AKadeMesh: Software-Defined Overlay Adaptation for the Management of IEEE 802.11s Networks

Michael Rethfeldt, Arne Wall, Peter Danielis, Björn Konieczek, Dirk Timmermann
University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18051 Rostock, Germany, Tel.: +49 381 498-7269
Email: michael.rethfeldt@uni-rostock.de

***Abstract*—The new standard amendment IEEE 802.11s enables low-level interoperability for future WLAN mesh networks. Support of the Hybrid Wireless Mesh Protocol (HWMP) and the Airtime Link Metric (ALM) for MAC-layer routing is mandatory. Its default distance vector routing mode facilitates scalability but also results in a limited network view per mesh node. Moreover, mesh mechanisms operate transparently to higher layers which makes the management and optimization of 802.11s networks a challenging task. Available on every standard-compliant node, ALM offers the potential to derive mesh topology information. We present *AKadeMesh* (Adaptive Kad-enhanced Mesh), a cross-layer approach specifically designed for 802.11s networks. It is based on the P2P protocol Kad and dynamically adapts its logical overlay to the physical mesh underlay by directly considering ALM. The resulting topology-aware P2P overlay is used to realize logical clustering for the distributed management of 802.11s networks, thereby maintaining unrestrained interoperability to the mesh standard. Our solution was implemented and evaluated in a real-world test bed. Results demonstrate its practical feasibility and verify the expected clustering benefit.**

I. Introduction

Compared to common centralized WLAN infrastructures, WLAN mesh networks feature automatic peering and multi-hop routing and thus provide a flexible and low-cost wireless network extension with higher scalability and robustness. The new standard amendment IEEE 802.11s enables low-level interoperability, integrating necessary mesh mechanisms directly into the 802.11 MAC layer [1]. Support of the Hybrid Wireless Mesh Protocol (HWMP) and the Airtime Link Metric (ALM) for MAC-layer routing are mandatory. The default reactive distance vector routing facilitates scalability but also results in a limited network view per node. Moreover, mesh mechanisms are transparent to higher layers which makes the management and optimization of 802.11s networks a challenging task. To overcome these limitations, we previously developed an *802.11s Management Framework* featuring centralized remote monitoring and configuration. Real-world evaluation proved that decentralization of the framework is necessary to ensure its scalability and robustness [2]. Common approaches in research propose logical network clustering with distributed management nodes, each being responsible only for a specific network region [3]. Nevertheless, to the best of our knowledge no specific solution for 802.11s exists that directly utilizes the standard's default mechanisms. Particularly HWMP and its ALM routing metric, available on every 802.11s-compliant node, offer the potential to derive mesh proximity information to serve as basis for cluster formation. Another current research field exists in the utilization of logical Peer-to-Peer (P2P) overlay networks to optimize distributed applications over wireless mesh networks, such as data organization, search, or synchronization [4]. Typically, P2P protocols do not consider the structure of the physical underlay which leads to low performance in lossy and dynamic environments, such as wireless mesh networks. Thus, cross-layer solutions must be applied that integrate underlay information into upper layers to mitigate this mismatch.

Consequently, we have developed *AKadeMesh* (Adaptive Kad-enhanced Mesh), a cross-layer approach specifically designed for 802.11s networks. It is based on the P2P protocol Kad and adapts the logical P2P overlay to the physical mesh underlay by directly utilizing the 802.11s standard's routing metric ALM. The resulting adaptive and topology-aware overlay is used to perform logical clustering for the distributed management and monitoring of 802.11s networks, thereby maintaining unrestrained interoperability to the 802.11s standard. Our solution was implemented and evaluated in a real-world test bed. Results demonstrate the practical feasibility of the approach and verify the expected clustering benefit.

The remainder of this paper is organized as follows: Section II outlines the basic principles of the 802.11s standard and its Linux reference implementation. We briefly introduce our previously centralized 802.11s management framework, that served as starting point for AKadeMesh. Subsequently, the P2P protocol Kad is motivated and its basic mechanisms are introduced. Section III describes the concept and design of AKadeMesh. In Section IV, we illustrate the results of our real-world test bed evaluation. In Section V, we discuss related work in the fields of mesh network management and the application of P2P protocols over wireless mesh underlays. Finally, we give a conclusion in Section VI and briefly state approaches for future research.

