TECHNICAL UNIVERSITY OF CRETE
SCHOOL OF ELECTRONIC AND COMPUTER ENGINEERING
TELECOMMUNICATIONS DIVISION

Large Scale Rooftop Networking

by

Dimitris Ntilis

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE MASTER OF SCIENCE OF

ELECTRONIC AND COMPUTER ENGINEERING

November 2015

Abstract

This work describes the design and implementation of a city-wide, low-cost, 802.11s testbed with cognitive principles; a wireless mesh network with links on the order of four kilometers and two multi-frequency channel algorithms on top of 802.11s are implemented and tested, that select appropriate frequency channels for the underlying network interfaces. Specifically, a channel probing algorithm collects packet error rate statistics of transmissions in each frequency channel at each link under test; a frequency channel allocation algorithm assigns channels to each link based on the collected information. The objective for the channel allocation algorithm is to offer frequency planning that minimizes interference experienced by the network nodes. As a result, *internal* network interference caused by neighboring nodes, as well as *external* interference from the environment is reduced. These algorithms were implemented in the *Aquanet* network, a city-wide mesh network connecting critical water tanks, storage reservoirs and pumping stations.

Acknowledgements

Thanks to my parents for the psychological support and for their unconditional love.

I would like to thank a lot my thesis supervisor, Aggelos Bletsas, for his patience, guidance, ideas, support and especially for his contribution in my knowledge. Many thanks to Panagiotis Oikonomakos and Vasilis Papadakis for their contribution in this project.

Special thanks to my girlfriend for her patience, her support, her love and especially for the most beautiful moments in my life.

Finally, I would like to thank a lot my friends for the adorable moments I had with them and for their support.

Table of Contents

List of Tables

List of Figures

Chapter 1

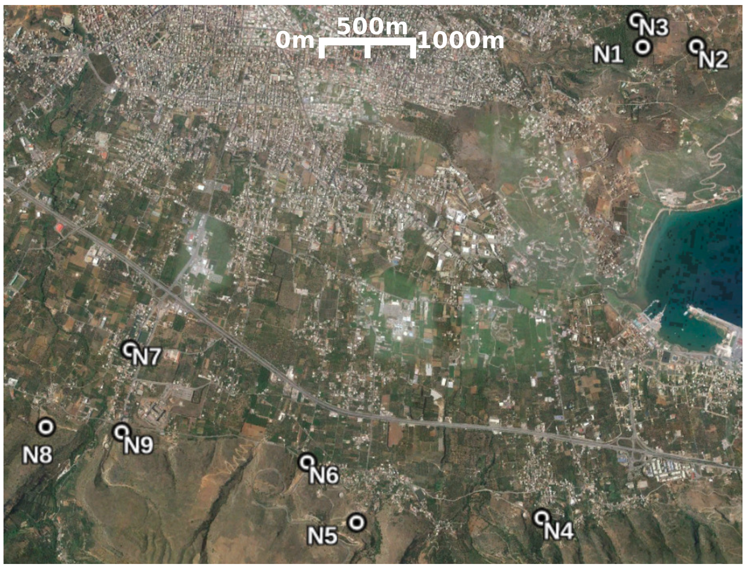
Introduction

1.1 Rooftop Networking

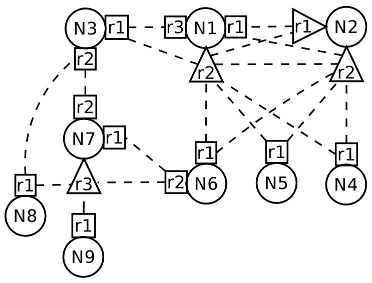
Creating an autonomous and city-wide wireless (mesh) network that connects multiple water tanks and pumping stations is challenging for several reasons. Network planners can only install mesh network nodes at specific locations, limiting the potential number of wireless paths between any two nodes, thus reducing diversity. Furthermore links on the order of $4-5$ kilometers must be implemented, with equivalent isotropic radiated power (EIRP) at each transmitter not exceeding 20 dBm. Additionally, 802.11 frequency bands in a city are unlicensed and crowded, causing interference to the various transmissions. Fig. 1.1a offers the locations of water tanks and pumping stations that must be connected at city-wide scale in a pilot network, targeting water management [1].

Given that long-distance links were needed, carrier frequency of 2.4 GHz was chosen, as opposed to 5 GHz, in order to reduce the free-space loss, while increasing the size of the utilized antennas (and thus, their gain) and reducing the number of available *orthogonal* frequency channels. Specifically, there are only 3 orthogonal frequency channels in 802.11b (2.4 GHz), as opposed to other versions of 802.11 in the 5 GHz regime.

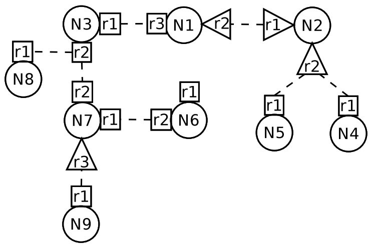
In this work, each wireless node was equipped with multiple low cost radio interfaces, with two types of high gain directive antennas: a) broad 3-dB beam-width of 120^o (triangle in Fig. 1.1) or b) narrow 3-dB beam-width of 15^o (rectangular in connectivity graph and routing tree of Fig. 1.1). Installing several radio interfaces per node (e.g. N_1 has radio interfaces 1, 2 and 3 denoted as $N_1 : r_1$, $N_1 : r_2$ and $N_1 : r_3$) was necessary in order to improve connection diversity and thus, network stability. However this imposed *self-interference*, when two or more radios of the same node operated at the same frequency, due to electromagnetic coupling. Furthermore, additional constraints were imposed by the use of the broad beam-width antennas on the frequency channel assignment problem, since links served by the same antenna should operate at the same frequency channel. Custom low-cost, configurable antennas were also designed and tested at the network [2].



(a) Overall Testbed Network Topology.



(b) Communication Link Graph of the Pilot Network.



(c) Routing.

Figure 1.1: Network Topology and Routing.

1.2 Prior Art

The main difference between this project and prior art in [3, 4, 5, 6] is that this project targets long range links in a real 802.11s, city-wide network; in order to maintain connectivity between the network nodes it is critical to find a frequency assignment with minimum interference *on specific links.* This departs from the work on [7], where polarization diversity was used in order to increase reliability on long-range links.

There are other frequency allocation algorithms trying to find the optimal frequency channel assignment in wireless mesh networks. One of them, the centralized Tabu algorithm [3], was applied in the simulation networks of 6.1 and the results were compared with the results obtained by the algorithms implemented in this project.

The Tabu-based algorithm makes use of Conflict graphs (CG), where each routing link between two stations is a node in the Conflict graph, as opposed to Multi-Radio Conflict graphs where the CG vertices are produced from routing links between radio interfaces. The algorithm tries to color the vertices V_c of the Conflict graph using K colors (frequency channels) while maintaining the interface constraint (a station must not be assigned more channels than its radio interfaces) and minimizing the number of monochromatic edges in the conflict graph. It consists of two phases. The first phase is used to find a good channel assignment without taking into consideration the interface constraints. In the second phase these interface constraint violations are removed to get a feasible channel assignment.

The first phase starts with a random initial solution f_0 where each CG vertex is assigned a random color in K. Then a sequence of solutions $f_1, f_2, ..., f_j, ...$ is created in order to reach a minimum interference assignment.

Given a solution f_j, the next one is created as follows. Firstly, a random number of neighboring solutions of f_j is created. Each neighboring solution of f_j is generated by selecting a random CG vertex u and reassigning it to a random color in $[K - f_j(u)]$. The neighboring solution with the lowest interference (fewer interfering links among network stations) is picked as the next solution f_{j+1}. The interference of the new solution does not have to be less than the interference of the previous solution, so as to allow escaping from local minimum. To achieve faster convergence, reassigning the same color to a vertex more than once, is avoided by maintaining a tabu list of limited size.

The first phase terminates when a maximum number of iterations have passed without any improvement in the remaining interference. The solution with the lowest interference is kept and then assigned to the stations.

The second phase eliminates the interface constraints by repeated application of the *merge* procedure. Among all the network stations whose number of radio interfaces is less than the number of distinct colors assigned to the incident communication links, the station wherein the difference between the above terms is the maximum, is picked. Let i be the selected station. The number of colors incident on i is reduced by picking two colors k_1 and k_2 incident to i and changing k_1 to k_2. This change is propagated to all k_1-colored links that are connected to the links whose color has just been changed from k_1 to k_2. The repeated application of this merge procedure, will resolve all interface constraints. The merge procedure is depicted in Fig. 1.2.

Besides frequency allocation algorithms, there exist projects that aimed to create city-wide wireless mesh networks. One of the most important works in this domain is the MIT Roofnet project [8]. This is a 802.11b/g wireless mesh network that aims to provide the users with internet access. The network's stations are cheap Linux computers equipped with an antenna and an Ethernet port. The Ethernet port is used by the stations that have direct internet access and the antennas were installed on the roofs of the students that agreed to take part in the experiment. The radio interfaces are used only for communication between the stations and do not serve as Access Points. Each station is in the range of some of the other stations. Communication with the rest is accomplished with multi-hop forwarding.

The computer used was an iDOT Slim PC with a Mini-ITX motherboard and 500 MHz x86 CPU. The low CPU power reduced the amount of heat produced. Each station was equipped with a Hyperlink Technologies 8 dBi omni-directional antenna. Finally the radio interface was the Senao/Engenious SL-2511CD PLUS EXT2 802.111b PCMCIA card. The chosen operating mode for this card was the Pseudo-IBSS mode which is a simplified version of the 802.11 Ad-Hoc mode.

The stations operated on Red Hat 9 Linux, kernel version 2.4.20. For route discovery and packet forwarding, the Click software router toolkit was used. Furthermore a Web, a NAT and a DHCP server was implemented in each station.

The routing protocol of the Roofnet project was the SrcRR, inspired from the Dynamic Source Routing (DSR) protocol. The SrcRR protocol tries to find high throughput paths. It uses the Expected Transmission Count (ETX) metric to choose the best paths. The ETX metric measures the loss rate between a station and its neighbors by periodically broadcasting probe packets. A value is given in each link which is an estimation of the number of times a packet needs to be sent for successful reception.

If a station does not know a path towards the packet's destination, it will broadcast Route Request packets. When a Route Request packet is received by the destination, it will reply with a Route Reply packet towards the sender station, that contains the needed path information.

The rest of the thesis is organized as follows: Chapter 2 offers information about the 802.11 protocol, Chapter 3 describes the 802.11s protocol and the Open80211s implementation, Chapter 4 presents the hardware and software

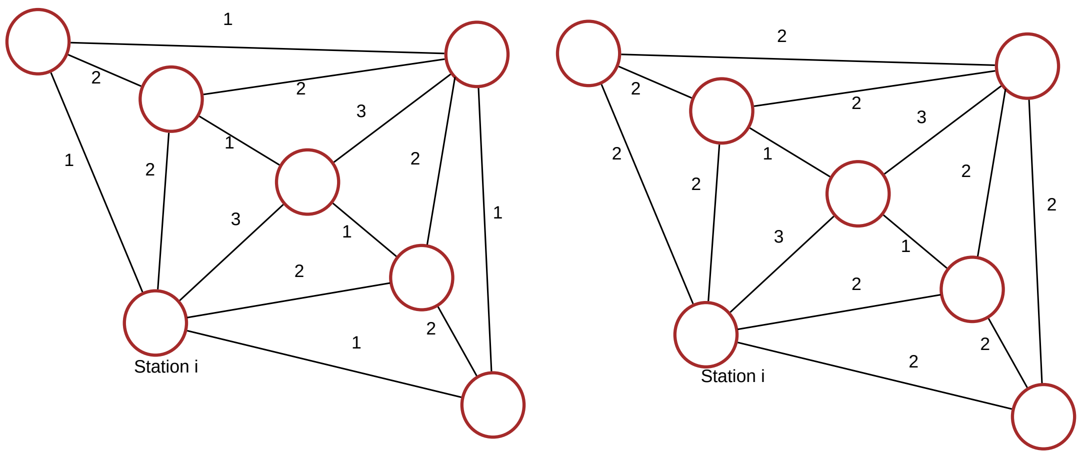


Figure 1.2: Merge Operation of Tabu Algorithm.

used in this project, Chapter 5 provides the conflict graph creation algorithm and two channel assignment algorithms, Chapter 6.2 discusses network deployment issues, Chapter 6 offers simulation and experimental results for the channel assignment algorithms as well as the presentation of a channel probing algorithm for statistics collection. Finally, work is concluded in Chapter 7.

Chapter 2

The IEEE 802.11 Protocol

2.1 General Description

The IEEE 802.11 refers to the set of standards that define communication for wireless local area networks in the 2.4 and 5 GHz frequency bands [9]. A listing of the wireless IEEE standards is shown in Table 2.1. In this project the 802.11b and 802.11s standards were utilized, that refer to wireless multi-hop transmissions between 802.11 stations.

There are several differences between wireless and traditional wired networks. Low cost as well as the lack of cabling have made wireless networks preferable to wired networks in many scenarios such as rooftop networking or home wireless networks. Another great advantage of wireless networks is the ability to serve mobile stations. They do not need to be tied to an Ethernet cable and can roam freely within the wireless local area network range. On the other hand since the propagation medium used by wireless networks for signal transmission is the air, they are less reliable than wired networks. One of the biggest drawbacks of 802.11 networks, is the problem of interference from other co-located 802.11 stations that increases the probability of transmission failure. Another disadvantage of wireless networks is that radio signals have a limited range. They cannot penetrate dense objects and can only project a certain distance before becoming useless. Finally there also exists the problem of fading that refers to the variable change of the

Table 2.1: IEEE 802.11

802.11	The initial release of the standard capable of transmissions of 1 to 2 Mbps and operates in the 2.4 GHz band.
802.11a	Capable of transmissions of up to 54 Mbps and operates in the 5 GHz band.
802.11b	Introduced in 1999, 802.11b is capable of transmissions of up to 11 Mbps and operates in the 2.4 GHz band.
802.11c	Defines wireless bridge operations.
802.11d	International roaming extensions.
802.11e	Defines enhancements to the 802.11 MAC for QoS.
802.11f	Defines Inter Access Point Protocol (IAPP).
802.11g	Capable of transmissions of up to 54 Mbps and operates in the 2.4 GHz band. Backwards compatible with 802.11b.
802.11i	Enhanced security.
802.11j	802.11 extension used in Japan.
802.11n	Higher throughput improvements using multiple-input, multiple-output (MIMO) antennas.
802.11s	Mesh networking.

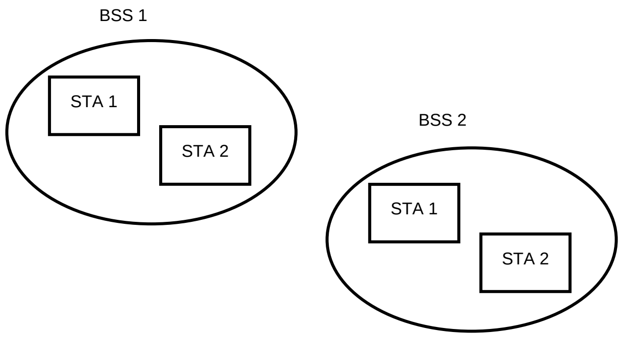


Figure 2.1: Basic Service Sets.

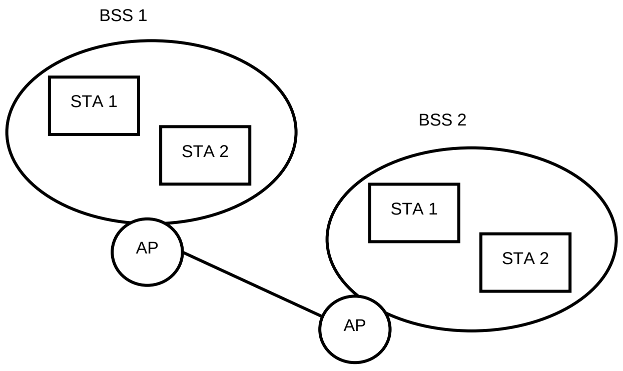


Figure 2.2: Extended Service Set.

attenuation affecting a radio signal over the propagation medium.

The basic structural element of 802.11 architecture is the Basic Service Set (BSS). The Basic Service Set is a set of all stations that can communicate with each other. Fig. 2.1 shows two BSSs with two stations in each one. Each BSS is managed by a base station which is called Access Point (AP).

A wireless network may contain only one BSS with a single AP, but in many cases it consists of several Basic Service Sets. In such cases, the Access Points that serve the BSSs are connected through a backbone network (Distribution System), such as an Ethernet network. An 802.11 network with multiple BSSs, is referred as an Extended Service Set (ESS). An example of an ESS is shown in Fig. 2.2.

Another type of wireless networks are the Ad-Hoc networks. In these the various wireless stations are able to communicate with each other without the use of an Access Point.

2.2 Medium Access Control

The 802.11 standard specifies a common medium access control (MAC) Layer, which provides a variety of functions that support the operation of 802.11-based wireless local area networks. In general the MAC Layer manages and maintains communications between 802.11 stations by coordinating access to a shared radio channel and utilizing protocols that enhance communications over a wireless medium. The data transmission between stations is based on an asynchronous, connectionless and best effort transmission of MAC level frames. There is no guarantee that the frame transmission will be successful.

The medium access for 802.11 stations is based on the Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol. The CSMA/CA protocol defines that prior to transmitting, a wireless station first listens to the shared medium to determine whether another station is transmitting or not. If another station was heard, the first station waits for a period of time before listening again for a free communications channel.

The MAC frames contain a fixed number of fields that appear in a fixed order. An exception to this are the Address and Frame Body fields, that do not appear in all the MAC frames. Fig. 2.3 shows the MAC Layer frame format. The various frame fields are explained below.

Duration/ID: Shows the duration of the current transmission.

Address fields: The MAC frame contains 4 address fields. They are used to show the BSS ID, the transmitter address, the receiver address and in the case of multi-hop networks, the address of the next hop neighbor. The frame type (management, control, data) defines which and how many of the Address fields are used during the frame transmission.

Sequence Control: It contains two subfields,the *sequence number* and the *fragment number*. These fields show the frame's sequence number and the frame's fragment number respectively.

Frame Body: It contains the data.

FCS: It contains the 32 bit Cyclic Redundancy Code (CRC) used for error detection.

The *Frame Control* field contains various subfields that are shown in Fig. 2.4. These subfields are explained below.

Protocol Version: It has a fixed value of 0. Reserved for future use.

Type-Subtype Fields: These two fields both define the functionality of the frame (for example if it is an Acknowledgment frame).

To DS: Its value is 1 if the frame is destined to the Distribution System. Otherwise the value is 0.

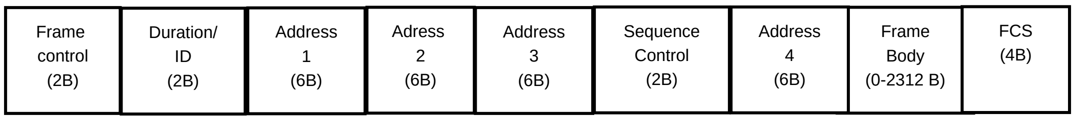


Figure 2.3: 802.11 MAC Frame Format.

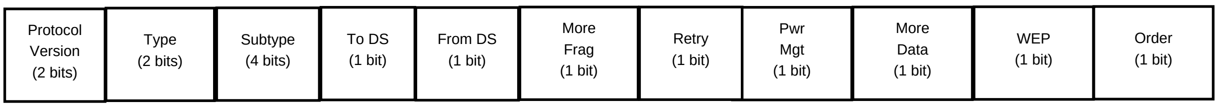


Figure 2.4: Frame Control Field.

From DS: Its value is 1 if the frame is coming from the Distribution System. Otherwise the value is 0.

More Fragments: Its value is 1 if there exist more fragments of the same frame for transmission.

