UNIVERSITY OF GENOA

Department of Computer and Information Science

Academic Year 2004/2005

Master Thesis

Applications of dynamic routing algorithms on wireless networks in harbour environments

Research carried out at Genoa's Fantuzzi Reggiane Electronic Department (FRED)

Thesis Advisor	Secondary Advisor	External Advisor
Prof. G. Dodero	Prof. M. Ancona	Dott. F. Parodi

Candidate: Daniele Venzano

Contents

List of Figures

LIST OF FIGURES

Introduction

The integration of information services in harbour activities is going to raise in importance and scope wherever there is the need to manage high levels of commercial traffic in a quick and efficient way. The availability of a TCP/IP network able to connect all machinery and operators on the yard with harbour's central offices facilitates a number of operations, from simple communications about goods and containers, to remote diagnostic of machinery, to tracking data with GPS and automatic guidance systems.

A wireless network with dynamic routing (MANET) lends itself very well to provide a flexible topology on wide harbour yards, with moving network nodes and a difficult environment full of obstacles and interference sources. Moreover the MANET solution offers big advantages also on the economical side, since it is not necessary to deploy fiber optic cables on working yards, with high costs of infrastructure building and the disruptions of normal harbour activity.

In this thesis a complete hardware and software system, easy to install, has been developed. It provides a MANET with the possibility to deploying network nodes on fixed and mobile positions. Dynamic routing is done by the OLSR protocol and its operative characteristics have been studied both, on a test bench and in the field.

To complete the system a remote management interface has been developed with web technologies. The hardware used has been assembled and tested to endure the extreme ranges of temperature, humidity and salinity typical of an harbour environment. The system, developed at the Fantuzzi Reggiane Electronic Department of Genoa, has been developed as a complete commercial product to be deployed at the Interporto Campano near Nola, Naples and is available in the product catalogue of Fantuzzi Reggiane under the name of MeshAP.

The chapters distribution in this thesis follows, more or less, the temporal development of the project. In the first part the results from research in available technologies is presented, both from the point of view of wireless networks and dynamic routing algorithms. The second part examines the hardware used and the tests developed to insure operation in harsh conditions. Then the software developed for the integration in the Nola project and the subsequent marketing is described. Finally the structure and the

integration of the MeshAP in the Nola terminal is described.

Part I

Evaluation of current state of the art

Chapter 1

Ad-Hoc mobile networks

This chapter describes the technologies at the basis of this thesis. Only levels from one to three of the ISO/OSI[15] model will be taken in consideration, moreover basic notions on Ethernet (IEEE 802.3[3]) and TCP/IP[18, 19] networks are taken for granted.

1.1 ISO/OSI model levels

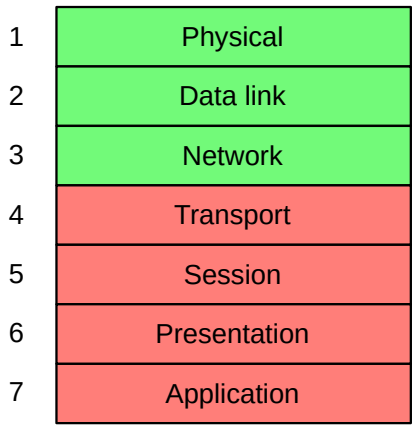

Figure 1.1: Scheme of the ISO/OSI model, in green are the levels that need to be redefined in a wireless network with dynamic routing, in red those that need to remain unchanged to insure a complete interoperability with existing software and technologies.

A wireless network with dynamic routing based on the 802.11 standard needs to implement the following levels of the ISO/OSI model:

- Physical and data link layer. In our application we decided to use the 802.11 suite of protocols for its wide availability and low cost. Other

technologies offer built-in dynamic routing capabilities at the data-link layer, giving a transparent view of the network to upper levels.

- Network layer. It is necessary to take into account the mobility of nodes when packet routing decisions are made. The various dynamic routing protocol studied do not change directly the IP protocol, but update its static routing tables using a number of parameters that are exchanged between all participant nodes.

The working hypothesis is that on layers above the IP protocol (ISO/OSI level 3) there is no knowledge of lower layers. In the real world, however, dynamic routing is not perfect and there is a high probability of small periods of time with disconnection between single or groups of nodes during the normal network operation.
Generic software is not robust against this type of error because disconnections on wired networks are rare and can be managed manually. On the contrary, in the application described in this thesis all software must be able to run in a non supervised way, since single nodes will be installed in almost inaccessible positions and each period of downtime of a single node can have heavy consequences on the whole system functionality.

1.2 Wireless networks

Wireless networks are gradually assuming a greater importance in all data transmission applications. Their great flexibility means quick installations and immediate connectivity, with less fault possibility and lower deployment and maintenance costs. Dropping the requirement for data transmission cables is of extreme importance in those applications where extensive masonry or excavation work is impossible due to historical or architectural constraints[11].
Depending on the technology and antenna type it is possible to provide a generic coverage of a wide area, useful, for example, to extend Internet access to an area difficult to reach with cables. At the same time different antennas can be used to create a point to point connection over long distances, for example to connect two building with wireless nodes on their roofs.

On the other hand wireless networks have interference problems that limit their range and bandwidth. At the time of this writing, latest commercial technologies based on microwaves offer theoretic bandwidths of 108Mbs, but those speed are available only in ideal conditions, while in reality a quarter of the advertised speed is more probable because of noise of various kinds. Moreover covering distances longer that one kilometres requires costly amplifiers and licensed radio frequencies because of limitations on free radio bands (as the one used by the WiFi system).

There are wireless systems based on infra-red light instead of radio waves. They use LASER receivers and transmitters that give long range transmissions with a very small probability of eavesdropping. But these system can be used only for point to point connections and require a precise alignment between nodes. In [16] there is a study on how this technology can be used to connect many Indian villages at low cost. In one of the earliest experiments a distance of 20 kilometres was covered with a video camera and a laser pointer, being limited on the bandwidth only by the number of frames the camera was able to record. Earth $\leftrightarrow$ satellite and satellite $\leftrightarrow$ satellite communications systems are also being studied[25].
The 802.11 standard has an infra-red working mode, as described in the 1.3 section.

1.3 WiFi networks (IEEE 802.11)

The IEEE 802.11[4] standard describes the physical and medium access layers for wireless communication on local or metropolitan networks using free electromagnetic frequency bands. The majority of available devices uses the 2.5Ghz band, ma latest products can also use the 5Ghz band, that is less cluttered by microwaves ovens, cordless phones and remote commands.

The first revision of the standard, of 1999, provides speeds of 1 or 2 Mbit using two complementary techniques for the physical layer: *Direct Sequence Spread Spectrum* (DSSS) and *Frequency Hopping Spread Spectrum* (FHSS). Subsequent revisions of the standard have brought the bandwidth up to 54Mbit (802.11g) by using *Orthogonal Frequency Division Multiplexing* (OFDM) that gives a greater transmission speed at the cost of reduced noise resistance and accordingly of the effective communication range.

A third mode is described, not compatible with the first two, based on frequencies between 850nm and 950nm (infra-red light). This mode provides communications with a maximum range of 20 meters and is able to use reflections of infra-red light to remove the requisite of devices' alignment[1]. The short communication range and the strong interference from the sun light prevent a wider adaptation of this technology.

At the MAC level two working modes are provided by the standard. emphInfrastructure and *Ad-Hoc*, described respectively in the sections 1.3.2 and 1.3.3.

Devices that implement 802.11 standards can obtain the WiFi (Wireless Fidelity) certification that guarantees the interoperability between different manufacturers. The acronym WiFi is now part of the common language and indicates all radio devices that use the standard 802.11 protocol family.
In the industry still are installed and requested devices that operate on the

[1] As, instead, it is required for the IRdA protocol, used by hand-held devices and cellular phones

standard 802.11b, having a maximum speed of 11Mbit and a good noise resistance. Moreover the propagation characteristics are well known and the wide range of different products covers the requirements of difficult working environments (temperature, humidity and vibrations).
On the contrary, the standard 802.11g is not well known and the wider bandwidth is often not needed for current industrial applications and the reduced communication range is counter productive. This and the fact that 5Ghz communications requires different antennas, that are more costly (because of a smaller market), is causing a very slow migration toward the 802.11g protocol.

1.3.1 Security

Radio waves transmissions are inherently insecure, since they are easy to intercept with simple, low cost equipment. For this reason, when it is needed a radio communication, cryptographic techniques are used to make it more difficult to read exchanged messages. During the second world war these techniques have had great development, the same as cryptanalysis and the decoding of encrypted messages.
The 802.11 standard provides clear and encrypted transmissions by using an RC4 cypher with a key of 64 or 128 bit that must be known by all users in the simpler and more used mode. This type of encrypted communications, called *Wireless Equivalent Privacy* (WEP) offers, by current standards, a very low level of security, useful only for non-confidential communications. As a matter of fact a number of weaknesses of the RC4 algorithm and attacks to the WEP implementation are known that can find the key by interpreting a few millions packets.[2].[13]

While waiting for the new 802.11i IEEE standard, the WEP algorithm has been improved to cover the known security holes. The *Wi-Fi Protected Access* (WPA)[1] mode uses a scheme of dynamic keys, the can be different for each user of the system, to reduce the probability of discovering the key by comparing statistically the network traffic, offering at the same time the possibility of user authentication, a functionality not offered by WEP.

1.3.2 Infrastructure mode

In this mode every area must be covered by an Access Point (AP), to whom all nodes refer to coordinate transmissions. The AP gives out its presence by transmitting at regular intervals special messages (Beacon Frame) that contain the parameters needed to connect. Each node keeps a list of all accessible APs and chooses the one with the strongest radio signal, sending an association frame.

[2]Even if a million packet seems to be a lot, on heavily used networks the attack can be completed in less than an hour.

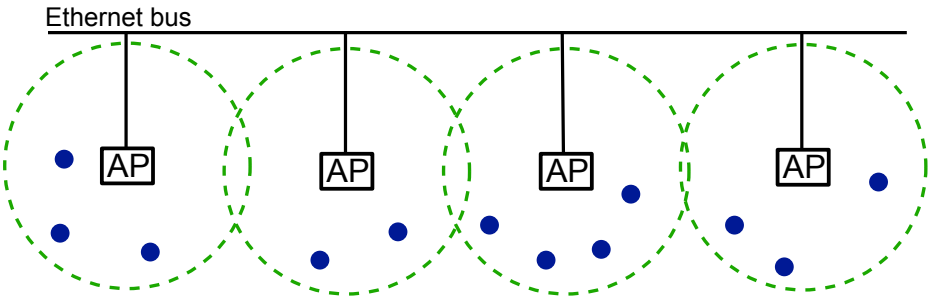

Figure 1.2: WiFi network in infrastructure mode. Every Access Point gives network access to all nodes within its own radio cover.

The standard does not describe a roaming functionality where one node disconnects from an AP to reconnect on another one dynamically, leaving the details to the manufactures that came up with different, and often incompatible, protocols. The Wireless Distribution System (WDS) is trying to provide a standard way to bridge packets between APs. In figure 1.2 is visible the diagram of an infrastructure mode WiFi network with four APs connected by an Ethernet cable that bridges packets from nodes connected to different APs. This network can be easily connected to the Internet, providing access to all of its nodes.

1.3.3 Ad-Hoc mode

The Ad-Hoc mode is thought as a quick method to exchange data, with a short range and a limited number of nodes. As a matter of fact at this level it is possible to concur simply on a radio channel and a network name (SSID) to start exchanging data. To use the IP protocol, also addresses need to be exchanged. As can be seen in figure 1.3 nodes can communicate between couples, but the 802.11 standard does not provides any routing mechanism, so that it is not possible to use an intermediate node as bridge without another software layer.

1.4 Dynamic routing

Dynamic routing protocols provide a way for nodes in an Ad-Hoc network to work both as access points and routers, providing a way to build an extended network in a short time, without the complexities associated with cabling, connection parameters and routing tables. Next, in the section 2.3 different routing protocols are examined, according to the criteria established on on the 2.1 chapter.

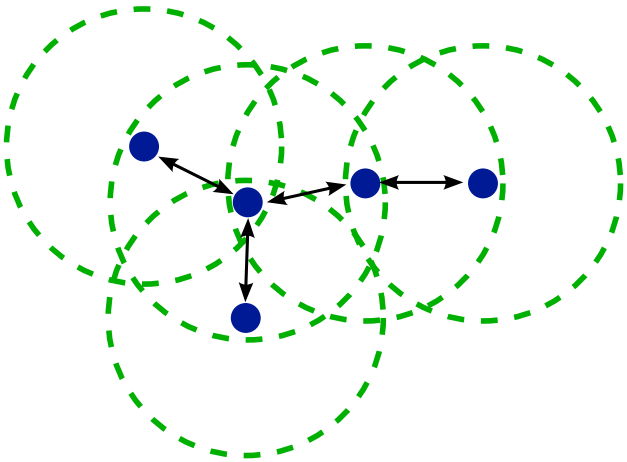

Figure 1.3: WiFi network in Ad-Hoc mode. The devices can communicate between couples, when they are within radio range.

A network of this kind is extremely useful in different scenarios, from coordination between emergency services in disaster areas where the normal communication infrastructure (cables, cellular towers) has broken down, to the possibility to provide Internet services in large areas at low cost, being able to cover difficult zones with additional nodes, without reconfiguration of the already installed ones.

