NarrowBand IoT

INTRODUCTION AND INVESTIGATION

Henning Håkonsen

Thesis submitted for the degree of
Master in Network and system administration

60 credits

Department of Informatics

Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2018

NarrowBand IoT

INTRODUCTION AND INVESTIGATION

Henning Håkonsen

NarrowBand IoT

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Contents

List of Figures

List of Tables

Preface

Part I

Introduction

Chapter 1

Background

Today there are approximately 30 billion smart devices in the world. The growth forwards will be exponential and analysis predict that there will be 50 billion smart devices by 2020 [2]. There is a demand for a new technology which enables communication with sensors and other low powered devices. We will introduce you to a set of technologies suited for this communication, with an emphasis on NarrowBand IoT (NB IoT). We are at a point where IoT is becoming a part of our society which enhances our life experience.

1.1 Goal of the thesis

The goal of this thesis is to show you how IoT will become a part our lives. We will discuss IoT applications, upcoming IoT technologies and how they differ. The main topic of this thesis is NB IoT and how it performs in regard to its specification. There are several claims to this technology, battery lifetime over 10 years, indoor and underground coverage and low cost. Through a cooperation with Q-Free and Telenor we will be able to get hands on tests of NB IoT. The interesting part of this thesis is that we are the first to really test NB IoT and this gives us an opportunity to verify the specifications without to much interference. There are several papers on how NB IoT has great power saving features, referring to the 3GPP specification, but there are not many discussing how they can be achieved in practice. Several factors will influence the power usage, such as environmental changes, downtime in the network, network configurability and software complexity. We will show you how these factors impact the power consumption and try to give an overview of the best practice usage.

1.2 Motivation

Easily accessed information, connected to all parts of society is a huge motivator for Internet of Things (IoT). In the future all things will be connected to the Internet, enhancing our lives. The meaning of IoT is a group of physical devices, for example vehicles, monitoring systems, watches and so forth, forming a network. The complexity and possibilities with this kind of network are incredible. However, the path towards this goal has not been easy. There has been attempts of similar networks, but they usually are to fixed, expensive and the competition has been more of an obstacle than an advantage. With to many poor solutions the growth has been rather low compared to what we now see a start to. With several emerging technologies suited for low powered devices we expect a huge growth. Based on numbers from an article anticipating the growth of the population by The United Nations [3], Cisco has made an illustration of the expected growth of devices towards 2020 1.1.

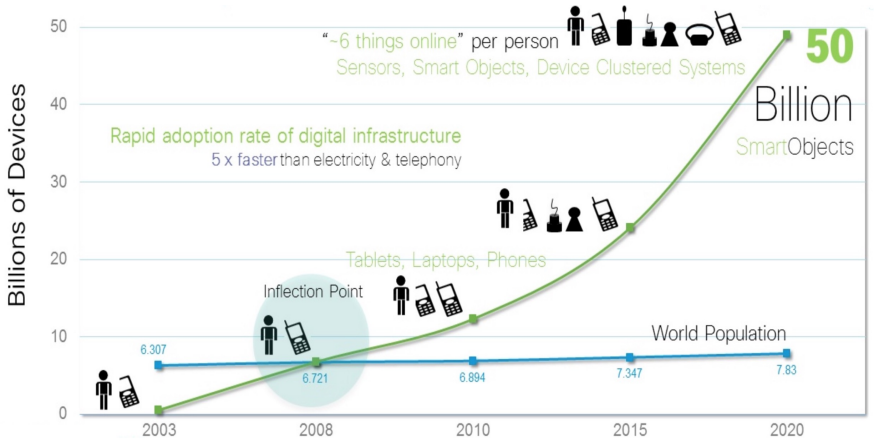

Figure 1.1: IoT Growth [2]

Currently we are at a point in time where it all comes down to how these Low Power Wide Area Networks (LPWAN) perform, which we will have a look at in this thesis. Up until now, IoT devices has communicated directly or indirectly with Internet, but in a rather complicated way. The typical scenario has been that the User Equipment (UE) communicate with a more complex device, often a wireless unit of some kind, and this device communicates with Internet over LTE or a wired connection. While this is an improvement and a step towards a future of connected things - the current mobile technology does not support the expected growth in this area.

The payload sent from IoT devices is usually small, typically 100bytes or less. However, the load of a LTE device on the current infrastructure, with all the signaling required, is too high. Hence, a simplified technology would allow less signaling and load on the telecoms infrastructure, while also enabling lowered monthly fees as a consequence. Price is always an

important factor when choosing a new product. The typical cost of a mobile subscription with LTE and 5-10GB of data per month cost between 30€ and 50€, supporting 50-100 devices. The cost of a subscription over NB IoT will according to Telenor and Q-Free be around 0.2€. In addition, hardware and installation costs are heavily reduced with NB IoT. The aim is that an NB IoT enabled device will cost around 60€. A similar LTE implementation would require a component that does LTE for embedded devices, which costs around 500€. A simple calculation[1.2] of the cost of 300 devices from a parking sensor use case shows that NB IoT easily outperforms the current solution. TODO - verifisere utregning

Subscription: Insert excel

In addition to lower cost, lower complexity there is a demand for increased coverage. Vehicles driving in vast areas, monitoring sensors position several meters under ground and devices located in packed cities are dependent of extreme coverage. NB IoT will add 20 dB to the link margin, giving a huge coverage increase. This will enable such devices to operate while holding the power usage to a minimum.

Another motivator for IoT is the expected easiness of the setup. With todays technology you have to use sensors communicating over bluetooth or Wi-Fi to a common gateway which is connected to the Internet. With a NB IoT device you will be able to connect directly to Internet, something many of us will be able to do without to much background experience. This will be an additional cause of growth in the number of IoT devices and is yet another reason why we need a new way of dealing with this technology.

The idea of some of the new Low Power Wide Area Networks (LPWAN) is to use the same hardware as LTE and add software to enable an IoT Platform. The most promising technology for sensor networks is NarrowBand IoT (NB IoT) and is enabled with a LTE core network as the bearing network. We will discuss how to implement NB IoT in a LTE network in section, 2.2 on page 20. This new technology should support an extreme amount of devices as well as them being in a secure environment. Security is a big concern when discussing IoT and we will elaborate on how this new network will provide security for the users in section, 2.2.2 on page 21. Another concern is bandwidth and power consumption as mentioned. The bandwidth is reduced to a minimum and the power consumption is one of the main focuses. In section 2.2 we will go into detail on this technology.

1.3 Current status

1.3.1 Internet

The main idea of the Internet we know today is quite similar to the past, but the scale and reach has outgrown what anyone thought could be achieved.

In 2015 Google's data centers achieved high speed transfers up to 1 Petabit. "According to Vahdat, that is enough bandwidth for more than 100,000 servers to exchange data at 10 Gbps each, or transmit all scanned contents of the Library of Congress in under one-tenth of a second"[20]. The reason for these data centers is the use of online resources. Offices, as well as home users require more bandwidth and reliability of the services provided from e.g. Google, VPNs, data storage and so forth. Internet is everywhere and will continue to grow the coming years. As we will discuss in a later section we are seeing signs of a future where literally everything comes online. This includes wearables, industry monitoring and management systems and this will hopefully engage in a smarter and healthier society.

1.3.2 Mobile networks

Norway's GSM network came online in 1993 [8] and Norway has pressured the current technology to reach higher standards from this time. The evolution of wireless radio technology has usually been focused on making it faster, while under prioritizing battery lifetime and simplicity. However, because this technology is used in mobile devices, it is for most users good news. LTE offers high Up Link (UL) and Down Link (DL) speeds and good coverage. One problem with LTE and older wireless technologies is that it consumes a lot of power. People are experiencing high speed transfers to their mobile devices, but at a cost draining their batteries. In the future, power efficiency has to be one of the main features of networks established. This is especially the case for sensor networks as these devices need to be connected to a mobile network. Such sensors are often established in remote locations without steady power. Using LTE or some other standard solution would give these devices short lifetime.

1.3.3 A closer look at LTE

Long-term Evolution (LTE) networks consists of some particular nodes for the network to connect to User Equipment (UE) and Internet. We call this structure Evolved Packet Core (EPC) [12]. In this section we will get a better look at some of the attributes of LTE and the parts which make the EPC network. We really need to understand the underlying network to get a grasp of the new technologies we will investigate. Some of the standards we will mention are using proprietary solutions, but most of them are using LTE as the main infrastructure. LTE was introduced in release 8 of the 3GPP in 2008. In a summation by Ericsson they state some facts about LTE from the initial release. LTE from release 8 supported 100Mbps DL and 50Mbps UL(Not peak speeds), reduced latency(down to 10ms) and was cost-effective to set up. [4] As of release 10 by 3GPP(year 2011) the speeds were at astonishing 1 Gbps DL and 500Mbps UL, however the network is not suited for low powered devices.

LTE is composed by 5 components which interact with each other. MME, HSS, S-GW and P-GW constitutes Evolved Packet Core (EPC). See outline of LTE topology in figure 1.2.

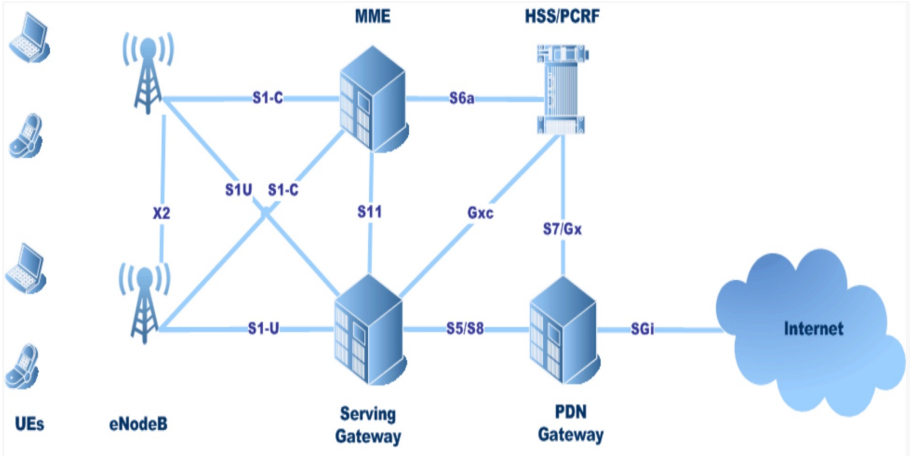

Figure 1.2: Evolved Packet Core [14]

1.3.4 EPC

MME

Mobile Management Entity (MME) is the main provider of signaling in LTE. MME is connected to eNB, HSS and S-GW, through S1-MME, S6 and S11 interfaces. It is in charge of authentication in cooperation with HSS which contains subscriber information, terminal-to-network negotiation and is a part of transition from LTE to 2G/3G.

HSS

Home Subscriber Server (HSS) is the main database for information in the networks. It is a joint service originating from Home Location Register (HLR) and Authentication Center (AuC). HSS keeps information about subscriber information, that being user identification, addressing and profiles for a subscriber. Profiles describe how the network should perform for this special user and may include parameters like QoS(bandwidth and traffic class) and special modes, or states for a subscriber.
HSS also holds information about authentication between network and UEs.

S-GW and P-GW

Serving Gateway (S-GW) and Packet Data Network Gateway (P-GW) deal with the user data plane and transports IP packets from EPC to external networks. The S-GW maintains paths from eNBs to P-GWs.

The P-GW is responsible for the communication between the mobile intra network towards the Internet.