Retry: This field is used to declare that the current frame transmission, is actually a retransmission of a previously failed frame transmission.

Power Management: This field shows whether the station is in power save mode, or not.

More Data: This field is used to inform stations in power save mode, that there exist in the Access Point, data destined to them.

WEP: The field's value is 1 when the WEP algorithm is used on the contents of the *Frame Body* field.

Order: Its value is 1 when the frame is transmitted with the use of the Strictly Ordered Service.

2.3 Distributed Coordination Function

The Distributed Coordination Function (DCF) is based on the CSMA/CA protocol and is the basic function that 802.11 stations use to obtain access to the wireless medium. The CSMA/CA protocol is designed to reduce the probability of collisions when multiple stations are trying to access the medium for transmission.

According to the DCF, a station with a packet to transmit, monitors the channel activity. If the channel is sensed idle for a time period equal to a Distributed Interframe Space (DIFS), the station will attempt to transmit its frame. Otherwise if the channel is busy, the station will continue to monitor the channel until it is idle for a DIFS period. Then, to reduce the probability

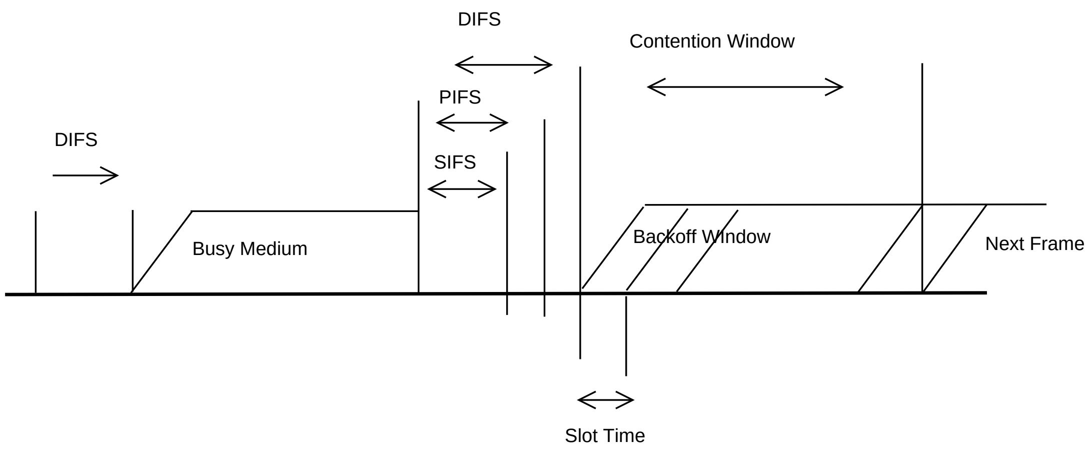


Figure 2.5: Basic Access Method.

of collisions with other stations that are trying to access the medium, the station will produce a random backoff interval. At the end of this interval, the station will transmit, if the channel was sensed idle. This is the collision avoidance part of the CSMA/CA algorithm.

Following a frame transmission, the receiving station must reply with an acknowledgement (ACK frame), to inform the transmitting station that it actually received the frame. If the transmitting station does not receive an ACK, it will conclude that the transmission failed and it will attempt to transmit the frame again. The stations have no information whether the collision occurred during the data or the ACK transmission. Wireless stations cannot sense their own transmission as in the case of wired transmissions. The basic access method is depicted in Fig. 2.5.

2.3.1 Carrier Sense Mechanism

There are two types of carrier sense mechanisms that 802.11 stations apply to find whether the wireless medium is idle or not; physical carrier sense and virtual carrier sense mechanisms. The first category involves the mechanisms used by a station to sense the channel and either transmit its frames or defer access because the channel was sensed busy.

The virtual carrier sense mechanism is provided by the 802.11 MAC and is referred as Network Allocation Vector (NAV). The NAV keeps a prediction of the future channel occupation from other stations, based on the information found in the *Duration/ID* field of the monitored frames. As said earlier the *Duration/ID* field of a frame shows how long the current transmission will last. This also includes the time needed for the acknowledgment transmis-

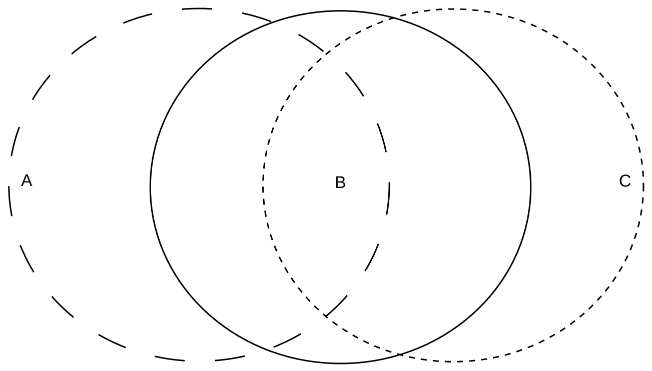


Figure 2.6: Hidden Node Problem.

sion. As long as the NAV informs the station that the channel is occupied, it will not try to transmit any frames.

Another way to update the Network Allocation Vector is through the Request to Send (RTS) and Clear to Send (CTS) frames. The use of those frames is not mandatory in the 802.11 protocol. Before a station begins transmitting a data frame, it may choose to send to the same receiving station a RTS frame, to inform it about the upcoming data transmission. When the receiving station receives the RTS frame, it will find out whether the channel is idle (through its own carrier sensing mechanisms). If the channel is idle, it will send to the first station (that wants to transmit data) a CTS frame. The sending station can begin its data transmission as soon as it receives the CTS frame.

The RTS and CTS frames help solve the hidden node problem (Fig. 2.6). Suppose there exist three wireless stations, A, B and C. Stations A and B are in transmission range. Same with B and C, but station A cannot transmit directly to C. If station A transmits a frame to B and at the same time C attempts to transmit to B, then there will be a collision and both frames will be lost.

If the RTS/CTS frames were used, then station A would first send a RTS to B. If the channel around B was idle, he would respond with a CTS. Station C would also receive this CTS, it would update its NAV and it would not attempt to transmit until the data transmission between A and B was over.

The virtual carrier sense mechanisms are used alongside the physical carrier sense mechanisms. Both these mechanisms reduce the probability of error during data transmission.

2.3.2 Backoff Algorithm

When a station is ready to transmit but through the carrier sensing mechanisms it realizes that the channel is busy, it will then use the backoff algorithm. The station will monitor the channel for a DIFS period, and if during that time the channel was sensed idle, the station will transmit. Otherwise the station keeps monitoring the channel until it is idle for DIFS time and them the backoff algorithm is employed. The algorithm is also used after failed transmissions. In such cases the station must wait for an extra time period before attempting to transmit. This time period is called $Backoff\_Time$ and is given by the formula bellow:

$Backoff\_Time = random * 1\_slot\_time$

where $random$ is a pseudorandom integer chosen from a uniform distribution in $[0, CW]$.

For efficiency reasons, the time following an idle DIFS is slotted and station are allowed to transmit only at the beginning of each time slot. The time slot duration is set as the time needed for a station to detect a packet transmission from any other station. Generally the slot time accounts for the time needed to switch from transmitting to receiving state ($RX\_TX\_Turnaround\_Time$), for the propagation delay and for the time needed to signal the MAC layer about the state of the channel.

The Contention Window (CW) depends on the number of failed transmissions for a given frame. At the first transmission attempt, CW is set equal to CWMin (Minimum Contention Window). After each unsuccessful transmission, the Contention Window is doubled to an upper limit, CWMax (Maximum Contention Window).

As long as the channel is sensed idle, the backoff timer decreases. In case of detected transmission, the timer will freeze and will be reactivated after the channel is idle for DIFS time. When the backoff timer reaches 0, the station will transmit and if the transmission is successful, the CW is set equal to CWMin.

The randomness of the backoff algorithm leads to fewer collisions is cases where the channel is contested among several stations that are attempting to transmit their packets. A simple example of the 802.11 medium access and backoff algorithm is depicted in Fig. 2.5. The station is trying to transmit but senses that the channel is busy. It postpones its transmission and waits until it senses that the channel is idle for DIFS time. Following that, the backoff procedure starts. A random integer is selected and for as many slots the medium is not busy, the backoff timer decreases. As soon as the timer reaches zero, the station transmits.

Table 2.2: Slot Time and Interframe Spaces

Slot Time(µs)	SIFS(µs)	DIFS(µs)	PIFS(µs)
20	10	50	20

2.3.3 Interframe Space

The 802.11 standard defines four types of interframe spaces. These are independent to the station's transmit rate and their value is fixed for a certain physical layer.

Short Interframe Space (SIFS): The Short Interframe Space is used before transmitting ACK frames or before transmitting the next fragment of a frame. The SIFS is the smallest interframe space used in 802.11, so stations waiting for a SIFS have greater priority than others that have to wait for a longer time period.

Point Coordination Interframe Space (PIFS): The Point Coordination Interframe Space is employed in the Point Coordination Function (PCF). PCF is not mandatory in 802.11 networks but it sometimes replaces DCF in infrastructure networks that are served by Access Points. The PIFS defines the time that the APs must monitor the channel before obtaining access. Its value is set as:

$PIFS = SIFS + 1\_slot\_time$.

Distributed Interframe Space (DIFS): The Distributed Interframe Space is used before the transmission of a new frame.

$DIFS = SIFS + (2 * 1\_slot\_time)$

Extended Interframe Space (EIFS): The Extended Interframe Space is the largest among the interframe spaces and is used when there was an error during a frame reception.

Table 2.2 shows the slot time and interframe spaces values for IEEE 802.11b.

2.4 802.11 Frequency

The 802.11b protocol uses the 2.4 GHz frequency band for data transmission. The same band is also used by the 802.11g protocol, so the latter was backwards compatible with 802.11b. There is a total of fourteen channels used by 802.11b/g around the globe, but not all of these channels are available in all countries. These channels are depicted in Table 2.3. In Europe only the first 13 channels are used.

Table 2.3: 802.11b/g Channels and Frequencies

Channel Number	**Lower Freq**	**Center Freq**	**Upper Freq**
1	2.401	2.412	2.423
2	2.406	2.417	2.428
3	2.411	2.422	2.433
4	2.416	2.427	2.438
5	2.421	2.432	2.443
6	2.426	2.437	2.448
7	2.431	2.442	2.453
8	2.436	2.447	2.458
9	2.441	2.452	2.463
10	2.446	2.457	2.468
11	2.451	2.462	2.473
12	2.456	2.467	2.478
13	2.461	2.472	2.483
14	2.473	2.481	2.495

Table 2.4: 802.11a Channels and Frequencies

Band	**ChannelNumber**	**Frequency (MHz)**
Lower Band	36	5180
	40	5200
	44	5220
	48	5240
Middle Band	52	5260
	56	5280
	60	5300
	64	5320
Upper Band	149	5745
	153	5765
	157	5785
	161	5895

By observing Table 2.3, it is clear that only three channels (for example channels 1,6 and 11) can be used simultaneously without causing interference. This is one of the biggest drawbacks of the 802.11b/g since the widespread use of 802.11 devices, makes it difficult for an interference free operation.

On the other hand 802.11a uses the 5 GHz frequency band and does not suffer such an interference problem. The frequency channels employed by 802.11a, are depicted in Table 2.4.

Chapter 3

The 802.11s Mesh Protocol

3.1 General Description

The IEEE 802.11s standard is an extension to the IEEE 802.11 which allows multiple wireless stations to connect with each other without having an Access Point to manage them [10]. The stations can form a multi-hop network where all the links are wireless, thus eliminating the need for wired infrastructure.

802.11s inherently depends on one of 802.11a, 802.11b, 802.11g or 802.11n carrying the actual data. It defines three types of nodes: *Mesh Point (MP)*, *Mesh Portal (MPP)* and *Mesh Access Point (MAP)*. All these nodes have frame forwarding capability and can thus forward frames originated at some node and destined for some other node.

The majority of wireless devices in a mesh network consists of *Mesh Points*. *Mesh Points* are able to discover neighboring *Mesh Points* by transmitting beacons. They also employ a hybrid routing protocol to communicate with stations that are further than one hop away.

Mesh Portals are 802.11s stations that are connected both to the internet and to the mesh network. They provide cheap internet access to the other stations.

Mesh Access Points are traditional Access Points augmented with mesh functionality. They can be a part of the mesh network and serve as an AP at the same time.

3.2 Path Selection in 802.11s

The *Hybrid Wireless Mesh Protocol (HWMP)* is the default routing protocol used by 802.11s wireless devices [11]. It works at MAC layer and contains both reactive and proactive routing components. The protocol defines four control packets: *Root Announcement (RANN)*, *Path Request (PREQ)*, *Path Reply (PREP)* and *Path Error (PERR)*. The *RANN*, *PREQ* and *PREP* control packets contain 3 important fields: *Destination Sequence Number (DSN)*, *Time to Live (TTL)* and metric. The *DSN* and *TTL* are used to

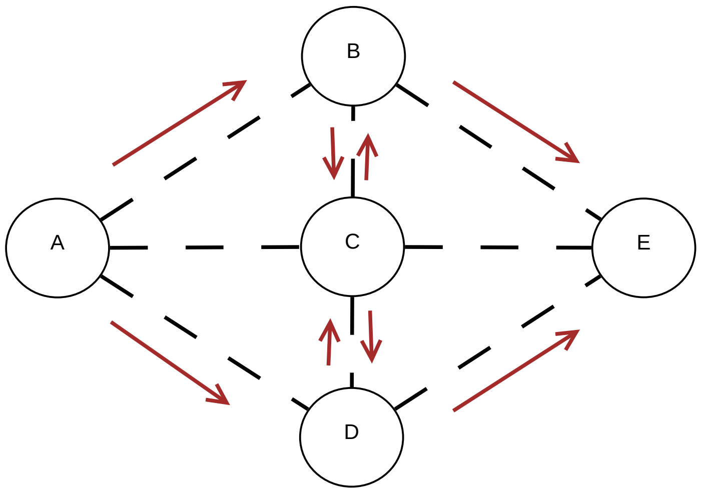


Figure 3.1: HWMP Reactive PREQ.

avoid the *count to infinity* problem. The metric is used to help the 802.11s stations to find better routes to their destination and not just rely on hop count.

The HWMP has two modes of operation. The reactive mode which is always enabled and cannot be disabled, and the proactive mode (not automatically enabled) that forms a routing tree with a specific mesh point as a root.

The reactive part of HWMP is based on *Radio-Metric Ad hoc On Demand Distance Vector (RM-AODV)*. When a station has data to send, it broadcasts a *PREQ* packet that contains the destination's MAC address. The intermediate stations that receive this *PREQ*, update their metric and path towards the source and re broadcast the *PREQ* packet. When the destination receives the *PREQ* packets, it checks the different metric and picks the best path towards the station that initialized the route discovery. Then the destination creates and sends a unicast *PREP* packet towards the source. Intermediate stations that receive this *PREP* packet, update their metric and path towards the destination and then forward the *PREP* packet. When the source receives the *PREP* from the destination, communication commences.

The intermediate nodes that receive the *PREQ* packet first check if the path towards the destination exists in their routing table. If this is the case they do not need to re broadcast the *PREQ*. This way the network flooding is prevented. Fig. 3.1 shows an example of a path request from station *A* to station *E* in a 802.11s network. The path reply mechanism is depicted in Fig. 3.2, assuming that the best route is *A*-*B*-*E*.

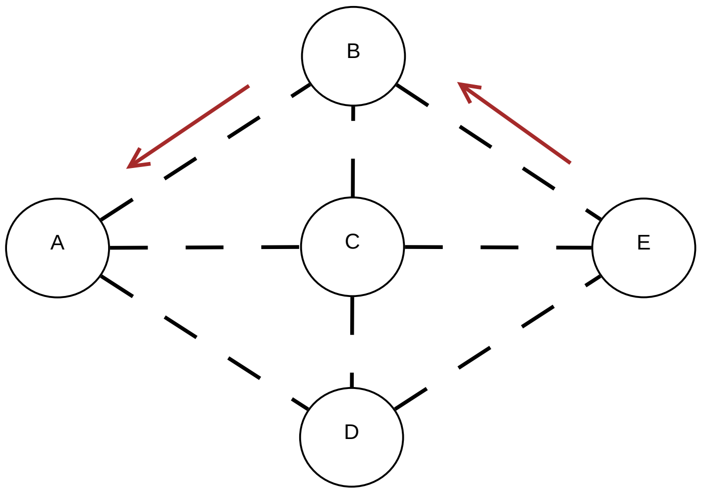


Figure 3.2: HWMP Reactive PREP.

In some scenarios a large proportion of the traffic will be destined to a single or a few stations. For example most of the traffic will be forwarded through a mesh portal that offers internet access. Proactive routing is useful is these kind of situations.

In order to use the HWMP proactive routing, a mesh station must be set up as a root. A tree then is formed with said station as the root. There are two ways to create the tree. One involves the use of proactive *PREQ* packets and the other the use or *RANN*.

In the first method, the root station periodically broadcasts *RREQ* packets with unique sequence number. The other mesh stations that receive the *RREQ*, record the routing metric, update the *RREQ* packet (TTL, Hop count,metric), rebroadcast the *PREQ* and create an inverse path towards the tree root. A forward path from the root to the other stations is established when the Proactive RREP flag in the *PREQ* is set to 1 by the root. In this case the receiving stations send a unicast *PREP* to the root so that a bidirectional link is available proactively. To minimize the routing overhead, forward paths are only created when there are data to send.

In the second method, the root periodically broadcasts *RANN* packets. *RANN* packets are only used to disseminate the path metrics and do not create or update the routing table. When a mesh station needs to create a path towards the tree root, it sends a unicast *PREQ* packet to the root. The path traveled by the *PREQ* packet is based on the routing metrics extracted from the *RANN* packets. Upon receiving the unicast *PREQ*, the root replies with a *RREP* and the a bidirectional path is established. A routing tree example of Fig. 3.1, is shown in Fig. 3.3 (assuming that the root is station

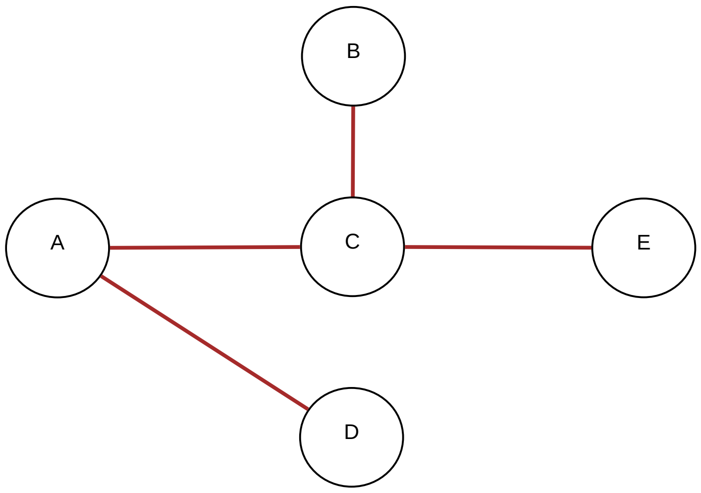


Figure 3.3: HWMP Routing Tree.

Table 3.1: HWMP Airtime Metric Parameters

Parameter	**802.11a Value**	**802.11b Value**	**Description**
O_{ca}	75 µs	335 µs	Channel access overhead.
O_p	110 µs	364 µs	Protocol overhead.
B_t	8224	8224	Number of bits in test frame.
r			Current bit rate in use.
e_{pt}			Packet error rate at the current bit rate.

A).

The use of proactive routing is not mandatory in HWMP but it is very efficient in fixed mesh networks. On the other hand, reactive routing offers great flexibility in changing environments. The combination of both routing methods makes HWMP suitable for employment in many different network configurations.

In both routing modes, mesh stations monitor the link status of next hops in active routes. When a link failure in an active route is detected, a *PERR* packet is used to notify the other stations about the link break. This is a broadcast packet to spread faster the information about the link failure.

