lists its neighbors and indicates each one-hop ascendant chosen. Thus, node v_i knows its one-hop descendents and can forward their messages. To forward routing messages originated at two- or h-hop nodes, for $h > 2$ in $\mathcal{D}(v_i)$, a recursive property holds. Let a descendent of v_i be h hops distant. Thus, $(h-1)$-hop nodes certainly forward routing messages originated at h-hop-node descendents because they receive their `HELLO` messages. Likewise, $(h-2)$-hop nodes forward routing messages from $(h-1)$-hop descendents. These $(h-2)$-hop nodes, however, also forward messages from h-hop nodes that were forwarded at last by their $(h-1)$-hop descendents. Considering that an $(h-1)$-hop ascendent is also in the ascendent set of $(h-1)$-hop descendents, $(h-2)$-hop nodes also forward messages from h-hop nodes. Therefore, according to our definition, these h-hop nodes are also in the descendent set of the $(h-2)$-hop nodes. Generalizing the example, we show that if a routing message was at last forwarded by an one-hop descendent of node v_i, e.g. d_1 in Figure 1, it is assumed that one of the d_1's descendents has originated the message. As d_1's descendents are also in $\mathcal{D}(v_i)$, the recursive property holds.

WPR uses topology control messages to disseminate link states. Nevertheless, a flag is needed on the message header to indicate the dissemination type. In WPR, in addition to controlled-flooding messages, the network is periodically flooded. This guarantees that all nodes are aware of the complete network topology. It is needed to flood the entire network because communications not including gateways are also possible and therefore the entire topology must be known. WPR uses a period T to send control messages and a proportional relation between the number of controlled-flooding messages and flooding messages. This is similar to the Fisheye State Routing (FSR) protocol [15], where the TTL used is periodically set to 255 in order to reach all nodes in the network. Both WPR controlled-flooding algorithm and FSR perform spatial flooding concentrating routing updates on a certain area [12]. WPR concentrates routing messages on the tree topology whereas FSR concentrated on nearby nodes in an expanding-ring basis.

Algorithm 1 Controlled-flooding algorithm of WPR.

```
Require: $m_c$, $\mathcal{T}(v_i)$, and g.
Ensure: Action taken on $m_c$.
 1: if $v_i \in \mathcal{R}(v_{i-1})$ then
 2:   if $m_c$ is a network-wide flooding message then
 3:     forward($m_c$);
 4:   else if $m_c$ is a controlled-flooding message then
 5:     if is_ascendent($v_1, v_i, g$) or is_descendent($v_{i-1}, v_i, g$) then
 6:       forward($m_c$);
 7:     else
 8:       discard($m_c$);
 9:     end if
10:   end if
11: end if
```

Algorithm 1 shows the operation of the controlled-flooding algorithm of WPR, considering the action taken by node v_i upon receiving a routing message m_c. We denote v_1 the source node of the control message m_c, v_{i-1} the neighbor of v_i which relayed the message m_c received by v_i, g the external gateway, and $\mathcal{R}(v_{i-1})$ the set of AMPR nodes of v_{i-1}. Considering that v_i knows whether it is in $\mathcal{R}(v_{i-1})$, it only forwards a routing message from node v_{i-1} if $v_i \in \mathcal{R}(v_{i-1})$ set. Thus, node v_i examines the type of message received. If m_c is a network-wide flooding message, node v_i simply forwards it. Otherwise, if m_c is a controlled-flooding message, node v_i uses



(a) Node v_i closer to the gateway. (b) Node v_i farther from the gateway.

Figure 3: Distance to the gateway effect on the controlled-flooding range of WPR nodes.

two functions to verify whether node v_{i-1} is in its $\mathcal{T}(v_i)$ set (Eq. 4). Function `is_ascendent`(v_1, v_i, g) verifies if the source node v_1 is in $\mathcal{A}(v_i)$ set, considering gateway g. On the other hand, function `is_descendent`(v_{i-1}, v_i, g) verifies if node v_{i-1} is an one-hop descendent of node v_i. Node v_i does not forward m_c if neither v_1 is in $\mathcal{A}(v_i)$ nor v_{i-1} is in $\mathcal{D}(v_i)$. Hence, $v_1 \notin \{\mathcal{A}(v_i) \cup \mathcal{D}(v_i)\}$.

The proportional relation between the transmission of controlled-flooding and flooding messages is adjusted according to the distance of each node to the gateway, given in number of hops. The closer the node to the gateway, the larger the number of descendents. Nodes closer to g are in a larger number of $\mathcal{T}_i$ sets, and therefore there is no need for them to flood the network as frequently as farther nodes. Note, however, that the frequency of topology control messages does not change. Only the number of controlled-flooding messages sent per network-wide flooding message changes. Figure 3 illustrates the key idea. Observe in Figure 3(a) that node v_i is closer to the gateway as compared with its position in Figure 3(b). As a consequence, the number of descendents also increases and a controlled-flooding message has almost the same effect as a network-wide flooding message. Based on this rationale, each node adjusts the relation between controlled-flooding messages and network-wide flooding messages according to the distance to the gateway, using the expression $r(h) = c(n) - h$, where $r(h)$ is the number of controlled-flooding messages per flooding message, h is the number of hops to reach the gateway, and $c(n)$ is the maximum number of controlled-flooding messages per network-wide flooding message. Function $c(n)$ is computed according to the number of nodes n in the network. The value of n is easily obtained with the knowledge of the network topology. Function $c(n)$ is equal to $R_{min} + \sqrt{n}$, where R_{min} is the minimum number of controlled-flooding messages per flooding message. As the node becomes farther from the gateway, the proportional relation $r(h)$ decreases. Hence, the number of controlled-flooding messages per flooding message also decreases. In this work, R_{min} is equal to 13, which is the same relation used by the Fisheye extension of OLSR [22] to flood the entire network. Using $r(h) > R_{min}$ for nodes closer to the gateway, the amount of control traffic injected in the network is lower because network-wide flooding messages are sent less often. It is worth mentioning that each router computes its own $r(h)$ and sends a sequence of controlled-flooding messages before each network-wide flooding message. Each forwarding router differentiates a controlled-flooding message from a network-wide flooding message by checking a flag in the packet header set at the originating node.

3.2. The AMPR Set Computation Algorithm

Algorithm 2 AMPR SET COMPUTATION ALGORITHM.

```
Require: $\mathcal{T}(v_i)$, $\mathcal{N}_1(v_i)$, $\mathcal{N}_2(v_i)$.
Ensure: $\mathcal{R}(v_i)$.
 1: $\mathcal{R}(v_i) \leftarrow \emptyset$
 2: if $\mathcal{N}_2(v_i) \neq \emptyset$ then
 3:   if $\mathcal{T}(v_i) \neq \emptyset$ then
 4:     /* First step: Compute the AMPR set for nodes in $\mathcal{T}(v_i)$ */
 5:     if $\mathcal{A}(v_i) \neq \emptyset$ then
 6:       $\mathcal{R}(v_i) \leftarrow \mathcal{A}(v_i) \cap \mathcal{N}_1(v_i)$
 7:     end if
 8:     if $\mathcal{D}(v_i) \neq \emptyset$ then
 9:       $\mathcal{N} \leftarrow \mathcal{D}(v_i) \cap \mathcal{N}_1(v_i)$
10:       $\mathcal{N}\mathcal{N} \leftarrow \mathcal{D}(v_i) \cap \mathcal{N}_2(v_i)$
11:       $\mathcal{R}(v_i) \leftarrow \mathcal{R}(v_i) \cup$ compute_mpr($\mathcal{N}$, $\mathcal{N}\mathcal{N}$)
12:     end if
13:     /* Second step: Compute the AMPR set for nodes in $\overline{\mathcal{T}(v_i)}$ */
14:     if $\overline{\mathcal{T}(v_i)} \neq \emptyset$ then
15:       $\mathcal{N} \leftarrow (\overline{\mathcal{T}(v_i)} \cap \mathcal{N}_1(v_i))$
16:       $\mathcal{N}\mathcal{N} \leftarrow (\overline{\mathcal{T}(v_i)} \cap \mathcal{N}_2(v_i))$
17:       $\mathcal{N}\mathcal{N} \leftarrow \mathcal{N}\mathcal{N} - \bigcup_{v_j \in \mathcal{R}(v_i)} \mathcal{N}_1(v_j)$
18:       $\mathcal{R}(v_i) \leftarrow \mathcal{R}(v_i) \cup$ compute_mpr($\mathcal{N}$, $\mathcal{N}\mathcal{N}$)
19:     end if
20:   else
21:     $\mathcal{R}(v_i) \leftarrow$ compute_mpr($\mathcal{N}_1(v_i)$, $\mathcal{N}_2(v_i)$)
22:   end if
23: end if
```

The algorithm to compute the AMPR set first finds the subset of one-hop neighbors in the tree topology to reach all two-hop neighbors also in the tree. Afterward, the AMPR set computation algorithms repeats a similar procedure to find all other one-hop neighbors needed to reach all two-hop ones not in the tree. In our case, however, the algorithm to compute MPR needs to be run twice, as seen in Algorithm 2.

