Simulating a City-Scale Community Network: From Models to First Improvements for Freifunk

Tobias Hardes, Falko Dressler and Christoph Sommer
Distributed Embedded Systems Group, Dept. of Computer Science, University of Paderborn, Germany
{tobias.hardes,dressler,sommer}@ccs-labs.org

***Abstract*—Community networks establish a wireless mesh network among citizens, providing a network that is independent, free, and (in some cases) available where regular Internet access is not. Following initial disappointments with their performance and availability, they are currently experiencing a second spring. Many of these networks are growing fast, but with little planning and limited oversight. Problems mostly manifest as limited scalability of the network – as has happened in the Freifunk mesh network operating in the city of Paderborn (approx. 800 nodes running the BATMAN IV protocol with control messages alone accounting for 25 GByte per month and node). In this work, we detail how we modeled this real-life network in a computer simulation as a way of allowing rapid (and worry free) experimentation with maximum insight. We describe how we parameterized and validated this model using gathered measurements. Based on the model, we were able to investigate alternative structures and parameterizations to improve the performance. The predicted performance gains map well to those measured after the proposed changes were deployed city-wide.**

I. Introduction

In recent years a sharp rise of wireless community networks could be noticed [1]. Such networks provide a wireless mesh network that is established and operated by private persons. The goals of such projects are diverse, from operating an independent network, to one that is free [2], to one that is available where regular Internet access is not, to economic incentives [3]. As a consequence of their goals and operation, the mesh networks are frequently managed in a decentralized way, with limited planning and little oversight. This, coupled with fast and unplanned growth, repeatedly leads to scalability problems. Addressing them in a structured way requires either experimentation or simulation.

Experimentation would allow quick and direct feedback on the suitability of extensions – and indeed a multitude of small to medium scale experiments in testbeds can be found in the literature [4]. However, broad parameter studies or deeper insights into network behavior cannot be afforded by experiments. At the same time, any experimentation in a large scale real-world deployment would require a coordinated effort among all participants while potentially putting the integrity of the network at risk.

Simulation, on the other hand, would allow maximum insight into how the network behaves and allow for quick, worry free parameter studies. Yet, simulation requires a holistic simulation model that captures the characteristics of such a complex network at scale.

In this paper, we focus on the Freifunk community network initiative, which is composed of a few hundred local communities, each serving from few dozens to thousands of nodes. In particular, we concentrate on the Freifunk community network composed of approximately 800 routers (nodes) in the city of Paderborn. This community network, like most Freifunk networks, employs the Batman [5] protocol (specifically, BATMAN IV) to operate its mesh network and allows anyone to join as a user, simply by connecting their laptop to a Freifunk node via WiFi. Like many, it suffered from scalability problems.

We tackle the investigation of these problems by providing a first city-scale simulation model of a BATMAN IV community network. To simulate the network, we first implemented a computer simulation model of BATMAN IV for the OMNeT++ discrete event simulation engine. We then gathered measurements in the real network to parameterize the model and validated its agreement with measurement data based on traces. We make all simulation models (and their parameterization to resemble the Freifunk network of Paderborn) publicly available as Open Source[1] to serve as a basis for future research on community networks.

As a case study, we investigate possible adjustments of protocol parameters to improve the performance of the Freifunk network. Our primary objective was to reduce the load caused by the transmission of control messages: control messages alone were accounting for 25 GByte per month and node.

One approach could be simply to increase the interval to send control messages. This would immediately reduce the load, but it would increase the time until a new mesh or non-mesh participant gets announced in the network as well. There are some mechanisms to alleviate those effects like the *SpeedyJoin* approach, which enables nodes to temporary add routes for unknown non-mesh participants. Nevertheless, the latency for new mesh participants would be increased.

Therefore, we opted for a different solution, reducing the base load without negatively impacting several metrics we deem important for smooth operation of the network. We conclude this paper with a report on this candidate improvement that we examined in the simulation, which led to substantial performance gains. This improvement has been deployed city-wide in the real world deployment as well. We were able to demonstrate that the predicted performance gains map well to those measured in the real deployment.

[1] http://www.ccs-labs.org/software/ffpb/

Our main contributions can be summarized as follows:

- We created a first computer simulation model of a city-scale community network based on BATMAN IV. Our model is based on the Freifunk network operating in Paderborn.
- We validated the model based on measurements, finding good agreement between metrics collected in simulation and the real network.
- We make the model publicly available as a basis for future research on city-scale community networks.
- We demonstrate that our model allows to extrapolate the impact of improvements from simulation to the real world.