The default metric used in the HWMP protocol is the *airtime* metric. It reflects the amount of channel resource consumed for transmitting a frame over a particular link. The metric (C_a) is calculated as:

$C_a = [O_{ca} + O_p + \frac{B_t}{r}] * [\frac{1}{1-e_{pt}}]$

The parameters used in *airtime* metric calculation are shown in Table 3.1.

3.3 The Open802.11s Implementation

Open802.11s is a vendor neutral, open source implementation of the 802.11 protocol for the Linux operating system. The goal of this project is to keep track of the 802.11s draft and support the interoperability of different 802.11s implementations. 802.11s introduces only minimal changes to the MAC layer so the 802.11s stack can be fully implemented in software and made to run in legacy 802.11 wireless cards. Hybrid Wireless Mesh Protocol is the implemented routing protocol in Open802.11s.

The Open802.11s implementation allows advanced tinkering of the routing paths as well as several mesh parameters such as the *TTL* and the root mode (proactive *PREQ* or *RANN*).

Besides the implementation of the 802.11s stack, several other Linux Kernels have been implemented, that offer various functionalities to the mesh stations. The most important of these implementations are described below.

Power Saving Kernel: This is a design to allow a longer uptime for battery powered mesh devices. In time of no activity the radio receiver is switched off to conserve power. The data transmission is carried in bursts which allow to power down the receiver in times of inactivity.

Channel Switching Kernel: Mesh stations usually have to operate in the same frequency channel after joining the mesh network. This kernel uses a protocol defined in IEEE 802.11s to mitigate the interference caused from other neighboring 802.11 devices. Any mesh station is allowed to trigger the mesh channel switching attempt by informing all the other stations in the network to switch to a new frequency channel. Since all the stations switch to the new operating channel, this implementation does not solve the problem of internal interference caused by the same stations that form the mesh network.

Multi-Channel Kernel: In this implementation each station's radio interface joins the mesh network on a given channel and the kernel forwards frames between the interfaces to present a single unified mesh among the stations and across the channels. The kernel automatically creates a bridge between the several radio interfaces. The HWMP is used for loop avoidance instead of a non-wireless protocol such as *Spanning Tree Protocol (STP)*, that is used in software bridge approaches. With the correct frequency allocation among the different radio interfaces, this implementation helps mitigate both the problem of internal and external interference. This kernel was installed on all the mesh stations used in this project.

Chapter 4

Hardware and Software

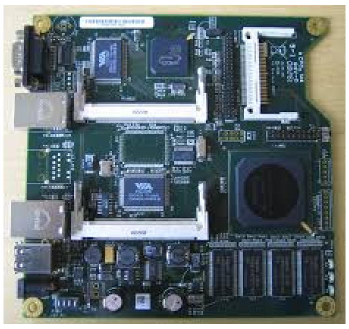
4.1 The Alix Board

The Alix boards are small form factor system boards optimized for wireless routing and network security applications. In this project two types of Alix boards were used. The Alix 3d2 and the Alix 2d2 board.

The Alix 2d2 board has two LAN, two minipci and two USB slots. The required power supply is 7 to 20V DC, about 3 to 4W at Linux idle, peak about 6W without minipci cards or USB devices. The board is shown in Fig. 4.1. The board's dimensions are 152.4 x 152.4mm.

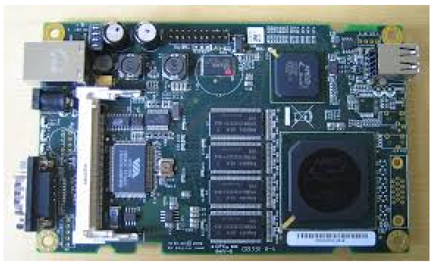
The Alix 3d2 board has one LAN, two minipci and two USB slots. The required power supply is 7 to 20V DC, about 2.5 to 3.5W at Linux idle, peak about 5W without minipci cards or USB devices. The board is shown in Fig. 4.2. The board's dimensions are 100 x 160mm.

For the radio interfaces the TP-LINK TL-WN360G (Fig. 4.3) and TL-WN861N (Fig. 4.4) mini-pci cards were used. When a node needed more radio interfaces, the TP-LINK TL-WN7200ND USB card was used (Fig. 4.5).


(a) Alix 2d2 Front.

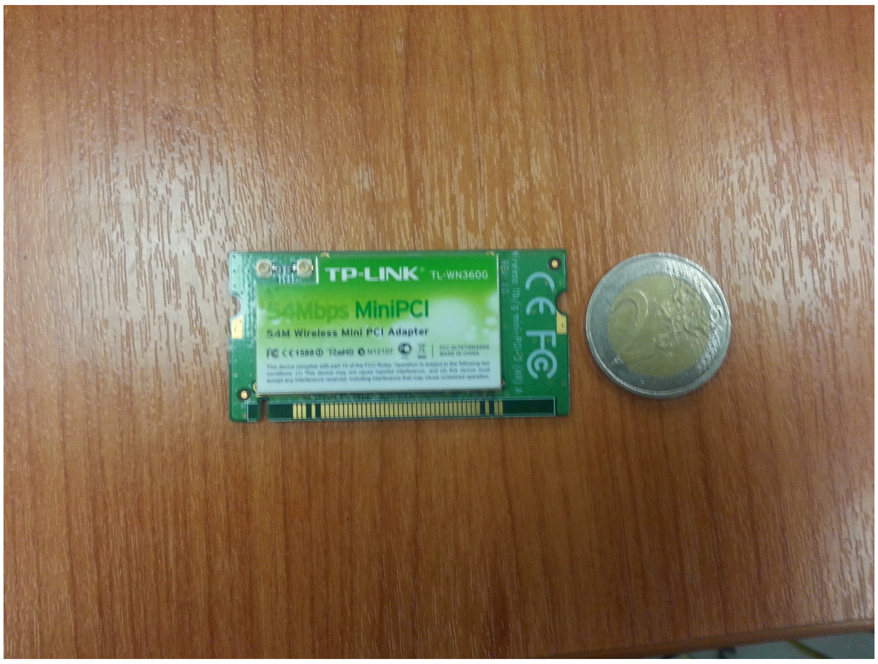
(b) Alix 2d2 Back.

Figure 4.1: Alix 2d2.

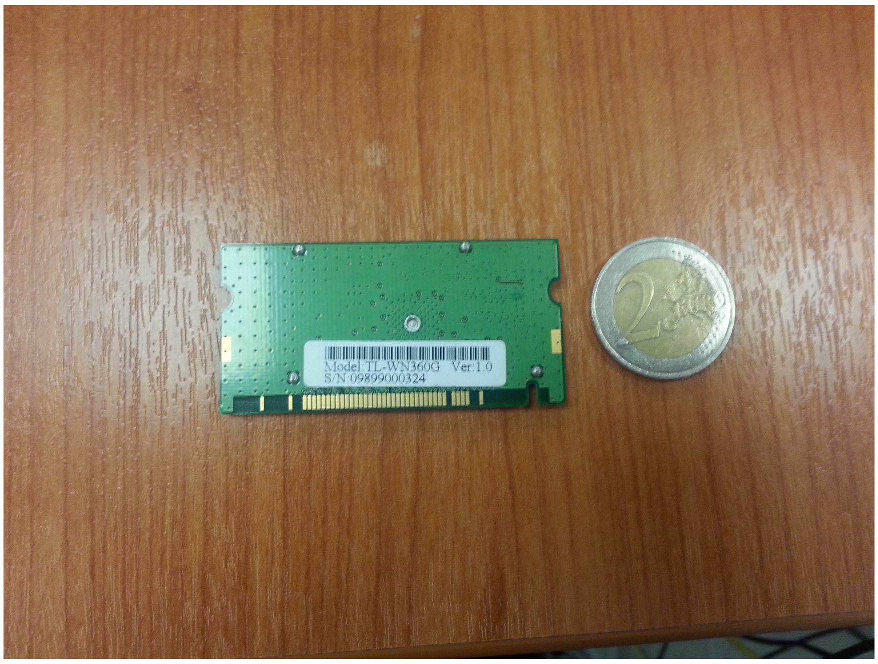

(a) Alix 3d2 Front.

(b) Alix 3d2 Back.

Figure 4.2: Alix 3d2.

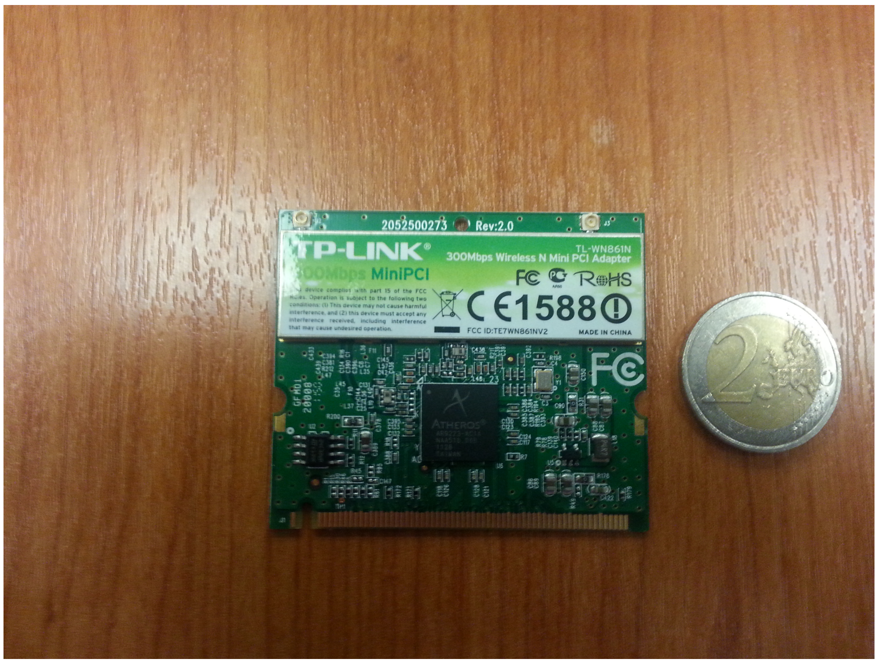


(a) TL-WN360G Front.



(b) TL-WN360G Back.

Figure 4.3: TL-WN360G.



(a) TL-WN861N Front.

(b) TL-WN861N Back.

Figure 4.4: TL-WN861N.

Figure 4.5: TP-LINK TL-WN7200ND.

4.2 Creation of a Network Node

In this section, the steps for creating a mesh network node with an Alix board are described. The needed tools are an Alix board, one RS-232 null modem cable, one RS-232 serial to usb adapter, an Ethernet cable, a card reader, a compact flash card (4 GB or more), a Linux machine and several 802.11 mini-pci or usb cards.

Firstly a Voyage Linux software package needs to be downloaded to a Linux machine. The software tarball is extracted with the following command:

tar –numeric-owner -jxf voyage.tar.bz2

Then the compact flash (cf) card is connected to the card reader and the latter is plugged to the linux machine. To find the device name of the cf card, the following commands needs to be executed in a terminal:

fdisk -l

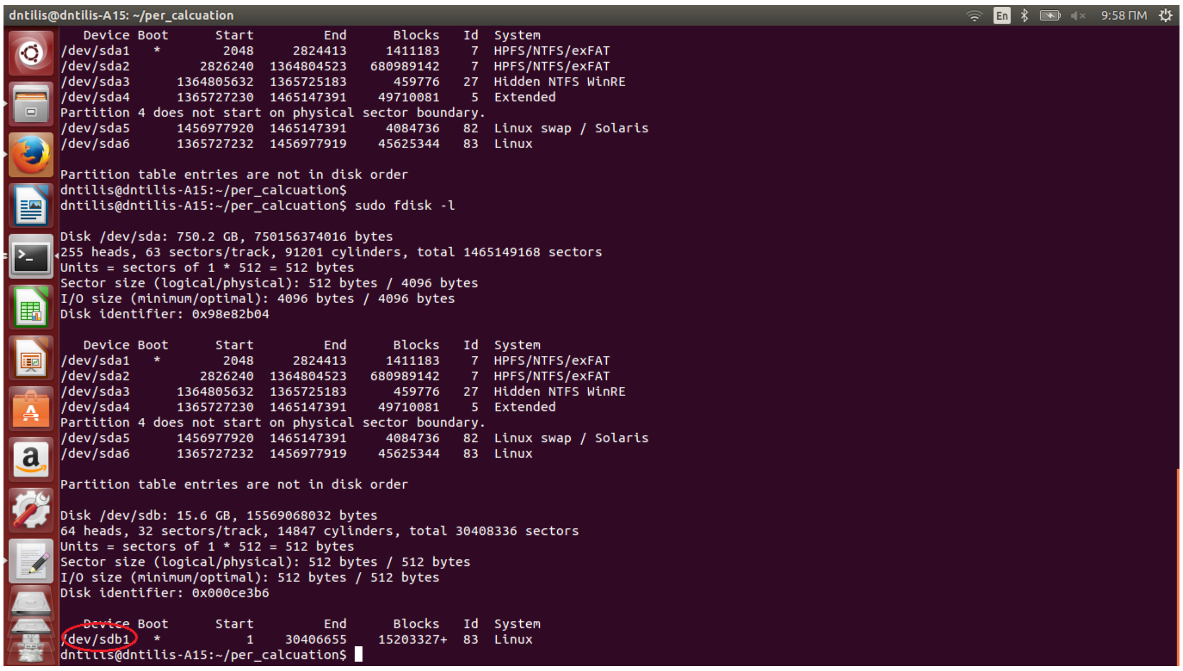
For example in Fig. 4.6, the device name is /dev/sdb.

When the device name is found, we need to change to the newly created Voyage Linux directory and execute the following commands:

./usr/local/sbin/format-cf.sh /dev/sdb

./usr/local/sbin/voyage.update

The first from the above commands formats the cf card and creates an



```
dntilis@dntilis-A15: ~/per_calcuation
   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2824413     1411183    7  HPFS/NTFS/exFAT
/dev/sda2         2826240  1364804523   680989142    7  HPFS/NTFS/exFAT
/dev/sda3      1364805632  1365725183      459776   27  Hidden NTFS WinRE
/dev/sda4      1365727230  1465147391    49710081    5  Extended
Partition 4 does not start on physical sector boundary.
/dev/sda5      1456977920  1465147391     4084736   82  Linux swap / Solaris
/dev/sda6      1365727232  1456977919    45625344   83  Linux

Partition table entries are not in disk order
dntilis@dntilis-A15:~/per_calcuation$
dntilis@dntilis-A15:~/per_calcuation$ sudo fdisk -l

Disk /dev/sda: 750.2 GB, 750156374016 bytes
255 heads, 63 sectors/track, 91201 cylinders, total 1465149168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk identifier: 0x98e82b04

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2824413     1411183    7  HPFS/NTFS/exFAT
/dev/sda2         2826240  1364804523   680989142    7  HPFS/NTFS/exFAT
/dev/sda3      1364805632  1365725183      459776   27  Hidden NTFS WinRE
/dev/sda4      1365727230  1465147391    49710081    5  Extended
Partition 4 does not start on physical sector boundary.
/dev/sda5      1456977920  1465147391     4084736   82  Linux swap / Solaris
/dev/sda6      1365727232  1456977919    45625344   83  Linux

Partition table entries are not in disk order

Disk /dev/sdb: 15.6 GB, 15569068032 bytes
64 heads, 32 sectors/track, 14847 cylinders, total 30408336 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000ce3b6

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1   *           1    30406655    15203327+  83  Linux
dntilis@dntilis-A15:~/per_calcuation$
```

Figure 4.6: Compact Flash Card Device Name.

ext2 file system and the second command installs Voyage Linux on the cf card. For the correct installation of the software package, the choices shown in Fig. 4.7, Fig. 4.8, Fig. 4.9 and Fig. 4.10 need to be made.

It is important to note that the */mnt/cf* directory must be manually created.

When the installation is completed, the cf card can be loaded on an Alix board and the latter can now be booted. With the use of the RS-232 null modem cable and the RS-232 to usb adapter, the board can be connected to the Linux machine (Fig. 4.11). The following command is executed on a terminal on the Linux machine and it grants access to the Alix board.

cu -l /dev/ttyUSB0 -s 38400

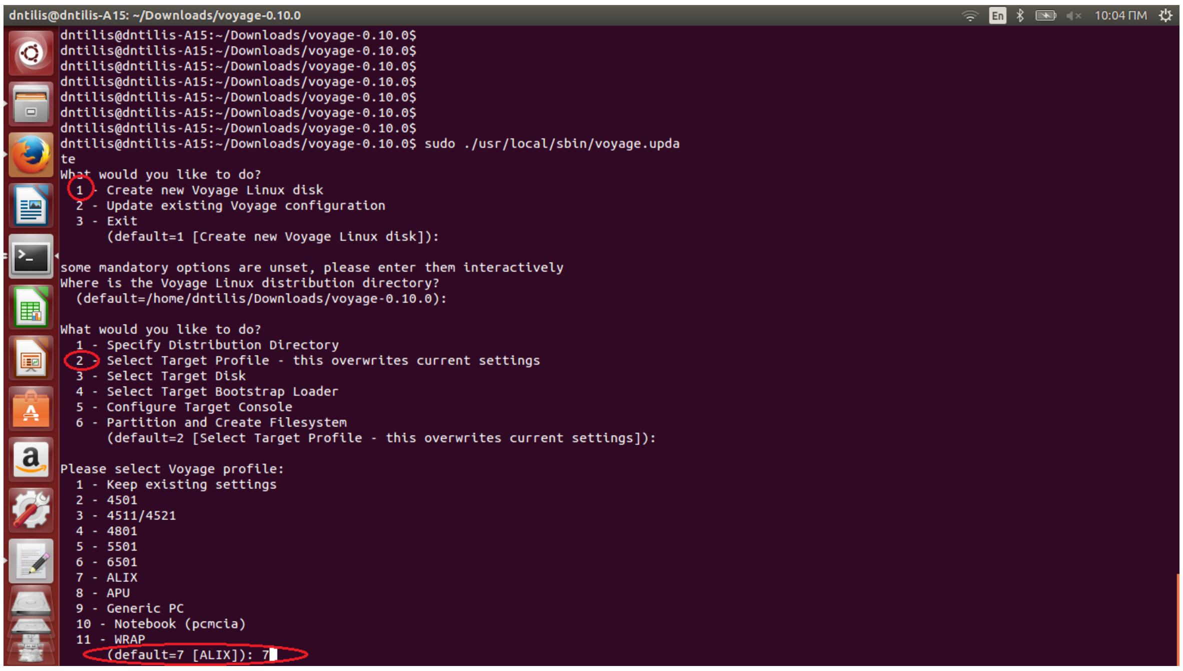
The board must be connected via Ethernet to the Internet and the following packages must be installed:

apt-get install build-essential libnl-genl-3-dev pkg-config gcc nano bridge-utils firmware-ralink

Each time the board boots, the compact flash is in read-only mode. To make changes to files that reside on the board, the compact flash should be set to read-write mode. This is accomplished with the command:

