UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Distributed Channel Assignment in Multi-Radio Multi-Channel Wireless Mesh Networks

Igor Zavalyshyn

A thesis submitted in partial fulfilment of the requirements for the degree of European Master in Distributed Computing

Thesis Committee

Supervisor:	Leandro Navarro Moldes
Co-supervisor:	Emmanouil Dimogerontakis

July, 2015

Acknowledgements

My deepest gratitude goes to my supervisors Emmanouil Dimogerontakis and Leandro Navarro for their unending support and help in making this research possible. Manos has spent enormous amount of time with me generating outstanding ideas and suggesting elegant ways to solve the issues that kept on occurring while developing the algorithm and testing it later. I also want to thank him for being a good friend despite of me stealing most of his personal life. At the same time I want to give special thanks to Leandro for being a mentor who was always there to listen and give wise advises. This meant a lot to me.

On a more personal level, I would like to thank my EMDC groupmates for being with me during these 2 amazing years. Their bright minds and unbelievable skills motivated me to learn and try new things. I've enjoyed every minute spent with them during our classes or trips to other countries and I will definitely miss them when this master is over. I would also like to thank my parents for making it possible for me to be here.

Barcelona, July, 2015

Igor Zavalyshyn

"Every great change is preceded by chaos"

Deepak Chopra

Abstract

Wireless Mesh Networks (WMN) have become extremely ubiquitous recently due to their ease of deployment and low maintenance cost. However, because of their multi-hop nature, such networks are extremely sensible to radio interference. With every additional hop performance of the network in terms of its throughput and capacity as well as network latency degrades. Multi-Radio Multi-Channel (MR-MN) WMNs are currently used to minimize the impact of radio interference on network performance by utilizing different non-overlapping channels for each hop. Efficient Channel Assignment (CA) technique is therefore required to promote channel diversity while maintaining network connectivity. This is especially challenging task and is considered to be NP-hard problem. We present a novel distributed self-stabilizing algorithm that dynamically assigns channels to multi-radio nodes only based on local information. The algorithm uses clustering technique to group nearby nodes and select a cluster head responsible for channel selection. External and internal sources of interference are considered when selecting a channel for each cluster. To maintain the basic network connectivity one of the radios of each node is tuned to a default channel. We demonstrate the efficiency of our algorithm on a real-world 20-node testbed consisting of nodes, each equipped with an 802.11g and an 802.11a cards. We show that the proposed CA algorithm significantly improves the performance of the network in comparison to common channel assignment.

Categories and Keywords

Categories and Subject Descriptors

H.3.4 [Systems and Software]: Distributed Systems

Keywords

Wireless Mesh Networks

Multi-Radio Multi-Channel Wireless Mesh Networks

Multihop Ad Hoc Networks

Ad Hoc Networks

Channel Allocation

Intelligent Channel Assignment

Index

List of Figures

Acronyms

WMN Wireless Mesh Network

MR-MC Multi-Radio Multi-Channel

CA Channel Assignment

CH Cluster Head

1 Introduction

1.1 Motivation

Wireless Mesh Networks (WMNs) have become a popular way to provide wireless Internet access to remote rural areas where traditional Internet Service Providers (ISPs) are not available or have a limited coverage. The so called Community Networks (CNs) are created and managed by its users, unite people of a neighborhood, small village or even a town in their need for fast and inexpensive Internet access. In this case the network itself is based on the principle of a shared access, when people that have Internet connection allow others to use it free of charge. Cheap off-the-shelf routers are usually used to maintain connectivity, which makes the deployment of such networks easy and cost-effective.

The topology of WMNs provides high flexibility as each mesh router is connected to multiple others in an ad-hoc fashion. Usage of proactive routing protocols that explore these multiple links between any pair of mesh routers ensures robustness of the network. Thus in case of failure of any node, the whole network will continue to operate in a normal mode.

However, the performance of the whole network in terms of its throughput and capacity depends on several important parameters: distance between the nodes, antenna type and its gain, frequency spectrum used, interference, channel diversity to name a few. While former two mentioned parameters are hard to control considering the unplanned deployment of nodes in most of the CNs, latter ones may significantly alter the performance of the network in case of careful selection and network tuning.

Interference problem is believed to be the crucial one and has been widely studied within the WMN environment. The idea behind it is as following: nearby wireless links interfere with each other if there are simultaneous transmissions in them. Moreover, in multihop WMN environment nearby hops of the same link may interfere as well. This in fact lowers the performance of the network dramatically. It gets even worse when the network scales and more sources of interference appear.

Various solutions have been proposed to reduce the impact of interference in WMN. Some

of them propose to switch channels used for each transmission. Other ones propose to use multiple radios on each router tuned to different channels or even different frequency bands. However, the number of radios available on commodity wireless devices and the number of available non-overlapping channels within the non-licenced spectrum is limited. Such limitation requires careful Channel Assignment (CA) technique that would ensure the connectivity of the network and decrease the impact of interference.

In this thesis we present a novel algorithm for efficient Channel Assignment in Multi-Radio Multi-Channel WMNs. The algorithm considers internal and external sources of interference while selecting a channel for each router's radio. For more efficient CA routers are grouped in clusters based on the physical proximity and signal strength. This allows to ensure high level of connectivity and minimum number of interference sources.

1.2 Research History

The initial step of this thesis was to study the characteristics and design principles of WMNs as well as existing problem areas. We have also performed a brief overview of possible solutions to this problems. Majority of previous research efforts were focused on channel assignment and routing techniques since this area is the most challenging. Moreover, new approaches are regularly proposed based on the latest software and hardware updates. We therefore had to get familiar with the latest techniques and implement a working prototype of a system. We designed a CA algorithm that is based on clustering technique and proactive routing protocol able to handle multi-radio multi-channel environment. After that we performed an evaluation of our algorithm at a WMN testbed.

1.3 Structure of the Document

The rest of this document is organized as follows. Chapter 2 provides an overview of ideas and techniques related to this work. Chapter 3 lists the challenges we have faced while working on this thesis. It also provides a summary of the related research. In chapter 4 we provide an overview of our developed algorithm for efficient channel assignment in WMN. In chapter 5 we describe the performance evaluation of it and discuss the obtained results. Finally, chapter 6 concludes this document by summarizing its main points and future work.

2 Background

2.1 Introduction

In this chapter we provide some theoretical background about ideas and techniques that are used in an effort to reach our goal. In the following section we introduce Wireless Mesh Networks, their architecture and characteristics.

In section §2.3 we introduce the idea of clustering in Wireless Mesh Networks. In section §2.4 we overview common techniques for channel assignment in such networks.

2.2 Wireless Mesh Networks

Wireless Mesh Networks (WMN) have become a promising solution to extend the wireless coverage in flexible and cost-effective way. These networks proved to be very efficient in expanding network connectivity in rural areas where access to traditional ISPs is limited or not available at all [1]. WMNs became popular because of their relatively low entry and maintenance cost, robustness, reliable service coverage and ease of deployment. They have broad applications in Internet access, community or emergency networks, public safety, and so forth.

Akyildiz and Wang, in [2], describe WMNs as dynamically self-organized and self-configured wireless networks, where nodes establish multihop ad-hoc mesh connectivity. The authors present figure 2.1 as a hybrid WMN, where mesh clients can access a WMN through other mesh clients.

The architecture of WMN may be divided into three levels:

- Top layer consists of gateway nodes that are directly connected to Internet providing Internet access to the rest of the mesh network and connected clients.
- Middle layer consists of multiple mesh routers, also called relays. They communicate with gateways and between each other forming a backbone of the network that is used to forward clients traffic through multiple wireless links.

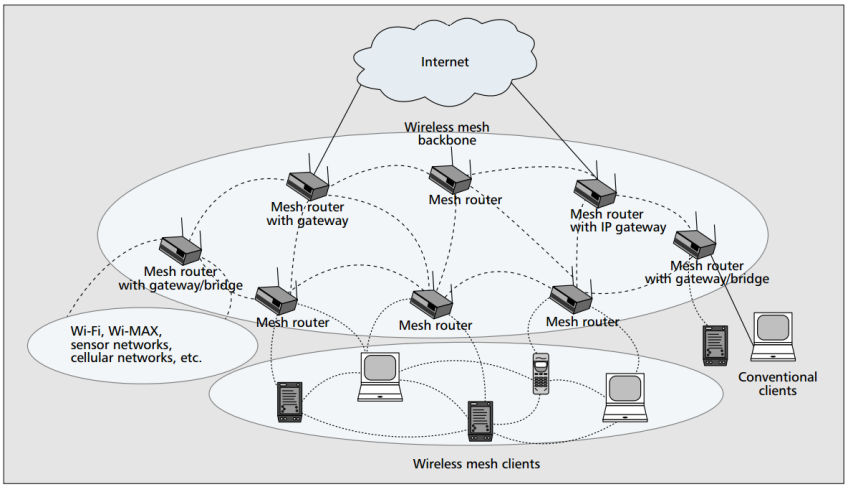

Figure 2.1: Typical WMN architecture.

- Bottom layer consists of wirelessly connected client devices (e.g. desktops, laptops, smartphones, etc.) or even WLANs. These clients are not aware of entire mesh network and do not participate in the routing process, but use wireless mesh routers as access points.

Nodes in WMN are usually static or have limited mobility. Thus wireless links of a backbone are also static. On the other hand, clients devices may be static or mobile. Clients mobility is supported by default given the wireless type of access.

Each node in WMN acts as a host and as a relay, forwarding packets to and from other nodes. Connectivity is established and maintained automatically in an ad-hoc manner using standard IEEE 802.11[1] stack.

Traffic loading may not be evenly distributed and sometimes tends to be gateway-oriented, for instance in WMNs that provide Internet access. Therefore links connecting gateways to their closest mesh routers are expected to receive more traffic than others. This should be properly handled by the routing protocol.

However, multihop ad-hoc environment cannot be handled efficiently by the standard IEEE 802.11[2] stack or ad-hoc protocols. Thus, many wireless protocols are revised to support WMNs and new WMN-specific protocols are created. Among them BMX6[3], B.A.T.M.A.N advanced[4]

[1]IEEE 802.11 http://standards.ieee.org/about/get/802/802.11.html
[2]IEE 802.11 http://standards.ieee.org/about/get/802/802.11.html
[3]BMX6 http://bmx6.net/
[4]B.A.T.M.A.N advanced http://www.open-mesh.org/projects/batman-adv/wiki

and IEEE 802.11s[5] are some of the most promising. This protocols are based on the idea of multipath routing that is aimed to balance traffic between multiple existing paths to the gateway. Moreover, such protocols make network robust to node failures. Therefore, if any node fails, others will still stay connected.

2.3 Clustering Algorithms

According to [3], clustering plays a vital role in improving network performance parameters like routing delay, bandwidth consumption and throughput. The whole clustering process may be explained as dividing the network into virtual sub-groups, called clusters. Each cluster consists of one cluster head, a number of cluster members and several border nodes that connect clusters with each other.

There are two types of cluster control architectures, specifically one-hop clustering and multi-hop (d-hop) clustering. In one-hop clustering each member is at most one-hop away from the cluster head (CH) and at most two-hops away from other cluster members. In multi-hop clustering members may be d-hops away from each other but still maintain an active cluster. In our work we concentrate on one-hop clustering technique due to lower convergence time, smaller traffic overhead and, what is more important, smaller cluster size that minimizes the impact of interference between cluster members.

Cluster head acts as a local coordinator within its cluster and manages available resources. In our work CH is responsible for selecting a channel to work on within a cluster and identifying the border nodes that will establish connections to neighboring clusters. Cluster heads therefore do not need to have knowledge of the whole network, using only the knowledge of part of it within their close proximity instead. This allows to increase the scalability of such approach.

Many different cluster head selection algorithms have been reported [4]. Most of them consider various attributes of the node to decide its role in a cluster, for instance, identification number (node ID), number of neighbours it has (node degree) or weight of a node that is calculated based on specific metric. In case of Wireless Sensor Networks, where lifetime of a node is limited, the remnant energy is sometimes used as a metric to select a cluster head to prolong the uptime of the network.