II. Technological basis

A. The IEEE 802.11s WLAN Mesh Standard

As first common industry WLAN mesh standard, IEEE 802.11s was ratified in September 2011 [1], [5]. It enables vendor-independent infrastructure-less multi-hop communication based on the widespread WLAN technology. Mesh functions like peering and routing are directly integrated into the 802.11 MAC layer. To ensure interoperability, every 802.11s

Mesh Point (MP) must support HWMP and its metric ALM as mandatory default profile for mesh routing [6]. HWMP is based on the reactive Ad-Hoc On-Demand Distance Vector (AODV) routing protocol [7]. Mesh paths are chosen according to the ALM. It represents the costs for transmitting a frame over a specific link in the mesh network by considering the applied WLAN physical layer and the wireless medium. The so-called *airtime cost* (c_a) is calculated as follows:

$$c_a = \left[O_{ca} + O_p + \frac{B_t}{r}\right] \cdot \frac{1}{1 - e_{fr}} \quad (1)$$

O_{ca} and O_p are constants for the channel access and MAC protocol overhead. B_t is the test frame size. By default, a frame size of 8192 bit is used. r denotes the test frame data rate, given in Mbit/s, whereas e_{fr} denotes the expected frame error rate. The estimation of e_{fr} as well as the values of the overhead constants are not predefined by the 802.11s standard but left open to vendor implementations [8]. Path tables on every MP contain forwarding rules to target nodes via best neighbors (smallest resulting ALM). Since ALM works as cumulative metric, it represents both overall path quality and length. The Linux project *open80211s* [9] is currently the most advanced open-source reference implementation of 802.11s. It is fully integrated into the *mac80211* kernel module, i.e. the Linux software WLAN MAC layer. Since some parameters in ALM calculation are left open to vendor implementations, open80211s provides own variants for error rate estimation and overhead constants. Real-world studies have demonstrated the routing performance of open80211s [8].

B. 802.11s Mesh Management Framework

Our solution is based on a centralized management framework for 802.11s networks that we developed previously [2]. Running on each mesh node, it encapsulates the features of the Linux 802.11s implementation *open80211s*. Every node runs an integrated SNMP agent (Simple Network Management Protocol). Local mesh status information, such as peer links or paths with ALM cost, and configuration functions are represented as SNMP data model. Status information and configuration functions can be accessed either by a local or an external SNMP client. Our framework further differentiates between an *Agent* and *Manager* role. The latter additionally integrates an SNMP client for remote query of status information and configuration of nodes. Thus, SNMP serves as interface to manage 802.11s nodes and to combine their otherwise limited network view to a global scope. Real-world evaluation with an increasing number of nodes revealed that decentralization of the existing framework is inevitable to ensure its scalability and robustness.

C. The Kad P2P Protocol

For realizing efficient structured P2P overlays, the Kademlia-based implementation variant *Kad* offers the best conditions [10]. To each peer, which is part of the Kad network, a hash value is assigned. Based on its hash value, a peer assumes a place in the logical address space. The distance needed to derive the similarity degree between two hash values is calculated by the XOR metric. Two values are similar if their distance is not greater then a defined search tolerance. In the protocol, the composition of the network and the communication between peers are specified [11]. To join the Kad network, a peer executes the so-called bootstrapping. For this purpose, it contacts a known peer, is inserted into the known peer's routing table, and learns further peers from the known peer's response. Each contact in a peer's routing table has a life time. An efficient maintenance mechanism makes sure that peers with expired life time are contacted and removed from the routing table if they do not answer. Furthermore, other peers are contacted periodically to learn new contacts (self- and random-lookup). Peers always have sufficient information about other peers in their direct neighborhood. In distant regions, they know at least one contact or more due to the routing table's flexibility. An iterative algorithm is applied for lookups on the address space [12], [13]. During a lookup, the searching peer first contacts peers closest to the requested hash value from its own routing table. In return, those peers answer the searching peer. Thereby, it learns new contacts until it knows contacts within search tolerance. The lookup algorithm is simplified by using the XOR metric, making Kad more consistent and efficient [14].