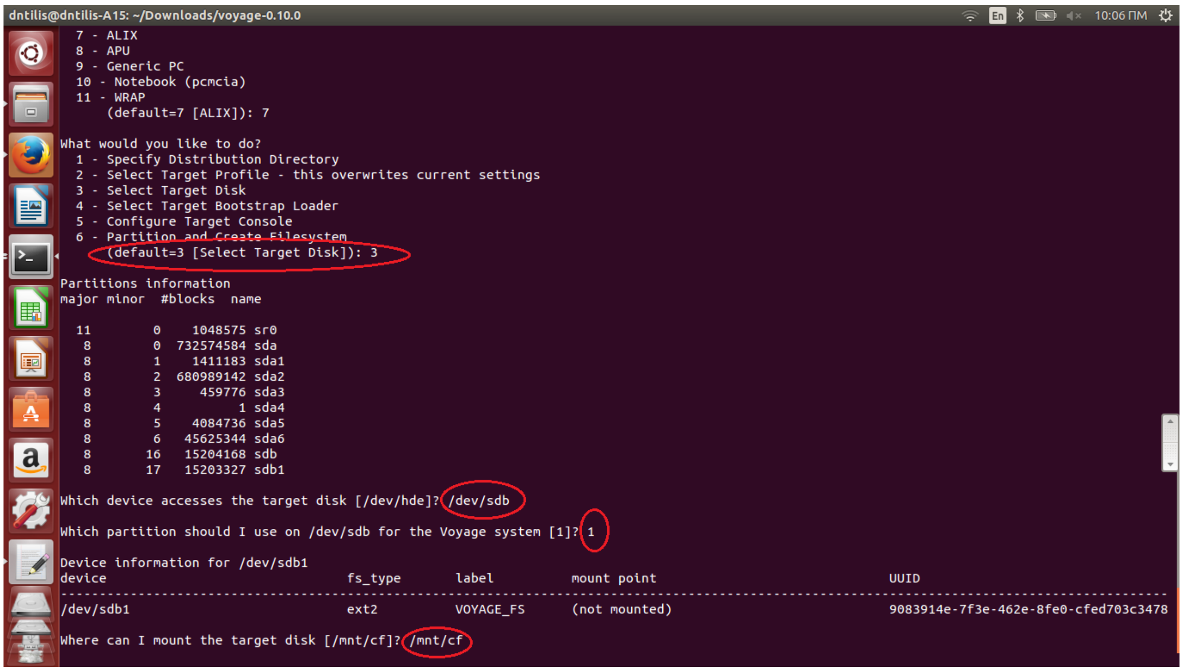
remountrw

This commands needs to be executed each time the board boots. To



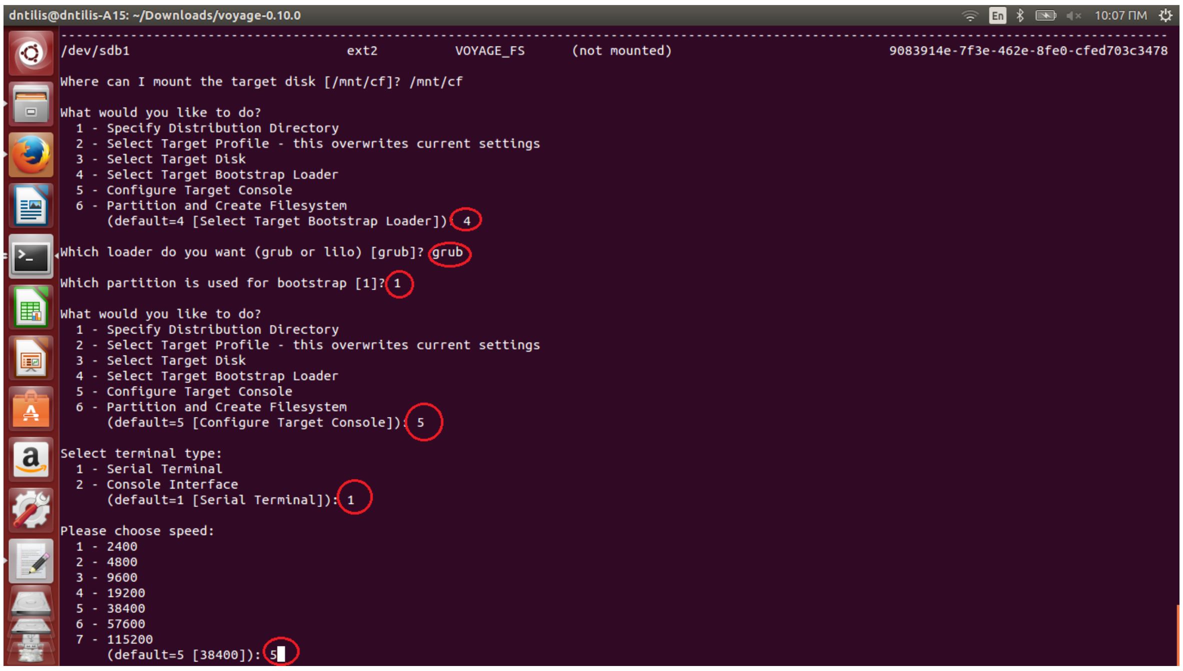
```
dntilis@dntilis-A15: ~/Downloads/voyage-0.10.0
dntilis@dntilis-A15:~/Downloads/voyage-0.10.0$
dntilis@dntilis-A15:~/Downloads/voyage-0.10.0$
dntilis@dntilis-A15:~/Downloads/voyage-0.10.0$
dntilis@dntilis-A15:~/Downloads/voyage-0.10.0$
dntilis@dntilis-A15:~/Downloads/voyage-0.10.0$
dntilis@dntilis-A15:~/Downloads/voyage-0.10.0$
dntilis@dntilis-A15:~/Downloads/voyage-0.10.0$
dntilis@dntilis-A15:~/Downloads/voyage-0.10.0$ sudo ./usr/local/sbin/voyage.upda
te
What would you like to do?
  1 - Create new Voyage Linux disk
  2 - Update existing Voyage configuration
  3 - Exit
      (default=1 [Create new Voyage Linux disk]):

some mandatory options are unset, please enter them interactively
Where is the Voyage Linux distribution directory?
  (default=/home/dntilis/Downloads/voyage-0.10.0):

What would you like to do?
  1 - Specify Distribution Directory
  2 - Select Target Profile - this overwrites current settings
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
  6 - Partition and Create Filesystem
      (default=2 [Select Target Profile - this overwrites current settings]):

Please select Voyage profile:
  1 - Keep existing settings
  2 - 4501
  3 - 4511/4521
  4 - 4801
  5 - 5501
  6 - 6501
  7 - ALIX
  8 - APU
  9 - Generic PC
  10 - Notebook (pcmcia)
  11 - WRAP
      (default=7 [ALIX]): 7
```

Figure 4.7: Installing Voyage Linux on Alix(1).



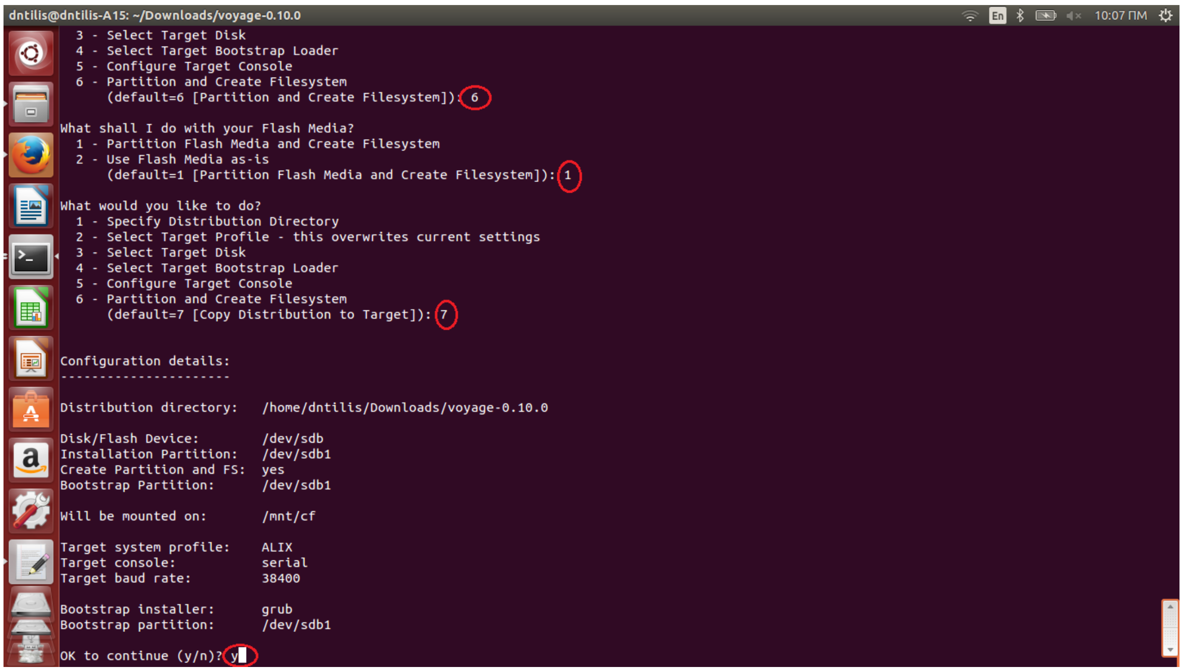
```
dntilis@dntilis-A15: ~/Downloads/voyage-0.10.0
  7 - ALIX
  8 - APU
  9 - Generic PC
  10 - Notebook (pcmcia)
  11 - WRAP
      (default=7 [ALIX]): 7

What would you like to do?
  1 - Specify Distribution Directory
  2 - Select Target Profile - this overwrites current settings
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
  6 - Partition and Create Filesystem
      (default=3 [Select Target Disk]): 3

Partitions information
major minor  #blocks  name

  11        0    1048575 sr0
   8        0  732574584 sda
   8        1    1411183 sda1
   8        2  680989142 sda2
   8        3     459776 sda3
   8        4          1 sda4
   8        5    4084736 sda5
   8        6   45625344 sda6
   8       16   15204168 sdb
   8       17   15203327 sdb1

Which device accesses the target disk [/dev/hde]? /dev/sdb

Which partition should I use on /dev/sdb for the Voyage system [1]? 1

Device information for /dev/sdb1
device                              fs_type       label          mount point                                 UUID
-------------------------------------------------------------------------------------------------------------------------------------
/dev/sdb1                           ext2          VOYAGE_FS      (not mounted)                               9083914e-7f3e-462e-8fe0-cfed703c3478

Where can I mount the target disk [/mnt/cf]? /mnt/cf
```

Figure 4.8: Installing Voyage Linux on Alix(2).



```
-----------------------------------------------------------------------------------------------------------------------------------
/dev/sdb1                          ext2          VOYAGE_FS      (not mounted)                            9083914e-7f3e-462e-8fe0-cfed703c3478

Where can I mount the target disk [/mnt/cf]? /mnt/cf

What would you like to do?
  1 - Specify Distribution Directory
  2 - Select Target Profile - this overwrites current settings
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
  6 - Partition and Create Filesystem
      (default=4 [Select Target Bootstrap Loader]): 4

Which loader do you want (grub or lilo) [grub]? grub

Which partition is used for bootstrap [1]? 1

What would you like to do?
  1 - Specify Distribution Directory
  2 - Select Target Profile - this overwrites current settings
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
  6 - Partition and Create Filesystem
      (default=5 [Configure Target Console]): 5

Select terminal type:
  1 - Serial Terminal
  2 - Console Interface
      (default=1 [Serial Terminal]): 1

Please choose speed:
  1 - 2400
  2 - 4800
  3 - 9600
  4 - 19200
  5 - 38400
  6 - 57600
  7 - 115200
      (default=5 [38400]): 5
```

Figure 4.9: Installing Voyage Linux on Alix(3).



```
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
  6 - Partition and Create Filesystem
      (default=6 [Partition and Create Filesystem]): 6

What shall I do with your Flash Media?
  1 - Partition Flash Media and Create Filesystem
  2 - Use Flash Media as-is
      (default=1 [Partition Flash Media and Create Filesystem]): 1

What would you like to do?
  1 - Specify Distribution Directory
  2 - Select Target Profile - this overwrites current settings
  3 - Select Target Disk
  4 - Select Target Bootstrap Loader
  5 - Configure Target Console
  6 - Partition and Create Filesystem
      (default=7 [Copy Distribution to Target]): 7


Configuration details:
----------------------

Distribution directory:   /home/dntilis/Downloads/voyage-0.10.0

Disk/Flash Device:        /dev/sdb
Installation Partition:   /dev/sdb1
Create Partition and FS:  yes
Bootstrap Partition:      /dev/sdb1

Will be mounted on:       /mnt/cf

Target system profile:    ALIX
Target console:           serial
Target baud rate:         38400

Bootstrap installer:      grub
Bootstrap partition:      /dev/sdb1

OK to continue (y/n)? y
```

Figure 4.10: Installing Voyage Linux on Alix(4).

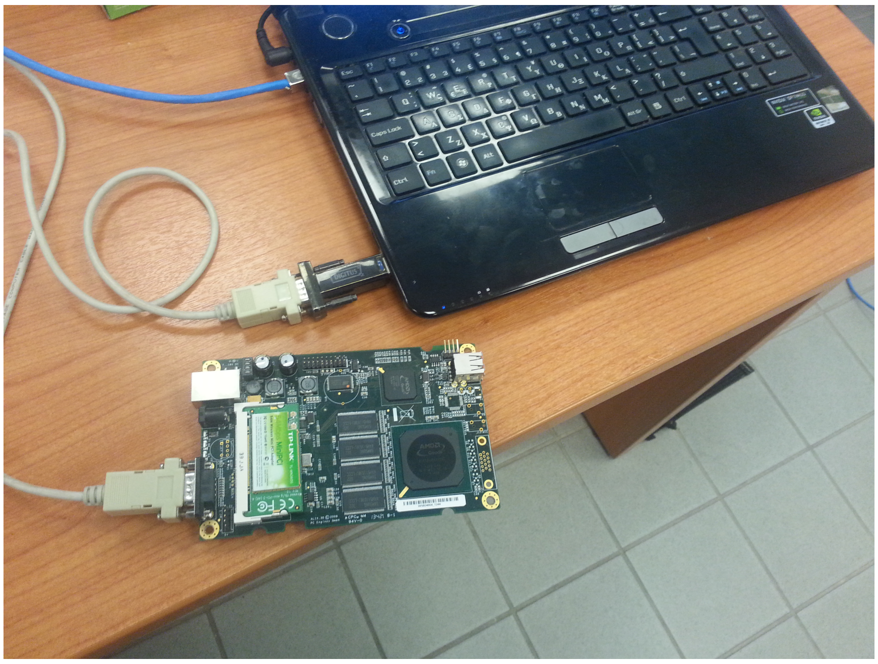


Figure 4.11: Alix Board Connected to a Linux Machine.

make the change permanent, we must find the */etc/init.d/voyage-util* file and comment the line containing */usr/local/sbin/remountro*.

The next step is to compile the Open80211s multi-channel kernel that is to be installed on the Alix board. Firstly the software packages of the kernel must be downloaded on the Linux machine. From the kernel directory we run in a terminal the command that allows us to choose which drivers we want to be included in the kernel:

```
make menuconfig
```

It is important to include the drivers needed for the correct functionality of the wireless cards. In our case, the Atheros drivers should be included. To clean the source tree and begin the kernel compilation we execute:

```
make-kpkg clean
fakeroot make-kpkg –initrd kernel_image
```

The kernel compilation is depended on the Linux machine and may take some time. When it is completed, a .deb file should be created (kernel.deb). This file should copied to the Alix board and it can be installed with the following command:

```
dpkg -i kernel.deb
```

This deb file can be used on future network nodes without demanding further kernel compilations. After the installation is complete, the board can be rebooted with the new multi-channel kernel. For automatic selection of the correct kernel during boot, the */boot/grub/menu.lst* file must be configured. In that file there is the *default* parameter. The value of this parameter defines the default boot kernel.

Now it is time for the configuration of the wireless interfaces. Suppose that a node with only one radio is needed. In the */home* directory the *mesh_setup* file is created:

```
touch /home/mesh_setup
```

The *mesh_setup* file contains the following commands:

```
iw wlan0 interface add mesh0 type mp
iw mesh0 set channel X
ifconfig mesh0 ip_addr up
iw mesh0 mesh join mesh_network_name
```

The first of the above commands configures the wireless card *wlan0* as a mesh point. The second command assigns a channel to the mesh point. The third command assigns an Internet Protocol address and the fourth command informs the mesh point about the mesh network it should connect to.

To setup the board so it automatically creates the mesh point after each boot, the following command is added in the */etc/rc.local* file:

```
./home/mesh_setup
```

In the case of a gateway, the *mesh_setup* file must also contain the following:

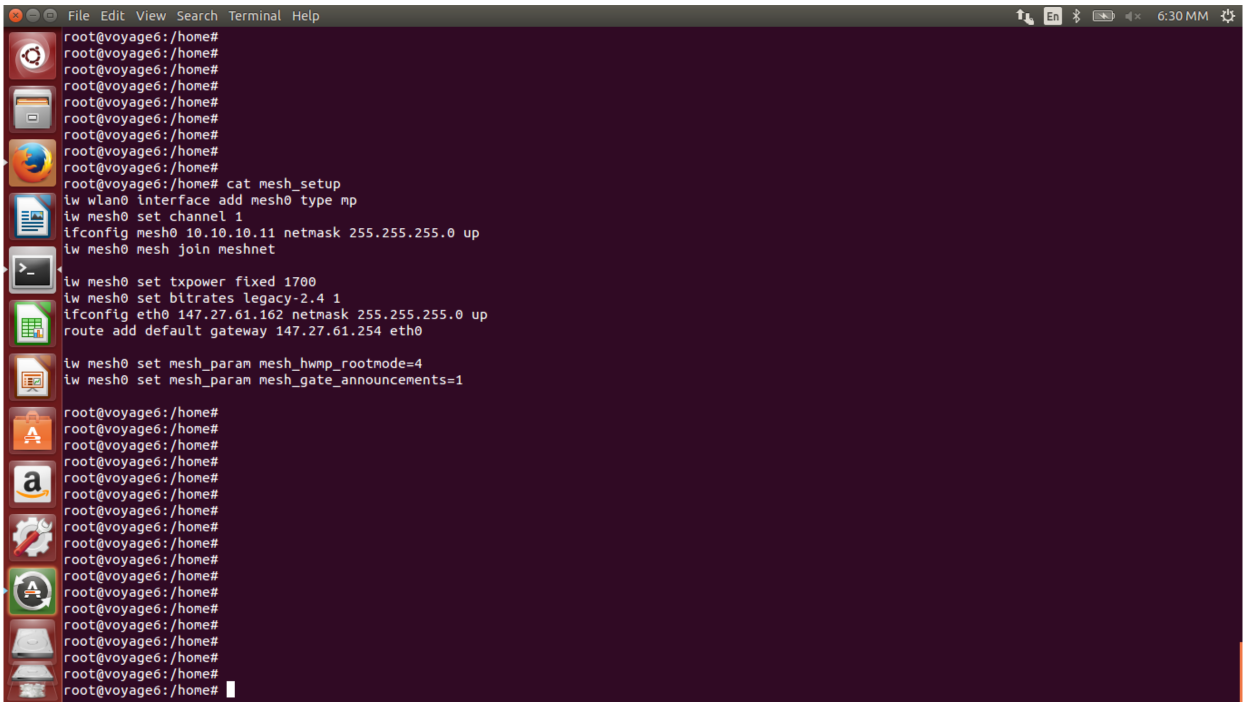
```
iw mesh0 set mesh_param mesh_hwmp_rootmode=4
iw mesh0 set mesh_param mesh_gate_announcements=1
```

The above two commands allow the gateway to be the root in the HWMP protocol and enable gate announcements. An example of a *mesh_setup* file, is shown at Fig 4.12.

Bellow are some additional packages that need to be included in the board:

```
apt-get install i2c-tools python python-smbus python-serial python-psycopg2
lm-sensors ntp ntpdate git htop crda
```

In the */etc/modules* file the line *i2c-dev* should also be added.



```
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home# cat mesh_setup
iw wlan0 interface add mesh0 type mp
iw mesh0 set channel 1
ifconfig mesh0 10.10.10.11 netmask 255.255.255.0 up
iw mesh0 mesh join meshnet

iw mesh0 set txpower fixed 1700
iw mesh0 set bitrates legacy-2.4 1
ifconfig eth0 147.27.61.162 netmask 255.255.255.0 up
route add default gateway 147.27.61.254 eth0

iw mesh0 set mesh_param mesh_hwmp_rootmode=4
iw mesh0 set mesh_param mesh_gate_announcements=1

root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
root@voyage6:/home#
```

Figure 4.12: Mesh Setup File.

