Centrality-based Route Recovery in Wireless Mesh Networks

Michele Segata, Nicolò Facchi, Leonardo Maccari, Gabriele Gemmi, Renato Lo Cigno
Dept. of Information Engineering and Computer Science, University of Trento, Italy
{name.surname}@unitn.it; gabriele.gemmi@studenti.unitn.it

Abstract—**Wireless Mesh Networks are subject to frequent node and link failures, and routing protocols currently used, such as Optimized Link State Routing (OLSR) or Babel, suffer from relatively long recovery times characterized by broken and looped routes due to long management timeouts that can not be shortened to keep the overhead at an acceptable level. This paper experiments a novel timer management technique named Pop-Routing on top of OLSR. Pop-Routing exploits the notion of betweenness centrality to tune timers depending on the node position in the network, so that failures that lead to larger traffic losses can be recovered faster. Pop-Routing maintains the overhead constant, but favors the most central nodes, whose failure is devastating from the performance point of view, and penalizes peripheral ones, whose failure has a very little impact on the entire network. Pop-Routing has been implemented as a plug-in in the OLSR daemon, coupled with an external process, named Prince, that computes centrality and timer values without interfering with the routing daemon. Experiments are run on the WiSHFUL[1] showing the benefit of Pop-tuning OLSR Hello and Traffic Control timers.**

I. Introduction

One of the key features that make link-state routing protocols attractive is their ability to rapidly react to a change in the network topology, for instance, after the failure of a node. This requires two phases, first, the nodes that are the neighbors of the failed one need to detect the failure and change their routing table accordingly. This local fix can possibly introduce routing loops, since nodes that are not direct neighbors do not know that the topology changed, and they might still use the old route. In the second phase the information about the change in the topology is propagated to all the other nodes in the network. After that, all the nodes have the same information base and take consistent routing decisions.

In a wireless mesh network the detection phase happens thanks to `HELLO` messages (`H`s from now on), which are broadcast messages generated by every node. `H`s are used to estimate the link quality and to perform link sensing, the loss of a sequence of `H`s is interpreted as a link failure. It is intuitive to understand that the lower is the timer used to generate `H`s, the faster is the protocol in detecting the link failure. On the other hand, the timer can not be arbitrarily reduced because control messages subtract resources to useful network traffic. A trade-off must be found between convergence speed and overhead. The propagation phase works in a similar way, and needs a similar trade-off (details are given in Section II-B).

After many years of research on routing protocols for wireless mesh networks the optimal solution to the trade-off problem was proposed with Pop-Routing [1]. Pop-Routing performs an optimal equalization of the generation timers that maintains the total overhead constant and minimizes the damage produced by a node failure. With Pop-Routing the timers of the nodes that are more *central* in the network topology (see Section II-A for a precise definition) are decreased, while the timers of the nodes that are more peripheral are increased. Therefore, the network recovers quickly from the failure of important nodes. Pop-Routing was designed for the Optimized Link State Routing (OLSR) protocol [2] and was so-far tested only in emulation.

This paper documents the implementation of Pop-Routing and the improvement in convergence speed in a real network. We analyze the process of route recovery in different network topologies and we measure the impact of node failures. We show that in a generic network most of the damage is due to the failure detection phase, while temporary loops have a lower impact. We quantify the gain that Pop-Routing introduces in absolute and relative terms, and we give valuable insights on the role of link-quality metrics in route recovery.

We show that Pop-Routing can be further improved in a way that is counter-intuitive, that is, removing from the optimization those nodes that are cut-point of the network topology graph. These nodes, albeit being central produce a physical network partitioning which simply can not be solved with routing, so that they could be removed from the optimization problem (see Section IV-A).

The paper is organized as follows. Section II provides some background on betweenness centrality, OLSR, Pop-Routing, and Prince. Section III describes the test-bed setup, the experiments, and the metrics used in the evaluation. Section IV analyzes the experiment results, while Section V concludes the paper and provides some future work ideas.

II. Background and Related Work

A. Centrality

Betweenness centrality is a graph metric that represents the fraction of all the shortest paths that pass through a single node. Given a network graph $G(\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ is the set of all nodes (of size N) and $\mathcal{E}$ is the set of edges (of size E) we

This work has been supported by the H2020 GA No. 645274 "Wireless Software and Hardware platforms for Flexible and Unified radio and network controL (WiSHFUL)" with the project "Pop-Routing On WiSHFUL (POPROW)" financed in Open Call 3.