At the beginning, Algorithm 2 resets the current AMPR set. Afterward, if there are two-hop neighbors, the algorithm checks if $\mathcal{T}(v_i)$ set is empty. If $\mathcal{T}(v_i)$ is not empty, the algorithm computes the subset of one-hop neighbors in $\mathcal{T}(v_i)$ able to reach all two-hop ones also in $\mathcal{T}(v_i)$. First, the algorithm inserts into $\mathcal{R}(v_i)$ the one-hop ascendent of v_i. Note that the one-hop ascendent of v_i must be in $\mathcal{R}(v_i)$. Otherwise, a routing message from node v_i will never be forwarded to all ascendent nodes, according to Algorithm 1. Then, the algorithm finds the AMPR set needed to reach all two-hop descendents. To accomplish this, the algorithm performs some initialization procedures. It uses two temporary sets, $\mathcal{N}$ and $\mathcal{NN}$, to represent $\mathcal{D}(v_i) \cap \mathcal{N}_1(v_i)$ and $\mathcal{D}(v_i) \cap \mathcal{N}_2(v_i)$, respectively. It is worth mentioning that because each node lists its neighbors in `HELLO` messages, the node can identify its $\mathcal{N}_2(v_i)$ set according to Eq. 1. The algorithm finds the subset of one-hop descendents in $\mathcal{N}$ needed to reach all two-hop descendents in $\mathcal{NN}$. If the $\mathcal{T}(v_i)$ is empty, the tree topology was not established, which makes the parameters passed to function `compute_MPR` to be the complete sets $\mathcal{N}_1(v_i)$ and $\mathcal{N}_2(v_i)$.

After the first step of Algorithm 2, all nodes in $\mathcal{T}(v_i)$ can be reached in a controlled flooding of WPR. Nevertheless, as mentioned earlier, we also perform network-wide flooding. Therefore, the algorithm has also to find the subset of one-hop neighbors not in $\mathcal{T}(v_i)$ (nodes not in $\mathcal{T}(v_i)$ are in $\overline{\mathcal{T}(v_i)}$) needed to reach all two-hop neighbors also in $\overline{\mathcal{T}(v_i)}$. In this second step, the algorithm resets the two temporary sets $\mathcal{N}$ and $\mathcal{NN}$ and initializes them to $\overline{\mathcal{T}(v_i)} \cap \mathcal{N}_1(v_i)$ and $\overline{\mathcal{T}(v_i)} \cap \mathcal{N}_2(v_i)$, respectively. After that, the algorithm excludes from $\mathcal{NN}$ possible two-hop neighbors already reached by some node in the AMPR set as computed in the first step. The algorithm then finally runs again function `compute_MPR` to complement $\mathcal{R}_i$ with the one-hop neighbors needed to reach all nodes in $\overline{\mathcal{T}(v_i)}$. The second time function `compute_MPR` is run guarantees that all two-hop nodes are reached during a network-wide flooding from v_i. The final AMPR set is, therefore, the union of the nodes found in the first and in the second time function `compute_MPR` is executed.

For comparison, we present the algorithm used by OLSR to compute the MPR set in Appendix A.

3.3. Discussion

WPR protocol requires the complete knowledge of the network topology map to identify nodes in the ascendent set. This requirement is achieved by using a proactive link-state-based protocol. Note that the main feature of the OLSR protocol is the MPR set, which also relies on the knowledge of one- and two-hop neighbors. This information is straightforward using link-state-based protocols. Distance-vector-based protocols do not meet this requirement and, therefore, need more investigation. Additionally, in WPR we do not consider user mobility [23]. Thus, if a user is willing to connect to the Internet, he must use a backbone node playing the role of access point. This is indeed the case in the upcoming standard for wireless mesh networks, IEEE 802.11s [24]. WPR neither considers network partitions [25, 26] nor unauthorized backbone nodes [27]. Because backbone nodes are static, we assume that network partitions do not occur very often. Security issues, on the other hand, are left aside since this is not the focus of WPR.

WPR nodes need to know the IP address of the wired-network gateway to compute the ascendent set. This is important for our controlled-flooding algorithm, but requires prior knowledge of the gateway IP address or route announcement. In WPR, the IP address of a gateway is obtained using the latter option. This feature is quite usual. Many protocols have messages to announce default routes to other networks. In OLSR, for instance, the HNA (Host and Network Association) messages are used with this goal [6].

Although in this paper we do not consider multiple gateways, it is possible to extend WPR to cope with this issue. For instance, if there were a set of gateways, denoted by $\mathcal{G}$, the ascendent set of node v_i would be the union of all ascendent sets to each gateway. Thus, $\mathcal{A}(v_i) = \bigcup_{g \in \mathcal{G}} \mathcal{A}(v_i, g)$, where $\mathcal{A}(v_i, g)$ is the path toward gateway g, as seen in Eq. 2. Likewise, the descendent set would be the union of all descendent sets with regard to each gateway. Hence, $\mathcal{D}(v_i) = \bigcup_{g \in \mathcal{G}} \mathcal{D}(v_i, g)$. This strategy incurs in higher overhead because more nodes would forward routing messages. In practice, however, even though there are multiple gateways, most routing protocols use only one gateway to communicate with the wired network. The simultaneous use of multiple gateways to send packets from the same data flow, i.e. for traffic balancing, is not usual because most routing protocols use single-path forwarding algorithms [28]. In this case, WPR algorithms also represent an efficient approach to deal with multiple gateways because each backbone router selects locally the best gateway according to the Dijkstra algorithm. If after another run of the Dijkstra algorithm a new gateway is selected, then a new tree is computed. Therefore, the use of the multiple trees would not be needed and WPR algorithms can be used without changes.

Another important issue in WPR is that there are no guarantees that a packet from a node will indeed be forwarded by only ascendent nodes, as discussed in Section 2.1. Nevertheless, the broadcast transmissions in the wireless environment allow nodes in the vicinity of the forwarding path to overhear routing messages. Therefore, considering that the differences between the path taken by the packet and the path computed by the originating node are not too large, routing messages can still be received by ascendent and descendent nodes.

4. Analysis of the Proposed Algorithms

In this section, we analyze the AMPR and MPR set computation algorithms in terms of running time and optimality. In addition, the controlled-flooding algorithms of WPR and OLSR are analyzed in terms of number of routing messages sent.

Lemma 1 *Let $|\mathcal{N}_1(v_i)|$ and $|\mathcal{N}_2(v_i)|$ denote respectively the number of one- and two-hop neighbors of node v_i, and $v = |\mathcal{N}_1(v_i)| = |\mathcal{N}_2(v_i)|$. In the least efficient case of MPR employment, i.e. $\mathcal{M}(v_i) = \mathcal{N}_1(v_i)$, the running time to an OLSR node to compute its MPR set is $O(v^2)$.*

Proof In OLSR, each node v_i needs to be aware of its one- and two-hop neighbors to compute the MPR set. Basically, adding a node to the MPR set requires node v_i to find its one-hop neighbor with the highest adjacency degree, considering only two-hop neighbors. The algorithm running on node v_i then counts the number of adjacencies each one-hop neighbor has. This operation is $O(v)$, where each step is an adjacency test. Computing the number of adjacencies of all v one-hop neighbors makes the complete operation takes $O(v^2)$ steps. In the least efficient case, all one-hop neighbors are in the MPR set of node v_i. Therefore, the algorithm does not enter into the while loop seen in Appendix A and terminates in $O(v^2)$ steps.