Bringing to the extreme the concept of dynamic network, there are *sensor networks*, also called *dust networks*. Those are network with thousands of nodes, that can be deployed by aerial or naval means. Each node is very simple, with a small cost and provides data that contributes to the traffic of adjacent nodes, offering the possibility to monitor large areas at low cost and without on-site operators[14]. This type of network has, today, only military applications, but interest is beginning to grow for commercial applications, such as the monitoring of large plants, as refineries or oil pipelines. For sensor networks, it exists an IEEE standard, the 802.15.4, and a consortium, similar to the WiFi one, called ZigBee, that extended the standard with additional layers to provide security and routing, with very low power requirements[5, 8].

Chapter 2

Dynamic routing algorithms

In preparation for the work that was going to be done, a classification of existing algorithms for dynamic routing has been done. A number of evaluation criteria were chosen to provide an objective comparison.
For some algorithms different implementations are available. In the evaluation phase these have been considered, checking their status and activity of the development group.

2.1 Classification

Routing algorithms can be classified according to different parameters and functionalities. The fundamental characteristic is the method used to build and maintain the routing tables.
Two approaches exists:

proactive : the topology of all the network is maintained and updated on fixed time intervals of a few seconds. All nodes know how to reach each other every instant.

reactive : the routing path is built every time it is needed and a cache is used for frequently used paths (lazy approach). These algorithms have a characteristic delay every time a packet needs to be sent to a new destination.

While a proactive approach allows a fast communication without delays, it requires a constant bandwidth and node's resources overhead on the network to keep the routing table updated. On the other hand it is suitable for those scenarios where all nodes want to communicate between themselves without preferred paths.

On the contrary a reactive approach will establish a routing path only when it is needed, limiting the use of resources to the bare minimum. However the cache needs to be very effective to prevent delays caused by new connections or changes in topology.

It is also possible to use a classification based on the hierarchical arrangement of nodes. Most of the algorithms are *flat* and the research done provided with only one hierarchical algorithm, the *Zoned Routing Protocol* - ZRP, but it has not been possible to find an implementation of it.

flat : all nodes participate in the same way to routing.

cluster based : nodes are divided in groups, every one of them with a special node that knows the topology of its sub network and that manages all the traffic to and from its group. (ex. ZRP - Zoned Routing Protocol).

It is needed to impose a hierarchy only if the number of nodes is big enough to reach limitations in the operating system's routing tables or if there is an evident difference between nodes, with some of them having strategic positions or particular hardware configurations.

When a packet needs to be routed and more than one route exists to get to the destination, a metric of each path is taken into consideration and the path with the lowest value is chosen. This metric is a numeric value that represents the distance between nodes. Only in very particular cases this distance is really a physical distance. Usually it is simply calculated as the number of hops needed to reach the destination (2.1).

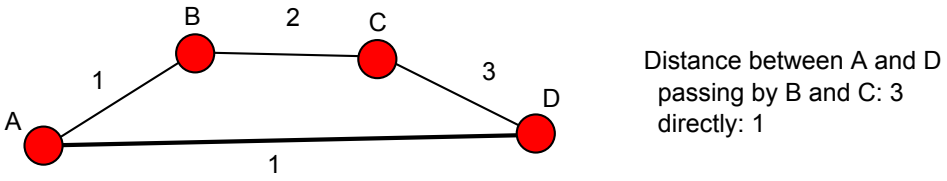

Figure 2.1: Distance calculated on the number of hops. The best path is the direct one between A and B, drawn with a thicker line.

In particular cases other parameters can be used to calculate the distance between nodes, for example in wireless networks the following factors can be useful:

- available bandwidth (fig. 2.2)
- power needed for the transmission
- signal to noise ratio
- association stability
- informations about the physical location of the nodes (GPS)[27]

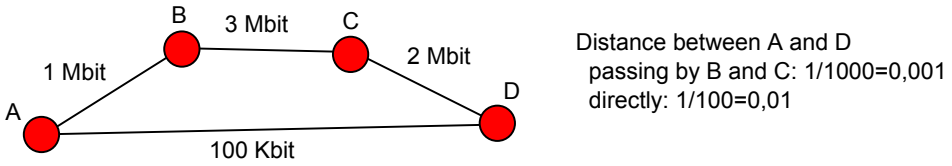

Figure 2.2: Distance calculated on the bandwidth available between the nodes. The best path is the one passing through nodes B and C.

Special metrics, while very useful, are often problematic to use in practice because of the difficulty of generating sufficiently accurate information with consumer hardware. Moreover different operating systems offer very different APIs for low level access to data, causing difficulties in maintaining multi-platform implementations.

The last important point that is strictly connected with the implementation of an algorithm is whether it is done in *kernel space* or in *user space*. Both solutions have advantages and disadvantages, as follows:

kernel :

In favour

- higher performance, since the packets need not to be copied in the memory space of the user process and then copied again in kernel space once the routing decision has been taken.
- better integration with the operating system

Against

- bigger difficulties in debugging, the internal structure of the kernel needs to be known and specific knowledge on the hardware architecture is needed to use a *kernel debugger*.
- a revision is needed with each new version of the kernel, in particular in Linux where internal ABI (Application Binary Interface)/API (Application Programming Interface) are not stable and can be changed from one version to the next.

user space :

In favour

- development and debug is simplified, high level languages can be used and libraries are available for all sort of tasks.
- independence from the kernel version, since most of the calls to the subsystem are hidden behind the standard interfaces of the C library for Linux and official APIs for Windows.

Against

- lower performance, every packets needs to be copied twice, to and from user space before it can be written on the network hardware buffers.

Some development teams tried to find a middle ground, implementing part of the algorithm in kernel space and part in user space. These solutions try to reduce the performance impact at the cost of increased complexity of the code to write and maintain.

2.2 Choosing the evaluation criteria

To decide which algorithms, among the existing ones, were the most suitable for the field of application, a list of criteria has been compiled. This list, also useful to evaluate the maturity of a particular implementation, is available here for reference, compiled without a particular order. Starting from section 2.2.1 the criteria and motivations behind each item will be explained.

1. Licence of the implementation that allows modifications to the source code
2. Maximum number of concurrent nodes active and communicating in the network of at least 40
3. Existence and maturity of the implementation
4. Implementation on multiple operating systems
5. Evidence of tests on real environments, with real hardware
6. Security

2.2.1 Licence

The necessity of an open source licence has been decided, so that modifications to the source code of the algorithm's implementation could be done. This is needed for any changes needed during the integration in the current Fantuzzi Reggiane system and to guarantee future maintainability if the current developers stop their contributions.

All implementations where the source code was available on the Internet are distributed with GPL or BSD licences. This requirement has excluded several proprietary solutions with algorithms often covered by software patents and described vaguely on the respective web sites.

2.2.2 Network size

Having to accommodate a growing of the network after the first installation and the use in areas of vastly different sizes, it is a requirement that the algorithm does not have a maximum active node count hard coded.

Among the algorithms taken into consideration only one (LUNAR[7]) suggest the use in a network with less than ten nodes. All others do not give restrictions or the limits are so high that can be ignored.

2.2.3 Maturity of the implementation

Taking into account the long times needed for the development of an implementation complete enough to be used in production, it has been decided to consider only algorithms with a source code implementation. Moreover a preference was given to active development teams.

2.2.4 Operating systems

The nodes that will be part of the network built in this thesis will use GNU/Linux as operating system, with a 2.4 kernel. Algorithms implemented also on other operating systems will be preferred, in particular on Microsoft Windows and handhelds OSes (Pocket PC, Linux Familiar). This requirement is connected to the external nodes problem, see section 2.2.7.

Many of the algorithms have been excluded thanks to this requirement because they exist only as simulations under the NS2 framework or similar applications. The rest all have a Linux (or other Unix like OS) implementation. It is very difficult to find this kind of software for Windows or Pocket PC.

2.2.5 Tests and tries

Implementations of dynamic routing algorithms are usually tested on a simulator. These tests give the possibility to see if the algorithm is working as intended, but cannot reproduce all the problems that can come up during the deployment in a real world scenario. For this reason the results of tests with at least three distinct nodes, with 802.11 connectivity have been searched.

This requirement was used to establish a preference for algorithms with documented tests. Unfortunately it is quite rare to find this kind of informations as most of the testing phase is done entirely on simulators.

2.2.6 Security

It is important that the algorithm can work with WEP cryptography activated. Moreover implementations with access control rules are to be pre-

ferred, so that unauthorized third party nodes cannot connect to the network.

2.2.7 External nodes

A requirement was also proposed as to the possibility to connect external WiFi devices to the dynamic network without additional software[1]

This requirement was too restrictive and no known algorithm or implementation considered such problem. After an email exchange with the author of `olsrd` the following conclusion was reached: it is possible to use only one node of the MANET network as router to an external network without multihop capability. It is not possible to use more gateway nodes because of big problems redirecting the existing connections and the difficulty from the external node to establish a default route.

2.3 Algorithms

Two algorithms were immediately visible as most developed: AODV e OLSR. In time different implementations were built with added complexity and less prototypical interfaces. Currently these two algorithms have been formalized in two RFC to establish a fixed point where all implementations need to converge to be interoperable. In the next sections also other algorithms are described.
Following this evaluation it has been decided to use the OLSR algorithm, with the implementation developed by Andreas Tønnesen at Oslo university, that is described extensively in the 3 chapter.

2.3.1 AODV - Ad-hoc On-demand Distance Vector (RFC 3561)

The algorithm is of the reactive kind and builds the path each time packet needs to be sent to a new or unreachable destination. This causes an initial communication delay for the time needed to get a valid routing path from the network. Once the path is established, it is kept in the cache of all intermediate nodes with the information on which is the next node to reach in the path. When one of these point to point links breaks, the last reachable node sends back a service message that causes the removal of the path from the cache and begins the search for a new path starting from the packet source.

Two implementations exist at a good development status, but both require a kernel module to be used, since the algorithm needs to know when

[1]These devices would use the rest of the network as routing domain, but would not participate in it themselves.

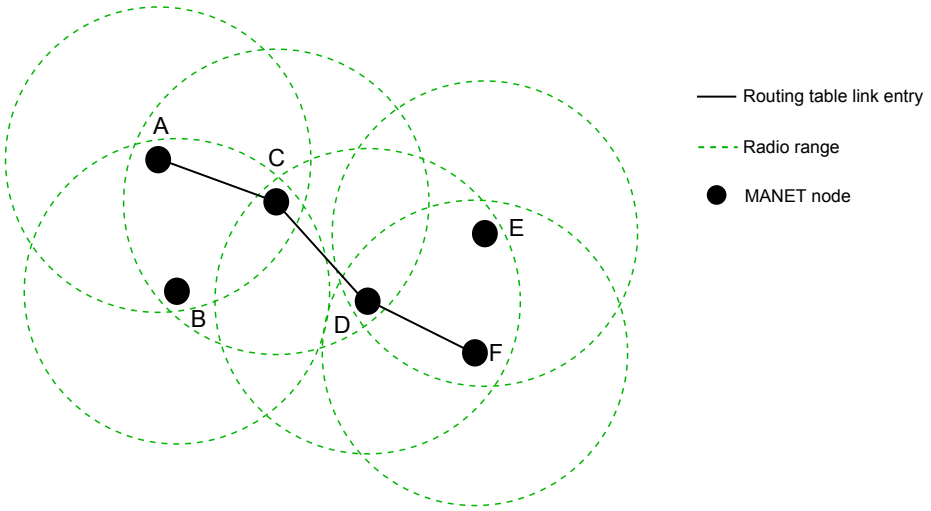

Figure 2.3: Links available in the routing tables for a path from node A to F

a packet is going to be discarded because of an invalid route[2].
Alternatively it is possible to do routing decisions in user space, rewriting sender and receiver IP address at each jump, with two memory copies for each packet (kernel → user and user → kernel).
A Windows implementation is also available, but it has not been possible to check the interoperability with the Unix/Linux versions. Implementations:

NIST[26] : kernel module for Linux 2.4, no real test evidence, only simulations. Source code release in the public domain, no licences needed.

UU (Uppsala University)[12] : implementation part user space, part kernel space. Uses netfilter to exchange packets between the two parts. Test for real with at most five nodes and 4 hops. Lots of simulations. The last version has a kernel module to load to reduce the performance penalty due to the memory copies. Open source licence (GPL).

In favour:

- Low bandwidth use for routing path maintenance, leaving more space for data transmission.
- Good use of the system resources with the kernel module.

Against:

[2]All operating systems discard packets for which there is not a valid route and some code needs to be added in the networking stack if such events need to be acted upon.

- High delay (up to a few seconds) with each new connection
- In-kernel implementation, complex management of new software versions
- Waste of resources with memory copies. Especially important on embedded systems.
- Very limited tests.

2.3.2 MIT SrcRR

Based on DSR, it is used in a metropolitan network in the Roofnet project[9] of the Massachusetts Institute of Technology (MIT). It is a proactive algorithm and uses a metric based on the connection quality, measured by estimating the number of times a packets needs to be retransmitted before being received correctly. The metric of a multihop path corresponds to the sum of the metrics of each hop, allowing the exclusion of longer paths or paths with high packet loss. The project is very active and currently in use, but it is engineered for nodes with fixed locations or very low mobility. The implementation is only for GNU/Linux and is distributed on a LiveCD to ease the installation of new nodes.

