CHAPTER 1

INTRODUCTION

1.1 NEED FOR THE SYSTEM

Today local tracking is provided by techology known as GPS. This is done with the help of many sattelites revolving around the earth and some base stations on earth. This is currently controlled mainly by the United States of America. Increasing dependence on the Global Positioning System has always resulted in lesser accuracy and more cost. The domination of certain country in this area has always been a cause of concern. Another alternative is to create a local positioning system. They are much more cost effective and are more accurate and less dependent. With the advancements in WiFi and WiMAX the relevance of the Local Postitioning System increases.

The accuracy of the GPS is also now a matter of concern. While it is possible for us to position units with GPS globally, it lacks accuracy. The estimate is that GPS can have an error of minimum 10m, with an average error upto 100m. This is obviously a big disadvantage. This is, however, not the case with LPS.

In the near future the WiFi and WiMAX are going to spread in the world and it will be a common thing to be found. This will also be much more cheaper considering the cost of installation and maintanence. Thus the service assosiated with it will also be available at a much cheaper cost. The increasing popularity and the extra services that are available with it, naturally will pull more research people into it and a lot of new technologies and assosiated softwares in this area will be soon a reality.

There is now an increase in availability of Hotspots, using which one can log into a public network to access internet. More people are now using this facility. But, the risk and threats assosiated are also high. Presently there is no mechanism to keep track of the people who are

connected to such public access points. This has lead to inscidents like sending threatening e-mails from public access points by unknown persons. Terrorism is increasing day by day and the terrorists are now using more advanced systems.

A tracking system that keep track of the users will be a useful application. When the position of users are obtained it can be used to isolate possible threat-some people. When this is combined with the visual media like survelence cameras, the criminals can be easily identified.

Any form of tracking can be done using wireless media. A possible application is replacing the navigation system. A wireless device interface on a vehicle can be used to locate onself as well as to transfer data from the service providers. So all these points justifies the need for a wireless tracking mechanism.

1.2 FEATURES OF THE SYSTEM

This is a software that tracks the position of the ad-hoc users who have logged into the network. For the ease of administrator, the software warns the administrator about the connection of a new MAC. Also the administrator is provided with a graphical view of the nodes that are presently connected to the network.

This project is aimed at creating an application that will record the location of nodes that are present in a Hotspot or any WiFi zone. The server keeps track of the location of each user logged into the network with their MAC. The signal strength is used as the basic measuring factor. The server uses another static computer that has a wireless interface for reference puposes. The program runs in GNU/Linux. The program records the data in a file with the MAC address, location and time. This can be retreived and compared with the log registry to identify location of users.

CHAPTER 2

SYSTEM STUDY

2.1 EXISTING SYSTEM AND ITS DRAWBACKS

Currently at Hotspots there is no mechanism to track the position of users using MAC address. The only method available is the video scanning that is done usually in every possible public place. But on the other hand, much more information can be obtained by analysing the content of data the user accessed. Backtracking using this data can lead to only the MAC address of the user atmost.

By the existing system there is no method to obtain the position of a user based on his MAC address. Ironically it is the most important information needed. Thus it is an important drawback of the present system that has to be filled.

2.2 PROPOSED SYSTEM

Its an obvious thing to correct the drawbacks of the existing system if the technology is available relatively cheap. Thus the system being proposed here satisfies the need of tracking the users in a wireless network and helping the administrator with the same process. The system effectively calculates the position of the users to a high order of precession. Its also made sure that the computational cost involved is as low as possible and the assumptions made are very little.

Administrator will be provided with a good graphical user interface. The administrator will have a graphical view of the nodes present in the network. He will also be presented with alert messages when ever a new user enters the network.

CHAPTER 3

SYSTEM REQUIRED

The project involves use of two servers and there can be any number of clients. Since the project is done using GNU/Linux, we need the servers to have the operating system GNU/Linux. And since the project is related to WiFi, every device is expected to have a wireless connectivity device. The details of the same are described below.

3.1 HARDWARE REQUIREMENTS

THE TWO SERVERS

Processor: x86 or compatible, with 500MHz or higher for better performance.
RAM: 128 MB (minimum) and 256 MB (for average performance).
Hardisk: 20 GB or above.
Network card: Ethernet card and Wireless cards (preferably Intel Pro 3945 series).

NODES CONNECTING TO SERVER

Processor: x86 or compatible, with 500MHz or higher for better performance.
RAM: 128 MB (minimum) and 256 MB (for average performance).
Hardisk: 20 GB or above.
Network card: Ethernet card and Wireless cards.

3.2 SOFTWARE REQUIREMENTS

TWO SERVERS

Operating System: GNU/Linux (preferably Debian 5.0 or above).
Packages: GTK+ 2.0, Cairo, iw, compat wireless driver, wireless utilities.

NODES CONNECTING TO SERVER

Operating System: Any operating system.

Packages: Basic software to connect to a wireless network.

CHAPTER 4

SYSTEM DESIGN

4.1 BASIC DESIGN

The basic idea of the system is very simple. We find the distance of every node in the network from two points inside the refernce area. A line connecting these two points is taken as the reference x-axis. Based on these two distances one can apply simple Pythagoras theorem to obtain two equations with two unknown variables. These two equations are solved to obtain the x co-ordinate and y-cordinate of that node.

The main program resides and runs in a computer that is called the “main server”. This server is positioned in a such a way that it can be considered as the origin. The program requires certain initialisation. Administrator has to enter the IP of the “Helping Node”. It searches for this node and a socket connection is established. It tries to find out the distance at which this “Helping Node” is by using the iw program.

The distance between the main server and the helping node is recorded for future calculations. The program iw is run from a shell script which makes it to dump the details regardin the nodes connected to the network. It also redirects the data regarding each of the nodes to a file. The data stored in file are only a part of what iw dumps, that part which is needed to find the distance of the nodes and recognise them.