[1]WiSHFUL is a European H2020 project supporting an experimental distributed testbed for wireless networks http://www.wishful-project.eu/

call $p_{i,j} = \{n_i, \ldots, n_j\}$ the sequence of nodes that represents the shortest path from node n_i to node n_j. Since in our case the graph represents an IP network, we assume that at each instant only one shortest path from n_i to n_j is used, so we consider $p_{i,j}$ unique. We call b_k the betweenness centrality of node n_k defined as:

$$b_k = \frac{1}{N(N-1)} \sum_{i=1}^{i=N} \sum_{\substack{j=1 \\ j \neq i}}^{j=N} I_{p_{i,j}}(n_k), \tag{1}$$

where $I_{p_{i,j}}(n_k)$ is the indicator function that takes value 1 if $n_k \in p_{i,j}$ and 0 otherwise. In a directed connected graph without self loops, for each node n_k there are exactly (N-1) destinations. A leaf node n_k has minimum centrality. We include endpoints in paths computation so there are 2(N-1) paths in which n_k is present, and its centrality is $\frac{2}{N}$. If instead n_k is the center of a star topology, it has maximum centrality since it is present in all the N(N-1) possible shortest paths, thus $b_k \in [\frac{2}{N}, 1]$. Computing centrality on wireless routers has been shown to be feasible for N as large as 1000 [3].

Another concept that we use in the paper is the "cut-point", a node n_i that, if removed, partitions the graphs in two or more components. A cut point has non-minimal betweenness by definition, because it is the connection between at least two disjoint components.

B. Link-State Routing and OLSR

A link-state routing protocol is a protocol in which all the nodes have the knowledge of the full network graph and can populate their routing tables as the result of applying Dijkstra's algorithm to the network graph. OLSR is one of the most studied link-state routing protocols for mesh networks, it powers hundreds of existing networks and recently it was fully re-engineered with a new protocol definition, see [4] and all RFCs it updates. One of the most notable use of OLSR is in Wireless Community Networks: bottom-up initiatives that use WiFi-based networks to improve Internet accessibility[2]

In extreme synthesis OLSR has two functions concurrently running:

- Every node n_i sends one H message every $t_{\texttt{H}}(i)$ seconds. Hs use 1-hop broadcast to discover and maintain neighborhood relationships. Each H message contains a *validity* field whose value is $t_{\texttt{H}}(i) \cdot \texttt{H}_{\text{mult}}$, where $\texttt{H}_{\text{mult}}$ is a configurable multiplier. A neighbor n_j of n_i sets a timer to the validity time at the reception of any H from n_i, if a new H is not received before the timer expiration, n_j considers link $\{n_i, n_j\}$ broken. Hs are also used to estimate the link loss and assign a link quality using the ETX metric [7]. ETX is an additive metric whose minimum value (best quality) is 1.
- Every node n_i also sends Topology Control messages (TCs from now on) every $t_{\texttt{TC}}(i)$ seconds (generally with $t_{\texttt{TC}}(i) > t_{\texttt{H}}(i)$). A TC generated by n_i contains the valid links $\{n_i, n_j\}$ for every neighbor n_j. TC messages are flooded and reach every node in the network so every node n_k is aware of the full (weighted) topology and can compute the shortest path to any destination and build its routing table. As for H messages, TC messages have a validity computed as $t_{\texttt{TC}}(i) \cdot \texttt{TC}_{\text{mult}}$, where $\texttt{TC}_{\text{mult}}$ is configurable.

As any link-state routing protocol, OLSR generates a large amount of control messages, and several techniques have been proposed to reduce it. Among them it is worth to mention Fisheye [8] which has the drawback of introducing temporary loops [9], and Multi-Point-Relays (MPRs) [10] which are natively supported by OLSR [11].