Since each head selection algorithm is optimised for a specific type of wireless network

[5]IEEE 802.11s http://standards.ieee.org/findstds/standard/802.11s-2011.html

and scenario, selecting the right one is especially important. Clustering schemes developed for networks with mobile nodes, such as MANETs, will not be suitable for mostly stable WMNs. Selected algorithm should consider nodes heterogeneity and quality of wireless links between them, as well as minimizing the number of created clusters and their impact on each other in terms of interference.

Among all proposed head selection algorithms there are two well-known. The first one is Lowest ID (LID) Algorithm [5]. This algorithm requires each node to have a unique ID that is used to define role of a node in a cluster. Every node broadcasts its ID to one-hop neighbors. Knowing IDs of its direct neighbors and it's own ID, node makes a decision whether to become a Cluster Head or join some other node's cluster. Node with the lowest ID considers itself a cluster head and broadcasts its status to neighbors. This algorithm guarantees that no Cluster heads will be next to each other and that all the network will be eventually clustered. It doesn't, however, minimize the number of created clusters.

Second well-known algorithm is pretty similar to the first one, but instead of using ID of the node, its degree (number of direct neighbors) is used to determine the role. Highest Connectivity Algorithm (HC) [6] tries to reduce the number of clusters it creates by selecting nodes with the highest degree as cluster heads. While minimizing the number of created clusters, this algorithm suffers from a decreased throughput. In case of WSN cluster heads will have more members to serve thus reducing the available throughput and system overall performance. In MANETs as mobility of the node changes, it may cause frequent head re-elections. Such changes may result in a situation, when change of one cluster head may cause a chain of changes in other clusters. However, in case of WMN where each mesh router can route packets from and to any other router despite of being a cluster head or a normal node, and mobility of nodes is limited, mentioned problems are not present.

In [7] Das, Mukherjee and Turgut have offered an algorithm that uses weight of the node to select cluster heads. For instance, in WSN a combined weight of the node may be calculated based on the battery power level, node degree and other metrics. Such approach, however, yields the communication overhead and causes frequent cluster head reelections. When re-election takes place the combined weight of every node needs to be calculated again thus resulting in higher computational cost.

All mentioned algorithms have their advantages and disadvantages. Combining best ideas from all of them allows to come up with the algorithm that suits WMN environment best. First

of all such algorithm should ensure that no CHs are direct neighbors of each other.The number of created clusters or the number of members in each of them should be minimized. Only one-hop neighbors of the cluster head should be members of its cluster. This allows to decrease the impact of interference between the neighbor nodes. Finally, cluster heads should be selected based on the metric that doesn't change frequently. In case of static WMN that have already been deployed and remains stable selected cluster heads should stay active for a long period of time.

2.4 Channel Assignment

Initially, WMNs were designed in a way that each node worked on the same channel and was equipped with a single radio [8],[9]. Such approach, despite of being simple and straightforward, significantly affects the capacity of the network and, what is more important, its ability to scale. Moreover, simultaneous transmissions in the nearby wireless links cause them to interfere with each other. According to [10], in a single-channel multi-hop scenario interference occurs not only between nearby flows (so-called inter-flow interference) but also between nearby hops in a same flow (intra-flow interference).

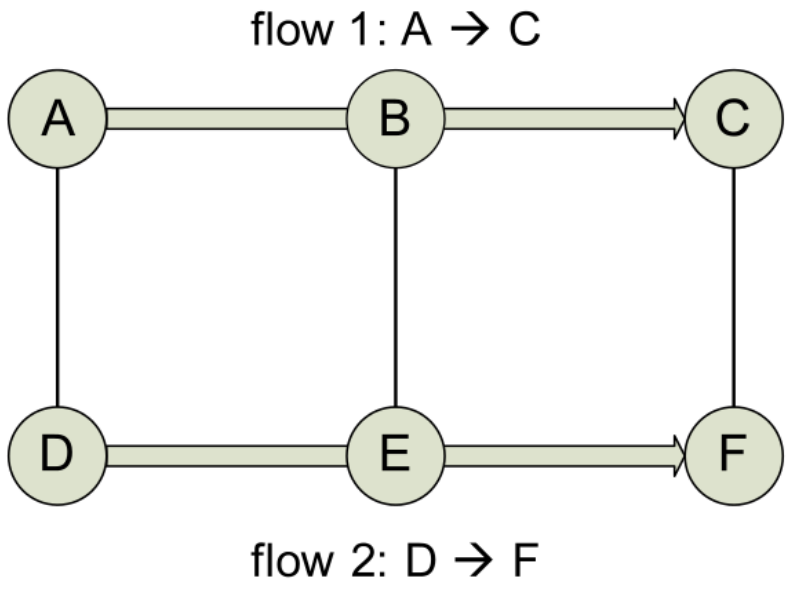

Figure 2.2: Inter-flow and intra-flow interference.

Figure 2.2 illustrates the above problem. Two concurrent flows Flow 1 and Flow 2 interfere

with each other while using neighboring links. At the same time hop A → B interfere with hop B → C within the same flow.

Several solutions have been offered to address this interference problem. Some of them were based on a frequent channel switching [11],[12]. Such approach takes advantage of a having multiple spectrum channels available, but causes significant delays in transmissions. According to [13] currently existing 802.11 hardware introduces 25 to 30 ms delay while switching to a new channel. Moreover, frequent channel switching causes low connectivity between nodes or even network partitioning. All nodes have to be synchronized in order to stay connected with their neighbors before and after channel switching. In practice this becomes a challenging task.

Alternative approaches are based on modified medium access control (MAC) or require the use of directional antennas. Unfortunately, deployment of such schemes becomes impractical on a wide scale and may be quite expensive.

As a result most of the current WMN use multi-radio multi-channel (MR-MC) architecture. Each node is equipped with several radios that can simultaneously work on different channels. There are three non-overlapping channels defined by IEEE 802.11bg [14] standard and twelve non-overlapping channels in IEEE 802.11a [15]. Using multiple radios configured to different non-overlapping channels or even different frequency bands, leads to more efficient spectrum utilization, increased throughput and capacity of the network.

However, in order to achieve the maximum possible network performance, efficient channel assignment (CA) technique is a key. We now present a classification of common CA techniques and schemes used in WMN and discuss their relative strengths and weaknesses.

2.4.1 Channel Assignment Approaches

Centralized Channel Assignment

In a centralized approach channel assignment is performed by a central component that has a global knowledge of the network. This component is usually called Channel Assignment Server (CAS). It may be a standalone node in the system or a part of existing node, for instance, the gateway. CAS collects information from all the nodes in WMN and creates a virtual graph of the network. It then visits each virtual link of a graph and assigns a channel according to a specific algorithm. Expected traffic load, interference level or available bandwidth may be selected as metrics for links ranking. Links that are expected to have higher traffic load, for instance links to the gateways, are assigned a higher rank and get a channel assigned first. When all the

virtual links are visited and have a channel assigned, CAS sends results to the nodes. Nodes then configure their radios to assigned channels.

Having a central component in the network reveals the limitations of such approach. If central component is disconnected from the network, the whole CA process is stopped. Moreover, such aproach doesn't scale with the size of the network. To address these issues distributed algorithms for CA have been proposed.

Distributed Channel Assignment

In a distributed CA, every node selects a channel to work on solely based on the local information. There is no need for central component as each node runs the same algorithm. Such approach is usually more complex and requires a lot of communication between the nodes. Decision may be based on current interference level or channel performance. It's extremely important to consider ways to avoid the so-called ripple effect [16] which may lead to endless channel changes. For instance, channel selection may be based on nodes' unique ID. So nodes with lower or higher IDs have priority while selecting the channel. In some cases such priority may be given to gateways, since in WMN traffic usually flows from or to the gateway.

2.4.2 Channel Assignment Schemes

Both centralized and distributed CA techniques may be further divided into three different categories depending on the frequency of changes made to the channel assignment [17]. These categories are static, dynamic and hybrid channel assignment. We now briefly review all of them.

Static Channel Assignment

Static CA schemes assign channels to interfaces permanently or for a long time intervals. Such schemes may be further divided into common channel and different channel assignment.

In common channel assignment all nodes get assigned the same set of channels. This significantly simplifies the process of CA and improves network connectivity. However, such approach doesn't consider the changes in the wireless environment, making it sensitive to interference from direct neighbors and external networks. Moreover, same channels used by each node often get overloaded in parts of the network where the traffic load is high, consequently affecting the network throughput.

Different channel assignment assumes varying sets of channels assigned to nodes interfaces. In this case there are less chances for nodes to interfere with their direct neighbors, but at the same time such approach may lead to network partitioning. Since assigned channels differ from node to node, despite of being in the direct range of each other nodes may not share any common channel. This approach requires careful and usually complex channel selection.

Dynamic Channel Assignment

In this approach, nodes are assigned different sets of channels but dynamically switch their interfaces from one channel to another to communicate with other nodes. Such scheme requires nodes to negotiate about the common channel to use for transmission, using predefined so-called rendezvous channels. However, strict synchronization between nodes is needed to reduce delays in transmission. Moreover, additional delays are introduced by frequent channel switching. This makes such approach infeasible for multi radio networks, but still efficient in single radio networks.

Hybrid Channel Assignment

Hybrid CA combines advantages of both previous approaches. One of the interfaces is assigned a static channel while other interfaces change their channels dynamically. This allows to simplify the coordination phase since nodes may always communicate with each other via default interface with a static channel. Moreover, such approach ensures network connectivity which is important in case of frequent channel switching.

3 Challenges & Related Work

3.1 Introduction

In this chapter we present the main challenges we have faced while designing a system as described in §1.1 and how some of them have been addressed in related research. The following section explains the challenges of channel assignment in WMN environment. In section §3.3 we present a brief state of the art with common CA approaches aiming to solve some of the above issues. Most of the mentioned issues are, in fact, closely related with each other and the task of CA in this case is to find a trade-off relationship between them.

3.2 Challenges

Efficient CA in MR-MC WMNs is a challenging task. The whole process has to address multiple issues including the ones related to the nature of WMNs environment. In this section we present the most relevant challenges that we faced in designing and implementing our algorithm.

Challenge 1: Hardware Heterogeneity

One of the main advantages of WMNs is support for heterogeneous devices. Wireless routers equipped with a single or multiple radios may exist and successfully communicate with each other in WMN. CA is therefore required to consider specific features of each device in the network.

Challenge 2: Wireless Links Instability

Since WMNs are usually deployed in large and heterogeneous areas, wireless links often experience performance degradation and quality fluctuations [18]. There are multiple factors like distance between wireless routers, environment obstacles, external wireless networks, that affect the quality of the links. This issue is usually handled by advanced routing protocols that explore multipath routing. Routing protocol itself should adapt to ongoing changes and create new links based on the current topology.

Challenge 3: Wireless Media Usage

WMNs usually use unlicensed wireless spectrum which is shared with other external networks and devices. Selected channel should be a least-used channel among neighbor nodes and external networks. CA should also adapt to frequent changes in channel performance, meaning that if a channel becomes overutilized, new channel should be selected.

Challenge 4: Node Wireless Resources

Although there are several non-overlapping channels available, the number of radios on each node is limited. This affects the number of channels that may be simultaneously used by each node.

Challenge 5: Network Connectivity

Neighbor nodes that want to communicate have to share a common channel in order to establish a wireless link. However, CA affects the topology of the network. Nodes that are in the direct range of each other may become disconnected after switching to assigned channels. CA has to maintain network connectivity and prioritize links between nodes that are physically close to each other.

Challenge 6: Shared Channel Usage

Number of neighbor nodes sharing a common channel should be minimized in order to decrease the impact of interference. This is especially challenging task since network connectivity has to be still maintained.

Challenge 7: Interference Influence

As it was discussed before CA has to consider inter- and intra-flow interference. Moreover, external wireless devices working on the same channels may interfere and significantly lower the performance of the network. CA technique should consider all sources of interference while selecting a channel for each node.

Challenge 8: Coordination and Synchronization

Coordination mechanism is needed to maintain connectivity between the nodes wishing to communicate. Nodes may regularly switch to a common channel to negotiate on a channel to use.