Our work is structured as follows. In Section II we start with an overview of BATMAN IV, followed by related work and previous investigations in this area. Section III presents the modelling procedure. Here, the focus is on techniques on how to gather data from an existing network in order to use this information to automatically generate all configuration files for the simulation. Section IV illustrates the validation of the model based on metrics and measurements from real nodes in the Freifunk network. We focus on one metric and describe common problems and limitations. Section V shows the implementation of one improvement. This improvement has been implemented in the simulation in order to analyze the impact for the real network. Later on, this improvement has been implemented in the real network. We demonstrate that the model gives a very useful representation of the real world scenario. Section VI summarizes the results and concludes the paper with proposals for future work.

II. RELATED WORK

A wide variety of protocols have been proposed to establish wireless mesh networks in the literature. Out of these, only few have been deployed in city-scale mesh networks, first and foremost among them modern variants of the Optimized Link State Routing (OLSR) [6] protocol. The Freifunk initiative has instead settled on variants of the Batman [5] protocol, which is a more community driven effort to mesh routing in typical scenarios of community networks. Most Freifunk networks today are relying on the BATMAN IV protocol version.

Other than comparable work, the BATMAN IV protocol operates on top of OSI Layer 2 [7], [8], transporting both all control messages and all data traffic using raw Ethernet frames. This way, BATMAN IV makes mesh routing transparent to any application run by the user (that is, all nodes appear to be connected via a big network switch). The goal of this is to allow any protocol stack (e.g., IPv4 or IPv6) as well as applications needing broadcast/multicast functionality (e.g., mDNS service discovery) to be used unmodified.

To maintain the mesh network, a node running BATMAN IV periodically broadcasts control messages, called Originator Messages (OGMs). OGMs are used to announce nodes in the network and to establish multi-hop routes. The goal of this routes is for any node to be able to reach any mesh or non-mesh destination in the network. Furthermore, OGMs are used to announce new non-mesh participants like user devices (i.e., unmodified smartphones or laptops connected via WiFi to a mesh node) and to measure the link quality to direct neighbors. Neighbors which are receiving the OGMs are rebroadcasting them according to specific rules in order to transmit them to all participants in the network.

Most of publications associated with Batman are bachelor and master theses [7]–[12]. However, there is no documentation of the BATMAN IV algorithm except a public wiki,[2] and there is no dedicated investigation of BATMAN IV at city-scale in terms of correctness, performance, or reliability. The wiki gives a rough overview about the basic concepts, configuration examples and several tutorials. Consequently a lot of scientific publications simply reference the wiki to argue about properties of algorithms.

There are a few publications that examine BATMAN III and other protocols by using network simulations. One of these is the work of Bowitz [9], which presents an implementation of a first version of BATMAN III in ns-3. Here the author aimed to extend BATMAN III in order to use X.509 certificates. The simulation model has been verified by using debug information that were gathered from a small real world network. However, there is no additional information about the concrete validation and about effects in the real world.

Further publications compare BATMAN III and BATMAN IV with other mesh routing protocols from layer 2 and layer 3. Some analyze BATMAN III in small [13] or in synthetic scenarios, but there is no publication yet about a mesh network in a complete city.

Many, such as Barolli et al. [14], compare OLSR and BATMAN III. As a further example, Kulla et al. [15] investigate the behavior of OLSR and BATMAN III in a stairwell spanning five floors. Here, the throughput, delay, and packet loss are used as metrics. The authors analyze different scenarios like with only stationary nodes or when some nodes are moving in the network. Yet, the results are highly dependent on the environment and the author does not provide any information about the concrete scenario. Further, the experiments are performed with only five nodes.

Similarly, Abolhasan et al. [16] compare the performance of three mesh routing protocols, OLSR, Batman, and Babel, in terms of bandwidth, packet delivery ratio, delay, and route convergence latency. Experiments were performed with eight mesh nodes deployed in an office environment.

It can be observed that most of the analyzed studies achieve slightly different results in their comparison of Batman with other protocols. This is a common consequence of different environments and uncontrolled experimental conditions like interference on the wireless channel or the environment. As none of the publications give a deeper insight to this, it is difficult to draw a conclusion to this.

In summary, to be best of our knowledge our investigation is the first publication on Freifunk and BATMAN IV that uses a model of a network from real world in a simulation with a configurable and controlled environment.