C. Pop-Routing

Pop-Routing is a recently introduced technique that exploits the presence of the validity field in the Hs and TCs to differentiate the timers [1]. Pop-Routing generalizes the ideas that lead to both Fisheye and MPR without the drawbacks of these solutions. With Pop-Routing $t_{\texttt{H}}(i)$ and $t_{\texttt{TC}}(i)$ are not the same for every node, but are dynamically computed by a function that depends on the node centrality. The rationale of Pop-Routing is that the failure of a node n_i with high betweenness triggers the re-computation of a large number of shortest paths, and thus possibly impacts a high amount of traffic: It is convenient to decrease $t_{\texttt{H}}(i)$ and $t_{\texttt{TC}}(i)$ in order to quickly detect the failure of n_i and quickly propagate this information to the rest of the network nodes. Instead, the failure of a node n_j with low centrality has a low impact and we can use large values for $t_{\texttt{H}}(j)$ and $t_{\texttt{TC}}(j)$. The equalization of the timers proposed by Pop-Routing maintains the total amount of control messages equal to the case in which the timers are set to their default value, $t_{\texttt{H}}(i) = 2\,\text{s}$ and $t_{\texttt{TC}}(i) = 5\,\text{s}$, but reduces the average time for which a shortest path remains broken due to a node failure.

The exact computation of the timers is given in Eqs. (2) and (3) where d_i is the degree of n_i and $O_{\texttt{H}}$ is a constant (see the original paper for the details [1]).

$$t_{\texttt{H}}(i) = \frac{\sqrt{d_i}}{\sqrt{b_i}} \frac{1}{O_{\texttt{H}}} \sum_{j=1}^{N} \sqrt{b_j d_j} \tag{2}$$

$$t_{\texttt{TC}}(i) = \frac{\sqrt{E}}{\sqrt{b_i}} \frac{1}{O_{\texttt{H}}} \sum_{j=1}^{N} \sqrt{b_j E} \tag{3}$$

D. Prince

Prince[3] is an open source software that implements Pop-Routing on top of the two versions of OLSR and it is a separated daemon from OLSRd (the daemon that implements the OLSR protocol). This design choice was necessary because the computation of the centrality value may take several seconds in large networks on constrained device [3], and OLSRd can not freeze for such an amount of time. Furthermore, using a separate daemon makes it possible to support any other routing

[2]See for instance [5], [6], and sites as http://netcommons.eu, http://guifi.net, http://https://freifunk.net/en/

[3]See https://github.com/AdvancedNetworkingSystems/poprouting

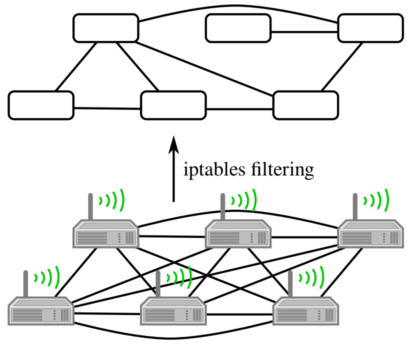


Figure 1: Custom topology through MAC layer filtering.

protocol with a plug-in system. OLSRd offers a set of interfaces that can be used by external applications to extract the network topology and set the configuration parameters. Prince exploits these interfaces to periodically receive the topology, calculate the correct values for $t_{\mathtt{H}}(i)$ and $t_{\mathtt{TC}}(i)$ according to Eqs. (2) and (3) and feed them back to the running instance of OLSRd.

III. Experiments Setup

We carry out our experiments using the WiSHFUL federate w.iLab1[4] test-bed. The test-bed is composed of 44 nodes disposed in a 4×11 grid, each row being spaced by $2\,\mathrm{m}$ and each column by $2.5\,\mathrm{m}$. Each node is equipped with an IEEE 802.11abgn NIC card for wireless experimentation and with an Ethernet interface for remote control.

Due to nodes' proximity, the topology of the test-bed is almost fully-meshed, with the exception of a few links. We exploit this characteristic to reproduce arbitrary topologies on top of the fully meshed network by means of MAC filtering through `iptables` (Fig. 1).

We developed a set of scripts which automatically take care of setting up the nodes, starting the required software, applying filtering rules to obtain the required topology, waiting for network convergence, and killing a node depending on a criterion at a certain point in time. Each experiment is repeated for standard OLSR and for Pop-Routing (i.e., with Prince) and can be performed multiple times to gain statistical confidence.

During the experiment, we periodically sample the routing table and the OLSR topology of all nodes, which makes it possible in post-processing to reconstruct the state of the network at any time. Nodes perform data sampling in a synchronous manner thanks to Network Time Protocol (NTP), giving us a synchronization precision in the order of $20\,\mathrm{ms}$, which is more than enough for our sampling interval ($300\,\mathrm{ms}$).