With focus on dynamically distributing the functionality of our 802.11s Management Framework at run time, Kad is best suitable due to its low maintenance efforts for adapting the overlay to dynamic changes [12], [13]. Moreover, its search tolerance mechanism can directly be applied for establishing cluster formation and control.

III. Concept

A. Mesh Management Scenario

In this paper, we present a decentralized management solution for 802.11s mesh networks. First, to illustrate the limited scalability of a centralized solution, we assume a simplified request response scenario for cyclic network monitoring. In every query cycle, a single management node (Manager) sends a request to each node, answered with a response containing the node's status information. With regard to the number of messages generated, a linear scaling is only given for mesh nodes in single-hop distance to the querying Manager, as no frames have to be forwarded by intermediary nodes. Considering a linear multi-hop path of nodes, the Manager being the first node of the path as a worst case, the resulting number of request/response messages per query cycle can be assessed as follows:

$$\#messages = 2 \cdot \sum_{i=0}^{\#nodes-1} (\#nodes - i) \quad (2)$$

Every node in the path must forward requests destined to its successors in the path. The same applies to all responses that need to be passed back to the Manager, located at the beginning of the path. After the first hop, *#nodes-1* requests and *#nodes* responses are forwarded, etc. Considering large mesh networks with tens to hundreds of nodes, it becomes obvious that a centralized monitoring approach is not suitable in terms of scalability and induced overhead.

B. AKadeMesh – Distributed 802.11s Mesh Management

Our framework *AKadeMesh* (Adaptive Kad-enhanced Mesh) focuses on the decentralized management of 802.11s WLAN mesh networks. It follows the Software Defined Networking (SDN) paradigm, shifting network control functions into higher abstraction levels. The aim of our approach is to create and maintain a logical network overlay that dynamically adapts to the mesh topology. Thereby, the adaptive overlay acts as middleware to abstract from the underlying physical network, providing a platform-independent address space as well as data organization and maintenance mechanisms. AKadeMesh is based on a self-developed, previously centralized 802.11s management framework and the Kad P2P protocol. The original framework features remote monitoring and configuration of *Mesh Agents* by a single *Mesh Manager*. AKadeMesh reduces the centralized monitoring overhead by distributing it to multiple Managers, each only being responsible for a part of the network. In this way, both scalability and robustness of the management plane are increased. The *Manager-Agent-Ratio* is used as service requirement for this application scenario. Constraints for AKadeMesh are overall mesh network size and topology.

The network topology is derived from the metrics of the mesh paths, i.e., the 802.11s standard's ALM. ALM represents both overall quality and length of a path. In [15] and [16] it has been shown that ALM outperforms other routing metrics, such as ETX and ETT. Thus, it is our preferred basis for cluster formation and Manager selection.

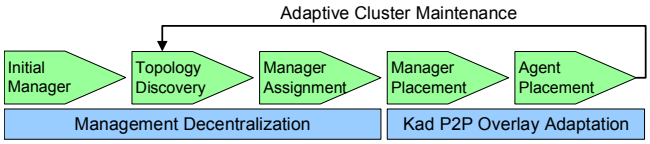

Fig. 1. AKadeMesh Work Flow

Figure 1 shows the work flow of AKadeMesh. In our prototype design we assume the existence of an initial Manager. In a practical scenario, this node could result from a preceding competition phase. The initial Manager performs a global topology discovery once, collecting the routing tables of all Agents, including the ALM costs to other nodes. Subsequently, additional Managers are promoted with respect to the *Manager-Agent-Ratio* service requirement and the current mesh topology, derived from the combined ALM. Based on the Kad protocol, all Managers are placed in a logical network overlay along with Agents in their responsibility zone, determined by ALM proximity. The resulting optimized overlay represents non-overlapping clusters of Agents, each assigned a separate Manager. Hereafter, the initial node placement in the logical overlay is dynamically adapted to the underlying mesh topology using the maintenance mechanisms of the Kad protocol. For instance, Kad provides for detection of node churn and roaming by means of *Bootstrap*, *Hello*, and *Ping* operations as well as peer life time expiry [11]. After overlay creation, distributed cluster maintenance and refinement is handled by the distributed Managers, again starting with topology discovery, i.e., intra-cluster ALM collection. Kad provides further operations for publishing and retrieval of information [11]. These are suitable to perform synchronization between Managers, such as the exchange of cluster status data or a handover of Agents due to roaming.