[2] https://www.open-mesh.org/projects/batman-adv/wiki/

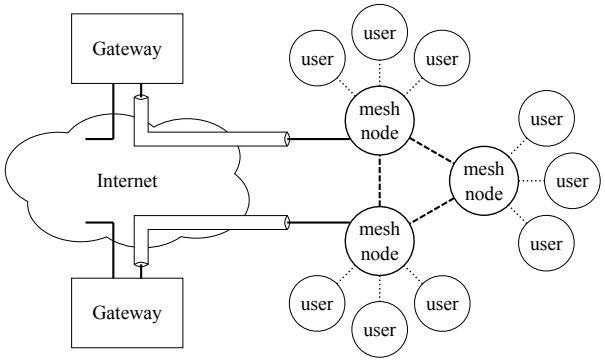


Figure 1. Architecture of the Freifunk Paderborn network: The core network is operated by mesh routers running BATMAN IV; users connect to these. Mesh routers can optionally connect to Internet services via a tunnel established through gateways. Two gateway connections were used to create redundancy.

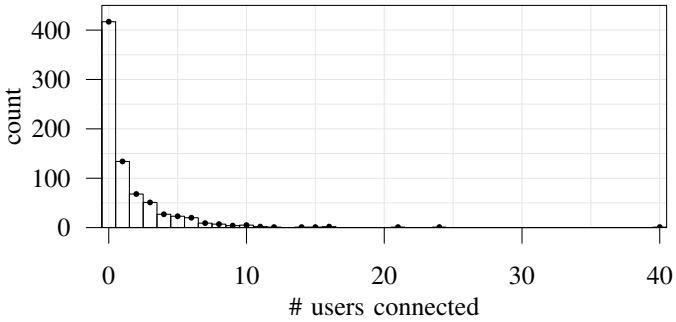


Figure 2. Number of users connected per mesh node in the simulation model.

III. MODELLING THE NETWORK

Figure 1 gives an overview of the network topology of the Freifunk network in Paderborn. Nodes are meshed via both wireless links and via (optional) connections to gateway nodes, tunneled over a standard Internet connection. Both are using BATMAN IV for routing. Aside from participating in the routing, the gateway nodes also provide connectivity to Internet services. For redundancy, each tunneled connection to a gateway is supplemented with a second fall-back connection. Users (i.e., their end devices) connect to mesh nodes to use services in the Freifunk network or the Internet. For this, mesh nodes operate as a standard WiFi access point.

Our first step in modeling the Freifunk network of Paderborn was to measure all needed input data. A Freifunk network is an open platform, which is usually documented by using various statistics in order to measure the use of the complete network. As, today, BATMAN IV is usually used in Freifunk, it is possible to collect arbitrary information of the network by using the *Alfred*[3] service. The daemon is used in Paderborn to share information about the network topology and node properties. We filtered and aggregated this information in order to derive data for our use case. In our case, several information elements have been used in order to build the model.

On a macroscopic scale, our model encompasses 4 gateways, 775 mesh nodes, 1151 users (a middle ground of the oscillating user count between 500 and 1500, depending on the time of day and day of week). For more detailed aspects of the model, we first needed to know which node is connected to which node via the wireless channel. This is important for the forwarding of OGMs as nodes without a tunnel connection need to use others as next hop to reach the remainder of the network. Complementary, we need to know all nodes which are connected to a gateway node by using a tunnel as well. To place nodes next to each other like they are placed in the real world, we are also interested in the physical location of nodes. Last but not least, we need to know about the amount of non-mesh participants (i.e., users), which are connected to a given node. This is important for our metrics we use to validate the model. As the Alfred information is always highly up to date, we can use it to continuously poll a representation of the system at subsequent moments. This allows us to use historical data to check the model validity. To gather data over a period of time, data of mesh nodes (cleaned of personally identifiable information) has been captured and stored in the network.

Figure 2 illustrates the results of investigating the distribution of the number of connected users (i.e., clients) per mesh node in the form of a histogram. Its distribution closely mirrors that of other community networks as reported by Maccari and Lo Cigno [1]. While the number of connected clients is important for the model, the clients properties itself are not required, as we will show later.

All nodes are registered on the gateways with their MAC addresses in order to establish the tunnel. By using the MAC of each individual node, it is possible to extract and filter data for each node of the network. Here, the geographic locations and known mesh neighbors are of most interest in order to have a simulation model that is as close to the real network as possible. If a new node gets deployed in the real world, the owner is able to annotate metadata with the node position in the form of GPS coordinates. All known GPS positions of available nodes are used for the simulation model. One reason for this is that we want to closely model not just the backend, but also the wireless channel in the network, so physical topology influences network topology: WiFi mesh connections can only be established in the model if two nodes are in range of each other. This plays a major role for any node that is not directly connected to the gateway infrastructure by using tunnel connections, but only by using the WiFi mesh to other nodes. There is also a higher network load on the wireless channel if more nodes are in physical proximity. Having a higher network load leads to more collisions on the channel and the distribution of OGMs from mesh nodes becomes harder.