4.3 Packet Error Rate Measurements

To determine the functionality and reliability of the network links, packet error rate tests should be conducted. For this reason two algorithms were created, written in C. One algorithm for the gateways and the other one for the rest of the nodes.

The algorithm for the gateways is shown in Fig. 4.13. In the algorithm there exist two different threads which listen for incoming udp packets on different ports. Thread A listens for incoming data packets and thread B listens for incoming scan packets.

Thread A saves the sequence number of the last successful transmission of any given network node. When it receives a new data packet, it compares the sequence number contained in the data packet with the saved sequence number. If the new sequence number is more than one greater than the saved one, it means that the gateway did not receive any packet from the previous batch of udp packets. In that case the gateway calculates the packet error rate of the previous transmissions and waits for a fixed amount of time needed for the transmission of the current batch of udp data packets. After that fixed amount the gateway determines the packet error rate of the current transmission based on the number of packets it has received. From each batch the gateway expects to receive a certain amount of packets. If the new sequence number is exactly one greater than the saved one, then the gateway waits for the fixed amount of time and then calculates the packet error rate of the current transmission. Then Thread A saves the calculated results in a log file.

Thread B is used for receiving the scan packets that the network nodes might send. Each scan packet received contains the calculation of the interference factor for all 802.11b channels. The transmission of scan packets is optional, so Thread B may not be used.

The algorithm for the non gateway nodes is shown in Fig. 4.14. Again there exist two different threads, Thread A and Thread B. Thread A is responsible for the transmission of the data packets that are used for calculation of the packet error rate. Thread B is used for the calculation and transmission of the interference factor.

Thread A is invoked every X seconds. It updates the sequence number and then it sends a batch of K udp packets to the gateway. The packet error rate results depend on how many of those packets the gateway receives. It is important to note that the packet error rate calculations are based on the whole path to the gateway traveled by the probe packets.

Thread B is invoked every Y seconds. It waits for any ongoing transmissions to be completed and then starts the scan procedure. During the scan procedure the node is not able to initiate data transmission towards the gateway with Thread A. The scanning consists of executing the following command in each available 802.11b channel:

iw mesh0 survey dump

This command returns the time the channel was sensed busy during the scanning procedure. Comparing this time with the overall scanning time for a given channel, is the base for the calculation of the interference factor.

$interference\_factor = \frac{busy\_time}{active\ time}$

When the interference factor for the desired channels is calculated, is sent to the gateway. The functionality of Thread B is optional and we may choose not to conduct scans at all. The interference factor may be used in the future for frequency allocation in graph coloring algorithms.

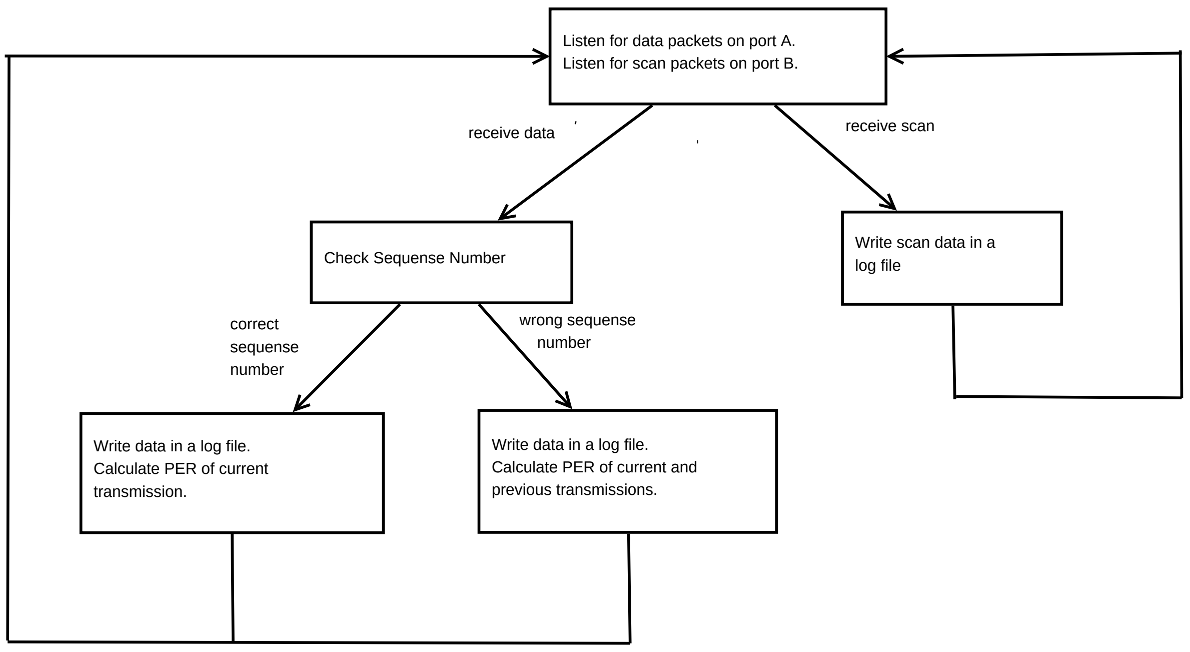

Listen for data packets on port A.
Listen for scan packets on port B.
receive data
receive scan
Check Sequense Number
Write scan data in a log file
correct sequense number
wrong sequense number
Write data in a log file. Calculate PER of current transmission.
Write data in a log file. Calculate PER of current and previous transmissions.

Figure 4.13: Packet Error Rate Algorithm on Gateways.

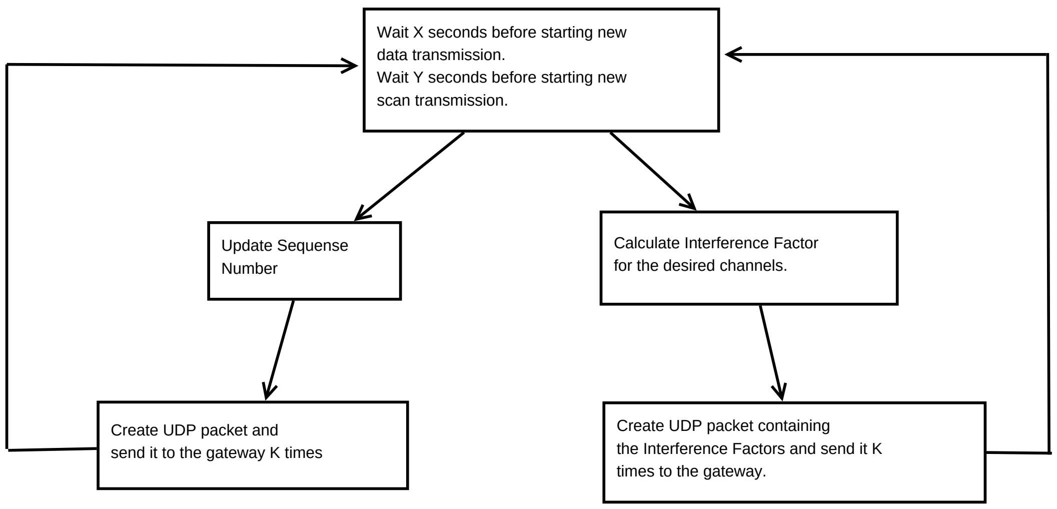

Wait X seconds before starting new data transmission. Wait Y seconds before starting new scan transmission.
Update Sequense Number
Calculate Interference Factor for the desired channels.
Create UDP packet and send it to the gateway K times
Create UDP packet containing the Interference Factors and send it K times to the gateway.

Figure 4.14: Packet Error Rate Algorithm on the non Gateway Nodes.

Chapter 5

Frequency Allocation

In this section the algorithms used in the frequency allocation of the network nodes, are described.

5.1 Connectivity Graph Analysis

Connectivity between any two radio interfaces exists when the received power is above the sensitivity of the radio interface hardware. To refer to a node's radio interfaces, e.g. radio interface 2 of node 1, the following notation is used: $N_1 : r_2$. To determine if a connectivity link is a communication link or an interference link, the routing tree has to be decided first. If a connectivity link exists in the routing tree, then it will be a communication link; otherwise, it will be an interference link.

The antennas used in this work have the same theoretical maximum gain of 17dBi in each side of the link. At distance d_i between transmitter and receiver i, the signal power has experienced "one-way" propagation link loss L_i. It is assumed that the transmitter antenna is mounted at height h_T and the receiver antenna at height h_R. Taking into account the line-of-sight (LOS) path between transmitter and receiver, as well as one reflection from the ground, one-way loss, L_i can be approximated by the two-ray model loss [12]:

$$L_i = \frac{\text{received power}}{\text{transmitted power}} = \begin{cases} G_T G_R \left(\frac{\lambda}{4\pi d_i}\right)^2 & \text{, if } d_i < d_0 \\ G_T G_R \left(\frac{h_T h_R}{d_i^2}\right)^2 & \text{, if } d_i \geq d_0 \end{cases} \tag{5.1}$$

where G_T and G_R are the gains of the transmitter and receiver antenna respectively, λ is the RF carrier wavelength and d_0 is given by:

$$d_0 = \frac{4\pi h_T h_R}{\lambda}. \tag{5.2}$$

Due to the placement of antennas at high altitudes, the first case of the above formula, i.e. free space loss is used for most links, where $d_i < d_0$.

It is noted that from the sensitivity formula (at 27^o C),

$$P_{\text{RX}}^{\text{min}} = -174\frac{\text{dBm}}{\text{Hz}} + \text{NF} + 10\log\text{BW} + \theta, \tag{5.3}$$

and setting noise figure NF = 5dB, bandwidth BW = 11MHz and $P_{\text{RX}}^{\text{min}} =$ -90dBm (@1Mbps), the minimum signal-to-interference-and-noise ratio (SINR) threshold can be calculated on the order of $\theta = 9$dB, for which reliable link communication exists, i.e. when SINR $> \theta$. Threshold θ will be needed at performance evaluation, subsequently.

In Table 5.1, the received power for some of the network links is presented.

Table 5.1: Received Power for Network Links.

Link	**Distance (m)**	$\mathbf{P_{RX}(W)}$	$\mathbf{P_{RX}(dBm)}$
$N_1 : r_1 - N_2 : r_1$	474	2136×10^{-12}	-56.7
$N_4 : r_1 - N_2 : r_2$	4410	24.68×10^{-12}	-76.08
$N_5 : r_1 - N_2 : r_2$	5100	18.4×10^{-12}	-77.35
$N_6 : r_1 - N_1 : r_2$	4760	21.2×10^{-12}	-76.73
$N_5 : r_1 - N_1 : r_2$	4895	20×10^{-12}	-76.98
$N_7 : r_2 - N_3 : r_2$	5230	17.56×10^{-12}	-77.55
$N_9 : r_1 - N_7 : r_3$	998	482×10^{-12}	-63.17
$N_8 : r_1 - N_7 : r_3$	667	1080×10^{-12}	-59.66
$N_4 : r_1 - N_1 : r_2$	4265	26.4×10^{-12}	-75.78

5.2 Conflict Graph (CG) Creation Algorithm

For interference modeling, the concept of conflict graphs is exploited. For the conflict graph construction, the connectivity graph $G(V, E)$ is needed. Each $v \in V$ represents a radio interface and each $e \in E$ represents a communication or interfering link between two radio interfaces. An example of a conflict graph is shown in Fig. 5.1. A routing graph $G_r(V, E_r)$ is also vital in the conflict graph construction. Each $e_r \in E_r$ denotes a link alongside a routing path towards the gateway and vice-versa. The conflict graph vertices correspond to the edges of the routing graph and an edge between two conflict graph vertices is created when the links represented by said vertices, interfere with each other when operating simultaneously on the same frequency channel. The final part needed for the graph creation is a radio interfaces table $R[i, j]$, where $i, j \in V$ and $R[i, j] = 1$ if i, j belong to the same node.

Each conflict graph vertex represents a routing link, $l \in E_r$, between two radio interfaces, $i, j \in V$. For example, in Fig. 5.1, the $N_2 : r_1 - N_1 : r_2$ conflict graph vertex exists. This means that in the original network $N_2 : r_1$ and $N_1 : r_2$ correspond to two different radio interfaces and also the link between these two radio interfaces is a routing link. The notation $i \in V_c$, where i is a radio interface, means that the conflict graph vertex that contains the radio interface i is marked. There exists connectivity between any two radio interfaces when the received power is above the sensitivity of the radio interface.

The algorithm is summarized in Algorithm 1. It starts by creating a node $g \in V_c$ for each one of the routing links given by E_r (Lines 1,2). It connects those $v \in V_c$ that contain a radio interface from the same node (Line 3). This is to avoid the self-interference problem when assigning channels.

Algorithm 1: CONFLICT GRAPH CREATION

```
Input: Connectivity Graph $G(V,E)$, Routing Tree $G_r(V,E_r)$, Radio Interfaces Table $R[i,j]$
Output: Conflict Graph $G_c(V_c,E_c)$
 1: for each edge $l \in E_r$ do
 2:     create a node $g \in V_c$
 3: In $G_c$ connect the nodes where: $i,j \in V_c, i \neq j$ and $R[i,j] = 1$
 4: for each node $v \in V_c$ do
 5:     mark the two radio interfaces $m, n$ that $v$ contains
 6:     $addToQueue(Q_1, m)$
 7:     $addToQueue(Q_1, n)$
 8:     while $size(Q_1) > 0$ do
 9:         $u = removeHead(Q_1)$
10:         for each $w \in V$ do
11:             if $(u,w) \in E$ then
12:                 $addToQueue(Q_2, w)$
13:             if $(u,w) \in E_r$ then
14:                 find $p \in V_c$ that contains the $u,w$ radio interfaces
15:                 connect $v$ with $p$ in Conflict Graph
16:     while $size(Q_2) > 0$ do
17:         $u = removeHead(Q_2)$
18:         for each $w \in V$ do
19:             if $(u,w) \in E_r$ then
20:                 find $p \in V_c$ that contains the $u,w$ radio interfaces
21:                 connect $v$ with $p$ in Conflict Graph
```

The algorithm then visits each one of the newly created vertices and tries to find all other vertices that it interferes with, when operating on the same frequency channel. In lines 5-7 it finds the two radio interfaces that the visited node $v \in V_c$ contains and adds them to a queue Q_1. Subsequently it visits every radio interface $i \in V$ that has been added in Q_1 and it finds the communication and interference links for the given radio interface. In lines 10-15 if a particular connectivity link between radio interfaces $i, j \in V$ happens to also be a routing link, it means that if it operates simultaneously on the same channel with the link contained in the visited vertex v, there will be interference. So the algorithm finds the vertex $u \in V_c$ that uses the i, j radio interfaces and connects it with v, $(v, u) \in E_c$. The algorithm also adds the radio interface j in a second queue Q_2, for later use. On the other hand if $(i, j) \in E$ but $(i, j) \notin E_r$, then it only adds the radio interface j in Q_2 (Lines 8-15).

When there are no more radio interfaces in $Q1$, the algorithm visits the radio interfaces that have been added in Q_2 (Lines 16, 17). As before, for every radio interface $i \in V$ that the algorithm visits, it finds its communication and interference links. If any one of those links is also a routing link (i.e. $(i, j) \in E_r$), then $(v, u) \in E_c$, where $u \in V_c$ the vertex that contains the i, j radio interfaces (Lines 18-21).

The resulting conflict graph of the connectivity graph and routing tree of Fig. 1.1, is shown in Fig. 5.1.

5.3 CG Coloring Based on Node Degree

In this section, a centralized heuristic algorithm is proposed that solves the frequency allocation problem based on vertex coloring on the created conflict graph.

The input for the channel assignment algorithm consists of the newly created conflict graph $G_c(V_c, E_c)$, a set of frequency channels K and the Euclidean distance D, between any two radio interfaces i, j, where $(i, j) \in E$. The Euclidean distance D, is used to separate the weaker links ie. links with large distance between the radio interfaces, from the stronger links. This information is then used by the algorithm in order to protect the weaker links. The set K contains only three channels denoted (for simplicity) as channel 1, 2 and 3, corresponding to 802.11b frequency channels 1, 6 and 11, respectively. This occurs because in the 2.4 GHz band, there are only three non-overlapping channels (channel 1, 6 and 11) that can be used by 802.11b radios simultaneously without causing interference to each other.

In order to protect the weaker links (i.e. those with large distance), the *maxDist* vector is utilized; this vector is initialized as a null vector and it is used when there are no interference free channel assignments for a particular conflict graph vertex. The *maxDist* vector will lead to an assignment that causes interference with conflict graph vertices that have smaller distance between the radio interfaces, thus protecting the weak links.

The channel assignment algorithm also makes use of an initial channel array, F. This array is used when some radio links should operate on a fixed frequency channel. For example if the interference level on a given link is much lower on channel 1 than the other available channels, then with the use of F, the algorithm will assign the channel 1 on this particular link and at the same time will try to find a channel assignment for the other links that minimizes the interference. In other words, F is used to fix certain frequencies to certain links exploiting the frequency probing algorithm presented in 6.4.

The algorithm is summarized in Algorithm 2. The algorithm starts by visiting the vertex $v \in V_c$ with the largest node degree that has no channel assigned (Lines 1-3). Based on the conflict graph, the algorithm then finds the interfering vertices $u_i \in V_c$ with v. The channels used by u_i are marked

Algorithm 2: CHANNEL ASSIGNMENT BASED ON NODE DEGREE

```
Input: Conflict Graph $G_c(V_c, E_c)$, Set of Channels $K$, Distance between
    radio interfaces $D(i,j)$, Channel Array $F$
Output: Channel Assignment $V_c \Rightarrow K$
 1: while $\exists v \in V_c$ with no channel assigned do
 2:     select $u \in V_c$ with largest node degree that has no channel assigned
 3:     $nodeDegree(u) = -1$
 4:     initialise $maxDist(1, numOfChannels) = \vec{0}$
 5:     set of available channels for $u \triangleq C(u)$, $C(u) = K$
 6:     for each $w \in V_c, w \neq u$ do
 7:         if $(w,u) \in E_c$ then
 8:             $C(u) = C(u) - channel(w)$
 9:             find $D(i,j)$ between radio interfaces $i$ and $j$ of $w$
10:             if $D(i,j) > maxDist(channel(w))$ then
11:                 $maxDist(channel(w)) = D(i,j)$
12:     if $C(u) \neq \emptyset$ then
13:         randomly assign channel $c \in C(u)$ to $u$
14:     else
15:         assign to $u$ the channel $c \in K$ that minimizes $maxDist$
16:     for each $w \in V_c, w \neq u, w$ and $u$ share a common radio interface do
17:         $channel(w) = channel(u)$
```

Algorithm 3: Channel Assignment based on Radio Link Distance

```
Input: Conflict Graph $G_c(V_c, E_c)$, Set of Channels $K$, Distance between
    radio interfaces $D(i, j)$, Channel Array $F$
Output: Channel Assignment $V_c \Rightarrow K$
 1: while $\exists v \in V_c$ with no channel assigned do
 2:     select $u \in V_c$ with largest distance between its two radio interfaces $i$
    and $j$, that has no channel assigned
 3:     $D(i, j) = -1$
 4:     initialise $maxDist(1, numOfChannels) = \bar{\mathbf{0}}$
 5:     set of available channels for $u \triangleq C(u)$, $C(u) = K$
 6:     for each $w \in V_c, w \neq u$ do
 7:         if $(w, u) \in E_c$ then
 8:             $C(u) = C(u) - channel(w)$
 9:             find $D(i, j)$ between radio interfaces $i$ and $j$ of $w$
10:             if $D(i, j) > maxDist(channel(w))$ then
11:                 $maxDist(channel(w)) = D(i, j)$
12:     if $C(u) \neq \emptyset$ then
13:         randomly assign channel $c \in C(u)$ to $u$
14:     else
15:         assign to $u$ the channel $c \in K$ that minimizes $maxDist$
16:     for each $w \in V_c, w \neq u, w$ and $u$ share a common radio interface do
17:         $channel(w) = channel(u)$
```

as unavailable for v (Lines 6-8). For those u_i that can interfere with v, the algorithm also marks the distance of the radio link so it can protect the link with the largest distance (Lines 9-11). Subsequently, the algorithm checks if there exist available channels for the vertex v. If there exist, it will randomly assign one of the available channels to v and if not, it will assign to v the channel that is used by the links with the smallest distance between the radio interfaces (Lines 12-15).