a) Manager Decentralization :

Based on the *Manager-Agent-Ratio* service requirement, the initial Manager selects *#managers* additional Managers from the remaining set of *#nodes* nodes. In our prototype design, the combination of Managers with maximum overall inter-Manager distance d is selected, using ALM as distance measure (see Listing 3).

$$\begin{aligned} \#comb &= \binom{\#nodes-1}{\#managers-1} \\ choice &= max \sum^{\#comb} \sum_{i!=j}^{\#managers} d_{ij} \\ i,j &= 1,...,\#managers \end{aligned} \tag{3}$$

After Manager promotion, all Agents are associated to their nearest Manager, denoted by smallest ALM. This simple approach prefers combinations with Managers located at the network edges, which helps in forming non-overlapping clusters but does not account for inhomogeneous mesh topologies. Nevertheless, it serves as satisfactory starting point and proof of concept for a clustering strategy, solely based on 802.11s ALM without the need for additional proximity information.

b) Overlay Adaptation :

Once determined, all Managers and their Agents are placed in the logical Kad address space that therefrom facilitates cluster maintenance. Figure 2 illustrates the Kad overlay adaptation.

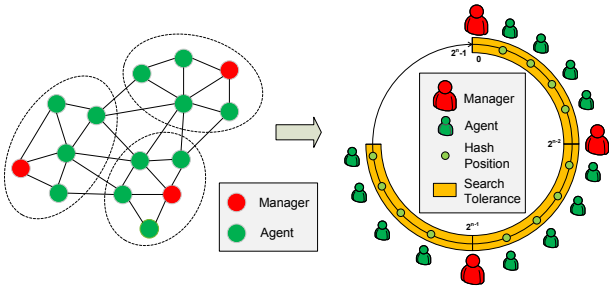

Fig. 2. P2P Overlay Clustering Approach

Depending on a given address space width and the number of Managers, the Kad search tolerance (ST) is adjusted as shown in Listing 4. It directly represents equally-sized clusters, i.e., the responsibility zone of each Manager. Thereby, the initial Manager obtains address 0 and is assigned the first logical cluster. The remaining Managers are placed consecutively in ST distance, each at the start address of its cluster.

$$\begin{aligned} i &= \lceil log_2(\#managers) \rceil \\ ST &= 2^{hashBitLength-i} \\ posMan_j &= j \cdot ST \\ j &= 0,...,\#managers-1 \end{aligned} \tag{4}$$

Subsequently, all Agents are sorted by ALM cost to their associated Manager in ascending order and placed equidistantly within their cluster address space, denoted by Manager position and ST:

$$posAgent_j = posMan + j \cdot \frac{ST}{\#clusterAgents} \qquad (5)$$
$$j = 1, ..., \#clusterAgents$$

Initial overlay creation and address assignment are performed by the initial Manager, acting as Kad bootstrap node. The new overlay addresses are synchronized in the network via Kad *Bootstrap* operations, triggered by the initial Manager. After first decentralization and overlay formation, Kad's *Hello* and peer timeout mechanisms [11] for the detection of incoming and leaving nodes can be used by the distributed Managers to handle on-line cluster maintenance. Each Manager issues the cyclic monitoring of mesh information only for Agents within its cluster, leading to a reduced network overhead induced by our 802.11s Management Framework. Manager synchronization and exchange of preprocessed status data is possible via Kad *Publish* operations [11].

c) Architecture :

Figure 3 shows the architecture of AKadeMesh. Our solution relies on the Linux WLAN MAC layer, implemented as Kernel module *mac80211*, which integrates the 802.11s implementation *open80211s*. As first application-layer component we run our Java-based Mesh Management Framework that automates the bootstrapping of an 802.11s node and acts as Agent (server) or Manager (client) to realize SNMP-based remote monitoring and configuration.