At the same time, dynamic CA with frequent channel switching requires tight synchronization between the nodes in the network.

Challenge 9: Ripple Effect & Network Convergence

CA should not cause a ripple effect, when change of active channel on one node causes a chain of changes on other nodes. Additionally, CA should produce a stable channel output meaning no need for further channels change. Therefore, dynamic metrics (for instance current channel utilization) should not be used for CA.

Challenge 10: Cluster Creation

Nodes should automatically organize in clusters and have a mechanism to select a cluster head based on some static parameter, for instance node's ID or weight. Clusters should be as small as possible to minimize the number of nodes sharing the same channel. Additionally, clustering algorithm has to handle node failures efficiently. If cluster head fails, cluster members should select a new one.

Challenge 11: NP-Hardness

The whole process of channel assignment is aimed to find a trade-off between minimum interference and maximum connectivity. This is especially challenging task and is usually described as NP-hard problem.

3.3 Related Work

Up to now, numerous CA approaches for MR-MC WMNs have been proposed [10]. In this section we briefly describe and compare the most promising of them. We start from centralized approaches and continue with the distributed ones. For each approach we explain the basic idea, discuss advantages and limitations, and, finally, compare with our solution.

3.3.1 Centralized Channel Assignment

Ramachandran et al. in [19] present an interference-aware algorithm with dynamic CA (BFS-CA). Similar to our approach authors propose to use a common channel to prevent network partitioning. This channel is also used for traffic redirection when other channels are

reconfigured. Nodes periodically measure the interference and utilization levels for each channel and report this data to CAS that is co-located with gateway node. CAS then assigns channels to each node using Multi-Radio Conflict Graph (MCG). Despite of having lots in common with our approach, described algorithm may become slow and resource consuming if the size of the network increases. Moreover, this algorithm does not provide any protection from the ripple effect.

Mahesh and Das in [20] present a novel algorithm called CLICA aiming to minimize maximum interference while maintaining network connectivity. Described algorithm uses conflict graph to model the interference between all possible transmissions in the network. Additional connectivity graph is created based on the network topology to preserve connectivity in a process of CA. Like many other centralized approaches, described one assigns channels to each edge of the graph using graph-coloring technique. However, such technique is proved to be not suitable for MR-MC WMNs [10] and requires additional steps to model the network properly. As mentioned by authors themselves, proposed solution does not scale and only supports small-size networks.

The authors of [21] propose ITACA - a hybrid interference and traffic aware CA scheme aimed to increase multi-hop path performance in WMN with gateway-oriented traffic pattern (e.g. used for Internet access). Channels are assigned starting from the gateway and giving priority to those links that require more bandwidth. Similar to previous approaches, every node configures one of it's interfaces to a common channel and uses it for communication and traffic redirection. Traffic redirection is triggered upon receiving ARP update message from a node that reconfigures non-default interfaces. Although proposed algorithm resulted in 111% performance increase during experiments, we argue that described traffic redirection technique is optimal. Frequent changes of ARP table may introduce significant delays. Moreover, this method dramatically increases communication overhead.

Raniwala et al. in [22] propose two novel load-aware channel assignment (LA-CA) and bandwidth allocation algorithms for MR-MC WMNs. The whole process of CA consists of two steps. At first, channels are assigned based on the network topology using Neighbor Partition Scheme (NPS). Then the algorithm output is further improved by exploiting traffic load information and giving priority to the links that carry most of the traffic. However, if the traffic load changes over the time, channels have to be re-assigned, causing a chain of changes at other links. Moreover, this algorithm as well as some of those mentioned earlier, assume having a

priori knowledge of traffic load distribution, which is not always possible. As a result, network has to work in idle mode for some time until the algorithm has enough data for efficient CA.

3.3.2 Distributed Channel Assignment

In [23] Anjum Naveed et al. introduce CoMTaC - a cluster-based approach that is quite similar to ours. The algorithm consists of two phases. The first phase groups nodes in clusters of small size with a cluster head responsible for channel selection. Second phase of the algorithm focuses on the channel assignment based on the created topology. Initially gateways act as cluster heads and group nodes that are k hops away from them. All the nodes that are out of cluster's radius create their own clusters following the same logic. Each node within a cluster configures one of the interfaces to the default channel, that is different for each cluster. Nodes on the border of cluster, so called border nodes, configure their second interface to the default channel of one of the neighboring clusters. Such mechanism ensures inter-cluster connectivity. To select a channel for the default interface external sources of interference are considered. To select a channel for the non-default interfaces, nodes analyze average link layer queue length as representation of interference level. However, grouping nodes in clusters based on the hop count may be inefficient if nodes are randomly distributed in the network. In this case such situation may occur when nodes are at the same cluster and share a default channel, but have poor connectivity with each other. We propose to use signal level instead as a better representation of nodes proximity.

Bong-Jun Ko et al. in [24] propose a self-stabilizing algorithm (SS-CA) that continuously tries to improve current channel selection. Like in previous approaches, common channel is used for providing initial connectivity and for communication with neighbors. To select a channel for additional interfaces authors propose to use interference cost function. This function compares the spectral distance between node's current channel and channels used by any of it's neighbors. As a result, nodes select channels that minimize the sum of interference costs within their interference range. The algorithm uses three-way handshake to prevent neighboring nodes from switching their channels simultaneously. Proposed approach suffers from the same problem as CoMTaC. Interference cost function doesn't consider location of nodes and results are solely based on channels they use. This may lead to situation when network is connected but links quality is poor.

Authors of [16] propose a fully distributed algorithm called Hyacinth, that uses only local

topology and traffic load information for channel assignment. For efficient load balancing algorithm creates a fat tree topology where gateways serve as root nodes. Each node then binds part of available interfaces to parent nodes (UP-NICs), and the other part to it's children (DOWN-NIC). Channels are selected only for DOWN-NICs. Channels for UP-NICs are assigned by a parent node. Selected channels are least used channels according to interference level and total load within the interference range of each node. One of the advantages of Hyacinth approach is simple protection from the ripple effect, since nodes are only responsible for channels assigned to their DOWN-NICs. However, the whole process of topology discovery and further communication is time consuming and may become a bottleneck when network scales.

Channel switching technique is introduced in [25]. Authors propose probabilistic channel usage based algorithm (PCU-CA). Nodes have two types of interfaces: fixed that stay on the same channel for a long time, and switchable that are constantly reconfigured to the channel used by fixed interfaces of their neighbors. Authors also propose a special routing protocol that selects channel diverse paths and considers channel switching cost. Such approach allows to utilize all the available channels as well as protect from the ripple effect. However, frequent channel switching introduces significant delays and may lower the performance of the network. Moreover, algorithm does not consider traffic load distribution.

A simple yet elegant algorithm called ROMA is proposed in [26]. It is designed to optimize the gateway paths in WMNs that provide Internet access. According to the proposed approach, each gateway produces a sequence of non-overlapping channels, e.g. c_1, c_2, ..., and sends it to all the neighboring nodes. Every node that is i hops away from the gateway selects c_i and c_{i+1} channels from the sequence and configures available radios to them. The process is repeated until all the nodes complete the procedure. Nodes periodically run route discovery process to find and compare different paths to the gateway. Despite of being simple to implement, such approach requires continuous flooding the network with control messages. This significantly increases communication overhead and becomes inefficient in large-scale networks.

Sridhar et al. in [17] use information provided by modified version of BATMAN [27] routing protocol for efficient CA. Their Localized CA (LOCA) algorithm is supposed to be used when nodes are incrementally added to the network. Two lists are maintained by each node: list of one-hop neighbors and a common neighbors list. Both lists also contain current channels choices for each node. Based on these lists node selects channels to maximize reachability and to connect to denser network areas. The limitation of such approach as mentioned by authors

themselves is inability to scale with network size. Moreover, this CA method can not be applied to the networks that were already deployed.

In general, centralized algorithms provide better results since their decision is based on throughout knowledge of a whole network. However, the network itself is assumed to be stable without fluctuations in traffic patterns and links quality. Any changes in the topology (e.g. link instability, node failures) or traffic patterns would cause the whole process to be repeated. Distributed algorithms, on the other hand, despite of providing near-optimal results, may actively adapt to ongoing changes since their decision is based on local information only.

We now compare all the described approaches according to five parameters: traffic load awareness, usage of a default common channel, protection from ripple effect, fault tolerance and scalability.

Algorithm	Type	Traffic Aware	Default Channel	Ripple Effect Protection	Fault Tolerance	Scalability
BFS-CA	centralized	No	Yes	No	No	Yes
CLICA	centralized	No	No	Yes	No	No
ITACA	centralized	Yes	Yes	No	No	No
LA-CA	centralized	No	No	No	No	No
COMTAC	distributed	Yes	No	Yes	Yes	Yes
SS-CA	distributed	No	Yes	No	Yes	Yes
Hyacinth	distributed	Yes	No	Yes	Yes	No
PCU-CA	distributed	No	No	Yes	Yes	Yes
ROMA	distributed	Yes	No	No	Yes	No
LOCA	distributed	No	No	Yes	Yes	Yes
Our Algorithm	distributed	No	Yes	Yes	Yes	Yes

Table 3.1: Comparison of CA approaches

As it can be seen from Table 3.1, most of the centralized approaches are not scalable and by definition unable to support fault tolerance. Less than a half of described algorithms use traffic load information during CA. This means efficient CA is still possible without relying on traffic patterns. Our algorithm like many others does not consider traffic load distribution for CA. However, in case of using proactive routing protocols, like BATMAN, this is not needed. Multiple paths between any pair of nodes are automatically created and continuously compared in order to react to changes in topology or link quality fluctuations. We, therefore, only provide an updated topology with minimum interference, leaving it for the routing protocol to efficiently use it.

4 Implementation

4.1 Introduction

In this chapter we describe our effort to implement the proposed CA algorithm. The goal was not only to produce a working prototype, but also prove its feasibility and performance gain by comparing it with static CA scheme with common channels.

In the following sections we describe the main features of the algorithm and provide a brief overview of the rest of the technologies that were used. Finally we explain how all these were used and deployed proving that our approach can successfully function in real environment.

4.2 Distributed Channel Assignment Algorithm

In this section we describe the proposed algorithm for assigning channels to radios. It's a distributed algorithm which considers a network with nodes that have at least one radio interface. It's important to mention it doesn't require all the nodes to have the same number of radios and may still be successfully used in the mixed network environment with single- and multi-radio nodes.

We make few assumptions of our algorithm:

- The position of backbone nodes in WMN is static and doesn't change through time.
- Every node has one of its radios tuned to a common channel that remains unchanged. Such approach provides basic network connectivity and prevents from any network partitioning during the channel assignment process. We use 2.4 GHz radio interface for this purpose and refer to it as default interface.
- Additional non-default radio interface (if any) is initially tuned to a common channel as well but later switch to different channel according to CA results. All modifications and channel assignment discussed further are performed under this additional interface unless

otherwise stated. To avoid interference with default radio of the same node we use 5 GHz interface for this purposes.

- We use 5 GHz band for intracluster communication since this band provides more non-overlapping channels (8-24 in 802.11a compared with only 3 in 802.11g standards). Moreover, 2.4 GHz band is much more prone to interference, as it is commonly used by external networks as well as cordless phones and other electronic devices. 5 GHz band, however, provides smaller coverage comparing with 2.4 GHz band but using it for connecting nearby nodes eliminates this disadvantage.

The algorithm was implemented as a Lua script. Lua programming language was selected due to it being powerful, fast and lightweight. Moreover, this scripting language is one of the few that have out-of-the-box support on most of the embedded devices.

Our channel assignment algorithm consists of two phases. In the first phase nodes start from discovering their neighbors. Then they organize in clusters based on the signal level and degree of their neighbors and select a Cluster Head (CH) which will later assign a channel to be used inside the cluster. During the second phase of the algorithm cluster heads select the nodes that will disable their default radio interface to decrease the interference level within the network.

We now describe in details each phase of the algorithm and explain the main ideas behind them.

4.2.1 Phase 1: Network Clustering

Phase 1 of the algorithm may be further divided into two main stages: cluster formation and channel selection. We now describe each of the stages in their execution order.