Lemma 2 *The running time of the MPR set computation algorithm is $O(v^3)$, if we do not consider the least efficient case, i.e. $\mathcal{M}(v_i) \subset \mathcal{N}_1(v_i)$.*

Proof If we do not consider the least efficient case, thus $\mathcal{M}(v_i) \subset \mathcal{N}_1(v_i)$, the MPR set computation algorithm enters into the while loop as seen in Appendix A. The number of iterations of the loop is upper bounded by $O(\min(|\mathcal{N}_2(v_i)|, |\mathcal{N}_1(v_i)|))$, which is equal to $O(v)$ according to our notation. In addition, finding the node with the maximum adjacency degree, which is executed inside the while loop, is $O(v^2)$. Therefore, the total running time of the MPR set computation algorithm is $O(v^3)$.

Lemma 3 *Let $|\mathcal{N}_1(v_i) \cap \mathcal{T}(v_i)|$ and $|\mathcal{N}_2(v_i) \cap \mathcal{T}(v_i)|$ denote, respectively, the number of one- and two-hop neighbors in $\mathcal{T}(v_i)$, and v_{in} the number of nodes from $\mathcal{N}_1(v_i) \cap \mathcal{T}(v_i)$ which we assume is equal to the number of nodes from $\mathcal{N}_2(v_i) \cap \mathcal{T}(v_i)$ used in the first step of Algorithm 2 to compute the AMPR set. In addition, let $|\mathcal{N}_1(v_i) \cap \overline{\mathcal{T}(v_i)}|$ and $|\mathcal{N}_2(v_i) \cap \overline{\mathcal{T}(v_i)}|$ denote, respectively, the number of one- and two-hop neighbors in $\overline{\mathcal{T}(v_i)}$, and v_{out} the number of nodes from $\mathcal{N}_1(v_i) \cap \overline{\mathcal{T}(v_i)}$ which we assume is equal to the number of nodes from $\mathcal{N}_2(v_i) \cap \overline{\mathcal{T}(v_i)}$ used in the second step of Algorithm 2. Thus, the running time needed to a WPR node to compute its AMPR set considering the least efficient case, i.e. $\mathcal{R}(v_i) \subset \mathcal{N}_1(v_i)$, is $O(v_{in}^2 + v_{out}^2)$.*

Proof In WPR, the AMPR set computation is similar to computing the MPR set twice. Nevertheless, the set of one-hop and two-hop neighbors used in both runs are subsets of the one-hop and two-hop neighbor sets used by OLSR. In the least efficient case, computing the nodes in the AMPR set in charge of forwarding controlled-flooding messages requires $O(v_{in}^2)$ steps. Similarly, computing AMPR nodes to reach nodes *not* in $\mathcal{T}_i$ is $O(v_{out}^2)$. Note that in the least efficient case, Algorithm 2 does not enter into the while loop in function `compute_mpr`. Hence, the complexity to compute the complete AMPR set is $O(v_{in}^2 + v_{out}^2)$. It is worth mentioning that $v_{in} + v_{out}$ can be less or equal to v because nodes not in $\mathcal{T}(v_i)$ can be already reached by nodes in the AMPR set before the second step of Algorithm 2. Therefore, v_{out} can be less than or equal to $|\mathcal{N}_1(v_i) \cap \overline{\mathcal{T}(v_i)}| = |\mathcal{N}_2(v_i) \cap \overline{\mathcal{T}(v_i)}|$. Additionally, the one- and two-hop ascendents may not be considered in v_{in} because the one-hop ascendent is required in the AMPR set whenever it exists. Therefore, v_{in} can be less than or equal to $|\mathcal{N}_1(v_i) \cap \mathcal{T}(v_i)| = |\mathcal{N}_2(v_i) \cap \mathcal{T}(v_i)|$.

Computing ascendent and descendent sets do not represent additional computation. The ascendent set is obtained with routing computation and the descendent set with `HELLO` messages.

Lemma 4 *The running time of the proposed AMPR set computation algorithm is $O(v_{in}^3 + v_{out}^3)$, if we do not consider the least efficient case, i.e. $\mathcal{R}(v_i) \subset \mathcal{N}_1(v_i)$.*

PROOF Similarly to Lemma 2, we have that the higher-order procedures are the two while loops and their respective bodies. Computing the AMPR set to reach all nodes in $\mathcal{T}(v_i)$ is $O(v_{in}^3)$ and computing the AMPR set to reach nodes in $\overline{\mathcal{T}(v_i)}$ is v_{out}^3. Therefore, the total running time is $O(v_{in}^3 + v_{out}^3)$.

Theorem 1 *The number of steps needed to compute the AMPR set is less than or equal to the number of steps needed to compute the MPR set.*

Proof As defined in Lemma 1, the number of one- and two-hop neighbors of v_i is the same and equal to v. On the other hand, as defined in Lemma 3, $v_{in} + v_{out} \leq v$. Therefore, the number of steps taken to compute the MPR set by an OLSR node is greater than or equal to the number of steps a WPR node requires to compute the AMPR set. In the least efficient case, because $v_{in} + v_{out} \leq v$, then $v_{in}^2 + v_{out}^2 < v^2$, for $v_{in}, v_{out} > 0$. Likewise, in the general case seen in Lemmas 2 and 4, $v_{in}^3 + v_{out}^3 < v^3$, also for $v_{in}, v_{out} > 0$.

Lemma 5 *Let $|\mathcal{M}(v_i)^*|$ denote the minimum number of nodes in the MPR set and v the number of two-hop neighbors of v_i. The solution found using the MPR set computation algorithm is $O(\ln v)$ worse than the optimal solution in terms of number of nodes.*

Proof We overview the proof of [20]. Considering the first set picked by the MPR set computation algorithm in the while loop has at least $v/|\mathcal{M}(v_i)^*|$ nodes, which is the average size, the number of nodes left for the second iteration is

$$v_1 \leq v - \frac{v}{|\mathcal{M}(v_i)^*|} = v\left(1 - \frac{1}{|\mathcal{M}(v_i)^*|}\right). \tag{5}$$

Similarly,

$$v_2 \leq v_1\left(1 - \frac{1}{|\mathcal{M}(v_i)^* - 1|}\right). \tag{6}$$

Note that the optimal number of nodes in the MPR set is decremented because the first node was already used in the first iteration of the while loop. Nevertheless, Eq. 6 can be upper bounded by

$$v_2 \leq v_1 \left(1 - \frac{1}{|\mathcal{M}(v_i)^*|}\right).$$

Combining Eqs. 5 and 6 we have that

$$v_2 \leq v \left(1 - \frac{1}{|\mathcal{M}(v_i)^*|}\right)^2. \tag{7}$$

Generalizing Eq. 7 for k iterations, we have the following expression:

$$v_k \leq v \left(1 - \frac{1}{|\mathcal{M}(v_i)^*|}\right)^k.$$

Therefore, considering that the number of iterations to compute the MPR set is equal to the number of nodes selected, we have that $|\mathcal{M}(v_i)| = k$. The algorithm terminates when $v(1 - 1/|\mathcal{M}(v_i)^*|)^k < 1$ because it cannot insert another one-hop node into $\mathcal{M}(v_i)$ set. Performing some algebraic manipulation, we have that

$$\begin{array}{lcl} & v\left(1 - \frac{1}{|\mathcal{M}(v_i)^*|}\right)^k & < 1 \\ \Rightarrow & \left(1 - \frac{1}{|\mathcal{M}(v_i)^*|}\right)^{|\mathcal{M}(v_i)^*| \frac{k}{|\mathcal{M}(v_i)^*|}} & < \dfrac{1}{v} \\ \Rightarrow & e^{-\frac{k}{|\mathcal{M}(v_i)^*|}} & < \dfrac{1}{v} \\ \Rightarrow & k & < |\mathcal{M}(v_i)^*| \ln v. \end{array}$$

This proves that the number of nodes in $\mathcal{M}(v_i)$ is $O(\ln v)$ worse than $|\mathcal{M}(v_i)^*|$.

Lemma 6 *In WPR, the number of nodes found by the AMPR set computation algorithm, $|\mathcal{R}(v_i)|$, is $O(\ln(v_{in}v_{out}))$ worse than the optimal solution $|\mathcal{M}(v_i)^*|$ in terms of number of nodes, if $v_{in}, v_{out} > 0$.*

Proof Let $|\mathcal{R}(v_i)'^*|$ and $|\mathcal{R}(v_i)''^*|$ denote the optimal number of nodes in the AMPR set in the first and in the second steps of Algorithm 2, respectively. The proof is similar to Lemma 5. In this case, however, we divide each set $\mathcal{N}_1(v_i)$ and $\mathcal{N}_2(v_i)$ in two.

