MAC-Filter based Topology Control for WLAN Mesh Networks

Tim Brockmann, Michael Rethfeldt, Benjamin Beichler, Frank Golatowski, Christian Haubelt
Institute of Applied Microelectronics and Computer Engineering, *University of Rostock*, Germany
Email: tim.brockmann@uni-rostock.de

***Abstract*—WLAN Mesh Networks (WMNs) are a promising network architecture for IoT applications due to their self-healing and robust network formation capabilities. The IEEE 802.11s amendment standard integrates mesh functionality into the WLAN MAC layer, ensuring seamless peering and routing. Initially designed for testing, the Linux implementation of IEEE 802.11s includes basic peer link blocking mechanisms. Based on this, we present a new MAC address filtering approach for reliable topology control in WLAN mesh networks. Our modifications to the Linux kernel enable persistent link blocking and bidirectional link teardown, preventing the sporadic establishment of unwanted connections. Real-world experiments demonstrate the effectiveness of our approach.**

***Index Terms*—WLAN mesh network, IEEE 802.11s, peering, MAC filter, topology control, Linux**

I. Introduction

WLAN Mesh Networks (WMNs) come as a promising network architecture for the (Industrial) Internet of Things due to their spontaneous network formation and self-healing capabilities, which eliminate the risk of a single point of failure and increase overall network robustness [1]. The WLAN amendment standard IEEE 802.11s defines mesh functionality for the IEEE 802.11 MAC layer, including peering and routing mechanisms, to ensure low-level mesh interoperability [2]. A reference implementation of IEEE 802.11s is available as part of the Linux kernel.

Originally introduced mainly for testing purposes, the Linux implementation of IEEE 802.11s also provides basic mechanisms to selectively block existing links between mesh nodes. However, this link blocking functionality is still subject to limitations and cannot yet guarantee reliable topology control. On the other hand, there are a multitude of applications that could benefit from the ability to individually restrict mesh peering in a reliable way. This includes creating reproducible test environments, experimenting with multi-hop topologies in confined spaces [3], and preventing the establishment of sporadic links in wireless backbones and community networks, where link conditions can be highly dynamic and unstable. Excluding such connections may help in avoiding oscillations in mesh paths. It could even be beneficial for addressing security concerns in critical infrastructures. For instance, after anomaly detection, infiltrated or malicious nodes can be isolated by rejecting link establishment until appropriate measures are taken, whether temporarily or permanently [4].

A wireless mesh testbed is currently being set up at the University of Rostock to evaluate time synchronization and multipath concepts, among other things. This requires a more reliable and sustainable means of topology control.

In this paper, we showcase a more sophisticated MAC address filtering approach for peer link blocking and topology control in IEEE 802.11s WLAN mesh networks. We present a proof-of-concept extension of the IEEE 802.11s Linux implementation and demonstrate its practicality through simple peering experiments.

II. Technical Background and Related Works

A. WLAN Mesh Networks and IEEE 802.11s

WLAN Mesh Networks (WMNs) are a type of network where each node connects dynamically and non-hierarchically to other nodes within radio range [1]. This decentralized structure enhances reliability, allowing the network to self-heal by reconfiguring mesh paths around failing nodes, thus eliminating single points of failure. WMNs are particularly well-suited for environments where traditional wired infrastructure is impracticable or too costly to deploy. IEEE 802.11s, an amendment to the IEEE 802.11 WLAN standard [2], defines protocols for peering and routing between WLAN stations and thereby enables mesh interoperability already at the MAC layer.

B. IEEE 802.11s Mesh Discovery and Peering

In IEEE 802.11 networks, mesh discovery and peering are crucial processes that ensure robust and dynamic connectivity. The default method for mesh discovery is passive, relying on beacon frames. These beacon frames are periodically broadcast management frames that advertise nodes and help initialize the network. Once potential peers are discovered via beacons, a four-way peering handshake is initiated to establish bidirectional links. For this purpose, three different management frames of sub-type “action frame” are specified in the WLAN standard [2]: *(mesh peering) open*, *(mesh peering) confirm* and *(mesh peering) close*. As shown in Fig. 1 the peering handshake consists of two *open* frames, one for each link direction, both of which are acknowledged by the opposite station with a *confirm* frame. The message type *close* (not shown) allows for controlled and explicit link teardown.

