CHaChA: Clustering Heuristic and Channel Assignment for IEEE 802.11s Mesh Networks

Michael Rethfeldt, Benjamin Beichler, Peter Danielis, Tim Brockmann, Christian Haubelt, Dirk Timmermann
Institute of Applied Microelectronics and Computer Engineering, University of Rostock, Germany
michael.rethfeldt@uni-rostock.de

***Abstract*—WLAN mesh networks are one of the key technologies for upcoming smart city applications and characterized by a flexible and low-cost deployment. The amendment IEEE 802.11s (.11s) introduces low-level mesh interoperability at the WLAN MAC layer. However, scalability limitations imposed by management traffic overhead, routing delays, medium contention, and interference are common issues in wireless mesh networks and also apply to .11s networks. Possible solutions proposed in research recommend a divide-and-conquer scheme that partitions the network into clusters and forms smaller broadcast and collision domains by assigning different channels. We present *CHaChA*, a distributed cross-layer approach for clustering and channel assignment that directly integrates the default .11s mesh protocol information and operation modes, retaining unrestricted compliance to the standard. The practical performance and implied benefits of *CHaChA* are demonstrated in a real-world testbed.**

***Index Terms*—WLAN Mesh Network, IEEE 802.11s, HWMP, Airtime Link Metric, Clustering, Channel Assignment.**

I. Introduction

WLAN mesh networks are characterized by their flexible and fail-safe network topology. Featuring spontaneous interconnection and multi-hop forwarding they allow for low-cost increase of wireless coverage, e.g., in access networks, backbones, or service infrastructures in the urban sector and public facilities [1]. However, the current trend towards proprietary commodity solutions impedes the large-scale deployability of wireless mesh networks in prospective IoT scenarios. To overcome this vendor lock-in, it is required to leverage standard technologies that enable low-level interoperability. Since 2011, the IEEE 802.11s (.11s) amendment standard integrates mesh mechanisms directly into the WLAN MAC layer [2]. It is subject to ongoing research that aims at optimizing its scalability and interplay with existing higher-layer applications and protocols. One main challenge is the design of interoperable optimization approaches that directly integrate the default mechanisms and metrics provided by .11s, without introducing MAC layer modifications or depending on specialized hardware and unavailable protocol features.

Typical scalability limitations of wireless mesh networks arise from management traffic overhead, routing delays and error rates that increase with node count and path length. Devices within radio range and on the same channel form a collision domain and are subject to medium contention and co-channel interference. Subsequent frame forwarding steps are required during multi-hop data delivery, which are again subject to possible transmission errors and collisions. These limitations also impair the performance of applications operating in the wireless mesh network. As an example, we previously developed a centralized status monitoring framework for .11s networks [3]. However, real-world evaluation proved that decentralization is necessary to ensure its scalability and robustness [4].

Common approaches in research propose network partitioning with the election of distributed cluster heads, each being responsible only for a specific network region, assigned to a non-overlapping channel [5], [6]. Such a clustering and channel assignment approach allows for improved utilization of the available spatial and spectral resources and implies particular performance benefit for distributed applications when mapped to appropriate clusters. Nevertheless, to the best of our knowledge no specific solution for .11s exists that directly integrates its default mechanisms without further modifications. Particularly the .11s routing protocol HWMP and its metric ALM, available on every standard-compliant node, offer the potential to derive topology information to serve as basis for cluster formation.

Consequently, we present *CHaChA* (Clustering Heuristic and Channel Assignment). The resulting solution is used to perform distributed clustering and channel assignment in .11s networks, maintaining unrestrained interoperability to the mesh standard. We validate the practicability of our method in a real-world testbed and demonstrate its potential performance benefit by means of an exemplary mesh network monitoring application.

Summarized, the key contributions of *CHaChA* are:

- Clustering and channel assignment in .11s mesh networks
- Compliance to standard hardware and .11 protocol stack
- Integration of .11s standard mesh features and metrics
- Flexible underlay-aware application layer approach
- Scalable distributed and heuristical clustering
- Practical prototype and testbed evaluation

II. IEEE 802.11s WLAN Mesh Networks

As the first common industry WLAN mesh standard, the amendment .11s was ratified in September 2011 [2]. It enables vendor-independent infrastructure-less multi-hop communication based on the widespread .11 technology. Mesh functions like peering and routing are directly integrated into the MAC layer specification. Thus, .11s comes as promising alternative to former, non-interoperable network-layer mesh routing protocols.

To ensure interoperability, every .11s node must support the *Hybrid Wireless Mesh Protocol* (HWMP) and *Airtime Link Metric* (ALM) for mesh routing [7]. The default reactive mode of HWMP is based on the *Ad-Hoc On-Demand Distance Vector* (AODV) routing protocol and determines a path as soon as it is needed. Path information is updated periodically unless it expires with an inactivity timeout. Optionally, HWMP supports using a tree-based proactive routing mode alongside the reactive mode. Nodes that will be contacted frequently can use this mode to periodically announce themselves in the network and enforce determination of path information to them in advance. This reduces routing latency at the expense of additional message overhead [7].

Path information are maintained as forwarding rules. For each destination node, the address of the best neighbor is stored to which forwarding results in the smallest ALM. Each node only calculates the ALM to its neighbors and individual link costs are accumulated and disseminated during path discovery. Due to this cumulative characteristic, the ALM of a multi-hop path

represents the overall average estimated time to transmit a single frame from source to destination. The airtime cost c_a (in μs) is calculated per link as follows:

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

O is a constant for the channel access and MAC protocol overhead. B_t is the test frame size, specified as 1 kB in the standard. r denotes the test frame data rate, given in Mbps, whereas e_{fr} denotes the expected frame error rate. The estimation of e_{fr} and the value O are not predefined by the .11s standard but left open to vendor implementations.