2.3.3 LUNAR - Light Underlay Network Ad-hoc Routing

LUNAR[6, 7] is developed by Uppsala university and is an hybrid algorithm, both proactive and reactive. It is implemented as a Linux kernel module and no other versions exist. It has been thought to be extremely easy to install and use, for example, as part of its execution, it assigns IP address to single nodes, a task usually left to the user. It is also thought for very small networks, at most of ten nodes, and of small radius, with nodes distance under 50 meters.

2.3.4 Other algorithms

DSR-Monarch[10] : Exists only an implementation for FreeBSD[21], an older version for Linux was abandoned and is no more available on the Internet. Had the problem, quite fundamental, of not being able to work with TCP connections.

TORA : (Temporally Ordered Routing Algorithm) needs synchronous clocks, it proposes to operate well in very dynamic environments. It seems to be developed by the US Navy, but it was not possible to find an implementation. Some references point to the *Ad-Hoc Networking Research Group* of Maryland university, but report also big stability problems.

Chapter 3

Optimized Link State Routing - OLSR

OLSR is the algorithm chosen for the implementation of the dynamic routing system. In this chapter the motivations behind this choice are described. Also the algorithm itself and its implementation are described, clarifying why that particular one is being used. In the following text, to exclude misinterpretations, OLSR will be used to refer to the algorithm, while `olsrd` for the implementation.

3.1 The algorithm

OLSR has been formalized in a RFC[23] proposed by the Hypercomm project of INRIA[1]. As of today it is still in an experimental phase, but the algorithm kernel is well defined and the remaining discussion regards some extensions that are being thought out.

The existence of an official formalization allows the different implementations to converge toward a common set of basic, interoperable functionalities. This convergence phase is slowly happening, but as of this writing not all implementations are interoperable yet.

In the last two years two *OLSR Interop & Workshop* events have been held, the first one in the United States, the second in Paris, sponsored by big companies, to ease the process of integrations and to strengthen the community that revolves around OLSR.

3.1.1 Operation

OLSR is an algorithm of the proactive kind, so at start-up it collaborates with neighbour nodes to build a routing table containing paths toward all

[1] Institut National de Recherche en Informatique

existing nodes. After this initialization, nodes exchange periodic messages to keep their topology data updated.

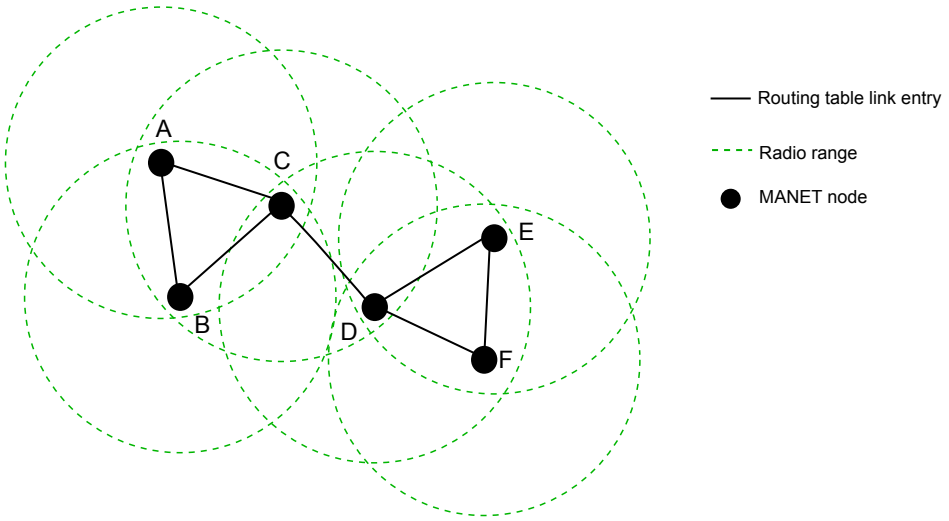

Figure 3.1: OLSR: nodes know the paths toward all other nodes

As the name of the algorithm suggests, OLSR uses the broadcast of the state of all known connections for each node to allow the reconstruction of the network topology. However this broadcast is optimized to limit the bandwidth waste by using a system called *MultiPoint Relaying* (MPR).

The MPR technique is born from the observation that in a non optimized broadcast situation each node receives multiple times the same information, causing a big waste of bandwidth and system computation. This situation can be improved by choosing a particular subset of nodes that can retransmit the information. Figures 3.2 and 3.3 show the difference in the number of retransmissions.
In OLSR each node chooses a subset of its symmetrical neighbours[2] so that each second neighbour is always reachable from this subset. It has been demonstrated[24] that this choice is an NP-complete problem, and the RFC, in section 8.3.1, describes a simple heuristic algorithm to define the MPR subset.

The MPR system is used by default for the retransmission of messages and is part of OLSR that has to be available in all implementations, allowing all nodes in a network to correctly retransmit messages even without comprehending the contents.

OLSR uses UDP packets to transfer control informations, each packet

[2]Node for which exists a direct, one hop, path in both ways. Something which is not always true in radio communication.

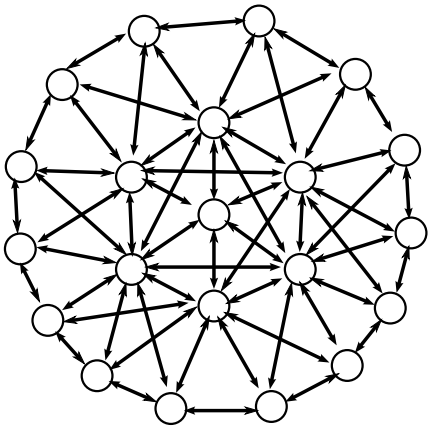

Figure 3.2: Full broadcast: each node receives the information multiple times, directly and via multihop paths on neighbouring nodes.

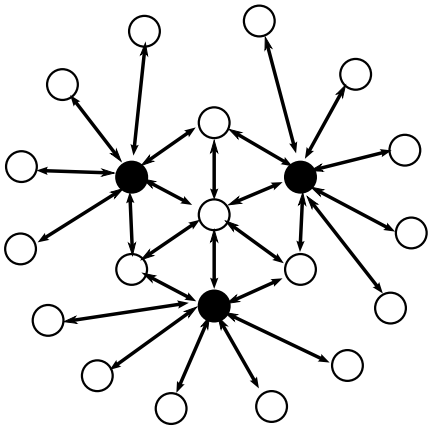

Figure 3.3: MPR broadcast: the central node selects a number of neighbouring nodes as MPR and only these do the retransmissions. The number of communications is much lower.

can contain more that one message, coming from different nodes, to better use the transmission time. A generic packet is described in the 3.1 table. The messages required by the basic OLSR functionality are:

- HELLO: sent at regular intervals, they are used to detect neighbour nodes, communicate MPR node lists and to do link sensing
- TC: are used to send topology data from the point of view of each node
- MID: used by nodes with multiple interfaces to declare their existence to the network.

To avoid synchronized transmission (packet collisions on the network), the RFC establishes that nodes should wait a random interval of time before each communication.

During the normal functioning each node updates the content of several tables, according to the content of the messages it receives. Typically each data row memorized in these table has an expiry time associated, after which the row is no more valid and should be deleted. The most important tables are:

- Neighbour set: set of the first neighbours, meaning those from which an HELLO message has been received.

Packet size		Sequence number
Message type	Validity time	Message size
Sender address		
Time to live	Jump number	Sequence number
Message		
Message type	Validity time	Message size
Sender address		
Time to live	Jump number	Sequence number
Message		

Table 3.1: Generic OLSR packet format containing two messages. The IP protocol headers are not shown.

- 2-hop neighbour set: set of nodes published as first neighbours from each node belonging to the neighbour set. The intersection between this and the preceding set is empty.
- MPR set: set of nodes that act as MPR for this node
- MPR selector set: set of nodes that chose this node as MPR
- Topology set: each node in the network has an entry in this table that contains its address and that of the node that published it as neighbour.

Moreover each node keeps a routing table, in common with the operating system, where all changes in the tables described are reported. Each time a change is detect, a shortest path algorithm is applied on the indirect graph that has as nodes all the known network nodes and as links the bidirectional links between first neighbours, reconstructing the full topology and the routing table.

3.2 The implementation

As of today several implementations exist for OLSR, at different levels of maturity. INRIA is developing OOLSR in C++, the Naval Research Laboratory of United States is working on NROLSR, the Communications Research Center in Canada was (till 2003) working on CRCOLSR and finally the *Laboratoire de Recherche en Informatique* of Paris-sud university is actively developing QOLSR, giving particular importance to *Quality of Service* selection criteria. Other than these there is the most mature implementation, that is `olsrd`[2], developed in a doctoral thesis at Oslo university by Andreas Tønnesen and still in full development.

`olsrd` has been tested in a mixed wireless/fixed network with about thirty nodes by Tønnesen during the Wizard of Operating Systems conference in Berlin in 2004. Is completely user space and uses standard mechanisms to update the kernel routing tables. Moreover it has a plugin system that allows functionality extensions without having to modify the code application. There already exist various plugins that offer additional services, such as the communication of battery status or node authentication. Another point in favour is that it is distributed with an open source licence, guaranteeing an extended life to the project even if the author should stop working on it.

Currently `olsrd` is being used in Lille, France for the Lille Sans Fil[3] project that intends to offer a coverage of the entire city (and possibly of the whole region) with a wireless network with dynamic routing. `olsrd` is part of the Freifunk[4] project for the diffusion of free networks in German-speaking areas.

Versions for Linux, Windows (2000 e XP), Os X and Linux Familiar (for handhelds) are available. This is important to allow the interoperation of the system developed in this thesis with systems produced by other manufacturers[5].

`olsrd` has some characteristics that became evident during this thesis development: in particular the parameters used to decide when a link is good enough had to be studied and fine tuned. These parameters are strictly dependent to the network type being deployed.

3.2.1 Hysteresis

The simpler method that `olsrd` can use to establish if a link is good enough is based on hysteresis. This method does not allow a weight to be associated with the link, it simply establishes a boolean information of the type available/not available. To work it needs three parameters, two thresholds and scaling value. When the link value drops below the minimum threshold, the link is given as no more active, while when it goes over the maximum threshold, it is set as active. The more the two thresholds are near, the more the routing is unstable, but better suited for rapid changes in the topology.

Two formulas are used in the calculation if the hysteresis for each link, one applied going up, when an HELLO message has been successfully received:

$$LinkQuality = (1 - scaling) * LinkQuality + scaling$$

and one going down, when an HELLO message is lost

$$LinkQuality = (1 - scaling) * LinkQuality$$

[3] http://www.lillesansfil.org
[4] http://www.freifunk.net
[5] See also the third requisite established during algorithms evaluation, section 2.2.7

since HELLO messages are sent at regular time intervals, it is easy to check if a message was lost during transmission.

3.2.2 Link quality

`olsrd` offers another method to determine the availability of a link, based on the number of lost packets in a fixed time interval. This method allows the assignment of a weight to each link according to the probability that a transmission will end correctly.
The definition of this method is not defined in the OLSR's RFC and is still being formalized.

3.3 Choice motivations

The decision to use OLSR and `olsrd` instead of something else was based more than anything else on the fact that OLSR is developed by a large, very active community able to offer support and evolution for a long time to come. Other algorithms, instead, seemed to be almost abandoned, often with web sites not updated since several years. This aspect was given a bigger importance other than technical motives. For example in the harbour environment communications happen only between two fixed subsets of nodes, with intermediate ones that are used only as repeaters. With this insight a reactive algorithm, such as AODV, could have been more specific. With OLSR lots of reachability data is managed that will never be used.

On the other hand, the network established with OLSR in the future could be used also to transport voice, to remove the current use of portable radios in the harbours. Such an application would make an extensive use of the network and would benefit from the proactivity of OLSR.

Follows a list of in favour and against arguments kept into consideration during the choice of OLSR and `olsrd`.
In favour:

- Very active development team
- Wide test coverage, with a high number of nodes
- Implementation completely in user space
- Plugin system
- No delay in finding new paths

Against:

- Use of bandwidth for maintaining paths that could never be used

- Waste of resources (memory and processor) to keep updated informations on paths that could never be used
- Initialization time, measure between 2 and a dozen seconds, according to the number of nodes and network topology.

Part II

Deployment at Interporto Campano, Nola (NA)

Chapter 4

Hardware

In this chapter the hardware used for the deployment in Nola's network is examined. In particular the three main components are described: two IBM compatible systems, the BlueBox and the Display, designed and assembled by Fantuzzi Reggiane and a WiFi router from Linksys[1] where the operating system was substituted with a modified one tailored for this thesis. Finally there is a brief description of the WiFi antennas used and are explained the hardware choices.

4.1 BlueBox and Display

The BlueBox is a computer designed and built to endure extremes in temperature and vibrations. The metallic box that encloses it is completely water-proof and allows its installation outdoors.
On one side the connectors are available for serial ports, Ethernet, WiFi radio antennas, and power. There are no connectors for video and keyboard because normally the connection happens via the serial port or with protocols like telnet[20, 17] or ssh on the Ethernet connection. The BlueBox is very versatile and can be used both as simple repeater of radio signals, by installing it, for example, on the top of light pole, or as processor for positioning calculations with GPS signals on moving machinery.

