Poster: Robust Wi-Fi Mesh Networking with SPIDERMAN

Sonali Deo[§], Markus Schuß[§], Michael Baddeley[*], and Carlo Alberto Boano[§]
[§]Institute of Technical Informatics, Graz University of Technology, Austria
[*]Technology Innovation Institute, United Arab Emirates
{sdeo, markus.schuss, cboano}@tugraz.at ; michael.baddeley@tii.ae

Abstract

Emergency response solutions are often based on ad-hoc mesh networks extending connectivity in isolated zones, and need to sustain high degrees of mobility while being resilient to sources of RF interference. As existing Wi-Fi mesh solutions do not meet these requirements, we propose SPIDERMAN, a protocol that utilizes flooding to enhance reliability in dynamic settings, frequency agility to escape RF interference, and time-slotted operations to ensure collision-free traffic flows through the network. Our preliminary implementation shows that SPIDERMAN allows uninterrupted connectivity even in highly-mobile settings and under jamming attacks.

1 Motivation

Robust Wi-Fi mesh solutions are crucial in the context of emergency response scenarios to establish network infrastructure (e.g., during natural disasters). Such solutions do not have as stringent energy concerns as traditional low-power wireless mesh, and need to quickly establish connectivity among first responders. To this end, they should enable reliable transmission of video/audio streams and be robust to highly-mobile conditions [9]. Moreover, due to congestion of the RF spectrum, such networks should also be resilient to (un)intentional interference [10].

Insufficiency of existing solutions. Several Wi-Fi mesh solutions running on commodity 802.11 hardware (HW) exist. IEEE 802.11s introduces the HWMP routing protocol [3] and can support other protocols such as OLSR [7] and BATMAN-adv [8]. However, these mainly use pre-computed routes or compute them on-demand, which results in a rigid topology. In mobile settings they therefore perform poorly as they do not leverage redundant routes. Moreover, these protocols usually operate on a single frequency, which makes them susceptible to jamming attacks and nearby RF interference.

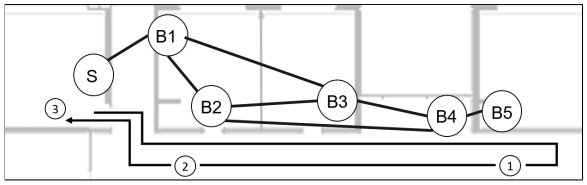

Figure 1. Five static nodes (`B1`–`B5`) and a mobile source (`S`) which moves according to the arrow. The three circled numbers mark specific instants in Figs. 2 and 4.

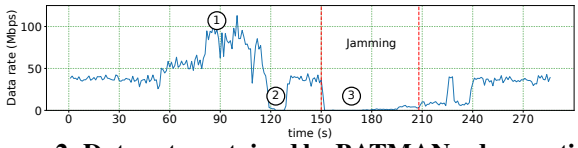

Figure 2. Data rate sustained by BATMAN-adv over time.

Case study: BATMAN-adv. We exemplify this problem by evaluating the performance of BATMAN-adv, a proactive layer-2 routing protocol that uses the ad-hoc mode of Wi-Fi cards to route packets between nodes without going via an access point. We use the testbed setup shown in Fig. 1, in which node `S` transmits packets to `B5` using Wi-Fi ch. 36 (40 MHz bandwidth) while moving along a hallway[1]. Fig. 2 shows the data rate sustained at `B5`: BATMAN-adv performs well as long as `S` moves towards the destination, as it favors routes with fewer hops ①. Once `S` moves away from `B5`, it sticks to the direct route to `B5` until connectivity is lost ②, which results in a disconnection for 10 s. Once `S` returns to its original position, we artificially introduce jamming using a Raspberry Pi 4 running JamLab-NG [10] in the same channel: as BATMAN-adv does not support frequency agility, connectivity is lost until a configuration with a lower modulation coding scheme (allowing `S` to reach `B5` directly) is found.

The gap to fill. These limitations call for alternative mesh networking solutions suitable for highly-mobile settings and harsh RF environments. Although Wi-Fi protocols embedding channel hopping [5, 11] and multi-hop routing [4] schemes exist, there is still no solution embedding both features at once that made it to mainstream applications.

Our contribution. We hence present SPIDERMAN, a time-slotted channel hopping protocol inspired by IEEE 802.15.4e TSCH [6] that uses flooding to increase reliability by leveraging multiple routes. In § 2, we detail the idea behind SPIDERMAN and discuss a preliminary implementation

[1]Detailed video available at: `https://youtu.be/jZpTEfdfH9c`

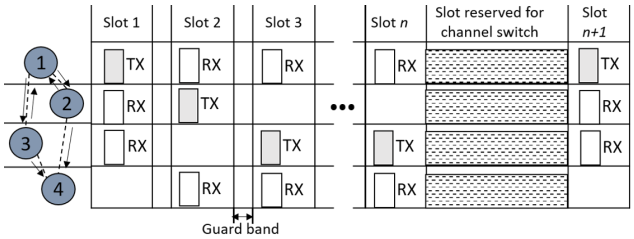

Figure 3. Overview of SPIDERMAN's concept.

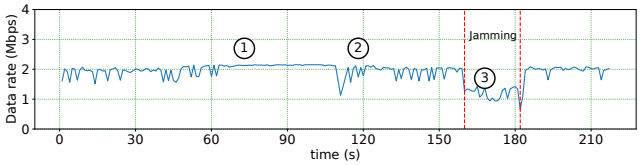

Figure 4. Data rate sustained by SPIDERMAN over time.

running on the popular `ath9k` card. In § 3, we present a first evaluation of SPIDERMAN's performance, showing its reliability despite mobile settings and resilience to jamming attacks. In § 4, we conclude with an outlook on future work.