In the post-processing phase, we collect all the timestamped routing tables from all the nodes and we navigate them for each source and each destination node in all sampling points, computing the number of broken and looped paths (non-functioning paths from now on)[5]. Broken paths are those where the routing table of one of the nodes along the route either points to the failed node or does not have an entry for the destination. In a running network they generate the ICMP *"Destination unreachable"* message. Paths with loops, instead, are the ones where one node appears twice in the route, causing packets to bounce back and forth until the Time to Live (TTL) expires. In a running network they generate the ICMP *"Time exceeded"* message.

[4]http://doc.ilabt.iminds.be/ilabt-documentation/wilabfacility.html

[5]The post-processing phase is quite complex, and it is only summarised here for brevity. It is the same procedure described in details in [1].

We compare the performance of OLSR against Pop-Routing by analyzing how many routes were non-functioning for how much time. More formally, let e be the total number of samples (timestamped routing tables for each node), r_h and T_h the number of non-functioning paths and the time-stamp of sample h, respectively. We define the *"combined empirical loss reduction"* $\tilde{L}$ as

$$\tilde{L} = \sum_{h=1}^{e} r_h \cdot (T_h - T_{h-1}), \tag{4}$$

which can be seen as the integral of the number of non-functioning paths over time. This concept is clarified in detail in Section IV (Fig. 3). The value $\tilde{L}$ is averaged over all repetitions.

To compare OLSR with Pop-Routing, we compute the absolute integral difference $\tilde{L}_{olsr} - \tilde{L}_{pop}$ for each of the failed nodes. In addition, we compute the global absolute and relative loss reduction as

$$\tilde{L}_A = \sum_{i=1}^{N_f} \tilde{L}_{olsr}(i) - \tilde{L}_{pop}(i), \quad \tilde{L}_g = 1 - \frac{\sum_{i=1}^{N_f} \tilde{L}_{pop}(i)}{\sum_{i=1}^{N_f} \tilde{L}_{olsr}(i)}, \tag{5}$$

where $\tilde{L}(i)$ indicates the loss reduction computed when killing the i-th most central node that is neither a cut-point nor a leaf and N_f is the number of experiments.

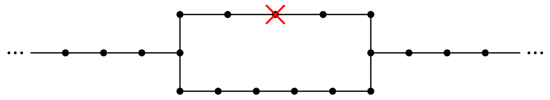
We test our approach in several scenarios, selecting topologies that have different characteristics. The first topology (Fig. 2a) is a simple chain with a bisection in the middle, and we use it to explain the phenomena. The two bisection branches are of length 5 and 6, respectively. Since the ETX values of the links are all close to 1, the shortest branch is always preferred to the longest one. A second test topology is a Barabási-Albert [12] with $m = 3$. In both cases we run experiments with 42 nodes to fine tune the configuration parameters of OLSRd. From this testing phase we observed that the default multiplier values used by OLSRd ($\mathtt{TC}_{\text{mult}} = 60$ and $\mathtt{H}_{\text{mult}} = 10$) are definitely more stable than the value of 3 suggested in the RFC, albeit this imply that after a failure the performance loss can be very large. In addition, we disable Fisheye as it often produces inconsistent routes coupled with small values of $\mathtt{H}_{\text{mult}}$ and $\mathtt{TC}_{\text{mult}}$.

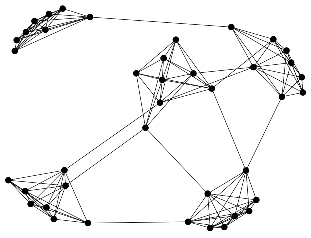
Next, we considered two well-known topology types with 40 nodes: caveman (Fig. 2b) and Waxman (Fig. 2c). The former is characterized by multiple, well-connected cliques, inter-connected by rewiring a few links [13] and is generally used to model human communities, and thus, ad-hoc networks. The latter is often used to model communication networks [14].

Depending on the topology, we use different strategies for selecting the nodes to kill. For the linear topology, we kill the

	$\mathtt{H}_{\text{mult}}$	$\mathtt{TC}_{\text{mult}}$	Killed node
Line	3, 10	60	Center node in shortest branch
Barabási-Albert	10	60	Most central
caveman	10	60	Five most and five least central
Waxman	10	60	Five most and five least central

Table I: Experiment parameters.


