UPPSALA
UNIVERSITET

IT 13 075
Examensarbete 15 hp
Oktober 2013

Platform-independent indoor positioning system

Robin Broberg
Fredrik Gadnell

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Platform-independent indoor positioning system

Robin Broberg and Fredrik Gadnell

The purpose of this thesis is to investigate the possibility and feasibility of a platform-independent positioning system capable of determining the location of mobile devices without imposing additional requirements on the hardware and software of the device other than supporting Wi-Fi. Focus is on designing a scalable system capable of positioning devices on multiple sites. The suggested solution uses commercially available Wi-Fi access points to observe the received signal strength when a device probes for nearby access points. The information gathered by the access points is forwarded to a server cluster that uses the signal strength data to determine the location of the device.

Handledare: Kenneth Andersson
Ämnesgranskare: Christian Rohner
Examinator: Olle Gällmo
IT 13 075
Tryckt av: Reprocentralen ITC

Table of contents

1 Introduction

Positioning mobile users using GPS became increasingly popular with the release of smartphones. GPS gives great accuracy as long as the device has a sufficient number of satellites within line of sight, but quickly loose precision indoors[1].

The most common solution to indoor positioning is to measure the signal strength to nearby Wi-Fi access points and use trilateration to estimate the position relative to the Wi-Fi access points[2]. This solution requires that the operating system of the device allow developers to acquire this information, which several popular systems such as iOS and Windows Phone do not[3]. Apple provides a much more abstracted point of entry to the location data. Developers can register their app as a receiver of location updates, which use their underlying private frameworks. These location updates are based on a crowdsourcing approach where devices that have acquired a location fix by GPS share their location and pair the location with the list of nearby access points[4]. This approach is great for global coverage, but lacks precision indoors due to the fact that there are no good reference locations to use as the reference locations are based on GPS. It is also possible to acquire the raw data (i.e. signal strength to each node in range) by reverse engineering the private framework headers, but doing so violates the AppStore guidelines. Because of this, most existing systems are only available for Android.

In this thesis we design and discuss the problems and solutions of implementing a platform-independent indoor positioning system. A system that can position a mobile device independently of what operating system it is running.

1.1 Background

This thesis project was carried out in collaboration with The Mobile Life in Stockholm, Sweden. The Mobile life desired a solution that could position mobile devices indoors at large shopping malls independent of the mobile phone's operating system. The platform-independency requirement originated from that they had identified that most existing indoor positioning solutions only supported the Android platform.

Some preliminary research was done on the subject and the conclusion was reached that an on-device software solution is impossible with the current version of iOS, unless custom firmware is installed[3].

An alternate approach to indoor positioning was discussed. The idea was to position the device by measuring the received signal strength at each access point instead of the received signal strength on the device. This would result in a platform-independent solution with support for tracking devices without software installed on the device.

The research department of Ericsson, Ericsson Labs, showed interest in our project and suggested we meet monthly to discuss the research and possibilities of the technology. Access to their maps API was provided for use in this thesis project. The API allowed users to visually create maps of the site of interest that was then stored in an XML based format for positioning systems to download [5].

1.2 Problem Description

To make a platform-independent system, this thesis focuses on passive listening of mobile phones and designing a system that can scale to handle and locate a large amount of devices. The platform-independent approach to indoor positioning requires a very different architecture than on-device software solutions. Custom software on each access point is required to forward the data received by devices in the area, in effect becoming sensors for the tracking system.

A thorough discussion covering the theory and methods required to develop and install such software on the access points is found in chapter 4.
The sensor data forwarded by the sensor network has to be received and used to compute probable locations for each device on the network. While an on-device software solution only needs to compute the location of a single device, the platform-independent solution discussed in this thesis has to be scalable to handle large networks. These issues, along with a more general overview of the system architecture required, are discussed in chapter 3.

1.2.1 Purpose

The purpose of this thesis is to explore a platform-independent approach to indoor positioning. By removing all requirements except the basic need for Wi-Fi from the device, a general tracking system

can be developed. Indoor positioning has many uses, including indoor navigation and visitor analytics.

The approach discussed in this thesis expands the possibilities in these areas and enables new features such as tracking arbitrary Wi-Fi enabled hardware, including tablets and laptops, independent their operating system.

1.2.2 Method

First, some initial research on the subjects of indoor positioning, distributed systems and Wi-Fi on smartphones was done. The frequency of transmissions from common devices determines the possibilities of implementing a platform-independent positioning system.

A proof of concept was developed to confirm that the transmissions of common Wi-Fi enabled devices can be monitored on local access points and that the signal strength can be extracted and forwarded to external servers. The software required on the access points to monitor outgoing signals of nearby devices was implemented, as well as the necessary communication between access points.

The general architecture of the positioning system was designed, with focus on scalability and stability. The necessary software was then implemented and tested locally by simulating a live environment. The access point network and the positioning system was then integrated and a working prototype was developed and tested.

1.2.3 Delimitations

This thesis discusses on the general problems and issues of designing and implementing a platform-independent indoor positioning system, with focus on scalability and stability.

The algorithms and methods used to compute locations and track devices are only briefly mentioned. Plotting locations on maps, path finding and navigation are not part of this report.

2 Device probing

Mobile devices with support for the 802.11 Wireless Local Area Network protocol emit signals that can be captured by the access points. When a signal is captured the access point calculates the arriving signal strength and translates it to an indicator known as Receiving Signal Strength Indicator (RSSI).
To work around the platform dependence that is enforced on mobile devices, extra hardware is added in addition to the mobile device in the form of access points that listen for signals on the wireless network frequency.

The RSSI at each access point is then used to compute the distance to the transmitting device by using a model of the signal's behavior in an indoor environment. Once 3 or more nodes have estimated the distance to the transmitting device, the position of the device can be computed by trilateration [6]. Trilateration is the process of computing the location of a point based on distance measurements to other points. In the case of device positioning, the point of interest is the device transmitting the signal. The measurements are determined by estimating the distances to the receiving Wi-Fi access points in range.

2.1 Active probing

Since signals transmitted by the device being positioned is required to compute a location, it is vital that the device transmits signals frequently. One efficient way to generate outgoing traffic from a device is to use ping. Ping is used to test the reachability of a peer, in this case the mobile device. When ping tests the reachability of a peer, the peer responds with an acknowledgement saying it is available. This response generates a signal from the mobile device that can be used to approximate a position. But there are some limitations to this solution as it demands that the mobile device is connected to the same network as the device triggering the ping. In this thesis, the access points used for capturing signals also provide a wireless area network for the mobile device to connect to.

2.2 Passive listening

Wi-Fi enabled devices still emit signals sporadically when not connected to any wireless local area network by sending out a type of signal known as a probe request.

The probe request are sent out when the mobile device needs to determine which access points are within range. The frequency at which the probe request is sent out is controlled by the mobile device operating system. Some devices only scan for wireless networks in passive scanning mode. When in passive scanning mode, the device only listens for ingoing signals from access points known as beacon frames. Devices that passively listen for nearby networks cannot be positioned using the system described in this thesis.

Multiple mobile devices were tested to see how often probe request are sent out, as it is the main signal generator for mobile devices that are not connected to the wireless network.

2.2 Evaluation of the tracking potential

A positioning approach without device-specific software still puts several requirements on the device that can be tracked. As the system is dependent on how often mobile devices transmit signals without any interaction from the user, several tests were made.

Since a very popular usage of positioning systems is to track mobile devices, a series of popular smart phones (including a tablet) were chosen as test devices.

