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

Contents

Listings

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisors, Ola Martin Lykkja and Yan Zhang, for their support. In addition, I would like to thank Q-Free for giving me the opportunity to work with Ola Martin, as well as supplying me with the necessary equipment to fulfill my work.

A special thank you to Telia which let me perform tests on their network. This gave the thesis an additional dimension and improved the results.

I would also like to thank my family, friends and especially my girlfriend, for supporting me through the last two years completing my master's degree.

Abstract

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. There is a demand for a new technology which enables communication with sensors and other low powered devices. This thesis resembles the work and research on a set of technologies suited for this communication, with an emphasis on NB IoT.

Please visit henninghaakonsen.me for the latest copy of the thesis, along with the related research and results. The web application is present to support my thesis with extra reading material as well as being a tool for my tests.

Chapter 1

Introduction

1.1 Goal

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 concerning 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 test NB IoT and it lead us to a motivational process, but also at times frustrating. There are several papers on how NB IoT has great power saving features, referring to the 3GPP specification, but there are few 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 guidelines.

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 applications. 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 of this kind of network are incredible. However, the path towards this goal has not been easy. There have been attempts of similar networks, but they usually are too fixed or expensive and the competition has been more of an

obstacle than an advantage. With too many poor solutions the growth has been rather low compared to what we now see a start too. 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 [12], Cisco has made an illustration of the expected growth of devices towards 2020 1.1.

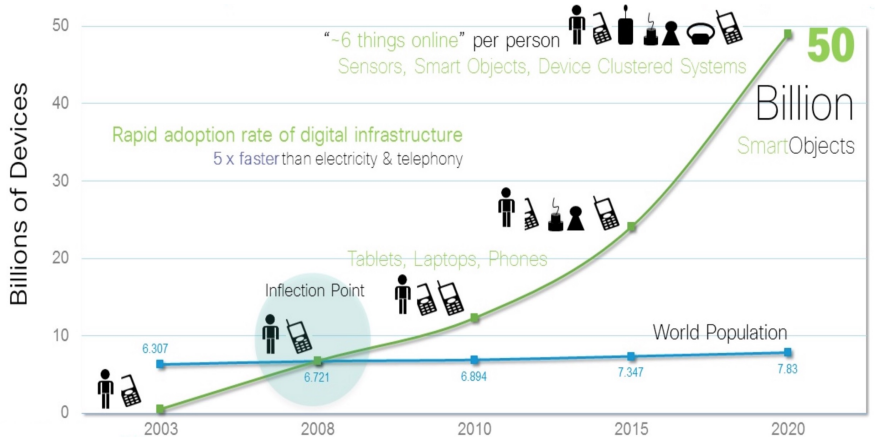

Figure 1.1: IoT Growth [11]

Currently, we are at a point where it all comes down to how these LPWAN perform, which we will have a look at in this thesis. Up until now, IoT devices have 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 100 bytes 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 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[1.1] of the cost of 300 devices from a parking sensor use case shows that NB IoT outperforms

the current solution. On the left-hand side, you see LTE specific additions related to installation and monthly fees from the LTE embedded device. The monthly fees and maintenance come from the rent of the hardware position. When using the LTE embedded device Q-Free needs to rent a space for it and pay for maintenance and power for that particular place. Note that this is just an outline of one example and certain parameters may change depending on hardware, installation company, and network provider.

Sensors per base station (LTE)	100
Subsciption cost - monthly (LTE)	€ 50
Installation cost - one time (LTE)	€ 1 500
Montly fees and maintenance (LTE)	€ 250
Number of years	3
Number of sensors	300

	NB IoT	**LTE**	
One time cost			
Cost of sensor	60	50	
Installation cost per sensor	0	5.00	
Montly cost			
Subscribtion	0.2	0.50	
Subscribtion 3years	7.2	18.00	
Fees and maintenance	0.3	0.83	
Fees and maintenance 3years	10.8	30.00	
Total cost per sensor	€ 78	€ 103	
Total cost 300 sensors	€ 23 400	€ 30 900	+32 %

Table 1.1: Cost comparison between NB IoT and LTE with of 300 devices. Euro calculated at 9.5 Norwegian kroner

In addition to lower cost and lower complexity, there is a demand for increased coverage. Vehicles driving in vast areas, monitoring sensors positioned several meters underground 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 today's 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 people without development experience more easily can learn. This will be an additional cause of growth in the number of IoT devices and is yet another reason why we need to properly implement LPWAN to meet the demands.

The idea of some of the new Low Power Wide Area Networks (LPWAN) is to use the same hardware as LTE and if the customer wants it they will also support 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 18. This new technology supports an extreme amount of devices as well as

them being in a secure environment. Security is a big concern when discussing general network related topics and we will introduce you to some security measurements provided by NB IoT. 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.

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"[39]. 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 [21] 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 not 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 consist of some particular nodes for the network to connect to User Equipment (UE) and Internet. We call this structure Evolved Packet Core (EPC) [26]. 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 NB IoT. 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[16]. 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.

1.3.4 Evolved Packet Core

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. In this section we will give a short introduction to the key features of each component.

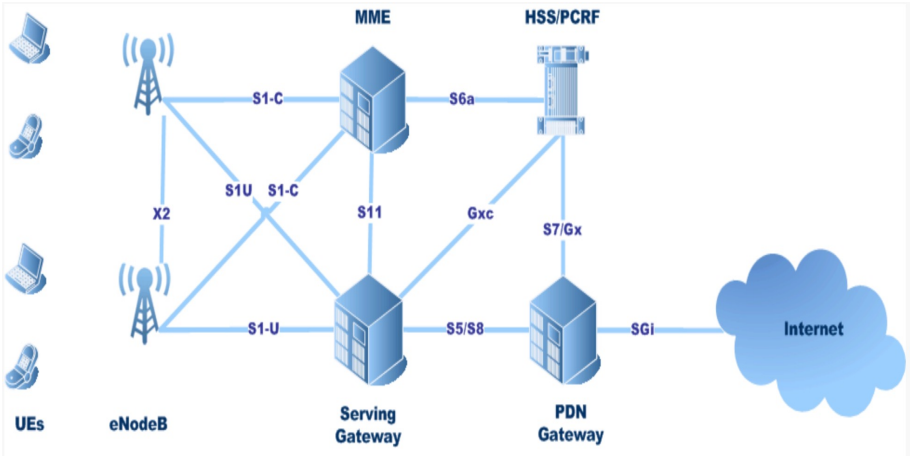

Figure 1.2: Evolved Packet Core overview [32]

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 the 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, 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 LPWAN devices. 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, 1.5 on page 12, 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 [34]. 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 combine 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/LTE networks. These networks provide average coverage, cost and power consumption, hence 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 following page, some bullet points are presented for each area and in the next section we will point out use cases for some of the areas.

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

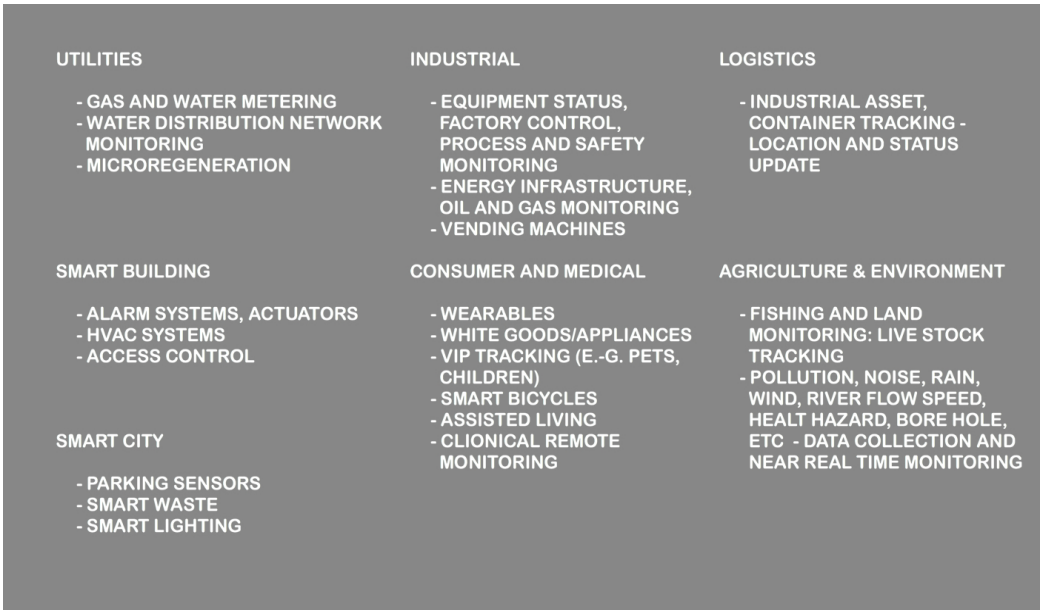

Figure 1.3: LPWAN applications [20]

