Poster: Seamless Client Integration for Fast Roaming in Wireless Mesh Networks

Martin Backhaus and Markus Theil and Michael Rossberg and Guenter Schaefer
Technische Universität Ilmenau, Germany
[martin.backhaus, markus.theil, michael.rossberg, guenter.schaefer][at]tu-ilmenau.de

***Abstract*—IEEE 802.11s enables rapid deployment of Wireless Mesh Networks (WMNs) to supply basic wireless connectivity for client hardware. However, if state-of-the-art routing protocols are up to the task in practice is not systematically evaluated, especially when considering seamless client integration without using central Wi-Fi controllers. This paper analyzes and enhances the Babel routing protocol for WMNs with attached clients. In particular, we improve roaming times by enhancing client route handling and integrating clients as if they were announcing themselves. We present client roaming performance based on real-world experiments leveraging a 20-node testbed.**
***Index Terms*—Wireless Mesh Networks, Mobility**

I. Introduction

While Wireless Mesh Networks (WMNs) have been an active research area for many years, a huge discrepancy exists between systems laid out in research publications and practical approaches using current IEEE 802.11 compliant hard- and software. Installations equipped with many clients and mesh routers are still unfeasible due to missing integration of non-mesh clients into mesh networks [1].

The distributed nature of WMNs cannot appropriately be translated into an available network architecture as most mesh solutions from companies like Cisco or Aruba use Wi-Fi controllers to handle client associations and build tunnels from mesh routers to the controller, which potentially poses a Single Point of Failure (SPOF). Even if the controller itself does not fail, mesh nodes can lose their controller connection, if mesh links or nodes on the path to the controller are hit by an outage. In a decentralized network, even in case of network partitions, a node can still communicate with every other node in its partition.

Existing open mesh routing protocols like OLSR [2] or B.A.T.M.A.N. [3] provide no means to support fast client mobility and robustness, i.e., fast and proactive route repair mechanisms. But WMNs might be valuable for first responder scenarios or industrial installations like mines or oil fields.

Given the sketched deficits, this paper studies the well-established Babel routing protocol [4] and provides enhancements which enable fast client mobility. We decided to enhance and fine-tune Babel over designing a completely new routing protocol, because it already provides a good starting point. Babel already employs distance vector mechanisms combined with an efficient update mechanism. It is also flexible, due to using Type Length Values (TLVs), and provides built-in hooks for extensions. Therefore, we decided to base our work on Babel and to systematically analyze and enhance it for fast routing convergence after roaming events. In particular, we suggest improvements for client prefix handling to yield fast re-convergence for mobile clients and evaluate our approach using real-world experiments using a 20-node testbed.

This work was supported by RWTÜV-Stiftung (Essen, Germany) under grant number S0189/10033/2018.

Annex to ISBN 978-3-903176-28-7

II. Requirements, Presumptions & Related Work

To work in real-life environments, a robust WMN has to work decentralized, i.e., there is no SPOF like a Wi-Fi controller. As a quantitative requirement, we want clients to finish the transition between Access Points (APs) in much less than 1 second – including time for routing re-convergence.

We assume that every router can learn the IP addresses of (mobile) Stations (STAs) immediately after they are connected to reduce the time which would occur when using dynamic address configuration. Fast mobility without central coordination is otherwise not achievable given current protocols. In the following, we will refer to this as *fixed IPs*. Furthermore, we assume every router to have a synchronized clock with an offset $\theta_t < 50\,\text{ms}$ (less than a Wi-Fi client's re-association time) to every other router, e.g., by using [5], [6].

Babel [4] is a distance vector routing protocol using sequence numbers to guarantee route information freshness, thus, preventing the count-to-infinity phenomenon. It has no explicit client mobility support – especially for the case of fixed IP addresses. A client's disassociation would most likely result in a retracted route, to be re-announced from the new AP. The new route will have a different router ID (as it originates from a new mesh router; these IDs identify Babel speakers). Changing router IDs may indicate formation of a routing loop, requiring reactive measures to resolve [4]. This measure takes one complete round trip to regain connectivity.