Neighbor Discovery and Cluster Formation

Clustering in WMN can be explained as virtual partitioning of nodes into various groups. These groups are formed based on the nodes proximity. Two nodes are considered to be neighbors if both of them appear within the transmission range of each other. Each cluster consists of one cluster head, a number of cluster members and border nodes which connect two clusters.

In our algorithm a node with the highest degree of connectivity becomes a cluster head. All the neighboring nodes that haven't yet decided their status become members of its cluster.

Such approach allows to minimize the delay in communication through the CHs and decrease the number of created clusters in the network.

According to our algorithm nodes start from discovering their neighbors. For that each node scans the network and records a signal power level for each of its neighbors. Typically, wireless interface cards report these signal values using a logarithmic scale, i.e. in decibels such as dBm[1]. To detect nodes of the same WMN we use unique BSSID. After this scan a Neighbors list is created. It consists of neighbor nodes IDs and their signal to the requesting node. Based on the obtained data, each node computes its degree and average signal level to all of its neighbors. This data will be later used for selecting a CH and forming a cluster. Nodes then exchange this computed data with all of their neighbors via multicast messages.

When a node receives the degree and average signal information from all of its neighbors it creates a list of so called leaders. Leaders are those neighbors that have higher degree than a node itself and have a chance to become a CH. If any neighbor has the same degree as the requesting node, average signal level is then compared. This allows to select a leader that not only has more neighbors but also has better connection quality. Finally, if average signal level is the same as well, unique numeric node's ID is then used to determine the leader.

However, selecting leaders solely based on their degree may lead to a situation when nodes are in the same cluster but are too far from each other to maintain a stable connection. This will significantly affect the performance of nodes within a cluster and a network itself. We therefore introduce additional check for leaders selection based on their signal level to the requesting node. We set a threshold that is equal to the average signal level between the node and all of it's neighbors. This threshold affects the size of the cluster as well as the number of clusters created. Higher value of a threshold leads to bigger cluster size and fewer clusters. Lower value of a threshold leads to smaller cluster size and more clusters.

If a node doesn't have any leaders, it becomes a CH and waits for incoming request to join its cluster. Each cluster has a unique ID that is the same as the ID of a CH. All the nodes that have leaders send join requests to the best ones from the Leaders list. Leaders with the highest degree and average signal are preferred. Only one leader is processed at a time.

We use a two-way message exchange on this stage of the algorithm. A node that wants to join a cluster of the selected leader sends a join message to it and waits for response. A node that receives this message may accept or reject the incoming request. It accepts the request if

[1]dBm is the power ratio in decibels (dB) of the measured signal power referenced to one milliwatt (mW)

it doesn't have any leaders itself and acts like a CH. Join request will be rejected if a selected leader is already a member of some other cluster. Nodes reply to incoming requests only when their cluster status is settled.

When a request is rejected node then sends a join request to the next available leader from the list and waits for reply. The process is repeated until any of the sent requests is accepted. If all the join requests got rejected by all the leaders, meaning all of them became members of other clusters, node then sends Merge request to the best leader. Upon receiving merge request a node replies with the ID of CH of the cluster it is part of. A requesting node will then send join request to this CH.

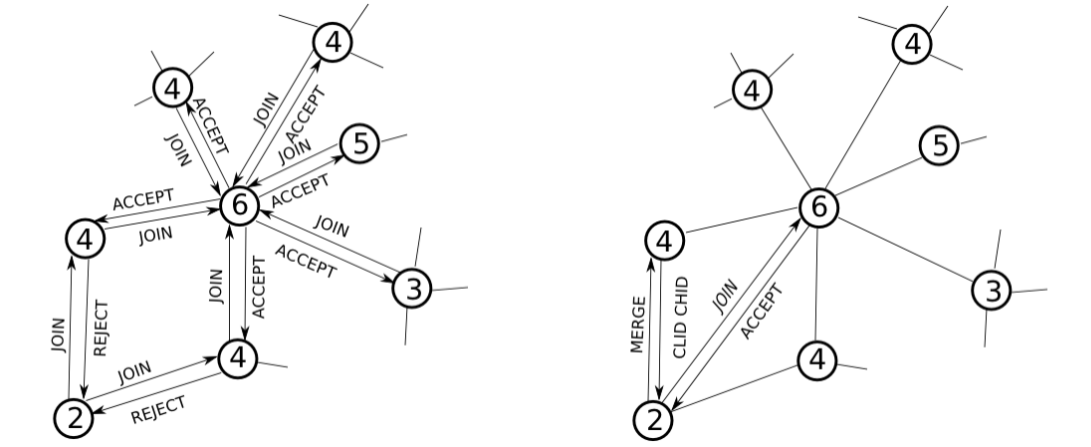

Figure 4.1: Cluster creation process.

We show the process of clustering on a figure 4.1. Numbers in the circles represent the degree of each node. A node in the center with the highest degree 6 receives JOIN requests from all the neighboring nodes that selected it as their leader. It accepts all the requests by replying with ACCEPT message. Node with the degree 2, on the other hand, has 2 leaders and both of them have rejected it's JOIN requests as they became members of the cluster of node 6. It then sends MERGE message (showed on the right side of the figure) to one of it's leaders. A leader replies with ID of the CH (CHID) and the ID of a cluster it is part of. Finally, node sends JOIN requests to the mentioned CH and gets accepted.

Nodes change their SSIDs once they become members of any cluster. Cluster heads do the same. SSID in this case not only consists of node's ID but also has ID of a cluster the node is part of. By periodically scanning the network and checking the SSIDs of nearby nodes, CHs check what nodes in the neighborhood haven't yet joined any cluster. This is essential for the

next stage of the algorithm: selecting a channel for a cluster. In case of continuously joining nodes, channel selection process may not provide accurate results, since the decision will be based on the outdated data. Therefore all the CHs wait until all the nodes have completed the clustering process and changed their SSIDs accordingly.

When CH detects that all the nearby nodes became clustered, it checks whether it has members in it's own cluster. If it doesn't have any members, it then stops the process of clustering and works independently with both of it's radios tuned to default channels. On the other hand, if CH has members in it's cluster it may proceed to the second stage of the first phase, channel selection.

Channel Selection

The main idea behind clustering is decreasing the interference level in the network by grouping nearby nodes in clusters and assigning channels to this clusters such that they don't overlap with other clusters. A channel for each cluster is selected avoiding the channels that are already used by nearby clusters as well as external networks working in the same frequency band. We use 5 GHz band for intracluster communication that provides 8 non-overlapping channels in Europe: 36, 40, 44, 48, 52, 56, 60, 64 [15]

CH starts from requesting all of it's cluster members to report channels used by other clusters and external networks. Since the cluster size and position of nodes may vary, the reported data may be different for each node. Every cluster member upon receiving such request scans the network and creates a list of channels used by other clusters and external networks. This list is then sent back to the requesting CH. CH in it's turn does the same procedure itself.

CH collects all the reported channels and combines them in one list. This list is then sorted by the number of clusters and networks using each channel. Channels used by other clusters are examined first to filter those channels that are not yet occupied. These free channels are then checked with the list of channels used by external networks. Finally, the least used channel is selected for a cluster to work on.

In a WMN consisting of dozens of nodes there is a high chance of CHs starting a process of channel selection at the same time. In this case they might end up with the same channels selected for intracluster communication. We solve this problem by using a special procedure each node in a cluster periodically performs. Each cluster member as well as CH scans the network every minute and checks if there is any other cluster working on the same channel. If

such cluster is detected, it's ID is checked. If the interfering cluster ID is higher than node's own cluster ID, a node then reports this issue to it's own CH. CH upon receiving such message, starts the procedure of channel selection once again and switches to different channel. If the ID of an interfering cluster is lower than node's cluster ID, a node does nothing leaving it for the other cluster to change the currently used channel. By comparing the ID of the interfering cluster we implement a simple protection from a ripple effect. It guarantees that nearby clusters will eventually select non-overlapping channels and will not stuck in endless process of channel selection.

Once the channel is selected CH requests all the cluster members to tune their 5 GHz radio to the specified channel. The process of channel selection is now completed.

4.2.2 Phase 2: Reducing The Interference Level

Second phase of the algorithm is intended to minimize the interference in the network, as well as increase network performance by maximising communication over one-hop links, by disabling 2.4 radio interfaces on some of the nodes, while still maintaining the same level of connectivity. In this case these nodes will remain connected to the rest of the network using their 5 GHz radio through the members of their corresponding clusters. This process has to be carried carefully since there is a high chance of disconnecting the nodes from the network completely.

CH starts the second phase of the algorithm right after all the nodes in the cluster have tuned their radios to the selected channel. It requests all the cluster members to report the signal power of all the nearby clusters. Nodes upon receiving such request scan the network using their 2.4 GHz radio and record the signal power of each cluster they can sense. If there are several nodes of the same cluster detected the one with the lowest signal power is recorded. In the end each of the cluster members as well as CH itself have a list of cluster they can sense from their side and their power signal. This collected data is then sent back to CH for further processing.

CH then combines the received information in one list and sorts it based on the number of different clusters each member can sense and the signal power to each of them. The main idea behind this process is to identify cluster members that will have connectivity with the most of the nearby clusters as well as having the strongest signal to each of them. We call such nodes the border nodes as they lie on the border between two clusters and act as a bridge between them. When such border nodes are identified CH sends them a request to leave 2.4 GHz radio

active. Rest of the nodes are requested to disable their 2.4 GHz. None of these nodes disables the interface yet but waits for a special command for that from a CH.

With frequent changes in the wireless environment links between two neighboring clusters may have different quality leading to different results of border nodes selection. Therefore, when a node of one cluster has been selected as a border node it has to make sure that the closest nodes of the neighboring clusters leave their 2.4 radio active as well even if they don't act as border nodes in their own cluster. This is very important, since if no action taken border nodes will lose connectivity to the previously reported clusters. We use two-way handshake for the border nodes communication with their closest neighbors from other clusters as well as with a CH. We show the described process on a figure 4.2

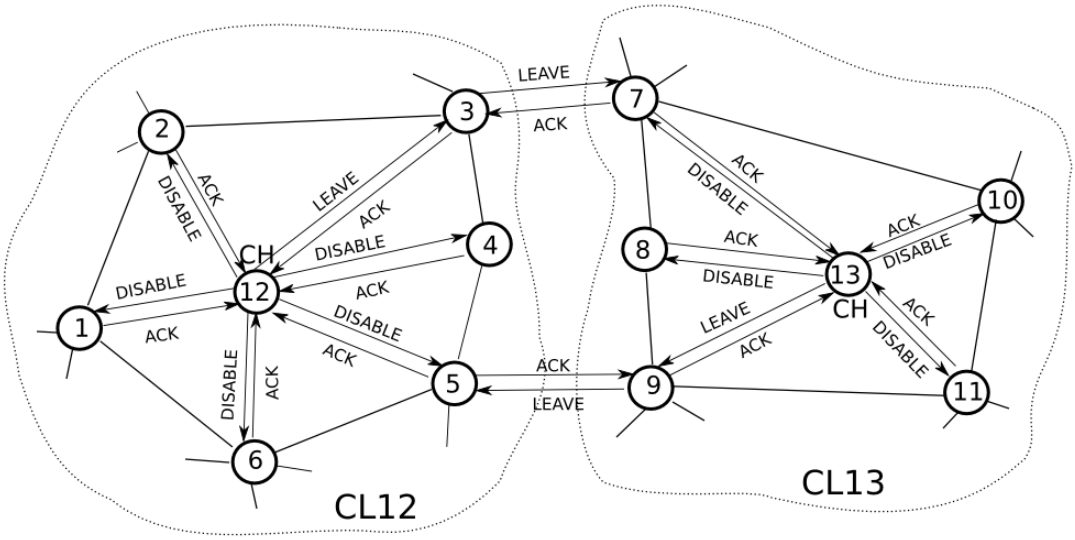

Figure 4.2: Message exchange before disabling 2.4 GHz radio interface.