1.4 Future - mobile networks

For the past ten years we have seen a rise in sensor activity. In the evolution of IoT devices, some has ported to LPWAN. However, as stated in the motivation, mobile IoT devices rely on good coverage and stability. Many people think of home electronics when discussing IoT, but this is not the main goal of IoT. Oslo's public transport operator(Ruter) has payment cards communicating with activation boxes over RFID, which in turn communicate with Internet. The Norwegian transport agency uses RFID in their tolling stations to communicate with cars. We also see smarter ways to implement these solutions. In the following sections we will give examples of application usage related to future mobile networks. In section 1.4.2 on page 13 we will discuss how to approach technical solutions of future applications.

1.4.1 Applications

Sensors used for applications are not new. Already back in 2009 a forum called ORGANIZATION FOR ECONOMIC CO-OPERATION AND DEVELOPMENT (OECD) discussed ways to take advantage of sensors to support a green growth [16]. This forum has members from all over the world, including Norway. The idea is that sensors can surveil and monitor emission numbers as well as act upon them. OECD came up with a group of applications which were important to investigate. Some of them are smart grids and energy control systems, smart buildings, transport and logistics, agriculture and other general industrial applications. Together these fields combines most of the energy use today and with smarter systems we can reduce power consumption.

With this in mind we can clearly see that many of the fields OECD discussed has been implemented, or is in the deployment stage. However the current status of industry activities related to these fields are running over 2G/3G networks. These networks provides good coverage while keeping the cost and energy low, but with billions of devices enrolling, this is not a sustainable model.

GSM Association (GSMA) has made an overview of the most applicable areas of use. In figure 1.3 on the next page some bullet points are presented

for each area. In the next section we will comment special use cases for some of the areas.

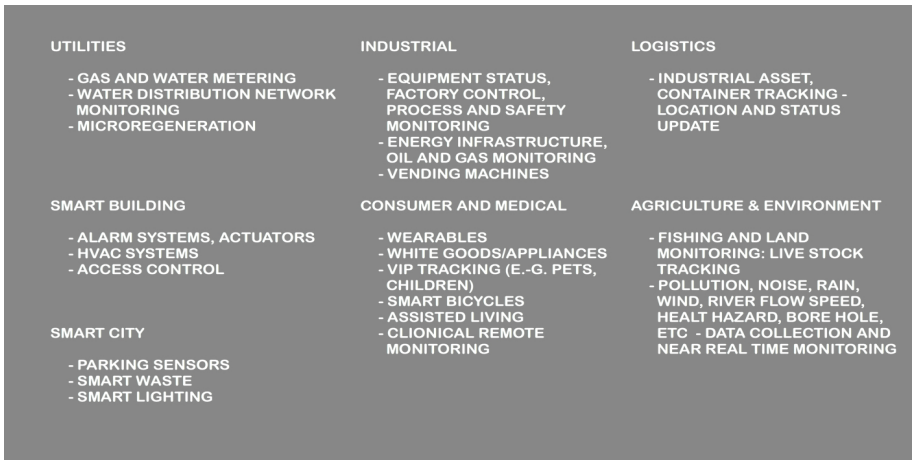

Figure 1.3: LPWAN example applications [6]

Utilities and smart cities

This domain resolves to what we are investigating in this thesis. Cities are getting smarter by the day and LPWAN can give us the stable network to deploy a large sensor network for monitoring and manage cities. Intelligent Transportation Systems (ITS) is an initiative to control and manage traffic. The system provides communication between cars, trucks and sensors. The sensors are mounted in traffic lights, signs and other objects known in the transport field. The connected devices can adopt to the environment and help the society in several ways. Emergency units can set a route to an accident and the overlaying ITS system will clear the way remotely to enable the emergency unit to quickly arrive at the location.

Other usage areas of sensors in cities are e.g. waste areas, power monitoring and CO_2 monitoring. Operations can heavily benefit from more information, as well as automated procedures. There are many use cases where monitoring can be difficult and costly. With simple and cheap devices one could monitor hundreds of different things. As a result the work effort could decrease and the processes can be more efficient. A good example is garbage disposal. If garbage containers were fitted with measuring sensors, the control system can be notified when it is time to empty the garbage. The use case for this will probably be bigger companies and industrial sites, however it might be used for private residents as well.

These devices need to be cost efficient since they will be deployed in large scale. In 2014 a power supplier in Norway called Lyse began installing smart power meters in over 140 000 households [13]. These devices communicate with the mobile network in the area over a proprietary standard. While it is important to explore new solutions, this is one

of many examples where standardized solutions will greatly outperform proprietary solutions.

Parking - a realistic use case Big cities are investing in smart parking even though cars may be banned from the cities in some cities. We know that many people will still prioritize traveling with car and facilitating for parking at a nearby location in these cases is crucial. These parking lots can be managed with sensors communicating over LPWAN. For Q-Free next generation parking sensor, they are mainly focusing NB IoT. The sensor is housed in by a small plastic case which will be leveled with the ground. It withstands extreme forces and detects cars by magnetometer and pulsed Doppler radar. The housing is the most expensive part of the sensor, being able to withstand rough conditions for over 10 years. The idea is that the sensor is drilled into the ground, and will communicate over NB IoT. The UE will send status of the occupancy of the parking spot with a fixed interval, as well as reporting parking events when they occur. See 1.4 for a photo of the parking sensor from Q-Free.

As of 2016 the parking sensors communicated to a common gateway(LTE embedded device), a device which communicates with the sensors using a proprietary narrowband communication technology in the ISM band, and to the Internet over LTE. This works fine, but as you might realized, the cost of the devices themselves and managing them are lowered significantly with LPWAN like NB IoT. Q-Free has chosen to go with NB IoT since this is the most promising technology providing the necessary specifications for their application. The communication between the sensor and the server is outlined in figure 1.5 on the facing page. The sensor sends data to the eNB which in turn sends the data to the IoT platform. If a server has been authenticated within the platform the message is sent to the server. The server receives the packet and can display the message however necessary.

Figure 1.4: Parking sensor [17]

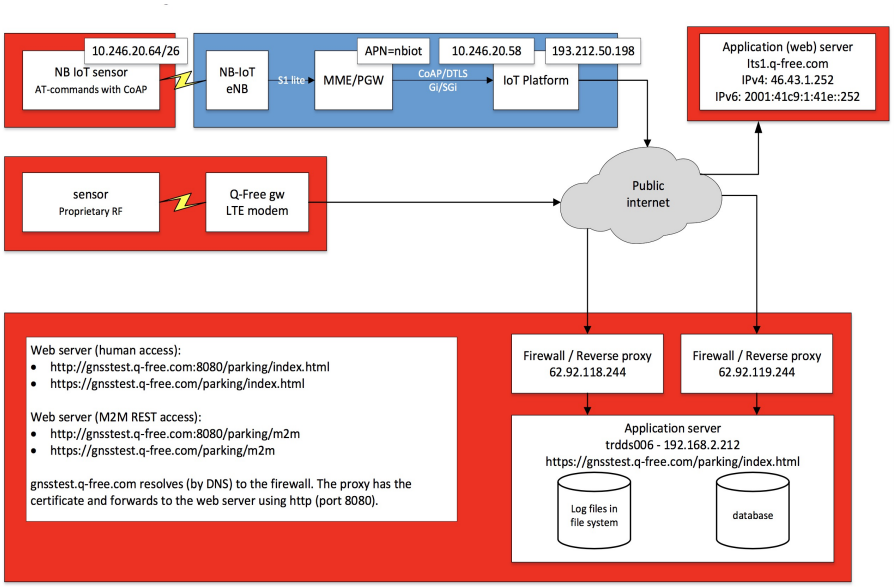

Figure 1.5: Outline of sensor communication [17]

In this setup, the sensor is the one sending data to the server. A more general approach to this problem is to implement the logic in the car. The parking spot would be equipped with a sensor with an ID of some sort. The sensor in the car could extract the ID and send a message about parking at that particular parking spot. This might enable several parking operators to cooperate over a general infrastructure.

Logistics

Today parcel tracking is a huge success and some companies are using sensors in their trucks and parcels for real time tracking. Using LPWAN networks will reduce cost of the devices and with long battery lifetime the sensors can be reused. In addition, the coverage for tracing the parcels will have to be great. However, for trucks it might be a better idea to wire the sensor to the trucks power outage. If you want real time, or near real time tracking, the sensor would have to send position data often and this will drain the battery substantially. There are some pros to use LPWAN networks for information collection, but this is probably not the biggest IoT area.

Industrial

Industry is a wide area and covers sites like gas stations, construction sites and factories. GSMA also includes vending machines in this category [6]. With sensors communicating over LPWAN industry stations can be more automated and it is easier to surveil the systems. Downtime on such systems means lost revenue and efficiency. Low cost sensors which are wireless will hopefully have good coverage and uptime so that these systems run better. This is probably one of the areas where these sensors will be used more frequently since these systems needs realtime monitoring. Another problem solver LPWAN could be for industry use is the excellent coverage. As mentioned good coverage would prevent downtime, but it is also important to notice where some of these sensors will be placed. Factories, sub-sea and underground are some of the locations where industry is often located and therefore coverage is especially important for industry usage.

Consumer

Many people think of consumer products when talking about IoT. We are seeing growth in smart watches and smart wearables. The biggest problem with these devices today is the battery life and this is due to the extreme cost of communicating either with your mobile telephone over bluetooth or over LTE. It will be interesting when these watches will use LPWAN. Some wearables, such as your watch may use a standard with higher bandwidth than e.g. NB IoT. For normal use, one would probably prefer CAT-M which offers bandwidth up to 1Mbit/s and could possibly provide music playback and phone calls.

Another interesting use case is smart homes. We have seen a move towards smarter homes for a long time, with better alarms, water and gas metering, light control and so forth. While this usually has communicated over

Wi-Fi, there is a potential market where tradition Wi-Fi is not possible or preferable, and hence LPWAN can be used.

Medical

A potential step into a healthier society is for medical clinics to be able to monitor and give realtime consulting to their patients. The patients device could automatically uploaded real time data to their doctor, enabling them to uncover symptoms or deceases at an early stage and by this increase the average lifetime. One can also collect a lot of data for patients with a special decease and research for a cure. For this to be realistic we really need LPWAN since these sensors will have to be cheap and the stability has to be good.

In Norway we are going into an era where the number of elderly will increase heavily. We should seize the opportunity to use this technology to take care of the elderly. A person which needs attention, but can live alone, would prefer a device which communicates with the care center. The person could also raise an alarm with this system to alert the care takers. In this way the person will probably have a better life and the cost will stay lower.

1.4.2 Developing wireless technologies

The general idea

As mentioned in the motivation the general idea of new wireless technologies is to reuse the current infrastructure. The new networks will use some combination of LTE and all its components as discussed in section 1.3.3 on page 6. One may also use GSM, as the current advantage of this network is that the coverage is better at the moment. In a couple of years LTE will be deployed all places GSM covers and we will probably see an increase in coverage as well. Coverage and power consumption are key features of developing wireless technologies and in combination with easiness it is the reason why some technologies stick while others fade away. Easiness is a requirement for LPWAN since the amount of data per packet is relatively small. For some of the technologies, guarantied delivery is not important, but it is important to acknowledge that we may need fast delivery for some applications which uses realtime monitoring. NB IoT will not suffice for these applications and we need to consider using technologies in parallel or speed up the transmission for e.g. NB IoT. In the future this may not be a problem considering the pace of wireless evolution. Never the less it is important to not suppress this issue. We know from earlier research that if a user of a platform has to choose between multiple technologies the easiness of the platform impedes.

eMTC/LTE-M and NB IoT

enhanced Machine Type Communication (eMTC) was standardized in the 12th release of 3GPP and updated with NB IoT in the 13th release. It is meant as an high bandwidth alternative to NB IoT and is often referred to as LTE-M1.