The Linux kernel module *mac80211* currently contains the most sophisticated implementation of .11s and HWMP [8]. By default, path information expire after 5 s and are refreshed 1 s in advance of a timeout. In proactive mode, a node announces itself every 5 s via broadcast, triggering a path discovery or update on all other nodes. Since some parameters in ALM calculation are left open to vendor implementations, Linux provides own variants for error rate estimation and overhead constants. While O is set to the constant value 1, data rate r is estimated by the rate control algorithm (RCA). The error rate e_f is updated on every frame transmission and calculated by a moving average filter. Depending on the WLAN hardware and driver, this estimation is often provided by the RCA as well.

III. Related Work

The surveys [5], [6] provide an overview of different centralized and distributed clustering and channel assignment schemes. However, no approaches are discussed in these surveys that use at least a part of the .11s standard such as [9]–[11]. Besides the support and integration of .11s for unrestrained interoperability, we further aim at a distributed solution. Compared to centralized schemes, distributed approaches exhibit a higher scalability and robustness and are feasible even when no global network knowledge is available [5]. Lastly, we focus on the evaluation in a real-world testbed which validates practical applicability.

In Table I, we briefly summarize clustering and channel assignment works that are related to our approach and utilize standard .11. All works are grouped in the rows of the table by whether they also use standard .11s, present a distributed clustering approach, and whether the results were captured in a testbed.

In contrast to the related work, our approach *CHaChA* presents a distributed clustering solution that is fully compliant with standard .11 and .11s, directly integrates .11s MAC-layer protocol information, and is evaluated using a realistic testbed.

IV. CHaCha Concept

A. Overview & Terminology

The two main tasks of *CHaChA*, the *clustering* and *channel assignment*, are performed in a sequence of phases, as shown in Fig. 1. The design choice for operating in phases avoids the need for a precise time synchronization in the network. We assume all nodes to be equipped with at least two physical WLAN interfaces. The dedicated primary interface remains on a fixed default channel for unrestricted connectivity and inter-cluster communication, as proposed in other works [11], [13]. All control messages, maintenance operations, and metrics of our approach are exclusively retrieved and exchanged via the primary interface. After cluster formation, the secondary interface is activated and used for intra-cluster communication only.

TABLE II: *CHaChA* Node Roles

Node Role	Description
Cluster-Free Node (CFN)	initial role, not in a cluster yet
Cluster Member (CM)	member of a cluster
Cluster Head (CH)	leader of a cluster
Proposed CH (PCH)	temporary CH candidate
Master CH (MCH)	central CH and phase coordinator

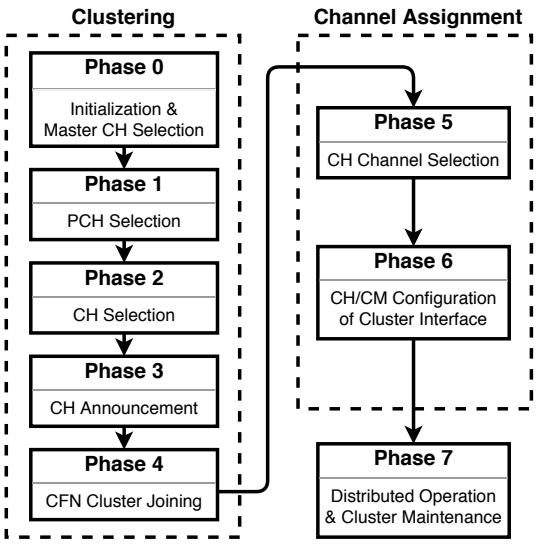


Fig. 1: Phases of the *CHaChA* Algorithm

Along the phase progression, nodes take different roles which are summarized in Table II. An initially dynamically selected master cluster head (MCH) is responsible for guiding the phase progression by announcing transitions via control messages. Phases 0 to 4 divide the network into non-overlapping clusters by considering the network topology. Each cluster represents a new mesh network on the secondary interface and comprises a leading cluster head (CH) and multiple cluster members (CMs). Clusters represent smaller broadcast domains in comparison to the default channel base network on the primary interface. This bears the potential to decrease network-wide load by mapping network applications to suitable clusters. If more than two interfaces are available, multiple applications in the same cluster could receive a different mapping. However, currently *CHaChA* does not prescribe the utilization of such additional interfaces. Phases 5 and 6 perform the channel assignment and configuration of the secondary interface. Spectral separation mitigates interference between adjacent clusters and the default channel and leverages parallel transmissions. A detailed phase description follows in Section IV-B.

A main goal of *CHaChA* is to be applicable with commodity hardware and an unmodified .11/.11s protocol stack for maximum interoperability and practical deployability. However, as a consequence of the .11s distance-vector routing protocol HWMP, every node has only incomplete knowledge of the network topology. Therefore, our approach implements a clustering heuristic, relying on the default .11s link and path information and metrics derived from it. This local MAC-layer knowledge is integrated into our application-layer algorithm. During the phases of *CHaChA* different metrics are used, which are listed in the following without particular order:

Airtime Link Metric (ALM): airtime cost of a frame transmission to a destination node, directly provided by the .11s standard routing protocol HWMP (see Section II). Due to its

TABLE I: Comparison of Related Work (*: Incomplete or Modified .11s without ALM)

Works	Standard .11s Utilization	Distributed Approach	Testbed Evaluation
CCAS [12], ISC [13], MCI [14], MCCA [15]	—	—	—
CCA [16], CoMTaC [17], DCITCA [18], Max-Min-D [19]	—	✓	—
CGCA [9]	*	—	—
Kapse et al. [10]	*	✓	—
JRCAP [11]	*	✓	✓

per-hop accumulation during path discovery, ALM denotes both the communication performance and path length between two nodes. In *CHaChA* it is used as distance measure, e.g., to let nodes join a proximate CH.