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 [27]. 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 requirements for their application. The communication between the sensor and the server is outlined in figure 1.5 on the next 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 [35]

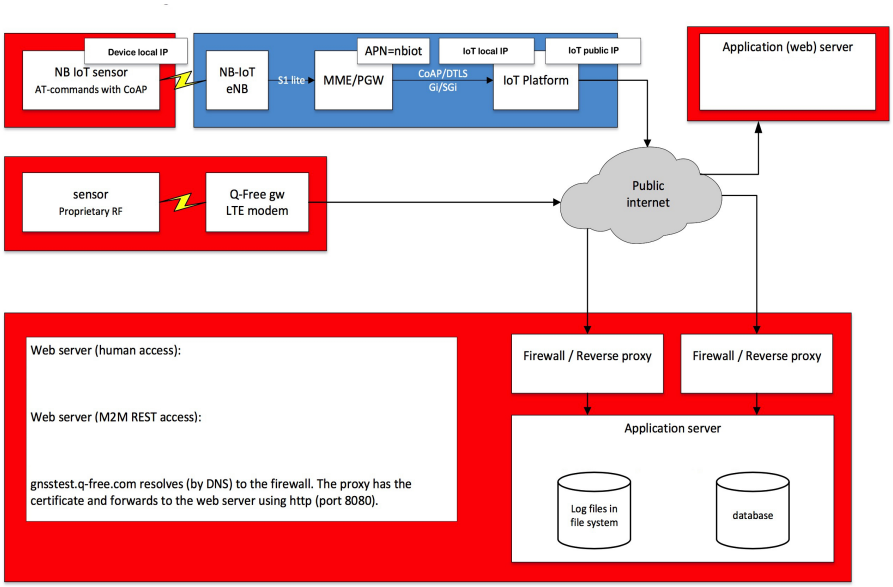

Figure 1.5: Outline of sensor communication [35]

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 [20]. 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, 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.

1.5 Developing wireless technologies

1.5.1 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 5. 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 at all locations which 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 real time 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, but never the less it is important to not suppress the 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.

In the following sub sections we will introduce you to the current comtenders in the competition of concurring the LPWAN market.

1.5.2 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 page 14 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 15. LTE-M1 can be used in the same matter.

1.5.3 LoRa

LoRa [9] 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.

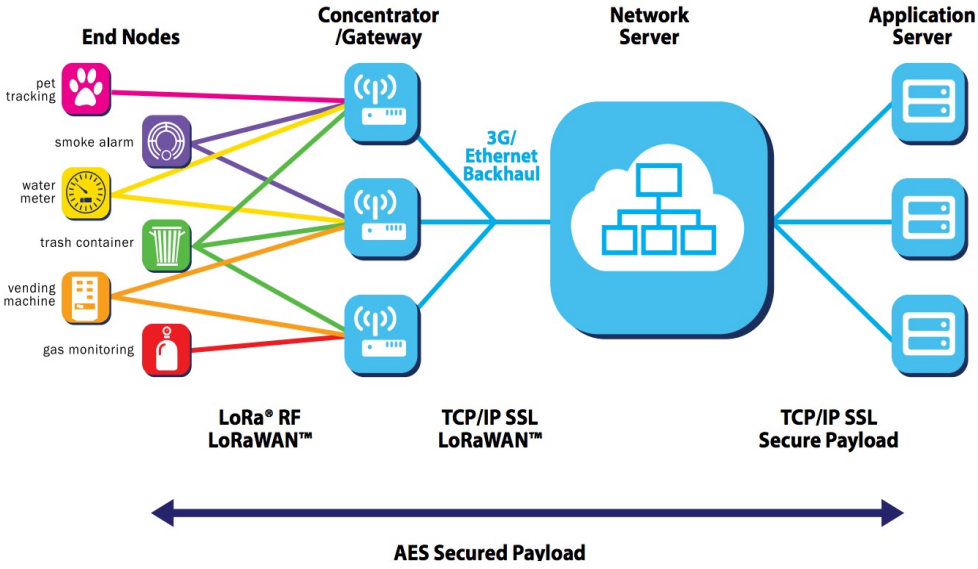

Figure 1.6: LoRa overview [9]

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 technologies [33]

Chapter 2

A dive into NarrowBand IoT

We will focus on a general approach towards NB IoT as technology and will test both Telenor and Telia's solution. In section, 5 on page 59, we will discuss how and where we did the tests as well as giving you examples and results.

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. At the same time, Telia had just announced that they had deployed their NB IoT network, but it was not yet production ready. At the time of delivering this thesis, neither Telenor or Telia had NB IoT in production. Without any specific date, they could say that the network would be deployed within a short period of time which indicates that the networks were not in production at the time of our tests, but at a stable state.

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 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 the following sections, we will introduce you to the structure of NB IoT and the design requirements. We have not emphasized security in this thesis and will only be mentioned in the appropriate sections.

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 [11], 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 [23]. 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 23.

- 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 [49]. 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 5 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 Deployment modes

NB IoT occupies 180kHz bandwidth and there are currently three ways to integrate NB IoT with LTE. The available deployment modes are standalone, in-band and guard-band - see figure 2.1 for illustrative definition. Standalone operation uses a dedicated GSM/LTE channel, utilizing the ability to deploy NB IoT in GSM as well as LTE. In figure 2.1 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 network provider. The reason for this is because of interference between the LTE and NB IoT sections in the carrier. When deploying in-band NB IoT allocates three resource blocks of an LTE channel.

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.

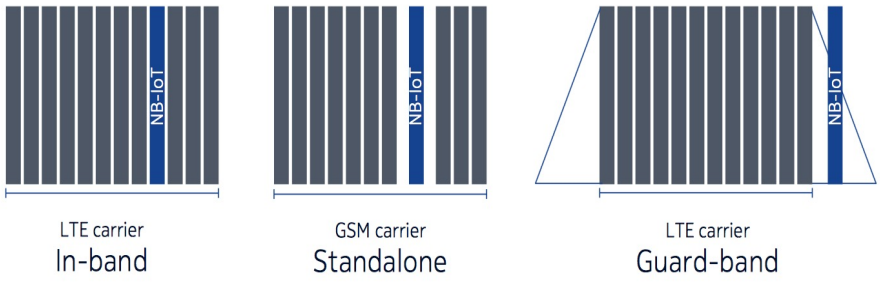

Figure 2.1: NB IoT deployment overview [33]

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 Discon-

tinuous 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 you can see an outline of the communication process for a device. Radio Resource Connection (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.

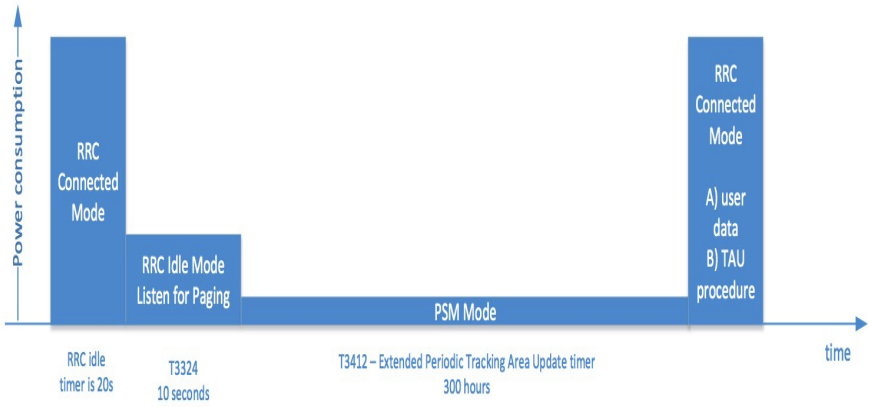

Figure 2.2: NB IoT transmit overview [35]

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 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 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 $230mA$ [47]. We can convert dBm to mW by calculating $10^{xdBm/10}$. Keeping $+23dBm$ in mind gives us $10^{23/10} = 200Milliwatts(mW)$. We can also convert from mA to mWh by multiplying mA with the voltage of the chip at 3.3 volts, gives us $230Milliampere(mA) * 3.3V = 759mWh$.

NB IoT operates between $-40dBm$ and $+23dBm$, where $-40dBm$ equals $10^{-4} = 0.0001mW$ and $+23dBm$ equals $10^{2.3} = 200mW$. 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 $6mA * 3.3V = 0.0198mWh$ [47].

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 (RAI), 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 device's status. Given that the network supports RAI the battery life will be extended, since we only send data for a short period rather than waiting for downlink data which in many applications is not needed.