According to the figure 4.2 CH of CL12 selects node 3 to be a border node as it reported the best signal level for connecting with CL13. On the other hand CH of CL13 selects node 9 as a border node. If the rest of the nodes proceed with disabling their 2.4 GHz radio interface, selected border nodes will end up with completely different connectivity situation and will have to connect to nodes with lower signal level (if any). Node 3 will have to connect to node 9 instead of the node 7 like it was supposed to, at the same time node 9 will have to connect to node 7 instead of connecting to the closest node 5. Links to those furthest nodes may have a poor quality or may not be available at all. Therefore, a special mechanism is needed to prevent such situation.

Every selected border node requests it's closest neighbors from other clusters to leave 2.4 GHz radio active. Those neighbor nodes reply with ACK to that request. Now even if CH requests these nodes to disable their 2.4 GHz radio they will leave it active to ensure connectivity with the neighboring cluster. Upon receiving ACKs from all of their closest neighbors border nodes send ACK message to their respective CHs. Nodes that were requested to disable their 2.4 GHz also send ACKs to the CHs.

Now, when all the nodes in the cluster acknowledged their state, each CH broadcasts its ready state to all other CHs. A CH will only send final request to disable 2.4 radios when all the nearby clusters are ready. This is done to prevent the situation when some clusters have already disabled their 2.4 radios, but others are still selecting the border nodes. Once all the nearby clusters are ready, CH sends a command to disable 2.4 radio to the previously selected nodes. Nodes that were requested to keep their 2.4 GHz radio active by the nearby clusters ignore this command.

4.3 External Software

Above, we presented the software components we developed to prototype our system. We now briefly present other existing software and hardware that our system uses.

4.3.1 Batman-advanced

B.A.T.M.A.N. advanced (often referenced as batman-adv)[2] is an implementation of the proactive B.A.T.M.A.N. routing protocol in form of a linux kernel module operating on layer 2. Batman-adv creates a virtual network interface, on top of an already existing Linux interface, which can be considered as a usual Ethernet device[28].

The protocol proactively maintains information about the existence of all nodes in the mesh network that are accessible via single-hop or multi-hop communication links. The strategy of B.A.T.M.A.N. is not to to find out the full routing path but to find one single-hop neighbor that can be used as a best gateway to communicate with the desired node.

B.A.T.M.A.N. periodically does statistical analysis of protocol packet loss and propagation speed and does not depend on state or topology information from other nodes. It also keeps track of new nodes appearing in the network and informs its neighbors about their existence.

[2]Batman-adv http://www.open-mesh.org/projects/batman-adv/wiki/Wiki

In 2010, Garoppo et al., in [29] showed that batmand-adv is by far more stable and provides more reliable performance in static environment comparing with it's latest competitor iEEE 802.11s standard. However, their results also revealed batman-adv inability to react to node failures and resume the communication in a timely manner. In 2011, Seither et al, in [30] have evaluated the performance of batmand-adv comparing it to another widely used AODV routing protocol. B.A.T.M.A.N. proved to be more stable and performed much better in environment with significant interference and longer distances between the nodes. In 2012, Quartulli et al. in [31] claimed that performance of batman-adv has been improved significantly due to the latest modifications in implementation. Batman-adv has out-of-the-box support for multiple radio interfaces working on the same node. So called interface alternating allows to switch between interfaces while communicating with neighbor nodes. In this case batman-adv forwards frames on a different interface than on which the frame was received. This allows to reduce interference, balance traffic between available radio interfaces and significantly increase the network throughput.

We decided to use batman-adv in our tests due to its mature performance and simple architecture. We use it as a routing protocol to create connectivity between the nodes of the testbed. It proved to be very efficient in dense environment with numerous external networks working side by side with the testbed nodes. At the same time, we have discovered that even in static environment batman-adv causes frequent routing flippings that result in unwanted routing changes. In our case such routes instabilities sometimes led to unpredictable results.

4.3.2 Alfred

A.L.F.R.E.D.[3], or simply Alfred is a user space daemon for distributing arbitrary local information over the mesh/network in a decentralized fashion [32]. It runs as a unix daemon in the background of the system. A user may insert data using the Alfred's command line tools and it will be distributed among all the Alfred servers running on other nodes in WMN. Any node may request this data and receive it from all the running Alfred servers in the network.

We have used Alfred for message exchange between the nodes. It proved to be quite stable and reliable in delivering messages in harsh environment. Alfred server was running on every node so even if any node was not able to deliver the requested information rest of the nodes did.

[3] A.L.F.R.E.D - Almighty Lightweight Fact Remote Exchange Daemon

4.3.3 Wibed Testbed

For evaluation of our algorithm we have used a Wibed Testbed that is deployed across buildings of Universitat Politecnica de Catalunya (UPC) Campus Nord, Barcelona. WiBed [33] is a platform for deploying and managing testbeds for experimenting on mesh networks built on top of cheap commodity routers. Such testbeds are commonly used on pre-production stage as they provide a physical environment on which novel algorithms and systems designed for wireless mesh networks may be tested and verified.

20+ commercial off-the-shelf (COTS) routers are currently available in a testbed. Each node is a TP-Link TL-WDR4300 router with Atheros AR9344@560MHz CPU and two 802.11a and 802.11g radio interfaces. Several nodes are used as gateways between rest of the nodes and a central component - server. Server controls the state of the nodes and sends experiment files uploaded by the system administrator.

All the nodes work as a finite-state machine and constantly report their current status to the server. Nodes are wirelessly connected with each other creating a mesh network where each node may have multiple ways to reach any other node in the network.

The nodes run OpenWRT Linux, a very small Operating System for embedded devices. . To communicate with each other batman-adv routing is used. It creates two network overlays: physical and virtual. Real network topology is thus obfuscated and all the nodes are virtually one hop away from each other.

Few modifications have been done to the Wibed scripts to make the mesh network more stable and reliable. We now briefly describe each of the modifications.

First of all, an additional option to reboot or reset any node was added to the Controller interface. This was needed for some of the nodes that had poor connection due to the number of concrete walls and metal doors separating them from the rest of the network. Every node has to periodically contact Controller and check if there are tasks for it to perform. Due to unstable wireless links, these poorly connected nodes, sometimes were unable to reach Controller on time or were unable to download experiment files completely. This led to a situation when these nodes sometimes got into error state and could not be used anymore. The only way to restore the working state of a node was by manually connecting through ssh[4] to it and rebooting. We've noticed that even if a node was not able to reach Controller it was however still accessible through other nearby nodes. Therefore, we have implemented a mechanism to reach any node

[4]ssh - OpenSSH SSH client (remote login program)

locally through its responsible gateway using set of re-broadcasted commands. This allowed us to reboot or reset problematic node without having direct connection to it. Alfred messaging daemon was used to send and receive these messages. The developed mechanism proved to be very efficient and reliable.

Secondly, few modifications have been done to nodes logs management. Every node was generating a log after each attempt to connect to the Controller. These attempts were made every 15 sec. Each new log was appended to the previously created log. This was eventually leading to a significant log size occupying 100% of space in the /root directory. We, therefore, created a simple script that was controlling the size of the log allowing to save only the latest log entries.

What is more, it took us some time to restore connectivity of several nodes with the rest of the mesh network. Because of the concrete walls and harsh environment some of the nodes were unable to connect to the server. We therefore had to adjust the location of other nodes in the testbed to provide connectivity for all of them in a fair way. Another problem that still remained unsolved was nodes freezing when loosing connection to the server. This bug has been reported and patched by OpenWRT community, however proposed fix was not working in our case. This bug led to a situation when we had to reboot the nodes manually one by one when such freezing happened. We keep on investigating the problem and hope to solve it with the help of other OpenWRT developers and enthusiasts at upcoming Battle Mesh event[5].

Finally, we have updated the firmware of each node with the latest versions of OpenWRT packages as well as patches to reported bugs. It was essential since the firmware previously used was 1 year behind the latest stable release of OpenWRT.

[5]The Wireless Battle of the Mesh is an event that aims at bringing together people from across the world to test the performance of different routing protocols for ad-hoc networks, like Babel, B.A.T.M.A.N., BMX6, OLSR, 802.11s and Static Routing.

5 Evaluation & Discussion

5.1 Introduction

In this section we report the results of performance evaluation of the our algorithm described in §4. Performance results were obtained from the mesh network consisting of 20 multi-radio Wibed nodes. We start by describing our testbed environment and testing methodology. We then present the evaluation results in section §5.2. Finally, in the Discussion section, §5.3 we review the obtained results and make the final conclusions about the overall algorithm performance.

5.1.1 Experiment Setup

Our mesh network testbed consists of 20 nodes, distributed across three floors of C6 building at Universitat Politecnica de Catalunya (UPC) Campus Nord, Barcelona. Figure 5.1 illustrates the location of each node.

Every mesh node is TP-Link TL-WDR4300 router with Atheros AR9344@560MHz CPU equipped with two 802.11a and 802.11g radio interfaces. All the nodes work in ad-hoc mode and create a mesh topology. Every node is running OpenWRT OS and uses batman-adv routing for communicating with other nodes in the network. Lua script with our CA algorithm is executed on each node at the same.

Nodes are placed in different rooms across several floors of the same building. Node with ID 3360 (marked with red color) acts as a gateway providing access to Controller and Internet to the rest of the nodes in the network. Three nodes with IDs 7CDE, 417E and 9DC4 have been placed two floors above the gateway node.

Nodes use 5 GHz radio for intracluster communication and 2.4 GHz radio for intercluster communication. 2.4 GHz radio is a default interface tuned to a common channel 6 which remains unchanged during the experiment. 5 GHz radio is initially tuned to a common channel 36. This channel is expected to change after algorithm execution.

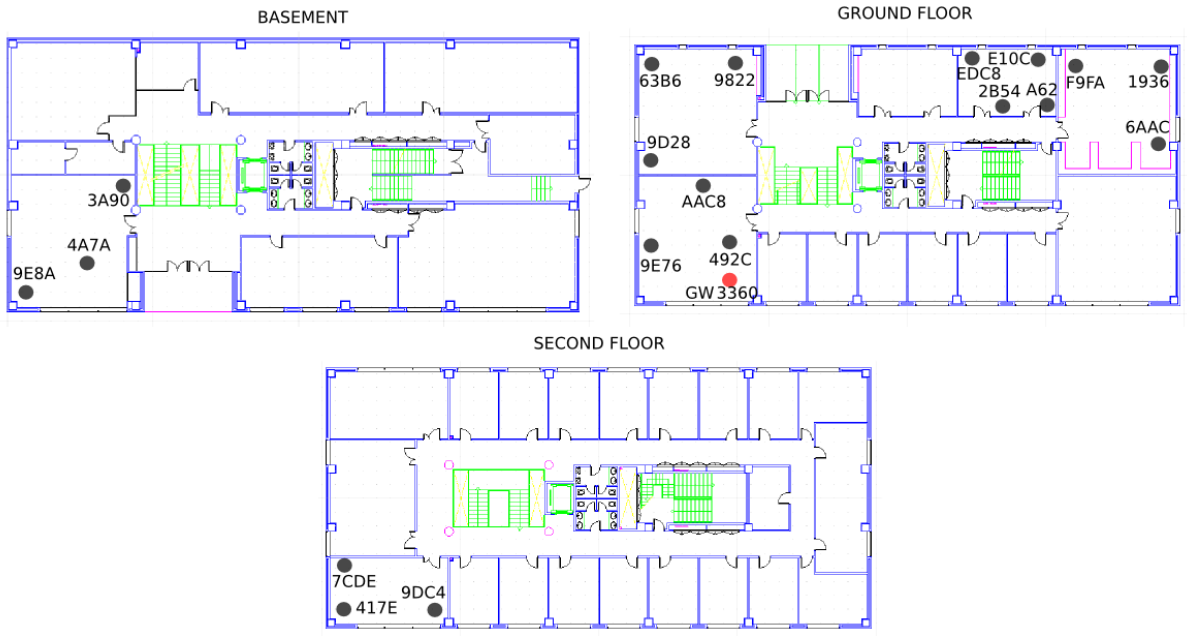

Figure 5.1: Node placement in the testbed.