The test was performed by setting up an access point in monitoring mode placed 2 meters from the device being tested. All packets received over a period of 5 minutes were recorded and some basic statistics were gathered from the data. Many popular smart phones perform several synchronization operations when before entering the sleep mode induced by locking the device, causing a burst of packets over the network. Because of this, the devices were locked for at least 10 seconds before each test.

Three properties were recorded and analyzed:

Number of packets
The number of outgoing packets transmitted by the device. This value is not guaranteed to be exact because of the possibility of packet loss over Wi-Fi.

Average silence
The average time between packets, excluding very short intervals. "Silence" is, for this usage, defined as all packet intervals greater than 1 second.

Longest silence
The longest interval between two transmitted packets, which in turn determines the longest delay between location updates. Again, packet loss may cause a slightly higher value than reality.

For each device, two tests were performed. One with the device connected to a network on the same channel as the monitoring access point, and one with the device disconnected from all nearby networks. In both cases, the Wi-Fi setting of the device was enabled.

Because of the already wide scope of the project, no attempts were made to determine the cause of packets transmitted by packet sniffing. Possible cases are probe requests made by the device, responses to pings, and acknowledgement packets from incoming requests from push-services to mention a few.

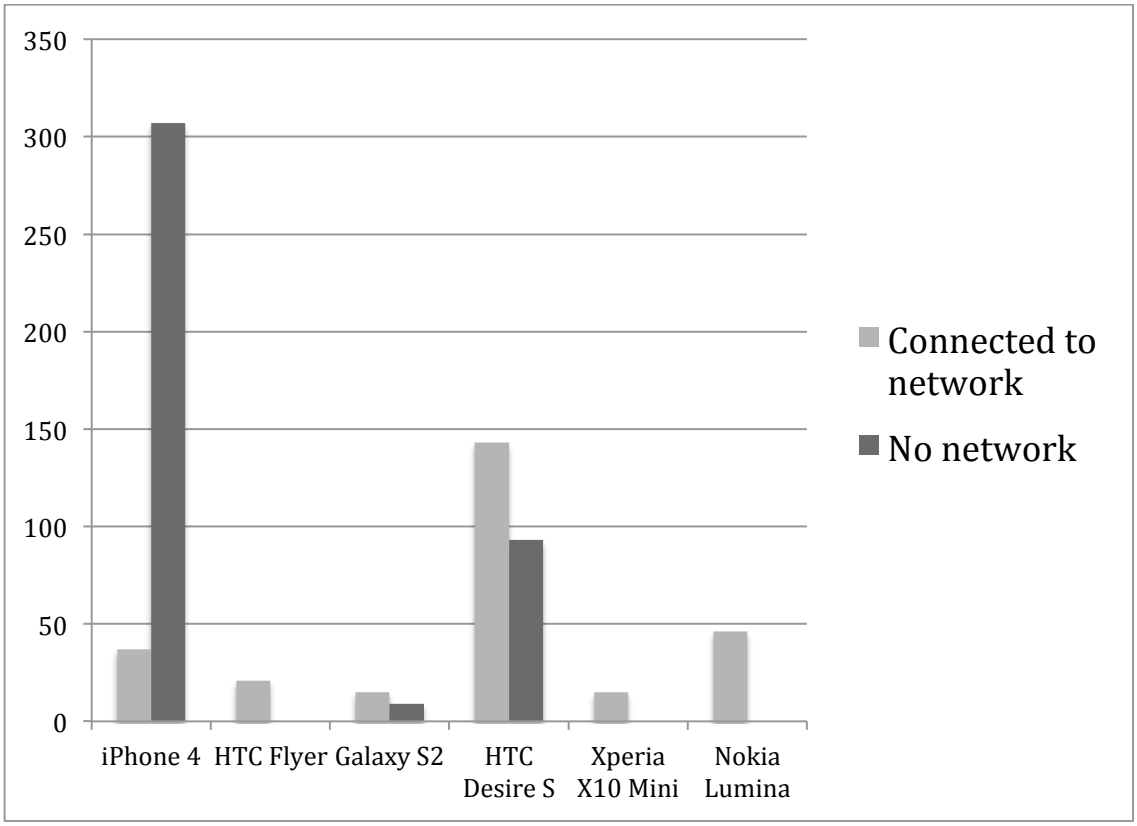


Fig 1: Number of packets transmitted over a 5 minute period

The tests resulted in varying results for each device. The iPhone 4 was the only device to transmit more packets when disconnected from the network. This is most likely probe requests sent by the device to probe for nearby networks. When connected to an access point the need for probe requests diminishes, and with it the number of packets transmitted.

Another interesting result is the lack of transmissions from several devices when disconnected from the network. The HTC Flyer, Xperia X10 Mini and Nokia Lumina stopped transmitting when locked and disconnected from the network. A common property of these devices was found and is likely the cause of the lack of transmissions: Neither of the devices had background data enabled. Background data, more commonly known on mobile devices as "push", are services that passively poll a web service for updates data, resulting in updates being almost instantly synchronized with the devices. 'Updates' can be anything the service is interested in: Incoming messages, social network posts or new RSS entries to mention a few.

Default settings were used for all devices and no attempts were made to create identical behavior for all devices by, for example, having the same push and networking settings on all devices. This was a conscious decision based on the fact that non-power users rarely change the default settings. Changing the network settings would not provide us with the every day behavior of the devices that we desired.

An observation was made that many popular smart phones only seem to probe for new networks passively by listening for beacon frames transmitted by nearby access points. These smart phones only transmit while locked if background data is enabled, in which case transmissions are made to synchronize data with online servers. To confirm this observation, a second test was made with background data enabled on the relevant devices. The test resulted in packets being transmitted as expected, confirming the initial hypothesis.

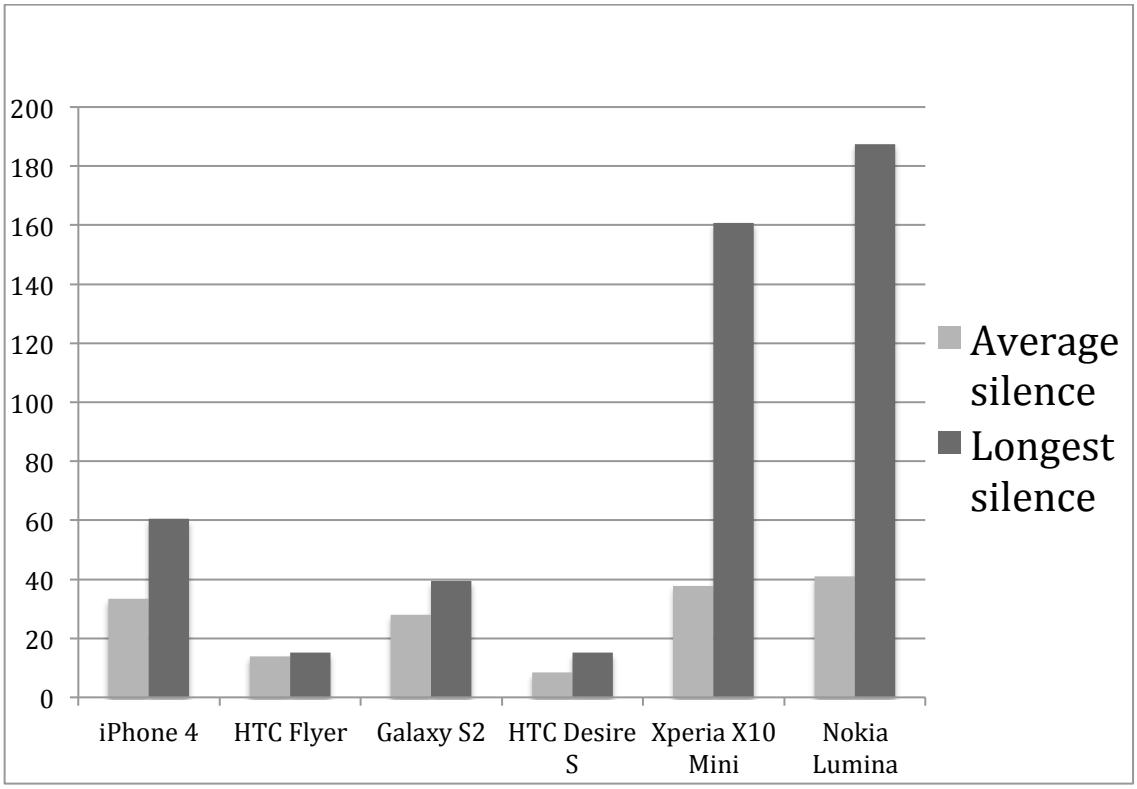


Fig 2: Transmission intervals over a 5 minute period. Device connected to network.