(a) line graph


(b) caveman

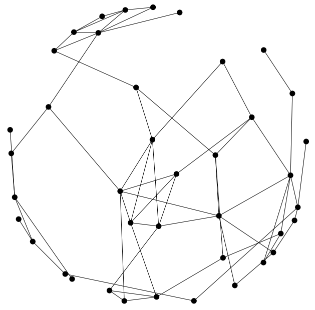
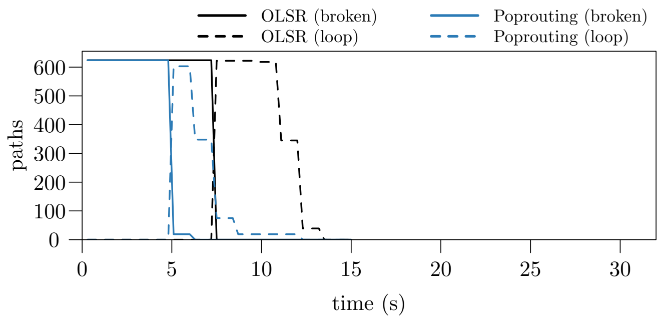
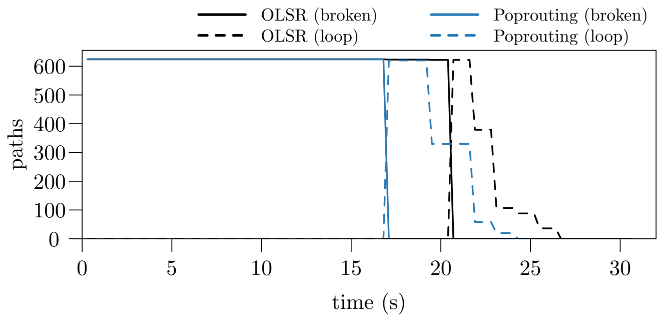

(c) Waxman

Figure 2: Topologies considered in the experiments.



(a) $\mathtt{H}_{\text{mult}} = 3$, $\mathtt{TC}_{\text{mult}} = 60$



(b) $\mathtt{H}_{\text{mult}} = 10$, $\mathtt{TC}_{\text{mult}} = 60$

Figure 3: Number of broken and loop paths as function of time for OLSR and Pop-Routing, for different values of the hello validity multiplier.

node in the middle of the shortest branch. In the Barabási-Albert topology, we kill the node with the highest betweenness centrality b_i. In the caveman and the Waxman topology, we kill the five nodes with the highest and the lowest betweenness centrality b_i. Table I summarizes the experiment parameters.

IV. Results Analysis

We begin the analysis by qualitatively comparing Pop-Routing with standard OLSR in the line topology in terms of time needed to fix non-functioning paths. Figure 3 shows the evolution of the number of non-functioning paths starting from the moment we kill one node. For both OLSR and Pop-Routing, killing a node breaks all the shortest paths passing through that node. As the topology is the same, the number of broken routes is equal. After approximately $\mathtt{H}_{\text{mult}} \cdot t_{\mathtt{H}}(i)$ seconds the neighbors of the killed node update the routing table by removing the killed node as a possible next hop. This causes the number of broken routes to decrease quickly but, on the other hand, creates some routing loops. These loops are removed when TC messages are sent and the information about the topology change is propagated.

As the killed node has a high betweenness centrality, the corresponding $t_{\mathtt{H}}(i)$ (and thus the validity $t_{\mathtt{H}}(i) \cdot \mathtt{H}_{\text{mult}}$) is lower than with OLSRd. As expected, the time taken to repair the routes is shorter when Pop-Routing is enabled.

In addition, we see that the time required to detect the death of the nodes is related to $\mathtt{H}_{\text{mult}}$. The lower $\mathtt{H}_{\text{mult}}$, the faster the detection. However, given that our logical topology is implemented on top of a physical full-mesh, the interference level might easily lead to consecutive packet losses. We observed that setting a low $\mathtt{H}_{\text{mult}}$ can pollute our results by mistakenly invalidating routes during the experiment. For this reason in the remaining experiments we keep the default OLSRd $\mathtt{H}_{\text{mult}} = 10$.