Considering first the one- and two-hop sets in $\mathcal{T}(v_i)$, we have that

$$v_{in,k'} \leq v_{in}\left(1 - \frac{1}{|\mathcal{R}(v_i)'^*|}\right)^{k'}, \tag{8}$$

where $v_{in,k'}$ is the number of remaining two-hop nodes in set $\mathcal{T}(v_i)$ after k' iterations. Because $|\mathcal{M}(v_i)^*| \geq |\mathcal{R}(v_i)'^*|$ (Proof in Appendix B), we can upper bound Eq. 8 by

$$v_{in,k'} \leq v_{in}\left(1 - \frac{1}{|\mathcal{M}(v_i)^*|}\right)^{k'}. \tag{9}$$

Similarly to Lemma 5, $k' \leq |\mathcal{M}(v_i)^*| \ln v_{in}$. Besides $v_{in,k'}$, we can also compute $v_{out,k''}$, where $v_{out,k''}$ is the number of remaining two-hop nodes in set $\overline{\mathcal{T}(v_i)}$ after k'' iterations. Equivalently to Eqs. 8 and 9, we have

$$v_{out,k''} \leq v_{out}\left(1 - \frac{1}{|\mathcal{R}(v_i)''^*|}\right)^{k''} \leq v_{out}\left(1 - \frac{1}{|\mathcal{M}(v_i)^*|}\right)^{k''}. \tag{10}$$

In this case, $k'' \leq |\mathcal{M}(v_i)^*| \ln v_{out}$. Therefore, the total number of nodes in the AMPR set, $|\mathcal{R}(v_i)|$, is upper bounded by

$$\mathcal{R}(v_i) = k' + k'' \leq |\mathcal{M}(v_i)^*| \ln(v_{in}v_{out}).$$

As observed, the number of nodes in $\mathcal{R}(v_i)$ is $O(\ln(v_{in}v_{out}))$ worse than the optimal value.

Theorem 2 *The AMPR set computation algorithm has a higher upper bound than the MPR set computation algorithm for $v < v_{in}v_{out}$ taking into account the number of nodes chosen by each one compared with the optimal solution.*

Proof Lemma 5 proves that the number of nodes in the MPR set is $O(\ln v)$ worse than the optimal case whereas Lemma 6 proves that the number of nodes in the AMPR set is $O(\ln(v_{in}v_{out}))$ worse than the optimal case. Although $v_{in} + v_{out} \leq v$ (Lemma 3), the upper bound found for the AMPR set computation algorithm is higher. This happens because $v_{in}v_{out}$ can be higher than v even if $v_{in} + v_{out} \leq v$. This difference becomes more relevant as v_{in} and v_{out} increase. In practice, the higher upper bound obtained is a consequence of using different sets in the first step of the AMPR set computation algorithm and in the MPR set computation algorithm. The former considers a subset of the one- and two-hop neighbors whereas the latter considers the complete sets.

Lemma 7 *Let n denote the number of nodes in the network. Therefore, the number of controlled-flooding messages sent using OLSR is $O(n^2)$.*

Proof Considering again that the MPR set of a node v_i is equal to its one-hop neighbor set, the number of times a routing message from v_i is transmitted in the network is n. First the message is sent from its originating node, after it is forwarded by all its one-hop neighbors, and then by all its two-hop neighbors, and so forth, until the message is forwarded by all network nodes. Each routing message is then forwarded n times. Because all n nodes periodically flood the network and each message is forwarded n times, the number of controlled-flooding messages is $O(n^2)$ if considering one round of routing updates per node.

Lemma 8 *The number of controlled-flooding messages sent using the WPR is* $O(\sum_{i=1}^{n}(|\mathcal{A}(v_i)| + |\mathcal{D}(v_i)| + 1))$.

Proof The number of times a controlled-flooding message from a node v_i is forwarded is equal to $|\mathcal{A}(v_i)| + |\mathcal{D}(v_i)| + 1$, where the last term 1 corresponds to the transmission of node v_i itself. This is the worst case of WPR because all nodes within $\mathcal{A}(v_i)$ and $\mathcal{D}(v_i)$ sets must forward the message. Considering that there are n nodes in the network, the average number of ascendents and descendents in the network is $\sum_{i=1}^{n} |\mathcal{A}(v_i)|/n$ and $\sum_{i=1}^{n} |\mathcal{D}(v_i)|/n$, respectively. Therefore, the total number of controlled-flooding messages sent in average per node is $\sum_{i=1}^{n}(|\mathcal{A}(v_i)| + |\mathcal{D}(v_i)| + 1)/n$. Considering that we have n nodes in the network, then the number of controlled-flooding messages is $O(\sum_{i=1}^{n}(|\mathcal{A}(v_i)| + |\mathcal{D}(v_i)| + 1))$.

We do not consider the overhead reduction because of gateway proximity as we are dealing with upper bounds.

Theorem 3 *The WPR protocol introduces less overhead than OLSR.*

Proof To prove that WPR introduces less overhead than OLSR, we must show that

$$\sum_{i=1}^{n}(|\mathcal{A}(v_i)| + |\mathcal{D}(v_i)| + 1) < n^2, \qquad (11)$$

according to Lemma 7 and 8. Considering that $\sum_{i=1}^{n} |\mathcal{A}(v_i)| = \sum_{i=1}^{n} |\mathcal{D}(v_i)|$ (Proof in Appendix C.) and after some algebraic manipulation, Eq. 11 becomes $\sum_{i=1}^{n} |\mathcal{A}(v_i)| < \frac{n(n-1)}{2}$. The right-side term of the inequality is equivalent to an arithmetic progression of n terms with common difference equal to 1 and initial term equal to zero. This topology is only obtained in a forwarding chain where the number of ascendents in the network is maximum. Therefore, in all possible topologies WPR introduces less overhead than OLSR except for the forwarding-chain topology.

4.1. Analysis Summary

Table 1 summarizes all results obtained in this section. An interesting observation is the opposite behavior of the AMPR set computation algorithm and the controlled-flooding algorithm compared with their counterparts of OLSR. The AMPR set introduces more redundant messages whereas the controlled flooding reduces the amount of control messages forwarded throughout the network. Our results show that despite this effect, WPR significantly reduces the total number of routing messages sent in wireless mesh networks.

	Protocols	
Parameters	**OLSR**	**WPR**
Running time	$O(v^2)$	$O(v_{in}^2 + v_{out}^2)$
Optimality	$O(\|\mathcal{M}(v_i)^*\| \ln v)$	$O(\|\mathcal{M}(v_i)^*\| \ln(v_{in} v_{out}))$
Number of controlled-flooding messages	$O(n^2)$	$O(\sum_{i=1}^{n}(\|\mathcal{A}(v_i)\| + \|\mathcal{D}(v_i)\| + 1))$

Table 1: Comparison summary.

5. Simulation Setup

We analyze the performance of WPR with Network Simulator 2 version 31 [29]. In this section, we improve the available PHY-layer and the IEEE 802.11 MAC-layer modules. We also present WPR implementation issues as well as we describe the scenario and the traffic pattern used in our simulations.

5.1. Physical and MAC Layers

The original PHY-layer module of ns-2.31 does not consider bit error rate (BER) and computes the maximum interference and reception ranges according to pre-determined values of transmission and reception power. We adapt to ns-2.31 the PHY-layer module developed by Xiuchao and Ananda [30] and then the BER is computed as a function of the channel signal-to-noise-ratio (SNR) and the physical transmission rates defined by IEEE 802.11b. The packet loss probability associated depends on the BER computed per frame. Therefore, we simulate an error prone wireless channel. The module is based on experimental results, thus it considers possible variations on medium conditions. We use the shadowing model to compute transmission ranges and set parameters according to the experimental results suggested in [31] and summarized in Table 2.

The IEEE 802.11 standard does not define a rate-control algorithm to handle the multiple physical rates. We implement the AutoRate Fallback (ARF) algorithm [32] in the MAC-layer module of ns-2.31. The operation of the algorithm is based on two counters and a timer. Basically, each node decreases its PHY-rate to the next value when it performs two consecutive

Parameters	**Values**
Transmission power	3.162×10^{-2} W
Operation frequency	2.472 GHz
Carrier sense threshold	3.162×10^{-14} W
Reception threshold	3.162×10^{-13} W
Shadowing path-loss exponent	4.000

Table 2: PHY-layer parameters.

retransmissions. On the other hand, each node increases its PHY-rate to the next value when ten consecutive frames are delivered or when a 60 ms timer expires. Whenever a frame is lost, both timer and successful transmission counter are reset. The ARF algorithm and the PHY-layer module are complementary. It is worth mentioning that without a rate-control algorithm, the BER of a channel is only a function of the signal-to-noise-ratio because the transmission rate is constant.