However, because of privacy concerns, the GPS position is optional information. Furthermore, the coordinates are static values in the node's configuration file and not automatically updated by some GPS sensor or by IP location. So there is a chance that a node's GPS position is not available, inaccurate, or simply completely wrong. This would lead to an error in the model. To deal with the absence or inaccuracy of GPS

[3]https://www.open-mesh.org/projects/alfred/wiki/

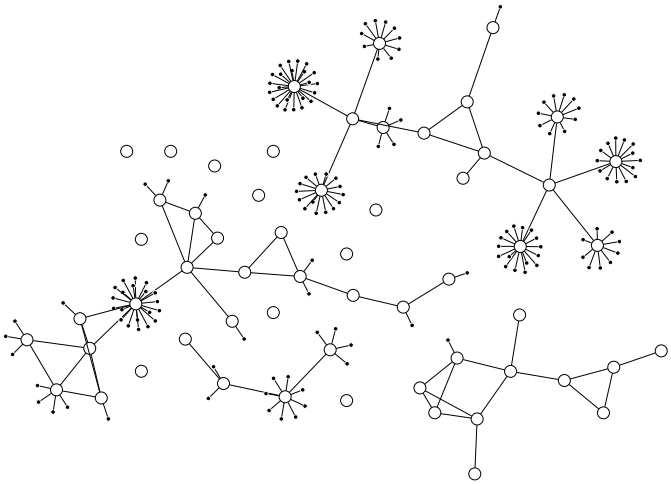


Figure 3. Small section of the Freifunk Paderborn wireless network simulation model visualized as a graph. Mesh links (long lines) connect mesh nodes (circles), which are surrounded by connected users (dots).

information, a second parameter is taken into account. The Alfred record contains information about the direct neighbors (one-hop neighbors) of a node. Those neighbors can be both mesh and non-mesh neighbors. As all nodes are well known, all non-mesh neighbors can be removed from the list and the result is a list of mesh neighbors for a given node. In the model, those nodes can be placed next to each other, thus being able to communicate. While, in reality, there are also special devices with directed antennas that cannot be mapped using this procedure, in Paderborn, the number of such devices is negligible. It further needs to be noted that this procedure also does not ensure a 100 % mapping to geographic location: there are also nodes without any neighbors or a GPS location. It still accurately maps the network topology, which is what we are concerned with in our study. Figure 3 illustrates the resulting topology of this process.

The gateway nodes are also listed as direct neighbors. Using this information, the direct tunnel connections to gateway nodes can be easily modelled in the simulation. All gateway nodes themselves are connected to each other in a full mesh topology. There are eight different gateways in Paderborn with different hardware resources. As VPN connections require a relatively high amount of CPU power, each gateway node is able to handle a different amount of nodes. With this, there is an invariant router distribution across all gateways in the network. According to the historical Alfred data, the rough number of nodes for each of the gateways remains the same over time. Gateway connections are thus non time variant in our model.

As an execution environment for the Freifunk network model we selected OMNeT++ [17], a module-based discrete event network simulation engine. It supports hierarchical modules which makes the development of models of computer networks straightforward: A network contains devices and each device contains multiple layers, each represented by one module. Furthermore, the simulation environment provides a graphical editor, which allows us the usage of animations and a visual representation of the network in order to validate the model in smaller dimensions [18]. It also ships with a large collection of simulation modules aimed at modelling computer networks, the INET Framework. This library provides different predefined device models for wireless and wired communication and also implementations of several standard Internet protocols like TCP, UDP, or IP. We use components like the *StandardHost* and the *WirelessHost* and adjusted those implementations in order to model real router and gateway nodes for the simulation. We also use available protocol components like DHCP or DNS. Furthermore, we are able to rely on its movement models of network participants.

While the INET Framework provides an implementation of BATMAN III, it does not provide one of our protocols under study, BATMAN IV. Therefore, we had to implement the BATMAN IV algorithm as well. Here, the announcement of the nodes is of most interest. Therefore, the complete OGM handling and gateway announcement needed to be modeled. We base this model on the publicly available source code[4] of BATMAN IV. Next, the announcement of new non-mesh clients is important in order to see whether any adjustment of the OGM processing leads to a negative impact on the time until a new client is known at each node in the network. Other parts of the algorithm are skipped as those are not used in Paderborn or as they do not have an impact on the OGM processing.