Figure 4 reports an interesting behavior observable when multiple paths from a source to a destination exist, like in a Barabási-Albert model. The figure shows that the network starts to re-converge before the expiration of the H validity time. This happens because OLSRd implements the ETX metric, which causes link quality to decrease for each missed H message. Even before the link is considered broken its quality degrades, and this information is propagated with TCs. Nodes progressively decide to switch to other shortest paths, and the dead node becomes less central. Pop-Routing still performs better than OLSRd, but the gain is smaller. This phenomenon is not observable in Fig. 3, as in this specific scenario the cost of the additional hop makes the shorter path preferable until the node failure is discovered.

We continue the analysis considering the caveman and the Waxman topology types. Figure 5 shows $t_{\mathtt{H}}(i)$, $t_{\mathtt{TC}}(i)$, and betweenness b_i for a single run using the Waxman topology. The nodes are ordered by decreasing values of b_i. Points drawn in light red are cut-points and leaves, i.e., the ones that can not be killed without partitioning the network. In addition, the plot shows the default $t_{\mathtt{H}}$ and $t_{\mathtt{TC}}$ values for comparison. The plot shows that most central nodes are assigned lower $t_{\mathtt{H}}(i)$ and $t_{\mathtt{TC}}(i)$ with respect to the standard values $t_{\mathtt{H}}$ and $t_{\mathtt{TC}}$. On the

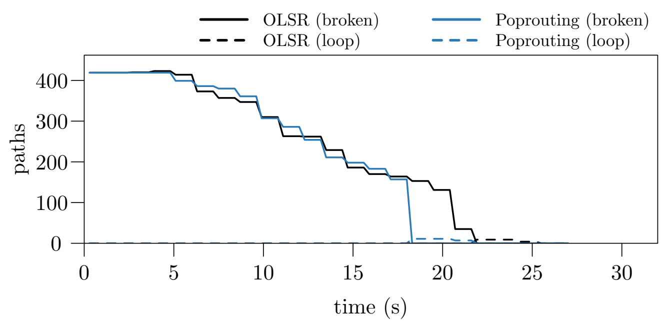


Figure 4: Stairs-like behavior due to link quality degradation in a Barabási-Albert graph ($\mathtt{H}_{\text{mult}} = 10$).

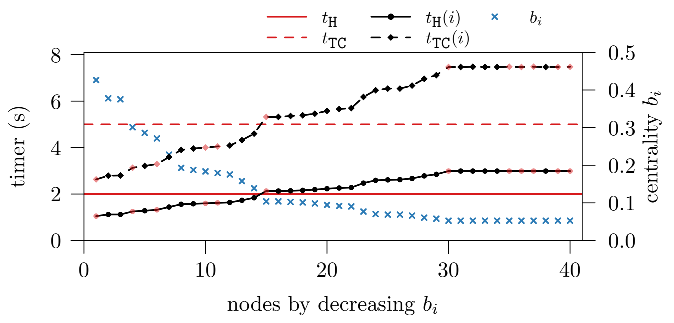


Figure 5: $t_{\mathtt{H}}(i)$, $t_{\mathtt{TC}}(i)$, and b_i for one run using the Waxman topology. Light red dots identify nodes that failing would partition the network.

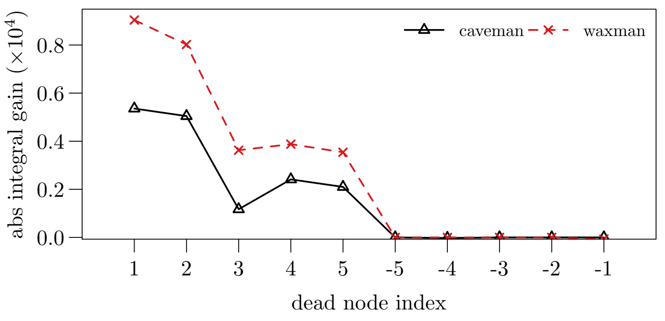


Figure 6: Absolute integral gain as function of the killed node index, for two different topology and two different interpretations of the degree. Nodes are ordered by betweenness centrality. Negative indexes indicates the least central nodes.

	Waxman	caveman
Absolute	2803.31	1606.06
Relative	0.26	0.21
Relative (theory)	0.18	0.20
Relative (theory, all nodes)	0.10	0.04

Table II: Absolute and relative gains for Waxman and caveman topology.

other hand, least central nodes are penalized and are assigned higher timer values.