Both standards are implemented using the current LTE infrastructure, but they differ in coverage, bandwidth and the number of bands used in a cell tower. We see in figure 1.7 on the facing page that LTE-M1 provides bandwidth up to 1Mbit/s while NB IoT peakes at around 200Kbit/s(rates from the release they were standardized). The coupling loss of LTE-M1 is said to be around seven times better than LTE, and NB IoT ten times better. We will cover the details about how NB IoT uses the current infrastructure in section 2 on page 17. LTE-M1 can be used in the same matter.

LoRa [1]

LoRa is the most popular non standard solution for enabling IoT devices. It is currently deployed in many cities and uses a proprietary solution. The devices which has a LoRa chip uses RF or WiFi to communicate with a common gateway, typically a "cell" tower. The current range is up to 15km, but in dense areas the range only covers 2-5km. The bandwidth is low, but in light of the messages being passed to the gateway that is fine. In figure 1.6 we see the overlaying infrastructure of a LoRa network. It uses many of the same features as we mentioned in the motivation where the devices communicate with a middle box(concentrator/gateway), which in turn communicates with Internet over 3G or ethernet.

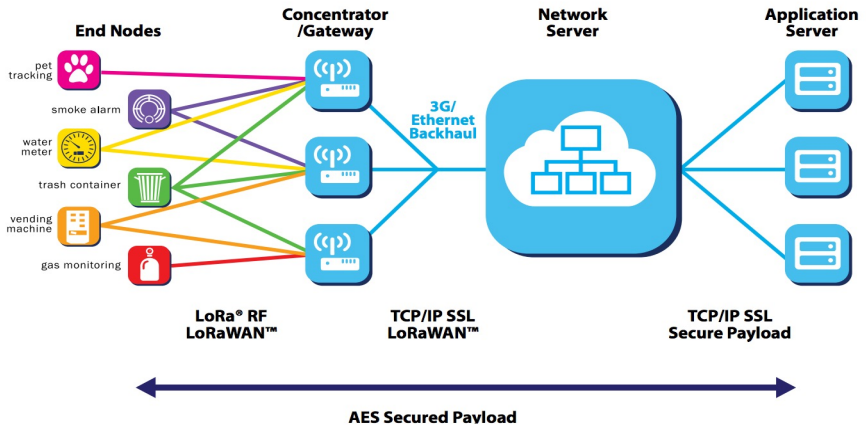

Figure 1.6: LoRa infrastructure [1]

The LoRa solution provides an LPWAN at relatively low cost and gives average coverage compared to NB IoT and LTE-M1. However, LTE is being

deployed globally and if the coverage is good enough it will be more costly to provide LoRa networks as well as NB IoT or LTE-M1 networks.

	LoRa	GSM (Rel.8)	EC-GSM-IoT (Rel.13)	LTE (Rel.8)	eMTC (Rel.13)	NB-IoT (Rel.13)
LTE user equipment category	N/A	N/A	N/A	Cat.1	Cat.M1	Cat.NB1
Range Max. coupling loss	<15km 155dB	<35km 144dB	<35km 164dB	<100km 144dB	<100km 156dB	<35km 164dB
Spectrum	Unlicensed <1GHz	Licensed GSM bands	Licensed GSM bands	Licensed LTE bands In-band	Licensed LTE bands in-band	Licensed LTE in-band guard-band stand-alone
Bandwidth	<500kHz	200kHz	200kHz	LTE carrier bandwidth (1.4 – 20MHz)	1.08MHz (1.4MHz carrier bandwidth)	180kHz (200kHz carrier bandwidth)
Max. data rate*	<50kbps (DL/UL)	<500kbps (DL/UL)	<140kbps (DL/UL)	<10Mbps(DL) <5Mbps(UL)	<1Mbps (DL/UL)	< 170kbps (DL) < 250kbps (UL)

*Max data rates provided are instantaneous peak rates.

Figure 1.7: LPWAN emerging solutions [15]

Chapter 2

A dive into NarrowBand IoT

Telenor started their NB IoT network test late 2016, with hardware mainly from Huawei. The goal of the test was to set up a NB IoT network and test the communication with Q-Free's parking sensors, and run own tests before releasing the network to the public. They aim to launch their NB IoT network in 2018 (HH)

We will focus on a general approach towards NB IoT as technology and will test both Telenor and Telia's solution. In section, 4 on page 43 we will discuss how and where we did the tests as well as giving you examples and results.

NarrowBand (NB) Internet of Things (IoT) is the most promising LPWAN solution and the focusing technology in this thesis. The standard is made by 3rd Generation Partnership Project (3GPP) and European Telecommunications Standards Institute (ETSI), and was launched late summer of 2016 with the 13th release of the The Mobile Broadband Standard. The technology takes advantage of the current LTE infrastructure in a very sufficient way. The deployment requires no extra hardware, so the software necessary can be implemented into current hardware. In section 2.2 on page 20, we will elaborate on the software required, as well as three ways to deploy NB IoT in the frequency plan.

2.1 System design

In a system design process it is important to analyze the use of the new technology. It is crucial to understand how people and the industry will take use of NB IoT. A good example of a standard which is causing problems today is IPv4. When this standard was introduced, no one could predict the growth of online devices and for this reason NB IoT has some specific system design features which hopefully will cover its use for foreseeable years ahead. In the next section we will briefly introduce the system requirements.

- Link budget improvement
  The sensors and devices in mind when deploying NB IoT need better coverage than the average phone or LTE modem. As mentioned these sensors will be used in vast areas, underground and indoors. It is therefor important that the coverage is suited for this use. The goal is to reach 20dB extended coverage compared to LTE. Not only does this extend the range that these devices can be deployed, but they can emit transmissions with lower power, and hence use less power.

- Density
  In the article from Cisco [2], they predict the growth of IoT devices. The prediction is that each household has on average 4 devices. According to Oslo council there are approximately 332 568 households per 1.1.2016 [10]. This means that if we were to calculate the device pool today it would be around 1.3 million devices only for house holds. Taking into account that these devices will be broadly used by the industry as well, the device density will be very high. To put this into perspective we can investigate the device density of GSM. For each GSM cannel (200 KHz) there are 8 time slots, giving 8 concurrently connected devices. In one cell tower there are approximately 8 channels, and they are often split between mobile providers. If we assume that all of them belong to one provider, one cell tower can support up to 64 devices concurrently. One GSM cell tower will provide coverage for the surrounding 1km, and since cell towers to close to each other will cause interference only one cell tower can provide coverage for 1km in radius. LTE has much more resources and can provide coverage for more devices, using time and wave division multiplexing.

  There is however a presumption to why NB IoT can support so many devices per carrier. In the specifications one NB IoT channel can support more than 50 thousand devices and keep in mind that NB IoT can be deployed in one GSM channel, which only supported 8 devices. The way this is supposed to work is due to the infrequent updates from the devices. One device might send a message every second hour and another might even send messages only once every day. One scenario where this might cause problems is if every programmer/company sends updates on specific times. Typically one would want updates at the hour marker(ie. 14:00, or 15:00). If thousands of devices do this at the same time, congestion and latency will definitively effect the experience for the devices.

- Low complexity
  A key feature for the devices supporting NB IoT is that the complexity level is low. The new standard is less intricate than ordinary LTE which means that the devices can be less complex, resulting in cheaper hardware. This is an important factor for success since these devices need to be cheap to be broadly deployed. The complexity in this technology resides in the core mobile network for it to support

many devices. Other than this, the technology supports average speed and average latency, perfect for monitoring purposes.

- Low power
  The power consumption of the communication needed for LPWAN need to be low for the estimated battery lifetime of a device to be kept. The goal is over 10 years of lifetime, and since these devices probably will be very cheap, it might be cheaper to change the device instead of changing the battery.

  Low complexity and better coverage will help the device to consume less power, but it will not enable the device to operate for 10 years. The system design includes a special operation mode, called Power Saving Mode. We will elaborate how this key feature will work in section 2.3.4 on page 24.

- Latency
  For most applications applied to this new technology, latency is not key feature. We would like the device to send frequent, or infrequent message to a server or another device, but the time used is not important. However, what is actually a long time? A ping request on a normal wired connection today uses around 5-100ms, and even less for fiber connections. On mobile networks like LTE, the usual ping time is on average higher than on a wired connection, but usually it does not exceed 100ms. NB IoT aims at a peak latency of 10 seconds. In comparison that is 100 times longer than a normal WAN connection, but it should suffice for most applications using LPWAN. The latency could probably be lowered, but it might interfere other design features, such as device density.

- Security
  A big issue in Internet today is security, specially DDoS attacks. These attacks are more frequent and is an attack form which enables thousands of machines to continually spam one or several IP addresses, which in practice disables the application. In the fall of 2016, the DNS service provider "Dyn" was down because of a DDoS attack. Being a big DNS server, Dyn's downtime impacted major Internet platforms in both USA and Europe [23]. Since an attack like this needs a huge device pool it is likely to think that IoT devices are a potential tool for hackers. If hackers could access millions of devices with Internet access, they could easily perform global DDoS attacks which would halt our infrastructure, in turn disabling many of us to do our jobs.

  With this in mind, the engineers and designers of NB IoT needed to consider security issues like this. We mentioned the IoT platform when discussing the deployment in section 2.2. This platform is the glue of NB IoT and also a huge contributor of security measures in cooperation with MME and HSS. The system requirement of NB IoT is that no one should be able to directly access a NB IoT device. One

additional security feature in general with mobile networks is SIM cards. Like cell phones, NB IoT devices will be fitted with a SIM card, which will enable the core network to authenticate the device, and visa versa. The SIM card has a subscription connected to the service provider, hence making a strong authentication scheme.

2.2 Deployment

In figure 1.2 on page 7 we saw the current infrastructure of LTE. To manage security issues and new features, a solution is to introduce a new term called IoT Platform (IoT Platform). This platform will keep track of communication between outside applications and inside nodes. In theory the communication would suffice without this platform, but due to the extreme security issues regarding sensors we need a way to authenticate traffic. Many IoT devices are easy to hack and can be used for Distributed Denial-of-Service (DDoS) attacks. We have seen many service struggling with downtime due to overloading of their servers because of DDoS attacks. The platform ensures that traffic between two entities is secure, where one being the sensor and the other being the application server. For an application to receive traffic from a sensor, it will have to negotiate with the IoT Platform to gain access. After the setup procedure the IoT Platform acts as a NAT device, as well as matching correct application to correct user group, often called subscription. We might see other solutions for securing the internal network without negotiations.

IoT Platform is just one way to solve these issues. This software is in fact just an application proxy, or a firewall. Either way it can be implemented in the current hardware, but it is also possible to provide an extra rack mounted device to deal with this. In Telenor's test case, spring 2017, all the servers/hardware required was located at one location and the IoT Platform was running on private hardware from Huawei.

2.2.1 Operation modes

NB IoT occupies 180kHz bandwidth and there are currently three ways to integrate NB IoT with LTE. The available operation modes are standalone, in-band and guard-band - see figure 2.1 on the next page for illustrative definition. Standalone operation uses a dedicated GSM channel, utilizing the ability to deploy NB IoT in GSM as well as LTE. In figure 2.1 on the facing page we illustrate this operation mode to point out that NB IoT can be deloyed in GSM, even though this is not for the long run.

In-band operation utilizes bandwidth within one LTE carrier. This is the most technical and advanced solution and requires more work by the mobile network provider. The reason for this is because of interference between the LTE and NB IoT sections in the carrier. It is worth noting that

not all resource blocks within one carrier is available for NB IoT. TODO verify??

Guard-band operation makes use of the bands not in use by LTE due to interference between LTE carriers. This in-between section is called a guard band and guards the carries for interfering with each other. This band can be used by NB IoT and is a solution where one would want to utilize all resources of a cell tower.