The test results when measuring transmission silences were, as with the number of packets, varied for each device. The most important observation made was that most popular smart phones transmit with long intervals when locked with no interaction from the user. Most devices had an average transmission interval greater than 20 seconds, which impacts the potential to track these devices greatly.

A combination of low transmission intervals and a high number of packets is required for good accuracy when tracking devices passively, without pinging or custom software on the device. The conclusion of the tests performed is that it is perfectly possible to passively track most popular smart phones, including the iPhone, without the need to induce additional traffic from the device. The accuracy and update rate of the position is heavily affected by the long transmit intervals of locked devices.

To increase accuracy, all tracked devices should be connected to a wireless network on the same channel as the monitoring access points used to position the device. This can be archived by broadcasting a Service Set Identifier (SSID), which in turn the tracked devices can connect to.

3 System design

Positioning directly on the device by gathering signal strengths to nearby access points has the advantage of having a very simple system architecture. Receiving sensor data, updating the model and positioning can all be done on the same hardware as the software has no need to be scalable. The device only keeps track of its own position.

When creating a platform-independent system, a more advanced architecture is required. In that scenario, the device being located (usually a mobile phone) has no software installed and only participates by periodically sending packets over Wi-Fi. The requirement of periodically sending packets is satisfied by most Wi-Fi enabled devices, including mobile phones, as probe requests are sent with a set interval to probe for new networks in the area.

As no additional software can be added to the device, the handling of sensor data and positioning must be performed externally on a server. In such a system, signals emitted by device are received by the nearby Wi-Fi access points and forwarded to a central server, which then processes the data and provides access to the location data generated. A mobile phone in the area can then simply access the location data through it's browser or an app by querying the server.

Even a relatively small number of devices generate a large amount of data when every single packet is considered. An iPhone 4 generates on average 60 packets per minute when not connected to a network. (See 2.2 - Evaluation of the tracking potential). Imagine a theoretical site with 5000 visitors a day with an average visit of 60 minutes. Even if only half of the visitors carried a mobile phone, this would result in 9 million packets a day.

A platform-independent system should be capable of handling multiple sites with any number of mobile phones and Wi-Fi access points. With this in mind, the system created during this thesis project was designed with scalability has a high priority. The focus of this thesis is designing a load balancer used to distribute responsibility for sites and devices to positioning servers that then handle sensor data and positioning.

The location data produced by the system can be accessed by either a device or a server by making a request to the responsible positioning server. This enables a huge amount of services to be built upon the positioning system.

3.1 Components

Two basic activity flows were identified during the system design phase.

1. The sensor nodes (Wi-Fi access points that gather signal strengths) must gather periodic data with the signal strength to devices within range and forward this to the positioning servers.

2. Applications that use the positioning system must have a means to access the location data produced by the positioning servers. Working under the assumption that the amount of sites with sensor nodes installed can become too great for a single server to handle, a means to delegate the incoming sensor data and application requests is needed.

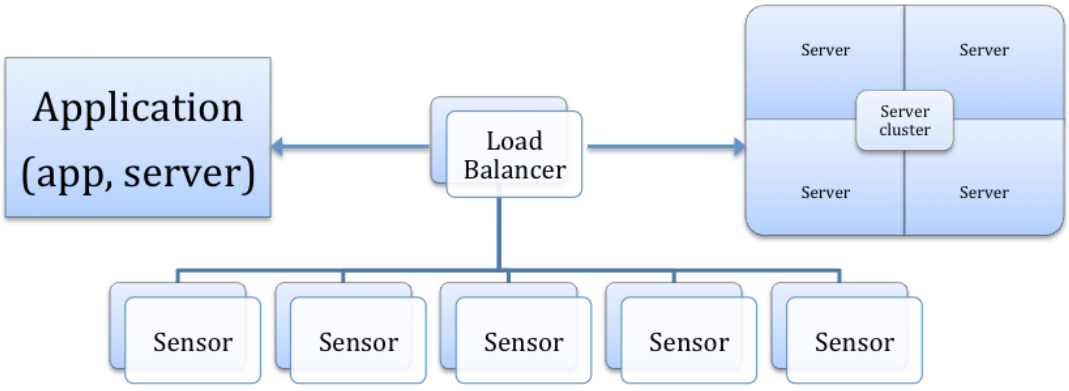


Fig 1. The system architecture

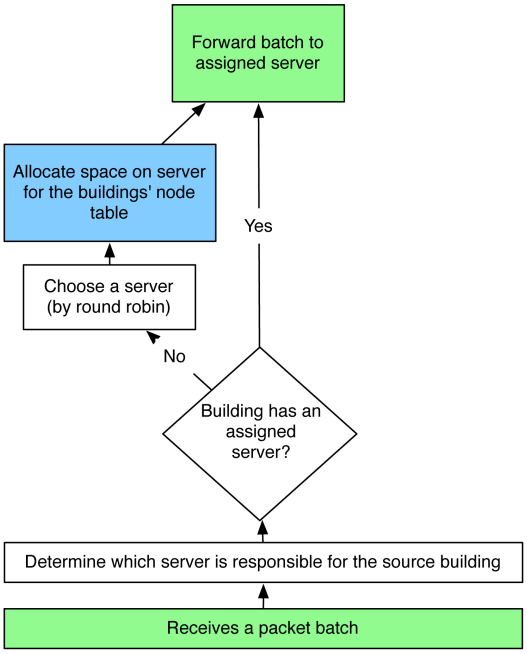
The load balancer acts as a gateway into the server cluster that handles sensor data and positioning requests. As new sites are added, the load balancer assigns the site to a positioning server in the cluster. Sensor data received from nodes in this site is then forwarded to the assigned server. Application requests to locate devices in the site are made to the load balancer, which then forwards the request to the assigned server.

Load balancing can be achieved in different ways. One approach is using load-balancing properties of the Domain Name System. The DNS Round Robin technique keeps a pool of IP addresses bound to a domain name. Upon receiving a DNS request, it hands over one of the IP addresses from the pool in a round robin manner [9]. This approach has some major issues when used with the architecture explained in this report. Firstly, the DNS server does not keep track of online servers in this IP pool. There are third parties software to handle this, but globally updating a DNS can take from several hours to days. Another issue is the process of adding a new server. A new public IP address would have to be obtained and added to all the IP pools on the DNS servers involved.

To avoid the issues of using DNS, a custom software load balancing approach was used.