The data stored in file are directly read by the server program. This data is regarding the power of the signal that was received at the main server. It is also called RSSI or Received Signal Strength Index. This is expressed in decible. It must be provided by the device as well as the device driver must provide this value to the kernel or user space. This value in decibles can be used to calculate the distance that the data has travelled to reach the particular node. The formula to calculate this is mentioned in a later section.

The program iw runs in helper node as well in the same manner. When the main server requests the helper node for data, it is provided using the file that a similar script in the helper node (as in the server) redirects the “grep”ed output into. This is obtained in decible. Thus we have obtained the power at which the data has been received at the “Helper Node”. This is used to measure the distance using a formula mention later. This file is transfered to the main server via a socket connection.

We have thus distance to a point from two different location. We also know where these locations are with respect to each other. Now to find out the location of the point all we need is simple mathematics. The related diagram as well as the method of calculation has been described.

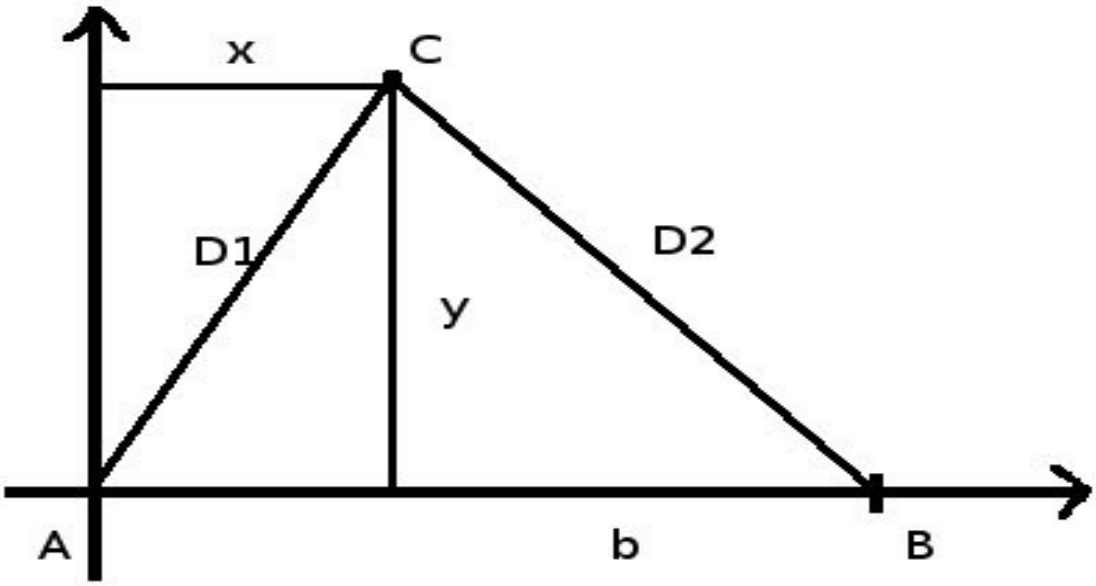

Fig 1.1 Finding location of a point.

4.2 MATHEMATICAL FORMULA

The main server is assumed to be situated at the origin of an x-y plane. It has been marked as point A. The helper node is at a distance 'b' from the main server and it is marked as point B. It is assumed that the point B is on the x axis so as to simplify the calculation. This assumption leads to the fact that the nodes shall only be present in just one quadrant of the plane. Though we can have

the first and second quadrant, the servers are placed in such a way that it covers the entire room. This makes the room as the first quadrant. This is a puposefully made assumption that reduces the complexity of the formula a lot.

Thus B is point (b , 0). Let the unknown point be C. It is evident from the figure 1 that we know three things: the distance of the point C from main server A – 'D1', the distance of the point C from the helper node B – 'D2' and the distance between the server and the helper node – 'b'. Applying the pythagorus theorem we get two equations in variable x and y. These equations can be solved to obtain the x and y co-ordinates.

The x and y cordinates of the point C is given by the equation:

$$\mathbf{x} = (D1^2 + b^2 - D2^2) / (2b)$$

$$\mathbf{y} = \sqrt{(D1^2 - x^2)}$$

In such way we can calculate the x cordinate and y cordinate of all the nodes.

CHAPTER 5

THE iw – MAIN TOOL USED

The wireless support for Linux had been limitted for past few releases. But when the Linux wireless team started its work on porting more drivers to Linux the whole story changed. They have introduced a new mac80211 stack in the kernel that removes a lot of old limitations in the kernel network stack. This new stack is based on the latest devices in the market and their capabilities. The Linux Wireless team also produced the mac80211 based drivers for almost all the devices in the market.

The introduction of new stack was followed by the arival of new userspace configuration interface, nl80211. Along with that a new in-kernel configuration was also produced – cfg80211. They currently support about 50 drivers. However, they don't implement all features and may have some issues, due to various reasons like companies not providing specs. All drivers can run in station mode, but not all support the other modes.

5.1 MAC80211

mac80211 is the Linux API used to write softmac wireless drivers. Right now the MLME is done in the kernel for station mode and userspace for AP mode but the goal is to eventually move it to userspace completely. Here is a quick review of the features supported in mac80211.

- IEEE 802.11abgn
- IEEE 802.11d
- Integration of work for the emerging 802.11s standard
- Roaming using wpa_supplicant (802.11r as well).
- Different types of interfaces.
- Vendor specific rate support.
- Quality of Service.

mac80211 supports 802.11d by processing country information element on beacons after association with an AP. One should still be able to associate to the AP in your region as cfg80211 allows users to set the regulatory domain from userspace before country information elements are parsed, this is expected to be set via wpa_supplicant upon initialization. It lets cfg80211 parse the country information element and deal with reviewing regulatory enforcement.

mac80211 rate control algorithms in current mac80211 are:

- PID (proportional-integral-derivative) rate control algorithm
- ministrel - a rate control algorithm making use of multi-rate retries

mac80211 creates creates one master device and as many other secondary devices as requested to represent interfaces for the wireless card you have. mac80211 asks for the master device to appear as named as *wmaster%d*, and *wlan%0* for the interfaces. udev may override the naming convention used though. wmaster%d is an internal master device used only by mac80211. It is currently visible only because it uses netdevice structure which we must allocate and use for QoS. It also serves as a holder for all interfaces we have, and represent the underlying hardware. For example, when TXing your wlan0 STA interface will actually add IEEE-802.11 header data to a frame with just Ethernet headers, and then pass it down to the master device for actual transmission using the low level drivers. The wlan%d devices (interfaces) are the devices you would use to configure your wireless settings.