State-of-the-art routing protocols like modified protocols stemming from ad-hoc networks, e.g., Hybrid Wireless Mesh Protocol (HWMP) [7], assume a high degree of node mobility, but (implicitly) assume all nodes are mesh capable. WMM [8] caches the client location and updates it on mobility. Flooding is used, when the location of a client is unknown. IP options are used to store and learn location information of the client. B.A.T.M.A.N. advanced [9] uses roaming advertisement messages to update client associations in the corresponding table of the last router. To sum up, there is no protocol that

efficiently provides convergence speedup for the subset of routes from mobile clients.

III. Babel-MC for Seamless Mobility Support

Our enhancements to Babel, called Babel-MC (short for Mobile Clients), convert client routes into virtual sources and announce them as if a client is an independent Babel instance for *seamless client integration*. Babel can cause short routing loops, if the same prefix is announced by multiple routers at the same time [4]. Mobile clients with fixed addresses (as presumed in this work) trigger this situation frequently, if the last router has not detected that a station moved to the next router – which already announces the client's prefix.

Furthermore, the Babel RFC [4] states that synchronizing sequence numbers between different routers originating the identical prefix is cumbersome and should not be pursued. Instead, Babel enables to announce the same prefix from different routers using distinct router IDs – therefore, sequence number synchronization is superfluous. As mentioned, mobile clients will cause multiple prefix announcements (from different routers), leading to update distribution for the same client using different router IDs. While announcing the same prefix on different routers is perfectly viable in case it is actually reachable (e.g. default routes or multiple adjacencies to the same subnet), announcing a client prefix from multiple points in the network, while it is only reachable from a single router with a Wi-Fi connection, renders the client temporarily unreachable from some parts of the network. Furthermore, a changing router ID may indicate a routing loop in formation and Babel may interpret it as a retraction – as explained before (see Sec. II). A new route will then be formed after the routing information is explicitly refreshed, resulting in at least one more round trip time of delay prior to successful client roaming.

This behavior is unsuitable for seamless client integration. To improve the mechanism, we choose consistent router IDs (for client routes only) which would result in the above-mentioned issue of sequence number synchronization. Omitting sequence numbers and replacing them with timestamps automatically resolves this issue. This creates two tasks: 1) Choosing a consistent router ID for each client route and 2) integration of timestamps associated with these routes.

1) Consistent Virtual Router IDs: As we presume clients to have a fixed IP address no matter to which router they are associated, we can announce this IP as a /32 prefix in case of IPv4 (and a /128 prefix for IPv6). A SHA-256 hash of this prefix can be truncated to obtain a router ID, a so called Virtual Router ID (VRID), resulting in a consistent router ID for each client that can be easily derived from every router in the network without any overhead. The VRID is then used to announce the client prefix containing only a single IP address. This way, it seems as if a client represents an independent Babel instance announcing itself, no matter to which router a client is actually associated.

2) Client Updates w/ Timestamps: Using timestamps instead of sequence numbers for virtual client routes automatically grants useful properties for roaming, which sequence numbers

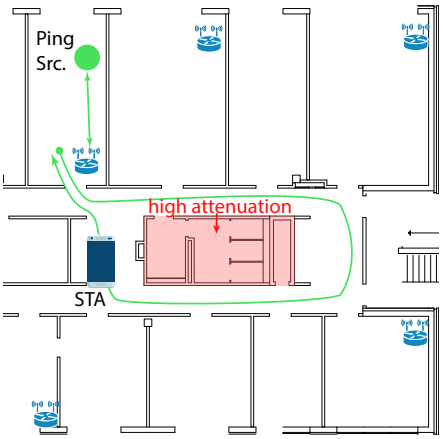

Fig. 1. Real-world testbed with client (incl. its trajectory) and a ping source. The highlighted red area in the middle (restrooms and a supply duct) inhibits Wi-Fi signals of our test APs passing trough it.