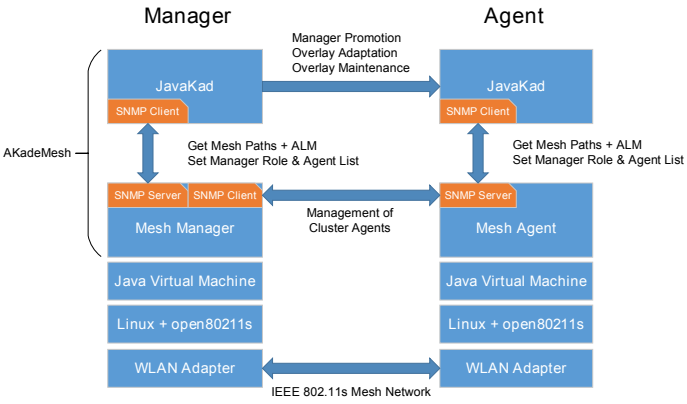

Fig. 3. Architecture of AKadeMesh

The network clustering and decentralization of the former centralized Manager role is done via Kad, running above the Management Framework. We use our own Kad implementation, called *JavaKad*. The combination of Management Framework and JavaKad composes *AKadeMesh*. Both components communicate locally via SNMP, as already supported by the Management Framework. This way, a mesh node's routing information, including ALM path costs, are integrated into JavaKad for overlay clustering. A mesh node configured as Agent can be promoted as Manager after receiving a corresponding command from another Manager via Kad. Thus, in top-down direction, JavaKad passes received promotions and cluster Agents to be monitored to the underlying Management Framework. Based on these, the latter performs cluster adaptation or switches between Agent and Manager role.

IV. Real-World Test Bed Evaluation

A. Experimental Setup

For practical evaluation, 12 devices were used to form a mesh network, comprising 10 Raspberry Pi boards and 2 notebooks running Linux distributions with Kernel v3.18. Every device was equipped with a WLAN USB adapter (Buffalo WLI-UC-GNM, Ralink chipset driver *rt2800usb*) configured to operate as 802.11s mesh point in 802.11g mode. All nodes were located in the same room within single-hop distance and configured to use a channel not overlapping with other networks in the building. We conducted both single-hop and multi-hop measurements. For the latter, open80211s enables the enforcement of multi-hop paths via peer link blocking. As the used WLAN adapters did not support transmission power reduction and a deployment range leading to path lengths of up to eleven hops was not feasible, enforcement was necessary for setting up a reproducible multi-hop environment. On each node, we blocked the establishment of links to all peers except predecessor and successor in the desired multi-hop chain. The enforced path represents a worst case scenario, as the possibility for frame collisions is increased and no parallel transmissions of non-adjacent nodes are possible, compared to a multi-hop topology caused by distance.

B. Test Cases

With focus on our mesh management application scenario we measured the amount of monitoring data sent by every mesh node, including both generated and forwarded SNMP messages. We used a separate WLAN adapter configured in monitor mode to capture the traffic of all nodes in the room using Wireshark v1.12. As communication over multi-hop paths generates additional frame transmissions due to forwarding on intermediary nodes, the induced network load is highly topology-dependent (see Section III-A). Thus, we developed three test cases to demonstrate the limited scalability of a centralized monitoring and the benefit of our decentralization approach using AKadeMesh.

The 10 Raspberry Pi boards were configured as Agents in all test cases. One of the notebooks was always configured as initial Manager. In the first test case, all nodes were allowed to communicate directly via a single hop. The initial Manager periodically queried status information from all Agents (including the second notebook), based on our centralized Management Framework. The second test case again comprised only a single Manager, with all Agents now connected in a linear multi-hop chain. Intended as a worst-case scenario, the Manager was located at one end of the path, the last Agent being the second notebook in eleven-hop distance. The third test case included AKadeMesh and the dynamic delegation of a second Manager. The second notebook, previously configured as Agent and located at the remote end of the path, was dynamically assigned the second Manager role. Following our decentralization strategy depicted

in Listing 3, both Managers were located at the end of the path, each responsible for a cluster of five Agents. After initial decentralization and cluster formation, both Managers sent SNMP requests only to Agents in their cluster. The SNMP query interval was set to 15 seconds and 25 query cycles were captured in all test cases.