5.2 NL80211 AND IW

nl80211 is the new 802.11 netlink interface public header. Together with cfg80211 it is intended to replace Wireless-Extensions. nl80211 and cfg80211 still under development. The main program that uses this nl80211 is the program iw tool. The iw tool is used to show or manipulate wireless devices and their configurations.

iw is a new nl80211 based CLI configuration utility for wireless devices. Currently you can only use this utility to configure devices which use a mac80211 driver as these are the new drivers being written – only because most new wireless devices being sold are now SoftMAC. These

programs are available for download in the main page of the linux wireless community. Using iw requires you to have libnl, the first working version is 1.0 pre8 as this release introduced genl, *Generic Netlink*, which nl80211 relies on. Most distributions of Linux are shipping 1.1 version of iw these days.

5.3 IW IN THIS PROJECT

This tool is used in this project to obtain the signal strength of nodes that are to be monitored. The shell scripts call the iw, redirect its output to a temporary file, parse the file to obtain required information and store it in a separate file for the main program to read from. A sample output from the iw is shown below:

```
$ iw dev wlan1 station dump
        Station 12:34:56:78:9a:bc (on wlan0)
                inactive time:  304 ms
                rx bytes:       18816
                rx packets:     75
                tx bytes:       5386
                tx packets:     21
                signal:         -29 dBm
                tx bitrate:     54.0 Mbit/s
```

Here the wlan1 interface's station's statistics are shown. A device whos MAC address is 12:34:56:78:9a:bc is detected in the network. The computer in which this program is running has been inactive with the above mentioned node for 304ms. It simply shows the time that has elapsed since the last transmission to the node mentioned. It also shows how many packets were received or transmitted and also the bytes assosiated with it.

The most important part as far as this project is concerned, is the signal strength. It shows the strength of the packet that was last received from that node. This might have changed in the

mean time. But if the packets have not arrived after the change has occurred, the signal strength will not have changed. It also mentions about the transmission bit rate.

For getting device capabilities of all devices, such as band information (2.4 GHz, and 5 GHz), and 802.11n information the following command can be used:

$ iw list

Scanning for nodes present in all network can be done using the following command. It shall show all the devices present in the locality:

$ iw dev wlan0 scan

There are different events assosiated with a wireless device. The events may be listesned to. Listening to an event means reporting or providing a particular output when the reffered event occurs. Listening to events can be done using iw by using the following command:

$ iw event

When debugging, it can be useful to see the auth/assoc/deauth/disassoc frames. These frames can be seen by using the command:

$ iw event -f

Timing is often as important as the above mentioned frames. Information regarding timing can be obtained by using:

$ iw event -t

If you want to get specific statistics against a peer your station is communicating with you can use the following command:

$ iw dev wlan1 station get <peer-MAC-address>

In the case of a STA the above <peer-MAC-address> would be the MAC address of your AP. When connected to an AP you cannot get station information of any other user.

Iw can also be used for adding interfaces. There are several modes supported. The modes supported are:

- monitor
- managed [also station]
- wds
- mesh [also mp]
- ibss [also adhoc]

To add a monitor interface we can use the command:

$ iw phy phy0 interface add moni0 type monitor

where you can replace monitor by any other mode and moni0 by the required interface name. We must also replace phy0 by the PHY name for your hardware (usually phy0 will be correct unless hotplugged or reloaded any modules.) If udev is configured incorrectly, the newly created virtual interface may be renamed by it right away, use ip link to list all interfaces.

Another example is to create a new managed mode interface. The interface is automatically put into AP mode when using hostapd. This would use the following command:

$ iw phy phy0 interface add wlan10 type managed

You can customize the type of monitor interface you create. This can be very useful for debugging purposes on end user systems. For example suppose we want to help a user. We can take

advantage of the fact that a monitor interface in mac80211 uses radiotap to pass upto userspace additional data. Say we want to help a user fish out data without affecting the device's performance by setting it it to a full monitor interface an monitor interface with no additional monitor flags can be created as follows:

$ iw dev wlan0 interface add fish0 type monitor flags none

We can then request the user to use tcpdump on a session:

$ tcpdump -i fish0 -s 65000 -p -U -w /tmp/fishing.dump

The nice thing about these type of alternative monitor interfaces is we can further extend radiotap even with vendor extensions to add more data to radiotap to help debug device specific features. This requires drivers to honor mac80211's flag requests strictly, so drivers like ath5k and ath9k which still enable flags based on operation mode need to be fixed to take advantage of this.

The following are monitor flags we can specify:

- none
- fcsfail
- plcpfail
- control
- otherbss
- cook

The command line for deleting interfaces with iw is shown next:

$ iw dev moni0 del

CHAPTER 6

IMPLEMENTATION

6.1 INITIAL IMPLEMENTATION

As an initial implemetation a model was created in a non-graphical-user-interface. It asks the user for the IP address of the host computer, the helping computer and also the nodes that are to be scanned. Files are created that are similar to the output from the iwspy utility. These files are expected to have a particular name and are expected to be in a particular location.

The files are opened and read. The IP to be scanned are present in an array. The first IP is compared with the IP's present in the file. When a match is found the corresponding signal strenght is recorded. Similary, the helper's file is also read. These values are converted into the distance using the concept of free space loss theory. These distances are used in the mathematical formula derived in the earlier section to obtain the cordinates of the same. This value is displayed. It is repeated for all the nodes present in the “to-be-scanned” list of IP addresses.