5.2. Routing Layer

We add the ETX metric to the OLSR module [33] according to one of its daemon implementation [22]. In such implementation, `HELLO` messages are used as probes to estimate the quality of each adjacent link. The `HELLO` message header includes two fields: Link Quality (LQ) and Neighbor Link Quality (NLQ). LQ indicates the fraction of `HELLO` messages received by node v_i from each neighbor in the last 20 s time window, whereas NLQ indicates the fraction of `HELLO` messages sent by node v_i that each neighbor received. The first value is obtained by counting the number of received `HELLO` messages in the last 20 s interval, and the second is obtained from the LQ field of the received `HELLO` messages. Each node computes $ETX = 1/(LQ \times NLQ)$ for each adjacent link.

In our simulations, both OLSR and WPR protocols use the ETX metric. WPR uses the proposed controlled-flooding algorithm and the algorithm to compute the AMPR set. In our simulations, both WPR and OLSR periodically send route updates at intervals of 5 s, as suggested in RFC 3626 of OLSR [34]. The minimum relation used by WPR between controlled-flooding and network-wide flooding messages (R_{min}) is 13. This relation is also used by the Fisheye extension of OLSR and, in WPR, is adjusted according to the number of hops needed to reach the gateway (Section 3).

5.3. Traffic Pattern and Simulation Scenario

We use two different traffic patterns in our simulations: web and combined traffic. The web traffic models requests for Internet web pages, which generate multiple responses containing one object each. The size of the request is constant and equal to 1 kbyte. In addition, the interval of time between different pages and between the transmission of consecutive objects is represented by a random variable exponentially distributed with average equal to 10 s. The size of each object is modeled according to a random variable with Paretto II distribution with shape 1.2 and average of 12 kbytes. The web traffic uses TCP and is bidirectional, where the request for a web page is always initiated by a backbone node to a wired node. We assume that the bandwidth bottleneck is the wireless part. Therefore, we do not simulate the influence of the wired network. The combined traffic is a mix of internal and web traffic. The internal traffic models communications between nodes within the wireless mesh. The traffic is modeled using constant bit rate (CBR) sources over UDP transmitting data at 56 kb/s. The duration of each flow is computed according to a random variable exponentially distributed with average of 48 s. CBR as well as web traffic sources are randomly chosen among all backbone routers but the gateway. Although the backbone routers are not the traffic sources, they are in charge of injecting traffic from users into the network. Therefore, for our simulation purposes the effect is the same. The duration of each simulation run is 90 s. These parameters model the typical traffic in wireless mesh networks [35].

Nodes are positioned in a grid with the gateway at one vertex. Hence, we do not consider any algorithm to improve gateway positioning [36]. The size of the network ranges from 9 to 49 nodes separated by 20 m between each other. Thus, each node has at most 20 neighbors at 1 Mb/s and 8 at 11 Mb/s according to our PHY-layer parameters (Table 2). We use only one gateway and all backbone nodes are equivalent. Therefore, we neither have different configurations nor different hardware in the network.

6. Simulation Results

The performance of WPR is compared with OLSR using five metrics: routing control traffic overhead, network aggregated throughput, delivery rate at the application level, average number of multipoint relays, and average hop distance to the gateway. We compute the control traffic overhead as the aggregated traffic generated by routing messages in the network. The aggregated throughput is the sum of all individual throughputs obtained from data communications in the network. The delivery rate is computed as the total amount of data received by destination nodes over the total amount of data sent by source nodes. Whenever a node runs its multipoint relay set computation algorithm (i.e. the MPR set computation algorithm for an

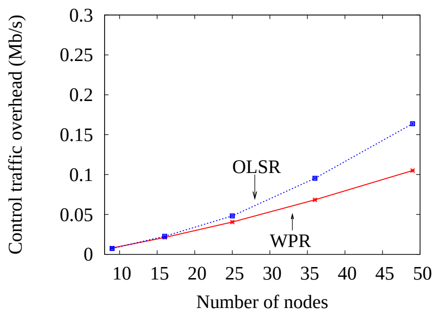
OLSR node and the AMPR set computation algorithm for a WPR node), we log the size of the set found. Then, we compute offline the average size over the multiple algorithm executions. Finally, we also log the distance in hops from every backbone router to the gateway, whenever it runs the Dijkstra algorithm. The average hop distance is then the sum of all distances found divided by the number of times the Dijkstra algorithm is run. All results are obtained using a confidence level of 95% represented by vertical bars in our plots.

6.1. Web Traffic

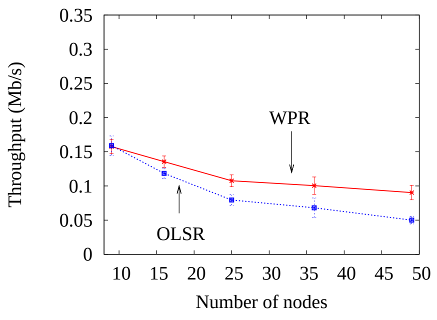
We consider web traffic to perform a favorable analysis. Considering all traffic sent toward the gateway, we benefit our controlled-flooding algorithm because we only use those links connecting backbone nodes to the gateway and vice-versa. In these simulations, we aim at analyzing the performance of WPR as the number of nodes increases.

In a first set of simulations, we consider that 75% of the backbone nodes produce traffic with the number of sessions equal to 20. We avoid network saturation by limiting the number of sources in the network. Each session is a sequence of web-page requests and responses of the respective objects. Figure 4(a) plots the control traffic overhead introduced by OLSR and WPR. Observe that WPR reduces the routing overhead up to 36%. Therefore, the behavior of both curves agrees with Theorem 3, which corroborates the algorithm analysis derived in Section 4. Moreover, we show that the benefits of the controlled-flooding algorithm overcome the additional redundant messages the AMPR set may introduce compared with the MPR set. The total overhead reduction is more significant as the number of nodes grows because the ratio between the nodes within the ascendent and descendent sets of a given node and the total number of nodes in the network decreases. Thus, the routing overhead of WPR shows a slower increase when compared with OLSR. This shows that WPR scales better than OLSR. Figure 4(b) plots the aggregated throughput obtained by WPR and OLSR. Note that reducing the routing overhead has a direct impact on the network aggregated throughput. As mentioned earlier, especially in wireless mesh networks, reducing overhead results in throughput gains because it likely reduces traffic on network bottlenecks. The aggregated throughput of WPR almost doubles the throughput obtained with OLSR. On the other hand, a better throughput does not imply in better delivery rate. For instance, one could obtain a better delivery rate by simply reducing the amount of data traffic injected in the network. Figure 4(c) shows that this is not the case. Reducing the amount of control overhead, we improve throughput and also the delivery rate by selecting better end-to-end quality paths. Figure 4(c) shows that delivery rate of WPR is up to 20% better than OLSR.

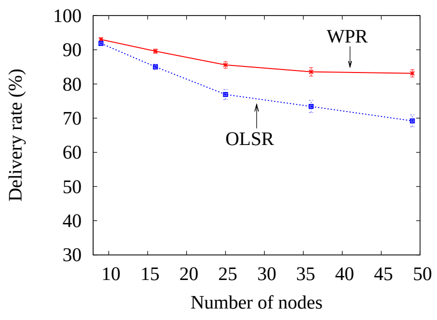
Figure 4(d) shows the number of nodes within MPR and AMPR sets. Note that the MPR set has, in average, fewer nodes than the AMPR set. This result is expected from our mathematical analysis as proved by Theorem 2 that shows that the AMPR set computation algorithm is less efficient than its counterpart in OLSR because it has a higher upper bound in the number of nodes. Although WPR has a larger number of multipoint relay nodes, the amount of overhead reduced because of the controlled-flooding algorithm overweighs this additional overhead and makes the overall performance of WPR better than OLSR. Figure 4(e) shows that OLSR uses longer paths than WPR because the higher overhead leads to network bottlenecks. Hence, OLSR tries to circumvent such bottlenecks using longer paths.



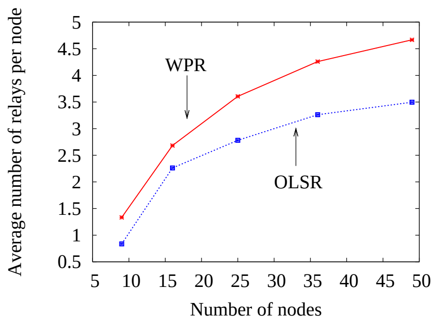
(a) Overhead.