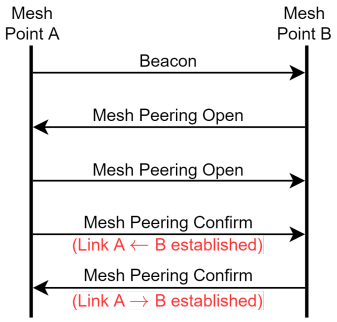


Fig. 1: Four-way mesh peering handshake

C. IEEE 802.11s under Linux

The Linux implementation of IEEE 802.11s is a direct component of the kernel and forms the basis of our work. Under the project name *open80211s*, the company Cozybit initially began developing an open-source reference implementation in 2011[1]. Since then, it has been continuously developed and maintained as an integral part of the WLAN protocol stack by the Linux Wireless team[2].

Fig. 2 shows the simplified architecture of the WLAN protocol stack in the Linux kernel. Most of the functions of the WLAN MAC layer, including the mesh mechanisms, are implemented by the `mac80211`[3] kernel module in software. However, if the functions of the MAC layer are already implemented on the hardware of the WLAN adapter by the manufacturer and are therefore independent of `mac80211`, they are referred to as full-MAC devices. In this case, the mesh operating mode is usually not available. If, on the other hand, the MAC layer is largely or completely implemented in software, it is a soft-MAC device. In return, only time-critical tasks are usually executed on the hardware, such as media access and flow control of the MAC layer as well as the functions of the physical layer. There are various soft-MAC drivers that support 802.11s mesh mode in conjunction with `mac80211`. The driver `ath9k` for Atheros devices was used in our testbed[4].

Above the `mac80211` layer, the kernel module `cfg80211` enables programs in the Linux userspace to access and configure various parameters related to the MAC layer and device status. Communication between the userspace and the kernel is message-based, facilitated by the Netlink socket interface. The `nl80211` API defines the WLAN-specific Netlink messages. For example, the tool `iw` utilizes the Netlink interface, providing users with powerful command-line capabilities to control and monitor their WLAN interfaces.

D. Mesh Peer Link Blocking Approaches

Mesh peer link blocking is a helpful mechanism for positively influencing network topologies. Originally introduced by Cozybit for testing purposes, a simple peer link blocking feature is available in the Linux implementation of IEEE 802.11s.

[1]https://github.com/o11s/open80211s/wiki/HOWTO
[2]https://wireless.wiki.kernel.org/
[3]https://wireless.wiki.kernel.org/en/developers/documentation/mac80211
[4]https://wireless.wiki.kernel.org/en/users/drivers/atheros

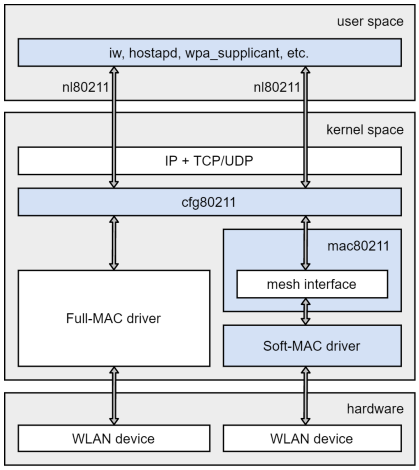


Fig. 2: Architecture of the WLAN implementation under Linux

By means of special *plink_action block* and *plink_action open* commands added to the Netlink API and usable via the tool `iw`, already established links to individual nodes can be blocked or unblocked.

Over time, this feature has found use in productive systems such as Freifunk[5], an open community-driven wireless network initiative. The Freifunk project leverages mesh peer link blocking to maintain network integrity in environments where link conditions are unreliable. The current approach, however, is only able to block already existing links. This limitation necessitates periodic reinvocation of the blocking mechanism in dynamic networks with changing node neighborhoods. This way, sporadic presence of unwanted connections cannot be reliably prevented. The effectiveness of the method depends heavily on the intervals at which beacons are broadcast and link blocking is re-invoked.

In Freifunk networks based on the Gluon firmware[6], periodic link blocking is managed via the package `wifi-mesh-macfilter` in userspace. While it is actively used and provides some level of control by accepting predefined MAC addresses to be blocked, it inherits all weaknesses of the current Linux kernel implementation. Specifically, the periodic nature of the blocking process means that it cannot fully prevent sporadic establishment of unwanted links in-between intervals. A similar blocking approach can be found under the name `mesh-blocker`[7].

This highlights the need for more sophisticated methods to ensure consistent mesh peering control, particularly in environments with highly fluctuating link conditions or even potential security risks and demand for reliable node isolation.