If RAI is not set the device will stay in RRC connected mode until it is notified otherwise from the network. This process happens after the RRC timer and requires two-way communication which you will see in our tests. The device acknowledges the release and goes into RRC idle mode.

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. In our tests we saw the device was only in RRC idle mode for approximately one second before entering eDRX, followed by PSM mode.

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 therefore 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 to 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 facing 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 itself 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.

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

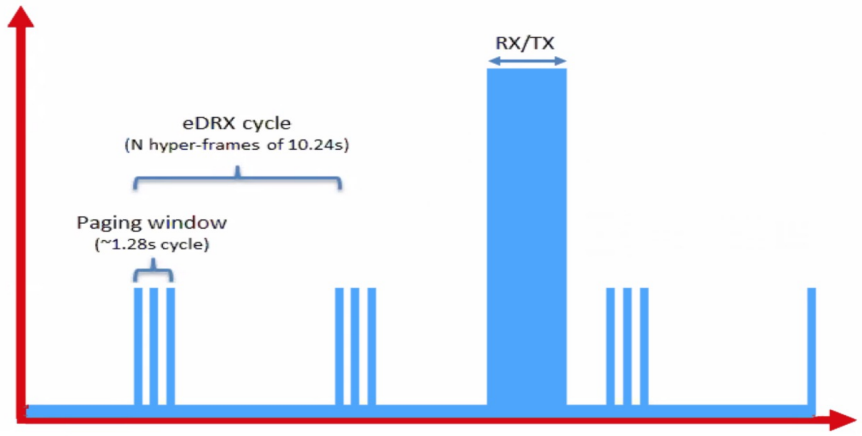

Figure 2.3: eDRX operation [25]

that paging is not in use. In section, 2.3.5 on the next 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 metadata 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 or eDRX period. If the server 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 Coverage Enhancement Level

As mentioned in section, 2.3.1 on page 21, 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 [41], coverage better than this resolves to ECL0. OML Telenor?

ECL is not important by itself, but an application fetching the ECL of the device will gain a lot of information about the reception, hence it might reconfigure the setup of the transmit process for example. Based on the signal strength the device will retransmit data to improve Transmit Success Rate (TSR) towards the application server. 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. In section, 5.1.4 on page 66, we will give you insight into the transmit process of two transmits, one with ECL0 and one with ECL2. The difference is big and will definitely affect the battery lifetime of the device.

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, $(10^{2,3})/(10^{-4}) = 2000000$. 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 200mWh for 5 seconds, resulting in a power usage of $((200mWh/60/60) * 5S = 0.27mWh$, 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, we see an outline of how the device chooses ECL mode dependent on the signal quality. In theory the device should lower its transmit power equal to the signal strength. The graph represents the theory behind the adjustment of the transmit power and you will see how this actually works in section, 5 on page 59.

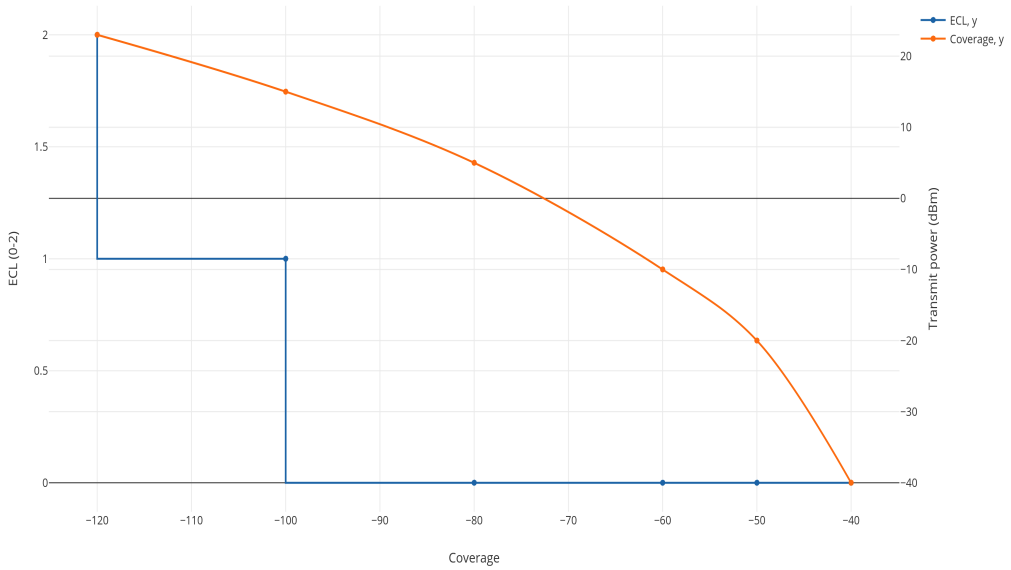

Figure 2.4: Relation between ECL and transmit power

2.4 The current market and deployment strategies

We have discussed several applications which suit LPWAN and there is a demand for chips which implement NB IoT or LTE-M1. In this section we will give you a brief introduction to the key firms involved with LPWAN and how the market looks like early 2018.

2.4.1 Manufacturers and stakeholders

There are many companies which manufacture hardware for the mobile industry. Historically Neul(UK), HiSilicon(China) and Qualcom(USA)

has been the main chipset producers. Neul and HiSilicon are owned by Huawei which makes them one of the biggest producers at the moment. In addition, there are three companies, Ublox, Telit and Quecktel, which have been using "Huawei" chipsets, only adding their custom software. At the moment the agreement towards Huawei has expired and the three software companies will need to fetch hardware from another manufacturer, which will most likely be Qualcomm.

Currently, there is a problem surfacing in USA which stresses the market. In Europe, GSM/GPRS has been the only alternative, while in USA there has been three network technologies, GSM/GPRS, IS95 and TDMA. Because of the different technologies, the process of developing chipsets has been difficult and they have now decided to take down IS95 and TDMA. Most users have gone over to LTE, but there are many companies which use IS95 and TDMA for low powered devices which now has no good alternative. The solution is to move over to another LPWAN, which for most companies will resolve to LTE-M1 because of the opportunities and spread of applications this technology provides. Because of the stress in the market and the increased demand for general LPWAN, all chipset manufacturers began their work on NB IoT and LTE-M1 chipsets. Most of the companies did however reuse their LTE chips to produce LPWAN chipsets. The chip we will be using is produced by Neul, with software from Ublox, and suffers from the process of reusing hardware originally developed for LTE. In section, 3.2.2 on page 35, we will give you a detailed introduction to how we used the chip, but it is worth noting that the chip was not production ready.

Other companies have seen the development of this market and started to take up the fight against the biggest companies. Nordic Semiconductor is one of these companies, and originally has produced chips for other technologies like Bluetooth. According to a talk they gave, they have used 1.000.000 working hours on chips with LTE-M1 and NB IoT, which is developed from scratch focusing on LPWAN technology. They have seen the stressed market in USA and will begin production of LTE-M1 chips this year, followed by NB IoT in 2019 [35].

Based on the research done in this thesis it will be interesting to see the next generation NB IoT chipsets. These will hopefully be more stable with an even bigger emphasis on power management. In the tests we have performed, the results are very promising given the prerequisites.

2.4.2 Deployment

In section, 2.2.1 on page 19, we looked at the three different deployment strategies in the current LTE frequency spectrum. The choice between

in-band, guard-band and standalone operation is based on the network providers frequency budget and often a tradeoff between satisfying mobile customers or low powered devices often deployed by companies. In addition to the operation mode the network providers offer an IoT platform, which we discussed briefly in section, 2.2 on page 18. In this subsection we will introduce you to which operation mode Telenor and Telia has used and explain what an IoT platform gives to the two parts of the technology stack, the customer and the network provider.

Telenor has deployed their NB IoT network in stand-alone operation mode, which means that they are using one dedicated LTE channel[35], hence the network does not suffer from noise from LTE. However, it occupies a large frequency area for which other users could use. In our results we saw that the stability of the network was good and performed close to what the specification dictates.

Telia has deployed their NB IoT network in in-band operation mode which means they occupy three resource blocks in an existing LTE channel. As discussed earlier this results in potential more frequent interference between NB IoT and LTE devices. In our results using Telia's network we saw a slightly more unstable behavior which might be related to the fact that they are using in-band operation mode. On the other side, this setup is closer to what we expect the situation to be in a production network with other users, hence the performance difference is arguably logical.