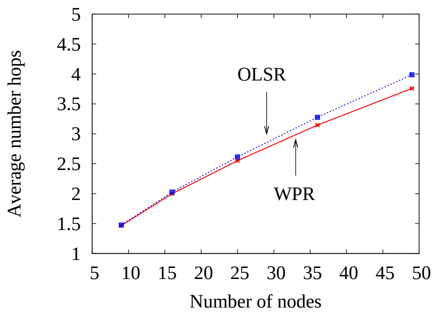
(b) Throughput.



(c) Delivery rate.



(d) Number of multipoint relays.



(e) Hop distance to the gateway.

Figure 4: Performance results obtained for traffic composed of web-page requests and responses.

We have also run a variant of WPR which does not use the proposed $c(n)$ function. The goal is to check the impact of this function on the amount of control overhead. Figure 5 plots the average number of TC messages sent per second by all network nodes. In our scenario, we have noted only a slight difference between using or not the $c(n)$ function. We give a zoom in the plot to show that $c(n)$ function indeed introduces less overhead, especially when there are more nodes in the network. Therefore, the impact would become more relevant in denser scenarios as well as scenarios with longer paths to the gateway. Because it does not yield additional findings, we do not include this test in the other scenarios. Nevertheless, a deeper investigation using different $c(n)$ function would be interesting.

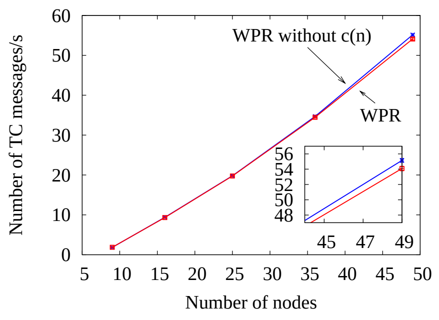
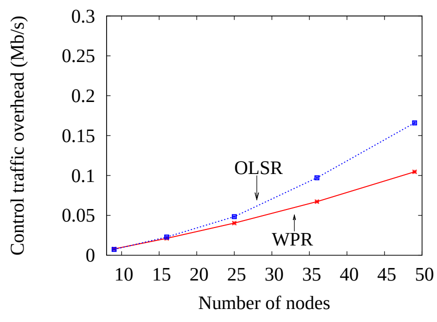


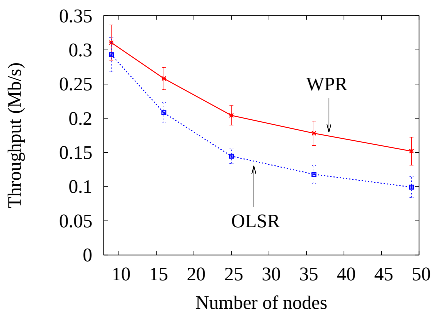
Figure 5: Impact of $c(n)$ function.

We have also changed the node density by increasing the distance between consecutive nodes from 20 m to 30 m. In this different scenario, we note that the throughput falls in both cases, i.e. in OLSR and in WPR. This is expected because the more distant the nodes, the higher the number of hops to reach the gateway. It is worth mentioning that the throughput of WPR is still better than the throughput of OLSR. The difference between them, however, decreases because the controlled-flooding algorithm is more efficient in denser scenarios, as well as the MPR and AMPR set computation algorithms. We do not add those plots because they are very similar to the previous ones.

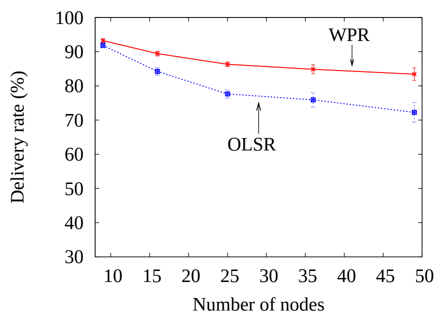
Figure 6 shows the results obtained increasing the web traffic injected in the network. We multiply by two the amount of traffic sent per node toward the gateway. The goal is to verify the performance of WPR with higher traffic load. We can observe in Figure 6(a) that the traffic control overhead remains the same compared with Figure 4(a). Because routing messages are periodically sent, the only way the overhead could increase would be by sending more triggered updates as a consequence of link breakages. Nevertheless, since we do not verify an overhead increase in our results, we infer that link breakages do not occur very often. This is because using a rate-control algorithm, IEEE 802.11 reduces its PHY-rate to avoid link breakages. Figure 6(b) shows that the network aggregated throughput grows compared with the same throughput obtained in Figure 4(b) for the same number of nodes. This demonstrates that the network can transmit more traffic and that WPR achieves better performance than OLSR. The aggregated throughput of WPR is up to 50% better than the throughput obtained with OLSR. The delivery rate shown in Figure 6(c) is similar to the previous scenario. This occurs because the amount of traffic added was not enough to impact the packet delivery rate. Figure 6(d) shows a result similar to that of Figure 4(d). Again, the average size of the AMPR set is higher than the size of the MPR set. Figure 6(e), similarly to Figure 4(e), shows also that OLSR uses longer paths, showing a slight increase in both cases because of



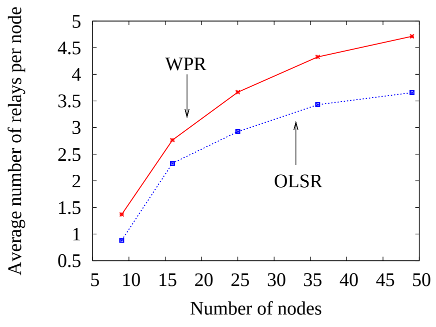
(a) Overhead.



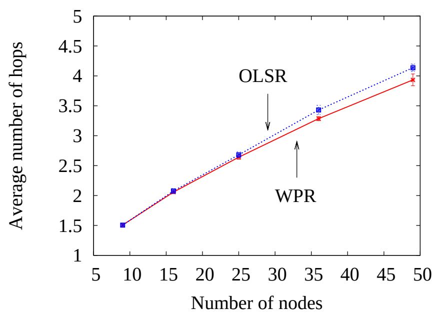
(b) Throughput.



(c) Delivery rate.



(d) Number of multipoint relays.



(e) Hop distance to the gateway.

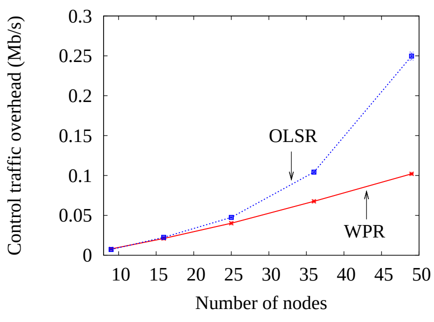
Figure 6: Performance results obtained for high-load traffic composed of web-page requests and responses.

the higher network load.

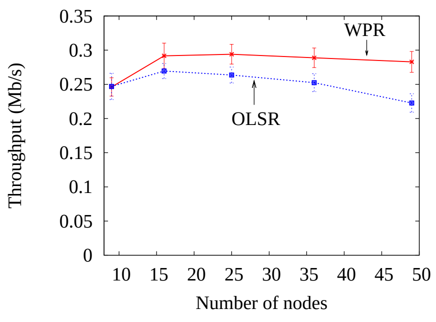
6.2. Combined Traffic

In the second simulation set, we combine web and internal traffic, i.e. traffic to and from nodes inside the mesh, to evaluate WPR performance in a non-favorable scenario. As we concentrate route updates on links toward the gateway, the topology maps known by the different nodes may become different from times to times. This may affect communications between nodes not within the ascendent or descendent set of each other. In this experiment, 25% of the backbone nodes produce web traffic using 20 sessions and other 25% of the backbone nodes are Constant Bit Rate (CBR) sources.