III. MAC-Filter based Topology Control

To ensure persistent blocking of specific neighbor nodes and prevent any sporadic establishment of unwanted links, it is essential for mesh nodes to take into account MAC addresses

[5]https://freifunk.net/
[6]https://github.com/freifunk-gluon/gluon.git
[7]https://github.com/nickbash11/mesh-blocker

to be blocked directly as part of the peering process. To achieve this behavior, we have extended parts of the mesh implementation of the Linux kernel in such a way that the original link blocking functionality is retained and userspace applications (e.g., `iw`) are not affected. Fig. 3 outlines our modifications highlighted in green.

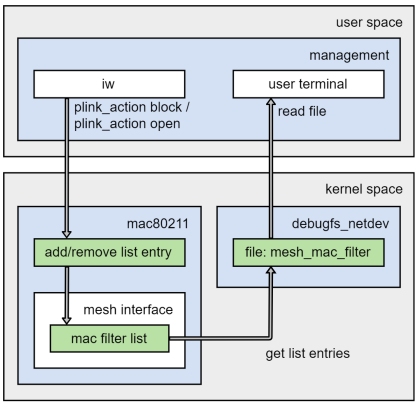


Fig. 3: Mesh MAC filter architecture

The only necessary information needed to block a link is the MAC address of the opposite station. To store these MAC addresses, a new linked list called *mac_filter* is initialized when a virtual mesh interface is created on the physical WLAN adapter. The `mac80211` kernel module has been extended with additional functions for manipulating this list, allowing new elements to be added, removed, and found in the list. Executing the regular `iw` commands *plink_action block* and *plink_action open* triggers the corresponding functions, which in turn fill or clear the list.

When a node receives a peering *open* frame, it first searches the filter list for the corresponding MAC address. If the address is found, the neighboring station is blocked immediately, and a *confirm* frame is never sent. Conversely, if the node receives a beacon frame from another node with matching mesh parameters and thus becomes a potential peering initiator, it also searches its filter list for the MAC address of the opposite node. If the address is on the list, the node will not trigger peering with an *open* frame. Our modifications therefore add blacklist functionality to the current Linux implementation of IEEE 802.11s.

Another modification improves the behavior of the original implementation. Previously, when a peer link was unidirectionally blocked, the link from the perspective of the opposite node was still established. As the link information is updated after a successful peering based on subsequent beacons, the link is retained and therefore seemingly available to userspace applications. As a result, attempts to establish a path to the other node continue, ultimately failing after a prolonged period, without properly tearing down the unusable link. To resolve this, our modification includes sending a *close* frame after successfully blocking the opposite node. This ensures a bidirectional link teardown, effectively removing the unusable link and preventing unnecessary and prolonged connection attempts.

To allow reading the MAC filter list for debugging purposes or as feedback for applications using our extension, the `debugfs_netdev` virtual file system was extended with a `mesh_mac_filter` file and the corresponding file operation functions. To provide feedback on whether a link is only predefined in the list or already actively blocked based on incoming *open* frames, the element *Blocked* was added to the list structure. As shown in Fig. 4 the listed MAC addresses are then marked according to their current status.

```
MAC: 04:f0:21:[illegible], Blocked: Yes
MAC: 04:f0:21:[illegible], Blocked: No
MAC: 04:f0:21:[illegible], Blocked: Yes
MAC: 04:f0:21:[illegible], Blocked: No
MAC: 04:f0:21:[illegible], Blocked: No
MAC: 04:f0:21:[illegible], Blocked: Yes
MAC: 04:f0:21:[illegible], Blocked: No
```

Fig. 4: Exemplary console output of the debugfs file `mesh_mac_filter`

IV. Proof of Concept and Discussion

To evaluate our implementation, we conducted experiments in a real-world testbed. It comprises three single-board computers of type APU2E2 from PC Engines. The stations are equipped with Compex WLE200NX mPCIe WLAN adapters (Atheros AR9280 chipset), supported by the `ath9k` driver. The operating system is a current version of Kali Linux (Debian 64-Bit) using the Linux mainline kernel v6.6.0.+, modified with our extensions.

All stations are positioned in close proximity with two of them (Node 1 and Node 2) acting as a pair of test stations, both configured in mesh mode with the default beacon interval (1000 ms) and sharing the same SSID. The third station is configured in monitor mode and captures the network traffic of Node 1 and Node 2 using the open-source tool `tcpdump`[8], which allows us to analyze the behavior of the test stations under various conditions with the open-source tool `Wireshark`[9].