C. Discussion

To compare the resulting overhead of the centralized single-hop, centralized multi-hop, and distributed multi-hop monitoring scenario, we averaged the SNMP traffic sent within 25 query cycles and normalized it to a single cycle.

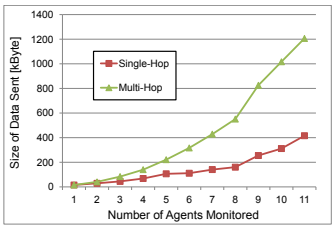

Fig. 4. Scalability of Centralized Monitoring

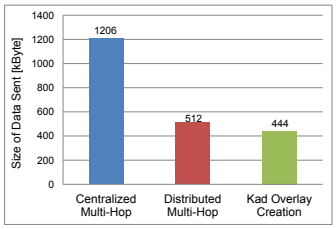

Fig. 5. Distribution Benefit of AKadeMesh

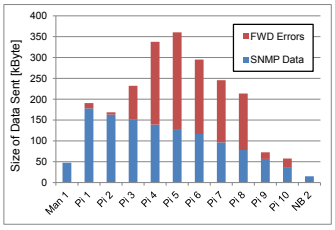

Fig. 6. Data Sent Per Node (Centralized Multi-Hop)

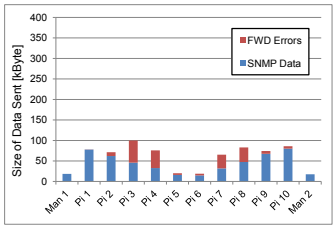

Fig. 7. Data Sent Per Node (Distributed Multi-Hop)

Figure 4 demonstrates the scalability of centralized monitoring in a single-hop and a multi-hop scenario, i.e., the first two test cases. In general, the amount of data sent increases linearly if all status information are queried directly within single-hop range. However, the availability of more nodes leads to larger SNMP responses per node, since mesh information, such as peer and path tables, increase with node count. In the multi-hop setup, data needs to be forwarded by intermediary nodes. Thus, in this scenario, the data sent in total increases non-linearly with path length (see Section III-A).

In the analysis of our traffic captures, we observed an open80211s bug causing routing loops in multi-hop setups with four and more hops (second and third test case). During the frame forwarding process, i.e., the SNMP response of an Agent was sent back to the Manager at the beginning of the path, occasionally intermediary nodes selected a neighbor leading in the wrong direction. The next hop node then forwarded the frame back in the right direction and frame circulation started. It ended after mesh frame time-to-live (TTL) had decremented to zero or path validity had expired, followed by a path refresh resulting in a possible correction of forwarding entries. Since mesh frame TTL is initialized to 31 by default, these circulations caused many more frame transmissions than a correct path traversal. Recently (June 2015), two patches were released in the development branch of the mac80211 kernel module, potentially addressing this issue [17], [18]. For now, we isolated the amount of SNMP data forwarded incorrectly by filtering all SNMP messages with unexpected mesh TTL. In contrast to the frame sequence number, the mesh TTL is only decremented after frame forwarding and not on retransmissions caused by collisions or bad frame integrity.

Figures 6 and 7 show the SNMP data sent by every node per query cycle in the second and third test case. All data sent correctly, including MAC-layer retransmissions, is given in blue bars, forming a descending step function. This corresponds to the expected multi-hop forwarding behavior, as introduced in Section III-A. In the centralized multi-hop setup (Figure 6), *Pi 1* forwards requests generated by the single Manager to all succeeding nodes in the path. The same accounts for all responses sent back to the Manager. Thus, nodes in greater distance to the Manager need to forward less messages. Figure 7 contains the data sent by each node in the distributed multi-hop scenario, after promotion of the second Manager. Again, the first node next to each Manager forwards most requests and responses. The amount of data sent incorrectly due to routing loops is shown in red bars for both test cases. This behavior mainly occurred on nodes in the middle of the multi-hop chain. Especially SNMP responses with a size of more than 1 kB circulated up to 30 times before mesh frame TTL ran out.