We compare the performance of our developed algorithm against static channel assignment strategy that is currently used by the Wibed testbed. Following this configuration all the nodes have their radios tuned to common channels: channel 6 for 2.4 GHz radio and channel 36 for 5 GHz radio. Such approach is quite common and is widely used in many multi-radio mesh deployments due to its simplicity [20]. However, these common channels often get overloaded in the most congested parts of the network and significantly reduce the overall performance of the system. We, on the other hand, propose to use a common channel only for intercluster communication, while different non-overlapping channels are used inside of each cluster. Such approach allows to reduce the impact of interference between or within the flow paths and increase the throughput.

Taking into account that interference primarily affects the throughput of the network [16] we measure the throughput of both UDP and TCP flows between the nodes in the mesh network using iperf[1] tool. For TCP we measure the maximum achieved throughput during the period of 10 seconds. We configure the UDP source (iperf server) to send datagrams during a period of 10 sec and report the bandwidth obtained. Both TCP and UDP measurements were performed ten times recording the average values.

[1]iPerf is a cross-platform tool for active measurements of the maximum achievable bandwidth on IP networks.

We performed the TCP and UDP flows measurement for three separate cases:

- TCP and UDP flows performance between all the pairs of cluster heads
- TCP and UDP flows performance between all the cluster heads and a gateway node
- TCP and UDP flows performance between cluster heads and the most remote node of the same cluster

Cluster to cluster measurements allow to get an idea of average bandwidth between any pair of the nodes in the network. Cluster to gateway measurements simulate the main use-case of WMNs that is Internet access and show the bandwidth available to the connected client devices. We also check the performance inside each of the cluster to see how it improves comparing with common channels case performance.

Additional experiments were performed to measure the download time of different file sizes from each of the cluster heads. Six files of 1, 2, 4, 8, 16 and 32 MB sizes were placed on the gateway node. All the cluster heads were downloading each file 5 times and recording the download time.

Our testbed was deployed in the building where numerous external wifi networks worked at the same time interfering with the testbed nodes. However, few of them were working on the channels used in our testbed. In order to evaluate the performance of our algorithm under the severe interference from nearby stations working on the same frequencies, we configured 6 nodes of the testbed to continuously exchange traffic and act independently from the rest of the mesh network which in those measurements contains the rest 14 nodes. Such stress test can show the performance gain of interference-aware CA algorithm over the static common channels case.

These interfering nodes used different BSSIDs but were tuned to the same common channels on both of their radio interfaces. In order to create equal level of interference in all parts of the network we used one node from each room for this purpose (see Figure 5.2).

With the above setup we compare the performance of our algorithm against the baseline (common channels on both radios) under the influence of interfering nodes and without them. For each of the mentioned test cases we measured TCP and UDP performance as well as download time for each file size.

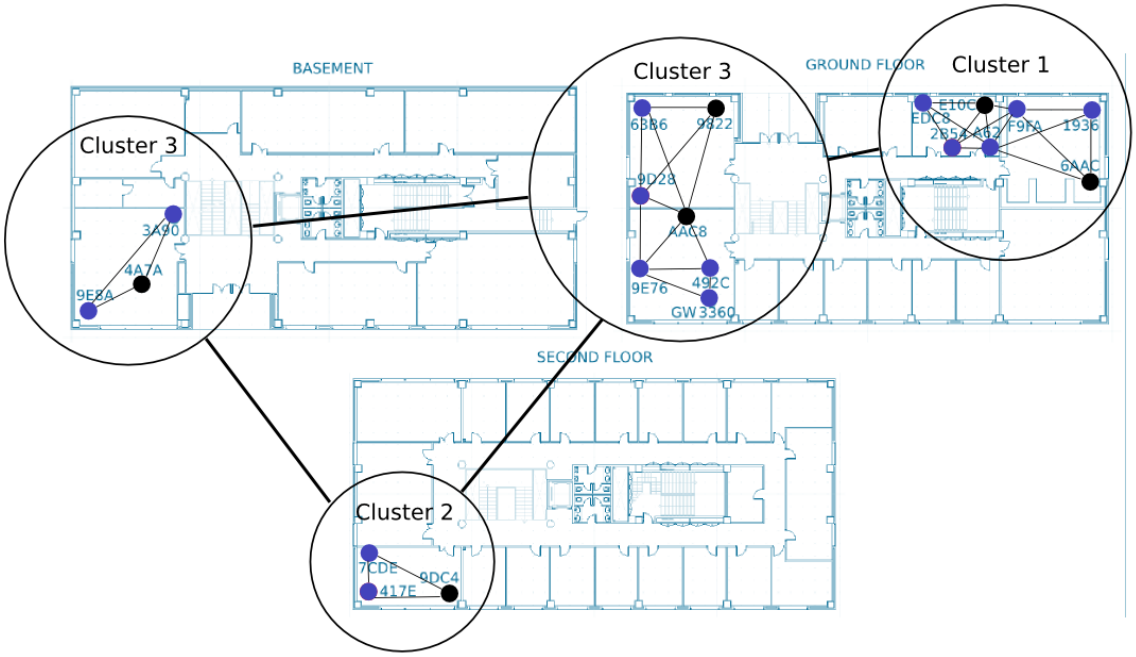

Figure 5.2: Nodes creating the interference in the network (marked in black).

5.2 Evaluation

After the execution of our algorithm three clusters were formed. Each cluster was using a separate channel that was not overlapping with channels used by any of the nearby clusters. Figure 5.3 shows the distribution of nodes in the created clusters. Nodes marked in red act as CHs. During the experiments with interference nodes same clusters were formed though they had less members.

We now present the results of performance measurements under the influence of interfering nodes and without them. We explain the results obtained for each case and compare them against the baseline.

Download Time

Download time was measured by downloading files of different sizes from the gateway node. As there were three different clusters formed we had three CHs to work with. We have noticed different behavior for each of the CHs. Node 2B54, for instance had the worst connection to the gateway and its download time was significantly higher comparing with the rest of the CHs. Node 417E appeared to be better connected to the gateway and showed lower download time.

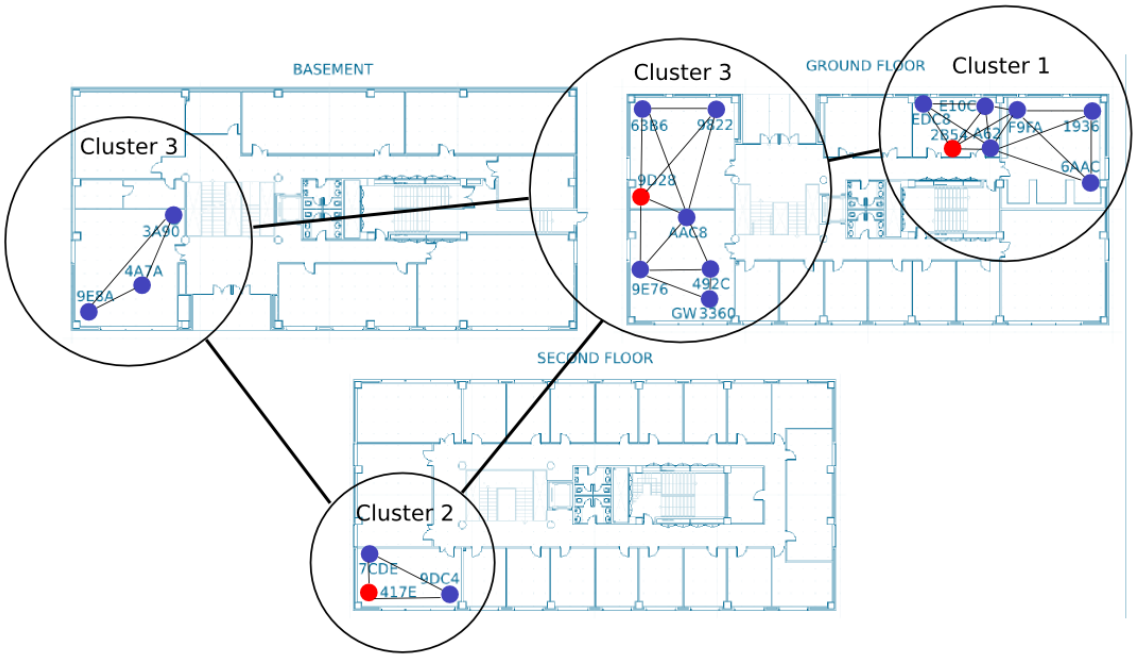

Figure 5.3: Nodes distribution between created clusters

Node 9D28 showed the best download time since it was a member of the same cluster as GW and had direct connection to it. We therefore discuss the performance of each cluster separately.

Figure 5.4: Download time for cluster 1 without external interference (left) and with it (right)

Without external interference download time for Cluster 1 as seen on the Figure 5.4 didn't change much after executing the algorithm. In this case the paths used to reach the gateway had no congestion and provided the same bandwidth. The performance, however, changed once

we introduced external interference. We can clearly see the difference in download time between algorithm and baseline results. By exploiting paths not affected by interference algorithm allowed to overcome congestion problems the baseline faced.

On the other hand, cluster 2 being separated by two floors from the gateway node showed much better results after algorithm execution in both cases (see Figure 5.5). Main speedup was achieved while downloading file of 16 and 32MB. Such a big difference between baseline and algorithm results may be explained by the amount of routing changes batman-adv had to perform while downloading each file. Since downloading of bigger files took more time, some of the paths used could become congested and had to be replaced with other ones with a better link quality. Proactive routing protocol like batman-adv does these changes all the time if the network environment is not stable, however such changes may cause delays and result in increased download time. At the same time paths created by the algorithm were less affected by the interfering nodes thus required less routing changes and performed better.

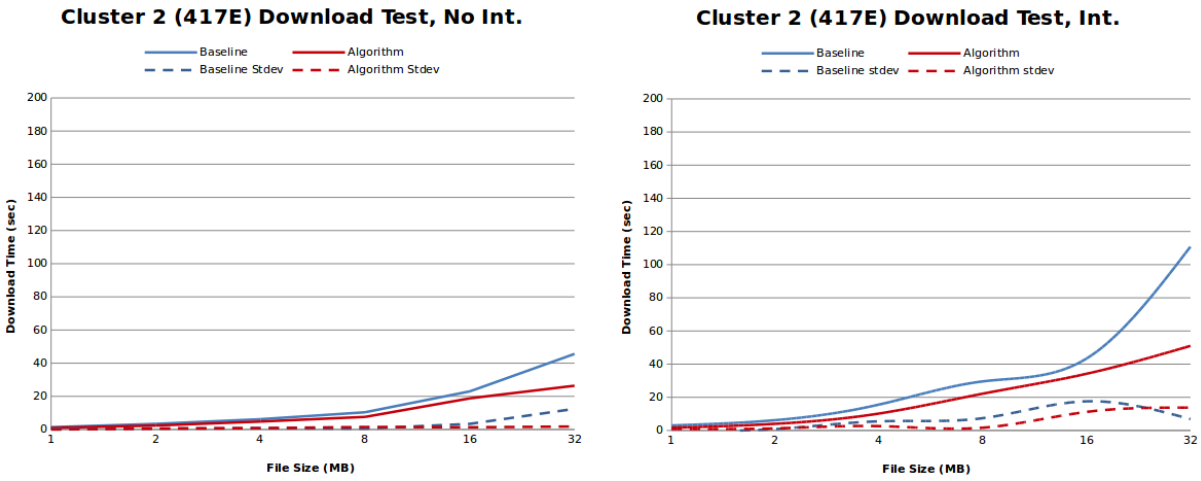

Figure 5.5: Download time for cluster 2 without external interference (left) and with it (right)

Cluster 3 as shown at Figure 5.6 being the closest cluster the the gateway node acted similar to Cluster 1 showing no difference in download time in case of no external interference. However, when the mesh network was affected by interfering nodes, download time has significantly increased. Considering that CH and a gateway node were the members of the same cluster and both had their 5 GHz radio tuned to a common channel not affected by the external interference, they could establish one-hop connection without intermediate nodes. In case of the baseline topology, however, same CH had to access the gateway in a multi-hop manner thus

Average Download Time	Without Interference	With Interference
Baseline	11.71s	39.81s
Algorithm	10.32s	24.23s
Speedup	12%	39%

Table 5.1: Average download time and speedup achieved after algorithm execution

obtaining less bandwidth.