To set up simulations, we wrote a small script to transform all topology and trace data into OMNeT++ configuration files. Client devices are joining the simulation based on the historical measurements from the real network. In the beginning of the simulation, every node is configured to boot with a random delay. This avoids artificially synchronized timers, which in turn would lead to artificially increased collisions on the wireless channel. A central module collects and stores statistics.

On a regular desktop PC (an Intel i7-2600 running at 3.4 GHz), the completed simulation model executes at $1/8^{\text{th}}$ speed, consuming approximately 2 GB of memory.

IV. MODEL VALIDATION

As a first step towards validating our model of a city-scale Freifunk network, we considered which metrics to judge the model by. We base our decision on the metrics chosen in the cited related work on mesh routing as well as general work on simulation [19] and on multi-hop wireless networks [20].

Following this, we picked the following four metrics:

- Delay for node announcement: Delay until a new node is known by all other nodes in the network.
- Delay for client announcement: Likewise, but for new clients.
- Packet aggregation performance: Number of OGMs that are aggregated into one packet.
- Routing overhead: The total number of routing packets transmitted during the simulation.

[4]https://git.open-mesh.org/batman-adv.git/, e0172510

The measurement of these metrics in the real world is not trivial. The number of transmitted OGMs and data for the packet aggregation is not part of the Alfred data and this information is not available in public, as it is not of interest in general. The measurement can still be done with network protocol analyzers such as *Wireshark*[5] and *tcpdump*[6]. However, the measurement has to take place on a real device, operating in the Freifunk network. The router software is an OpenWrt operating system that is extended with special Freifunk properties and therefore it is possible to install additional packages like tcpdump. Unfortunately, most of the devices have too little storage to install tcpdump so that this is only possible on more powerful devices. To install tcpdump and to perform the measurements, root access on the device is required. The login by using a password is disabled by default due to security reasons, but a device can be accessed by using an SSH connection with a public key authentication. However, to use the SSH login, the public key must be added to the router manually and as all devices are owned by individuals, it is impossible to access them by using this procedure. However, there is a rather small subset of three routers where access is possible.

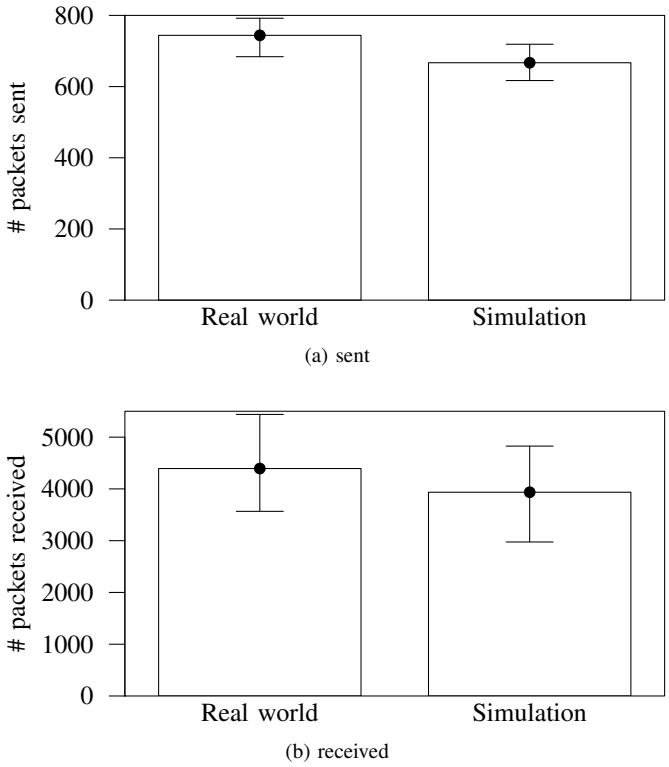
Like the Alfred data, the number of OGMs per node might also be affected by different factors. Therefore, we collected those measurements at different daytimes and different days of the week. As we have detailed historical data of the network from the Alfred daemon, we use the historical data validation in order to measure and validate the routing overhead. This can be measured with tcpdump directly on the nodes.

To measure the metrics in simulation, we instrumented the discussed simulation model, augmented by the ability to collect a variety of metrics. As the simulation contains stochastic elements, we repeat each experiment 50 times to derive meaningful statistics. Before we can start the measurement of the required metrics, the network needs to be fully established in the simulation. This means that every node is known to all other nodes in the network and a complete routing table is established in every node. To achieve this, a warm-up time is used during which no metrics are recorded.