cannot. In particular, all events within one roaming process and different roaming processes themselves can be chronologically compared. As long as these events result in routing updates, each node will be able to determine their freshness. Timestamp encoding uses two unsigned 64-bit numbers for seconds and nanoseconds. Using timestamps requires synchronized clocks (see Sec. II). As we continue to use sequence numbers between mesh routers, a time synchronization protocol can rely on established mesh routes in order to reach a time server or other mesh nodes. As each Babel TLV should be uniquely decodable based on its type, we added a TLV called *ClientUpdate* for these timestamps.

If a *client is detected to be lost*, Babel-MC announces the client's prefix with an "expensive" metric – which is not as severe as a route retraction. Choosing an expensive metric implies other routes (as soon as they come up) to be better and therefore preferred in the future. The reason not to send a definite retraction in case of roaming is that the client will be most likely reachable again after a very short transient phase. Therefore, explicit retractions are too harsh, especially if roaming events are restricted to a small area of the network, i.e., client roaming between neighboring mesh nodes. Finally, if a client *connects to a new router*, we announce the client with its precise metric to update information on other nodes. Leveraging timestamps in this process leads to seamless client roaming without explicit retractions in case of fixed client IPs.

Concerning fixed IP addresses of clients, our real-system implementation uses a small helper tool, which adds client routes to announce when a client signaled his IP address(es). In our implementation, this requires X.509 certificates on each mobile client, which includes an IPv4 and/or IPv6 address, but it is also possible to achieve by other means.

IV. Evaluation

We discuss our mechanism qualitatively first, before giving quantitative results of Babel-MC using a real-world experiment. Numerical results are averaged over 32 repetitions with 99% confidence intervals.

Our developed approaches meet initially stated functional requirements. Clients are seamlessly integrated; we were able to

reuse and extend Babel functionalities to do so. The presented approach requires no centralized instance, therefore, no SPOF exists. Using VRIDs, routes of clients never need to be retracted completely in case of roaming and they are announced to work together with Babel's mechanisms to deliver feasible routes without the need for explicit sequence number requests or route requests (as they would occur in Babel's usual operation with roaming clients carrying the same IP address). This provides seamless client reachability when roaming.

In our experiment, we measure the total time of a roaming process in the routing protocol, i.e., the client moves from one node to another and the new route is available in the network (such that the client is reachable again). A mobile client (smartphone) carried by a pedestrian moves in our testbed while being pinged from a stationary notebook, see Fig. 1. Measurements are done in a simplified fashion using ICMP echo requests and replies between the ping source and the client. We record and examine ping traces, which will give an upper bound for the time required for successful client roaming when counting the quantity of lost replies. As it is not obvious how many replies x will be lost when trying to distinguish client roaming and regular packet loss, we show results for varying values of x. We compared our approach to *babeld* [10] as a reference. This standard Babel implementation uses a rather large TLV flush interval of 2.0 s by default, which we patched to 100 ms to match our approach.

Because we are performing measurements within our testbed, some specifics to this real-world scenario have to be discussed. As our approach should be usable with off-the-shelf Wi-Fi devices, we test with an unmodified Samsung Galaxy S8 smartphone. Therefore, its roaming decisions are hard to anticipate. Samsung lists multiple criteria for its Android-based smartphones, like lost beacons, low RSSI, or high channel utilization [11]. As we cannot completely control these criteria, the smartphone roams with a different frequency between runs. We therefore only measure the outage after a successful roaming event and cannot use or anticipate its exact point in time. The ping source is a Lenovo Thinkpad T470s running Linux 4.15. Babel-MC is implemented as a C++ daemon running on Linux-based 1GHz x86_64 embedded PCs (PC Engines APU2C4). We leverage NTP for clock synchronization in our prototype.