The display has the same hardware as the BlueBox, explained below, but has also an LCD screen (resolution 800x600) equipped with a touch screen for the user's interaction. The Display is installed in the operator's cabin, where it is used to show and interact with a big number of different informations and parameters. For example external cameras can be shown to ease the manoeuvres, or diagnostics on the machinery status with alarm signals for overloads or malfunctions, or a simple list of operations to complete.

[1] A Cisco brand

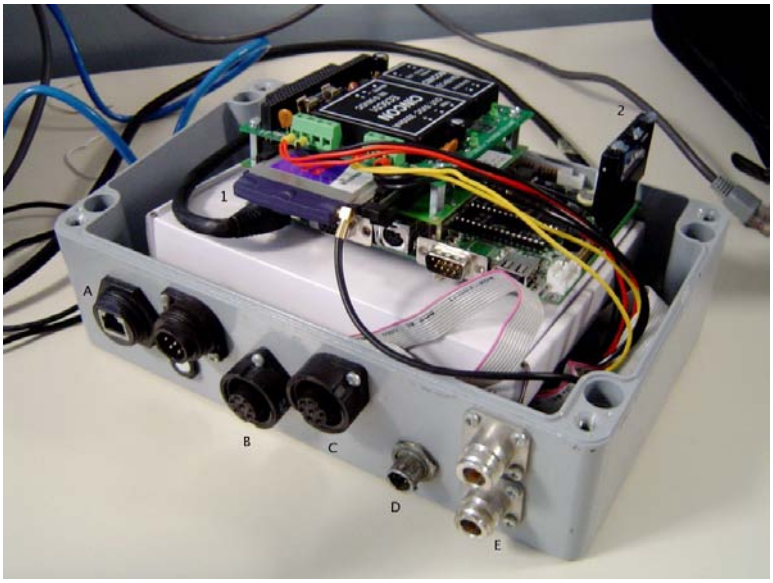

Figure 4.1: Photo of the BlueBox used during tests. The internal hardware was raised to ease the viewing.
Hardware's caption:
1: PCMCIA wireless card
2: Solid state memory (Disk on Module)
Connectors' caption:
A: 100Mbit Ethernet
B, C: Serial ports
D: 24V power
E: Connections for external antennas

4.1.1 Hardware

The BlueBox is based on a 230Mhz Geode CPU, compatible with the Intel IA-32 architecture, that integrates also all the functionalities from audio and video hardware. The processor is mounted on a Boser motherboard that uses a common Realtek chip for networking and four serial ports. Also available are a floppy connector, a parallel port and two USB 1.1 ports. The hard disk is implemented with a DOM (Disk On Module) with a mini-IDE interface IDE (44 pin connector), the same used on laptop's hard drives. Normal, spinning disk, hard drives are not used because of the low resistance to vibrations. On the motherboard it is possible to mount on a tower configuration some expansion cards with a PC/104 bus, very similar to ISA. In the used configuration a 24 Volt power card and a PCMCIA[2] expansion were installed.

[2]Personal Computer Memory Card International Association

Integration of the Senao card

Traditionally on BlueBoxes Cisco WiFi cards are used, that offer good performances, but lack the support to became AP (Access Point). It was the necessary to look for a PCMCIA card compatible with the WiFi standard 802.11b, able to work in AP mode and with a good Linux drivers. Following a search on the various open source projects, the conclusion was reached that the only driver able to work in that mode was HostAP[22], that unfortunately supports only cards based on the Prism chipset. More researches between hardware producers led to the choice of a card produced by Senao[3], that covers all the technical requisites established for the integration in Fantuzzi Reggiane's systems.

4.1.2 Software

The operating system is GNU/Linux with kernel version 2.4 and Busybox version 1.0 for everything related to shell commands and kernel module management.

Additional programs

All the software added the the base configuration was recompiled with the versions of C libraries and gcc compiler[4] available on the BlueBox to keep the binary compatibility with the existing system. To do that a separate compiler environment had to be created, as the versions distributed now with the same libraries are different. When possible, as a matter of fact, it is inadvisable to compile software directly on the BlueBox for reasons ranging from the slow processor, to the excessive wear of the EPROM flash that is used in the DOM. Moreover the size of the software packages needed for compilation would exceed the disk size.

With the described procedure was completed the installation of `olsrd`, the process that realizes the OLSR routing protocol (see the 3 chapter) and the HostAP driver needed for the WiFi card, with all associated programs. Moreover all scripts needed to complete the system for functioning without supervision were written and tested.

4.2 Linksys WRT54G e WRT54GS

The possibility to use a commercially available device was evaluated, as a low cost substitute of one or more BlueBoxes. The cost difference would be around one order of magnitude against the BlueBox, so the hypothesis

[3]The model chosen is the 2511CD PLUS EXT2, that has no integrated antenna, but has two connectors for external ones.

[4]The C compiler from the GNU suite

Figure 4.2: Front visual of the WRT54g wireless router, the lights used to check the router status are visible. The LED marked as DMZ is used to give feedback on the boot-up of the operating system, as it is turned off when the boot sequence is terminated and it is ready to start routing operations.

was evaluated with much attention. Linksys sells two similar routers, the WRT54g and its bigger brother, the WRT54gs. The only difference is in the bigger flash memory on the gs model, so they are equivalent for this thesis use and can be exchanged anytime.

The reliability of the device was the more evident problem. The WRT54g is a consumer product and was not designed to withstand the rigours of an hostile environment. The temperature interval where the operation is guaranteed according to the specifications (from 0 to 40 Celsius degrees) is very narrow, but can be expanded adding a conditioning system inside the water-proof container that would be needed anyway to be able to install the WRT54g outdoor.

Another reliability problem, more subtle, was related to the software. The device needs to be able to restart the operations automatically in the events of blackouts and must not have any path that requires manual intervention or a manual reboot.

4.2.1 Hardware

The Linksys WRT54g is a router equipped with four Ethernet ports, in switch configuration, one port for the WAN connection (ADSL or dedicated line) and a wireless card with two antennas in *diversity* mode. These are connected with a screw connector and can be substituted with other with

different characteristics.

The processor is based on the MIPS architecture and is directly connected to the wireless card 802.11g[5], to the volatile memory and to the flash memory. This last is divided in three areas, where only the middle one is easily accessible and writeable:

1. Boot loader: run at system boot reads several parameters from NVRAM, setting up the hardware and trying to receive a TFTP connection for some seconds on the Ethernet interface to download a new operating system package. Then jumps to the code in the first block of the middle area of the flash.

2. System: uses the biggest part of the flash memory and contains all the operating system and user programs. Depending on the configuration it can be divided in two partitions or only one.

3. NVRAM: uses a few kB and keeps a number of different parameters in the form of name=value. Has the objective of keeping configuration data between reboots and event between operating system upgrades. It is possible to record text or binary data.

4.2.2 Software

The Linux based operating system provided by the factory in the WRT54g was overwritten with the OpenWRT[6] distribution, created specifically for several compatible models from Linksys. This distribution provides all the base packets and allows others to be downloaded from the Internet with a simplified procedure using the ipkg packet system, common in handheld distributions. Moreover the OpenWRT projects provides the toolchain needed to compile software for the MIPS architecture.

OpenWRT is one of the two systems run on the WRT54g and is the one that passed the tests described in the 6 chapter. The other, called FreiFunk, from the name of a regional wireless network project aimed to German-speaking areas, has show stability problems in the wireless drivers, but has an excellent web interface for configuration, that was modified as described in section 5.4.
Other projects exist, some not free, for alternative firmwares for the WRT54g, but were not taken in consideration because they do not offer the source code of some packages and do not offer a flexible installation system as the one on OpenWRT.

[5] Also able to work with the protocol 802.11b, requisite condition for the use with other devices already deployed.

[6] http://openwrt.org

4.3 Antennas

The antennas choice was extended and quite difficult because many parameters has to be taken in account, other than the effective quality of the equipment, already difficult to measure by itself. In particular the length of the cables, the connector type, the commercial availability, price and mounting procedure, had all to be taken in account. Antennas differ also for the irradiation pattern. Omnidirectional, sector or point to point (parabolic) types exist. As a matter of fact diffusion patterns are shown on the horizontal and vertical planes to explain the antenna gain in the different directions on the product datasheet.

4.3.1 Cisco AIR-ANT2506

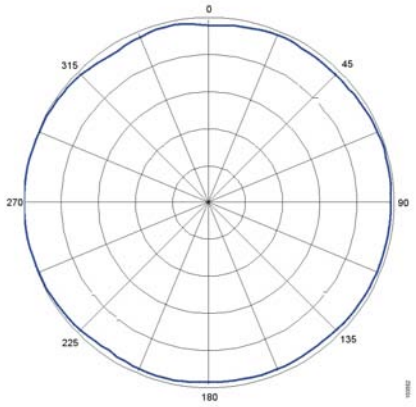

Figure 4.3: Horizontal plane diagram for the Cisco antenna.

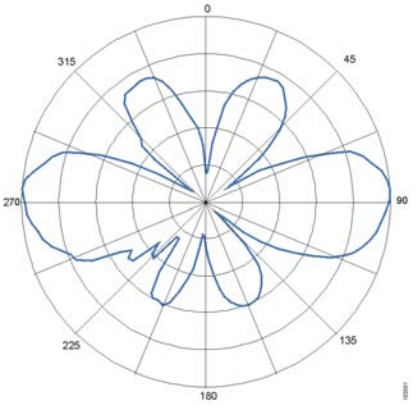

Figure 4.4: Vertical plane diagram for the Cisco antenna.

This Cisco antenna is provided with a one meter cable, an RP-TNC connector (quite difficult to find) and a metal holder for pole mounting. It is omnidirectional on the vertical plane and generates high and low lobes on the vertical one, able to cover areas directly above and below the mounting point.

4.3.2 Huber+Suhner SOA 2400/360/4/20/V

This antenna needs to be ceiling mounted and is provided with several screws and tassels for this purpose. It is thought to irradiate downwards in a wide area, as a train station or a waiting area at airports. It has an N connector without cables. As can be seen from the irradiation diagram, it is a valid alternative to the Cisco antenna for mounting on top of light poles, guaranteeing a good coverage both on the horizontal plane and downwards.

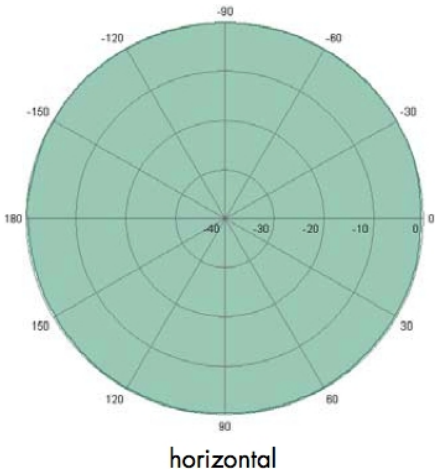

Figure 4.5: Huber+Suhner horizontal plane diagram.

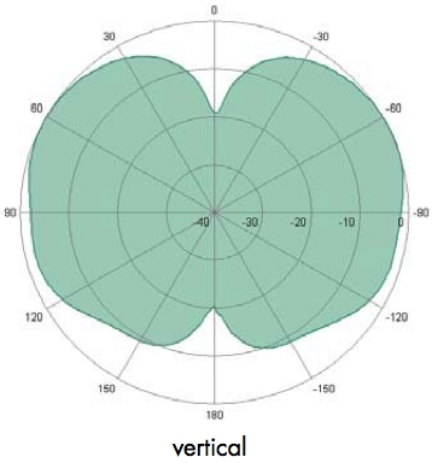

Figure 4.6: Huber+Suhner vertical plane diagram.

4.4 Choices and motivations

The application at Nola's interport will need the use of three of four fixed nodes, applied on top of light poles with an height of about 20 meters and of two mobile nodes, mounted on *stackers*[7].

It is planned to use seven nodes, all with a single antenna. For simplicity it has been decided to use the same antenna and the same hardware for all nodes. With the small difference in performance between the two antennas, as evidenced by the tests described in the 6 chapter, it was decided to use the Cisco antenna as more suitable for mounting on stackers. For the hardware, after the endurance tests, it was decided to used the Linksys router that, even with the necessary heater, thermostat and water-proof enclosure, has a much lower cost.

[7]Mobile machinery with an extensible jib able to grab a container from a rail wagon or truck and position it on pile high at most 12 meters.

Chapter 5

Software

The software development followed various paths, touching all the levels, from the operating system to user applications. The first part of the implementation concerned the new Linux distribution that was installed on Linksys routers. These are, for all purposes, embedded devices that need a particular kernel and applications specially recompiled. Moreover the final application required a personalization of some software elements to adapt them to the requirements of Fantuzzi Reggiane. Inside the company it had never been developed an embedded system with a non-Intel architecture and so little memory before.

For the operating system and the kernel the work was based on the OpenWRT Linux embedded distribution. The set of installed packages and kernel configuration was modified. Moreover, since not all the needed software was available in the OpenWRT archives, some new packages were created, modifying OpenWRT's build system to include these applications. This way they would became part of the distribution and ease every future update operation that should become necessary.

The user level application software that was developed allows the technicians that manage the network to observe at every instant the status of each MeshAP node, both fixed and mobile, thank to the monitoring program. Should problems arise a diagnostic program was developed to be executed from a laptop directly connected to the malfunctioning node with a pre-installed Ethernet cable.