Centrality (CENT): estimate of the centrality of a node's position in the network. The node with highest centrality will become MCH. In a setup with homogeneous WLAN hardware and equal transmission parameters, the ALM cost of a mesh path is mainly affected by its length, unless individual link conditions manifest in a continuously high error rate. Therefore, it can be reasonably assumed that the most central node of an arbitrary mesh topology has the lowest mean ALM to other nodes as it communicates over the shortest path length on average. After having determined the ALM to all other nodes by (temporarily) activating the proactive mode of the .11s HWMP (see Section II), each node can calculate its centrality as $CENT = \frac{1}{\varnothing ALM}$.

Neighbor Count (NC): number of active .11s mesh connections. Nodes with maximum link count among their neighbors can reach the highest share of nodes directly and have the lowest probability of getting isolated in their respective network region. They are assumed proper CH candidates and will compete as "proposed" CHs (PCHs) until being sorted out further. The NC metric has also been proposed in many other research works under a different name, such as *Highest Connectivity (HC)* metric [13], [16].

PCH Neighbor Count (PCHNC): number of neighbors that are currently in the PCH role.

NC-to-PCHNC Ratio (NPR): ratio of NC and PCHNC. To obtain a value between 0 and 1, NPR is further normalized to the overall number of nodes, network size N, which is derived from the number of mesh paths that are known after a proactive HWMP discovery. In the progress of CH election, those PCHs with highest NPR will be given a higher chance to win because they occupy a neighborhood with less PCH redundancy. All PCHs calculate this metric as $NPR = \frac{NC}{(1+PCHNC)\cdot N}$.

Weighted NPR (WNPR): a node's NPR multiplied with its centrality $CENT$, normalized to the maximum centrality among all nodes, $CENT_{max}$ (centrality of the MCH). All PCHs compete based on this metric, calculated as $WNPR = NPR \cdot \frac{CENT}{CENT_{max}}$.

By introducing a centrality weight factor to the NPR, PCH competition finds a compromise between the sparseness of surrounding CH candidates and the proximity to the MCH in the network center. The motivation for this is two-fold: firstly, the centrality weight avoids plain NPR to overly prefer PCHs at the outward rims and corners of network regions, which are in general sparsely populated with CH candidates. Secondly, preserving a certain proximity to the central MCH implies benefit for distributed applications that perform inter-cluster communication and inter-CH synchronization.

Numeric Value of MAC Address: In a tie situation, i.e., if metrics of two nodes match exactly, the node with larger MAC address value wins the comparison. This supports fast and clear decision-making in several competition stages of our approach.

The control messages defined as part of the *CHaChA* protocol are given in Table III. They include unicast messages that are mainly exchanged between neighbors and broadcast messages that announce information to all nodes. Lastly, several timing parameters and thresholds are summarized in Table IV. The given example configuration corresponds to that used in our practical prototype evaluation, described later in Section V-B. A detailed description of the message and parameter utilization follows in Section IV-B.

B. Phase Description

In the following, we attribute the metrics, messages, and parameters of *CHaChA* to its different phases, shown in Fig. 1.

Phase 0 – Initialization & MCH Selection: All nodes start as CFN and are connected on the default channel via their primary interface. Firstly, a CFN checks for the reception of CH announcements for CH_THRESH multiples of CH_PERIOD. Reception indicates that the network was clustered previously and the CFN may directly join a CH. Otherwise, the CFN hooks into an already running later phase, if announced by a respective message, or continues in phase 0.

During phase 0, all CFNs periodically unicast NC messages every NC_PERIOD to their neighbors, for a later NC metric comparison in phase 1. Every CFN further temporarily activates

TABLE III: *CHaChA* Unicast (UC) and Broadcast (BC) Messages

Message Type	UC/BC	Description
CENT	BC	carries centrality metric
NC	UC	carries NC metric
WNPR	UC	carries WNPR metric
PCH	UC	PCH role announcement
CH	BC	CH role and cluster info announcement
JOIN	UC	join a new cluster
LEAVE	UC	leave a previous cluster
CHAN_SEL	UC	carries pairs of CH MAC and channel
PHASE_#	BC	phase no. # announcement

TABLE IV: *CHaChA* Timing Parameters and Thresholds

Parameter	Value	Description
CENT_PERIOD	500 ms	CENT message sending interval
CENT_THRESH	20	consecutive CENT messages threshold
NC_PERIOD	5 s	NC message sending interval
CH_PERIOD	5 s	CH message sending interval
CH_THRESH	4	no. of periods to wait for new CHs
PHASE_#_PERIOD	500 ms	PHASE_# message sending interval
PHASE_#_TRIES	20	PHASE_# message repetition count
PHASE_#_DELAY	10 s	delay before announcing phase no. #

the proactive mode of HWMP for a network-wide path discovery. As result, each CFN knows the current network size N as well as ALM costs to all other nodes. The CFNs now calculate their centrality and broadcast it via CENT messages every CENT_PERIOD to contend for the MCH role. Gradually, CFNs with smaller metric (or smaller MAC address on a tie) lose comparison and withdraw. Finally, the CFN with highest centrality becomes MCH after having sent CENT_THRESH consecutive CENT messages without receiving any.