3.2 The load balancer

Sensor data generated by the positioning nodes installed on the site are received as batches of received signal strength indicators (RSSI) to nearby devices. The positioning nodes are configured to send this data periodically to the load balancer.



When the load balancer receives a new batch of sensor data, it determines which server is currently assigned to the site and forwards the data. If no server has been assigned, the site-relevant data is sent to the server in an assignment request. This procedure creates a highly dynamic architecture. If the connection between the load balancer and one of the servers is lost, the load balancer simply considers all assignments to this server invalid. When a new batch is received from one of the dead server's sites, the load balancer handles this like it would any new site and a new server is assigned to the site.

As the load balancer is the entity responsible for keeping track of sites and servers, it needs to be easily configurable. Information of each site using the system must be available to the load balancer as well as the internal IP address of all positioning servers in the cluster.

The amount of positioning nodes a load balancer can handle is mainly determined by how fast it can handle and forward incoming sensor data. Each node sends a new batch on average every 200 ms and the assigned server of each batch must be quickly determined upon arrival.

3.2.1 Assignment

By applying a subnet mask to the IP of the positioning node, the site can quickly be determined. The subnet address of the node is then used to perform a hash table lookup to find the currently assigned server.

If no such entry is found, a new server must be assigned. There are many possible techniques for choosing a new server. If the load of each incoming request is roughly the same a suitable approach is Round Robin, where the load balancer keeps a list of all available servers and simply picks the next server in the list each time a request is received [8].

A different, more advanced, approach takes several factors into account such as the current workload of each server [8]. The load of each server can be estimated by counting the number of sites currently assigned to the server. The number of positioning nodes and/or devices of each site can also be used as weights when determining the load of a server.
A very simple approach was used in the prototype created during this thesis. A new server is selected using Round Robin and the selected server is sent an assignment request. The server then approves or denies the request depending on its current load. If the server is overburdened the load balancer simply continues the Round Robin and selects the next server in the list until a suitable server is found.

3.3 The positioning servers

The positioning servers are responsible for a set amount of sites. Each server handles incoming sensor data and can respond to positioning requests from applications.

Once a site has been assigned to a server, all incoming signal data batches are forwarded directly to the server. The server then unbatches the packets and updates the model accordingly. How the received signal data can be used to locate the source device is discussed briefly in this paper, but is not entirely within the scope of the thesis.

In the prototype created, the received signal data is averaged over time and stored on the server. The server makes no attempt to locate the device unless a request is made from an application using the system.

3.3.1 Receiving an assignment

When the server receives an assignment request from the load balancer it either denies the request (because of too high workload) or acknowledges it and allocates space for the new site. The site-specific data, such as node information and floors, is received from the load balancer. Device-specific data is dynamically allocated as new devices are discovered in incoming batches.

To ensure a stable system, the positioning servers periodically send an "I'm alive"-message to the load balancer over UDP. If the connection is lost, all sites currently assigned to the server is assigned to a new server.

3.3.2 Updating the model

The system architecture explained in this paper makes platform-independent positioning possible while keeping the positioning algorithms loosely coupled to the under laying system. Most existing methods used to locate devices from signal strength data can be used in conjunction with the system, as long as the MAC address of the mobile device (used as unique identifier by our system) can be acquired. As the platform-independent solution was the focus of this thesis only the basics of indoor positioning is discussed, as well as the most popular positioning methods.

3.4 Indoor Positioning

The signal strength between devices and access points in a building are noisy and heavily affected by the environment. A single wall can have great impact on the signal strength value. Water also has a resonance frequency of 2.4 GHz and since the human body contains over 70% water, people in the building become moving signal absorbers [11].

These factors turn the indoor positioning problem into a matter of determining an appropriate signal propagation model. An exception is the fingerprinting method, which avoids this problem by learning the environment manually, explained in 3.4.1.

The height of the access points also have great impact on the signal strength values received. By placing the receivers a few meters higher than the normal walking height (preferably the roof) the problem of people absorbing the signal can be avoided [10].

4 Positioning methods

Many techniques exist for determining the position of a device using the ingoing or outgoing signal strength of the device. The router-based approach proposed in this thesis measures the outgoing signal strength of the device. Most, if not all, of these techniques fall in one of two categories: Trilateration or fingerprinting [6].

The method of trilateration attempts to determine the distance between the sender and receiver using a propagation model based on the characteristics of the signal and the environment. This distance information, combined with previous knowledge of the exact locations of each access point, can then be used to compute the location of the device.

The second approach, fingerprinting, consists of two phases: The offline learning phase and the positioning phase. During the learning phase, the signal strength to each access point is measured from a predefined set of locations. The information gathered in this phase is used to construct a map of the signal strength received at each location [6][7]. In the (live) positioning phase, gathered data is compared to the map created during the learning phase.

By comparing the received signal strengths to each access point with the stored data from the learning phase, one of the predefined locations are chosen as the most probable location. The most simple approach is to choose the location that minimizes the euclidean distance of the signal strengths of the live sample and the learned data [7].

4.1 Simulating signal data

A simulation tool was developed to assist in testing how the various components handled different situations. By simulating an environment where the access points brake down seemingly at random, the simulation tool helps evaluate and improve the system. The simulation tool creates signal batches programmatically and sends these to the load balancer with static site and access point identifiers, effectively simulating any number of sites and access points.

A big weakness of simulating packets in this way is the unnatural signal strength values. To create a more natural simulated environment, randomized noise (using a normal distribution) was added to the signal values before the simulated packets were batched and sent to the load balancer. Despite adding randomized noise, signal values created programmatically poorly match a real environment. Real signals are affected by many factors such as reflection, multi-path phenomena and moving objects that absorb the signal.

Due to the problems of creating a simulated natural environment for the system, a secondary method of testing was used. By recording the received packets during live experiments, a replay file was created which could be played back by a replay tool. The replay tool parses the replay file, reconstructs the packets in the replay file and sends these to the load balancer at the specified timestamps in the replay file.

5 Implementing the positioning node network

In this chapter we describe the topics of implementing a positioning node network as described in chapter 4. As each Wi-Fi access point (AP) must be able to forward the packets to an external positioning server, they must all be connected to some sort of network. As each of the access points have a wireless network controller they can offer a wireless network for the mobile device. But this wireless network also allows them to connect to each other, creating an environment that does not require wired connections between the access points. In this thesis, the network was implemented using Wireless Distribution System, commonly known as WDS. WDS is a static configuration that states what other access point an access point is to be connected to. As an access point is now forwarding data wirelessly and offering a wireless network to mobile devices it is no longer in monitoring mode, rather it is in the WDS mode

5.1 Software development for embedded devices

With the demand on the embedded device to capture all incoming signals as well as determine the arriving signal strength of each signal, software performing these tasks is needed. Developing software for an embedded device compared to developing for a major platform such as Windows or Mac OS X requires some additional steps. As the embedded devices (in this case access points) are small and designed for routing traffic rather than to be a developing machine, they suffer from limited memory and CPU. Developing tools and compilers take a fair amount of disk space, more then most of the access points can offer. For this reason, it is preferred that the development of software is not done on the device itself but rather on a separate machine.

Most compilers compile programs to be executable on the development machine. But for the software created for the access points a concept called cross compiling was used. Cross compiling allows one CPU architecture to compile for a different CPU architecture, allowing the development machine to create a software that can be run on the access points.

Much like on an ordinary PC, the software cannot run without a operating system that communicates with the hardware. In this thesis a modifiable operating system called OpenWrt was used.