Figure 5 visualizes the overall benefit of AKadeMesh, applied in the third test case. A single Manager polling status information in a multi-hop chain (second test case) caused a total amount of 1206 kB SNMP data sent per query cycle. Dynamically assigning a second Manager and building two clusters, the resulting data size sent per query cycle was decreased to 512 kB. This corresponds to a 57 % reduction of monitoring traffic. To initialize the logical overlay, i.e., to perform dynamic Manager promotion, Agent association and logical address distribution (Kad bootstrapping), only 444 kB of Kad data were sent once, causing even less overhead than a monitoring cycle after decentralization.

V. Related Work

A survey on mesh network monitoring techniques is presented in [3]. Relevant work includes ANMP, representing an extended SNMP version. Contrary to our solution, clusters are not dynamically established. Moreover, ANMP has not been implemented and tested. Furthermore, Mesh-Mon is presented, which is a hierarchical overlay network for propagation of monitoring information. Problems with scalability in large-scale networks have been identified, which we address in our paper. In [4], an overview and analysis of P2P algorithms over MANETs are given. The work includes a categorization of algorithms and an analysis of challenging issues. The four categories comprise DHT-based, flooding-based, advertisement-based, and social-based methods. The conclusion states that one challenge is to minimize the consumption of network resources and to divide the burden of sharing data equally among the set of nodes by considering network topology. This is exactly the challenge that we address in this paper. The authors in [19] introduce a DHT-based Cluster Routing Protocol (DCRP), improving network performance by reduc-

ing the time for path selection and number of communication hops in large-scale WMNs. The concept is implemented as OLSR plugin, which is no longer applied by IEEE 802.11s, and thus represents an outdated approach. Preliminary simulation results only consider the clustering aspect but do not investigate the DHT-based service. The work in [20] pursues the localization of mobile users in WMNs without flooding to decrease latency and overhead, using a DHT-based routing scheme. DHT routing is inferred from the underlay resulting in one logical hop. Compared to our approach, the authors have a different focus. Only localization messages are routed whereby our application is optimized for the management of 802.11s networks. The motivation for the work in [21] consists in minimizing overhead transmissions for signalling purposes to make P2PSIP feasible in mobile sensing networks. An adaptive algorithm based on Kademlia and dSIP for the calculation of the refresh time is proposed. However, there is no topology-aware overlay as only maintenance intervals of the Kademlia network are adapted. To achieve an efficient file query by clustering peers by their common interests, the approach in [22] suggests a proximity-aware and interest-clustered P2P file sharing system (PAIS), based on the Cycloid P2P network. PAIS focuses on wired communication over the Internet. A node's Hilbert number serves as proximity index, calculated from the distance vector to a landmark. Contrary, we use ALM to derive proximity in 802.11s WMNs.

VI. Conclusion

In this paper we present AKadeMesh (Adaptive Kad-enhanced Mesh), a framework for decentralized management of 802.11s WLAN mesh networks. Combining a self-developed 802.11s Management Framework and the P2P protocol Kad, AKadeMesh provides an adaptive topology-aware network overlay for Manager decentralization and logical clustering, increasing scalability and leading to a reduced management overhead. We integrate the 802.11s standard's mandatory routing metric ALM into the Kad protocol to consider the physical mesh topology in the logical overlay. To the best of our knowledge, our cross-layer solution is the first approach to perform ALM-based mesh clustering. Thereby, no changes to the 802.11s standard are made and no additional proximity information are required. AKadeMesh further acts as middleware, granting efficient lookup of mesh status information, monitored and organized in the distributed management plane. Real-world evaluation in a 12 node test bed demonstrates practical feasibility and verifies the expected clustering benefit in a worst-case multi-hop scenario. In future research, we will perform large-scale evaluation both in a 40 node test bed and in a simulation environment, that are currently under preparation. Using this, we want to investigate strategies to determine the *Manager-Agent-Ratio* for different network sizes and further develop distributed cluster maintenance and Manager synchronization.

Acknowledgment

The authors would like to thank the German Research Foundation (DFG), Research Training Group 1424 (MuSAMA) for their financial support.

References

[1] "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 802.11-2012*, 2012.