IoT platform can be rewarding for both customer and network provider. As stated in section, 2.2 on page 18, and according to the network providers, an IoT platform may enhance the security of the communication. However, in order to utilize the full potential of an IoT platform the network provider will have to open the content of the data and do a repacking of the data. In our opinion, this is not as secure as an end to end transmission. And another motivation for a network provider to promote the IoT platform is that if they have access to the customer's data they can perform analysis on it and give customer based subscriptions or potentially sell your information. A good example of usage of these kinds of data is the possibility to stream music for free, which was enabled by Telenor and Telia in 2018. The networks has to view the data to know that the data is related to music, ie Spotify, Tidal, to give the customer this data for free. Likewise the network providers will have access to the data in the IoT platform which is not preferable for many customers. However, if your application sends non-confidential information and you want the

application up and running quickly the IoT platform might enhance your application. There are always tradeoffs to using an abstract implementation, like high order programming languages or something like an IoT platform.

2.5 Challenges

There are limitations to NB IoT which are imminent that the customers and the developers understand. With the kind of bandwidth we have today there is less concern with the amount of data transmitted. NB IoT introduces developers to a new kind of applications which should be provident and fully operate for potentially ten years - this introduces difficult programmatic problems. We have stumbled upon some of these issues and we have included a set of best practice guidelines based on the experience we got from the tests in section, 7.1 on page 95.

The following part of the thesis will cover the project related to the thesis - the testing phase. This part was obviously the most demanding part of the thesis and we have tried to stress the network in ways not intended. In addition because of the pre-released hardware and software[2.4.1] we need to state that the results, especially for the long-term test, should be considered as base results and will change as hardware and software evolves. We will test both Telenor and Telia as they provide NB IoT in Oslo, and it is important to note that we are trying to give an objective statement about NB IoT and not compare the two network providers.

Part I

The project

Chapter 3

Where, why and how?

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, 5.1 on page 59, 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 Testing environment

The project was performed in Oslo and the test were executed several locations - UiO, at my apartment at Lambertseter(Avstikkeren 7) 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, 3.1 and 3.2, to see the setup at UiO. In section, 3.1.1 on the next page, you can see the distances between the different test sites. Unfortunately we have not been able to get the location of Telenor's base station at UiO, but it is likely that it is located at the same place since there are many cells at the location of Telia's cells.

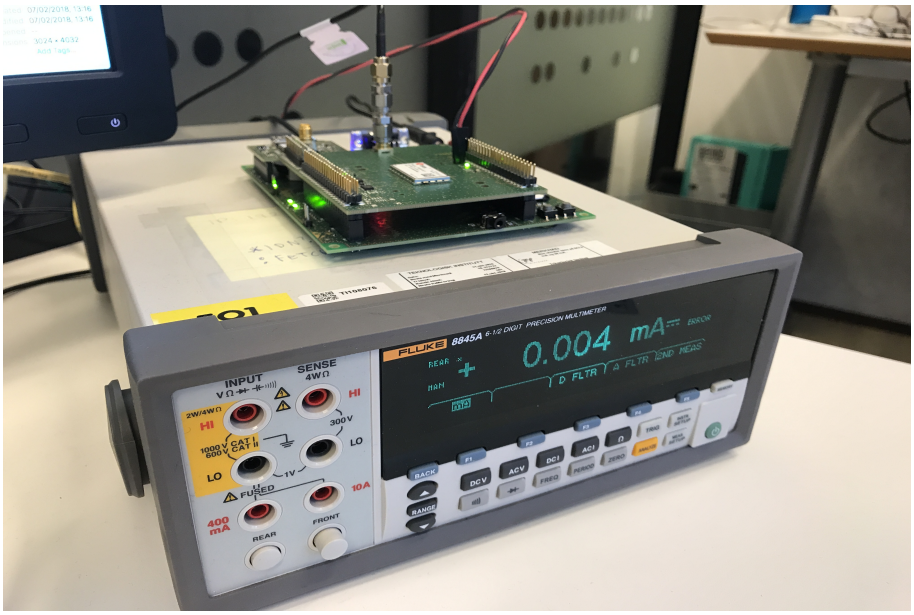

Figure 3.1: NB IoT lab setup

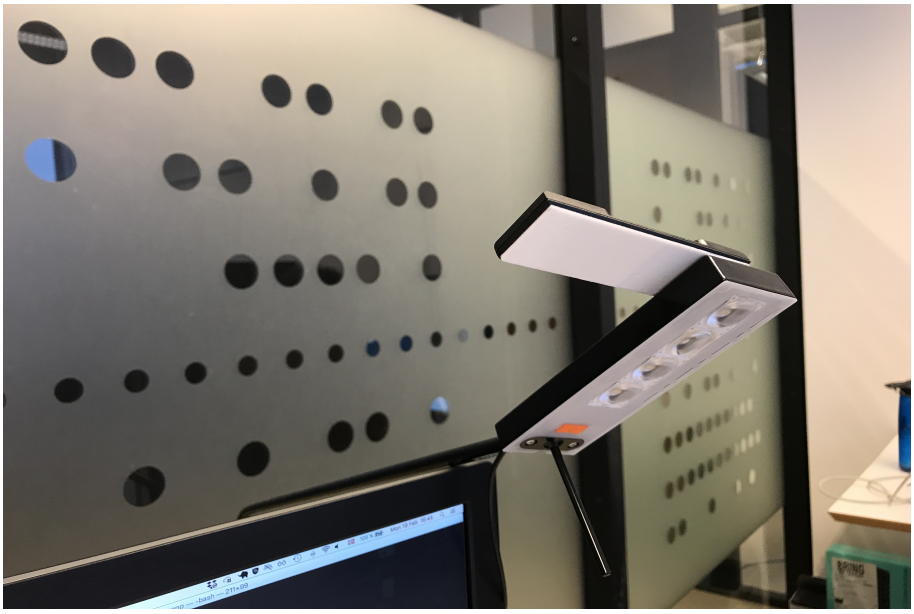

Figure 3.2: Development kit, antenna position

3.1.1 Maps and distances

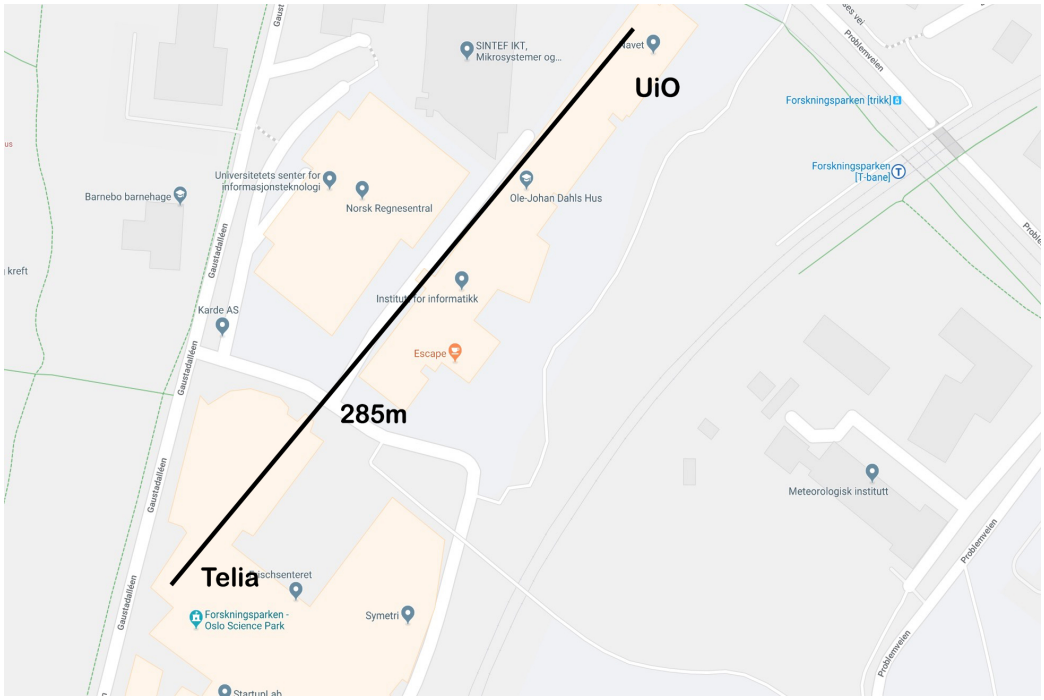

Figure 3.3: Device distance map over IFI, UiO. Referring to www.finnsenderen.no[18], we believe Telenor's cells are located close to Telia's, but have not been able to confirm this by Telenor.