6.2 LIMITATION OF IWSPY

Even though initially the tool iwspy was used, the fact that only few drivers support it was soon realised. The limitation with the driver was evident when the nodes to be spyed showed zero as the signal strength. This was due to the fact that support of iwspy or any of the wireless extensions were not provided by either the manufacturing company or the driver developers.

The limitations with iwspy was also evident as newer capabilities could not be implemented with the wireless extensions very old and premitive design. This lead to the creation of the linux wireless team which created the next generation wireless drivers and utilities based on newer stacks.

Thus in this project also the use of iwspy was soon discarded. The program now uses the iw

utility that uses the latest compat drivers. The compat drivers are capable of showing the signal strength of the nodes when connected in ad-hoc and managed mode at the time of development of this project. Since fast upgrades and high amount of work is going on in the linux wireless team, the support for all the devices and all the functionalities available for them should soon be completed.

6.3 GUI IMPLEMENTATION

The main program was implemented with a graphical user interface. The program is divided into various modules. The implementation is explained in detail here.

6.3.1 SERVER PROGRAM

The main program runs in the server. It is implemented with a good graphical user interface. This will aid the administrator to properly control the program to benifit the most. The main program is responsible for the following things:

- Establish a connection with the helping server.
- Collect signal strength information regarding the nodes present in the network including the helping server.
- Collect signal strength information regarding the nodes present in the network from the helping server.
- Use the signal strength information received to calculate the distance of every node from the server as well as from the helping server.
- Use these distance to calculate the co-odrinates of the nodes present in the network.
- Plot this information in the GUI.
- Alert the administrator when a new node enters the network.

All these functionalities are modules of the main program. They are explained furthur. When the administrator sets the helping servers IP and create a network including the helping server also in the network, he must press the scan button for the program to start scanning the network. Now

this event of clicking the button triggers a flag to be set in the function handling this event. This flag is used to denote whether the action to be done is scanning or stalling. If it is scanning, the function creates a thread. This thread continously scans the network. Creating a thread is necessary for proper working of the GUI. Had it been in non-GUI, it would not have been a problem without thread.

The thread is the most important part of the main program. Its called scan_thread. It checks the flag that was earlier set to indicate scanning or stalling action. It continues to work in an infinite loop as long as the flag is set. It calls functions to measure the strength of all nodes from within the main server. Then it calls the function that receives the corresponding data from the helping server. Using these data, it calls another function to locate and plot the co-ordinates of the nodes. Thus this thread becomes the crucial part of the program.

Measuring signal strength from within the server os done by the measure_strength_ip function. It runs the iw command, redirecting the output to a file named iw_data. This file is then immediately opened. First, the first node's MAC address is read. This is compared with the list of MAC addresses we already have. This is stored in a global array track_ip. If found, the index is noted for the furthur processing. If it is not found in the array track_ip, it means that we hav found a new node connected to the system. The program now responses with a warning message for the user in the message display area. This will thus help the administrator to be aware of new entries of nodes into the network.

If the node is one that was already present in the network in the previous scan, its signal strength is converted to the corresponding distance and stored in another global array dist1 with the same index as that of corresponding MAC address in track_ip. This is repeated for all the nodes obtained from iw dumbing. The new node found is also given a new location in track_ip, incrementing the total number of IP's being tracked which is stored in global varialble no_of_ip. The index is used to fill its distance also from the main server in the dist1 array.

The measure_from_helper function is responsible for filling the array dist2 in similar fashion. It also reports when it finds a new node. This function initially calls a function receive_help

that connects with the helping server using TCP. The helping server will be configured to listen to a port to which this function connects. It the receives the file similar to iw_data and store locally with name iw_data_help. This will contain the dump of iw in the helping server. This will be parsed as in the measure_strength_ip function and desired action is done.

Thus we are having dist1 and dist2 containing the details about the distances of the nodes from the main and helping server respectively. The next function called is locate_and_draw function. This function uses the two arrays dist1 and dist2 and use the pythogorus equation to obtain two functions mentioned in an earlier section. These equations will have two unknowns, x and y co-ordinate of the node. The two equations can be solved to get this value. This is also done in the function locate_and_draw.

The solution of equation gives x and y co-ordinates of the node. This is stored in anothr global array x_cord and y_cord. This is done so that it can be plotted as well as displayed in words in the program. This recording is done for all the no_of_ip nodes. Then this value is used to display in another page of the notebook styled display present in the program.

The scan_thread thread then invlidates the entire window. Invalidation forces the GTK library to re-draw the invalidated region again immediately. This is done by emitting an event “expose”. The drawingarea which displays the nodes' positions graphically has a function that handles the expose event. This function uses Cairo library to draw the positions. It uses the x_cord and y_cord array to get the location of each node.

The drawing area is redrawn by first drawing the x and y co-ordinates. Then the function scans the x_cord and y_cord one by one. It takes the x and y cordinates of i'th node, move to that location, draw a small rectangle of size 5 pixels denoting the node. It is stroked and this is repeated for all the nodes. Stroking forces the image to be drawn on to the drawable area. Thus after the expose-event assosiated function finishes execution the latest position of the node is obtained.

The program also has a message display area that keeps track of the actions of administrator like starting and stalling of scanning as well as the warnings that are to be given to the administrator

like entry of a new node in to the network. Each time such events occur the required message to be displayed is appended to the message display area.

The program also contains a page that shows the co-ordinates of the nodes in the network. While the drawing are shows the co-ordinates graphically, this page, titled Nodes, shows the co-ordinates in words. This page is updated by the locate_and_draw function.

6.3.2 HELPING SERVER

Helping server is an important entity of the project. It is responsible for the following functions:

- Obtain the statistics regarding the nodes in the network using iw station dump command.
- Store these values in a file.
- Listen for connection request from the main server at a pre-defined port.
- When connected provide the main server with the file in which the station dump was stored.