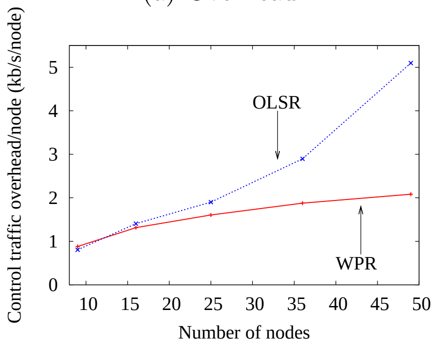
Figure 7(a) plots the control traffic overhead introduced by OLSR and WPR. We observe that WPR reduces the routing overhead by up to 60%. Note that the overhead of WPR remains the same, compared with Figures 4(a) and 6(a), whereas the overhead of OLSR grows. This indicates that OLSR is more susceptible to the increase on the network load, because the triggered routing messages result in larger number of forwarded control traffic. Figure 7(b) shows the aggregated throughput obtained by WPR and OLSR. Again, the aggregated throughput of WPR is better than OLSR. The difference can be as high as 27%. We observe, however, a different behavior in the aggregated throughput as compared with our web traffic experiments. In the previous experiment there is a congestion control algorithm in action as packets are lost. In this experiment, on the other hand, we use CBR over UDP traffic combined to the web traffic over TCP. In this case, the UDP does not react to congestion and the throughput increases from 9 to 16 nodes as the offered CBR load grows, until the network saturates. We plot the control overhead divided by the number of nodes to check the amount of overhead



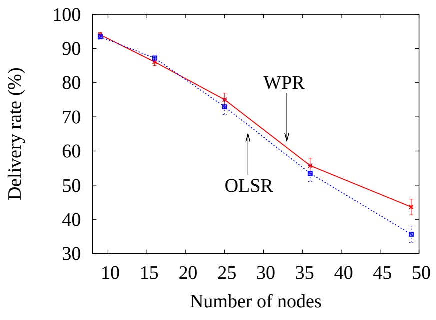
(a) Overhead.



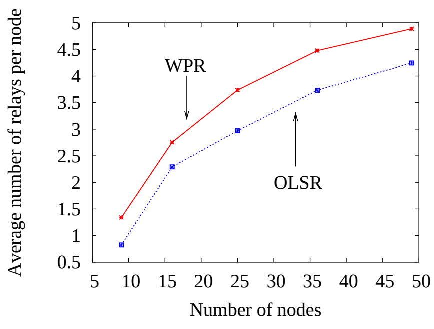
(b) Throughput.



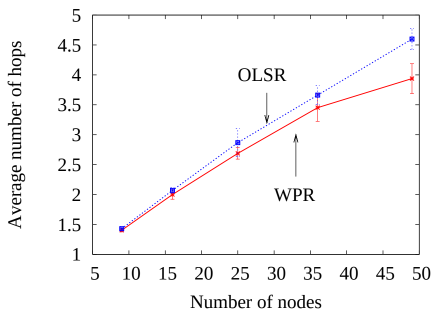
(c) Overhead.



(d) Delivery rate.



(e) Number of multipoint relays.



(f) Hop distance to the gateway.

Figure 7: Performance results obtained for combined web and internal traffic.

introduced per node. Figure 7(c) shows that WPR indeed scales better than OLSR in our scenario since the amount of control traffic introduced per node using WPR increases in a log-scale basis whereas it increases exponentially using OLSR.

Presenting a better throughput, however, does not imply in better delivery rate. This also depends on the end-to-end quality of the chosen path as discussed in our results for web traffic. The controlled-flooding algorithm deployed by WPR concentrates route updates on some nodes. Figure 7(d) shows that despite our assumption on traffic convergence toward gateways, WPR shows a delivery rate at the application layer up to 22% better than OLSR even in presence of internal traffic. Nevertheless, the tradeoff of using CBR traffic is observed on this evaluation metric. Note that because the offered load continues to grow, the delivery rate decreases faster than in the web traffic experiment. Figure 7(e) shows that the average size of the MPR set increases compared with the same result for web traffic. This result also confirms that OLSR is more affected by the increase on the network load because the protocol selects more MPR nodes to guarantee routing messages reception by all two-hop neighbors. Following the same tendency, the average hop distance to the gateway also increases as shown in Figure 7(f).

7. Conclusion

This work proposed WPR, a link-state routing protocol for wireless mesh networks. WPR is characterized by the controlled-flooding and by the AMPR (Adapted MultiPoint Relay) set computation algorithms. The former reduces the flooding procedure overhead whereas the latter avoids redundant routing messages. Both algorithms are tailored to the typical traffic pattern of wireless access networks. Our analysis shows that the AMPR set computation algorithm has a lower running time than the computation of the MPR (Multi-

Point Relay) set used by the Optimized Link-State Routing (OLSR) protocol. Nevertheless, the AMPR set computation algorithm provides a higher upper bound on the number of relay nodes than the algorithm to compute the MPR set. Concerning the performance of the controlled-flooding algorithm, we demonstrate that WPR introduces less controlled-flooding messages than OLSR. The performance of WPR is also investigated via simulation analysis. Our results corroborate our complexity analysis, and confirm the better performance of WPR compared with OLSR in terms of amount of overhead, aggregated throughput, and delivery rate. These results are obtained for favorable scenarios in which all traffic flows toward the gateway, and also for non-favorable scenarios in which there is a combination of traffic toward the gateway and internal traffic. Therefore, even reducing the amount of routing information used to maintain routers' forwarding tables updated, WPR outperforms OLSR because its algorithms yield a more appropriate use of the wireless resources considering particular characteristics of the wireless mesh networks. Even though we do not consider mobility and network partitions, using more efficient routing algorithms is important because the network is prone to frequent changes. Currently, we are implementing WPR in `olsrd` to evaluate the performance of WPR in a prototype.

Appendix A. MPR Set Computation Algorithm

Algorithm 3 shows the greedy heuristic used by OLSR to compute the MPR set [6]. This algorithm is executed by a node v_i in the network. Lines 1 and 2 show variables initialization. In Line 3, the algorithm inserts into the MPR set, $\mathcal{M}(v_i)$, all one-hop neighbors, $\mathcal{N}_1(v_i)$, that are the only relay node

Algorithm 3 MPR SET COMPUTATION ALGORITHM.

```
Require: $\mathcal{N}_1(v_i)$ and $\mathcal{N}_2(v_i)$
Ensure: $\mathcal{M}(v_i)$
1: $\mathcal{M}(v_i) \leftarrow \emptyset$
2: $\mathcal{NN} \leftarrow \mathcal{N}_2(v_i)$
3: $\mathcal{M}(v_i) \leftarrow \{v_k \in \mathcal{NN} \mid |\mathcal{N}_1(v_i) \cap \mathcal{N}_1(v_k)| = 1\}$
4: $\mathcal{NN} \leftarrow \mathcal{NN} - \bigcup_{v_j \in \mathcal{M}(v_i)} \mathcal{N}_1(v_j)$
5: while $\mathcal{NN} \neq \emptyset$ do
6:     compute $v_j \in \mathcal{N}_1(v_i)$ that maximizes $|\mathcal{N}_1(v_j) \cap \mathcal{NN}|$
7:     $\mathcal{M}(v_i) \leftarrow \mathcal{M}(v_i) \cup \{v_j\}$ \* Add $v_j$ to $\mathcal{M}(v_i)$ *\
8:     $\mathcal{NN} \leftarrow \mathcal{NN} - \mathcal{N}_1(v_j)$
9: end while
```

to a two-hop neighbor in the two-hop neighbor set denoted by $\mathcal{NN}$. The algorithm then removes from $\mathcal{NN}$ the two-hop descendents already covered by these first $\mathcal{M}(v_i)$ nodes. The while loop executes until all two-hop neighbors are covered by at least one node in $\mathcal{M}(v_i)$. At each iteration, the algorithm inserts into $\mathcal{M}(v_i)$ the one-hop neighbor that covers the maximum number of remaining two-hop neighbors in $\mathcal{NN}$.

Appendix B. Proof of Claim in Lemma 6

In Lemma 6, we claim that the optimal number of nodes in the MPR set is less than or equal to the optimal number of nodes in any of the two subsets found by the AMPR set computation algorithm. Hence,

$$|\mathcal{M}(v_i)^*| \geq |\mathcal{R}(v_i)'^*| \text{and} |\mathcal{M}(v_i)^*| \geq |\mathcal{R}(v_i)''^*|.$$

PROOF This proof is by contradiction. Considering that

$$|\mathcal{M}(v_i)^*| < |\mathcal{R}(v_i)'^*|,$$

$\mathcal{M}(v_i)^*$ would not be able to cover all two-hop neighbors of v_i. Since $\mathcal{R}(v_i)^*$ is optimal and it is able to cover a subset of two-hop neighbors of v_i, the complete set of two-hop neighbors would require a greater number of nodes in $\mathcal{M}(v_i)^*$ set. In this case, $\mathcal{M}(v_i)^*$ would not be optimal which is not in accordance with our initial claim. The proof for $|\mathcal{M}(v_i)^*| \geq |\mathcal{R}(v_i)''^*|$ is equivalent.