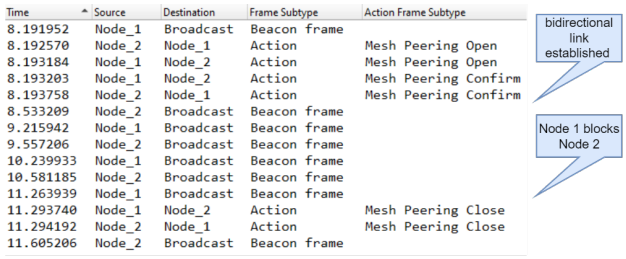
Scenario A: Subsequent unidirectional blocking

In this scenario, peering between the test stations is initially allowed. As can be seen in Fig. 5 the handshaking process takes place and the link is bidirectionally established. The exact sequence of handshaking frames depends on beacon timing and the implementation of the IEEE 802.11s peering protocol. The Linux implementation specifies that an *open* frame sent by a peering initiator should first be responded to with an *open* frame for the opposite direction, before sending the actual *confirm* frame for the incoming *open*. The initiator then replies to the *confirm* frame with a *confirm* frame as well.

After successful peering, a subsequent blocking action is performed on Node 1 to cancel the previously established connection to Node 2. Due to our modification, this results in sending a *close* frame. Node 2 responds with its own *close* frame to complete link teardown. Without our extension, Node 1 would only remove the link on its side and mark it

[8]https://github.com/the-tcpdump-group/tcpdump.git
[9]https://www.wireshark.org/

as blocked; from Node 2's perspective, the link would remain established.



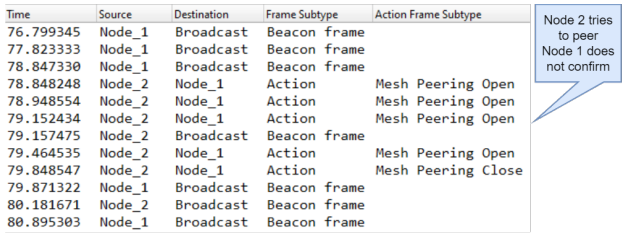
Time	Source	Destination	Frame Subtype	Action Frame Subtype
8.191952	Node_1	Broadcast	Beacon frame	
8.192570	Node_2	Node_1	Action	Mesh Peering Open
8.193184	Node_1	Node_2	Action	Mesh Peering Open
8.193203	Node_1	Node_2	Action	Mesh Peering Confirm
8.193758	Node_2	Node_1	Action	Mesh Peering Confirm
8.533209	Node_2	Broadcast	Beacon frame	
9.215942	Node_1	Broadcast	Beacon frame	
9.557206	Node_2	Broadcast	Beacon frame	
10.239933	Node_1	Broadcast	Beacon frame	
10.581185	Node_2	Broadcast	Beacon frame	
11.263939	Node_1	Broadcast	Beacon frame	
11.293740	Node_1	Node_2	Action	Mesh Peering Close
11.294192	Node_2	Node_1	Action	Mesh Peering Close
11.605206	Node_2	Broadcast	Beacon frame	

Fig. 5: Node 1 blocks Node 2 after successful peering

Scenario B: Initial unidirectional blocking

In this scenario, Node 1 initially blocks Node 2 to prevent establishment of a peer link.

Fig. 6 shows that Node 2 attempts to open a link by sending an *open* frame. Node 1 receives the frame, finds Node 2 in its MAC filter list, and therefore does not respond. At this point, Node 2 is marked as actively blocked in the MAC filter list of Node 1. Consequently, link establishment between Node 1 and Node 2 never occurs. Depending on the mesh parameter *max_retries* (default 3), Node 2 sends additional open frames and ends its attempt to build up a link by sending a *close* frame. In the original Linux implementation, a-priori blocking was not possible, as only existing links could be blocked.



Time	Source	Destination	Frame Subtype	Action Frame Subtype
76.799345	Node_1	Broadcast	Beacon frame	
77.823333	Node_1	Broadcast	Beacon frame	
78.847330	Node_1	Broadcast	Beacon frame	
78.848248	Node_2	Node_1	Action	Mesh Peering Open
78.948554	Node_2	Node_1	Action	Mesh Peering Open
79.152434	Node_2	Node_1	Action	Mesh Peering Open
79.157475	Node_2	Broadcast	Beacon frame	
79.464535	Node_2	Node_1	Action	Mesh Peering Open
79.848547	Node_2	Node_1	Action	Mesh Peering Close
79.871322	Node_1	Broadcast	Beacon frame	
80.181671	Node_2	Broadcast	Beacon frame	
80.895303	Node_1	Broadcast	Beacon frame	