5.1.1 OpenWrt

OpenWrt provides a fully writable file system that can be customized through the usage of packages. OpenWrt own description of their operating system:

"OpenWrt is a highly extensible GNU/Linux distribution for embedded devices. Unlike many other distributions for these routers, OpenWrt is built from the ground up to be a fully-featured, easily modifiable operating system for your routers. In practice, this means that you can have all features you need with none of the bloat, powered by Linux kernel that's more recent than most other distributions.".[12]

There are a few other choices available such as DD-Wrt or Tomato. The choice of OpenWrt was made because of its open architectures. This makes it easier to inspect arriving packets to determine the RSSI value. OpenWrt also uses a powerful tool called Buildroot that makes OpenWrt stay bleeding edge compared to the other options. Last, but not least, OpenWrt have by far the largest community of these three candidates. [12]

5.1.2 Development Issues

Tools needed to develop a firmware image for an embedded device are somewhat specific. These tools are not included in any common operating system. Installing Linux on a development machine as well as a compiler and every development tool offered would still not suffice to produce a firmware image. The firmware image stores read-only memory and program code inside the image itself. The reason for this is that the embedded device is likely to be incompatible with the development machine as it represents an entirely new hardware platform. This issue is solved by cross compiling. As previously mentioned, cross compiling allows one CPU architecture to compile for a different CPU architecture. Using cross compiling, combined with tools such as Buildroot, a complete firmware image can be created for a targeted embedded device. The firmware image can then be installed on the embedded device using a web interface supplied by the device vendor. [13]

The second issue raised when developing for embedded devices is how to get the software to the embedded device itself. OpenWrt offers by default the possibility of using programs such as secure copy, more commonly known as SCP. Making it easy to transfer the executable to the embedded device.

5.1.3 Building an image

As mentioned above, OpenWrt uses a tool called Buildroot. Buildroot is essentially a set of makefiles or rules that controls the configuration and compiling process of a firmware image. Buildroot downloads all sources needed, including sources for the cross compiler and makes sure everything is up to date. Normally, the cross compiler created for an access point is supplied by the access point vendor, rather than that the developer creates its own. While this saves time, the binary containing the cross compiler is likely to be outdated and not use the latest set of tools [13]. As Buildroot downloads the latest sources for the cross compiler and kernel automatically it keeps OpenWrt bleeding edge.

5.2 Listening to signals

The task dedicated to the embedded devices in this thesis is to listen to arriving signals and determine the transmitter of the signals along with the arriving signal strength. To be able to both send and listen to signals the access points has a component called the Wireless Network Interface Controller (WNIC). WNIC can operate in six different modes: Master (acting as an access point for other devices), managed (more commonly known as client or station in a wireless network), repeater, mesh, ad-hoc and monitor mode. The mode most frequently discussed in this thesis is the monitor mode.

While the WNIC operates in monitoring mode, no frames are transmitted. However, while operating in monitoring mode the WNIC accepts all frames, including those that are not dedicated to the monitoring device and would usually be dropped. As wireless network frames are essentially packets encapsulated with headers used when transmitting over the wireless network, packet capture software that works on Ethernet can with a small extension also work on wireless networks.

5.2.1 Capturing packets

Packet capturing software on Ethernet based network works as follows. When a frame is received at the network card it checks that the destination MAC address matches the MAC address of the network card. If a match is found it generates an interrupt request that is handled by the network card driver. The network card driver then handles the interrupt request and timestamps the arrival time of the packet. Normally when there is no packet capture software running, the packet is forwarded to the protocol stack. But when a packet capture software is running it is also copied and sent to the packet filter. What the packet filter accepts is user defined and by default it accepts all packets. This process is also performed for transmitted packets. [14]

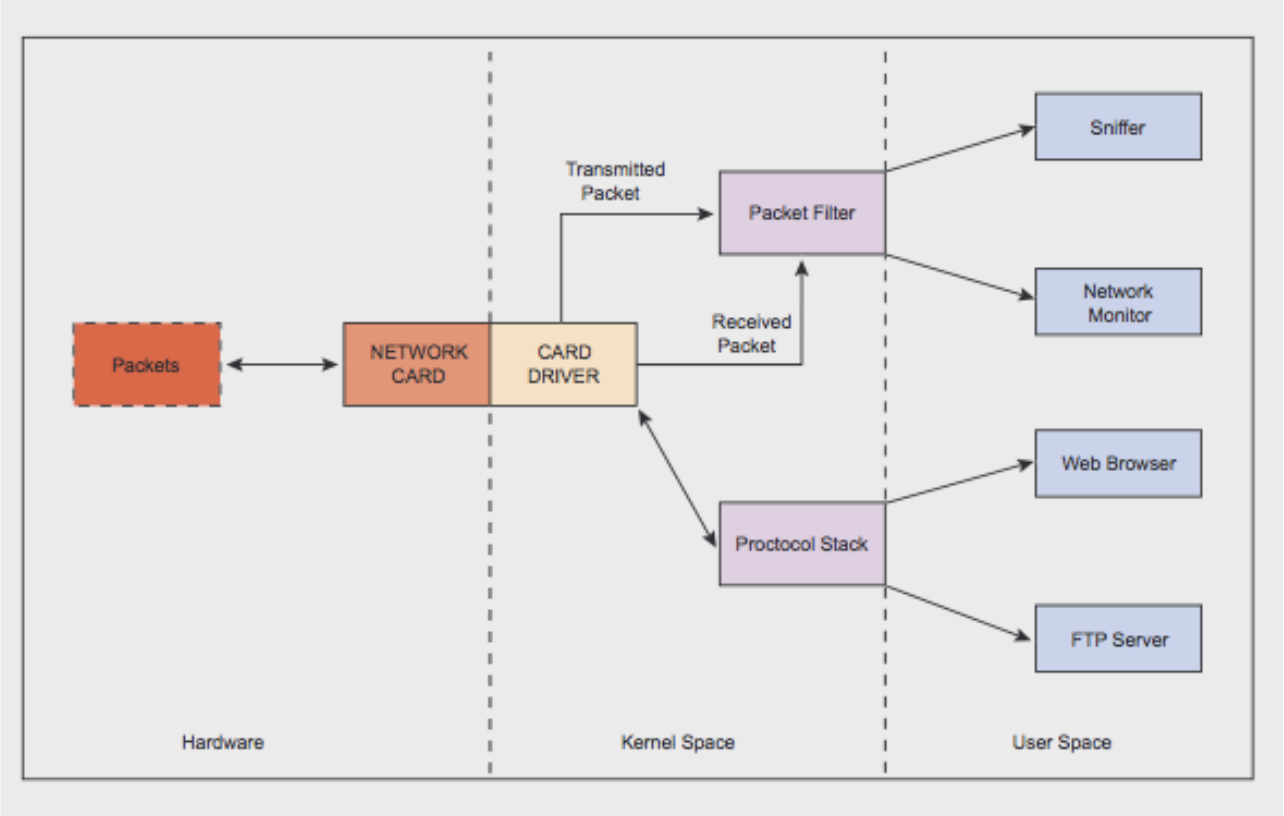


Fig 3: Monitoring packets [14]

Frames captured contain all data needed to determine the RSSI value and transmitter of the signal. But this data must be parsed out of headers that have been applied around the data for each layer in the TCP/IP stack.

5.2.2 Monitoring software

The tasks of the monitoring software deployed on the embedded devices is to capture all arriving signals, measure the signal strength of each arriving signal and determine the source of the signal. The following section explains how these tasks are accomplished.

In computer networking there are several layers that are encapsulated around the actual data being sent. These layers main purpose is to guide the data to its destination. The layers also offer features such as encryption and authentication. This model is commonly known as the OSI model but in this thesis we will refer to the more compact and simple version called the TCP/IP stack.