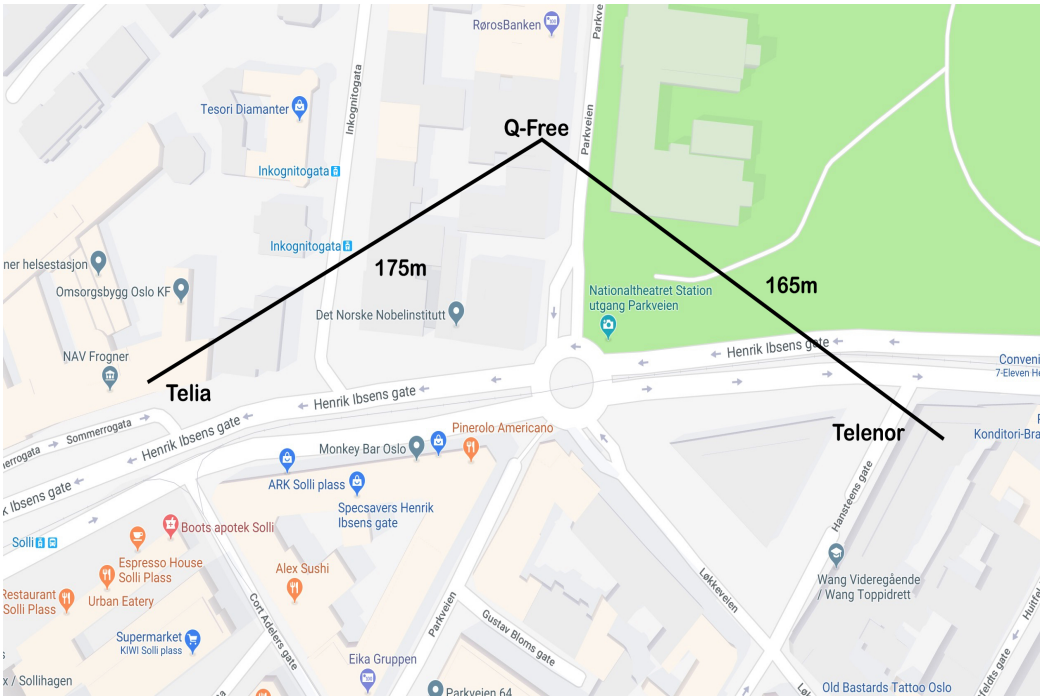

Figure 3.4: Device distance map over Q-Free. The distances are similar, but there are more obstacles between Q-Free and Telia's eNB which affects the reception.

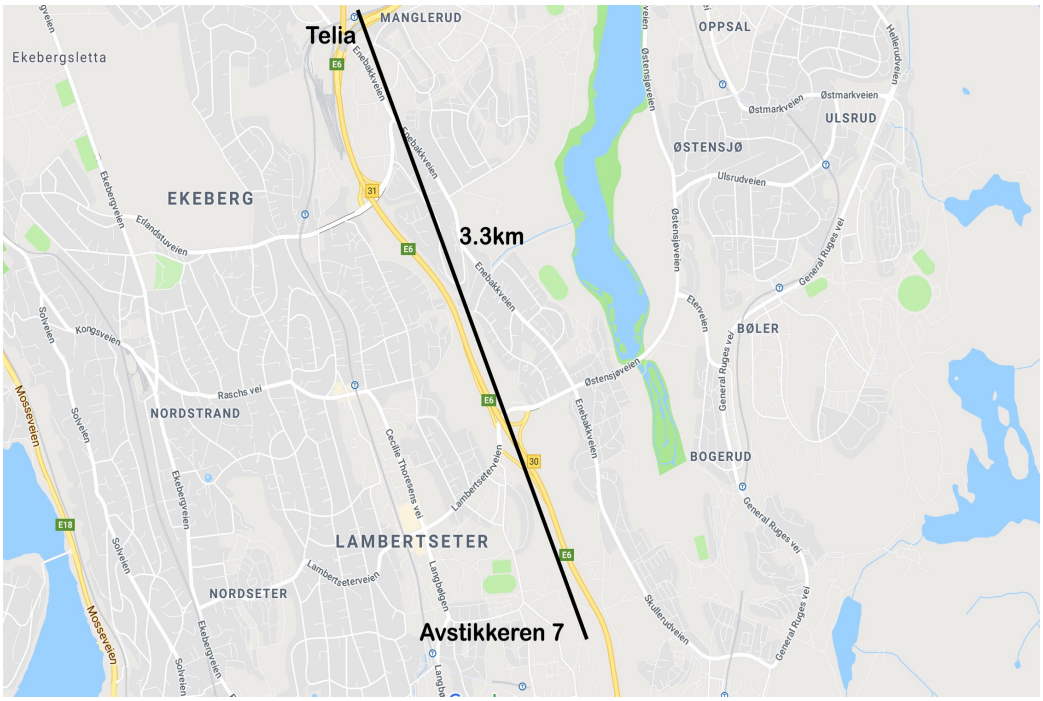

Figure 3.5: Device distance map over Lambertseter, Avstikkeren. The device did not manage to connect to Telenor's network at this location.

3.2 Devices

We have used a set of devices to achieve the results and in the following subsections we will give you an introduction to the devices used and why they were necessary for our tests.

3.2.1 Server

To host our web application we applied for a server at UH-IaaS, which hosts servers for university projects. We received a well suited server with the necessary specifications suited for our appication located in Bergen. The server was at times unstable due to power outages, but no results were damaged due to the downtime. In section, 4 on page 47, we will give a detailed description of the web application.

3.2.2 NB IoT development kit

Q-Free received early 2018 a development kit for NB IoT from the software company Ublox. This is a kit with a NB IoT chip called SARA N2 from Neul and software from Ublox, see picture 3.6 on the next page. The development kit enabled us to easily develop software for our tests which was crucial. The development kit was connected to a computer over USB and allowed us to send and receive data through the serial port of the chip. We used this setup for all our tests, which includes packet size, coverage, latency and general behavior of the network. We also used this setup 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 features and for our long-term tests we will send **AT+NUESTATS** to the application server so that we can process the information. In many cases we will also log test runs on the computer for detailed graphs of the behavior of the chip. The following subsection will give a detailed introduction to the chip we used.

Sara N2 Ublox chip

In section, 1.4.1 on page 8, we looked at Q-Free's parking sensor, which houses an identical NB IoT Ublox chip which will use for our tests. 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 19, there are configurable timers in the network, as

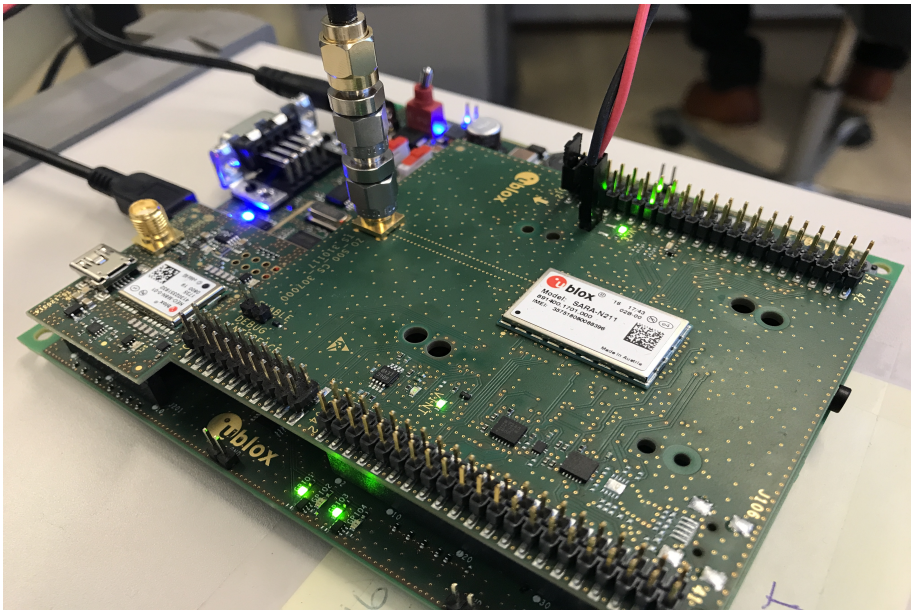

Figure 3.6: Ublox NB IoT development kit. The picture is taken with both attenuators attached, which is a total of $-30dBm$ signal loss.

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[1]. 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 [48].

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 most of this information. In table, 3.1 on the facing page, you can see the output of the command.

[1]We have only used the default timers in this paper

Property	Description
Power	NB-IoT signal power expressed in tenth of dBm
Total power	Total power within received 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

Table 3.1: **NUESTATS** command

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.

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. However, we soon realised that the internal clock wsa not very good and after the device had been operational for a day the internal clock would be skewed and the latency results were not accurate. We will elaborate in more detail about this problem in section, 6 on page 93, but our suspicion is that the clock is not updated very often and with an imprecise internal clock the time will be wrong. As we were perfoming all tests with a computer connected to the development kit we started using the clock of the machine connected to the kit and send this timestamp to the server. This did not only give us a more updated clock, but the timestamp also included milliseconds which gave the result higher precision.

eDRX settings - CEDRXS

In section, 2.3.3 on page 22, 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 for example issue this AT-command, **AT+CEDRXS: 1, 5, "0101"**, to put the UE into eDRX mode with cycle length of 163.84 seconds[2].