Fig. 6: Node 2 blocked by Node 1 before joining the network

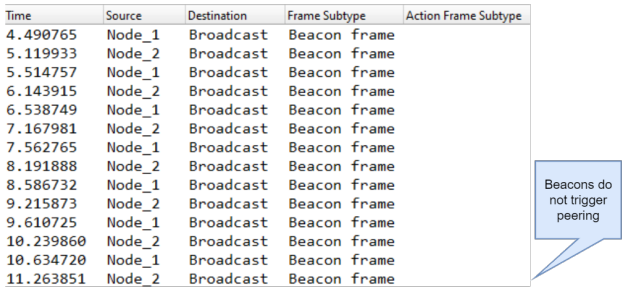
Scenario C: Initial bidirectional blocking

In this scenario, both test stations block each other initially. According to Fig. 7 the test stations are sending beacon frames periodically, but none of them will reply with an *open* frame to initialize link establishment.

Previously, this was not possible, as it would have been necessary to first establish a link in order to block it. Since links can break down temporarily depending on environmental conditions, mechanisms were necessary to reblock the link when the respective station reappears and the link is reestablished. This means that software had to either periodically perform blocking actions or determine when an unwanted link has been established in order to actively block it.

V. Conclusions and Outlook

In this work, we present a MAC filter approach for reliable peering and topology control in WLAN mesh networks. We demonstrate that, with our extensions, periodic link blocking solutions in userspace are no longer necessary. This reduces



Time	Source	Destination	Frame Subtype	Action Frame Subtype
4.490765	Node_1	Broadcast	Beacon frame	
5.119933	Node_2	Broadcast	Beacon frame	
5.514757	Node_1	Broadcast	Beacon frame	
6.143915	Node_2	Broadcast	Beacon frame	
6.538749	Node_1	Broadcast	Beacon frame	
7.167981	Node_2	Broadcast	Beacon frame	
7.562765	Node_1	Broadcast	Beacon frame	
8.191888	Node_2	Broadcast	Beacon frame	
8.586732	Node_1	Broadcast	Beacon frame	
9.215873	Node_2	Broadcast	Beacon frame	
9.610725	Node_1	Broadcast	Beacon frame	
10.239860	Node_2	Broadcast	Beacon frame	
10.634720	Node_1	Broadcast	Beacon frame	
11.263851	Node_2	Broadcast	Beacon frame	

Fig. 7: Both Node 1 and Node 2 block each other initially

management overhead in the mesh network and potentially enhances stability and security by effectively preventing sporadic establishment of unwanted links.

Careful blocking of unstable peer links is beneficial to counteract obstacle-induced variable link conditions and oscillating link states. Depending heavily on the responsiveness of routing protocols and metrics in use, these can otherwise lead to problematic path selection decisions.

Our modifications could be further combined, e.g., with MAC-layer trust zone formation approaches such as those proposed by Wall et al. [4, 5] to allow not only a separation of devices into different mesh networks, but also fine-granular control of individual links within the same network.

Further steps include implementing a whitelist mechanism to selectively allow only known participants in the network. Our present modifications are available as a patch[10] and we plan to integrate them into the Linux mainline kernel.

Acknowledgment

This work has been achieved in the project CargoAssist, which is funded by the maritime research program, an initiative of the German Federal Ministry of Economics and Climate Protection (BMWK) under the funding code 03SX601B.

References

[1] A. Cilfone, L. Davoli, L. Belli, and G. Ferrari, "Wireless Mesh Networking: An IoT-Oriented Perspective Survey on Relevant Technologies," *Future Internet*, vol. 11, no. 4, 2019.

[2] "IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, Feb. 2021.

[3] M. Rethfeldt, B. Beichler, H. Raddatz, F. Uster, P. Danielis, C. Haubelt, and D. Timmermann, "Mini-Mesh: Practical assessment of a miniaturized IEEE 802.11n/s mesh testbed," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2018.

[4] A. Wall, H. Raddatz, M. Rethfeldt, P. Danielis, and D. Timmermann, "ANTs: Application-driven network trust zones on MAC layer in smart buildings," in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV: IEEE, Jan. 2018.

[5] A. Wall, H. Raddatz, M. Rethfeldt, P. Danielis, and D. Timmermann, "Performance evaluation of MAC-layer trust zones over virtual network interfaces," in *2018 Fourth International Conference on Mobile and Secure Services (MobiSecServ)*, Feb. 2018.

[10] https://gitlab.amd.e-technik.uni-rostock.de/mesh/mesh-mac-filter.git