Appendix C. Proof of Claim in Theorem 3

In Theorem 3, we claim that the total number of ascendent and descendent nodes in a wireless mesh network is the same. Hence,

$$\sum_{i=1}^{n} |\mathcal{A}(v_i)| = \sum_{i=1}^{n} |\mathcal{D}(v_i)|.$$

Proof This proof is by induction. The proof of the base case is straightforward. If $n = 1$, the number of $|\mathcal{A}(v_i)| = |\mathcal{D}(v_i)| = 0$. In the step case, however, if we add a node to a tree, this node will be a leaf. Hence, the number of ascendents in a $(n + 1)$-node network becomes $\sum_{i=1}^{n} |\mathcal{A}(v_i)| + l$, where l is the length of the branch of the tree the node was connected. This represents the number of nodes in the ascendent set of the joining node. The number of descendents, on the other hand, becomes $\sum_{i=1}^{n} |\mathcal{D}(v_i)| + \sum_{j=1}^{l} 1$ because every node in the branch of the joining node adds one more descendent in their descendent sets. Therefore, the number of ascendents and descendents in the network remains the same.

Acknowledgment

The authors would like to thank CNPq, Faperj, CAPES, FINEP, and FUNTTEL for the work support. The authors would also like to thank Igor M. Moraes and Marcelo D. D. Moreira for their valuable comments.

References

[1] Akyildiz, I.F., Wang, X., Wang, W.. Wireless mesh networks: A survey. Computer Networks 2005;47(4):445–487.

[2] Robinson, J., Knightly, E.W.. A performance study of deployment factors in wireless mesh networks. In: IEEE Conference on Computer Communications (INFOCOM). 2007, p. 2054–2062.

[3] Robinson, J., Uysal, M., Swaminathan, R., Knightly, E.W.. Adding capacity points to a wireless mesh network using local search. In: IEEE Conference on Computer Communications (INFOCOM). 2008, p. 1247–1255.

[4] Waharte, S., Ishibashi, B., Boutaba, R., Meddour, D.E.. Design and performance evaluation of iar: Interference-aware routing metric for wireless mesh networks. Mobile Networks and Applications (MONET) 2009;14(5):649–660.

[5] Gawedzki, I., Agha, K.A.. Resilience to dropping nodes in mobile ad hoc networks with link-state routing. In: IFIP Networking. 2008, p. 99–111.

[6] Clausen, T., Jacquet, P., Laouiti, A., Muhlethaler, P., Qayyum, A., Viennot, L.. Optimized link state routing protocol. In: IEEE International Multi Topic Conference (INMIC). 2001, p. 62–68.

[7] Perkins, C., Bhagwat, P.. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: ACM SIGCOMM. 1994, p. 234–244.

[8] Perkins, C.E., Royer, E.B.. Ad-hoc on-demand distance vector routing. In: IEEE Workshop on Mobile Computing Systems and Applications. 1999, p. 90–100.

[9] David B. Johnson, D.A.M., Broch, J.. Ad Hoc Networking; chap. 5. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks; Addison-Wesley; 2001, p. 139–172.

[10] Ni, S.Y., Tseng, Y.C., Chen, Y.S., Sheu, J.P.. The broadcast storm problem in a mobile ad hoc network. In: ACM International Conference on Mobile Computing and Networking (MobiCom). 1999, p. 152–162.

[11] Costa, L.H.M.K., de Amorim, M.D., Fdida, S.. Reducing latency and overhead of route repair with controlled flooding. ACM/Kluwer Wireless Networks 2004;10(4):347–358.

[12] Campista, M.E.M., Passos, D.G., Esposito, P.M., Moraes, I.M., de Albuquerque, C.V.N., Saade, D.C.M., et al. Routing metrics and protocols for wireless mesh networks. IEEE Network 2008;22(1):6–12.

[13] Draves, R., Padhye, J., Zill, B.. Routing in multi-radio, multi-hop

wireless mesh networks. In: ACM International Conference on Mobile Computing and Networking (MobiCom). 2004, p. 114–128.

[14] Bicket, J., Aguayo, D., Biswas, S., Morris, R.. Architecture and evaluation of an unplanned 802.11b mesh network. In: ACM International Conference on Mobile Computing and Networking (MobiCom). 2005, p. 31–42.

[15] Pei, G., Gerla, M., Chen, T.W.. Fisheye state routing in mobile ad hoc networks. In: ICDCS Workshop on Wireless Networks and Mobile Computing. 2000, p. D71–D78.

[16] Chen, J., Lee, Y.Z., Maniezzo, D., Gerla, M.. Performance comparison of AODV and OFLSR in wireless mesh networks. In: IFIP Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net). 2006, p. 1–8.

[17] Campista, M.E.M., Costa, L.H.M.K., Duarte, O.C.M.B.. WPR: A proactive routing protocol tailored to wireless mesh networks. In: IEEE Globecom 2008 Ad Hoc, Sensor and Mesh Networking Symposium (GC AHSN). 2008, p. 1–5.

[18] Wellons, J., Dai, L., Xue, Y., Cui, Y.. Augmenting predictive with oblivious routing for wireless mesh networks under traffic uncertainty. Computer Networks 2010;54(2):178–195.

[19] Raniwala, A., Chiueh, T.C.. Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In: IEEE Conference on Computer Communications (INFOCOM). 2005, p. 2223–2234.

[20] Viennot, L.. Complexity results on election of multipoint relays in wireless networks. Tech. Rep.; INRIA Rocquencourt; 1998.

[21] Passos, D.G., de Albuquerque, C.V.N., Campista, M.E.M., Costa, L.H.M.K., Duarte, O.C.M.B.. Minimum loss multiplicative routing metrics for wireless mesh networks. Journal of Internet Services and Applications 2011;1(3):201–214.

[22] olsrd. Accessed in http://www.olsr.org; 2007.

[23] Nguyen-Vuong, Q.T., Agoulmine, N., Ghamri-Doudane, Y.. A user-centric and context-aware solution to interface management and access network selection in heterogeneous wireless environments. Computer Networks 2008;52(18):3358–3372.

[24] Camp, J.D., Knightly, E.W.. The IEEE 802.11s extended service set mesh networking standard. IEEE Communications Magazine 2008;46(8):120–126.

[25] Fernandes, N.C., Moreira, M.D.D., Duarte, O.C.M.B.. An efficient filter-based addressing protocol for autoconfiguration of mobile ad hoc networks. In: IEEE Conference on Computer Communications (INFOCOM). 2009, p. 2464–2472.

[26] Laufer, R., Dubois-Ferrière, H., Kleinrock, L.. Multirate anypath routing in wireless mesh networks. In: IEEE Conference on Computer Communications (INFOCOM). 2009, p. 37–45.

[27] Velloso, P.B., Laufer, R.P., Duarte, O.C.M.B., Pujolle, G.. Analyzing a human-based trust model for mobile ad hoc networks. In: IEEE

Symposium on Computers and Communications (ISCC). 2008, p. 240–245.

[28] He, J., Rexford, J.. Towards Internet-wide multipath routing. IEEE Network 2008;22(2):16–21.

[29] Fall, K., Varadhan, K.. The ns Manual. UC Berkeley, LBL, USC/ISI, and Xerox PARC; 2009.

[30] Xiuchao, W., Ananda, A.L.. Link characteristics estimation for IEEE 802.11 DCF based WLAN. In: IEEE Conference on Local Computer Networks (LCN). 2004, p. 302–309.

[31] Xiuchao, W.. Simulate 802.11b channel within NS2. Tech. Rep.; National University of Singapore; 2004.

[32] Holland, G., Vaidya, N.H., Bahl, P.. A rate-adaptive MAC protocol for multi-hop wireless networks. In: ACM International Conference on Mobile Computing and Networking (MobiCom). 2001, p. 236–251.

[33] Ros, F.J.. Accessed in http://masimum.dif.um.es/um-olsr/html/; 2005.

[34] Clausen, T., Jacquet, P.. Optimized link state routing protocol (OLSR). IETF Network Working Group RFC 3626; 2003.

[35] Baumann, R., Heimlicher, S., Lenders, V., May, M.. HEAT: Scalable routing in wireless mesh networks using temperature fields. In: IEEE WoWMoM. 2007, p. 1–9.

[36] Papadaki, K., Friderikos, V.. Gateway selection and routing in wireless mesh networks. Computer Networks 2010;54(2):319–329.