If major changes in the whole wireless network should become necessary, all MeshAP nodes publish a web interface that allows the setup of wireless parameters, `olsrd`, IP addresses and firmware update, all protected by a password authentication system.

5.1 OpenWRT

OpenWRT is the Linux distribution used as base for the modified operating system installed on WRT54G routers. OpenWRT provides the kernel and a set of base applications, with the ipkg package manager among these.

Adding `netperf` to OpenWRT was needed to perform some tests. It is a software for measuring the bandwidth of the network. This program has several problems when being compiled for the mips architecture and required several modifications to its source code. Also the recompilation of the latest version of `olsrd` was done, since it was not available in OpenWRT. Other packages were also created, one for the monitoring and diagnostic process and one for the programs and scripts used in testing (cap. 6).
To add these applications in the OpenWRT build system, several *makefile* had to be modified, the instructions for ipkg package creation had to be written and finally a description file with the package contents, author, licence, web site, etc. needed to be created.

The porting and development of the software was not always straightforward because the mips architecture available on the WRT54g devices has a different byte ordering[1]. Also OpenWRT uses ucLib as system C library, that provides the majority, but not all, of the system calls available on the more widely used (but bigger in size) GNU libc. These differences can lead to compilation problems on software developed on Intel without precautions for porting on other environments.

OpenWRT provides a series of instruments to speed up the preparation of a new system image that can be written over the standard one by Linksys.

5.2 Monitoring software

The software has a client/server structure, with the communication being made with the UDP protocol on port 9000. Figure 5.1 helps to comprehend the system architecture: the server is executed on every node, both fixed or mobile (M or R) and collects all the informations to be sent to the client, active on the system administration computer (D or L). The client present a graphical interface able to visualize all operating data a node at a time, showing the following informations:

- Reachability
- Number of packets and bytes sent and received
- Number of transmission and receive errors
- Signal and noise levels in dBm from the radio hardware

[1] mips is big endian, while Intel is little endian

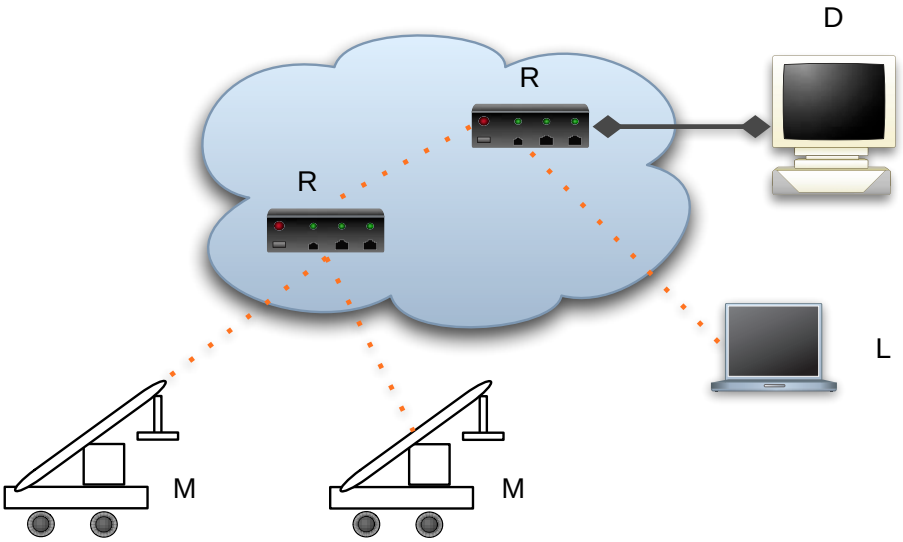

Figure 5.1: MeshAP based system structure. Mobile nodes M and L communicate via WiFi to a number of fixed nodes R that implement dynamic routing. Finally one of these nodes is connected with a cable to the harbour network infrastructure and with the system administration computer (D).

- Uptime since last reboot
- Free memory
- Number of processes in execution
- Access to the network graph drawn by a `olsrd` plugin and to the web interface of each node

The list of nodes is available on the left part of the interface (see figure 5.2) and shows immediately the state of all the network. For each node a colour is used:

- green: reachable node
- red: unreachable node

Selecting a node, on the right, all available informations are shown, updated at the time the last packet was received. The client process discards only out of order packets, but does not ask for the retransmission of lost ones.

In the lower left the buttons to modify the nodes list are available, the add button will show a window where all data for the new node should be entered. Then all data entered is checked for validity before resuming

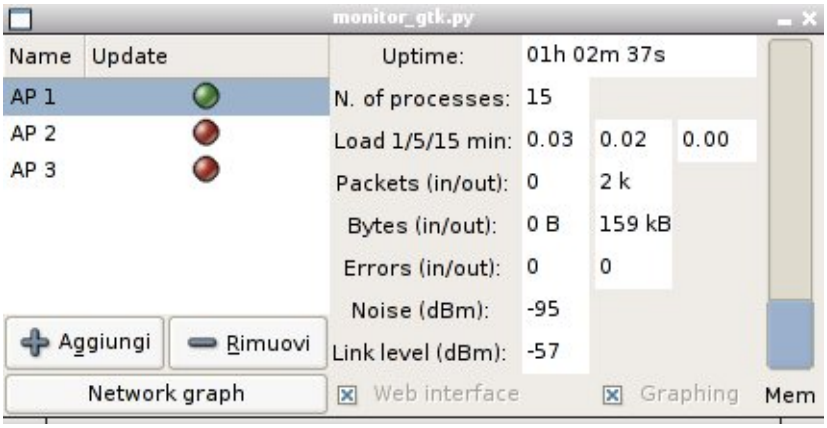

Figure 5.2: Main window of the monitoring program. The image was taken on Linux, but the application runs on Windows and Mac OS X without changes. The node AP 1 is reachable, the other two are not.

operations. The remove button will ask for confirmation to avoid mistakes. In both cases modifications are then saved to disk immediately.

The client program provides also the possibility to draw a graph of the dynamic routing network if there is at least one node with the needed `olsrd` plugin. The system works by reading the data provided by `olsrd` in the DOT format and converting it to a PNG image shown to the user. It is possible to save the image to disk for future reference.

The server process is written in C, keeping in mind the limited resources of the hardware where it is run. At start-up the execution stops, after a short initialization, waiting for command from the client. After that it starts to send data read by various system calls at regular intervals. The length of this interval is specified by the client and is measured in seconds.

5.3 Diagnostic software

This software has also a client/server structure and shares the server process with the monitoring applications. This choice was done to save on the resource usage and to avoid code duplication.

The program is thought to be run from a Windows laptop connected directly to the malfunctioning node with an Ethernet cable left in an easy reachable position after the initial installation. The operations that can be done are:

- Reboot of the network interface
- Software reboot of the node
- Access to the web interface

Figure 5.3: Main window for the diagnostic applications.

- Access to the terminal via an ssh remote shell

The first two operations are to be done only if the node is not reachable from the monitoring application. During the test phase it has not been possible to recreate a complete block of the system, but it could happen in extreme temperature conditions. In particular the Ethernet interface was always completely dependable, allowing cable access even if the WiFi hardware, much more delicate and prone to problems, was unreachable.

The web interface provides operating data much more detailed of those available with the monitoring software, it allows also to access an administrative section where it is possible to modify some operating parameters.

Terminal access provides a shell where most of the UNIX commands are available. The access is possible with a user without root privileges and the `su` command needs to be used before running privileged commands.

5.4 Web interface

To simplify the most frequent tasks of nodes' administration a web interface has been developed. It provides two distinct areas: the first one, with unrestricted access, is able to show detailed data on the operating status of the node. The visualized data are:

- Generic
  - IP addresses
  - interface status
  - identification codes
  - boot log (dmesg)
  - system log (syslog)
- WiFi scan for other network available in the same area

- Routing table
- `Olsrd` internal status

In the second area, for administrators, the following entries are available:

- Change password (root user and web interface)
- `Olsrd` configuration
- WiFi settings (802.11 parameters and IP address)
- MAC address filtering
- IP LAN addresses
- Firmware update
- Reboot

The access to this area is controlled by a password that is kept synchronized with the one of the root user. This is possible because communications between nodes will be always encrypted with WEP. Otherwise the password would travel as plain text every time an administrative page is loaded. A note informs the user of this danger.

This web configuration system, based on the ideas of the FreiFunk interface, but written for the most part from the ground up, works thanks to lighttpd, an HTTP server a few tens of kilobytes in size, and a series of CGI scripts in shell POSIX language[2].

While most of the pages are based around a set of system calls to modify NVRAM parameters or configuration files, the firmware update interface has been written in a particular way: running from a CGI script, it is started from the HTTP server process, that reads the new system image sent by a POST request and starts to write it to the flash after the transfer is completed and some integrity checks are completed. The problem lies in the fact that the writing process must not be interrupted to avoid that the router, as it said, becomes a brick useful only as a door stop. To this requirement an additional request for some feedback was done.
Once the write operation is completed, three reboot of the system are needed: the first one, started from the CGI script, is needed to initialize the JFFS2 filesystem, the second, started by a special script in the boot sequence, is needed to make it writeable. The third one is needed to delete the special script of the second reboot, as the filesystem is read only at the time. Luckily all the operation is quite speedy and does not require more than a couple of minutes.

[2]only ash is available on OpenWRT, a minimal shell that uses a subset of the bash language.

The administration interface was written keeping in mind some needs of expandability, each page constitutes an independent module built with a common infrastructure and a specific part. This allows an easy adding or removal of modules without causing involuntary damage to other parts of the interface.
Thanks to this planning and the language used it is possible to use the same interface on a wide range of situations and architectures with a very limited number of changes.

5.5 Olsrd configuration

In this section the configuration file of `olsrd` is described, highlighting, when needed, the differences between the configuration of a border node from an internal one in the dynamic routing network.

5.5.1 HNA (Host Network Advertise)

```
Hna4
{
# Published subnet:
192.168.1.10 255.255.255.255
}
```

HNA is a special type of message that `olsrd` uses to publish to all other nodes the availability of a gateway toward another IP network. This configuration is very important for border nodes, as they are connected by the Ethernet interface to one or more hosts that are not running OLSR. This used with proxy ARP, allows the creation of a transparent network for users, that need not to install additional software or change network configuration to access all the network.

5.5.2 Hysteresis

```
UseHysteresis yes

HystScaling 0.30
HystThrHigh 0.70
HystThrLow 0.15
```

Hysteresis and the mathematical formula used by `olsrd` to calculate node reachability are described in the 3.2.1 section. The values above were chosen empirically, based on field tests (see 6.4.3), with the purpose of keep stable the path choice, that otherwise is too quick adapting to topology changes and less stable in slow changing configurations.

5.5.3 Interfaces and validity times

```
Interface "vlan0" "eth1"
{
    # Hello interval in seconds
    HelloInterval    2.0

    # HELLO validity time
    HelloValidityTime 10.0
}
```

`olsrd` needs to know which interfaces are available to send or receive its packets. For internal nodes, it is not needed to list the vlan0 interface (cabled network), but there are no problems in leaving it there. On the contrary it allows an operator with an `olsrd` enabled laptop to access the whole network and solve various problems.

HELLO messages are used to build the list of first neighbours, first step in building the network topology. Having lengthened the time when those messages are considered valid, the routing rate of change was slowed even more, giving a more stable network.

5.5.4 Plugin

```
LoadPlugin "olsrd_dot_draw.so.0.3"
{
        PlParam "port" "2004"
        PlParam "accept" "192.168.1.10"
}

LoadPlugin "olsrd_httpinfo.so.0.1"
{
        PlParam "port" "8080"
        PlParam "Net" "192.168.1.0 255.255.255.0"
}
```

The LoadPlugin sections is used by `olsrd` to know which plugins it needs to load and their configuration. This is used only on border nodes, so that it can pass all the data needed to the monitoring program to draw the network graph and to provide an HTTP access to the OLSR internal status.

All section described above were highlighted in a particular way, so as to ease their modification from the web interface.

5.6 Proxy ARP

Commands listed below activate the proxy ARP feature between wireless and Ethernet interfaces.

```
echo 1 > /proc/sys/net/ipv4/conf/vlan0/proxy_arp
echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy_arp
```

The proxy ARP solves in a very elegant and inexpensive way a big problem that presented itself when a node without dynamic routing capability wants to access the network with the minimum possible impact on its internal routing configuration. The situation arises when a MeshAP is used as bridge between a traditional network and one based on dynamic routing.

These border nodes are connected to another network that can be e proper LAN, or only one host, as is the case of Displays on stackers, where a MeshAP node will be connected to Displays with a simple Ethernet cable.

Setting the MeshAP as IP router is not enough, as the non-OLSR nodes would need anyway to setup a specific routing path, just what was not desiderable in the first place. ARP requests are not passed between interfaces inside the MeshAP node, since it is only doing IP level routing. Proxy

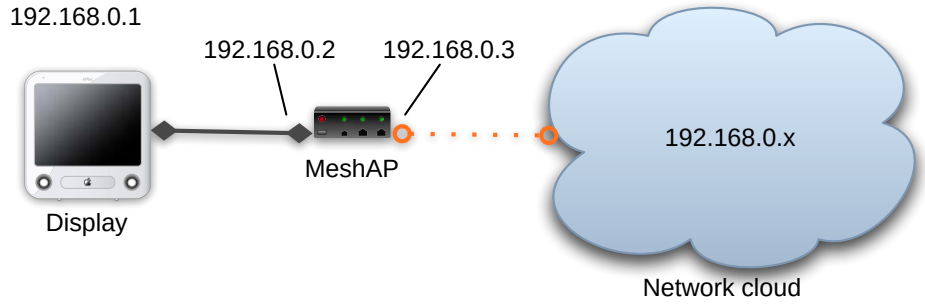

Figure 5.4: Schematic of an example network with proxy ARP

ARP works by replying with its own MAC address to all ARP requests addressed to IP addresses for which there is a path in its routing table. In table 5.1 the steps followed by two nodes, one with normal configuration and one with proxy ARP are described. The network used is the one shown in figure 5.4.