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)[3] with this command. If PSM is enabled, the device will enter deep sleep after expiry of the timer Active Time (T3324)[4].

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.

Send command with flags - NSOSTF

This AT-command is used to send UDP packets on a given port with a specified 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 on the next page, for details.

[2]This is the factory-programmed eDRX timer for SARA N2 chip
[3]The factory-programmed value of T3412 is 54 m.[48]
[4]The factory-programmed value of T3324 is 60 s.[48]

Mode	Description
0	No flags are set
1	Send message with high priority
2	Indicates RAI. 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.

Table 3.2: Transmit flag options

The data we transmit towards our server adds an additional COAP header followed by the actual data. We decided to do this to reduce logic on the server and create the environment similar to a real world application.

Signaling connection status - CSCON

We can verify the connection status of the UE with the command **AT+CSCON?**. If the reply 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 is inactive. Note that this does not mean that the device is in PSM mode. This only occours after the configured T3324 timer expires.

Network registration status - CEREG

We can verify the network registration status of the UE with the command AT+CEREG. This command is a very useful command for any application which wants to check if the UE is still connected to the network.

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 is very low [5].

3.2.3 Fluke precision multimeter

Q-Free owns a precision multimeter which we used to measure the current through the NB IoT chip. With the multimeter we managed to pinpoint

[5] $3\mu A$ in deep sleep mode

different stages of the chip. In addition we added corresponding statistics from **AT+NUESTATS** which increased the precision. As sketched in figure, 3.7 on the facing page, 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 lower precision 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, 3.3, 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 ","

Table 3.3: Fluke commands

3.3 Software

Writing software which should produce a set of results is hard, particularly when you don't know the expected results. We developed many test programs with a specific setup(see figure, 3.7 on the facing page, for an overview) as well as an web application to give us the knowledge and experience to give a precise report about the status of NB IoT the spring of 2018. A thing we learned during the testing phase is that the output format often changed and since our programs outputs data which is already processed it meant that we needed to reperform some tests. In hindsight it would be better to capture the raw data and develop a program which generated graphs based on the data. However, we are satisfied with the end result and you will see many of the generated graphs in section, 5 on page 59.

We wrote test features for the development kit using python with a connection between the program and the serial port. We configured the device to make sure that everything is according to our setup, see code 3.1.

Listing 3.1 Development kit initiation

```
1 uart_modem = serial.Serial(<serial port>, 9600,
    timeout = 0)
2 uart_modem.close()
3 uart_modem.open()
```

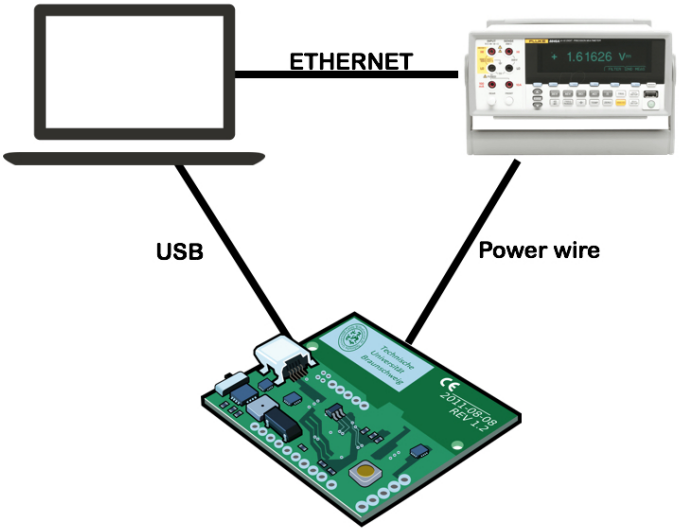

Figure 3.7: Lab setup [36] [19] [22]

```
4   uart_modem.flushInput()
5   uart_modem.flushOutput()
```

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, 3.2, 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 3.2 NB IoT sample transmit

```
1   AT+NSOSTF=0,"158.39.77.97",5683,0x0,88,"500230
2   30B131112AFF2D313039365F2D313034325F3233305F39
3   3732315F37363838305F333434333934353 25F315F3132
```

```
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 starts off by clearing all buffers with **:INIT** and then fetch for data on a regular interval with **:FETCH?; :INIT**. With each reading we converted the 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.

In the following sub section we have included some documentation on the most essential test programs. In the following chapter, 4 on page 47, we will describe the web application.

3.3.1 Test programs

at_command_test.py is a program well suited for testing different AT commands which is necessary to explore and research how the device handles different features. The program is like a REPL and requests an optional message and a command. If a message is supplied the program assumes that the request is a send command. If not the command is handled and the respons is printed in the terminal. Note that this program is written in python2 while the other programs expect python3 to execute. See AT command REPL[43] for complete code.

nbiot_labtest.py 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 takes in a list of parameters described in table, 3.4 on the next page.

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 RAI. Default: false
-l	Set device logging. Default: false
-rb	Set test to reboot NB IoT chip directly after execution. Default: false

Table 3.4: **nbiot_labtest.py** parameters. See NB IoT labtest[45] for complete code

To get more stable tests and to automate the process we added logic to perform a set of tests when starting the program. If byte size is not given to the program the execution will perform 4 tests on 50, 100, 200 and 512 bytes. Those 4 tests are a combination of logging and RAI.

For each iteration the program sends a message towards the server, followed by a period of logging. If logging is enabled the program fetches information from the chip with **NUESTATS** and three other commands - **AT+NPSMR?**, **AT+CSCON?** and **AT+CEREG?**. From **NUESTATS** we retrieve information about transmit and receive time, transmit power, ECL level and signal power. We choose these parameters because they give us good information by themselves, as well as they combined gives a good indication of what the chip is doing at any time. These results were logged approximately four times per second - limited to the speed of the serial bus between the computer and the development kit. Every fifth second we also fetch stored information from the precision multimeter. Depending on the configured precision of the multimeter we logged 50-400 samples per second which we in turn could correlate to the logged information from the chip. With all results the program produces a set of graphs which we will use for analysis of the network and the chip.

Because we were not able to collect the timestamp of each measurement of the multimeter we had to flatten the result and apply a timestamp to

each measurement. This means that we assume that the interval is equal. When looking at the figures in section, 5.1 on page 59, the power graphs are mostly linked to the results created from **NUESTATS**, but often one or two seconds subsequent of the other graphs.

Graphs generated:

- Receive: Shows the percentage of how much time the device were receiving compared to the actual interval.
- Transmit: Shows the percentage of how much time the device were transmitting compared to the actual interval. Both receive and transmit scales are incorrect, and should actually be multiplied with 100, so that 1 actually is 100%.
- Transmit power: Shows the transmit power of the device.
- Coverage: Shows the coverage of the device.
- ECL level: shows which coverage mode the device is currently in.
- PSM: Shows the status of PSM.
- RRC state: Shows the RRC state.
- Registration status: Shows the registration status of the chip. 1 is connected, 4 is unknown and 5 is searching for network.
- Power usage: Shows the current through the chip in mA.

Based on the average current over the time of the test we can also calculate total power usage. With the average current we can convert this to mWh and convert this to mW/s and multiply the result with the time of the test. This is an example from webapp [38]: $((9.2089mA * 3.3V)/60/60) * 60 = 0.5064mWh$. With this information we can show the difference in power usage with different scenarios and try to use the results to outline the lifetime of a device under different kinds of environments.

power_calculator.py calculates the power consumption over a given period from one of the test results from the short-term tests. We needed this tool to figure out how much power specific parts of the transmit used. It extracts all power measurements, in mA, between two timestamps and calculates the average current, multiplying this with the time delta between the two timestamps supplied. The program takes in a list of parameters described in table, 3.5 on the next page.

Parameter	Description
-f	File name. Must be from short-term test and have .html format
-s	Start time
-e	End time

Table 3.5: **power_calculator.py** parameters. See power calculator[46] for complete code

nbiot_labtest_details.py tests different packet sizes. The program transmits 6 packets with different packet sizes[6] a given number of times. After the given iterations are processed the program produces two graphs, one with all recordings and one with normalized data[7]. The program takes in a list of parameters described in table, 3.6.

Parameter	Description
-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.
-r	Set RAI. Default: false

Table 3.6: **nbiot_labtest_details.py** parameters. See NB IoT details[44] for complete code

[6] 25, 50, 100, 200, 400 and 512 bytes

[7] We have choosen to include average, sum, min and max values for each packet size

Chapter 4

The web application

The crucial thing to decide before doing research is to think about how the data will be used. The simple solution in our case would be to collect the data and import it into an excel document for analysis. However, there are approximately eighteen thousand elements in the database, thus maintaining and keeping track of this data 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 section, 2.5 on page 28, we mentioned that we gained most knowledge from the short-term test. However, we gained most of the practical knowledge from results coming into the web application and we would not perform the kind of tests which gave the most interesting results without the web application. We have included some results in the paper, but if you would like to explore the results we recommend looking at Q-Free, Telenor 09-10 March and Avstikkeren, Telia 13-15 March[1].

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

4.1 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[2]) and should use the appropriate request method, such as POST, GET, PUT and

[1]Here you can see that the latency is affected by the clock bug we will discussed in section, 3.2.2 on page 37

[2]Our server only implements create and read

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 many kinds of 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, 4.3.2 on page 55.

4.1.1 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 it is SQL driven which means all data has to fit into a pre-defined configuration. Since this project is small and the data could potentially changed during the test phase, we choose 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. MongoDB was a bit easier to use, so we decided to use mongoDB for our data storage. In subsection, 4.2.5 on page 51, we will explain how we used mongoDB for our implementation.

4.2 JavaScript packages for the server

We used different packages from NPM to include certain features on our web server. By including quality packages our server logic and size was kept low and we could focus on the data. In the following subsections we will give a short introduction to the most important packages used when developing the web server.

4.2.1 Express

Express is a fast and robust package for handling routes and offers nice APIs for listening on specific ports and so forth[3]. In the following code example, 4.1, you can see the base setup for making a 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 4.1 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);
});
```