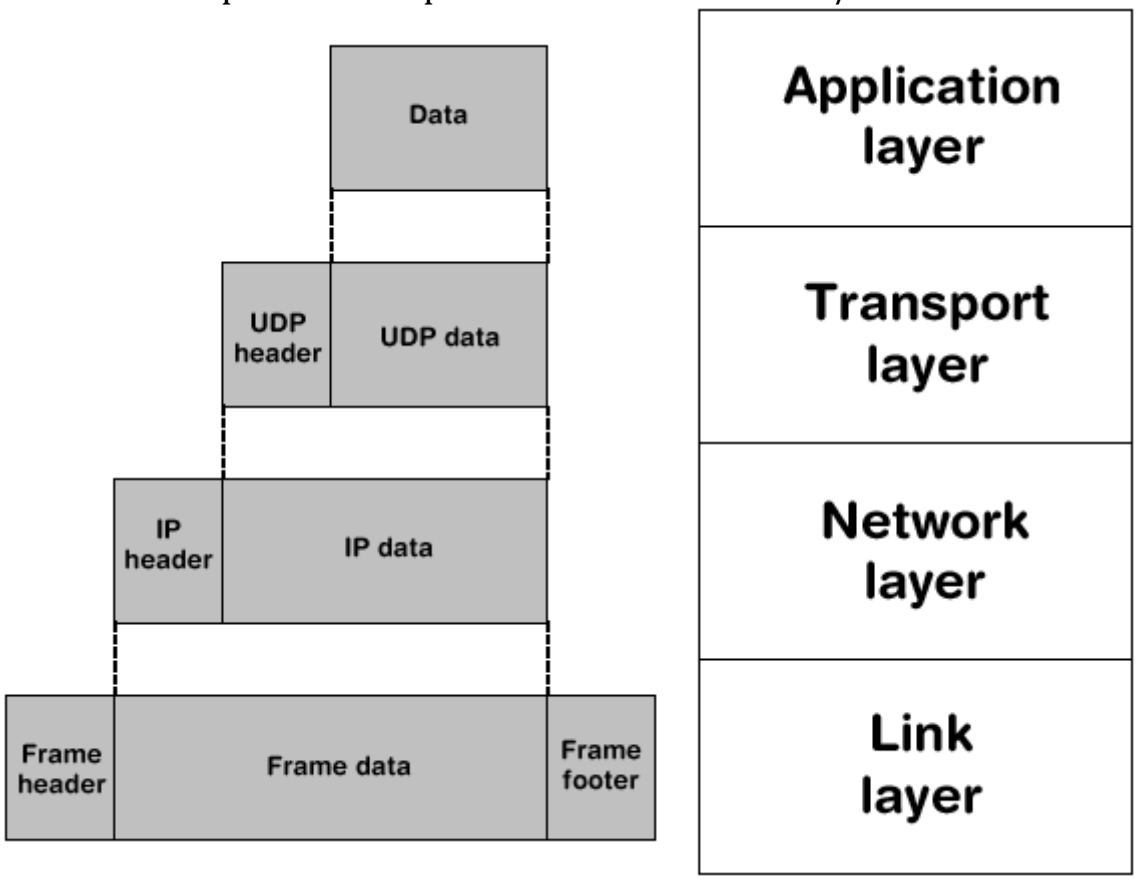


Fig 4: Headers and layers of the TCP/IP stack

The lowest layer in the TCP/IP stack is the Link layer [15]. The Link layer contains information used when sending frames over the wireless network, including information about the receiver and transmitter. A frame that is to be sent over the wireless network will be encapsulated by a frame header and a frame footer.

The information regarding the packet that needs to be extracted on the access points lie in the header, which is why the focus of this section is mainly on the frame header. The definition of the frame header is ambiguous. In some cases it includes both the Physical Layer Convergence Protocol header as well as the Medium Access Control header, while in other cases it contains only the Medium Access Control header.

5.2.3 Medium Access Control header

To be able to identify the transmitter of a signal there is a unique identifier for each Network Interface Card called Medium Access Control Address or MAC address [16]. The MAC address of a sent signal is found in the MAC header. The MAC header contains four address fields, and the difficulty of extracting the MAC address lies in determining which of these four fields contain the MAC address. The content of the fields is controlled by two bits, commonly referred to as ToDS and FromDS [17].

In the 802.11 protocol the MAC headers format depends on what type of packet it is handling. There are three types of packets: Data packets, network management packets and control packets [17].

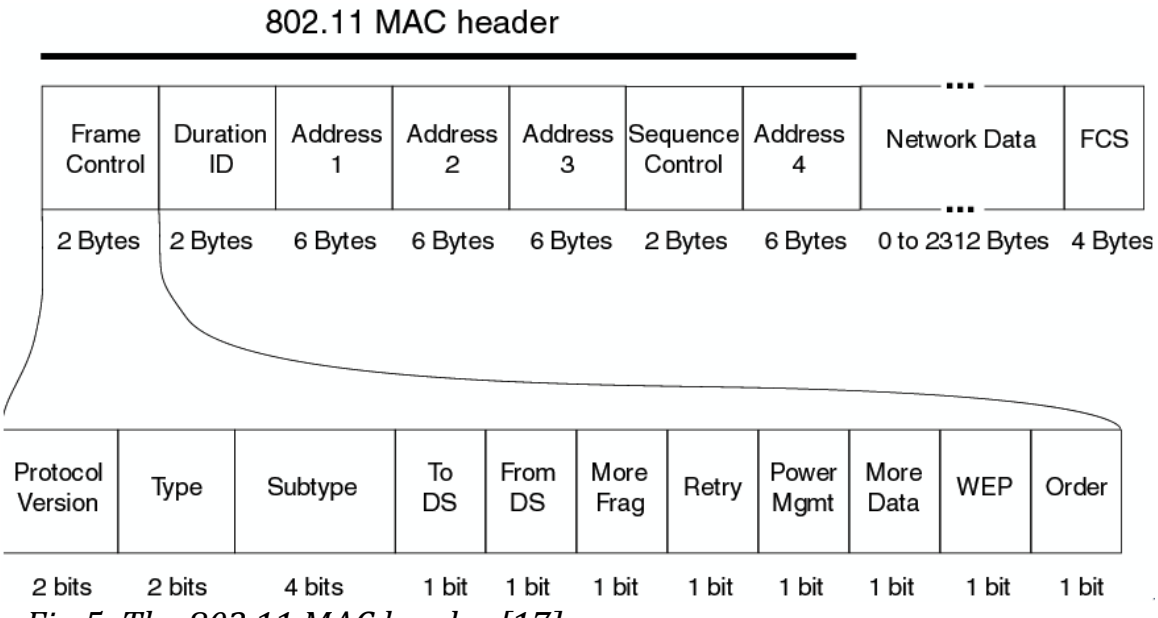


Fig 5: The 802.11 MAC header [17]

The difference between the three packet types is that the management and the control packets only have the first three out of four address fields shown in fig 5. One of the tasks assigned to the monitoring software is to determine the transmitter of the frame.

This information is stored in one of the address fields. However the ToDS and FromDS (To- and From Distributed System) bits that controls routing in the wireless network also changes the content of the address fields [17]. Depending on if the sender or receiver is inside or outside the network, the ToDS and FromDS bits assume different values, as specified in fig 6.