Figure 5.6: Download time for cluster 3 without external interference (left) and with it (right)

In table 5.2 we present an average download time as well as a speedup achieved for all three clusters combined and for both cases with and without interference. In general, the outcome of the algorithm showed much better performance comparing with the baseline case. Improvements in download time has been observed in both cases with and without interference reaching almost 40% of speedup under the influence of external networks working on the same channels.

TCP and UDP Performance

We now present the results of TCP and UDP bandwidth measurements as an empirical CDF of the values obtained for each pair of CHs. As it was discussed previously three different cases were examined. We start from analyzing TCP and UDP performance between each pair of CHs (intercluster communication). Then we analyze the CH-to-gateway performance. Finally, we measure the performance of intracluster communication between CH and the most remote

node of the same cluster.

Intercluster Communication

Algorithm execution resulted in general in higher TCP and UDP bandwidth available to links connecting different CHs in the network. As it can be seen at the Figure 5.7 without any external interference 40% of the links between CHs created with algorithm obtained 7 MBits/sec TCP bandwidth while the same bandwidth was available only to 20% of links created in the initial baseline case. At the same time, algorithm results still allowed to obtain higher bandwidth under the influence of interference reaching on average 2 MBits/sec bandwidth while only 1 MBits/sec in average was available at baseline case.

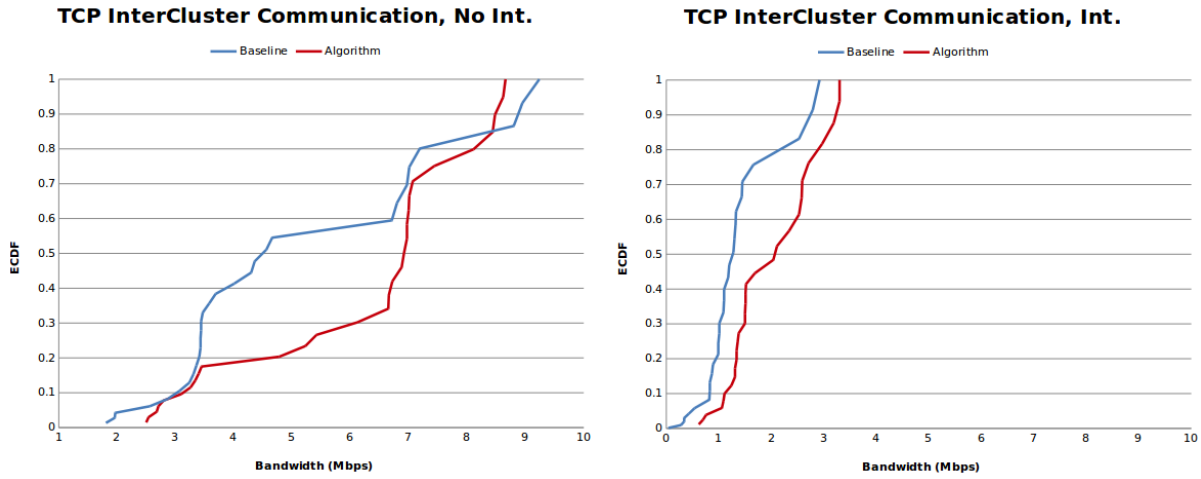

Figure 5.7: Measured intercluster TCP bandwidth without external interference (left) and with it (right)

UDP performance for intercluster communication as seen at Figure 5.8 had a similar pattern. Without any interference from external networks almost half of the links created after executing the algorithm were operating with 7 MBits/sec bandwidth. Same bandwidth was available to few links in a baseline case, while the average UDP bandwidth was around 4.5 MBits/sec. UDP performance was however the opposite in case of interference influence. Both algorithms obtained similar results though baseline reached slightly higher bandwidth in general.

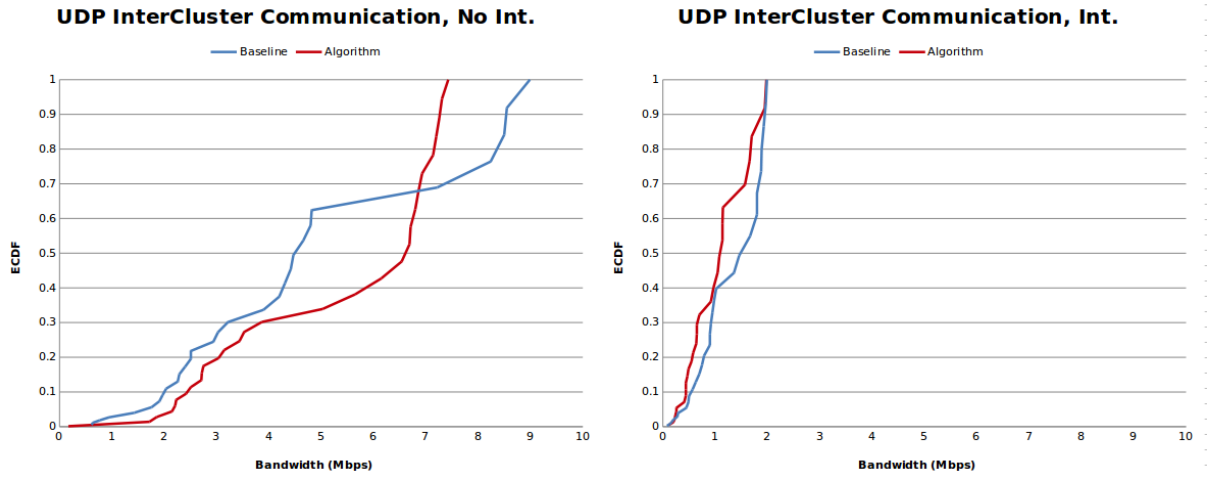

Figure 5.8: Measured intercluster UDP bandwidth without external interference (left) and with it (right)

Cluster-to-gateway Communication

In cluster-to-gateway communication we were checking the bandwidth as seen by each CH while exchanging traffic with the gateway node. As it was mentioned before CHs had different distances to gateway thus were getting different bandwidth.

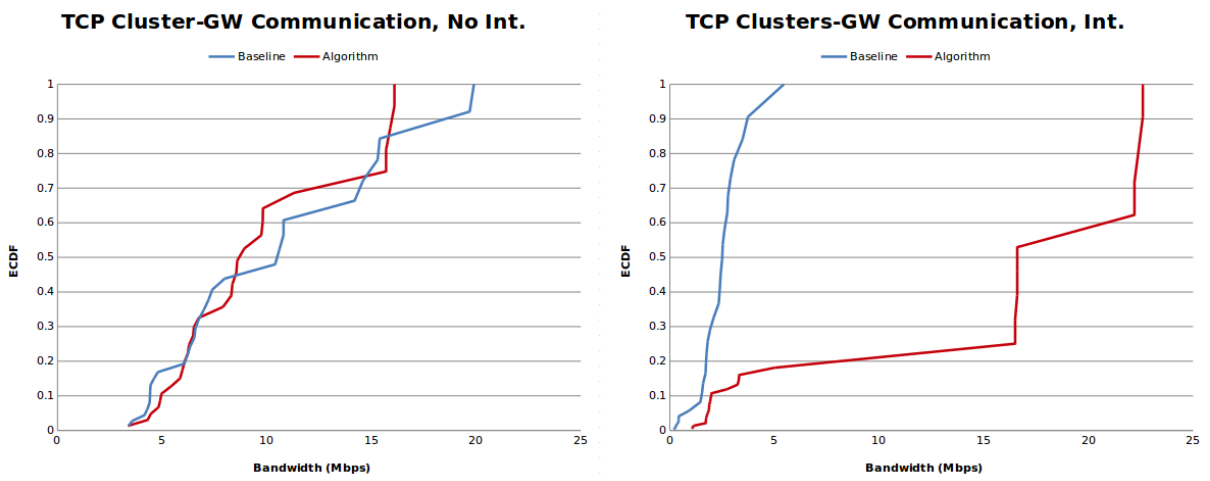

Figure 5.9: Measured TCP bandwidth for CH-to-GW communication without external interference (left) and with it (right)

TCP performance of algorithm in case of no interference as seen on Figure 5.9 is similar

to the baseline and doesn't change much after applying the algorithm. At the same time, with interference we can clearly see significant bandwidth improvement. Moreover, it is possible to notice a specific pattern of EDCF behavior. EDCF of algorithm values has three steps on the graph each representing the performance of different CH. Low values (below 4 MBits/sec) were obtained by CH 2B54 that had the worst link quality to the gateway. Then follow the values obtained for another CH 417E that was better connected to GW and obtained higher bandwidth values of 17 MBits/sec on average. Finally, values of CH 9D28 that appeared to be in the same cluster as gateway yield the bandwidth to the maximum of 22 Mbits/sec.

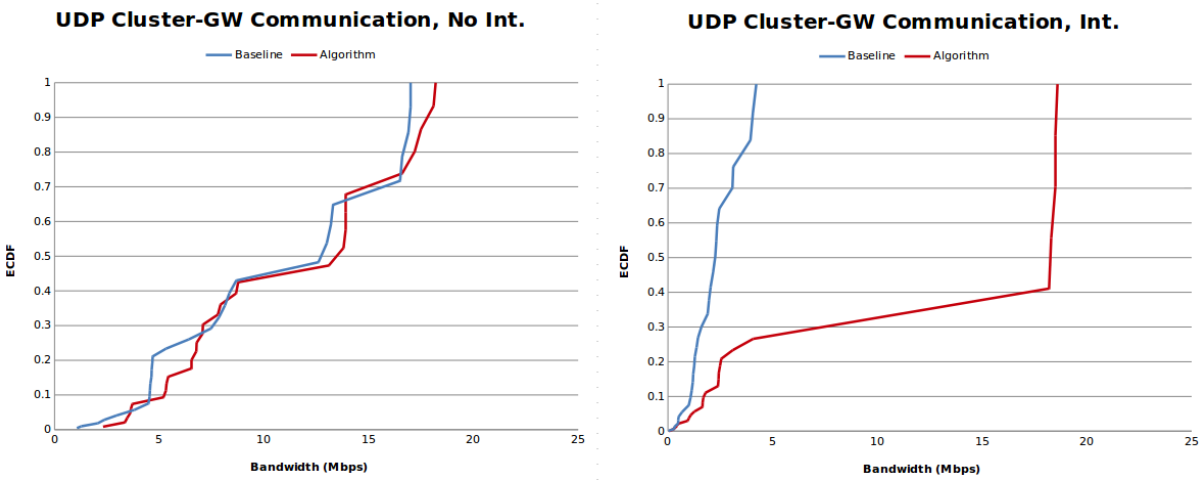

Figure 5.10: Measured UDP bandwidth for CH-to-GW communication without external interference (left) and with it (right)

Behaviour of UDP bandwidth had a similar pattern (see Figure 5.10. No or little improvement of bandwidth in case of no interference and significant gain for the case of having external interference. Such a huge difference in bandwidth between two cases may be explained by existence of paths created after the algorithm execution that were not affected by the interference. Nodes proximity to the gateway also plays an essential role in bandwidth performance.

Intracluster Communication

Bandwidth between the nodes of the same cluster is supposed to be high since every pair of nodes at most one hop away from each other. They also share the same channel that is less affected by the interference from other clusters and external networks. At the same time without having clusters and using set of common channels on all the nodes may need to make several

hops in order to reach each other and utilize congested paths that are used by other nodes at the same time.