Proxy ARP works by tricking the host which is trying to communicate by poisoning its ARP cache with a false couple (IP, MAC), but one able to deliver the packets through the MeshAP.

The union between proxy ARP in one direction and HNA messages in the other allows the use of a completely transparent wireless network with dynamic routing to all connected nodes. As a matter of fact these need not to have special configurations, having at the IP layer the complete visibility

Table 5.1: Exchange of message with and without proxy ARP. In the left column the operations done by the Display (the 'ignorant' node) and on the right those of the MeshAP.

Without proxy ARP	
Packet to be sent to 192.168.0.10	
Sending ARP request for 192.168.0.10	Request discarded because the address is unknown
No reply, packet transmission failed	
With proxy ARP	
Packet to be sent to 192.168.0.10	
Sending ARP request for 192.168.0.10	Request received, the address is reachable according to the routing table by another interface.
Waiting	Sending an ARP reply with own MAC address
Sending data packet to the MeshAP, but with the correct IP destination	Routing of the packet toward the wireless interface.

of the network, as it would be with a traditional cable network. All the complexity of mobile nodes is managed transparently by the MeshAP and the OLSR protocol.

5.7 Future developments

Both monitoring and diagnostic programs and the web interface can be easily reused for other projects, since they were developed with modularity and portability concepts integrated in their design.

The system available on the MeshAP is able to offer WiFi coverage to vast areas, but with the limitations, due to its size, weight and power requirements, that it is mounted only in fixed positions or motorized vehicles. The possibility to develop a MeshAP on different hardware, able to work with batteries and to provide access from a laptop or an handheld is very interesting and should be investigated.
It is also possible to use the MeshAP hardware directly for other tasks, other than the routing of packets, that is not keeping the central processor busy. It is possible to build a number of applications, from data collection and visualization, to voice communications with *Voice over IP*.

Chapter 6

Field testing

In this chapter the tests done to verify the performance of hardware and software before the deployment at Nola are described. The tests were done with the objective to evaluate the OLSR implementation and the various hardware devices (cards, antennas) and were made indoor with the help of firewalling techniques to simulate node visibility and outdoor, trying to reconstruct the environment available in Nola as distances, heights and speed of the network nodes.

6.1 OLSR routing latency

This test series had the objective to verify how much the routing would weight on network latency. Since in Nola the expectation was to have paths at most of three/four jumps, it was possible to build a simulation with four devices: two laptops, one WRT54g and a BlueBox. All devices were connected with Ethernet through the switch integrated in the Linksys router. The cabled connection was used to communicate commands and receive data, without polluting the wireless network with traffic unrelated to the test.

This test gives a measure of the efficiency of the Linux kernel routing, more than of the latency introduced by `olsrd`. Once `olsrd` establishes a path toward a certain destination, it adds an entry in the kernel routing table that will work from the instant onwards just as a manually entered one. Moreover the test was static and was started only after `olsrd` reached a stable configuration, visible below:

```
--- 08:42:41.12 --------------------------------- LINKS

IP address       hyst   LQ     lost   total  NLQ    ETX
192.168.128.1    1.000  0.000  0      0      0.000  0.00
192.168.128.2    1.000  0.000  0      0      0.000  0.00
192.168.128.11   1.000  0.000  0      0      0.000  0.00
```

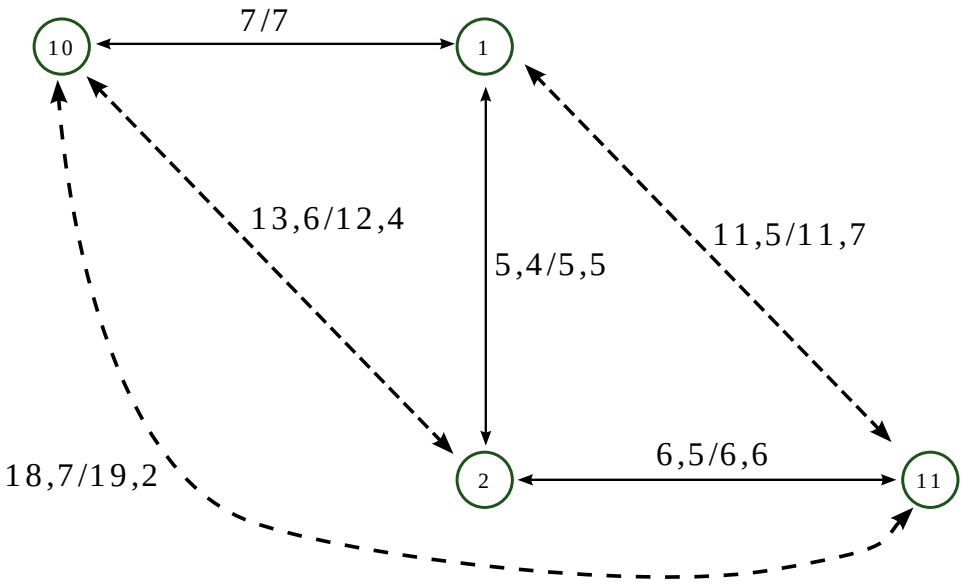

Figure 6.1: Graph of the network used for latency tests, data was calculated from the mean of 40 measurements done with ICMP echo request/reply packets.

```
--- 08:42:41.12 ------------------------------ NEIGHBORS

IP address       LQ     NLQ    SYM   MPR   MPRS  will
192.168.128.1    0.000  0.000  YES   NO    NO    3
192.168.128.2    0.000  0.000  YES   NO    YES   3
192.168.128.11   0.000  0.000  YES   NO    NO    3

--- 08:42:41.12 ------------------------------- TOPOLOGY

Source IP addr   Dest IP addr     LQ     ILQ    ETX
192.168.128.1    192.168.128.10   0.000  0.000  0.00
192.168.128.1    192.168.128.2    0.000  0.000  0.00
192.168.128.2    192.168.128.1    0.000  0.000  0.00
192.168.128.2    192.168.128.11   0.000  0.000  0.00
192.168.128.10   192.168.128.1    0.000  0.000  0.00
192.168.128.11   192.168.128.2    0.000  0.000  0.00
```

The sections named LINKS e NEIGHBORS show informations on neighbours of the node that gave these informations. The TOPOLOGY section shows the network topology, listing the couples of sender/receiver IP addresses. A graphic representation of the network is available in figure 6.1,

where the latency time for 64 bytes packets are also shown.
It can be seen that the time needed to go from the first node to the last is the sum of the times for each jump. The small difference between the two times can be explained with the additional time each node uses for routing decisions, that is anyway at most a millisecond.

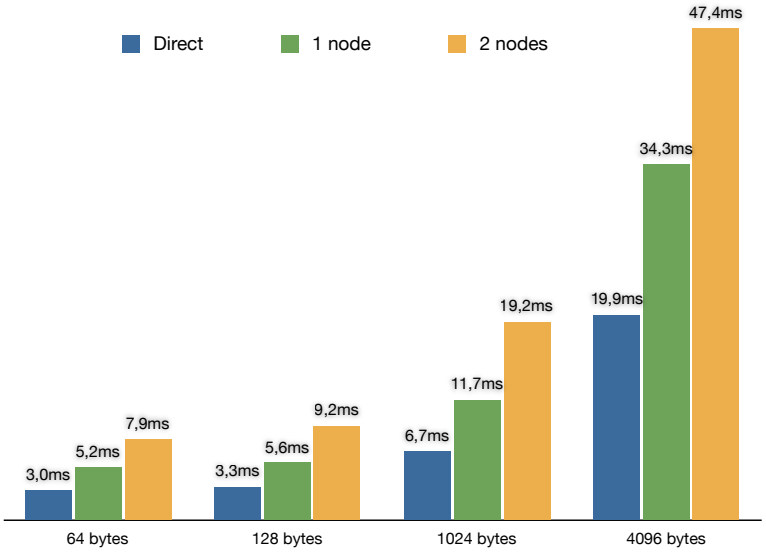

Figure 6.2: Latency of the OLSR network, varying the intermediate nodes number and the size of the packets used for measurement. Data are from the mean of 40 measurements made with ICMP *echo request/reply* packets.

With the same data it was possible to draw the histogram visible in figure 6.2, where it is evident the increase of latency in linear progression with the increase of number of nodes that a packets needs to traverse before reaching its destination. Every jump adds a constant delay due to the time needed to:

- read the frame from the network interface
- wait for the arrival of all other frames if the packets had been fragmented
- choose a routing path according to the routing tables[1]
- send the packet to the network interface, possibly fragmented

[1]This step takes O(log(n)) with n the number of entries in the routing table in Linux

Measures where also done with packets bigger than the maximum size accepted by the 802.11 protocol of 1500 bytes. In this case the IP layer had to manage several frames of smaller size, reassembling and dividing the packets at the passage in each node. As it was expected this did not introduce noteworthy delays in routing.

6.2 Reboot performance

These tests were deemed necessary after several episodes when, with the Freifunk firmware, the WRT54g did not load the wireless configuration during start-up. Since this behaviour was very dangerous for a device that would need to operate continuously, without supervision, it was decided to use a more recent firmware revision that, while experimental, would guarantee better stability.

For the test a timer was used that cycled the power supply on and off every five minutes for a period of four days. To verify the functionality of the network interfaces a laptop with a WiFi card was used, where a script was developed to log reachability informations on both wireless and wired network interfaces.
The new firmware, OpenWRT, had operated with a 100% success percentage in this test, but only after a command was added to the start-up sequence to force the WiFi chipset in 802.11b mode, that otherwise would remain stuck on 802.11g and would not communicate with the rest of the network.

6.3 Climatic chamber tests

Another test needed to verify the possibility to use the WRT54g outdoors, was a review of its operation in extreme conditions, both for measure its real working temperature range, and to decide on the need for a complete conditioning solution inside the MeshAP box. Since the WRT54g already has overeating problems when under heavy computational load, a fan was mounted that would create a flow of air to the principal components of the internal hardware, even before starting temperature tests. The climatic chamber used has the size of a big refrigerator, but it is able to generate in its interior temperatures ranging from -20° and +90° and relative humidity up to 100%, allowing the test of hardware in very different conditions, in a very controlled manner.

During the whole test, of about 4 hours, the WRT54g with its power supply was always connected by Ethernet cable and by Wifi, with a working mode similar to the one described in section 6.2. Using ping, set up to ask for a reply once every second, and telnet, to manually verify the general operating system responsiveness, it was possible to monitor the system state during all the test duration.

The graph in figure 6.3 shows the ping times (latency) varying the humidity and temperature.

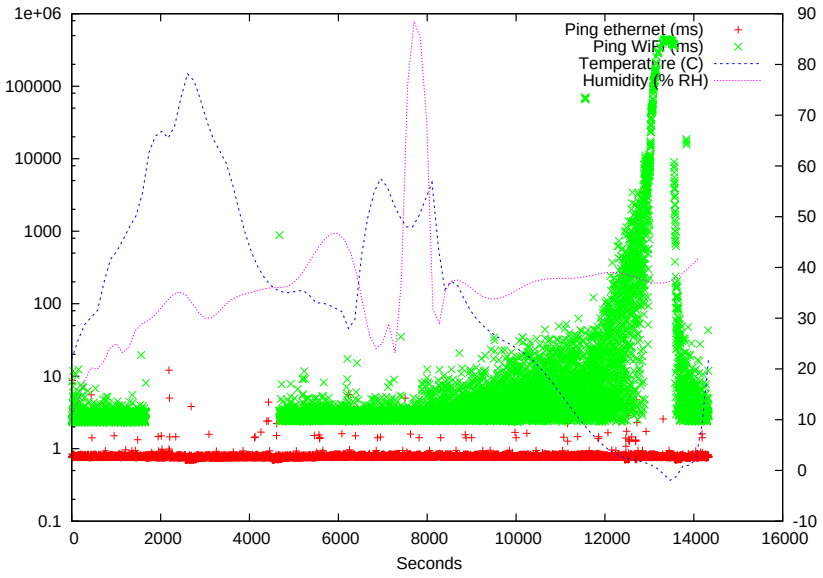

Figure 6.3: Graph that describes the latency variation during the climatic test measured with ping. On the y axis on the left, times are shown, while on the right axis temperature and humidity are shown.

6.3.1 High temperature

From 0 to 6000 seconds the high temperature test was performed, starting from root temperature (about 23° centigrades), it arrived to about 80°. While the Ethernet connection did not show any problem, without even producing variations outside the mean of ping times, the wireless hardware stopped working around 65-70 degrees, without restarting until the command `iwconfig` was executed to ask for informations about the wireless driver status.

The producer specifications did show a maximum working temperature of about 50°, so it was already planned the use of an enclosure with fans, but the test confirmed this necessity, since if the closed metal box is mounted outside, under the sun, in particular near Naples, it can easily go higher than 70°, without any kind of ventilation.