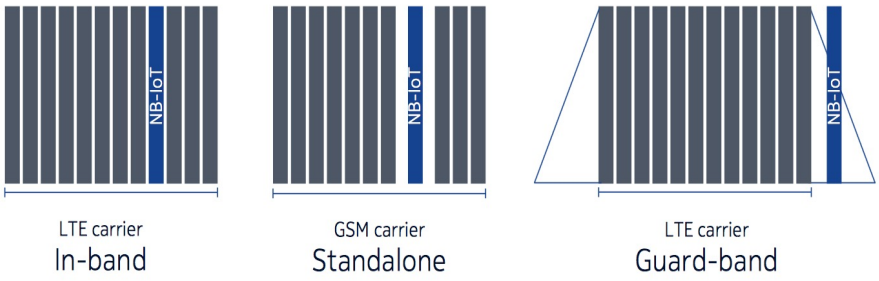

Figure 2.1: NB IoT operation modes [15]

2.2.2 Security

2.3 Introduction to energy saving

Power saving is an important part of mobile communications for sensors. Several techniques are used to reduce power consumption, and Discontinuous Reception (DRX) and Power Saving Mode (PSM) are two of them. DRX has existed for 20 years and is implemented in LTE, while PSM was introduced with the development of LPWAN. These two modes, together with paging2.3.5 introduces an improved network, especially suited for sensors. In figure 2.2 on the next page you can see an outline of the communication process for a device. RRC Connected mode is the mode where the device actually can communicate with the network. The following procedures shows a possible time convergence graph for a device. We will explain the techniques used, as well as the specific time periods of the figure in the next sections.

Prerequisites To get a good understanding of the results we need to closely look at the outcome of the results and understand the meaning. In the following sections will describe details about how NB IoT works and what the specifications are saying. As stated we will be mostly focus our tests on coverage and power usage, which relates closely. Coverage is measured in Decibel / referenced to milliwatts (dBm) and provides a relation between the distance between the base station and the UE, and the transmit power of the UE. In theory a device will transmit with the relative strength according to the coverage, so that the signal is received at the base

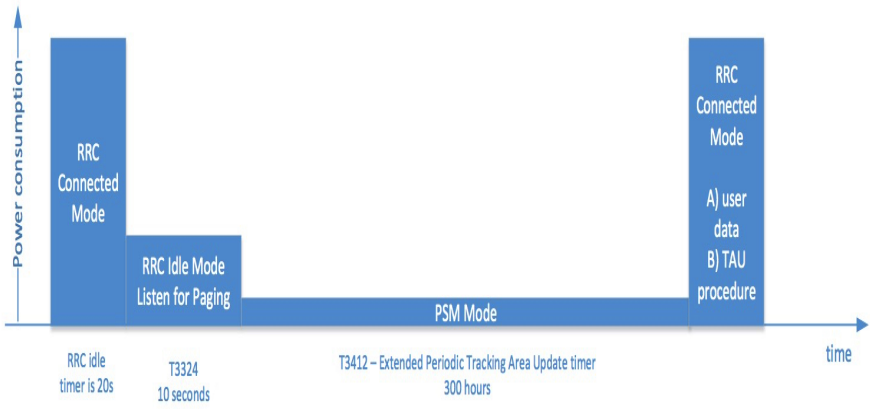

Figure 2.2: NB IoT time convergence [17]

station. Depending on the coverage of the device it will adjust the transmit power.

With this in mind we can convert coverage and transmit levels to output power. In worst case scenarios the device will transmit data with +23dBm, which under normal circumstances is converted to 230mA [21]. We can convert dBm to mW by calculating 10^(dBm / 10). Keeping +23dBm in mind gives us 10^(23 / 10) = 200Milliwatt hour (mWh). We can also convert from mA to mWh by multiplying mA with the voltage, this gives us 0.230Ampere (A) * 3.3V = 759 mWh. The voltage into the chip is rated at 3.3V and we can assume that the spare power is used for keeping the device in connected mode which we will discuss later. HH

NB IoT operates between -40 and +23dBm, where -40dBm equals 10^(-4)=0.0001mWh and +23dBm equals 10^(2.3)=200mWh. We will see how this will influence the power usage in the tests and we will also touch on the subject in the following section.

2.3.1 RRC connected mode

This mode is used when the device wants to communicate with the network, either UL or DL. The time spent in RRC connected mode can vary, but the default NB IoT timer is 20 seconds. In this mode the device will use a lot more power, hence we want to move away from this mode as soon as we are finished transmitting or receiving data. When the chip is in active/connected mode it uses 6mA=0.006A * 3.3V =0.0000198mWh.

When the transmit operation is finished the chip will stay in connected mode until the time period ends. There is an optional flag, "release assistance indicator", in the AT send command which puts the device into PSM directly after the transmit operation. The network needs to support this action to give the UE permission to sleep. It is the network which

decides what the UE should do according to the network and the devices status. Given that the network supports release assistance indicator the battery life would be extended, since we only send data for a short period rather than waiting for downlink data which in many applications is not needed.

The calculated numbers in this section are based on specifications from the manual of the NB IoT chip[21]. We will try to test and verify these numbers in section, 4.2 on page 48.

2.3.2 RRC idle mode

After ending RRC connected mode the device enters RRC idle mode listening for paging, using approximately 1mA. The default NB IoT time is 10 seconds and in this mode the device can receive data from the network with a paging request, meaning that there is data for the device in the network. If there is more data in the network the device will re enter RRC connected mode and try to receive data from the network. If there is no more data for the device it can enter PSM or eDRX

2.3.3 Extended Discontinuous Reception (eDRX)

Discontinuous Reception (DRX) is a way in LTE to keep the device connected, but reducing the power usage by only checking for new data(paging) on time slots. After a sequence of communication the device goes into a DRX state where it periodically checks if the network has new information for the device. If there is information for the device, the device enters RRC connected mode where it can receive and send data. In the figure the DRX time(Active Time (T3324)) is set to 10 seconds, after a period of 20 seconds in RRC idle mode.

When designing NB IoT, DRX had to be implemented. However if the device always checks for paging the battery would decrease at a high rate. eDRX is therefor an outcome of the idea of DRX in the special case of sensor networks. It is a new way to handle communication for sensors which needs to wake up for certain events or messages from the network. In an article about eDRX and PSM for LTE-M1 we find a great figure showing eDRX 2.3 on the next page. Even though this is for LTE-M1 it applies for NB IoT as well. After a device has communicated, either sending or receiving packets over the network, we want it to use as little power as possible. By default the device will enter DRX mode for a network-specified time and if configured go into PSM mode after the given time. As mentioned, for some applications we want the device to be able to receive packets without waking it self up.

The device will then go into eDRX instead of PSM mode. This means that for the most of the time the device will sleep, but periodically it will

check for new information(paging). The period where the device sleeps is configurable in the network. Since the internal clocks in the devices often are unreliable, and the time between these syncs may be long(typically 1-2 hours), we need to first sync the clock and information between the device and the network. We call this operation sync guard. We have to do this due to communication based on time(time slots/Time division multiple access). After this process the device will operate in normal DRX mode for a certain period(10 seconds) and go to sleep after that period. This is a very power efficient way to keep the device more synchronized with the network, but it raises some practical problems. (either remove the last sentence or discuss it later). In the parking use case we only need to send information, hence we don't need to activate eDRX mode.

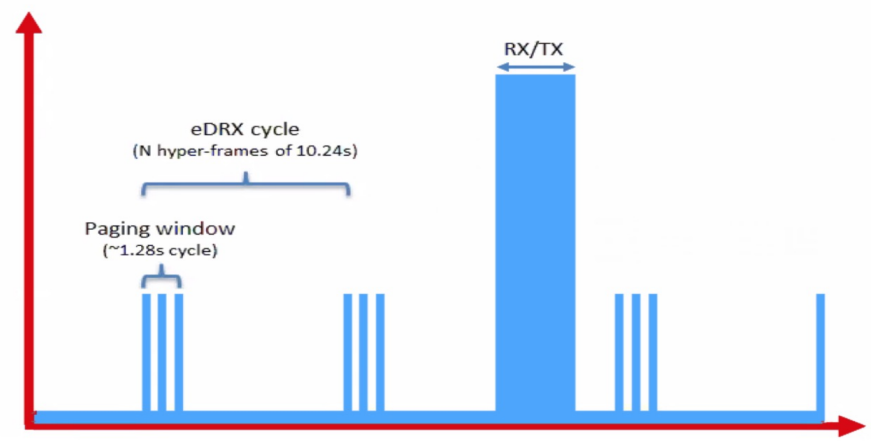

Figure 2.3: eDRX operation mode [11]

eDRX has a set of configurable parameters which define how the UE behave when entering eDRX mode. In short terms we want to specify eDRX cycle length and paging time window. The cycle length is the length of an eDRX cycle. This means the time from start of a period to the start of the next period. The cycle length can vary from 5 to 2621 seconds. The paging time window can be set to 0 to 20 seconds, where the value 0 means that paging is not in use. In section, 2.3.5 on the facing page, we describe what paging is.

2.3.4 Power Saving Mode (PSM)

PSM is the core of power saving in NB IoT, as well as many other LPWAN. The idea of almost all LPWAN is that the devices send information periodically, while sleeping 99% the rest of the time. After a typical connected period we want to enter a sleeping mode where very little power is drawn, usually by an internal clock. This mode is called PSM and enables the device to sleep for a network configured time period. In our test case this time(Extended TAU Timer (T3412)) is set to 300 hours, almost two weeks. That means that the device will be removed from all meta data

in the MME after 300 hours if there is no communication. This technique is foremost for the devices which only sends information to a server. The way it works is that the device goes from connected mode, to DRX mode listening for information from the network. After this the device enters PSM mode if activated. When in PSM mode the only thing going on in the device is an internal clock which can start the system at a specific time. This is necessary since we want the device to wake up at a specific time and perform POST/SEND operations towards a server.

In PSM the chips power usage will be as low as 3μA, which is extremely low and the main reason why we can expect these types of sensors to achieve 10 years of battery lifetime.

2.3.5 Paging

Paging is a procedure to initiate communication from the network. If paging is enabled the UE will listen for data from the network at given time intervals within a RRC idle or eDRX period. If the serve wants to communicate with the sensor it sends a message over the network and it will be stored at the IoT Platform until the device is paged in.

2.3.6 ECL

As mentions in section, 2.3.1 on page 22, the UEs will enter different coverage modes based on their signal strength towards the base station. There are three levels of Coverage Enhancement Level (ECL). At which coverage level devices switch over to the different levels is decided by the network. Telia uses -105dBm for ECL1 and -115dBm for ECL2 [**mail:teliaMailThread**], coverage better than this resolves to ECL0. OML Telenor?

In ECL0 the device will transmit the data once, and in level 1 and 2 the UE will retransmit the data to ensure delivery of the packet.

2.3.7 The transmit procedure

We have looked at several techniques for NB IoT and the specifications looks very good when the UE has good coverage. In good coverage areas devices will transmit data with -40dBm, with an average actual transmit time of 200ms. However in worst case situations, the sensor will boost the signal up to +23dBm, which in terms of power usage is two million times as high - (10$\hat{2}$,3) / (10$\hat{}$4) = 2 000 000. In addition, as stated, a sensor with bad reception will try to retransmit data. Depending on the coverage the device will enter different coverage modes, described in section 2.3.6. Given that the sensor has bad coverage, and is operating in ECL2 mode the transmit time might be as long as 5 seconds for a single data transmit. The actual

power usage increases rapidly and the chip would use 200mW for as long as 5 seconds, resulting in a power usage of 200mW*5S=1000mW/s, which actually is 50 million times increase in power usage compared to optimal conditions. However, as we will see in our tests, -40dBm is normally not possible to achieve.