Since the administrator will not be present at the helping server once it has been setup, the helping server need not have a GUI. It has only one function that does all the required work. Initially it is attached to the network to which the main server belongs. Then its IP is configured. It then creates a socket and waits for the connection request from the main server. Once the main server has established a connection with the helping server, it executes the station dump, redirects it to a file, and then immedialtely open that file. The main server is then provided with this file before the connection is closed. It again starts to wait for the main server to connect again.

This simple program keeps on running within the helping server unless it is stoped explicitly. This program thus provides the iw station dump to the server and does no calculation at all.

CHAPTER 7

GTK +

GTK+, or The GIMP Toolkit, is a cross-platform widget toolkit for creating graphical user interfaces. It is one of the most popular toolkits for the X Window System, along with Qt. GTK+ was initially created for the GNU Image Manipulation Program (GIMP), a raster graphics editor, in 1997 by Spencer Kimball and Peter Mattis, members of eXperimental Computing Facility (XCF) at UC Berkeley. It is licensed under the LGPL, GTK+ is free software and is part of the GNU Project.

7.1 DESIGN

GTK+ is written in the C programming language, and its design uses the GObject object system. The GNOME platform provides language bindings for:

- C++ (gtkmm)
- Perl (Gtk2-perl)
- Ruby (ruby-gtk2)
- Python (PyGTK)
- Java (java-gnome) (not available for Microsoft Windows)
- C# (Gtk#)
- PHP (PHP-GTK)

Others have written bindings for many other programming languages (including Ada, D, Haskell, Lua, Ocaml, Pascal, Pike, Javascript, Tcl, Euphoria and all .NET programming languages). GTK-server provides a stream-based IPC interface to GTK+ allowing it to be used from any language with I/O capabilities, including shell scripts. Bindings for many languages can be generated automatically via Gobject-introspection. Languages purpose-written for GObject and therefore GTK+ include Vala and GOB.

Like Qt, but unlike several other widget toolkits, GTK+ is not based on Xt. This allows

flexibility and allows GTK+ to be used on platforms where the X Window System is unavailable. However, without this dependency, GTK+ lacks access to the X resources database, the traditional way for customizing X11 applications.

GTK+ initially contained some utility routines that did not strictly relate to graphics, for instance providing such data structures as linked lists and binary trees. Such general utilities, along with the object system called GObject, have now migrated into a separate library, GLib, which programmers can use to develop code that does not require a graphical interface.

7.2 PLATFORMS

GTK+ was originally targeted at the X Window System, and this remains its primary target platform. Other targeted platforms are Microsoft Windows (Windows 2000 and upwards, near complete support), DirectFB, and Quartz (Mac OS X v10.4 and upwards, still under development).

The end-user can configure the look of the toolkit, down to offering a number of different display engines. Engines exist which try to emulate the look of other popular toolkits or platforms such as Windows 95, Motif, Qt and NEXTSTEP.

7.3 COMPILING

Suppose that a program named file.c has been developed using GTK+ library. To compile it use:

gcc -Wall -g file.c -o file `pkg-config --cflags gtk+-2.0 --libs gtk+-2.0`

This uses the program pkg-config. This program reads the .pc which comes with GTK to determine what compiler switches are needed to compile programs that use GTK. pkg-config --cflags gtk+-2.0 will output a list of include directories for the compiler to look in, and pkg-config --libs gtk+-2.0 will output the list of libraries for the compiler to link with and the directories to find

them in. In the above example they could have been combined into a single instance, such as pkg-config --cflags --libs gtk+-2.0. Note that the type of single quote used in the compile command above is significant.

The libraries that are usually linked in are:

- The GTK library (-lgtk), the widget library, based on top of GDK.
- The GDK library (-lgdk), the Xlib wrapper.
- The gdk-pixbuf library (-lgdk_pixbuf), the image manipulation library.
- The Pango library (-lpango) for internationalized text.
- The gobject library (-lgobject), containing the type system on which GTK is based.
- The gmodule library (-lgmodule), which is used to load run time extensions.
- The GLib library (-lglib), containing miscellaneous functions; only g_print() is used in this particular example. GTK is built on top of GLib so you will always require this library. See the section on GLib for details.
- The Xlib library (-lX11) which is used by GDK.
- The Xext library (-lXext). This contains code for shared memory pixmaps and other X extensions.
- The math library (-lm). This is used by GTK for various purposes.

7.4 PACKING WIDGETS

When creating an application, you'll want to put more than one widget inside a window. When you want to put more than one widget into a window, how do you control where that widget is positioned? This is where packing comes in.

Most packing is done by creating boxes. These are invisible widget containers that we can pack our widgets into which come in two forms, a horizontal box, and a vertical box. When packing widgets into a horizontal box, the objects are inserted horizontally from left to right or right to left depending on the call used. In a vertical box, widgets are packed from top to bottom or vice versa. You may use any combination of boxes inside or beside other boxes to create the desired effect.

To create a new horizontal box, we use a call to gtk_hbox_new(), and for vertical boxes, gtk_vbox_new(). The gtk_box_pack_start() and gtk_box_pack_end() functions are used to place objects inside of these containers. The gtk_box_pack_start() function will start at the top and work its way down in a vbox, and pack left to right in an hbox. gtk_box_pack_end() will do the opposite, packing from bottom to top in a vbox, and right to left in an hbox. Using these functions allows us to right justify or left justify our widgets and may be mixed in any way to achieve the desired effect. We will use gtk_box_pack_start() in most of our examples. An object may be another container or a widget. In fact, many widgets are actually containers themselves, including the button, but we usually only use a label inside a button.

By using these calls, GTK knows where you want to place your widgets so it can do automatic resizing and other nifty things. There are also a number of options as to how your widgets should be packed. As you can imagine, this method gives us a quite a bit of flexibility when placing and creating widgets.