To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	DA	SA	BSSID	N/A
0	1	DA	BSSID	SA	N/A
1	0	BSSID	SA	DA	N/A
1	1	RA	TA	DA	SA

Fig 6: FromDS and ToDS address location table

Abbreviations
DA - Destination Address
SA - Source Address
RA - Receiving Address
TA - Transmitting Address
BSSID - Base Service Set Identifier

The sender of the signal is either the Transmitting Address or the Source Address and it is these two that the monitoring software parses out and binds together with the RSSI value.

5.2.4 Obtaining the received signal strength indicator (RSSI)

The Medium Access Control header itself does not contain information about the received signal strength. The signal strength is measured by the network card driver. The received signals are first measured in milliWatts and then translated to a logarithmic scale. A large issue with this none-standardized implementation is that network card manufacturers end up using different scales in the RSSI spectrum.

To obtain the signal strength data that the network card driver have measured, the Packet Capture Library (Libpcap) supplies the developer of a monitoring software with a extra layer encapsulated around the captured frame called the Radiotap header [18].

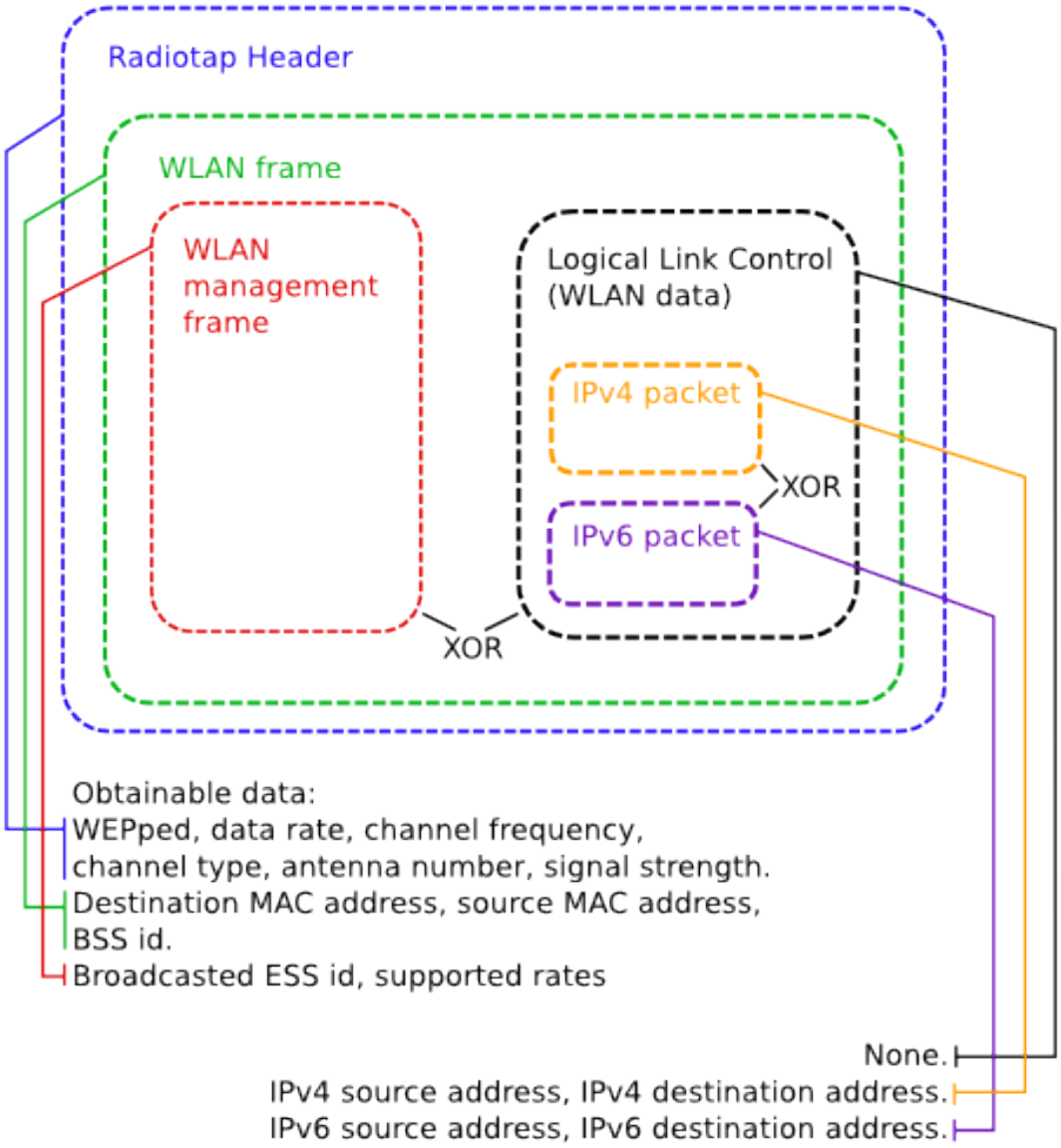


Fig 6: Packet data scheme [19]

The radiotap header was created to supply user space applications with additional information about transmitted and received frames [18]. As shown in fig 6 the radiotap header not only contains the signal strength value, but also the channel type and antenna number among other things that is known to the network card driver.

5.3 Forwarding gathered data

As the embedded devices are limited by their CPU and memory it is not ideal to perform positioning algorithms on them. Instead, an external positioning server is used and the monitoring software is designed to forward the obtained information to this positioning server.

The software installed on the embedded devices sends information about received signals strength, what device that sent the captured signal and when it arrived. As previously mentioned in this thesis. The network card driver timestamps each received frame with the arrival time of a signal, which is found in the radiotap header. Let us for the sake of simplicity; call this bundled information a packet.

A problem with forwarding the assembled data about received signal strength, source device and arrival time (packet), is that it creates an infinite loop. Consider that each monitoring device listen to signals, captures them, measures the signal strength and finally it timestamps the signals time of arrival. Then it forward the information about this signal to a remote server through the wireless network, which in turn creates one or multiple signals of its own. Signals that other monitoring devices captures and transfer to same remote server. Creating an infinite loop for each captured packet that is being forwarded.

Two solutions to the issue of an infinite loop being created would be either to filter out packets from other monitoring devices or batch several packets together. By batching several packets together a monitoring device would only generate one packet for several captured signals. The amount of captured signals would exceed the amount of sent packets from monitoring devices and so the infinite loop would be avoided.

By batching packets, not only is the infinite loop problem solved. But the load on the remote server and the network is lowered as well. As each network packet contains a certain amount of overhead, a bundled packet contains the same amount. Meaning that for each packet (bundled data about a signal) that can be put into the batch, the overhead is spared. And less separate packets for the remote server to unpack (handle in the different network layers of the TCP/IP stack).

5.4 Evaluation

In this thesis, we have suggested a system that can gather signal data from mobile phones only by passive listening, removing the need for on-device software. By distributing an open Wi-Fi network for the mobile phones to connect to, the amount of traffic generated by the devices can be increased. The signal strength information gathered and stored by the system can be used to locate the devices emitting the signals by for example trilateration or RSSI fingerprinting (See chapter 4). The software used by the Wi-Fi access points to gather and forward signal data can be installed on hardware already available on the market.

Using a load balancer backed by a server cluster of variable size the system can scale to handle a large amount of sites (where a site is an area covered by the custom Wi-Fi access points used to gather signal data for the system).

The load balancer discussed can handle a very large amount of work while still distributing work in an efficient manner when there are only a few servers in the cluster.

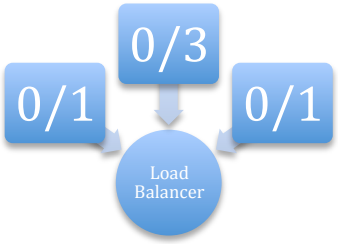


Fig 7: Initial state of the system. Numbers indicate the sites assigned to each server out of the total number of sites that can be handled.