The rationale behind measuring delays as one of our metrics is straightforward: If the handling of OGMs is changed, the time until a new node gets announced in the network might be affected. This would be a critical situation, as routing is not possible during this time. Because of this problem, the time until a new node gets announced should approximately be the same as in the real network. The same problem appears when a new non-mesh client gets announced, as this is also done by using the OGM mechanism. Those delays are not measured in the real world at the moment and therefore we do not have any data for this validation. We therefore instead compared delay measurements with theoretical calculations and found good agreement.

[5]http://www.wireshark.org/

[6]http://www.tcpdump.org/



(a) sent

(b) received

Figure 4. Sent/Received OGMs in the real world and in the simulation as a bar plot.

The other metrics are compared based on real world measurements.

The rationale behind measuring packet aggregation performance is straightforward as well: BATMAN IV is able to aggregate several OGMs. The goal of this mechanism is to reduce the control message overhead by sending one big packet instead of several small ones. Thus, packet aggregation performance is of prime interest. Interactions between the protocol layers can cause more or fewer OGMs to be eligible for aggregation into a single packet sent to other mesh nodes. We found that, typically, only a single OGM is sent per packet, but for approximately 25 % of packets up to 19 OGMs could be aggregated into a single packet in both the simulation and in the real world. This is indicative of a very good match between simulation model and real world on many levels.

The last metric we consider, and the most interesting one, is the routing overhead. For this, both sent and received packets are counted separately. On a high level, reducing the number of transmitted OGMs affects a lot of features of BATMAN IV. Based on the considerations of Law [19] we use the graphical approach in order to compare the model with the real system because of the rather small sample size.

The sent OGM count is shown in Figure 4a. The plot compares the simulation results with the data gathered from three nodes in the real world. We display the median packet count as well as the span of measurements (ignoring the top and bottom 1 ‰, which we treat as outliers). Here, the overall number of packets is slightly below the data from the real

world, but in light of the overall spread of data we deem it an acceptable fit. The received OGM count is shown in Figure 4b. As can be seen, received packet counts (which we focus on in the following) agree even better between real measurements and simulations: The spans are largely overlapping.

V. Network Improvements and Validation

A node needs only one connection to a gateway node in order to gather knowledge about all nodes and non-mesh clients in the network. However, to increase resilience, the Paderborn Freifunk network was configured to always establish a connection to a second gateway as well. Those connections are chosen by the BATMAN IV protocol and by the firmware on the router. The rationale is to allow nodes to quickly switch to the second connection if the primary becomes unusable. A gateway could become unavailable in case of a reboot or simply in case of a critical error.

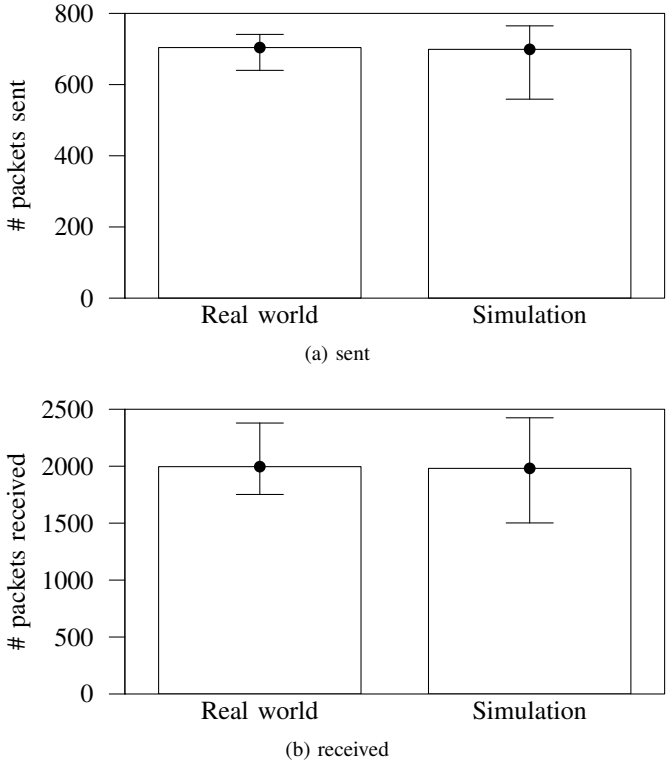
The downside is that, in this network topology, each node receives almost every OGM twice. This is because of the broadcast mechanism. Even though a node uses only one tunnel connection to send OGMs, due to the full mesh between all gateways, the second gateway broadcasts each received packet soon after the first gateway broadcasted the one it received. Because of the OGM processing of BATMAN IV, the OGM is identified as a duplicate and gets immediately dropped. Therefore, the second transmission is useless anyway in most cases.