7.5 BUTTONS

You can use the gtk_button_new_with_label() or gtk_button_new_with_mnemonic() to create a button with a label, use gtk_button_new_from_stock() to create a button containing the image and text from a stock item or use gtk_button_new() to create a blank button. It's then up to you to pack a label or pixmap into this new button. To do this, create a new box, and then pack your objects into this box using the usual gtk_box_pack_start(), and then use gtk_container_add() to pack the box into the button.

The Button widget has the following signals:

- pressed - emitted when pointer button is pressed within Button widget.
- released - emitted when pointer button is released within Button widget.
- clicked - emitted when pointer button is pressed and then released within Button widget.
- enter - emitted when pointer enters Button widget.
- leave - emitted when pointer leaves Button widget.

7.6 GLADE INTERFACE DESIGNER

Glade Interface Designer is a graphical user interface builder for GTK+, with additional components for GNOME. In its third version, Glade is programming language–independent, and does not produce code for events, but rather an XML file that is then used with an appropriate binding (such as gtkada for use with the Ada programming language). Glade comes in three versions, one for GTK+ 1 and two for GTK+ 2. Glade is free software distributed under the GNU General Public License. Here we have used Glade 2 for GTK+ 2.0.

Glade 2 automatically generates some of the codes related to the GUI that was designed using it. This is only the basic skeleton code which will do nothing when executed other than showing all the widgets that were placed.

CHAPTER 8

CAIRO

Cairo is a software library used to provide a vector graphics-based, device-independent API for software developers. It is designed to provide primitives for 2-dimensional drawing across a number of different backends. Cairo is designed to use hardware acceleration when available. Although written in C, there are bindings for using the cairo graphics library from many other programming languages, including Factor, Haskell, Lua, Perl, Python, Ruby, Scheme, Smalltalk and several others. Dual licensed under the GNU Lesser General Public License and the Mozilla Public License, cairo is free software.

The cairo API provides operations similar to the drawing operators of PostScript and PDF. Operations in cairo including stroking and filling cubic Bézier splines, transforming and compositing translucent images, and antialiased text rendering. All drawing operations can be transformed by any affine transformation (scale, rotation, shear, etc.)

8.1 CAIRO'S DRAWING MODEL

In order to explain the operations used by cairo, we first delve into a model of how cairo models drawing. There are only a few concepts involved, which are then applied over and over by the different methods. The nouns are explained first: destination, source, mask, path, and context. After that describe the verbs which offer ways to manipulate the nouns and draw the graphics you wish to create.

NOUNS

Cairo's nouns are somewhat abstract. To make them concrete I'm including diagrams that depict how they interact. The first three nouns are the three layers in the diagrams you see in this section. The fourth noun, the path, is drawn on the middle layer when it is relevant. The final noun, the context, isn't shown.

DESTINATION

The destination is the surface on which you're drawing. It may be tied to an array of pixels like in this tutorial, or it might be tied to a SVG or PDF file, or something else. This surface collects the elements of your graphic as you apply them, allowing you to build up a complex work as though painting on a canvas.

SOURCE

The source is the "paint" you're about to work with. I show this as it is—plain black for several examples—but translucent to show lower layers. Unlike real paint, it doesn't have to be a single color; it can be a pattern or even a previously created destination surface. Also unlike real paint it can contain transparency information—the Alpha channel.

MASK

The mask is the most important piece: it controls where you apply the source to the destination. I will show it as a yellow layer with holes where it lets the source through. When you apply a drawing verb, it's like you stamp the source to the destination. Anywhere the mask allows, the source is copied. Anywhere the mask disallows, nothing happens.

PATH

The path is somewhere between part of the mask and part of the context. It is manipulated by path verbs, then used by drawing verbs.

CONTEXT

The context keeps track of everything that verbs affect. It tracks one source, one destination, and one mask. It also tracks several helper variables like your line width and style, your font face

and size, and more. Most importantly it tracks the path, which is turned into a mask by drawing verbs.

Before you can start to draw something with cairo, you need to create the context. The context is stored in cairo's central data type, called cairo_t. When you create a cairo context, it must be tied to a specific surface—for example, an image surface if you want to create a PNG file. There is also a data type for the surface, called cairo_surface_t. You can initialize your cairo context like this:

```
cairo_surface_t *surface;
cairo_t *cr;
surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32, 120, 120);
cr = cairo_create (surface);
```

The cairo context in this example is tied to an image surface of dimension 120 x 120 and 32 bits per pixel to store RGB and Alpha information. Surfaces can be created specific to most cairo backends, see the manual for details.

VERBS

The reason you are using cairo in a program is to draw. Cairo internally draws with one fundamental drawing operation: the source and mask are freely placed somewhere over the destination. Then the layers are all pressed together and the paint from the source is transferred to the destination wherever the mask allows it. To that extent the following five drawing verbs, or operations, are all similar. They differ by how they construct the mask.

STROKE

The cairo_stroke() operation takes a virtual pen along the path. It allows the source to transfer through the mask in a thin (or thick) line around the path, according to the pen's line width,

dash style, and line caps.

cairo_set_line_width (cr, 0.1);
cairo_set_source_rgb (cr, 0, 0, 0);
cairo_rectangle (cr, 0.25, 0.25, 0.5, 0.5);
cairo_stroke (cr);

FILL

The cairo_fill() operation instead uses the path like the lines of a coloring book, and allows the source through the mask within the hole whose boundaries are the path. For complex paths (paths with multiple closed sub-paths—like a donut—or paths that self-intersect) this is influenced by the fill rule. Note that while stroking the path transfers the source for half of the line width on each side of the path, filling a path fills directly up to the edge of the path and no further.

cairo_set_source_rgb (cr, 0, 0, 0);
cairo_rectangle (cr, 0.25, 0.25, 0.5, 0.5);
cairo_fill (cr);

CHAPTER 9

NETWORKING IN LINUX

The fundamental question is this: Is a Linux system capable of integrating with other access points in a wireless network to provide connectivity between fixed nodes and roaming wireless clients? The actual installation of a wireless LAN is not such a big deal; the key lies in what you buy, and whether it is supported under a Linux system. Check to make sure that you have the hardware for a specific operating system driver. Keep in mind that if you are looking for high performance (that is, gigabit data transfer rates), wireless is the wrong choice for you; even the latest standards offer transmission rates under 100 Mbps. Roaming presents a different challenge: maintaining a connection between various access points. To help with this, set up the Wired Equivalent Privacy (WEP) key to detect an access point that you can connect to. Keep things down to earth and remember that the main purpose of an access point is to be a bridge; that is, it should route packets from one network to another. The Linux Wireless Tools package is installed by default probably meets most of your 802.11a/b needs.