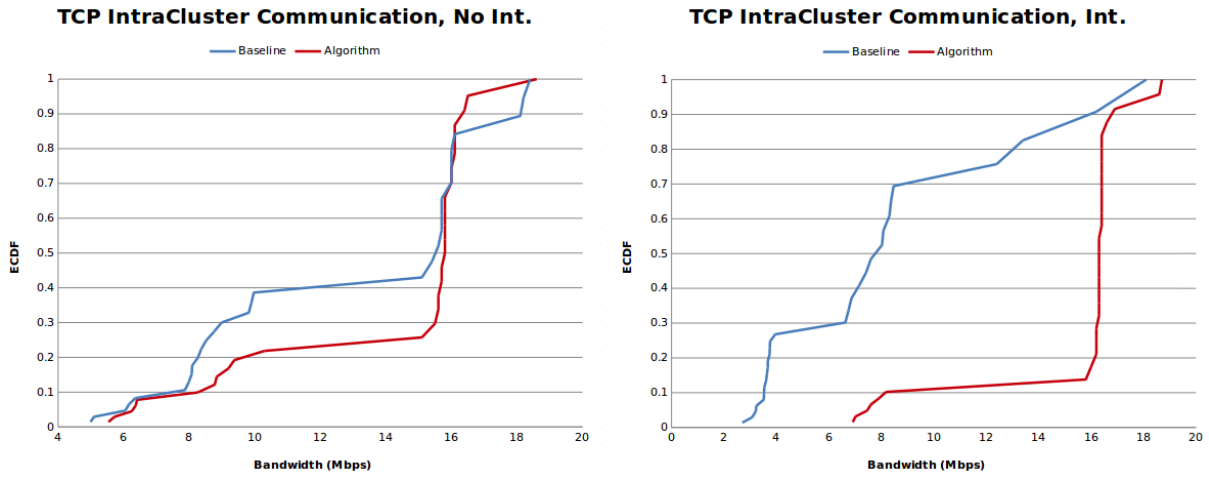

Figure 5.11: Measured intracluster TCP bandwidth without external interference (left) and with it (right)

Algorithm results for no interference case allowed to provide higher TCP bandwidth for nodes within a cluster. 80% of links had not less than 16 Mbits/sec while same bandwidth was provided to 45% of links in the baseline. In case of interference, we noticed no change in behaviour of links created by the algorithm while the baseline case performance was halved.

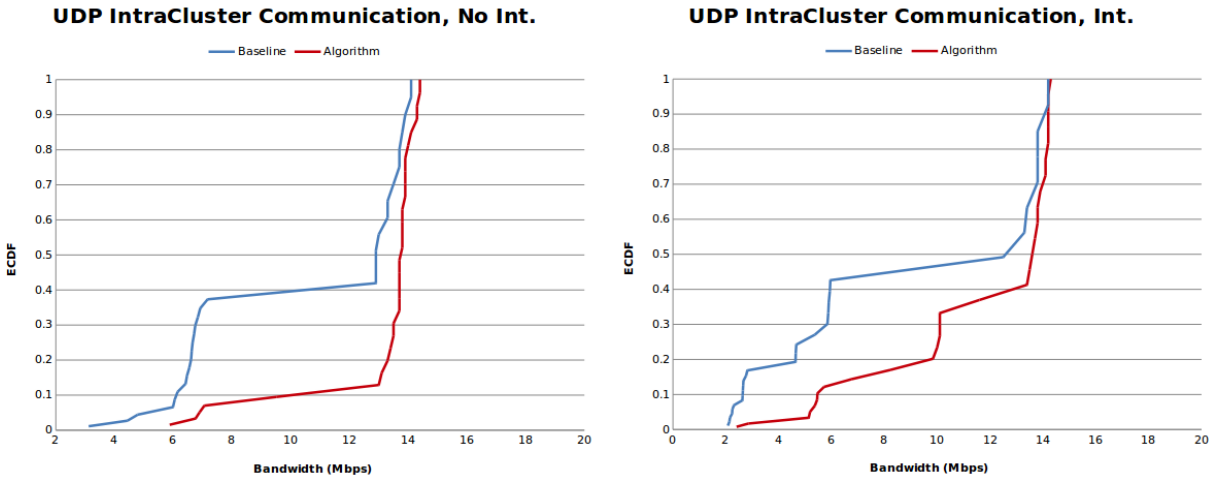

Figure 5.12: Measured intracluster UDP bandwidth without external interference (left) and with it (right)

UDP performance of intracluster communication follows the same pattern as TCP. We observed significant improvement in both cases with or without interference.

5.3 Discussion

In this section we review the results of our algorithm and argue about the success of our decisions and our general approach.

First of all, we managed to implement distributed channel assignment algorithm that aimed to reduce the interference within the network and provide a mechanism to avoid the interference from the external networks. Algorithm performed as expected and not only created clusters that were working on non-overlapping channels but also maintained the connectivity between any pair of nodes in the network despite of disabling radios on some of the nodes.

Experiments results showed that both TCP and UDP performance was significantly better than the baseline performance reaching 40% speedup in download time and up to 50% increase in bandwidth. Channel diversity from cluster to cluster allowed to gain more bandwidth by exploiting non-interfering paths between nodes. Therefore, links created after the algorithm execution were, in general, more stable and less affected by the inter-flow and intra-flow interference as well as interference from external networks. At the same time nodes proximity to the gateway was essential for good performance since there was a high chance of getting into the same cluster with it and having it as a one-hop neighbor. In this case the bandwidth was only limited by the protocol standards.

We conclude that our developed CA algorithm may be successfully used for WMN that operate in urban dense environment where the frequency bands are shared with numerous external networks and channel overlapping can not be avoided. At the same time, our algorithm is able to regularly review its current channel selection and make changes accordingly if needed.

6 Conclusion

6.1 Conclusions

Throughout this document we presented a distributed channel assignment algorithm developed and successfully tested on a real-life testbed. We proposed a novel technique in channel assignment based on the idea of grouping nodes into clusters and selecting non-overlapping channels for each cluster as well as disabling radios on some of the nodes to minimize the interference within the network. Our algorithm is self-stabilizing and allows to react to changes in the wireless environment in a timely manner. Experiments results proved the feasibility of our approach. Moreover, we performed a performance analysis of our system reaching the conclusion that it allows to significantly improve the performance of WMN in dense environment where frequency bands are shared with other networks. Finally we discussed how our effort satisfies our goals and under which conditions developed solution may be used in current WMNs.

6.2 Future Work

The next step would be to use channel utilization as input for our algorithm during the process of selecting a channel for cluster. In this case algorithm will be able to not only select the channel that is used by least number of networks but also the one that is not congested. At the same time, we might decrease the interference within a network even more by using different channels on 2.4 GHz interfaces for connecting with nearby clusters. It would be interesting to test our approach in the WMN with greater number of nodes and in harsh environment like city or rural areas where distance between the nodes may be significant. We would also like to check the performance of our algorithm under high volume of traffic to see if 2.4 links between clusters get overloaded.

Bibliography

[1] D. Vega, L. Cerda-Alabern, L. Navarro, and R. Meseguer, "Topology patterns of a community network: Guifi. net," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on*, pp. 612–619, IEEE, 2012.

[2] I. Akyildiz and X. Wang, "A survey on wireless mesh networks," *Communications Magazine, IEEE*, vol. 43, no. 9, pp. S23–S30, 2005.

[3] S. Chinara and S. K. Rath, "A survey on one-hop clustering algorithms in mobile ad hoc networks," *Journal of Network and Systems Management*, vol. 17, no. 1-2, pp. 183–207, 2009.

[4] K. Ramesh and D. K. Somasundaram, "A comparative study of clusterhead selection algorithms in wireless sensor networks," *arXiv preprint arXiv:1205.1673*, 2012.

[5] A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," *Proceedings of the IEEE*, vol. 75, no. 1, pp. 56–73, 1987.

[6] A. K. Parekh, "Selecting routers in ad-hoc wireless networks," in *Proceedings SBT/IEEE Intl Telecommunications Symposium*, pp. 420–424, 1994.

[7] M. Chatterjee, S. K. Das, and D. Turgut, "Wca: A weighted clustering algorithm for mobile ad hoc networks," *Cluster Computing*, vol. 5, no. 2, pp. 193–204, 2002.

[8] R. Bruno, M. Conti, and E. Gregori, "Mesh networks: commodity multihop ad hoc networks," *Communications Magazine, IEEE*, vol. 43, no. 3, pp. 123–131, 2005.

[9] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer networks*, vol. 47, no. 4, pp. 445–487, 2005.

[10] W. Si, S. Selvakennedy, and A. Y. Zomaya, "An overview of channel assignment methods for multi-radio multi-channel wireless mesh networks," *Journal of Parallel and Distributed Computing*, vol. 70, no. 5, pp. 505–524, 2010.

[11] A. Nasipuri, J. Zhuang, and S. R. Das, "A multichannel csma mac protocol for multihop wireless networks," in *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, pp. 1402–1406, IEEE, 1999.

[12] A. Tzamaloukas and J. Garcia-Luna-Aceves, "A receiver-initiated collision-avoidance protocol for multi-channel networks," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, pp. 189–198, IEEE, 2001.

[13] R. Chandra and P. Bahl, "Multinet: Connecting to multiple ieee 802.11 networks using a single wireless card," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 882–893, IEEE, 2004.

[14] L. standards Committee *et al.*, "Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE-SA Standards Board*, 2003.

[15] I. S. Association *et al.*, "802.11-2012-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," 2012.

[16] A. Raniwala and T.-c. Chiueh, "Architecture and algorithms for an ieee 802.11-based multi-channel wireless mesh network," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, pp. 2223–2234, IEEE, 2005.

[17] K. Sridhar, C. Casetti, and C.-F. Chiasserini, "A localized and distributed channel assignment scheme for wireless mesh networks," in *Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on*, pp. 45–52, IEEE, 2009.

[18] K.-H. Kim and K. G. Shin, "On accurate measurement of link quality in multi-hop wireless mesh networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking*, pp. 38–49, ACM, 2006.

[19] K. N. Ramachandran, E. M. Belding-Royer, K. C. Almeroth, and M. M. Buddhikot, "Interference-aware channel assignment in multi-radio wireless mesh networks.," in *Infocom*, vol. 6, pp. 1–12, 2006.

[20] M. K. Marina, S. R. Das, and A. P. Subramanian, "A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks," *Computer networks*, vol. 54, no. 2, pp. 241–256, 2010.

[21] R. Riggio, T. Rasheed, S. Testi, F. Granelli, and I. Chlamtac, "Interference and traffic aware channel assignment in wifi-based wireless mesh networks," *Ad Hoc Networks*, vol. 9, no. 5, pp. 864–875, 2011.

[22] A. Raniwala, K. Gopalan, and T.-c. Chiueh, "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 2, pp. 50–65, 2004.

[23] A. Naveed, S. S. Kanhere, and S. K. Jha, "Topology control and channel assignment in multi-radio multi-channel wireless mesh networks," in *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE International Conference on*, pp. 1–9, IEEE, 2007.

[24] B.-J. Ko, V. Misra, J. Padhye, and D. Rubenstein, "Distributed channel assignment in multi-radio 802.11 mesh networks," in *Wireless Communications and Networking Conference, 2007. WCNC 2007. IEEE*, pp. 3978–3983, IEEE, 2007.

[25] P. Kyasanur and N. H. Vaidya, "Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 10, no. 1, pp. 31–43, 2006.

[26] A. Dhananjay, H. Zhang, J. Li, and L. Subramanian, "Practical, distributed channel assignment and routing in dual-radio mesh networks," in *ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 99–110, ACM, 2009.

[27] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich, "Better approach to mobile ad-hoc networking (batman)," *IETF draft, October*, 2008.

[28] S. Wunderlich, M. Lindner, and W. Tsai, "B.a.t.m.a.n. advanced documentation," June 2009.

[29] R. G. Garroppo, S. Giordano, and L. Tavanti, "Experimental evaluation of two open source solutions for wireless mesh routing at layer two," in *Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on*, pp. 232–237, IEEE, 2010.

[30] D. Seither, A. König, and M. Hollick, "Routing performance of wireless mesh networks: A practical evaluation of batman advanced," in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pp. 897–904, IEEE, 2011.

[31] A. Quartulli and R. Lo Cigno, "Improving mesh-agnostic client announcement in b.a.t.m.a.n.-advanced," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on*, pp. 659–666, 2012.

[32] S. Wunderlich and S. Eckelmann, "Alfred architecture," June 2009.

[33] P. Escrich, R. Baig, E. Dimogerontakis, E. Carbo, A. Neumann, A. Fonseca, F. Freitag, and L. Navarro, "Wibed, a platform for commodity wireless testbeds," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on*, pp. 85–91, IEEE, 2014.