An MCH in perfect central position induces only minimum mesh forwarding overhead when sending broadcast control messages. The MCH will act as CH of one of the resulting clusters and now handles the subsequent phase progression. To trigger transition to phase 1, it broadcasts PHASE_1 messages for PHASE_#_TRIES times every PHASE_#_PERIOD before advancing its own state. This procedure is analogous for phases 1–5.

Phase 1 – PCH Selection: All nodes compare their NC with that of their neighbors. The nodes with highest NC in their neighborhood become PCHs (a tie of metrics is permitted in this case). The PCHs announce their role to their neighbors via unicast PCH messages. After a PHASE_#_DELAY the MCH triggers transition to phase 2.

Phase 2 – CH Selection: All PCHs calculate their WNPR and unicast WNPR messages to their neighbors. Those PCHs with highest WNPR metric in their neighborhood (or highest MAC on a tie) become CHs. The remaining PCHs switch to the CFN role again. After a PHASE_#_DELAY the MCH triggers transition to phase 3.

Phase 3 – CH Announcement: All CFNs deactivate the proactive HWMP mode and operate in the more resource-saving reactive mode again. Only the CHs (including the MCH) keep this mode active to ensure that every node knows ALM costs to them at any time. Additionally, CHs and MCH now activate the periodic broadcast of CH messages to announce their role and cluster information to all nodes in the network. After a PHASE_#_DELAY the MCH triggers transition to phase 4.

The CH message also carries the cluster parameters which are used for configuration of the secondary interface. In phase 3, prior to initial cluster joining and channel selection, the message only carries the cluster's mesh SSID, generated from the CH's MAC address. Since .11s mesh routing is handled at the MAC layer, nodes may communicate over multiple hops using only their link-local IPv6 addresses, which are derived from MAC address information. Thus, we consider further concepts for IP configuration to be out of scope of *CHaChA*.

Phase 4 – CFN Cluster Joining: All CFNs determine their most proximate CH in terms of ALM cost with a preference of CHs in their neighborhood. Additionally, if the MCH is a neighbor, it will be prioritized over other neighboring CHs. This strategy is motivated by distributed application scenarios that perform frequent synchronization of cluster information with the MCH. This way, information exchange can be saved for all nodes that already belong to the MCH's cluster. For joining a cluster, CFNs send a unicast JOIN message to their respective CH. After a PHASE_#_DELAY the MCH triggers transition to phase 5.

Phase 5 – CH Channel Selection: In this phase, a greedy channel selection sequence is performed, using a predefined pool of pairwise orthogonal channels. The MCH starts the sequence and claims the first channel from the pool. It appends the pair of own MAC address and selected channel to the payload of a CHAN_SEL message and sends it to its closest CH, according to ALM. This procedure is repeated by the next CH until all CHs have claimed a channel. If the channel pool is used up, a CH re-uses the channel of the most faraway previous CH, according to ALM. This heuristic proactively mitigates interference between adjacent clusters. The last CH sends a message with all MAC-channel pairs back to the MCH, which triggers transition to phase 6.

Phase 6 – CH/CM Cluster Interface Configuration: Starting in phase 5, the CH message broadcasts also carry the cluster's channel and the current number of joined CMs. In phase 6, all nodes configure the secondary interface according to the parameters advertised for their respective cluster. CHs activate the proactive mode of HWMP on the cluster interface as well. As result, paths between CMs and their CH are always refreshed and node failures can be mutually detected by checking the local .11s path information. After cluster interface configuration, nodes directly switch to phase 7.

Phase 7 – Distributed Operation & Cluster Maintenance: Having reached phase 7, initial clustering of the network is complete. Distributed applications can benefit from increased performance due to parallel operation on non-overlapping channels with reduced broadcast domain size. Note that unrestrained inter-cluster network connectivity is always ensured via the primary interface, configured on the default channel.

While we consider strategies for coping with network dynamics, we have not yet practically evaluated *CHaChA* under dynamic topology changes. However, the described protocol mechanisms and integrated .11s HWMP information allow for continuous cluster maintenance. The information included in CH announcements can be considered by CMs or newly arriving CFNs as criteria to join and roam between clusters, closely following the strategy described in [11]. While an unsupported channel prevents a cluster from being chosen, more proximate CHs with fewer CMs have to be preferred for long-term cluster balancing. For this, CMs observe the ALM distance to all CHs as well as their current CM load. CH or CM node failure can be detected on a timeout of path information or a sudden absence of periodic CH announcements. In most cases, a failed CH is likely to be replaced by a CM of the same cluster. However, severe topology changes might also let CHs enforce a re-clustering of the network by triggering a transition to phase 0.

V. Experimental Validation

A. CHaChA Prototype and Testbed Setup

We implemented a Java prototype of *CHaChA* that runs on Linux .11s mesh nodes. It executes the described phase progression in a state machine. TCP is used for sending all unicast and UDP for all broadcast messages, respectively. To retrieve the mesh link and HWMP information on each node, we rely on existing operating system interfaces accessed via process calls. The Linux .11s implementation maintains *link list* and *path list* data structures on kernel level that can be inspected via the command-line utility *iw* [8].