Figure 6 shows the absolute integral difference $\tilde{L}_{olsr}(i) - \tilde{L}_{pop}(i)$ for all killed nodes. The node index does not consider cut-points and leaves. Pop-Routing shows a positive performance gain with respect to OLSRd when killing the five most central nodes. When killing the five least central nodes (negative node index), in most cases the performance loss is null, as there is no shortest path passing through the failing node. In the remaining cases, the loss is negligible compared to the gain obtained by increasing the $\mathtt{H}$ and $\mathtt{TC}$ message frequency for the most central nodes.

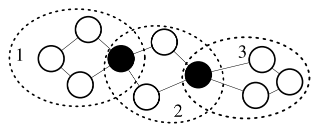
We would expect the absolute performance gain to be monotonically decreasing with the node index, as the timers in Fig. 5 are monotonically decreasing. The values in Fig. 6, however, are heavily affected by noise as the shortest paths might change from one experiment run to the next. Indeed, as highlighted in Fig. 5, the two most central nodes have a very similar betweenness centrality value, so slight fluctuations in signal quality can lead to route changing and affect the results.

To obtain a complete comparison we would need to repeat the experiments killing each possible node; however, this requires an enormous amount of time to run the experiments. As an alternative, we complement experiments with the theoretical gain that Pop-Routing can achieve on a certain network (the relevant computations and formulas are reported in [1]). Table II reports the theoretic gain $\tilde{L}_g$ computed considering all nodes. The gain is smaller than when considering the five most and least central nodes as well, but it is in any case positive, showing a performance improvement by $10\,\%$ and $4\,\%$ for the Waxman and the caveman topology types, respectively, with no additional protocol overhead. It is interesting that the experimental gain is larger than the theoretical when we consider the same modes. This is probably due to noise in experiments, as already mentioned, but its interpretation requires additional investigation.

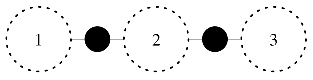
A. *Dealing With Cut-points*

Cut-points are nodes that cannot be removed without partitioning the network. A bi-connected component instead is a sub-graph of the network for which the removal of one node will not partition the network. Any graph can be decomposed in a so-called block-cut tree made of biconnected components and cut-points, as shown in Fig. 7. This representation shows that typically cut-points have high betweenness, and thus we expect their timers to be lower than the timers of other nodes.

On the other hand, if a cut-point fails the network is physically partitioned, and the routing protocol can not fix it. That is the reason why in our experiments we never kill a cut-point, because some routes would never converge, and the network before and after the failure would not be comparable. In general for node n_i, knowing that a piece of the network is not reachable anymore is not particularly useful: if n_i is exchanging traffic with n_j and due to a failure there exist no path anymore from n_i to n_j applications stop working, and routing cannot do anything about it. If we look at Pop-Routing as an optimization strategy, given a certain amount of a resource (the airtime devoted to control messages) Pop-Routing distributes it among nodes (tuning their timers) to reduce the



(a)



(b)

Figure 7: An example decomposition in a block-cut tree.

	Waxman	caveman
Relative (theory, all nodes)	0.14	0.10

Table III: Absolute and relative gains for Waxman and caveman topology with modified centrality (Eq. (6)).

outage due to failures. The problem is that Pop-Routing assigns resources to cut-points, but their failure can not be fixed.

A straightforward conclusion would be to artificially treat cut-points as nodes with minimum centrality and distribute their resources to other nodes. This is not entirely correct, because part of the centrality of a cut-point is due to the shortest paths that connect nodes in the same block. If we do not consider this we ignore the fact that, when the cut-point fails, some of the shortest paths that pass though it can actually be fixed. The correct decision would be to compute centrality for cut-points considering only the shortest paths whose endpoints are in the same block, and ignore the paths that have both endpoints in different blocks. More formally, if n_k is a cut-point belonging to a number l of bi-connected components $\{B_1, \ldots, B_l\}$, N_h the size of each bi-connected component, and b_k^h the betweenness centrality of n_k in B_h computed using Eq. (1), then its modified betweenness b'_k is defined as:

$$b'_k = \frac{1}{N(N-1)}(\sum_{h=1}^{l} b_k^h(N_h(N_h-1))+2\cdot(N-\sum_{h=1}^{l} N_h)) \quad (6)$$

where the multiplication by $(N_h(N_h-1))$ is needed to remove the normalization applied by Eq. (1) and the term $2 \cdot (N - \sum_h N_h)$ takes into account the shortest paths that have as endpoints n_k and any another node $n_i \notin B_h$ when n_k is considered in the component B_h. This is a further optimization we will introduce in Prince in the future. We can anyhow compute the theoretical expected gain, which is reported in Table III. By comparing these values with the ones in Table II a clear further advantage emerges.