A potential improvement is thus to remove the second tunnel in order to save almost 50 % of the OGM transmission. Unfortunately, this also removes the redundancy; consequently, the downtime of a node in case of a gateway failure is expected to be higher – potentially leading to ripple effects in the network. This could make the complete network unusable.

This risk is compounded by a potential real world experiment taking a substantial amount of time: To test this change in the network, firmware for all employed types of routers needs to be changed and tested. Afterwards, the new firmware needs to be deployed on all nodes in the network. The Freifunk network of Paderborn uses an automatic procedure to update all nodes in the network, so this would not require a high effort. However, the deployment to all nodes takes up to 24 hours. When all nodes are running with the new network, the effect needs to be analyzed and monitored which also takes some time. Only then could (potentially catastrophic) changes be rolled back and network functionality restored.

Therefore, we first implemented this proposed change in the simulation model in order to analyze the effects on the real network by using the model. Based on the results in Section III, we showed that our simulation model seems to be a good representation of the real network and results from the model should be representative for the real network. We thus first run the simulation again and analyzed the results in the same way as with the initial model validation in Section III.

As a first observation, the amount of used gateways does not affect the networks performance. The throughput of unicast packets remains unchanged. Based on our analysis, all other metrics are also not influenced in the simulation model. This was to be expected, as each node drops the OGM received from the second tunnel connection immediately. This confirms that, even in the original network, only the first OGM was used for the client and node announcement.



(a) sent

(b) received

Figure 5. Sent/Received OGMs by using one tunnel connection instead of two as a bar plot.

As the simulations indicated an expected reduction of received OGMs of almost 50 % with no adverse side effects, this change has been implemented in the real network. Firmware images for all router models have been adjusted to make nodes using only one tunnel connection, then tested and deployed to all nodes in Paderborn. After all nodes were running with the new firmware, we performed measurements on our nodes again to compare the real world result with the one gathered from the simulation.

The delays for client and node announcement are unmodified. This is not surprising, as our optimization only removes duplicate packets.

The results for packet counts are illustrated in Figure 5. Figure 5a shows the results for the sent packet count metric. The number of sent packets is similar to the measured value in the model validation. This was expected because only one tunnel connection is used to send packets to the gateway in both cases. Therefore, this metric is independent from the overall number of established connections.

Figure 5b shows the result of the received packet count as a bar plot. As expected, the amount of packets, that is, the control overhead in the network has nearly halved. By taking the average size of an OGM into account, we get a new load of

approximately 10 GB of data for each node and month caused by the transmission of OGMs. Here, as well, the data shows very good correspondence between the simulation result and the data from the real world, that is, the model reflects the behavior of the real network.

What is left to investigate is the impact of this change on delay if a gateway fails. After all, by deploying the discussed changes to the real network, we lost the redundancy and we expect a much higher downtime of a node in case of a gateway failure. The uptimes of gateway nodes in Paderborn fluctuate between one and one hundred days, with most gateways running no longer than 10 days. Therefore, it is not uncommon that a gateway becomes unavailable. In our scenario, the failures are mostly caused by gateways locking up; a reboot is then required to make the gateway running again.

To investigate the delay in case of a gateway failure in the real network by using two tunnel connections, we intentionally made one gateway unavailable and measured the time until this has been recognized by a given node (i.e., until the node switched to the second tunnel connection). For unmodified firmware, the time was between 120 s and 150 s. We repeated this measurement after we changed the firmware to use only one tunnel. After the changes were deployed, the delay was between 180 s and 200 s. So, in a worst case scenario, using only one tunnel instead of two, we increase the node's downtime by about one minute. Still, to put this into perspective, if we are using only one tunnel, even as much as one gateway failure per day causes the node to be unavailable for less than 1 % of the day. This is in contrast to a drop of control message overhead to almost half, so we deem this slight increase in potential downtime acceptable.

VI. Conclusion

In this paper, we described our process for simulating a city-scale community network based on BATMAN IV. This simulation model was designed to reflect a real-world deployment of such a network, that of the Freifunk community in Paderborn. The simulation model is implemented as a set of modules, one of them being our implementation of BATMAN IV, and configuration data for the OMNeT++ simulation engine. The complete model is composed of 4 gateways, 775 mesh nodes, and 1151 users joining mesh nodes following recorded data.