We evaluated our *CHaChA* prototype in a practical testbed, comprising 25 Intel Galileo single-board computers. Based on our approach "Mini-Mesh" [20], using RF attenuators and reduced transmission power, we achieve a substantial reduction of transmission range. This allows us to create indoor multi-hop setups in a scale of approx. $1:560$. Our device configuration is given in Table V. Besides an on-board Ethernet interface, attached to a wired control network, each node is equipped with a dual-antenna .11n mPCIe card. All cards operate at a fixed transmission rate

TABLE V: Testbed Configuration

Parameter	Value
Device	Intel Galileo Board (Gen. 1)
CPU	Quark X1000 (Single-Core 400 MHz)
RAM	256 MB DDR3
OS	Debian 8 (Linux Kernel v4.9)
.11 NIC	Compex WLE200NX .11a/b/g/n (mPCIe)
NIC Chipset	Atheros AR9280 (ath9k driver)
Antennas	2 x 5 dBi Dual-Band Omni-Direct.
Attenuators	2 x Mini-Circuits VAT-30+ (30 dB)
Default Channel	149 (5745 MHz, HT20, Long GI)
TX Rate	MCS 3 (16-QAM 1/2, 26 Mbps)

and start on a default channel, not used otherwise in our institute building. While each node could be equipped with a secondary WLAN interface via USB, there are no products available that use the same chipset and support external RF attenuators. We therefore avoid a mixed hardware setup and analyze *CHaChA* and the performance benefit of multi-channel clusters by using only the primary interface, as described in the following sections.

We arranged our testbed in a regular 5x5-node grid setup, offering a network diameter of 4 hops and a symmetric center position. Fig. 2 shows the grid geometry.

After a performance analysis of *CHaChA* in Section V-B, the potential benefit of our approach is demonstrated by means of a network monitoring application in Section V-C. Here, a distributed monitoring approach, utilizing the cluster topology and CHs created by *CHaChA*, is compared to a centralized reference scenario, where a solitary MCH gathers the status data of all nodes. Fig. 2 marks the reference worst (node 1, red) and best case (node 13, green) choice for the position of the solitary MCH in terms of communication effort (average path length).

B. CHaChA Clustering Performance

We evaluated our *CHaChA* prototype in the 5x5-node grid base topology w.r.t. resulting cluster constellation, clustering duration, and induced traffic overhead. Algorithm parameters were set as given in Table IV. In each measurement run, *CHaChA* was started simultaneously on all devices of the unclustered base topology. A clustering run was denoted complete as soon as all nodes entered phase 7. By design, *CHaChA* communication is performed exclusively via the primary network interface that connects all nodes on a default channel. As indicated before, we spared a secondary interface in our evaluation. Therefore, all nodes executed phase 6 in a "dry" manner without actually applying the respective cluster configuration but only saving it for analysis. Besides log files generated by our prototype, each node recorded its self-sent traffic using the tool *tcpdump*.

We conducted a total of 50 measurement runs and averaged their results. Fig. 3 shows the different cluster constellations that occurred across all runs. Note that the clusters and their nodes