One exception to this rule that requires a special treatment would be the case of gateways. When the block containing a gateway is not reachable anymore, then it is crucial to know it as soon as possible, in order to switch to another working gateway (if it exists). This special case will be treated directly in the implementation in Prince.

V. Conclusions

Pop-Routing introduces a new concept in wireless routing: it allows networks to grow organically while the protocol seamlessly adapt the allocations of signaling airtime resources according to node centrality optimizing the networks recovery performance after a failure. Since betweenness centrality is a naturally associated to routing, the idea behind Pop-Routing can be used for several other similar optimizations. Before extending the application field of Pop-Routing, however, it is necessary to thoroughly validate its principles implementing it in controlled environments. This paper does exactly this: leveraging the infrastructure made available by the WiSHFUL project to test the performance of Pop-Routing, it gives insights on the way it contributes to speed up route recovery.

Our findings confirm that Pop-Routing is indeed able to improve route recovery. We observed that, *without increasing the overall traffic*, Pop-Routing reduces the metric we use to quantify network outage by 26 % and 21 % in two kinds of realistic network topology. We also observed that the presence of the ETX metric, in a topology where there are several alternatives to the shortest path, decreases the overall outage, while it has no effect when the number of alternatives is minimal. Finally, based on the experience we gathered on the test-bed, we proposed a further improvement to Pop-Routing which takes into account the role of cut-points in the topology. While in this work we were able to only estimate its impact in the theoretical framework of Pop-Routing, in our future works we will investigate the impact of this strategy in a real network.

References

[1] L. Maccari and R. Lo Cigno, "Pop-Routing: Centrality-Based Tuning of Control Messages for Faster Route Convergence," in *IEEE Int. Conf. on Computer Communications (INFOCOM)*, Apr. 2016, pp. 1–9.

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

[3] L. Maccari, Q. Nguyen, and R. Lo Cigno, "On the Computation of Centrality Metrics for Network Security in Mesh Networks," in *IEEE Global Communications Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[4] C. Dearlove and T. Clausen, "Multi-Topology Extension for the Optimized Link State Routing Protocol Version 2 (OLSRv2)," IETF RFC 7722, Dec. 2015.

[5] L. Navarro, R. Baig, F. Freitag, E. Dimogerontakis, F. Treguer, M. Dulong de Rosnay, L. Maccari, P. Micholia, and P. Antoniadis, "Report on the Existing CNs and their Organization (v2) – netCommons Deliverable D1.2," Sept. 2016. [Online]. Available: http://netcommons.eu/?q=content/report-existing-cns-and-their-organization-v2

[6] L. Maccari and R. Lo Cigno, "A week in the life of three large Wireless Community Networks," *Ad Hoc Networks*, vol. 24, Part B, pp. 175 – 190, 2015.

[7] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, 2005.

[8] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, "Scalable routing strategies for ad hoc wireless networks," *IEEE Jou. on Selected Areas in Communications (JSAC)*, vol. 17, no. 8, pp. 1369–1379, Aug. 1999.

[9] Y. Faheem and J. L. Rougier, "Loop avoidance for Fish-Eye OLSR in sparse wireless mesh networks," in *IEEE Int. Conf. on Wireless On-Demand Network Systems and Services (WONS)*, Feb. 2009, pp. 231–234.

[10] O. Liang, Y. A. Sekercioglu, and N. Mani, "A survey of multipoint relay based broadcast schemes in wireless ad hoc networks." *IEEE Communications Surveys & Tutorials*, vol. 8, no. 1-4, pp. 30–46, 2006.

[11] L. Maccari and R. Lo Cigno, "How to reduce and stabilize MPR sets in OLSR networks," in *IEEE 8th Int. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2012, pp. 373–380.

[12] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[13] D. J. Watts, "Networks, Dynamics, and the Small-world Phenomenon," *American Jou. of Sociology*, vol. 105, no. 2, pp. 493–527, 1999.

[14] B. M. Waxman, "Routing of Multipoint Connections," *IEEE Jou. on Selected Areas in Communications (JSAC)*, vol. 6, no. 9, pp. 1617–1622, Dec, 1988.