In the following figure, 2.4, is an outline of how the device chooses ECL mode dependent on the signal quality. In theory the device should lower its transmit power equally to the signal strength. The graph represent the theory behind the adjustment of the transmit power and you will see how this actually works in section, 4 on page 43.

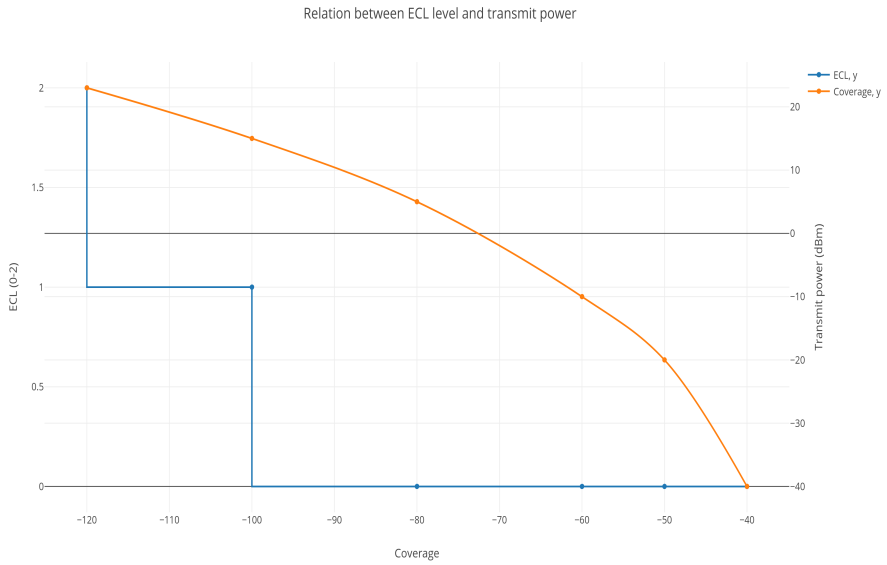

Figure 2.4: ECL and dBm

2.4 Challenges

what if the network fails? power outage? what should the application do if the sensor wont send? how long should the sensor wait until doing anything?

With new standards there are always complications and challenges. One challenge in particular is the amount of communication between UEs and the network. Say that your application needs to communicate every minute and the producer wants to use NB IoT. The implementation will work, but might not be the optimal solution.

Another challenge for software developers and power usage is that some UEs connected to NB IoT will be online for a decade. This means changes in the network and possible power outages, which implies that the sensor has to cope with network changes and downtime in the network. Each time the

sensor has to wake up and search for network the power usage increases and the lifetime of the sensor decreases. See section, 4.2.6 on page 49, for tests.

And consider the parking use case. Imagine a parking lot at a football arena, maybe in conjunction with a shopping mall. For most of the time the devices will send period messages when people are shopping and the sensors close to the main entrance will of course send messages more often then the ones long away. At the same time when a match is about to start the parking lot will be full instantaneous. This raises a particular question, how will the network cope with so many devices sending data close to each other. The number of devices activated in that particular area will be increased in a short time which may cause congestion and incorrect updates as NB IoT is not fast or "reliable" in the sense that the data can be lost or delayed. The network may cope with these problems just fine, however it is worth considering these problems.

Even though parking could raise congestion problems, the flow of cars to a parking lot is limited. All parking spaces will not be filled at the same time. However, for water, electricity or other measuring procedures the sensors might be configured to send data at the same time. Take electricity measurement as an example. According to wikipedia [18] there are approximately 5 000 residents per square kilometer in Oslo. In the city center this number is probably a bit higher as well. With this in mind one cell tower will provide coverage for around 2-3 thousand house holds and their electricity measuring device. If all of these devices sends updates at the same time, as well as other traffic is incoming, the network might be overloaded.

We will try to test these types of edge cases to see whether NB IoT can handle difficult scenarios or if these applications would have to use a different technology.

Part II

The project

Chapter 3

Planning the project

To get accurate and meaningful results we had a long planning phase. First of all we needed to decide how and what to test. We decided to set up a server, which can receive data from a NB IoT chip. In addition, this server will be able to analyze the data and display the results in a meaningful way. We also needed a way to send data over NB IoT to the to the server. We were so lucky to borrow a development kit for NB IoT from Q-Free which enabled us to connect to the network. We also borrowed a SIM-card from Q-Free which used Telenor network. We also managed to get a SIM-card from Telia so that we could try to do some comparison between the two network providers. In section, 4.2 on page 48, we will go into further detail about how the setup was and why we did the different tests.

With this in mind, we will describe and give you an introduction to the different parts of the setup and show you how they are tied together.

3.1 The web application

In order to collect data we needed a web server and we needed to consider a few things, what should be included in the server, which platform should the server run on and where should the server be located. At the same time it was important to think about how the data would be used. The simple solution would be to collect the data and import it into an excel document for analysis. However, the amount of data is big - in fact the db consists of XXX (HH) GB of data with over XXX elements. Rendering this in an excel document would not suffice for an effective analyzation of the data after the test. We decided to implement a web app to display information about the sensors. This web app can display latency, coverage and power related data about each sensor as well. In this way it is easy to view the data and gain knowledge about the statistics created.

In the following sections we will discuss the implementation of the server and the web app.

3.1.1 Backend: Node.js

We decided to use node.js for what would become the backend of the server. Node.js is a new and modern way to create simple REST APIs. A REST API follows a set of rules to make it robust and stateless. It also should include all CRUD operations(create, read, update and delete) and should use the appropriate request method, such as POST, GET, PUT and DELETE. A big difference from standard backend programming is that Node.js uses javascript as programming language. Even though javascript is not the cleanest programming language it is efficient and the codebase is kept low. In addition node.js takes advantage of using Node Package Manager (NPM). NPM offers packages for all applications, both frontend and backend, and we will discuss some of the packages used in the server.

The server hosts multiple API endpoints. The most important are of course collecting data and retrieving data. The database contains a lot of data collected from different tests and it would require a lot of rendering to do analysis on the client side. We decided to analyze the data at specific intervals so that the client would simple display the data. We will describe the analysis and data which is displayed in section, 3.1.2 on page 38.

The database

There are multiple ways to store data and there is always a trade off when choosing the platform. PostgreSQL is a great example of a database which is fast and reliable, but it can be complicated to set up and is SQL driven which means all data has to fit into one or several tables. Since this project is small and the data is be represented in a couple (HH describe in "the project") of different ways we have chosen to use a NoSQL database. There are many good tools for storing data in a NoSQL database. ArangoDB and MongoDB are the most commonly used databases and both works well with Node.js. ArangoDB is often used for geo specific tasks and mongoDB was a bit easier to use, so we decided to use mongoDB for our data storage. In paragraph, 3.1.1 on page 35, we will explain how we used mongoDB for our implementation.

Node Package Manager

Express

Express is a fast and robust package for handling routes and offers nice APIs for listening on specific ports and so forth. In the following code example, 3.1 on the next page, you can see the base setup for making your server handle requests. The port is set with a combination of the processes port and user specific port selection. The reason why this might be useful is if your server is hosted on a payed server which might be deployed on

different IP and port for each deploy. With this implementation you won't need to consider this. In the example you can see a GET for a specific path, however you can also redirect all routes to a separate file for cleaner code. The last thing we need to do is to listen for incoming requests on the specific port. Now express will handle all request with an event loop.

Listing 3.1 Base express setup

```
// express setup
const server = express();
const port = process.env.PORT || 8020;

server.get('/', function (req, res) {
  res.send('Hello World')
})

server.listen(port, () => {
  console.log(`app started on`, port);
});
```

Coap Coap is a simple network protocol and offers request like HTTP, but without all the overhead. Coap is designed for low powered devices and will be used for our tests. I have used node-coap package for including coap support on the server. Since the protocol is simple, we need to handle the request a bit different. The following code example, 3.2, shows a simple node-coap setup.

Listing 3.2 Base coap setup

```
var coap = require('coap')
var coap_server = coap.createServer()

coap_server.listen(port, () => {
    logger.log("info", `Worker ${process.pid} started coap
        server on ` + port);
})

// All request is handled by this function. It is not
    possible to request a specific url path
coap_server.on('request', function(req, res) {
    // Payload of the request is a byte stream. Parse the
        stream and handle the data
    var data = JSON.parse(req.payload.toString());
})
```

Cluster With this simple setup all request will be handled by one thread by express as Node.js is single threaded. However with a package called cluster we can handle several request asynchronous. It is based on web

workers which are an abstraction for processes in Node.js. The following code example, 3.3, shows how you can enable multiple processes to handle your request to improve efficiency on your server. The cluster package offers a set of functions, so the first thing we do is to verify which worker is master. The master then forks a number of processes which then will request the path '/' and listen on the specified port. To avoid that all workers handle every request, cluster passes request in turn to the workers, normally in round robin fashion where worker 1 gets the first request, worker 2 the next and this goes on in a loop. Since the server will run over a long time, we need to handle workers which dies. The master process will pickup workers which die and respawn processes so that the server can continue processing requests.

Since the server basically puts everything on hold while analyzing the data we needed a way to handle request at the same time. The cluster package enables the server to handle incoming request as the analysis is ongoing.

Listing 3.3 Express setup with cluster

```
1  const server = express();
2  const port = process.env.PORT || 8020;

4  const cluster = require('cluster');
5  const numCPUs = require('os').cpus().length;

7  if (cluster.isMaster) {
8      console.log(`Master ${process.pid} is running`);

10     // Fork workers.
11     for (let i = 0; i < numCPUs; i++) {
12       cluster.fork();
13     }

15     // Respawn workers on exit
16     cluster.on('exit', (worker, code, signal) => {
17       console.log(`worker ${worker.process.pid} died`);
18       cluster.fork();
19     });
20 } else {
21     server.get('/', function (req, res) {
22       res.send('Hello World')
23     })

25     server.listen(port, () => {
26       console.log(`app started on`, port);
27     });
28 }
```

Webworker-threads Webworker-threads is an additional safety feature to prevent degraded latency results. Web workers enables our application to

run background threads without delaying the event loop of Node.js. I use webworkers when getting requests to run analysis on demand.

MongoDB Storing data is key in any server and we use the MongoDB package for our server. The package includes a nice API for MongoDB storage and is simple, but highly effective. In contrast to SQL, NoSQL databases uses different key words to describe data. MongoDB can consists of several databases, and each database can contain several collections. Within each collection we call each entry a document and the document can contain data with any size and information. As with SQL one has to setup the MongoDB server so that applications can connect to it. On linux it is as simple as "sudo apt get mongodb". The install process creates a service, mongodb.service, and this service starts up at boot time and is ready for incoming connections.

The following code example, 3.4, shows how to set connect to the db and insert a simple document in a collection with an API call.

Listing 3.4 MongoDB setup and insertion

```
1  const MongoClient = require('mongodb').MongoClient;

3  MongoClient.connect('mongodb://localhost:27017/db', (err, db
       ) => {
4      if (err) return err

6      server.post(api + '/nodes', (req, res) => {
7          const data = req.body;
8          db.collection('nodes').insert(data, (err, result) =>
               {
9            if (err) {
10             logger.log("error", "insert failed: " + err);
11             res.send({ 'error': 'An error has occurred' });
12           } else {
13             res.send(result.ops[0]);
14           }
15         });
16     });
17 }
```

Moment Storing data with MongoDB is easy, but deciding the format of the contents can be difficult. A normal problem on server applications is server time, versus client/sensor time. A common approach is to use UTC time everywhere and moment.js for npm is a great tool to keep track of time. The following code, 3.5 on the next page, takes in a timestamp and creates a moment object in UTC time. This original timestamp is not converted, so the sensor also needs to send the timestamp as UTC time. The conversion of timestamps happens in the web application.