We validated the compound model against real world measurements using trace data and make it available as Open Source to serve as a basis for future research on community networks. We further demonstrated that our model allows researchers to extrapolate the impact of changes made to the network from simulation to the real world deployment. As a case study, we present results for a simple adjustment of the network to trade (in our view) negligibly longer outages for an almost halved control message overhead.

References

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

[2] T. Heer, R. Hummen, N. Viol, H. Wirtz, S. Götz, and K. Wehrle, "Collaborative municipal Wi-Fi networks - challenges and opportunities," in *8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM 2010)*, Mar. 2010, pp. 588–593.

[3] R. Baig, L. Dalmau, R. Roca, L. Navarro, F. Freitag, and A. Sathiaseelan, "Making Community Networks Economically Sustainable: The Guifi.net Experience," in *2016 Workshop on Global Access to the Internet for All (GAIA 2016)*, Florianópolis, Brazil: ACM, Aug. 2016, pp. 31–36.

[4] S. Vural, D. Wei, and K. Moessner, "Survey of Experimental Evaluation Studies for Wireless Mesh Network Deployments in Urban Areas Towards Ubiquitous Internet," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 223–239, 2013.

[5] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich, "Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.)," IETF, Internet-Draft (work in progress) draft-wunderlich-openmesh-manet-routing-00, Feb. 2008.

[6] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg, "The Optimized Link State Routing Protocol version 2," IETF, Internet-Draft (work in progress) draft-ietf-manet-olsrv2-19, Mar. 2013.

[7] M. Hundebøll and J. Ledet-Pedersen, "Inter-Flow Network Coding for Wireless Mesh Networks," Master's Thesis, Aalborg University, 2011.

[8] H. I. Kobo, "Situation-aware routing for wireless mesh networks with mobile nodes," Master's Thesis, University of the Western Cape, Mar. 2012.

[9] A. G. Bowitz, "Simulation of a Secure Ad Hoc Network Routing Protocol," Master's Thesis, Norwegian University of Science and Technology, 2011.

[10] D. Furlan, "Improving B.A.T.M.A.N. routing stability and performance," Master's Thesis, University of Trento, 2011.

[11] J. Klein, "Implementation of an ad-hoc routing module for an experimental network," Master's Thesis, Universidat Politecnica de Catalunya, Sep. 2005.

[12] F. Oehlmann, "Simulation of the 'Better Approach to Mobile Adhoc Networking' Protocol," Bachelor's Thesis, Technische Universität München, Sep. 2011.

[13] E. Chissungo, E. Blake, and H. Le, "Investigation into Batmand-0.3.2 Protocol Performance in an Indoor Mesh Potato Testbed," in *26th IEEE International Conference on Advanced Information Networking and Applications (AINA 2012), Workshops*, Fukuoka, Japan: IEEE, Mar. 2012, pp. 526–532.

[14] L. Barolli, M. Ikeda, G. De Marco, A. Durresi, and F. Xhafa, "Performance Analysis of OLSR and BATMAN Protocols Considering Link Quality Parameter," in *23rd IEEE International Conference on Advanced Information Networking and Applications (AINA 2009)*, Bradford, UK: IEEE, May 2009, pp. 307–314.

[15] E. Kulla, M. Hiyama, M. Ikeda, and L. Barolli, "Performance Comparison of OLSR and BATMAN Routing Protocols by a MANET Testbed in Stairs Environment," *Elsevier Computers & Mathematics with Applications*, vol. 63, no. 2, pp. 339–349, Jan. 2012.

[16] M. Abolhasan, B. Hagelstein, and J. C.-P. Wang, "Real-world performance of current proactive multi-hop mesh protocols," in *Asia-Pacific Conference on Communications (APCC 2009)*, Shanghai, China: IEEE, Oct. 2009, pp. 44–47.

[17] A. Varga, "The OMNeT++ Discrete Event Simulation System," in *European Simulation Multiconference (ESM 2001)*, Prague, Czech Republic, Jun. 2001.

[18] R. G. Sargent, "Verification and validation of simulation models," in *39th Winter Simulation Conference (WSC 2007)*, Piscataway, NJ: IEEE, Dec. 2007, pp. 124–137.

[19] A. M. Law, *Simulation, Modeling and Analysis*, 4th ed. Singapore: McGraw-Hill, 2007.

[20] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," in *4th ACM International Conference on Mobile Computing and Networking (MobiCom 1998)*, Dallas, TX: ACM, Oct. 1998.