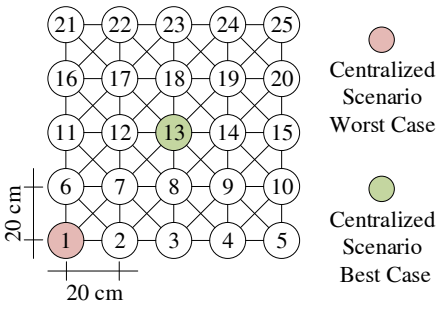


Fig. 2: Mesh Grid Setup and Centralized Reference Scenarios

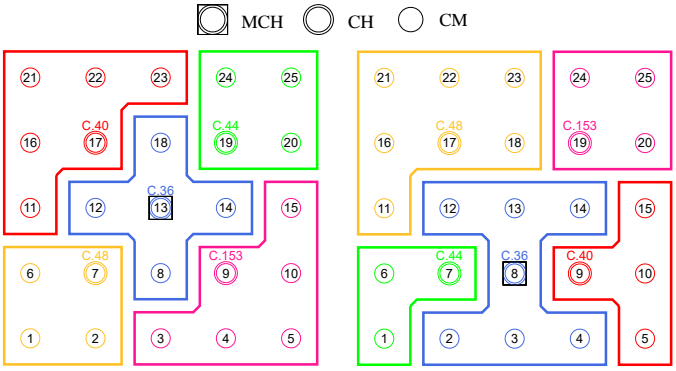


Fig. 3: Occurrence Rates of Cluster Constellations in 50 *CHaChA* Runs

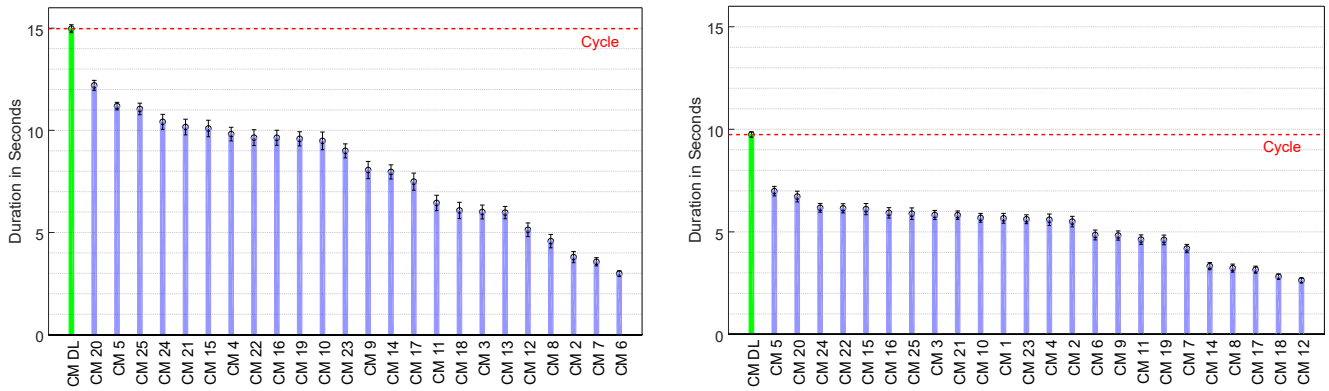
are highlighted in different colors, depending on the respective channel selected in phase 5. MCH, CH, and CM roles are marked as given in the top legend. All in all, only two different cluster constellations were formed by *CHaChA*. As expected for the static testbed setup, one variant was created predominantly, having an occurrence rate of 94 % (47 of 50 runs). Considering the phases described in Section IV, node 13 was chosen as MCH in phase 0 according to the centrality metric. In phase 1, the inner 3x3 grid nodes promoted as PCHs, all having a maximum NC metric (8 neighbors). Among these, corner nodes 7, 9, 17, and 19 expectably won the WNPR metric competition in phase 2, becoming final CHs. The neighborhood- and ALM-based joining of CMs in phase 4 already led to nearly balanced clusters. Prospectively, a CM roaming strategy will provide for full cluster balancing after reaching phase 7. Constellation 2 mainly differs in the position of the MCH, caused by temporary ALM fluctuations that affect the centrality metric in phase 0. The slightly different clusters are a direct result of the displaced MCH and the preference of CMs for joining the MCH's cluster in phase 4.

We further determined the average clustering duration and traffic overhead induced by *CHaChA*. For our very generous choice of default timing parameters (see Table IV), clustering took approx. 155 s. This was expected and closely corresponds to the given configuration. During phases 0–7, a total amount of 1.1 MB TCP and 630 kB UDP data were sent across the network, including forwarded multi-hop frames. TCP unicast traffic makes up the greater share due to its inherently higher protocol overhead compared to UDP. Consequently, unicast messages for metric exchange are kept local and are limited to a node's neighborhood in our approach. Additionally, all broadcast messages only originate from the MCH and CHs, except for CENT messages in phase 0. This design implies low long-term overhead after initial clustering as only these nodes will send periodic announcements. Apart from this, CMs may occasionally unicast LEAVE/JOIN messages if they decide to roam between clusters.

Generally, *CHaChA* parameters need to be chosen depending on the required robustness and overhead tolerance for an expected network size and transmission error probability. In future investigations we will analyze how small parameters can be set for different scenarios and how duration and traffic overhead are affected.

C. Example Application: Mesh Status Monitoring

A potential use case for *CHaChA* is the optimization of a municipal wireless mesh backbone or access network. Within this context, administrative tasks such as fault management and status monitoring need to scale accordingly in terms of performance and robustness. By means of an exemplary monitoring application, we demonstrate the benefit of clustering, as done by *CHaChA*.



(a) Worst Case: Node 1 as MCH

(b) Best Case: Node 13 as MCH

Fig. 4: Average Monitoring Cycle and Per-CM Download Duration for the Centralized Reference Scenarios

A distributed single- and multi-channel monitoring approach are compared to a centralized single-channel worst and best case. We consider the monitoring *cycle* time as performance metric, which denotes the duration of bulk retrieval of all CM status data and their aggregation at the MCH. In our experiments, all CMs provided exemplary fixed-size status information (file of 250 kB random data), hosted by a *ucspi-tcp tcpserver* instance. On the MCH (and the CHs in the distributed scenario) *netcat* was used as client-side application. A separate client process was started for every CM and all status data were requested simultaneously via TCP at the start of each cycle.

1) Centralized Monitoring of an Unclustered Network

In the centralized reference scenario, we distinguished a worst and best case, as marked in Fig. 2. In both cases, the network was considered as one cluster and all CM data were gathered by the solitary MCH. Node 1 in the network corner was set as worst case and node 13 in the center as best case MCH. As a simplified theoretical consideration, we sum up the minimum number of 1-hop transmissions required to deliver the status data of each CM to the MCH. We assume a fixed .11 data rate on all links and that one transmission is performed for each constant status data size. Note that this estimation neglects any client-side requests, error-related retransmissions, or TCP-specific effects. With node 1 as MCH, the number of 1-hop transmissions is 70. Contrary, with node 13 as MCH in the grid center, the sum is 40, a reduction of approx. 43 %.

We measured the individual download duration of every *netcat* instance on the MCH via the Linux *time* command in millisecond precision. Timestamps before start and after successful termination of all downloads were taken via the *date* command, respectively. The overall download duration was calculated as the timestamp difference between the last returned netcat instance and the beginning of the experiment and includes the scheduling impact for all *netcat* processes.

Fig. 4 shows the average cycle and per-CM download duration for the centralized worst (a) and best case (b). All results are averages of 50 monitoring runs and plotted with their standard error. The individual download times are shown as blue bars. The total CM data download duration "CM DL" is shown by the green bar. In the centralized scenario, this directly corresponds to the cycle time, which is highlighted further as red dashed horizontal line. With the fixed 26 Mbps transmission rate in our testbed and node 1 set as MCH, cycle time was approx. 15 s. With node 13 as MCH, a cycle time reduction of 35 % was observed, an 8 % deviation from our rough estimate (43 %). As expected, individual download times were predominantly shorter for proximate CMs in each case. The best case cycle time serves as reference benchmark for the distributed monitoring approach.