Listing 3.5 Simple moment example

```
server.post(api + '/nodes/:id', (req, res) => {
    let data = req.body;
    let timestamp = moment.utc(data.timestamp);

    ...
}
```

3.1.2 Frontend: React

With a solid backend in Node.js we went with React for our web app. We will not go in detail on the different packages used for the web app, since this is not related to the thesis. However, we will give a brief introduction on how to use the app and what kind of analysis you can do with it.

Layout

The web app is used for hosting data related to long term tests and how the device performs on a more abstract level than with the detailed tests in section, 4.2 on page 48. The home page contains a short paragraph about the thesis and links to the latests version of the thesis, source code related to the project and results from the detailed tests, see screenshot 3.1 on the next page. Further more you can select nodes on the left which includes data from different sites and network providers. In some of the intersections there might be data from other network providers, hence we can disregard data which looks completely different from the rest of the data related to that node, see screenshot 3.2 on the facing page and 3.3 on page 38.

By default the graphs will display data one day backwards from the latest data entry. So if the day you visit the page is 20.07.18, and the latest data entry is 02.03.18, you will see data from 1-2 March of 2018. The time range is stored across nodes, so please double click on one of the graphs to display all data. You can also inspect certain areas by selecting an area of one of the graphs and the other graphs will adjust. In a production version of the web app we would apply some logic to separate x axis range between selected nodes.

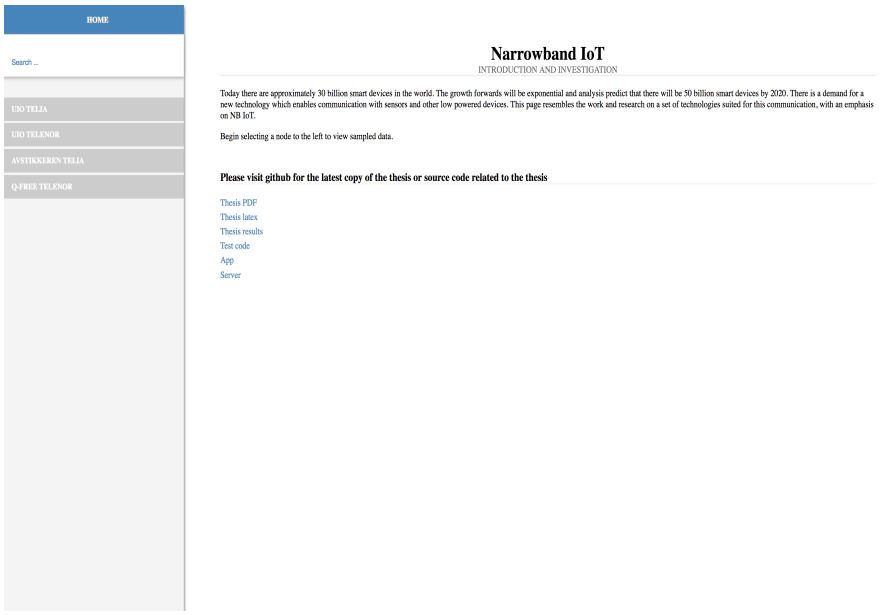

Figure 3.1: SensorApp homepage

Figure 3.2: SensorApp nodepage part 1

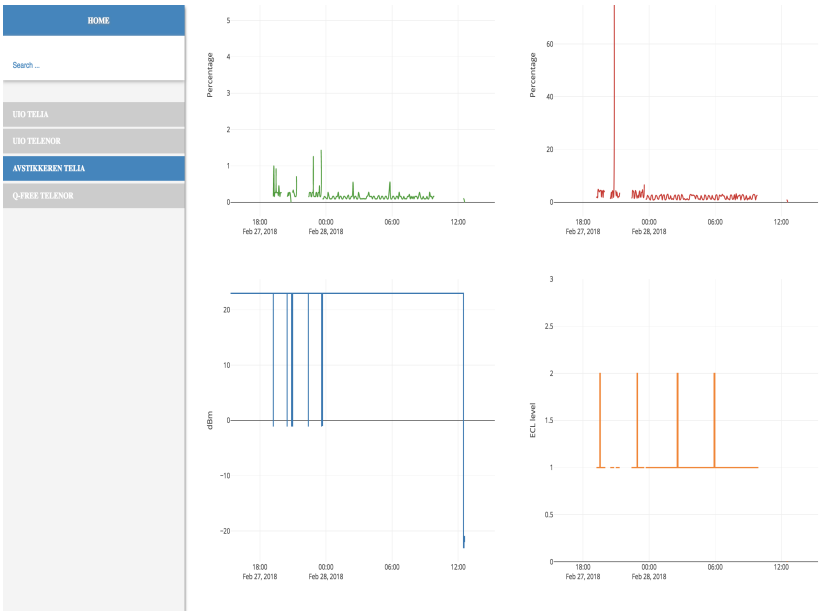

Figure 3.3: SensorApp nodepage part 2

Analysis

When selecting a node to explore you can explore the different graphs and we think this is a good way to display statistical data. The graphs data represents the output of the analysis on the server and we will try to describe what the graphs present and why we chose the different aspects of the behavior of the network. Remember that the data is mostly produced by **AT+NUESTATS**, so read, 4.4 on page 50, for any deviations.

Coverage and Latency One of the main features of NB IoT is coverage, hence we wanted to display coverage statistics over time. One interesting point of coverage is that it is closely related to the power usage of the device. The latency of NB IoT is not a key feature, but the desired latency is under ten seconds according to the specifications **??** on page ?? and by displaying the latency in the same graph as coverage we can clearly see how the different coverage levels inflict latency in the network. The output of coverage and latency is directly from the output of **AT+NUESTATS**.

Recieve and transmit time The output from **AT+NUESTATS** gives us a counter from receive and transmit time in milliseconds. We used the value between each entry to calculate how much time the receive and transmit time. The analysis produces a percentage which relates to how much time the device was in receive or transmit state in one interval. This is also closely related to the coverage of the device and also the ECL level which indicates the number of retransmits per packet sent. The reason why we would like to monitor this data is because the device will mostly use power

in these periods, so if the device is active for longer periods in either receive or transmit state we know that the expected lifetime is degraded. Most of the time the active percentage will be low due to usage of release indicator flag and PSM mode, however if the transmit frequency is low the device will more frequently be in some kind of active state, hence the transmit and receive time will increase.

ECL The coverage mode is also closely related to the power usage of the device and an important factor we could monitor. You can clearly see that the power usage increases when the device enters ECL mode 1 and 2 in the testing section, 4 on page 43. The output of ECL level directly from the output of **AT+NUESTATS**.

Transmit power The amount of power used for transmit is decided by software and is key to power usage. The output we get from **AT+NUESTATS** show us the latest transmit power level of the device. Sometimes the device will send a negative-acknowledgement (NACK), which is always sent with +23dBm signal strength and is what **AT+NUESTATS** sometimes picks up as the latest transmit power level. In the long term tests with graphs from the web app the logging of transmit power might not be correct related to what transmit power level the device actually used for transmitting the packet. However in the section on detailed tests, 4.2 on page 48, we will log the status of the device up to ten times a second which will give a better understanding of what is happening with the transmit power level.

3.2 The chip

In section 1.4.1 on page 10, we looked at Q-Free's parking sensor, which houses a NB IoT Ublox chip. The chip is produced by one of the leading producers in chip design and implementation. The chip features specifications in the same lines which NB IoT specifies and we will use this chip for our testing. In this section we will look at the most essential modem commands for the NB IoT chip. Modem commands are often referred to as AT-commands and we will use this notation throughout this paper. As mentioned in section, 2.3 on page 21, there are configurable timers in the network, as well as flags to keep in mind while developing software for low powered sensors. These parameters can be configured with AT-commands and has proven to give good results when taking power usage into consideration. In addition we will show you the AT-command for sending data, and how we intend to structure the data being sent.

In the following subsections we will give a brief introduction to the most important AT-commands taken from the AT Commands Manual for the

NB IoT chip [22]. In section, 4 on page 43, we will give a more detailed description on the parameters used.

3.2.1 Monitoring - NUESTATS

Monitoring is an important part of developing and a way to verify that your software is running as optimal as possible. In our case we wanted a way to monitor the performance of the NB IoT chip, hence we wanted to take advantage of the precise statistics given from the command **AT+NUESTATS**. According to the specifications of NB IoT, there are a number of interesting thresholds and **AT+NUESTATS** will give us this information. In table, 3.2.1, you can see the output of the command.

Power	NB-IoT signal power expressed in tenth of dBm
Total power	Total power within receive bandwidth expressed in tenth of dBm
Tx power	Transmit power expressed in tenth of dBm
Tx time	Elapsed transmit time since last power on event expressed in milliseconds
Rx time	Elapsed receive time since last power on event expressed in milliseconds
Cell id	Physical ID of the cell providing service to the module
ECL	Last ECL value
SNR	Last Signal to noise ratio (SNR) value
EARFCN	Last Evolved Absolute Radio Frequency Channel Number (EARFCN) value
PCI	Last Physical Cell ID (PCI) value
RSRQ	Last Reference Signal Received Quality (RSRQ) value

We believe that using **AT+NUESTATS** will give us accurate results. In addition we will use a high precision multimeter to monitor the current through the NB IoT chip. We will introduce you to this device in section 4.1.1 on page 44. See section, 4.4 on page 50, for any deviations.

3.2.2 Time - CCLK

Time is another tool we often use for developing software. We want to know how long a process endures, and in our case we want to monitor the latency of a send operation from the sensor to the server. With the AT-command **AT+CCLK?** we can read the time from the chip. This time is kept in sync with the network and stored in the MT. The read operation returns a string with the time in the following format "yy/MM/dd,hh:mm:ss+TZ", where the characters represent, year, month, day, hours, minutes, seconds and time zone.

3.2.3 UE behaviour - NCONFIG

With the AT-command **AT+NCONFIG** we are able to decide how we want the UE to behave. HH skal vi bruke denne?

3.2.4 CSQ

signal quality? Different from nuestats?

3.2.5 eDRX settings - CEDRXS

In section, 2.3.3 on page 23, we explained how eDRX works and how we can configure the UE. With the AT-command **AT+CEDRXS** we are able to define how eDRX is performed on the UE. We can issue this AT-command, **AT+CEDRXS: 1, 5, "0101"**, to put the UE into eDRX mode with cycle length of 163.84 seconds.

3.2.6 PSM settings - CPSMS

We can also define how PSM is used on the UE. We can enable PSM, and set the timer Extended TAU Timer (T3412) with this command. If PSM is enabled, the device will enter deep sleep after expiry of the timer Active Time (T3324).

3.2.7 NPOWERCLASS

Force bands to use a given power class?

3.2.8 Create socket - NSOCR

Opens a port on the UE, which enables data transfer on the given port. In our application we will open an UDP socket on a port. The port used can be a number between 0-65535, except for 5683 since this is used by the chip to send coap messages. The command returns a socket number which will be used for trailing send(NSOSTF) commands.

3.2.9 Send command with flags - NSOSTF

This AT-command enabled the device to send UDP packets on a given port with a flag. The command takes a set of parameters and the data in hexadecimal format. The parameters are, socket number(given by the AT-command NSOCR), remote IP address, remote port, flag, data length and

actual data. The flag can be used to specify the behavior of the UE after the data has been transmitted, see table, 3.2.9, for details.

Mode	Description
0	no flags are set
1	Send message with high priority
2	Indicate release after next message. This mode will put the UE into PSM mode directly after transmitting data. Meaning less time in power heavy modes.
3	Indicate release after next message has been replied to.

3.2.10 Signaling connection status - CSCON

We can verify the connection status of the UE with the command AT+CSCON. If the reply of AT+CSCON? is 1 this means that the UE is in RRC connected mode and is able to send and receive packets. If the reply is 0 this means that the UE is in idle mode, hence the device uses little power and is inactive.