In the final stage, the algorithm finds all other $p \in V_c$ that share a common radio interface with v and assigns them the same channel (Lines 16-17). This is due to the fact that it must be ensured that the algorithm assigns only one channel to each radio interface.

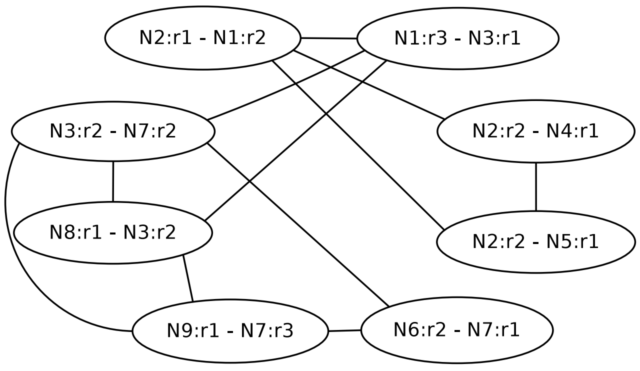
Due to the random selection of the available channels on each link, the final assignment might not be the best solution. For this reason the assignment algorithm is executed multiple times and the assignment with the lowest interference between the network nodes is kept; the lowest interference criterion used was the number of interfering links. There is active, ongoing research regarding measurement and estimation of network interference and relevant quality criteria of frequency allocations.

5.4 CG Coloring Based on Radio Link Distance

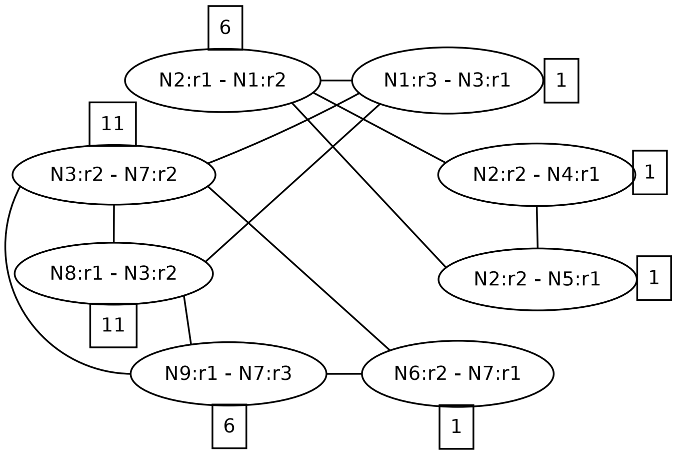
This is similar to the Algorithm 2 with one key difference; the algorithm visits the vertices of the conflict graph, based on the largest distance of a routing link (which is equivalent with the vertex $v \in V_c$ that contains the radio interfaces that form the link) (Lines 1-3), instead of the node degree. The next steps are exactly the same as in Algorithm 2.

Again this algorithm is executed multiple times and the assignment with the lowest interference between the network nodes is kept. The algorithm is summarized in Algorithm 3.

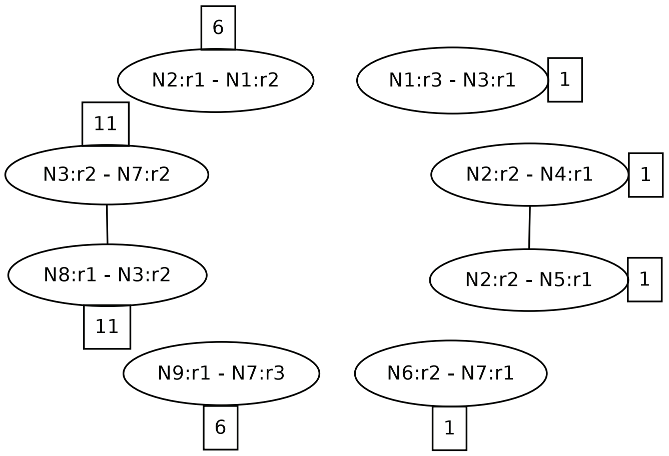
The resulting channel allocation based on Algorithm 2 for the 802.11 network of Fig. 1.1, is shown in Fig. 5.1. The channel allocation algorithm used for the frequency assignment of the network nodes is shown in Fig. 5.2.



(a) Conflict Graph.

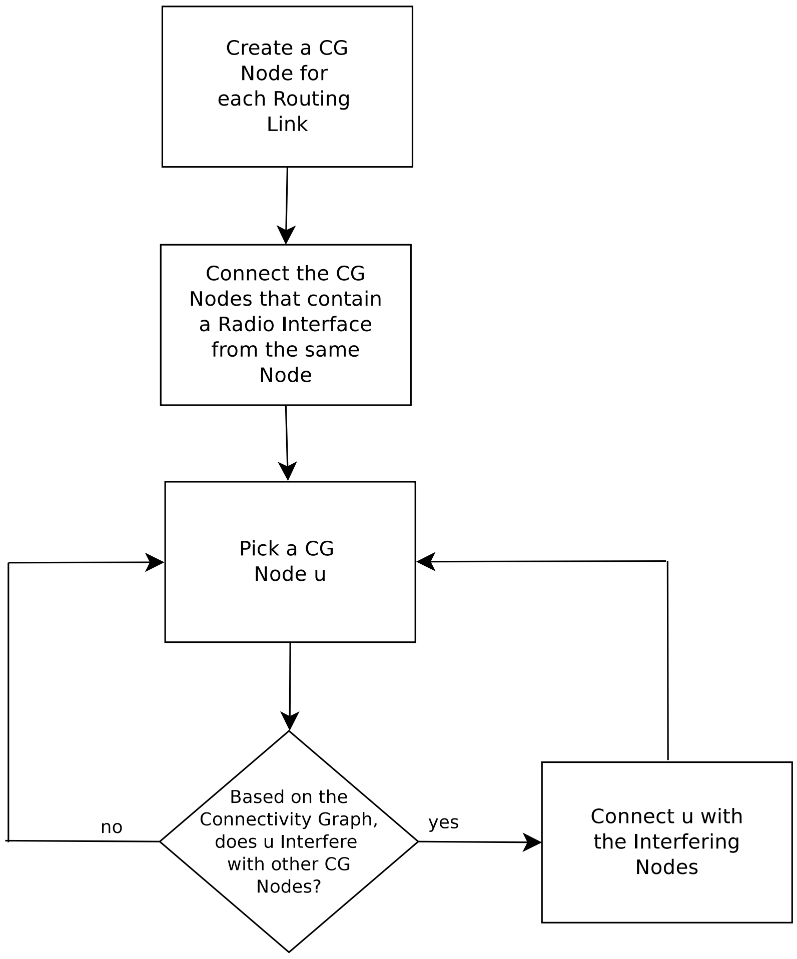


(b) Conflict Graph with a Channel Assignment.

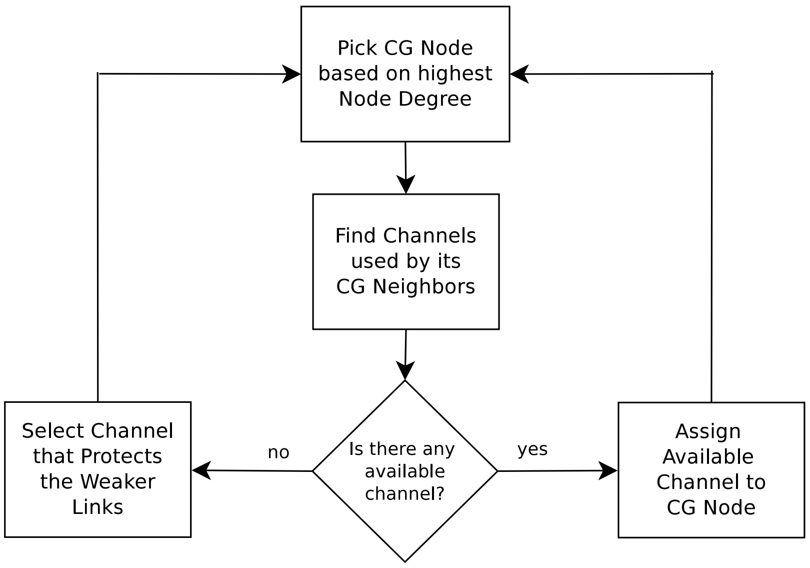


(c) Remaining Interference.

Figure 5.1: Conflict Graph before and after channel allocation and Remaining Interference.


Create a CG Node for each Routing Link
Connect the CG Nodes that contain a Radio Interface from the same Node
Pick a CG Node u
Based on the Connectivity Graph, does u Interfere with other CG Nodes?
no
yes
Connect u with the Interfering Nodes

(a) Conflict Graph Creation.


Pick CG Node based on highest Node Degree
Find Channels used by its CG Neighbors
Is there any available channel?
no
yes
Select Channel that Protects the Weaker Links
Assign Available Channel to CG Node

(b) Channel Allocation.

Figure 5.2: Channel Assignment Algorithm

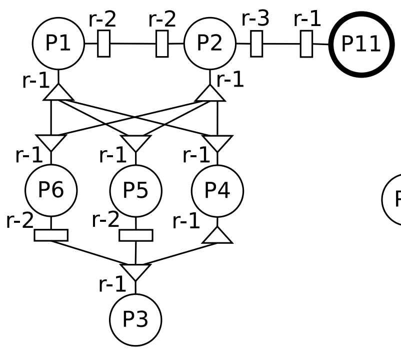
Chapter 6

Case Study Evaluation

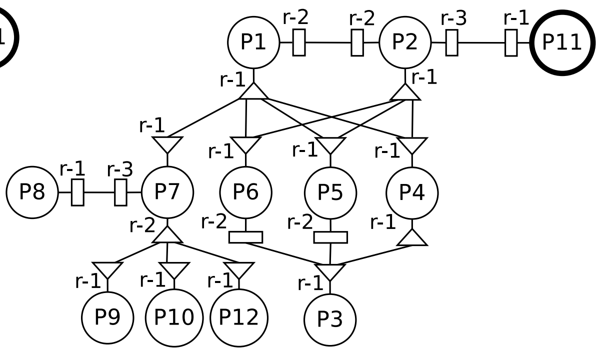
6.1 Simulation Results

Frequency channel allocation is evaluated in terms of remaining interference. The frequency allocation algorithms are firstly tested in two different network scenarios; the first one is shown in Fig. 6.1a and consists of 7 nodes and the second network is shown in Fig. 6.1b and consists of 12 nodes. The resulting frequency allocations are then compared with the assignments produced by another frequency assignment algorithm, the Tabu algorithm. These two networks are used for simulation purposes only and do not represent a real 802.11s mesh network. To refer to a simulation node's radio interfaces, e.g. radio interface 2 of node 1, the following notation is used: P_{1-2}. When the node has only one radio interface, the node's name is used instead, e.g. P_4.

One possible routing tree for the small and the large network, is shown in Fig. 6.2a and Fig. 6.2b, respectively. Based on those routing trees, the conflict graphs are constructed using the aforementioned Algorithm 1. The results are shown in Fig. 6.3a and Fig. 6.3b for the small and the large

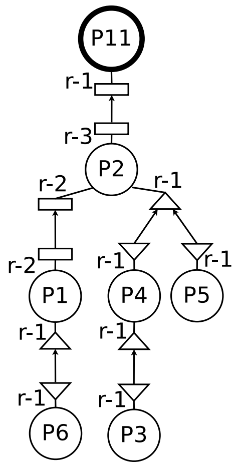


(a) Communication Link Graph of the Small Simulation Network.

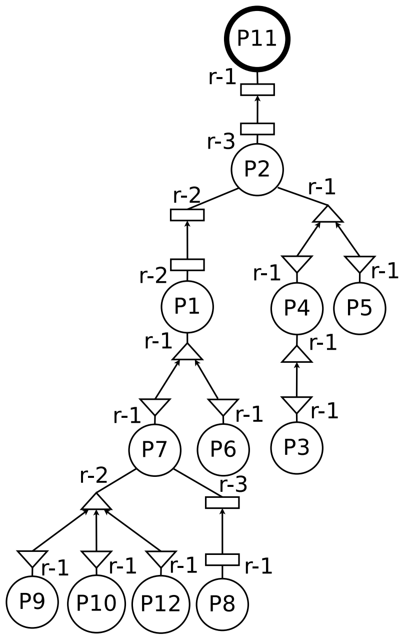


(b) Communication Link Graph of the Large Simulation Network.

Figure 6.1: Communication Link Graph of the 2 Simulation Networks.



(a) Routing of the Small Network.



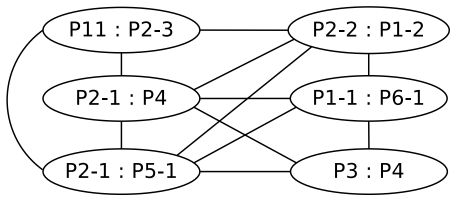
(b) Routing of the Large Network.

Figure 6.2: Routing of the 2 Simulation Networks.

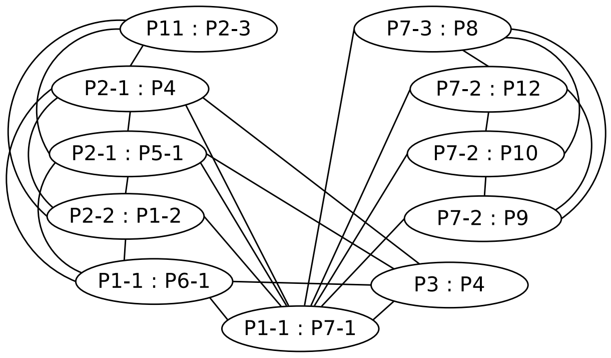
network, respectively.

The channel allocation for the small network of Algorithm 2, Algorithm 3 and the Tabu [3] is shown in Fig. 6.4, Fig. 6.5 and Fig. 6.6, respectively. Multi-radio node P_2 is assigned different channels in the first two algorithms, however, remaining 2-hop interference (imposing hidden-node terminal) is not addressed with the 3rd algorithm. The rest of remaining interference is due to the directional antennas, serving more than one link and could be alleviated only with time-based medium access control. We should note here that the Tabu based algorithm does not always result in assignments with more interference than those created by the proposed algorithms; due to its continuous randomized selection of channels, the Tabu algorithm may result with the same remaining interference as our algorithms, but not better, for the studied network cases. The resulting frequency allocation network for the above 3 algorithms in the small network, is shown in Fig. 6.7 (left), Fig. 6.7 (right) and Fig. 6.8.

Channel allocation for the large network of Algorithm 2, Algorithm 3 and Tabu is shown in Fig. 6.9, Fig. 6.10 and Fig. 6.11, respectively. All three algorithms offer the same interfering links, although different iterations



(a) Conflict Graph of the Small Network.



(b) Conflict Graph of the Large Network.

Figure 6.3: Conflict Graphs of the 2 Simulation Networks.

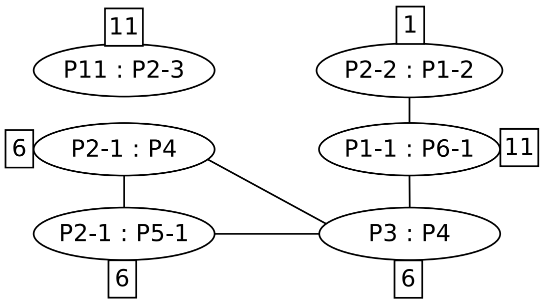


Figure 6.4: Remaining Interference after Channel Assignment of Algorithm 2 on the Small Network.

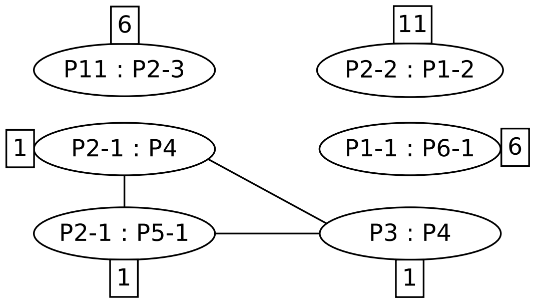


Figure 6.5: Remaining Interference after Channel Assignment of Algorithm 3 on the Small Network.

of the Tabu based algorithm may result with more interfering links. Both multi-radio nodes P_2 and P_7 are assigned different frequency channels among their radio interfaces, in all cases. As with the small network, the remain-

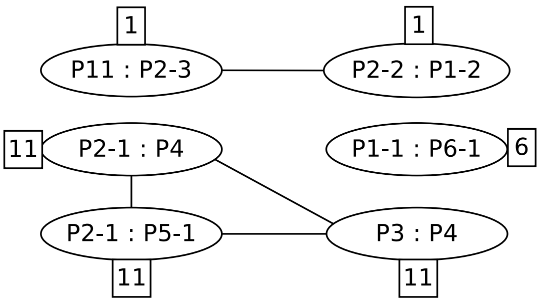


Figure 6.6: Remaining Interference after Channel Assignment of Tabu Algorithm on the Small Network.

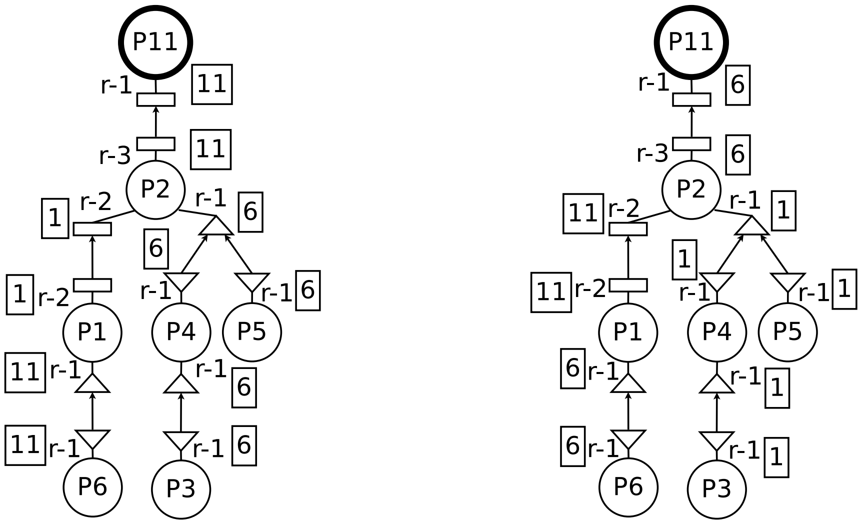


Figure 6.7: Routing with the Channel Assignment of Algorithm 2 (left) and Algorithm 3 (right) on the Small Network.

ing interference is due to the fact that the same antenna serves more than one links and the connected radio is constrained to use the same frequency channel. The resulting network planning with frequency allocation for the above 3 algorithms is shown in Fig. 6.12 (left), Fig. 6.12 (right) and Fig. 6.13. The frequency allocation results are very similar in all cases. That may seem surprising and is attributed to the multiple constraints imposed on the problem, including operation at a specific frequency channel per radio card (even when that card serves multiple links through a broad beamwidth antenna) and inability to place antennas in specific places. Nevertheless, broad beamwidth antennas increase network robustness through path diversity and also decrease installation cost (since path diversity is achieved with one radio).

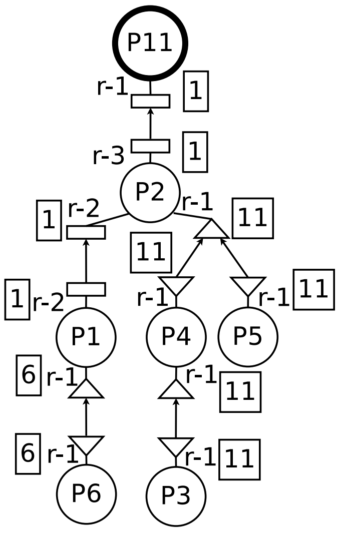


Figure 6.8: Routing with the Channel Assignment of Tabu Algorithm on the Small Network.