6.3.2 High humidity

Once the temperature was lowered, the humidity test was started, that run until 8500 seconds. Relative humidity was made to go up to 90%, when, since the WRT54g did not present problems, the test was interrupted abruptly by opening the chamber door, to avoid the formation of condensation that could have damaged permanently the system.

6.3.3 Low temperature

The last part of the test was needed to verify the performance at low temperatures, showing that the lower limit of 0°, given by the hardware producer, was very precise. As can be seen from the graph, when the critical temperature neared, the router started to lose packets and to require a longer time to reply to test packets, arriving to a latency of several minutes just before shutting down completely.

6.3.4 Conclusions

This test demonstrated the need to enclose the router in a box with thermostatic control and heating hardware, since it is possible that in winter the temperature reaches 0 degrees. The need for an external air fan, however, needs to be tested again by leaving a finished prototype under a the sun for a day and verifying its performance at regular intervals.

6.4 Two different antennas performance and VNC use

The objective of this test was to measure the ability of the two antennas more suitable for installation in the fixed nodes, positioned on top of the light poles. To better simulate the final positioning, the antennas were mounted on top of a building, at an height of about 20 meters. A first node was at the third floor of the same building, connected with an Ethernet cable to the second one, on the terrace. From there the antennas provided coverage to the third node, positioned on a car moving along the path shown in figure 6.4. The test was performed two times, once with an antenna and then with the other. For this reason the two graphs 6.5 and 6.6 are not synchronized between the two antennas, as the short path gave higher weight to the local traffic conditions.

On numbered points the measurements were done to record the signal quality and to try a VNC connection with a computer at the office. This to try the usability of a remote desktop facility over a MESH network.

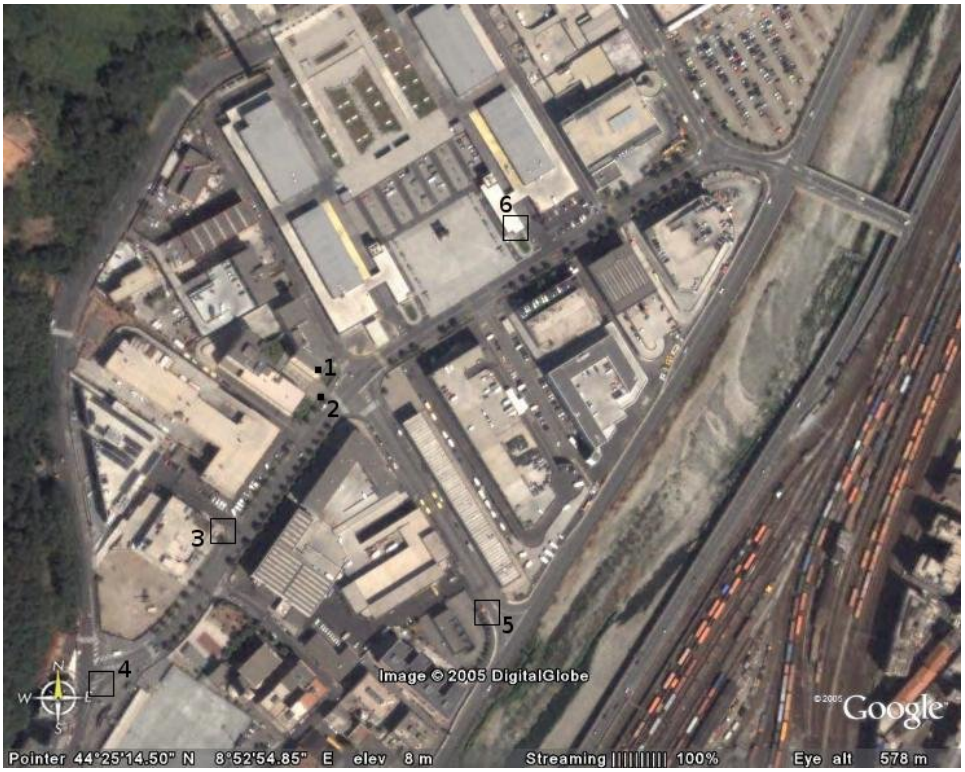

Figure 6.4: Visualization of the path followed to for the measurements of the signal quality. The numbered points corresponds to the positions appointed on the graphs 6.5 and 6.6. In point number 1 are the fixed antennas. The distance between points 1,4 1,5 and 1,6 is of about 200 meters.

6.4.1 Conclusions - antennas

The comparison between the two antennas showed that they have a similar behaviour, so the decision could be shifted to other parameters, such as cost and commercial availability. The measured differences lie in the different propagation patterns, that are visible in figures 4.3 and 4.6.
The Cisco antenna offers a stronger signal for a receiver at the same height, and this is an advantage, since all fixed nodes will be positioned at the same height, while the two mobile nodes will have the coverage of two antennas for most of the time.

The good signal to noise ratio available at a distance of 200 meters allows to calculate a wider maximum communication range, superior to the planned distance between fixed nodes.

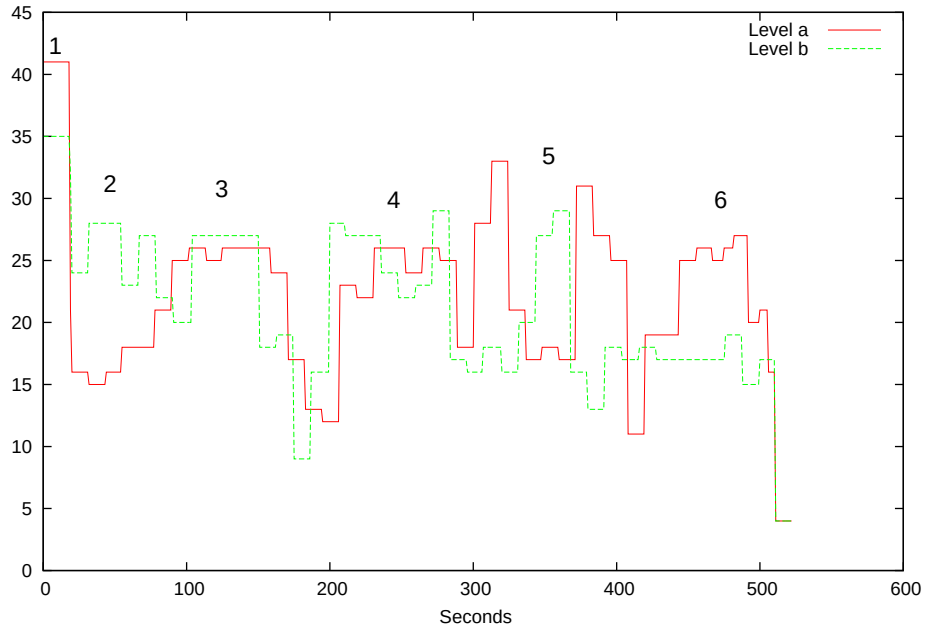

Figure 6.5: Signal strength for the two tested antennas. The 'a' antenna is Cisco (sect. 4.3.1), and the 'b' antenna is Huber+Sunher (sect. 4.3.2).

6.4.2 Conclusions - VNC

The use of VNC was deemed good enough at a resolution and bit depth higher than those required for the Nola application. A problem that was seen repeatedly was the speed with which the VNC server closes the connection when a slight packet loss happens. An immediate reconnection was always sufficient to re-establish the working conditions.
This problem can be solved by setting a higher time out on the server or by configuring the client to reconnect at the first signs of difficulty.

Another test, very similar to this one, will be needed to verify the performance of VNC with two intermediate nodes. This is very important since static experiments showed that the bandwidth is lower with each additional intermediate node that needs to be traversed.

6.4.3 Final simulation with three 3 hops and four nodes

This simulation was needed to verify if the stability of the `olsrd` dynamic routing is enough to allow a continuous use of the network and if the available bandwidth is wide enough to keep up with two simultaneous VNC connections.

The test helped in determining new values for the hysteresis and the

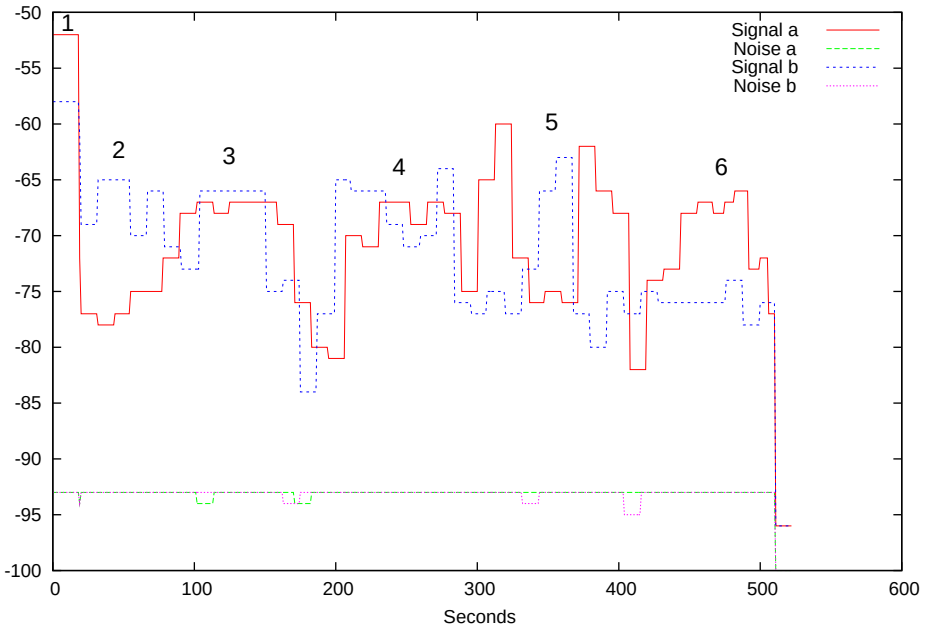

Figure 6.6: Graph of the signal and of the noise received at the mobile antenna, antennas 'a' and 'b' are the same of figure 6.5.

HELLOValidityTime (see sections 3.2.1 and 5.5), more correct for a slow changing network topology (on 7 nodes, 5 will be fixed and only 2 will move, at most at 30Km/h.) as the one that is planned for Nola.

The test showed also how much the WiFi signal is sensitive to environmental factors, such as reflections over metal surfaces. In particular during the test a truck that was manoeuvring nearby caused enough reflections to cause `olsrd` to think that a new path was possible, bypassing an intermediate node, causing heavy packet loss.

The disposition of nodes is visible in the 6.7 figure: A is a laptop with Windows 2000 and VNC and FTP servers. In execution there is a beta version of graphical interface that will be used in Nola. B and C are WRT54g routers, with the modified OpenWRT distribution, able to do dynamic routing. Finally D is a second laptop, with Linux, a logging system written for this test, an FTP client and a VNC visualizer.

The FTP protocol is used to measure the bandwidth available between the first and the last node, transferring a file of 2 MB. At the same time the FTP traffic was used to simulate a second VNC connection, to verify the effects on the real VNC connection.

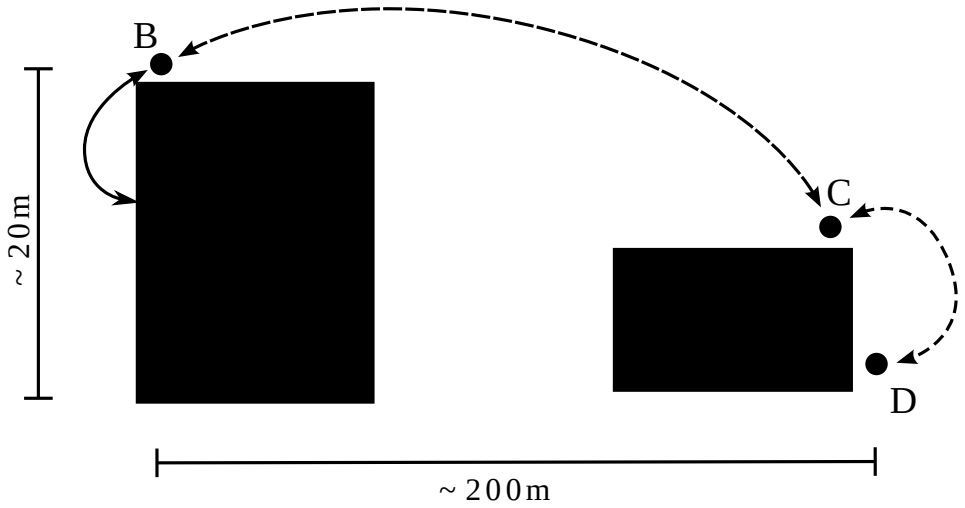

Figure 6.7: Distribution of the nodes during the simulation. The connection between nodes A and B is made with an Ethernet cable, while the other nodes use 802.11 WiFi. All routing is done by olsrd.

Conclusions

This simulation was of crucial importance. It has enabled the understanding of the need to find correct values for `olsrd` parameters, so as to avoid unstable routing that would cause packet losses and changes in the available bandwidth, with disastrous consequences for VNC, that is in many ways, very similar to a video stream. Also, being a simulation on the field, it permitted to test the functioning of all apparatus in realistic conditions.

6.5 Other data got from the tests

While performing the tests described above, a series of results were obtained, that while not relevant for the established objectives, were nonetheless deemed interesting and are described below.