9.1 USING IWCONFIG FOR WIRELESS-TOOLS CONFIGURATION

After physically installing your Linux-compatible NIC, you need to configure your NIC's IP and wireless settings before Wireless Tools works. You can configure your NIC's IP settings as if the NIC were a regular Ethernet device. After you use the ifup command the NIC becomes active, but it will not function correctly as its wireless settings haven't been configured yet.

The most commonly used command in Wireless Tools is iwconfig, which you can use to configure most of the wireless parameters, including the SSID and the wireless mode. For the wireless mode, Managed means that there is a wireless access point (WAP) on the network and Ad-hoc signifies that there is none. For example, if your wireless NIC is named eth0 and your managed network's ESSID is homenet, then the commands would be.

iwconfig eth0 mode Managed

iwconfig eth0 essid homenet

Your NIC should now become fully functional. You will need to run these iwconfig commands each time you use the ifup command, however; forgetting to do so can be problematic. After testing your ad-hoc configuration, you will need to make the changes permanent. The methods for doing this vary slightly by distribution.

Fedora / RedHat

With Fedora / RedHat, wireless configuration will require some additional statements in your NIC configuration files.

1. Configure your /etc/sysconfig/network-scripts/ifcfg-eth0 file normally as if it were a regular Ethernet NIC.

DHCP Version ***Fixed IP Version***

========== **==========**

DEVICE=eth0 ***DEVICE=eth0***

USERCTL=yes ***IPADDR=192.168.1.100***

ONBOOT=yes ***NETMASK=255.255.255.0***

BOOTPROTO=dhcp ***ONBOOT=yes***

BOOTPROTO=static

2. Add the following statements to the end to specify that the NIC is wireless; provide the ESSID to use (in this case homenet), and choose Managed (a WAP on present of the network) or Ad-hoc (no WAP) for the wireless mode. "Managed" is the most likely setting if you have a wireless router or WAP on your network.

If you are using a 802.11g wireless router and NIC, you can specify the higher speed 54Mbps maximum data rate this protocol provides, if not, the NIC will default to the 11 Mbps maximum rate of slower protocols. The NIC will automatically negotiate the protocol type with the WAP. You just need to set the maximum rate.

#
 # Wireless configuration
#
TYPE=Wireless
MODE=Managed
ESSID=homenet
RATE=54Mb/s

These commands need only be on the main interface file. They are not needed for IP aliases. Your wireless NIC should function as if it were a regular Ethernet NIC using the ifup and ifdown commands.

Debian / Ubuntu

In Debian / Ubuntu systems configuration requires the addition of a valid wireless-essid parameter to the /etc/network/interfaces file.

#
 # File: /etc/network/interfaces
#

 # The primary network interface
auto eth1

iface eth1 inet static

address 192.168.1.100

netmask 255.255.255.0

wireless-essid homenet

auto eth0

iface eth0 inet dhcp

wireless-essid jamrock

In this example interface eth1 uses an ESSID of homenet while interface eth0 uses an ESSID of jamrock.

9.2 WEP ENCRYPTION CONFIGURATION

Linux supports both the WEP and WPA encryption schemes. Here's how you can configure them on your system. WEP encryption requires an encryption key that you can make up yourself or you can generate a random one using the dd command as shown here.

$ dd if=/dev/random bs=1 count=5 2>/dev/null | xxd -ps

c276246d65

By default, Linux WEP uses a 40 bit key formatted in hexadecimal notation, ie. numeric values between 0 and 9 and alphabetic characters between A and F. This requires you to use a byte count of 5, which will generate a key containing twice as many (ten) hexadecimal characters. Table 13.1 shows the byte counts required for generating keys of varying lengths, and the corresponding number of hexadecimal characters to expect in the key. If you decide to make up your own key, then remember to use the correct number of hexadecimal numbers.

WEP Key Configuration for Fedora / RedHat

Your WEP key can be temporarily added to your NIC configuration from the command line, using the iwconfig command. Be sure that there are no colons or any other non-hexadecimal characters between the characters of the key. There should be ten characters in total:

iwconfig eth0 key 967136deac

The same rules (no colons or non-hexadecimals between the ten total characters) apply when using the /etc/sysconfig/network-scripts files to add encryption:

#
 # File: ifcfg-eth0
#

DEVICE=eth0
IPADDR=192.168.1.100
NETMASK=255.255.255.0
ONBOOT=yes
BOOTPROTO=static
TYPE=Wireless
MODE=Managed
ESSID=homenet
KEY=967136deac

Newer versions of Fedora only support the use of a keys file in the /etc/sysconfig/network-scripts directory. The file format is the same as in the older interface configuration file method. Remember, the KEY statement in interface configuration file won't be supported.

#
 # File: /etc/sysconfig/network-scripts/keys-eth0
#

KEY=967136deac

WEP Key Configuration for Debian / Ubuntu

In Debian / Ubuntu systems configuration requires the addition of a valid wireless-key parameter, alongside the wireless-essid parameter, in the /etc/network/interfaces file.

```
#
# File: /etc/network/interfaces
#

# The primary network interface
auto eth1
iface eth1 inet static
        address 192.168.1.100
        netmask 255.255.255.0
        wireless-key 967136deac
        wireless-essid homenet
```

In this example our WEP key of 967136deac and the ESSID of homenet have been used and will become utilized once the eth1 wireless interface is activated.