2) Distributed Monitoring of a Clustered Network

For the experiment in a distributed monitoring scenario, we used the *CHaChA*-generated cluster constellation 1, consisting of one MCH and four CHs. This constellation was evaluated in a single-channel (all clusters set to the same channel) and in a multi-channel setup, as given in Fig. 3.

As opposed to the centralized scenario, the monitoring cycle time was composed of two subsequent steps. In step 1, MCH and CHs concurrently gathered the CM status data within their cluster, which results in a per-cluster "CM DL" as in the centralized scenario. As soon as each CH finished downloading, it immediately uploaded its cluster information (CM data set complemented by the CH's own status data) to the MCH. Like in the centralized scenario, a cycle was denoted complete after all data were received by the MCH, i.e., a current snapshot of the global network status was obtained from an administrator's viewpoint.

As indicated before, we used only the primary mesh interface per node. Normally, two separate interfaces would provide simultaneous connection to a default channel and to a cluster. However, our experiment could be exceptionally run with one interface only as we performed the intra- and inter-cluster communication in sequence and the CHs were in 1-hop distance to the MCH. In the single-channel setup, interfaces of all nodes were configured on the same channel and logical separation of clusters was achieved by using different mesh SSIDs only. To enable cluster data upload in step 2, CHs temporarily changed their SSID to that of the MCH cluster. In the multi-channel setup, additional spectral separation of clusters was achieved by

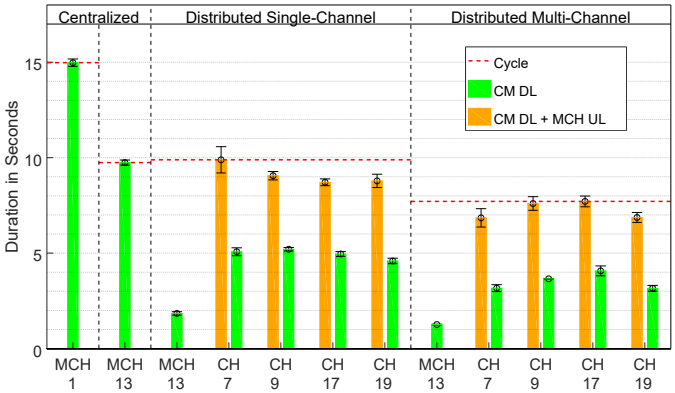


Fig. 5: Comparison of Centralized and Distributed Monitoring Performance

using different channels. Here, CHs temporarily changed both their SSID and channel to that of the MCH cluster in step 2.

Fig. 5 compares the cycle times of the distributed monitoring in single- and multi-channel cluster configuration with that of the centralized single-channel worst and best case. All results are averages of 50 measurement runs and shown with their standard errors. For the distributed scenarios, the per-cluster download step duration (CM DL, green) and the sum of download and upload step (CM DL + MCH UL, orange) are given. The distributed monitoring cycle time directly corresponds to the maximum DL+UL duration among all CHs. We relate the time differences between the clusters to the respective cluster sizes and unfairness effects of TCP. Since the MCH only performs step 1 and receives cluster information from the other CHs in step 2, Fig. 5 only shows the CM DL time for it.

Compared to the centralized worst case, distributed monitoring in the single-channel cluster configuration exhibited a cycle time reduction of approx. 34 %. This improvement is comparable to that of the centralized best case and can again be confirmed by our simplified theoretical estimation. Summing up all 1-hop transmissions in step 1 and 2, a minimum of 40 transmissions is needed, equal to the centralized best case. Moreover, parallel transmissions are limited here as in the centralized single-channel scenario. With all clusters set to the same channel in step 1, concurrent CM data download suffers from mutual interference in the edge regions of adjacent clusters. Note that in step 2 concurrent upload of cluster information to the MCH is subject to inevitable co-channel interference. Therefore, despite an inherent increase of robustness achieved by distributing the monitoring application, cycle time showed no improvement to the centralized best case yet.

Contrary, utilization of a different channel per cluster revealed a clear reduction of download time already for our medium-sized testbed. The improvement in step 1 directly corresponds to a 48 % and 21 % improvement of overall cycle time compared to the centralized worst and best case, respectively. Now using the full potential of clustering and channel assignment as provided by *CHaChA*, per-cluster CM data retrieval was performed in non-overlapping collision domains and drew benefit from increased concurrency. In a practical use case, even greater long-term improvement can be expected. Subsequent monitoring cycles can be interleaved, i.e., CHs may start a new intra-cluster download while still uploading data to the MCH on the default channel. Depending on application requirements, upload to the MCH may also be performed less frequently and cluster information may be preprocessed or compressed in advance.

VI. Conclusion

In the paper at hand, we introduce *CHaChA*, a scalable distributed approach for clustering and channel assignment in 802.11s (.11s) mesh networks. To the best of our knowledge, the presented solution is the first approach that integrates unmodified .11s HWMP and its routing metric ALM while no changes to the MAC layer are required. The default .11s information, retrievable via existing system interfaces on every node, are used to derive the clustering metrics for our approach. Additionally, we propose reasonable combination of the reactive and proactive HWMP mechanisms in a clustered mesh network.