Using the assignment strategies discussed in 3.2, fig 7 shows the state of the system using 3 work servers when initially set up. Each server is dormant with no sites assigned. When a batch of packets is received from the first site, the load balancer picks the first server using round robin and asks the server if it can be assigned a new site. The server has no load and approves the request. Future batches from this site are forwarded to the designated server.

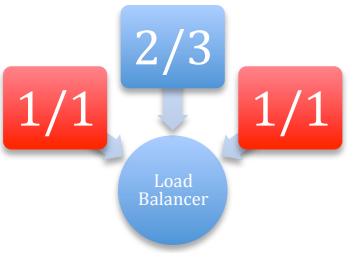


Fig 8: State of the system after 4 sites have been added.

Two more sites appear and the load balancer continues to select servers using round robin. When a 4th site is introduced to the load balancer, the first server will again be asked for site assignment. This time, the server determines that the load of the previous site is too great and denies the request. Continuing the round robin, the load balancer requests that the next server handles the site and the request is approved.

6 Conclusion

A platform-independent indoor positioning system can be implemented using existing popular low-cost hardware on the market. The approach explained in this thesis has many advantages over existing on-device solutions, at the cost of ease of installation. Once the hardware has been installed on site, the configuration required is minimal.

The system design discussed in this report allows a highly scalable and stable system that can handle any number of networks and devices given that additional servers are added to the cluster as the workload is increased. The location of tracked devices can easily be acquired by making a request to the server, allowing new uses of indoor positioning that are not possible with an on-device software based solution.

While the location accuracy of devices that are not on the network is lower than existing solutions due to the high transmission interval of most devices, the accuracy can be greatly increased when devices are connected to networks on the same channel as the positioning network through pinging.

6.1 Future work

This thesis discusses the problems of designing and implementing the basic architecture of a platform-independent indoor positioning system. There are many improvements that can be made to increase accuracy, ease of installation and performance.

There are many improvements that can be made regardless of positioning method. A Bayesian filter (such as the Kalman filter) can be applied to the positioning data acquired by trilateration or fingerprinting. By letting previous position and direction affect the probability of new locations, not only can the tracking be made a lot smoother but the accuracy can also be greatly increased. Other factors can also be taken into account, such as statistics from other users. For example, if a common pattern among users is to walk directly from the entrance to the reception the probability of locations along this path can be increased for future users.

There are also several techniques to increase accuracy that require software on the device. By using device sensors such as compass (direction) and accelerometer (acceleration and speed) to affect the probability distribution of the tracking filter, a more accurate location can be achieved even when signal data is sparse.

In this thesis, the wireless local area network provided by the access points not only serves the purpose of positioning the mobile devices. The wireless local area network also allows the access points to connect with each other devoid of any Ethernet backbone. This is accomplished by using an architecture known as Wireless Distribution System.

While the Wireless Distribution System covers the needs of this thesis it would become a bottleneck and limitation in a larger implementation of an entirely wireless environment. The first issue in a large implementation is the static routing of the Wireless Distribution System. When configured, it chooses a specific access point to interconnect with which easily creates a chain of single points of failure. Secondly the throughput of each interconnected client in this chain is halved [15].

A future implementation of this system could use Wireless Network Mesh Protocol. The mesh network implementation solves the routing issue that the Wireless Distribution System suffers from in larger networks by using the routing protocol known as Optimized Link State Routing [15]. The Optimized Link State Routing protocol consistently sends packet over the network to determine network topology and to create a routing table [15]. By allowing each of the access points to create their own routing table dynamically in comparison with the more static Wireless Distribution System, the mesh network implementation is able to self-heal if an access points goes offline.

6.2 Related work

Like the system described in this thesis, Ericsson Lab's positioning API also use a central server for computing location data from signal data. However, their system differs on some important points. The system described in this thesis puts no requirements on the mobile phone being located and instead moves all responsibility to external servers. Ericsson Lab's solution requires that the mobile phone gathers signal data from the Wi-Fi access points. The data is then transmitted to their positioning server which, combined with previously uploaded maps of the location of each access point, can compute an estimated location. This solution greatly simplifies system architecture at the cost of requiring special software on the mobile phone being located.

6.3 Work report

This thesis was written in collaboration by two students and covered two major topics on the same subject. The research and tests performed were done in collaboration between the two. Collaborative discussions and findings are covered in the introduction and basic concept chapters. Fredrik Gadnell focused his research on the design of the system and wrote chapter 3, 'System Design'. Robin Broberg approached the subject from a hardware and network perspective and wrote chapter 5, 'Implementing the node network'.

7 References

[1] Elliott D. Kaplan, Christopher J. Hegarty. Understanding GPS: Principles and Applications, 2006

[2] Russell C. Brinker and Roy Minnick. The surveying handbook-2nd ed, 1994

[3] Tim Burks (2013-11-20). No way to determine WiFi SSID/Network name via SDK. Available: http://openradar.appspot.com/6407431 [2013-11-21]

[4] Apple QA on Location Data. Available: http://www.apple.com/pr/library/2011/04/27Apple-Q-A-on-Location-Data.html [2013-11-21]

[5] Ericsson Labs Indoor Maps and Positioning. Available: https://labs.ericsson.com/blog/ericsson-labs-2011-highlights [2013-11-21]

[6] B. Li1, J. Salter2, A, Dempster1 and C. Rizos. Indoor positioning techniques based on Wireless LAN. 1st IEEE Int. Conf. on Wireless Broadband & Ultra Wideband Communications, Sydney, Australia, 13-16 March, paper

[7] P. Prasithsangare, P. Krishnamurthy and P.K. Chrysanthis, On Indoor Positioning Location With Wireless LANs, IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2002, 18-18 Sept, paper

[8] K.J. Salchow, Jr. Manager in Product Management at f5, Load Balancing 101: Nuts and Bolts, http://www.f5.com/pdf/white-papers/load-balancing101-wp.pdf

[9] S.I. Ulland, High-Level Load Balancing for Web Services, paper

[10] M. Helén, J. Latvala, H. Ikonen and J. Niittylahti, Using Calibration in RSSI-based Location Tracking System, International Conference on Convergence Information Technology, 2007, 21-23 Nov, paper

[11] G. Deak, K. Curran and J. Condell, Device-free Passive Localisation using RSSI-based Wireless Network Nodes, Jun 2010, paper

[12] OpenWRT. Available: http://openwrt.org [2013-11-21]

[13] OpenWRT Manual. Available: http://kamikaze.openwrt.org/docs/openwrt.html

[14] L.Garcia, Programming With Libpcap Sniffing the Network from our own Applications, http://recursos.aldabaknocking.com/libpcapHakin9LuisMartinGarcia.pdf

[15] R. Braden, Internet Engineering Task Force, Requirements for Internet Hosts -- Communication Layers, http://tools.ietf.org/html/rfc1122

[16] Introduction to Wireless Networking, WildPackets Inc, https://mypeek.wildpackets.com/elements/whitepapers/Intro_to_Wireless_Networking.pdf

[17] WLAN Packets and Protocols, WildPackets Inc, http://www.wildpackets.com/resources/compendium/wireless_lan/wlan_packets

[18] Radiotap Header Information, The Madwifi Project, http://madwifi-project.org/wiki/DevDocs/RadiotapHeader

[19] Image taken from http://www.di.unipi.it/~cornolti/iwtan/iwtan_reference.htm [2013-11-21]