4.2.2 COAP

Constrained Application Protocol (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. We have used node-coap package for including COAP support on the server[4]. Since the protocol is simple, we need to handle the request a bit different. The following code example, 4.2, shows a simple node-coap setup.

Listing 4.2 Base COAP setup

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

[3] See [17] for documentation
[4] See [28] for documentation

```
4 coap_server.listen(port, () => {
5     logger.log("info", `Worker ${process.pid}
        started coap server on ` + port);
6 })

8 // All request is handled by this function. It is
    not possible to request a specific url path
9 coap_server.on('request', function(req, res) {
10     // Payload of the request is a byte stream.
        Parse the stream and handle the data
11     var data = JSON.parse(req.payload.toString());
12 })
```

4.2.3 Cluster

With express 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[5]. It is based on web workers which are an abstraction for processes in Node.js. The following code example, 4.3, 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 4.3 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;
```

[5]See [24] for documentation

```
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 }
```

4.2.4 Webworker-threads

Webworker-threads is an additional safety feature package to prevent degraded latency results[6]. 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.

4.2.5 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[7]. In contrast to SQL, NoSQL databases uses different key words to describe data. MongoDB can consists of several

[6] See [10] for documentation
[7] See [30] for documentation

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, 4.4, shows how to set connect to the db and insert a simple document in a collection with an API call.

Listing 4.4 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 }
```

4.2.6 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 the package moment is a great tool to keep track of time[8]. The following code, 4.5 on the facing page, takes in a timestamp

[8] See [29] for documentation

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 4.5 Simple moment example

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

4.3 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 relevant to the thesis. However, we will give a brief introduction on how to use the app and what kind of analysis you can view with it.

4.3.1 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, 5.1 on page 59. 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 4.1 on the following page. Further more you can select nodes on the left which includes data from different sites and network providers, see screenshot 4.2 on the next page and 4.3 on page 55.

By default the graphs will display data one day backwards from the latest data entry of the selected node. This means that 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. You can also inspect certain areas by selecting an area of one of the graphs and the other graphs will adjust.

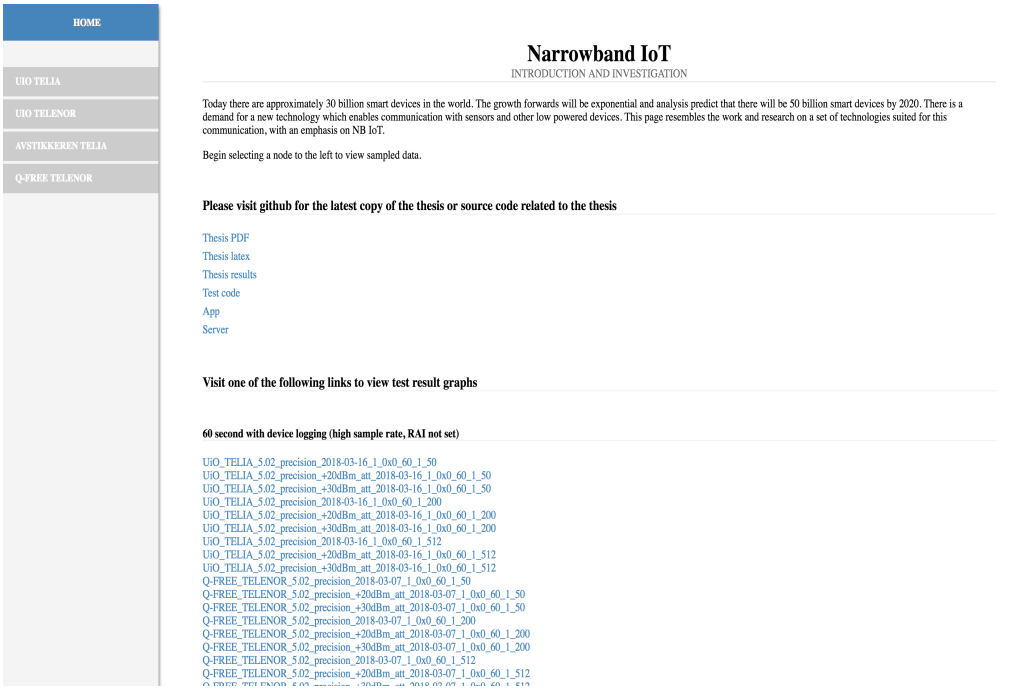

Figure 4.1: SensorApp homepage. Additional material are linked at this page, and you can select a node on the left to explore the long-term results.

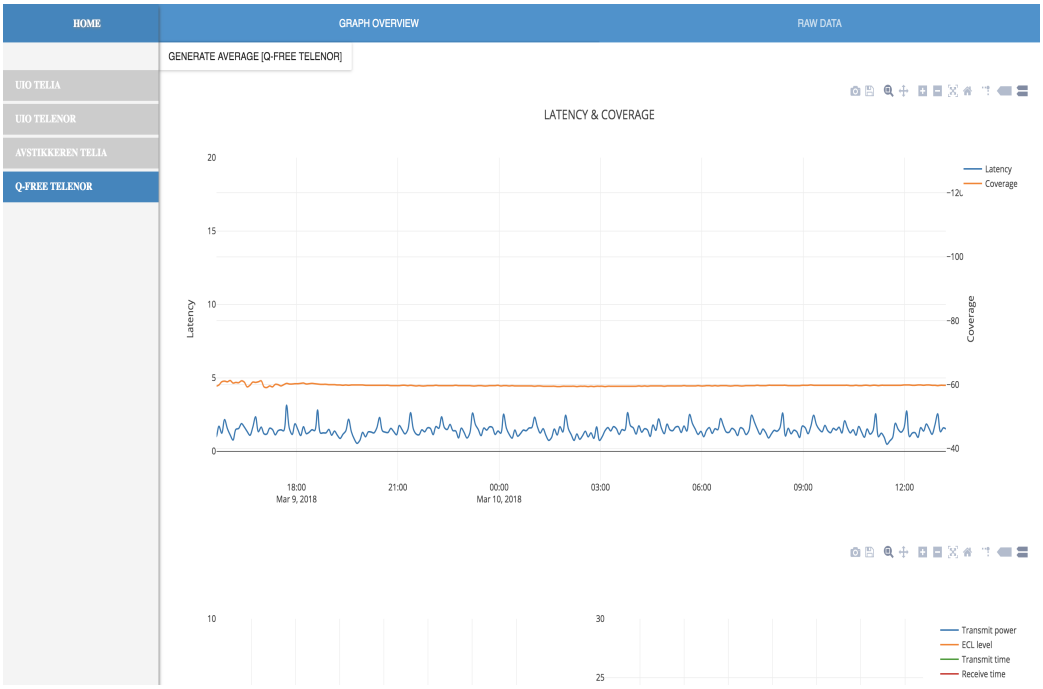

Figure 4.2: SensorApp nodepage part 1. The figure displays the coverage and latency graph from Q-Free, Telenor 09-10 March, with very good results.