6.5.1 Wireless hardware drivers under Linux

The support for wireless cards under the Linux kernel is still very primitive in respect to the one provided by other operating systems. This is not only caused by the lack of hardware specifications needed to write complete drivers, but also from the lack of a unifying wireless subsystem able to generalize the hardware differences and present an unified interface to the user.

During the different measurements different drivers would use different measuring units to report signal quality and noise levels. In some cases the provided informations could not be converted to decibels, as they use a completely different scale (for example values for 1 to 5). Moreover driver report that kind of data in different ways, some doing a mean on the values read on certain amount of time, causing problems when trying to measure the differences that there are, for example, by keeping the antenna in various positions. For the hostap driver a delay of about 10 seconds between the visualized values and the real value was measured, both for signal quality and noise. The Cisco driver (airo_cs driver), instead, shows the values without trying to stabilize them, causing big variations in the observed values, with high variance. Other problems had to be solved due to the fact that different drivers assign to wireless interfaces different names, causing a constant pain changing IP addresses from wlan0 (hostap) to eth1 (Cisco).

Chapter 7

Deployment to Interporto Campano (Nola)

The system, complete with hardware and software, as described in the preceding chapters, will be used in the day to day operations at Interporto Campano, at Nola, near Naples. The deployment of the system should have been happened during this thesis. The material was ready for installation several weeks before the deadline, but several changes at Nola's site caused delays, causing the shifting of the start up date.

In this chapter, then, it is described how the Nola's plant will be deployed and how it will work once the system will be brought to full working order. Finally some considerations are made on future uses of the same technology.

7.1 Planimetry and topology

Nola's interport is a big structure being completed, with direct connections to the rail network and to the A30 highway. Part of the interport is dedicated to the exchange of containers between trucks and rail convoys. On this area it was established the provision of two stackers and one wireless network to Fantuzzi Reggiane. Another company had to develop the software application that would communicate with the database (based on AS/400) and to present to the operator the list of operations to complete.

The area where the two machines will move is a rectangle of about 1.5 kilometres in length and 500 meters wide. Containers will be stacked at most on two/three layers because of strong winds possible in the region. All the region, as a matter of fact, has the same risk evaluation of Trieste, with the possibility of winds over 100Km/h. At a distance of 100 meters from one another, there are light towers 25 meters in height where 220 Volt power outlets are available.

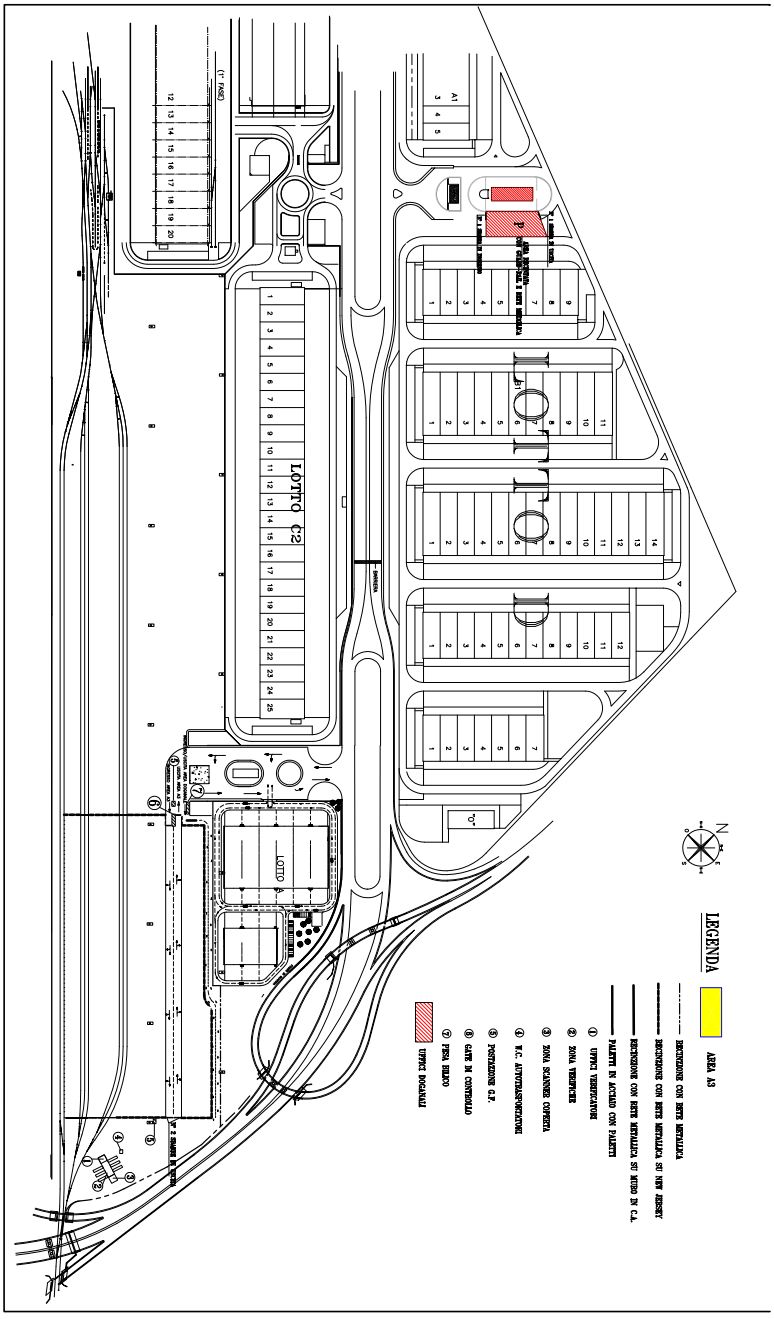

Figure 7.1: Complete planimetry of Nola interport

7.1.1 Fixed nodes

It is expected to install three nodes on top of light towers, at a distance of 400 meters, with a reserve node to cover eventual dark corners. As power supply the one already available on place for the lamps will be used. For each of these nodes two antennas in *diversity* mode will cover the two areas along the longer side of the terminal. Finally an Ethernet cable will be installed to allow updates and diagnostics to be done at ground level.

Figure 7.2: Aerial view of the Nola interport with the disposition on fixed nodes mounted on top of the light towers, corresponding to the red squares.

7.1.2 Mobile nodes

Mobile nodes will be positioned on the two stackers that will operate inside the area. Since on this machinery only 24 Volts power is available a small 24V/12V supply will be needed instead of the 220V/12V available from Linksys. Moreover these nodes will be installed in a covered position, probably under the lifting arm, so as to protect them from direct solar exposure. On these nodes only one antenna will be mounted, on top of a pole installed for this reason at the stacker's tail, at an height of 3-4 meters from the ground.

These MeshAP will give access to the wireless network in a completely transparent way to the Display mounted in the operator cabin with an Eth-

ernet cable. On the Display several screens will be shown with informations on the machine state, alarms and other events. One of these screens will show the list of operations the operator needs to perform. To simplify at most the interface between the Display and the application on the database side, this screen will work by exporting with VNC the remote desktop of a PC kept at the office. This computer will then keep a resolution of 640x480 at 16 bit of colour and one full screen application.

Figure 7.3: Positioning of the MeshAp on the stacker: the letter A shows the position of the hardware, while B is the position of the antenna on top of a pole.

7.2 Mounting

The final deployment will be done in two steps. The mobile MeshAPs will be installed on the stackers in the Fantuzzi Reggiane plant at Lentigione (RE) before the delivery, to reduce the work that needs to be done in Nola. The fixed nodes will need to be installed in place by the technicians responsible for the light towers.

7.3 Future uses

The MeshAP is a very innovative product in its field and it will be used in many future wireless network installations by Fantuzzi Reggiane. The use of this kind of networking technology is getting more and more attention thanks to the low cost and absence of licensing costs of reserved radio channels.

The next step on the MeshAP development will require to reduce the size and weight to offer it as solution for a wider variety of problems. The objective is to reach a stage where it can be powered by batteries for a reasonable length of time, to substitute voice radios with Voice over IP software.

Bibliography

[1] W. Alliance. Wi-fi protected access: Strong, standards-based, interoperable security for today's wi-fi networks, 2003. URL `http://www.wifialliance.com/OpenSection/pdf/Whitepaper_Wi-Fi_Security4-29-03.pdf`.

[2] Andreas Tønnesen. Implementing and extending the Optimized Link State Routing protocol. Master's thesis, UniK - University Graduate Center, 2004. URL `http://www.olsr.org/docs/report.pdf`.

[3] ANSI. *Information processing systems: local area networks — Part 3. Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.* American National Standards Institute, 1430 Broadway, New York, NY 10018, USA, ANSI/IEEE Std 802.3-1990 edition. International standard ISO/IEC 8802-3, IEEE product number: SH13482 edition, 1992. ISBN 1-55937-049-1.

[4] ANSI. *Information processing systems: local area networks — Part 11. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* American National Standards Institute, 1430 Broadway, New York, NY 10018, USA, ANSI/IEEE Std 802.11-1999 edition. International standard ISO/IEC 8802-11, IEEE edition, 2003. ISBN 0-7381-1658-0.

[5] E. H. Callaway. *Wireless Sensor Networks: Architectures and Protocols.* Auerbach Publications, 2003. ISBN 0-84931-823-8.

[6] Christian Tschudin and Richard Gold. LUNAR - Lightweight Underlay Network Ad-hoc Routing. Technical report, Department of Computer Systems, Upsala University, Apr. 2002. URL `http://www.it.uu.se/research/reports/2003-021/2003-021-nc.pdf`.

[7] Christian Tschudin, Richard Gold, Olof Rensfelt, and Oskar Wibling. In proceedings of next generation teletraffic and wired/wireless advanced networking. In *LUNAR - A Lightweight Underlay Network Ad-hoc Routing Protocol and Implementation*, 2004. URL `http://cn.cs.unibas.ch/pub/doc/2004-new2an-lunar.pdf`.

[8] W. C. Craig. Zigbee: "wireless control that simply works", 2004. URL http://www.zigbee.org.

[9] Daniel Aguayo, John Bicket, Sanjit Biswas, Douglas S. J. De Couto, and Robert Morris. *MIT Roofnet Implementation*. MIT, 2003. URL http://pdos.csail.mit.edu/roofnet/doku.php?id=design.

[10] David B. Johnson, David A. Maltz, and Josh Broch. *Ad Hoc Networking*. Addison-Wesley, 2001.

[11] G. Dodero, V. Gianuzzi, and M. Ancona. Ramses: a mobile computing system for field archaeology, 1999. URL http://www.disi.unige.it/person/DoderoG/ramses/papers/pubramses.htm.

[12] Erik Nordström, Björn Wiberg, and Henrik Lundgren. AODV-UU, 2002. URL http://core.it.uu.se/AdHoc/AodvUUImpl.

[13] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. *Lecture Notes in Computer Science*, 2259:1–24, 2001. URL citeseer.ist.psu.edu/fluhrer01weaknesses.html.

[14] *Design considerations for distributed microsensor systems*, 1999. IEEE. URL citeseer.ist.psu.edu/chandrakasan99design.html.

[15] International Organization for Standardization. *ISO/IEC 7498-1:1994: Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*. International Organization for Standardization, Geneva, Switzerland, 1994. URL http://www.iso.org/cate/d20269.html.

[16] A. Mehta. Wireless optical communications, a fast and cost-effective means to bring telecommunications to villages, 2001. URL http://indataportal.com/optical/WOC.htm.

[17] B. Miller. RFC 1097: Telnet subliminal-message option, Apr. 1989. URL http://www.rfc.net/rfc1097.txt. Status: UNKNOWN.

[18] J. Postel. RFC 791: Internet Protocol, Sept. 1981. URL http://www.rfc.net/rfc791.txt. Obsoletes RFC0760. See also STD0005. Status: STANDARD.

[19] J. Postel. RFC 793: Transmission control protocol, Sept. 1981. URL http://www.rfc.net/rfc793.txt. See also STD0007. Status: STANDARD.

[20] J. Postel and J. K. Reynolds. RFC 854: Telnet Protocol specification, May 1983. URL http://www.rfc.net/rfc854.html. Obsoletes RFC0764, NIC18639. See also STD0008. Status: STANDARD.

[21] R. M. Project. Dsr implementation for freebsd, 2000. URL http://www.monarch.cs.cmu.edu/dsr-impl.html.

[22] J. M. SSH Communications Security Corp. HostAP driver and related software, 2001. URL http://hostap.epitest.fi/.

[23] T. Clausen and P. Jacquet. RFC 3626 - Optimized Link State Routing Protocol (OLSR). Technical report, Network Working Group, Project Hypercom INRIA, 2003. URL http://www.rfc.net/rfc3626.html.

[24] L. Viennot. Complexity results on election of multipoint relays in wireless networks. Technical Report RR-3584, INRIA, 1998. URL http://citeseer.ist.psu.edu/viennot98complexity.html.

[25] William H. Mott IV and Robert B. Sheldon. *Laser Satellite Communication, The Third Generation*. Quorum Books, Westport, Conn. 2000, 2000. ISBN 1-56720-329-9. URL http://bex.nsstc.uah.edu/RbS/greenwood.html.

[26] Wireless Communications Technologies Group NIST. Kernel aodv, 2001. URL http://w3.antd.nist.gov/wctg/aodv_kernel/.

[27] Xia Jiang and Tracy Camp. A review of geocasting protocols for a mobile ad hoc network, 2002.