3.2.11 Network registration status - CEREG

We can verify the network registration status of the UE with the command AT+CEREG. This command is very useul for detailed tests to check if the UE is still connected to the network under the test.

3.2.12 PSM status - NPSMR

Gives us the status of PSM on the device. It will return 1,1 if the device is in PSM mode which means that the power usage should be very low.

Chapter 4

Testing

In the test chapter we will be looking at tests, with as many scenarios as possible. We will describe testing environment, devices and parameters that effects the tests. At the end we will try to conclude wether or not NB IoT is as promising as it looks on paper.

The test phase began early January and was completed early March. The reason for the long test phase was due to a number of things. First of all the hardware and software related to NB IoT was not available on a stable platform until late 2017, and Telenor and Telia had limited NB IoT enabled base stations. Even at the beginning of 2018 the devices acquired were not production ready, hence our tests were somewhat inadmissible. However the equipment was at a stage where the error rate was very low and the test phase could start. Given the hypothesis we needed to test the chip and the network with good equipment and at several locations. In section, 4.1, we will give you an overview of the testing premissis and the devices used.

4.1 Testing environment

The testing took place at several locations - UiO, at my apartment at Lambertseter and at Q-Free's office by Solli Plass. With three locations and two network providers we could collect more information, hence we got a better understanding of the results.

Not all base stations in these areas are NB IoT enabled, but you will see that different coverage levels will affect the results and we can compare the results to a "perfect world"/specification setup. The coverage of the chip is related to the supplied antenna and the direction of this. With the NB IoT development kit there was an antenna and we could position it the way with the best result. See pictures, 4.1 on the next page and 4.2 on page 45, to see the setup at UiO.

HH add maps with base stations

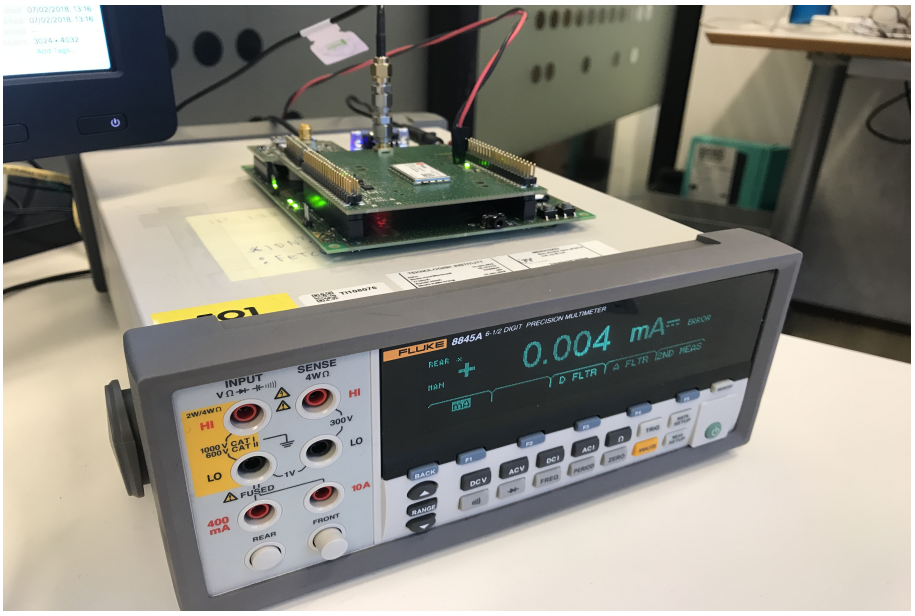

Figure 4.1: NB IoT and multimeter setup

4.1.1 Devices

NB IoT development kit

We are using a development kit from Ublox with their NB IoT chip called SARA N2, see picture 4.3 on page 46. The development kit is connected to a computer over USB and allows us to send and receive data through the serial port of the chip. We will use this setup for both detailed tests such as packet size tracking, coverage tracking and network provider tracking, and use it for longer tests where we continuously send messages with a more practical frequency to the server. The computer will run different python programs to test the different challenges and most of the time we will send **AT+NUESTATS** to the application server so that we can process the information. In certain cases we will also log test runs on the computer for detailed graphs of the behavior of the chip.

Fluke precision multimeter

Q-Free was so lucky to borrow a precision multimeter from ... (HH - OML) which we used to measure the current through the NB IoT chip. With the multimeter we managed to pinpoint different stages of the chip. In addition we added corresponding statistics from **AT+NUESTATS** which increased the precision. As sketched in figure, 4.4 on page 47, we connected the multimeter to the computer using an ethernet cable, enabling us to fetch readings from the device. Depending on the selected precision of the measurements we collected between 30-300 samples per second. The most detailed tests required high sample rate, while we used higher precision

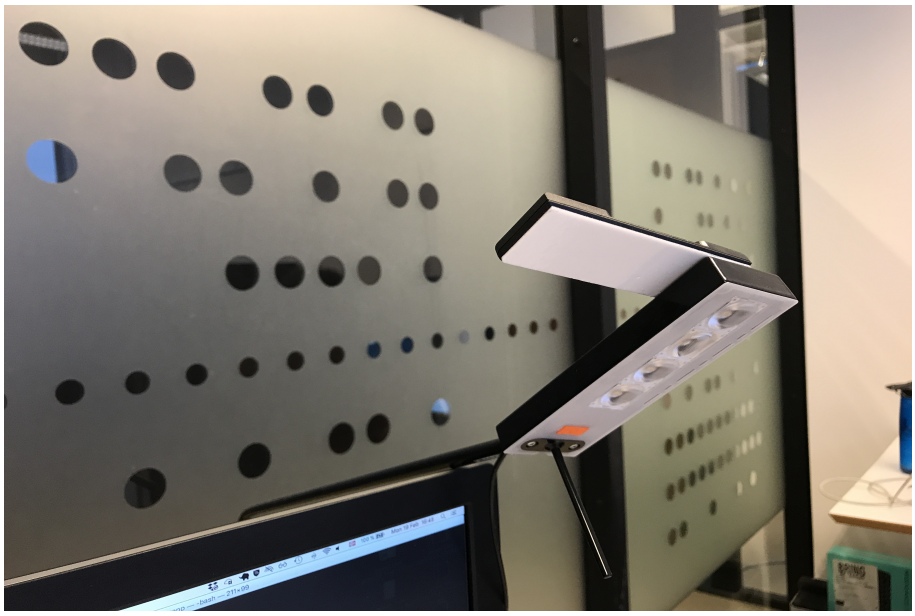

Figure 4.2: NB IoT antenna position

measurements on longer tests. The device responded on commands much like a Read-Eval-Print-Loop (REPL), where you issue a request with a command and the device will respond with an answer. In table, 4.1.1, you can see the commands we used and a short description related to them.

Command	Description
:INIT	Clears old readings
:FETCH?	Fetches all readings from the device. The response is a long string with numbers divided with ","

4.1.2 Writing tests

You should at this point have an idea of the setup and how things are connected, if not, see figure 4.4 on page 47 for description.

We started writing test features for the development kit using python. We setup a connection between the program and the serial port and configure the device to make sure that everything is according to our setup, see code 4.1.

Listing 4.1 Development kit initiation

```
1 uart_modem = serial.Serial('/dev/tty.usbserial-146100', 9600
      , timeout = 0)
2 uart_modem.close()
3 uart_modem.open()
4 uart_modem.flushInput()
5 uart_modem.flushOutput()
```

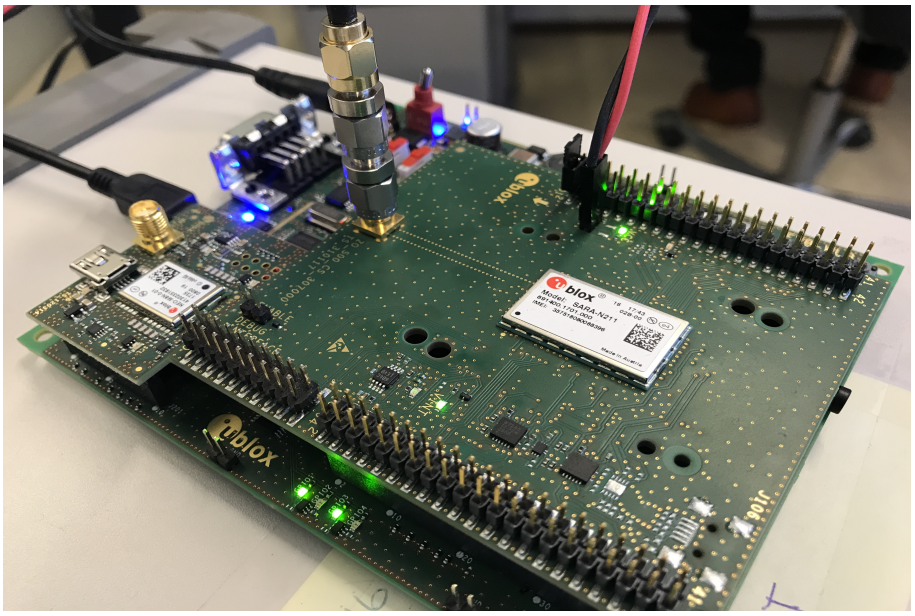

Figure 4.3: Ublox NB IoT development kit

We started experimenting with different AT commands and created small programs to test the functionality of the chip and the connection towards the server. We created a simple program which sent **AT+NUESTATS** to the application server and went on transforming this program to helper methods we could use later on. A sample packet with a COAP header would look something like the example, 4.2, where the payload is converted to hexadecimal. We used some time creating the correct COAP packet for the server to accept it as a COAP packet. If you don't want to use COAP you can simply fill the payload with data without the COAP header, but this means that you are missing the features of COAP which could be beneficial for your application. Dropping the COAP header means that you are sending a pure UDP packet through the network and it is harder to verify the packet when it is received on an eventual application server.

Listing 4.2 Sample transmit 158.39.77.97:5683, 88 bytes

```
1  AT+NSOSTF=0,"158.39.77.97",5683,0x0,88,"500230
2  30B131112AFF2D313039365F2D313034325F3233305F39
3  3732315F37363838305F33343433393435325F315F3132
4  335F363235325F39385F2D3130385F31382F30322F3139
5  2C31363A31363A35362B30305F31325F"
```

We proceeded working on communication towards the multimeter, which was quite easy. The multimeter was given an IP address which was related to the IP address of the connected machine. Creating a socket to this IP address in python and connecting to it was simple. The test programs would start off by clear all buffers with **:INIT** and then fetch for data on a regular interval with **:FETCH?; :INIT**. With each reading we converted the

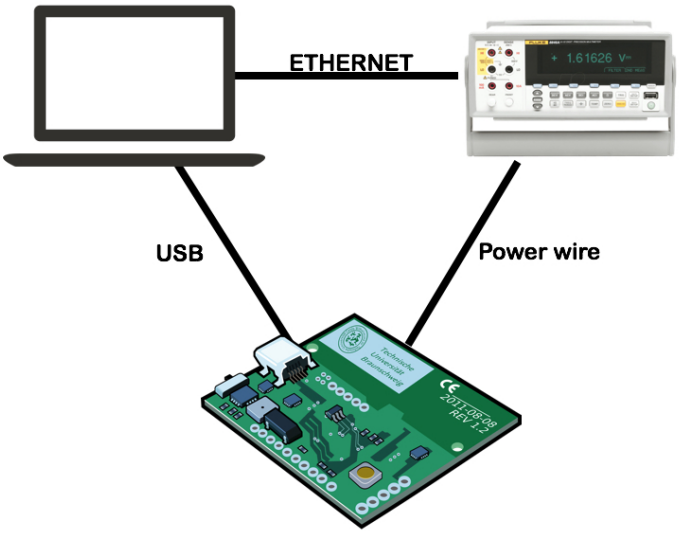

Figure 4.4: Lab setup [19] [5] [9]

string into a list of numbers which we appended to a global list of readings. We found that fetching for new data every 5th second was a good tradeoff between fetching all the time and very rare. We added these functions to the helper file, **nbiot_labtest_helpers.py**, and went on writing a couple of programs to monitor the behavior of the chip.

nbiot_labtest.py This program continuously transmits packets over NB IoT - either status packets towards the server or random data of different sizes. It enabled us to test different scenarios and log the results in a representable way with graphs. The program uses The program takes in a list of parameters described in table, 4.1.2.