The client roaming time should be as fast as possible, but some restrictions apply. As we use EAP-TLS for exchanging certificates with IP addresses, the STA requires at least 50-70 ms to roam to a new AP, after it has scanned its environment. Additionally to the client roaming to another AP, Babel-MC has to recognize the newly attached STA in form of a new client route and distribute this route. Therefore, we can estimate the overall roaming time (until echo replies can be received again), to be at least the STA Wi-Fi roaming time of 50-70 ms and several seconds as a maximum. Fig. 2 shows the achievable end-to-end roaming time assuming a varying quantity of consecutive lost pings, which were send in 10 ms intervals. As transmissions over Wi-Fi naturally experience some packet loss, we should not consider one or two lost packets as an interesting roaming timespan candidate. Being

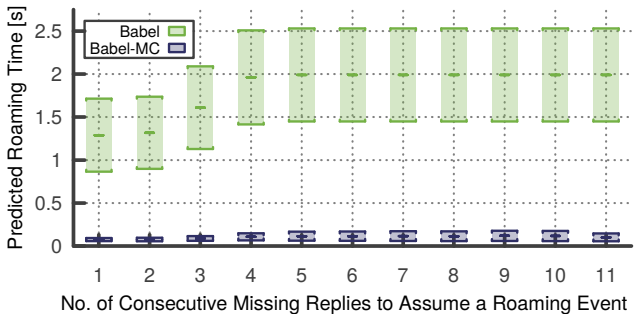

Fig. 2. Predicted time for client roaming events.

conservative, we consider outages to originate from roaming events in case of at least seven consecutively missing echo replies. This yields a mean roaming time of 118 ms for our implementation, while babeld roams in 1.99 s. If we compare 118 ms with our estimate, it is residing in the lower end of the possible spectrum and can therefore be considered fast. Babeld is slower as it involves more steps to move the client IP prefix to another router.

V. Conclusion and Future Work

In this article, we presented Babel-MC, an extension of Babel that seamlessly integrates clients into distributed WMNs. Our evaluation with real-world experiments proved its ability to provide fast client roaming times. This work is only one of many building blocks needed for large and scalable, distributed mesh networks. Future work is necessary in automatic multi-channel selection strategies along with distributed client roaming assistance, e.g., estimated neighbor lists for fast scans from wireless clients.

References

[1] S. Sampaio, P. Souto, and F. Vasques, "A Review of Scalability and Topological Stability Issues in IEEE802.11s Wireless Mesh Networks Deployments," *International Journal of Communication Systems*, 2016.

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

[3] D. Johnson, N. Ntlatlapa, and C. Aichele, "Simple Pragmatic Approach to Mesh Routing Using BATMAN," in *IFIP Intl. Symposium on Wireless Communications and Inf. Techn. in Developing Countries*, 2008.

[4] J. Chroboczek, "The Babel Routing Protocol," RFC 6126, 2011.

[5] J. So and N. H. Vaidya, "A Distributed Selfstabilizing Time Synchronization Protocol for Multi-Hop Wireless Networks," Technical report, UIUC, Tech. Rep., 2004.

[6] M. Rossberg, R. Golembewski, and G. Schaefer, "Attack-Resistant Distributed Time Synchronization for Virtual Private Networks," in *Intl. Conference on Computer Communications and Networks*, 2012.

[7] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 2016.

[8] D.-W. Huang, P. Lin, and C.-H. Gan, "Design and Performance Study for a Mobility Management Mechanism (WMM) Using Location Cache for Wireless Mesh Networks," *IEEE Transactions on Mobile Computing*, 2008.

[9] Open Mesh, "B.A.T.M.A.N. advanced," 2020.

[10] The Babel Routing Daemon Git Repository. Accessed: 20-Jan-2020. [Online]. Available: https://github.com/jech/babeld

[11] Samsung: Enhanced Roaming Algorithm. Accessed: 21-Jan-2020. [Online]. Available: https://support.samsungknox.com/hc/en-us/articles/115013403768-Enhanced-Roaming-Algorithm