We validate the applicability of our approach in a static 25-node real-world testbed. Results show that *CHaChA* forms a well-balanced initial cluster constellation at a 94 % reproducibility rate. With a very generous choice of timing parameters for the given network size, clustering takes approx. 2.5 minutes at a traffic overhead of circa 1 MB. We further demonstrate the benefit of multi-channel clusters using the example of a network monitoring application. A distributed approach, utilizing the clusters formed by *CHaChA*, is compared to the worst and best case of a corresponding centralized approach. Besides the inherent increase of failure resilience, average monitoring cycle time is reduced by more than 20 % w.r.t. the centralized best case, already for a 5x5-node setup. In future research we will analyze *CHaChA* in larger and more dynamic scenarios, exploring further base topologies and parameter choices. For this, we will use a combined emulation/simulation framework that we developed at our institute [21].

References

[1] I. Yaqoob, I. A. T. Hashem, Y. Mehmood, A. Gani, S. Mokhtar, and S. Guizani, "Enabling communication technologies for smart cities," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 112–120, 2017.

[2] "IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, 2012.

[3] 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 *IFIP/IEEE Int. Symposium on Integrated Network Management*. IFIP/IEEE, 2015, pp. 902–905.

[4] M. Rethfeldt, A. Wall, P. Danielis, B. Konieczek, and D. Timmermann, "AKadeMesh: Software-Defined Overlay Adaptation for the Management of IEEE 802.11s Networks," in *IEEE 13th Consumer Communications & Networking Conference*. IEEE, 2016, pp. 477–482.

[5] A. Alzubir, K. Abu, A. Yousif, and A. Abuobieda, "State of the Art, Channel Assignment Multi-Radio Multi-Channel in Wireless Mesh Network," *Int. Journal of Computer Applications*, vol. 37, no. 4, pp. 14–20, 2012.

[6] V. S. Kapse and U. N. Shrawanakar, "Survey of channel assignement schemes in wireless mesh network," in *3rd Int. Conference on Electronics Computer Technology*, vol. 3, 2011, pp. 103–107.

[7] S. Bari, F. Anwar, and M. Masud, "Performance study of hybrid wireless mesh protocol (HWMP) for IEEE 802.11s WLAN mesh networks," in *2012 ICCCE*. IEEE, 2012, pp. 712–716.

[8] open80211s, 2018. [Online]. Available: http://www.open80211s.org/

[9] A. Manikandan and Y. Palanichamy, "Optimized Group Channel Assignment Using Computational Geometry over Wireless Mesh Networks," *Mobile Information Systems*, vol. 2015, pp. 1–18, 2015.

[10] V. Kapse and U. Shrawankar, "Interference-aware channel assignment for maximizing throughput in WMN," *Int. Journal on AdHoc Networking Systems*, vol. 1, no. 1, 2011.

[11] S. Ghannay and S. M. Gammar, "Joint routing and channel assignment protocol for multi-radio multi-channel IEEE 802.11s mesh networks," in *4th Joint IFIP Wireless and Mobile Networking Conference*, 2011, pp. 1–8.

[12] A. Naveed and S. S. Kanhere, "Cluster-based channel assignment in multi-radio multi-channel wireless mesh networks," in *IEEE 34th Conference on Local Computer Networks*, 2009, pp. 53–60.

[13] N. Letor, C. Blondia, S. Bouckaert, I. Moerman, and P. Demeester, "A cluster driven channel assignment mechanism for wireless mesh networks," in *5th IEEE Int. Conference on Mobile Ad Hoc and Sensor Systems*, 2008, pp. 659–665.

[14] A. Brzezinski, G. Zussman, and E. Modiano, "Distributed Throughput Maximization in Wireless Mesh Networks via Pre-Partitioning," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1406–1419, 12 2008.

[15] S. Avallone and I. F. Akyildiz, "A Channel Assignment Algorithm for Multi-Radio Wireless Mesh Networks," in *16th Int. Conference on Computer Communications and Networks*, Aug 2007, pp. 1034–1039.

[16] S. A. Makram and M. Gunes, "Distributed channel assignment for multi-radio wireless mesh networks," in *IEEE Symposium on Computers and Communications*, July 2008, pp. 272–277.

[17] A. Naveed, S. S. Kanhere, and S. K. Jha, "Topology Control and Channel Assignment in Multi-Radio Multi-Channel Wireless Mesh Networks," in *IEEE Int. Conference on Mobile Adhoc and Sensor Systems*, 2007, pp. 1–9.

[18] B. Raju, K. Athota, and A. Negi, "A Distributed Cluster based Interference-Traffic aware CA for MRMC WMN," in *5th Int. Conference on Wireless Communication and Sensor Networks (WCSN)*, 2009, pp. 1–6.

[19] C. Liu, Z. Liu, Y. Liu, H. Zhao, T. Zhao, and W. Yan, "A Clustering-based Channel Assignment Algorithm and Routing Metric for Multi-channel Wireless Mesh Networks," in *5th Int. Conference on Parallel and Distributed Processing and Applications*, ser. ISPA'07. Springer, 2007, pp. 832–843.

[20] M. Rethfeldt, B. Beichler, H. Raddatz, F. Uster, P. Danielis, C. Haubelt, and D. Timmermann, "Mini-Mesh: Practical Assessment of a Miniaturized IEEE 802.11n/s Mesh Testbed," in *IEEE 16th Wireless Communications and Networking Conference*. IEEE, 2018, pp. 1–6.

[21] M. Rethfeldt, H. Raddatz, B. Beichler, B. Konieczek, D. Timmermann, C. Haubelt, and P. Danielis, "ViPMesh: A Virtual Prototyping Framework for IEEE 802.11s Wireless Mesh Networks," in *IEEE 12th Int. Conference on Wireless and Mobile Computing, Networking and Communications*. IEEE, 2016, pp. 1–7.