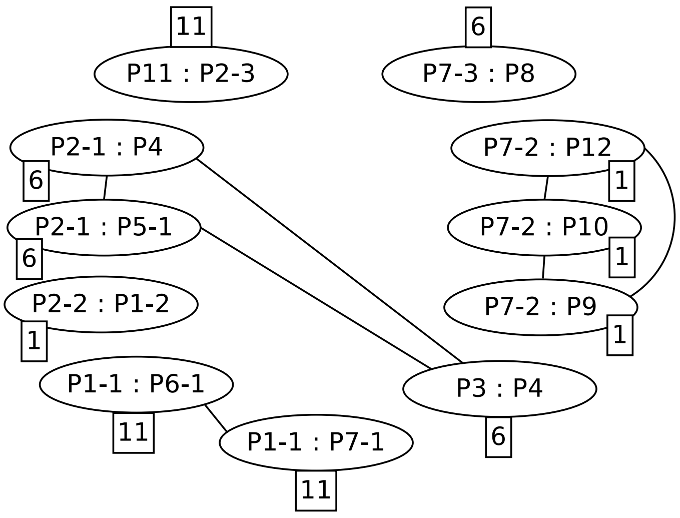


Figure 6.9: Remaining Interference after Channel Assignment of Algorithm 2 on the Large Network.

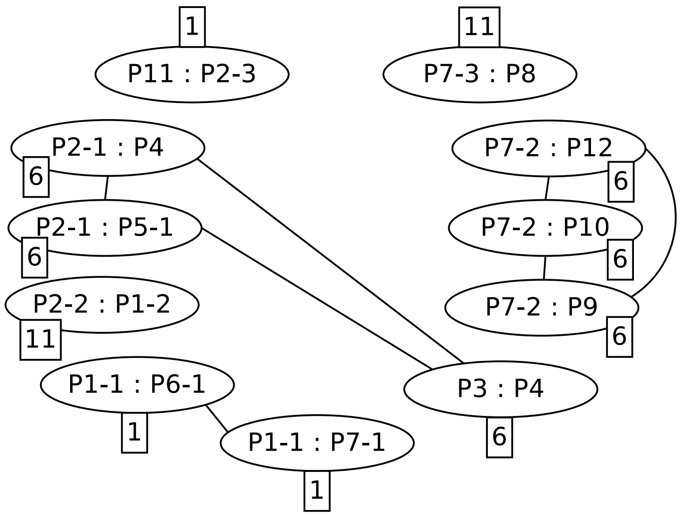


Figure 6.10: Remaining Interference after Channel Assignment of Algorithm 3 on the Large Network.

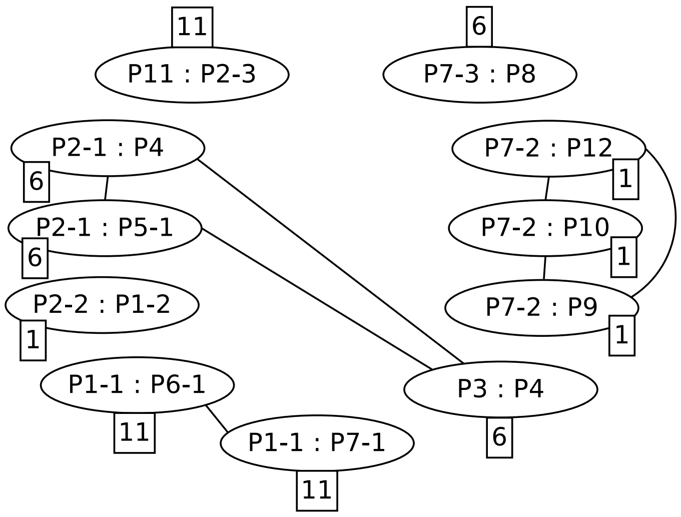


Figure 6.11: Remaining Interference after Channel Assignment of Tabu Algorithm on the Large Network.

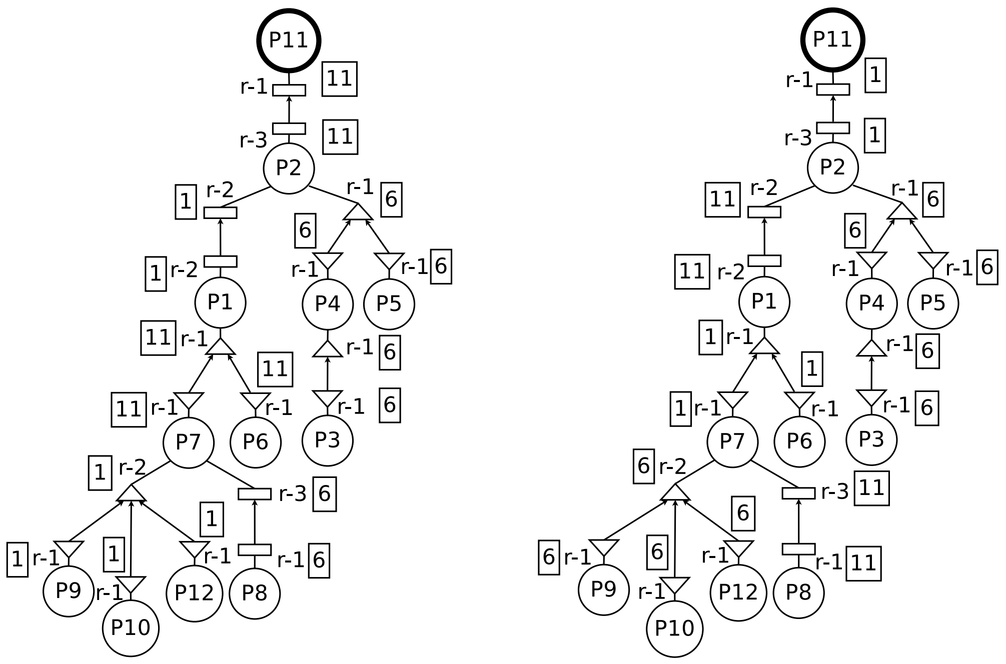


Figure 6.12: Routing with the Channel Assignment of Algorithm 2 (left) and Algorithm 3 (right) on the Large Network.

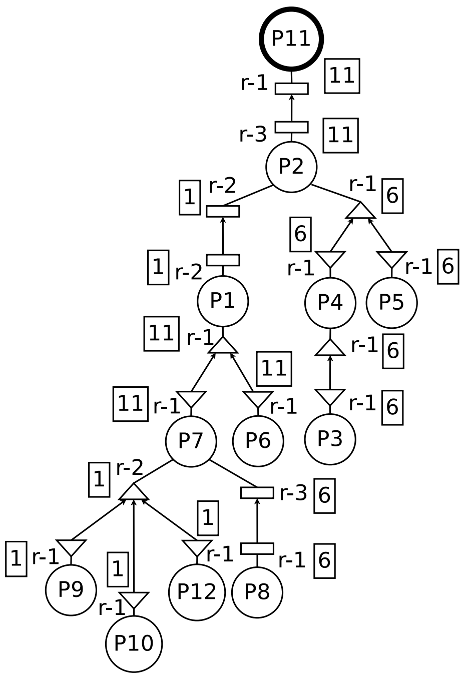


Figure 6.13: Routing with the Channel Assignment of Tabu Algorithm on the Large Network.

6.2 Deployment

Each wireless node consisted of an Alix 3d2 board, enclosed in a waterproof case (Fig. 6.14). Each board was equipped with one or more 802.11b low cost mini PCI or USB radios. The Alix 3d2 has had two mini PCI and two USB slots so each network node could employ up to four radio interfaces simultaneously.

The challenging part during the deployment phase of each node was to find reliable links with neighboring nodes. Due to the large distance of the network links, the antenna alignment was not a trivial task. It was accomplished with the use of a spectrum analyzer and a frequency generator (Fig. 6.15). The deployment of low cost high gain antennas (Fig. 6.16) was necessary to further increase link budget and thus, reception reliability.

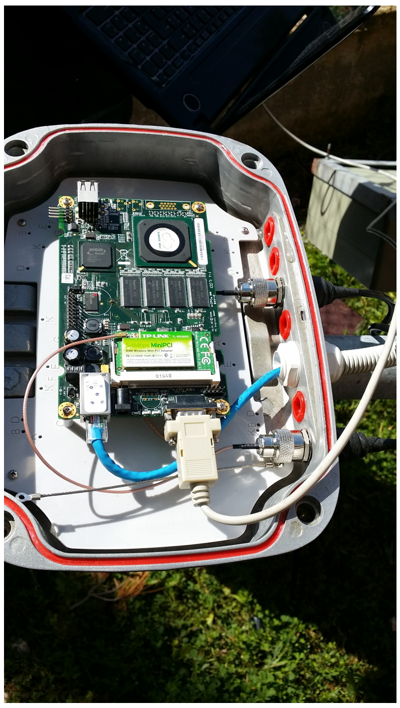


Figure 6.14: Alix Board in the Waterproof Case

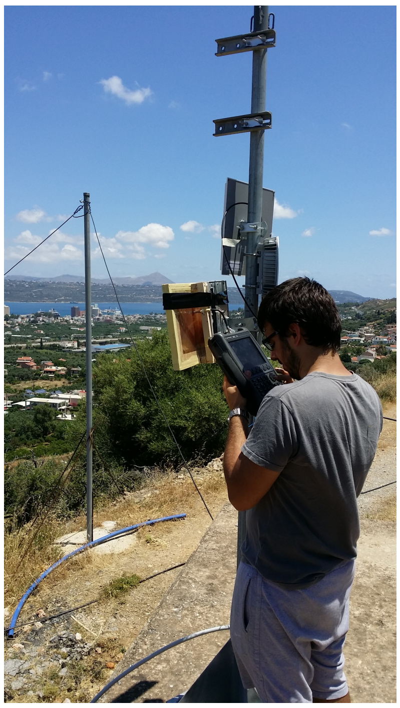


Figure 6.15: Alignment Procedure

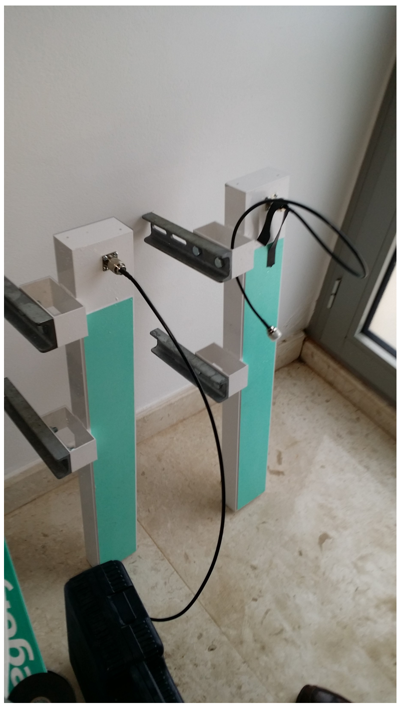


Figure 6.16: Antennas

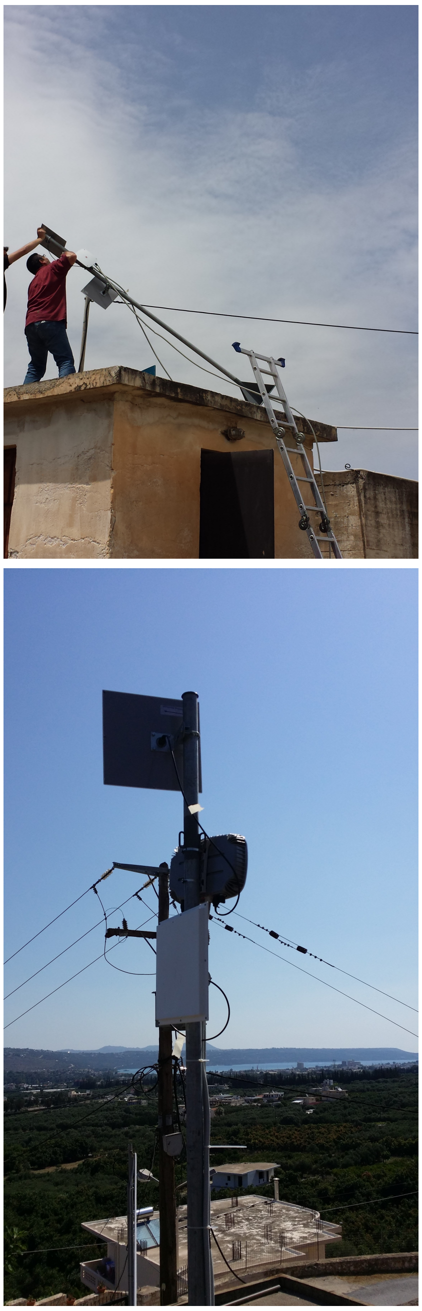


Figure 6.17: Deployment of a Wireless Node

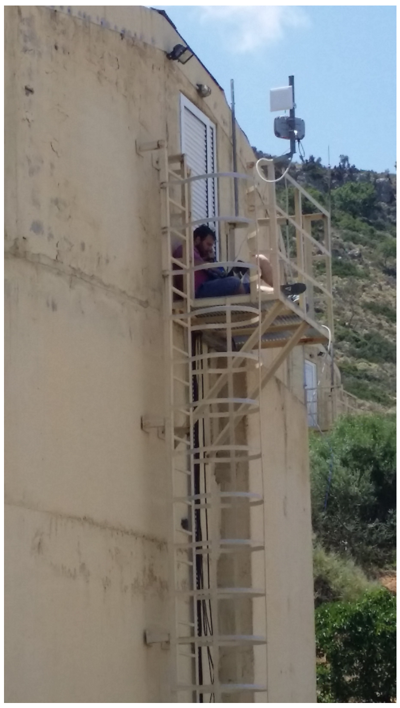


Figure 6.18: Installation of N_8

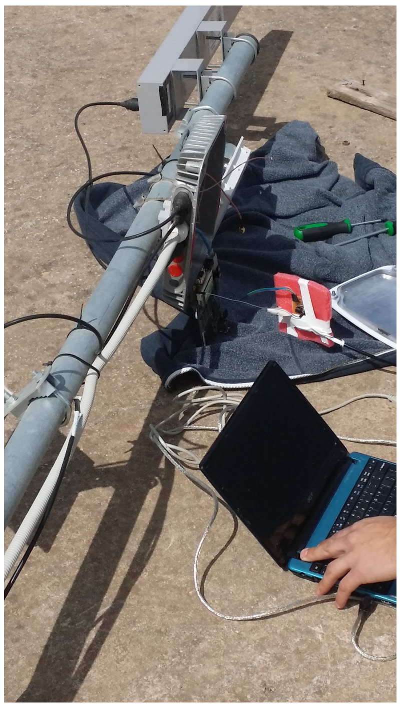


Figure 6.19: Installation of N_2

6.3 Experimental Results

In this section, experimental results of the channel assignment based on node degree algorithm used on the real mesh network of Fig. 1.1, are presented. It is important to note that this algorithm assigns frequency channels based only on internal network interference. The overall interference (internal plus external) can be measured with the probing algorithm presented in the next section. The channel assignment based on radio link distance algorithm, produces similar results.

To test the efficiency of the channel assignment algorithm, two experiments were conducted. Both experiments used the routing tree of Fig. 1.1. There were employed various antennas for the creation of the network: two different broad beamwidth type antennas and two different narrow beamwidth type antennas. Each one of them has different gain in the 3 available 802.11 frequency channels. To obtain credible results from a real 802.11 network, it is critical to ensure for each routing link, that the total transmission power (radio interface transmission power plus antenna gain) is the same among all the available frequency channels. The various antenna gains are shown in Table 6.1.

In Case 1, all the radio interfaces operated on the same frequency channel. Each node was set to transmit every two minutes twenty UDP packets towards its gateway for over 24 hours. The gateway node was able to compute the packet error rate of the current transmission based on the number of the received packets. There were two nodes that acted as gateways, N_1 and N_3. The packet error rate results and the antenna type are shown at Table 6.2a. The displayed error rate is calculated based on the whole path to the gateway traveled by the probe packets. Table 6.2b shows the antenna types of the gateway nodes.

In Case 2, the channel assignment algorithm was used to assign frequency channels on the radio interfaces that formed the routing tree. Similarly each node sent the same amount of data to its gateway and for the same duration. The packet error rate results (calculated again on the whole path from the sender to the gateway) as well as the antenna type and the new channel assignment are shown at Table 6.2c. Table 6.2d shows the new channel assignment for the two gateway nodes.

With the exception of one node, the results clearly show that the channel assignment imposed by the implemented algorithm reduces the packet error rate among the network links, offering an increased network stability. It is highly likely that the increased packet error rate of N_8 in Case 2, was caused by interference from other co-located 802.11 devices. There were no packet error rate results for N_7 because this node was used only for relaying data

	2412MHz (1)	**2437MHz (6)**	**2462MHz (11)**
sector	12.9634 dBi	9.7330 dBi	5.9416 dBi
panel (big)	11.6863 dBi	13.0259 dBi	14.0251 dBi
panel (small)	9.5863 dBi	10.5359 dBi	10.6751 dBi
sector (C)	10.2863 dBi	11.5559 dBi	11.7451 dBi

Table 6.1: Antenna Type Gains Measured in Channel X

packets between N_9 and N_3.

In a second set of experiments conducted a few weeks later than the first one, the channel assignment based on node degree algorithm produced different results, as shown in Table 6.3. As with the previous experiment, in Case 1 all the radio interfaces operated on the same channel and in Case 2, Algorithm 2 was used to assign channels to the radio interfaces that formed the routing tree. The results show again that the packet error rate among network links that are allocated frequency channels based on Algorithm 2, is greatly reduced compared to the common channel case.

Both experiments also showed that the packet error rate and the network reliability is greatly depended on weather conditions and on interference from neighboring 802.11 terminals.