[2] M. Rethfeldt, P. Danielis, G. Moritz, B. Konieczek, and D. Timmermann, "Design and Development of a Management Solution for Wireless Mesh Networks based on IEEE 802.11s," in *Integrated Network and Service Management (IM), 14th IFIP/IEEE Symposium on*, 2015.

[3] R. Pinto, "WMM: Wireless Mesh Monitoring," *Ad Hoc Network*, 2009.

[4] L. Liu, Y. Jing, Y. Zhang, and B. Xia1, "A Survey on P2P File Sharing Algorithms over MANETs," *Consumer Electronics Times*, 2013.

[5] R. C. Carrano, L. C. S. Magalhães, D. C. M. Saade, and C. V. N. Albuquerque, "IEEE 802.11s multihop MAC: A tutorial," *IEEE Communications Surveys & Tutorials*, 2011.

[6] S. Bari, F. Anwar, and M. Masud, "Performance study of hybrid wireless mesh protocol (HWMP) for IEEE 802.11s WLAN mesh networks," in *Computer and Communication Engineering (ICCCE), 2012 International Conference on*, 2012.

[7] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561, 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3561.txt

[8] R. G. Garroppo, S. Giordano, and L. Tavanti, "A joint experimental and simulation study of the IEEE 802.11s HWMP protocol and airtime link metric," *International Journal of Communication Systems*, 2012.

[9] "open80211s," http://open80211s.org/open80211s/. [Online]. Available: http://open80211s.org/open80211s/

[10] P. Danielis, V. Altmann, J. Skodzik, T. Wegner, A. Koerner, and D. Timmermann, "P-DONAS: A P2P-based Domain Name System in Access Networks," in *ACM Trans. Internet Technol.*, 2015.

[11] R. Brunner, "A Performance Evaluation of the Kad-Protocol," Master's thesis, Mannheim, Germany, 2006.

[12] D. Stutzbach and R. Rejaie, "Improving Lookup Performance Over a Widely-Deployed DHT," in *INFOCOM*, 2006.

[13] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *IPTPS*, 2002.

[14] R. Steinmetz and K. Wehrle, *P2P Systems and Applications, Springer Lecture Notes in Computer Science*. Springer-Verlag Berlin, 2005.

[15] K. S. Nagegowda, H. R. Ranganath, C. Puttamadappa, and T. G. Basavaraju, "Performance Evaluation of Scalable Routing Protocols using Routing Metrics for Wireless Mesh Networks under different Network Scenarios," *IRACST - International Journal of Computer Networks and Wireless Communications (IJCNWC)*, 2014.

[16] S. Islam, M. M. Alam, M. A. Hamid, and C. S. Hong, "High throughput path selection for IEEE 802.11s based Wireless Mesh Networks," in *4th International Conference on Uniquitous Information Management and Communication*, 2010.

[17] "mac80211 Patch: don't invalidate SN on discovery failure." [Online]. Available: http://comments.gmane.org/gmane.linux.kernel.wireless.general/139013

[18] "mac80211 Patch: don't special case routing to transmitter." [Online]. Available: http://www.spinics.net/lists/linux-wireless/msg138083.html

[19] M. Pinheiro, S. Sampaio, F. Vasques, and P. Souto, "A DHT-based approach for path selection and message forwarding in IEEE 802.11s industrial wireless mesh networks," in *14th IEEE international conference on Emerging technologies & factory automation*, 2009.

[20] M. Bezahaf, L. Iannone, M. de Amorim, and S. Fdida, "Transparent and Distributed Localization of Mobile Users in Wireless Mesh Networks," in *Quality of Service in Heterogeneous Networks*. Springer Berlin Heidelberg, 2009.

[21] P. Sendín-Raña, F. González-Castaño, F. Gómez-Cuba, R. Asorey-Cacheda, and J. Pousada-Carballo, "Improving management performance of P2PSIP for mobile sensing in wireless overlays," *Sensors*, 2013.

[22] H. Shen, G. Liu, and L. Ward, "A Proximity-Aware Interest-Clustered P2P File Sharing System," *Parallel and Distributed Systems, IEEE Transactions on*, 2015.