2 SPIDERMAN: Design & Implementation

Principle. Fig. 3 illustrates an example where the nodes transmit *only* in their corresponding slot. In the first slot, node 1 broadcasts to all neighbors, reaching 2 and 3, which consequently broadcast in their own slots. This flooding approach not only prevents collisions on-air, but also provides redundancy in case one link is lost (e.g., the packet can reach node 4 via two different links). Since the nodes utilize all their immediate links, no route calculation is required.

SPIDERMAN is implemented on top of monitor mode, where switching channels is possible without announcing it to the network beforehand. However, initiating a channel switch from userspace takes time (≈10ms) during which the Wi-Fi chip is reset. Therefore, SPIDERMAN switches channel only every n slots. Since all nodes are calculating the current absolute slot of the network, they hop channels synchronously. This way, in case of (un)intentional interference, the nodes can still communicate with each other on a different channel without needing to disseminate channel hop information throughout the network beforehand.

Implementation. We implement SPIDERMAN on Raspberry Pi 4 devices and an `ath9k` Wi-Fi card. The latter supports monitor mode, which allows the transmission of packets with a desired modulation and data rate from userspace using Python applications like Scapy [2], and has a SoftMAC that provides control over physical layer attributes. As Wi-Fi's clear channel assessment (CCA) randomly backs off in presence of traffic, we disable this feature [1] to avoid a variable delay that can interfere with the timely transmission of packets in a given slot. We dimension the timing of SPIDERMAN by accounting for the worst-case userspace duration from packet transmission to correct reception, the jitter of the processor, and the time to switch channels, which results in 1 ms slots for transmission and 15 ms slots for channel switching. SPIDERMAN's hopping schedule employs three 40 MHz channels in the 5 GHz band (7, 36, 44). These are iterated every n slots, with $n = 150$ in our implementation.

3 Preliminary Evaluation

We reuse the setup shown in Fig. 1 to evaluate the performance of SPIDERMAN: Fig. 4 shows the data rate at `B5`[2].

Robustness to mobility. As `S` starts moving towards `B5`, data is forwarded via `B1`–`B4` and slight fluctuations in the data rate are seen due to changes in the number of hops. At ① `B5` has a direct link with `S`, resulting in a constant data rate. As `S` returns to the starting point, the hops to reach `B5` again increase, resulting in the same fluctuations observed at the beginning. However, *connectivity is always preserved during this mobility*. This is in contrast to BATMAN-adv (Fig. 2), which sees significant drops in throughput and 'blackout' periods due to its preference for routes with the fewest number of hops (i.e., BATMAN-adv does not switch route until no packets have been received within a given period).

Resilience to jamming. As in § 1, we artificially introduce jamming using JamLab-NG [10]. Please note that we use a modified version that creates interference in the 5 GHz spectrum by jamming the upper half of a 40 MHz channel: this ensures that the jamming signal is not detected as valid packet. At ③ JamLab-NG interferes with `B4` on channel 36, which causes a 30-50% drop in throughput (Fig. 4). However, as SPIDERMAN employs channel hopping, `B4` still receives the packets sent by `S` in the other two channels (i.e., 7 and 44) and can forward them. BATMAN-adv, instead, uses a single channel and suffers a complete loss of connection (Fig. 2).

4 Conclusions and Future Work

While this initial implementation of SPIDERMAN exhibits low data rates due to large slot sizes, previous works have shown these can be eliminated with beacon interrupts for precise timing directly on HW [11]. Moreover, SPIDERMAN's current implementation is made in Python rather than a low-level programming language, which adds processing overhead. Addressing these points in future work will allow us to reduce slot sizes and increase SPIDERMAN's throughput.

Acknowledgments. This research work is supported by the SPiDR project funded by the Technology Innovation Institute.

5 References

[1] Atheros patches. [Online] `https://tinyurl.com/nhbnu9hw`.
[2] Scapy. [Online] `https://scapy.net/` – Last access: 2023-06-23.
[3] J. Avinash et al. IEEE P802.11 Wireless LANs – HWMP Specification, 2006. doc.: IEEE 802.11-06/1778r1.
[4] Y. Cheng et al. Det-WiFi: A Multihop TDMA MAC Implementation for Industrial Deterministic Applications based on Commodity 802.11 Hardware. *Wireless Communications and Mobile Computing*, 2017.
[5] S. Djuraev et al. Channel Hopping Scheme to Mitigate Jamming Attacks in Wireless LANs. *EURASIP J Wirel Commun*, 2017.
[6] IEEE. 802.15.4e-2012 Standard for Local and metropolitan area networks, Part 15.4: Low-Rate Wireless Personal Area Networks, 2012.
[7] P. Jacquet et al. Optimized Link State Routing Protocol for Ad Hoc Networks. In *Proc. of the IEEE INMIC Conf.*, 2001.
[8] Open-Mesh. BATMAN. [Online] `https://tinyurl.com/59s43ckt`.
[9] D. Reina et al. A Survey on Multihop Ad Hoc Networks for Disaster Response Scenarios. *IJDSN*, 11(10), 2015.
[10] M. Schuß et al. JamLab-NG: Benchmarking Low-Power Wireless Protocols under Controllable and Repeatable Wi-Fi Interference. In *Proc. of the 16th EWSN Conf.*, 2019.
[11] A. Sharma et al. FreeMAC: Framework for Multi-Channel MAC Development on 802.11 HW. In *Proc. of the 1st PRESTO Worksh.*, 2008.

[2]Detailed video available at: `https://youtu.be/jbHPVB1aUek`.