Parameter	Description
-id	ID which you want to POST towards the server. Default: 0
-gn	Output graph name. The supplied name will lead the heading, followed by a list of the rest of the parameters.
-d	Set the desired delay between each iteration. Default: 5
-i	Set the desired iterations. Default: -1 which means that the program will loop until the user presses ctrl+c
-b	Set the desired length of the payload in bytes. If 0 is requested, statistics from **NUESTATS** will be sent to server. Default: 100
-r	Set release indicator. Default: false
-l	Set device logging. Default: false

4.2 Short term tests

In this section we will investigate details about how the NB IoT chip handles normal usage and edge cases where non default behavior is activated. What defines normal behavior? Normal operation of the UE is when ECL level is 0 and coverage better than -100dBm. When the UE has bad reception it may retransmit packets and operation like this over longer periods is not sustainable. With normal operation we also expect the UE to transmit packets with release indicator set and PSM activated. According to the specifications the device will enter deep sleep mode trailing a each transmit.

We define short term tests as a logging sequence over a short period with frequent data points. The duration of the tests are under ten minutes and the sample rate is high, over 300 samples per second for some tests. These tests will show you how the device performs on a detailed level and will give a good understanding og the transmit process. The reason for including these tests are related to the actual power usage over a short period. By defining power usage statistics for different kinds of transmit we will be able to normalize the data and give an estimation of expected lifetime based on our tests. The detailed tests also will show us if the network providers follow the specifications. It is very interesting to follow PSM, connection status, coverage and power usage over a short period and see how the device handles different scenarios.

We have sectioned the results into categories and we will refer to a set of graphs located after the section. In this way you will be presented with one problem at a time while pointing to the same graphs. To explore the full potential of the graphs you may also visit the thesis web page[7] and visit the link with the related caption of the graph.

4.2.1 Latency test

4.2.2 Coverage test

ECL modes

4.2.3 Packet size

Try different packet sizes and check power usage (are there any perfect packet sizes? - I think 64B is one datablock)

4.2.4 Reboot of sensor?

4.2.5 Cell selection test

4.2.6 Downtime test

4.3 Long term tests

A device operating over long time is prone to many bugs and the quality of the software is essential. In this section we will give you an overview of how the network handles continuous transmits over a longer period. We have tested the device at all locations with both network providers where device managed to connect to the network. We define long term test as a set of operations over a long period compared to the detailed tests, typically one or several days. When trying to test many scenarios, the test period per network provider and location is limited. However, the results are comparable to the short term tests and there are clear indications of the networks pros and cons. The reason for the long term tests is to monitor the normal behavior of the chip over a longer period, which would give us statistics at a more abstract level and combine the results with the short term tests. With this relation we will give you an overview of the most common states of a device and how different aspects reflect the behavior.

Also in this section we have divided the results in categories with the related graphs located after the section.

4.3.1 Coverage and latency

4.3.2 Operating modes

4.3.3 Cell selection

4.3.4 Uptime

4.4 Deviations

4.4.1 Imprecise clock

With the command **AT+CLOCK?** we get the time from the network. This timestamp is not a precise time, but we believe that it is good enough to test the specified 10 second latency in the NB IoT network. When doing the detailed tests we tried to verify that the timestamp received from the network was accurate. In most cases the time was very precise, only with a deviation of 10ms. HH verify

4.4.2 Network load

Since the NB IoT network was not in production at the time of the tests, the network was not heavily loaded. This means that the latency could be a bit higher with more general activity on the network. When the network is available the latency should be low and that is our impression on the NB IoT network as well. HH verify

4.4.3 Network density

Since the network was not in production, the density of the cell towers with NB IoT were limited. In the future all cells will implement NB IoT, meaning that the uptime will increase. The coverage will probably also increase since we could possibly connect to a closer cell tower if the density is higher.

load on server + server park

4.5 Early test faults

Changing cell when not in bad coverage Looses connection to network and sudden bursts of packets are received Some packet loss, which should not appear since the load on this network was low in the test period.

Part III

Conclusion

Chapter 5

Results

Bibliography

[1] LoRa Allience. *A technical overview of LoRa® and LoRaWAN™*. URL: http://www.semtech.com/wireless-rf/iot/LoRaWAN101_final.pdf (visited on 02/23/2017).

[2] Cisco. *IoT Growth*. URL: http://www.cisco.com/c/en/us/about/security-center/secure-iot-proposed-framework.htm (visited on 02/01/2017).

[3] UN DESA. "World Population to 2300." In: *United Nations Department of Economics and Social Affairs, New York, NY [available at www. un. org/esa/population/publications/longrange2/WorldPop2300final. pdf]* (2004). URL: http://www.un.org/esa/population/publications/longrange2/WorldPop2300final.pdf (visited on 02/01/2017).

[4] Ericsson. *LTE: AN INTRODUCTION*. URL: https://www.ericsson.com/res/docs/2011/lte_an_introduction.pdf (visited on 01/14/2017).

[5] *fluke8846a-bench-multimeter-500x500.png (500×335)*. http://3.imimg.com/data3/AR/PU/MY-754870/fluke8846a-bench-multimeter-500x500.png. (Accessed on 02/19/2018).

[6] GSMA. *Mobile internet of things low Power wide Area Connectivity*. URL: http://www.gsma.com/connectedliving/wp-content/uploads/2016/03/Mobile-IoT-Low-Power-Wide-Area-Connectivity-GSMA-Industry-Paper.pdf (visited on 02/10/2017).

[7] Henning Håkonsen. *Thesis webpage*. URL: henninghaakonsen.me.

[8] Thor Hansen. *History of GSM*. URL: https://snl.no/GSM (visited on 01/19/2017).

[9] *inga_gruen.png (2400×1349)*. https://openclipart.org/image/2400px/svg_to_png/219943/inga_gruen.png. (Accessed on 02/19/2018).

[10] Oslo Kommune. *Statistikk over husholdninger*. URL: https://www.oslo.kommune.no/politikk-og-administrasjon/statistikk/befolkning/husholdninger/ (visited on 03/09/2017).

[11] LinkLabs. *LTE eDRX and PSM Explained for LTE-M1*. URL: http://www.link-labs.com/blog/lte-e-drx-psm-explained-for-lte-m1 (visited on 03/17/2017).

[12] lte. *LTE Network Infrastructure and Elements*. URL: https://sites.google.com/site/lteencyclopedia/lte-network-infrastructure-and-elements (visited on 01/25/2017).

[13] Lyse. *Automatiske strømmålere*. URL: http://www.lysekonsern.no/prosjekter/automatiske-strommalere-article565-321.html (visited on 02/10/2017).

[14] Nokia. *EPC and components*. URL: http://www.rcrwireless.com/20140509/diameter-signaling-controller-dsc/lte-mme-epc#prettyPhoto/1/ (visited on 01/25/2017).

[15] Nokia. *LTE evolution for IoT connectivity*. URL: http://resources.alcatel-lucent.com/asset/200178 (visited on 01/24/2017).

[16] OECD. *Smart Sensor Networks: Technologies and Applications for Green Growth*. Dec. 2009. URL: https://www.oecd.org/sti/ieconomy/44379113.pdf (visited on 04/06/2017).

[17] "Ola."

[18] *Oslo*. URL: https://no.wikipedia.org/wiki/Oslo (visited on 08/29/2017).

[19] *pc.png (350×280)*. https://www.reactos.org/images/pc.png. (Accessed on 02/19/2018).

[20] Yevgeniy Sverdlik. *Custom Google Data Center Network Pushes 1 Petabit Per Second*. June 18, 2015. URL: http://www.datacenterknowledge.com/archives/2015/06/18/custom-google-data-center-network-pushes-1-petabit-per-second/ (visited on 01/19/2017).

[21] "Ublox NB-IoT chip." In: (Oct. 9, 2017).

[22] "Ublox NB-IoT chip - AT commands manual." In: (Oct. 3, 2017).

[23] Wikipedia. *2016 Dyn cyberattack*. URL: https://en.wikipedia.org/wiki/2016_Dyn_cyberattack (visited on 03/09/2017).

Acronyms

3GPP 3rd Generation Partnership Project. 3, 6, 14, 17

A Ampere. 22

AuC Authentication Center. 7

CAT-M LTE Cat-M. 12

dBm Decibel / referenced to milliwatts. v, 21, 22, 25, 26, 39

DDoS Distributed Denial-of-Service. 19, 20

DL Down Link. 6, 22

DRX Discontinuous Reception. 21, 23–25

EARFCN Evolved Absolute Radio Frequency Channel Number. 40

ECL Coverage Enhancement Level. v, 25, 26, 38–40, 48, 49

eDRX Extended Discontinuous Reception. iii–v, 23–25, 41

eMTC enhanced Machine Type Communication. 14

eNB Evolved Node B. 7, 8

EPC Evolved Packet Core. iii, v, 6–8

ETSI European Telecommunications Standards Institute. 17

GSM Global System for Mobile communications. 13, 18, 20

GSMA GSM Association. 8, 12

HLR Home Location Register. 7

HSS Home Subscriber Server. 7, 19

IoT Internet of Things. v, 3–5, 8, 10, 12, 14, 17–20

IoT Platform IoT Platform. 20

IP Internet Protocol. 46

IPv4 Internet Protocol version 4. 17

ITS Intelligent Transportation Systems. 9

Kbit/s Kilobit per second. 14

LoRa . v, 14, 15

LPWAN Low Power Wide Area Networks. v, 4, 5, 8–10, 12–15, 17, 19, 21, 24

LTE Long-term Evolution. iii, 4–7, 10, 12–14, 17–21, 23

LTE-M1 . 14, 15, 23

mA Milliampere. 22

Mbit/s Megabit per second. 12, 14

MME Mobile Management Entity. 7, 19, 25

MT Mobile Terminal. 40

mW Milliwatts. 22, 26

mW/s Milliwatts seconds. 26

mWh Milliwatt hour. 22

NACK negative-acknowledgement. 39

NAT Network Address Translation. 20

NB NarrowBand. 17

NB IoT NarrowBand IoT. v, 1, 3, 5, 10, 12–15, 17–27, 31, 38–40, 43–48, 50

NPM Node Package Manager. 32

OECD ORGANIZATION FOR ECONOMIC CO-OPERATION AND DEVELOPMENT. 8

PCI Physical Cell ID. 40

P-GW Packet Data Network Gateway. 7, 8

PSM Power Saving Mode. iii, iv, 19, 21–25, 39, 41, 42, 48

QoS Quality of Service. 7

REPL Read-Eval-Print-Loop. 45

RSRQ Reference Signal Received Quality. 40

S-GW Serving Gateway. 7, 8

SNR Signal to noise ratio. 40

T3324 Active Time. 23, 41

T3412 Extended TAU Timer. 24, 41

TDMA Time division multiple access. 24

UDP User Datagram Protocol. 41, 46

UE User Equipment. iv, 4, 6, 7, 10, 21–26, 41, 42, 48

UL Up Link. 6, 22

WAN Wide Area Network. 19