Table 6.2: Experimental Results(1)

Node	**PER**	**Interface**	**Antenna Type**	**Channel**
N_2	0.0004	$N_2 : r_1$	sector(C)	1
		$N_2 : r_2$	sector	1
N_4	0.538	$N_4 : r_1$	panel(big)	1
N_5	0.54	$N_5 : r_1$	panel(big)	1
N_8	0.07	$N_8 : r_1$	panel(small)	1
N_9	0.5	$N_9 : r_1$	panel(small)	1
N_7	-	$N_7 : r_2$	panel(big)	1
		$N_7 : r_3$	sector(C)	1

(a) Channel Assignment Algorithm: PERs for Case 1

Gateway	**Interface**	**Antenna Type**	**Channel**
N_1	$N_1 : r_2$	sector	1
	$N_1 : r_3$	panel(big)	1
N_3	$N_3 : r_1$	panel(big)	1
	$N_3 : r_2$	panel(big)	1

(b) Channel Assignment Algorithm: Gateway Nodes for Case 1

Node	**PER**	**Interface**	**Antenna Type**	**Channel**
N_2	0.01	$N_2 : r_1$	sector(C)	6
		$N_2 : r_2$	sector	1
N_4	0.41	$N_4 : r_1$	panel(big)	1
N_5	0.32	$N_5 : r_1$	panel(big)	1
N_8	0.03	$N_8 : r_1$	panel(small)	11
N_9	0.55	$N_9 : r_1$	panel(small)	6
N_7	-	$N_7 : r_2$	panel(big)	11
		$N_7 : r_3$	sector(C)	6

(c) Channel Assignment Algorithm: PERs for Case 2

Gateway	**Interface**	**Antenna Type**	**Channel**
N_1	$N_1 : r_2$	sector	6
	$N_1 : r_3$	panel(big)	1
N_3	$N_3 : r_1$	panel(big)	1
	$N_3 : r_2$	panel(big)	11

(d) Channel Assignment Algorithm: Gateway Nodes for Case 2

Table 6.3: Experimental Results(2)

Node	**PER**	**Interface**	**Antenna Type**	**Channel**
N_2	0.01	$N_2 : r_1$	sector(C)	1
		$N_2 : r_2$	sector	1
N_4	0.85	$N_4 : r_1$	panel(big)	1
N_5	0.77	$N_5 : r_1$	panel(big)	1
N_8	0.061	$N_8 : r_1$	panel(small)	1
N_9	0.21	$N_9 : r_1$	panel(small)	1
N_7	-	$N_7 : r_2$	panel(big)	1
		$N_7 : r_3$	sector(C)	1

(a) Channel Assignment Algorithm: PERs for Case 1

Gateway	**Interface**	**Antenna Type**	**Channel**
N_1	$N_1 : r_2$	sector	1
	$N_1 : r_3$	panel(big)	1
N_3	$N_3 : r_1$	panel(big)	1
	$N_3 : r_2$	panel(big)	1

(b) Channel Assignment Algorithm: Gateway Nodes for Case 1

Node	**PER**	**Interface**	**Antenna Type**	**Channel**
N_2	0.006	$N_2 : r_1$	sector(C)	11
		$N_2 : r_2$	sector	1
N_4	0.75	$N_4 : r_1$	panel(big)	1
N_5	0.67	$N_5 : r_1$	panel(big)	1
N_8	0.009	$N_8 : r_1$	panel(small)	6
N_9	0.14	$N_9 : r_1$	panel(small)	11
N_7	-	$N_7 : r_2$	panel(big)	6
		$N_7 : r_3$	sector(C)	11

(c) Channel Assignment Algorithm: PERs for Case 2

Gateway	**Interface**	**Antenna Type**	**Channel**
N_1	$N_1 : r_2$	sector	11
	$N_1 : r_3$	panel(big)	1
N_3	$N_3 : r_1$	panel(big)	1
	$N_3 : r_2$	panel(big)	6

(d) Channel Assignment Algorithm: Gateway Nodes for Case 2

Child Interface	Father Interface	Channel	PER
$N_9 : r_1$	$N_7 : r_3$	1	0
		6	0
		11	0.015
$N_8 : r_1$	$N_3 : r_2$	1	0
		6	0.025
		11	0.03
$N_2 : r_1$	$N_1 : r_2$	1	0.02
		6	0.04
		11	0.12

Table 6.4: Probing Algorithm: PERs

6.4 Frequency Channel Probing

In this section a channel probing algorithm [1] for statistics collection is presented. This algorithm is used for the measurement of the overall network interference for each link. This information can be used along with the previously presented channel assignment algorithm, to provide a centralized way of frequency allocation and to test various frequency allocation algorithms.

As input, the channel probing algorithm needs the Routing Tree $\mathcal{T}$, the Network MAC and IP addresses table $\mathcal{A}$, the Antenna Type Set $\mathcal{B}$, the Primary Channel $\mathcal{P}$ and the Set of Secondary Channels $\mathcal{S}$. The Routing Tree is used by the algorithm to determine the father of each child interface. The algorithm needs the Network MAC and IP addresses table to translate the MACs into IPs for the UDP transmissions that take place. Each antenna type that the network nodes use has different gains, depending on the transmission channel, which are known. The algorithm uses the Antenna Type Set to adjust the transmission power. The Primary Channel is considered to be the main channel of the link and the Secondary Channels are the channels under testing.

The block diagram for the probing algorithm is shown in Fig. 6.20 for the child interface and in Fig. 6.21 for the father interface. The complete algorithm is summarized in Algorithm 4. Initially it places the father interface on listening mode on primary channel. The child interface randomly selects the secondary channel and includes that information in the probe packets that it sends on primary channel to its father interface. The child interface then changes to the secondary channel. When the father interface receives

[1]In collaboration with MSc. Grad Student Panagiotis Oikonomakos

the probe packets on the primary channel, it calculates the packet error rate (PER) and then switches to the secondary channel based on the information that the probe packets contain.
The child interface starts the transmission of the probe packets on secondary channel. When the father interface receives the probe packets on the secondary channel, it calculates the PER and then both the child and the father interface switch back to the primary channel. The process is then repeated. In case that the father interface does not receive any packets on primary channel, the link is considered not established and no communication is available between those two interfaces.
In case that the father interface does not receive any packets on secondary

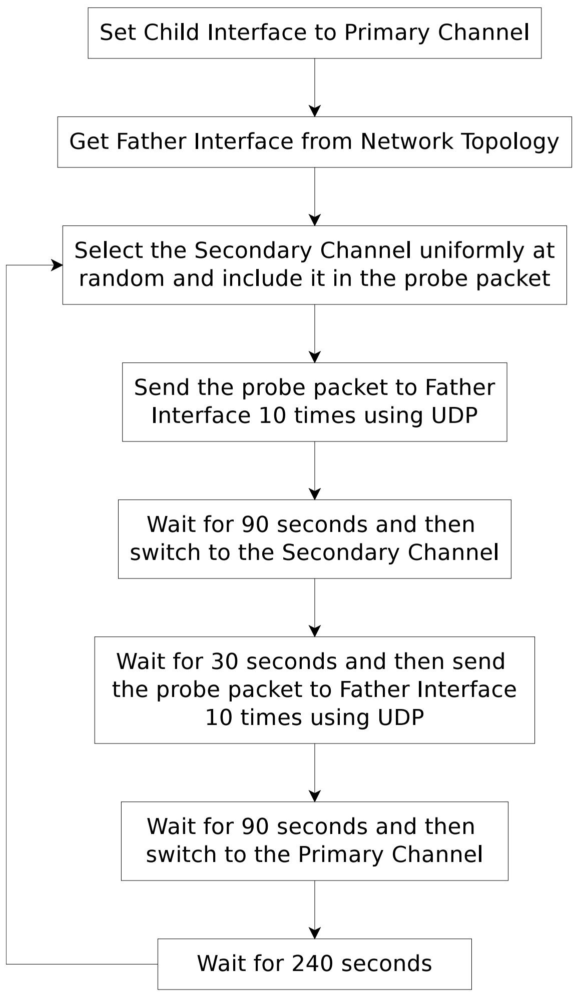

Set Child Interface to Primary Channel
Get Father Interface from Network Topology
Select the Secondary Channel uniformly at random and include it in the probe packet
Send the probe packet to Father Interface 10 times using UDP
Wait for 90 seconds and then switch to the Secondary Channel
Wait for 30 seconds and then send the probe packet to Father Interface 10 times using UDP
Wait for 90 seconds and then switch to the Primary Channel
Wait for 240 seconds

Figure 6.20: Probing Algorithm Child Interface.

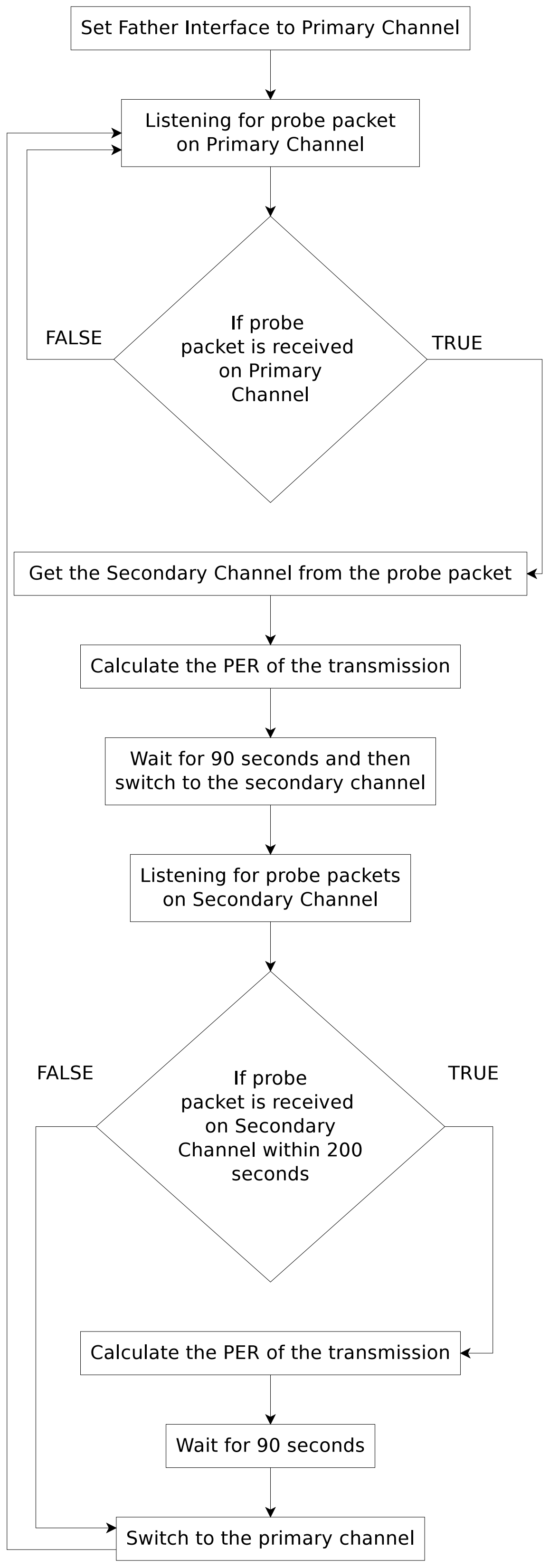

Set Father Interface to Primary Channel
Listening for probe packet on Primary Channel
If probe packet is received on Primary Channel
FALSE
TRUE
Get the Secondary Channel from the probe packet
Calculate the PER of the transmission
Wait for 90 seconds and then switch to the secondary channel
Listening for probe packets on Secondary Channel
If probe packet is received on Secondary Channel within 200 seconds
FALSE
TRUE
Calculate the PER of the transmission
Wait for 90 seconds
Switch to the primary channel

Figure 6.21: Probing Algorithm Father Interface.

channel, there is a backup timer that will force the interface to swich back to the primary channel before the child interface starts the next transmission. The recorded PER of each transmission is stored into a file in the father node. The format of these records is presented below:

```
2015-6-12 11:21:42
From IP: 10.10.10.2 | To IP: 10.10.10.4 | PER: 0 | Reported Channel: 6 | seq num: 159
2015-6-12 11:23:54
From IP: 10.10.10.2 | To IP: 10.10.10.4 | PER: 0 | Reported Channel: 1 | seq num: 159
2015-6-12 11:28:9
From IP: 10.10.10.2 | To IP: 10.10.10.4 | PER: 0 | Reported Channel: 11 | seq num: 160
2015-6-12 11:30:21
From IP: 10.10.10.2 | To IP: 10.10.10.4 | PER: 0 | Reported Channel: 1 | seq num: 160
```

The IP addresses determine the interfaces that participate in the transmission. *From IP* is the child interface and *To IP* is the father interface. *Reported Channel* is the channel that the transmission took place and the *seq num* is the sequence number of the transmission. The same sequence number is used for the transmission on the primary and on the secondary channel. Each record has a timestamp which is the time that the father interface received the first probe packet of each transmission.

The probing algorithm was tested on three links of the network for approximately 3 hours on each link. The results are shown in Table 6.4. Based on those results the $N_9 : r_1 - N_7 : r_3$ link should be set on channel 1 or 6, the $N_8 : r_1 - N_3 : r_2$ link on channel 1 and the $N_2 : r_1 - N_1 : r_2$ link on channel 1. The results of the probing algorithm can be used by the channel assignment algorithm, providing an allocation based on both the internal and the external interference.

Algorithm 4: Channel Probing Algorithm

Input: Network Topology $\mathcal{T}$, Network MAC and IP Table $\mathcal{A}$, Antenna Type Set $\mathcal{B}$, Primary Channel $\mathcal{P}$, Set of Secondary Channels $\mathcal{S}$.
Output: Link PER on Primary and Secondary Channels.

Child Interface:

```
1: Switch the Interface to Primary Channel $\mathcal{P}$.
2: Get Father Interface $t$ from Network Topology $\mathcal{T}$.
3: Get MAC and IP of $t$ from Network MAC and IP Table $\mathcal{A}$.
4: Ping $t$ (5 ICMP packets).
5: Select Uniformly at random a Channel $s$ from Secondary Channels set $\mathcal{S}$.
6: Create Probe Packet $d1$ to send to $t$ including $s$.
7: Start a Timer that will switch the Interface to Channel $s$ after 90 seconds.
8: Start a Timer that will start the next Transmission in 120 seconds.
9: Send $d1$ to $t$ 10 times using UDP.
10: Switch the Interface to Channel $s$ from step 7.
11: Create a Probe Packet $d2$ to send to $t$ including $\mathcal{P}$.
12: Send $d2$ to $t$ 10 times using UDP from step 8.
13: Start a Timer that will switch the Interface to Channel $\mathcal{P}$ after 90 seconds.
14: Wait for 240 seconds and go to step 2.
```

Father Interface:

```
1: Switch the Interface to Primary Channel $\mathcal{P}$.
2: Start Listening for Probe Packets $d1$ on Channel $\mathcal{P}$
3: **if** $d1$ is Received **then**
4:     Get the Secondary Channel $s$ from $d1$
5:     Start a Timer that will switch the Interface to Channel $s$ in 90 seconds.
6:     Calculate PER of the current transmission.
7:     Switch the Interface to Channel $s$ from step 5.
8:     Start a backup Timer that will switch the Interface to Channel $P$ in 200 seconds.
9:     Start Listening for Probe Packets $d2$ on Channel $s$.
10:    **if** $d2$ is Received **then**
11:        Start a Timer that will switch the Interface to Channel $P$ in 90 seconds.
12:        Calculate PER of the current transmission.
13:        Switch the Interface to Channel $P$ from step 11
14:    Switch the Interface to Channel $P$ from step 8.
15:    Go to step 2.
```

6.5 Bottleneck and Outage Analysis

In the large network of Fig. 6.1b there is always radio interface P_{7-2} receiving from 3 different radio terminals P_9, P_{10}, P_{12}, for all three frequency allocation algorithms. Therefore, one possible way to mitigate interference among the three links is through time-sharing (e.g. CSMA) of the 1 Mbps capacity; in that case, the end-2-end bottleneck bandwidth will be dominated by the above sharing, since all other radios in the network serve no more than two links.

Similar results hold for the smaller network of Fig. 6.1a, where time sharing among at most 2 links is required in order to mitigate remaining interference. Algorithm 2 and Algorithm 3 produce the same results regarding remaining interference after frequency channel assignment (Figs. 6.4, 6.5). In the Tabu-based frequency assignment (Fig. 6.6) there are two radio interfaces of a node in the same frequency channel (P_{2-2} and P_{2-3}), which may create self-interference between those radio interfaces.

For worst-case analysis, one could calculate the outage probability when interference cannot be mitigated. Taking into account average received power for each receiver as calculated in the connectivity graph section and assuming Rayleigh fading (which may not be appropriate for the considered setup but could offer a baseline metric), outage probability for J interfering radios is given by:

$$\Pr\left(\mathrm{SINR}_{\mathrm{RX}} \le \theta\right) \triangleq \Pr\left(\frac{g_{\mathrm{RX}} P_{\mathrm{RX}}}{N_0 + \sum_{j=1}^{J} g_j P_j} < \theta\right) \tag{6.1}$$

$$= 1 - e^{-\frac{\lambda_0 \theta N_0}{P_0}} \prod_{j=1}^{J} \frac{1}{1 + \frac{\lambda_0}{\lambda_j} \frac{P_j}{P_{\mathrm{RX}}} \theta}, \tag{6.2}$$

where $\{g_j\}$ is exponentially distributed with unit parameter ($\lambda_0 = \lambda_j = 1$), θ is given from Eq. 5.3 and N_0 is receiver's thermal noise power. For example, the outage probability for link $P_9 \to P_{7-2}$, assuming P_{10} and P_{12} also transmit at the same frequency channel, is calculated equal to 85.3%. Similar calculations can be easily conducted for all links with remaining interference.

Chapter 7

Conclusion

In this work, cognitive principles based on graph coloring on top of 802.11s were presented for effectively reducing the amount of interference created from (internal) wireless nodes placed at water tanks and pumping stations, as well as (external) 802.11 nodes in the city. The conducted experiments demonstrate that the employed techniques can achieve significant reductions of packet error rate, despite the practical constraints in city-wide deployments.

Several challenges were encountered during the implementation of the project. Because of the long range of the network links, the use of the directional antennas made the alignment procedure very hard to accomplish. Furthermore those long range directional links were susceptible to weather conditions, as heavy winds were enough reason for link failure. The solution to these problems was to replace one directional antenna with a sector antenna in long range links.

Another challenge was the failure of the network stations. This could be caused due to power failure or board failure, among other reasons.

Future work includes the replacement of the antennas with other that offer better gain. Also the Alix boards themselves could be replaced with smaller more power friendly boards. A candidate under consideration is the Olinuxino A10 Lime board. Finally maintenance work is necessary in order to maintain the operation of the network nodes.

Bibliography

[1] P. N. Alevizos, E. Vlachos, and A. Bletsas, “Factor graph-based distributed frequency allocation in wireless sensor networks,” in *Proc. IEEE GLOBECOM*, Austin, TX, Dec. 2014.

[2] S. D. Assimonis, A. Theopoulos, and T. Samaras, “A new high-gain and low-complexity pattern-reconfigurable antenna,” in *Proc. IEEE Europ. Conf. Antennas and Propag.*, Lisbon, Portugal, Apr. 2015.

[3] A. P. Subramanian, H. Gupta, S. R. Das, and J. Cao, “Minimum interference channel assignment in multiradio wireless mesh networks,” *IEEE Trans. Mobile Comput.*, vol. 7, no. 12, pp. 1459–1473, December 2008.

[4] D. Karger, R. Motwani, and M. Sudan, “Approximate graph coloring by semidefinite programming,” *J. ACM*, vol. 45, no. 2, pp. 246–265, Mar. 1998.

[5] J. Riihijarvi, M. Petrova, and P. Mahonen, “Frequency allocation for WLANs using graph colouring techniques,” in *Proc. IEEE WONS*, Washington, DC, Jan. 2005, pp. 216–222.

[6] E. G. Villegas, R. V. Ferré, and j. P. Aspas, “Implementation of a distributed dynamic channel assignment mechanism for IEEE 802.11 networks,” in *Proc. IEEE PIMRC*, Berlin, Germany, Sept. 2005, pp. 1458–1462.

[7] A. G. Dimitriou, D. Ntilis, P. Oikonomakos, A. Inglezakis, V. Papadakis, and A. Bletsas, “Designing limited-eirp, long-distance links for water management networks using polarization diversity and redundant routing paths,” in *Proceedings of the 1st ACM International Workshop on Cyber-Physical Systems for Smart Water Networks*, Seattle, WA, USA, Apr. 2015, ACM.

[8] John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris, “Architecture and evaluation of an unplanned 802.11b mesh network,” in *Pro-*

ceedings of the 11th Annual International Conference on Mobile Computing and Networking, New York, NY, USA, 2005, ACM.

[9] "IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1–2793, March 2012.

[10] G.R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke, "IEEE 802.11s: The wlan mesh standard," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 104–111, February 2010.

[11] S.M.S. Bari, F. Anwar, and M.H. Masud, "Performance study of hybrid wireless mesh protocol (hwmp) for IEEE 802.11s wlan mesh networks," July 2012, pp. 712–716.

[12] W. C. Jakes, *Microwave mobile communications*, IEEE Press classic reissue. IEEE Press, 1974.

[13] D. Ntilis, P. Oikonomakos, V. Papadakis, A. Inglezakis, A. G. Dimitriou, and A. Bletsas, "Frequency planning for a multi-radio 802.11s city-wide water management network," in *Proceedings of the 1st ACM International Workshop on Cyber-Physical Systems for Smart Water Networks*, Seattle, WA, USA, Apr. 2015, ACM.