9.3 WPA ENCRYPTION

Linux WPA relies on a supplicant daemon program that both requests authentication admittance and executes data encryption on behalf of the operating system. It runs independently of the networking daemon and so, for WPA, network interfaces are not configured for encryption at all. Installation is simple. Install the wpa_supplicant RPM or the wpasupplicant DEB package.

The wpa_supplicant.conf File

The main WPA Supplicant configuration file is /etc/wpa_supplicant/wpa_supplicant.conf and its configuration is well documented, with examples, in the man pages.

$ man wpa_supplicant.conf

With Debian / Ubuntu the file may not be created during installation, and you will have to create it manually like this:

$ mkdir -p /etc/wpa_supplicant
$ vi /etc/wpa_supplicant/wpa_supplicant.conf

In this example, we have set the SSID to homenet and are using WPA-PSK encryption with an encryption key of "ketchup_and_mustard".

```
#
# File: wpa_supplicant.conf
#
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=root
network={
        ssid="homenet"
        key_mgmt=WPA-PSK
        psk="ketchup_and_mustard"
}
```

If you are concerned about people being able to read your wpa_supplicant.conf file, then encrypt the PSK using the wpa_passphrase command to generate a sample configuration. It requires the SSID and unencrypted key as arguments. In this example we see that the unencrypted string psk="ketchup_and_mustard" can be replaced with and encrypted equivalent that does not use quotes.

```
$ wpa_passphrase homenet ketchup_and_mustard
network={
        ssid="homenet"
        #psk="ketchup_and_mustard"
        psk=aeaa365d1703f88afc11715cd997b71038ce5798907510bd1b1c6786d33c8c3a
}
```

The only place that an encryption key needs to be defined is in the WPA configuration file.

CHAPTER 10

FUTURE ENHANCEMENTS

This project has immense possibilities. This can actually be aimed at being implemented in an airport or a place where public hotspot is available. Other areas of application like using it for navigation are also possible and must be considered. Some of the future enhancements are mentioned below:

- Providing an Access Point configuration wizard.
- Provide a more generic measuring tecnique involving more helping nodes and not limitting the area to the first quadrant of the plane.
- Consider the noise as well as signal loss due to absorption, in the calculations. This will provide more accurate results.
- Implementation as a navigation system is a possible application of signal strength measurement.
- Emergency calls' location tracking can also be a good implementation.

CHAPTER 11

CONCLUSION

The project has been implemented and this has been tested. More error handling must be implemented so as to make the program more robust. There were also minor errors found in the signal strength measurements. This is due to the energy saving modes that have been implemented in the drivers to save power consumed. Some mechanism can be devised to correct this behaviour.

The project has immense scope as a commercial product. Its social importance in security implementation also adds to the value to the project.

CHAPTER 12

REFERENCES

- www.linuxwireless.org
- www.linuxhomenetworking.com
- GTK+ 2.0 documentation (available at library.gnome.org/devel/gtk*)*
- Cairo documentation (available at cairographics.org/documentation)

APPENDIX A

SCREEN SHOTS

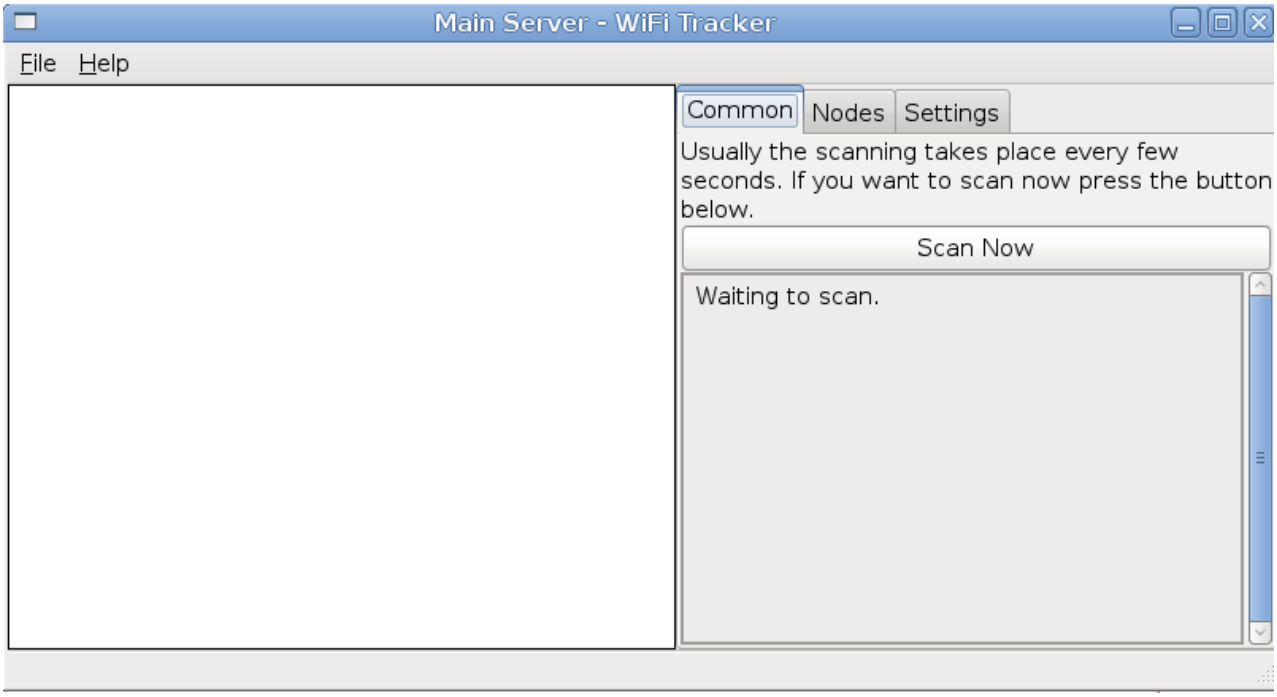

Fig 1: The main server program.