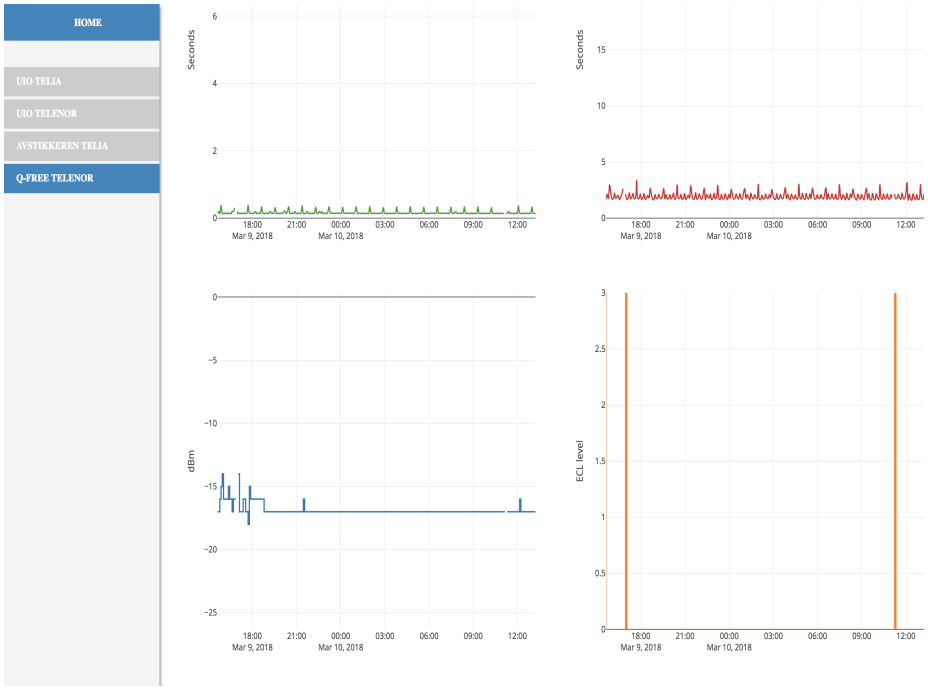

Figure 4.3: SensorApp nodepage part 2. The figure displays the general statistics from Q-Free, Telenor 09-10 March, with very good results.

4.3.2 Analysis

When having selected a node 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. In the following subsections we will introduce you to a set of features we have included in our analysis. Remember that the data is mostly produced by **AT+NUESTATS**, so read, 6 on page 93, 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[47] 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**. To give a

clearer understanding of coverage we have classified the coverage level into categories, see table 4.1.

Coverage level(dBm)	Category
-40 to -60	Excellent reception
-60 to -80	Good reception
-80 to -100	Quite good reception
-100 to -110	Bad reception
-110 to -130	Very bad reception

Table 4.1: Coverage categories

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 value which represents how much time the device was in receive or transmit mode 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 mode we know that the expected lifetime is degraded. Most of the time the active percentage will be low due to usage of Release Assistance Indicator (RAI) 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 level is also closely related to the power usage of the device and an important factor we wanted to monitor. You can clearly see that the power usage increases when the device enters ECL 1 and 2 in the testing section, 5 on page 59. The output of ECL is taken 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[31]. 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, 5.1 on page 59, we will log the status of the device up to ten times per second which will give a better understanding of what is happening with the transmit power level. In addition we will monitor the actual power usage with a multimeter, giving us more accurate test results.

Chapter 5

Testing

In this part we will discuss the tests we performed. We will describe why and how we did each test and give you a detailed explanation of the results. In addition you will be presented an overview of the testing environments, devices and parameters that affected the tests. We have tried to include as many tests as possible and throughout this part we will analyze the results and at the end we will give a summary of the current NB IoT situation.

The test phase began early January and was completed late March. The reason for the late test phase was due to a number of things. First of all, the hardware and software related to NB IoT were 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 we used were not production ready, but at that stage the error rate was very low. Because of the early adaption we encountered some interesting results, giving us an indication of the state of the technology. Given the hypothesis we needed to test the chip and the networks with good equipment and at several locations. In chapter, 3 on page 31, we gave you an overview of the testing premises and the devices used.

In section, 7.1 on page 95, we will introduce a set of best practice guidelines to NB IoT and we will include the results from the testing phase too so that you will see why the guidelines are as described.

5.1 Short-term tests

In this section we will show our results based on short-term tests. We will look into how the device handles normal usage and edge cases where non-default behavior is activated. We define normal operation when ECL level

is 0 and coverage is 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 RAI set and PSM activated.

We define short-term tests as a logging sequence over a short period with frequent data points. The duration of the tests are under two 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 of 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 subset of the results in each section with a reference to a more detailed version in each caption. In this way you will be presented with one problem at a time which will give you more in-depth knowledge.

5.1.1 General

Most of our short-term tests were performed with an emphasis on power usage, rather than coverage and latency. It is however clear that high power usage is a side effect of bad coverage. All our tests show that better reception gives lower latency and power usage. In one of our test cases with Telia at Q-Free we saw that the device was struggling even with good reception. Looking at figure, 5.1 on the next page, we see that the device is not entering PSM mode even though the expired 120 seconds has elapsed. We saw that the device does not leave RRC connected mode, hence the power consumption stays at $6mA$. The reception of the device at this location was good and the SNR values were much like Telenor's at the same location. In this test we did not use RAI, so the expected behavior is transmit at the beginning of the program, following a period of 20 seconds in RRC connected mode. After this the device should enter eDRX mode, only receiving data on a set interval, normally 2-25 seconds, with a number of receive transactions within each eDRX period. After this period the device should enter PSM mode if it is enabled by software. However, in this test it looks like the device is performing DRX and polling for data from the network very frequent. Looking at the network connection graph you see that the device does not leave this state until it suddenly

indicates loss of connection and rapidly reestablishing connection and transmits the last packet. Directly before the last transmit we see a short period of frequent receive transactions. This might be related to the loss of connection, presuming that the ECL level is delayed until connection is restored. An alternative is that due to the bad configuration the time of the paging window negotiation is longer than expected, resulting in a period of constantly trying to receive data from the network. If we continue looking at the time of the last transmit we see something interesting with the transmit power. After renegotiating the connection towards the network the transmit power is adjusted from around $+6dBm$ to $+23dBm$ even though the signal power is kept at the same level. At this location most of the tests showed some kind of flaw in the network, either related to packets being dropped, long transmit times or trouble following the normal transmit procedure. It would be interesting to investigate what happened of the network side of this setup. The network becomes a black box and when the network performs out of the ordinary it is hard to debug.

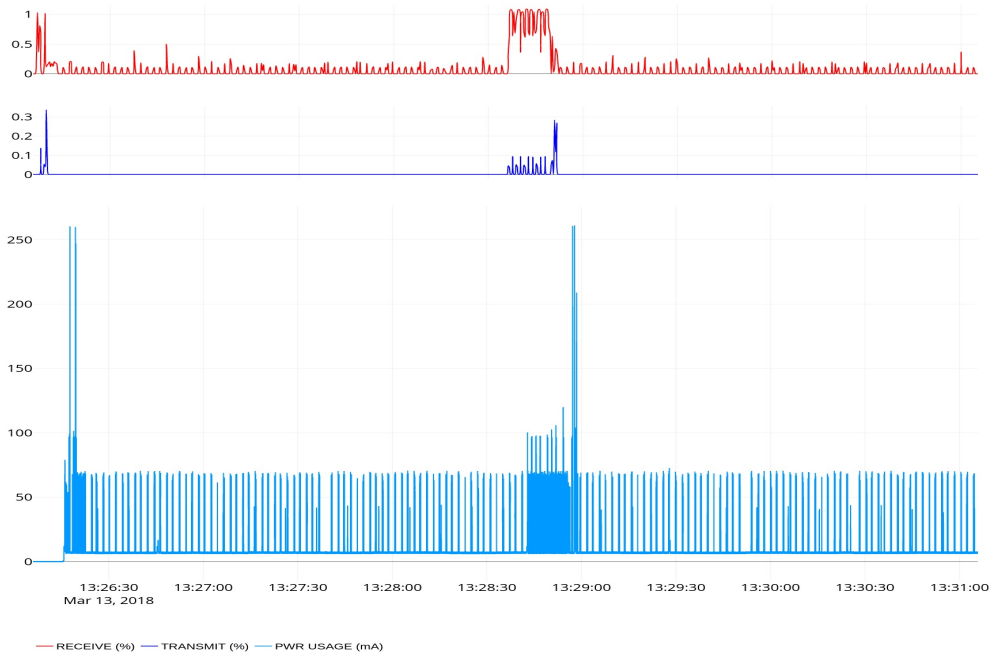

Figure 5.1: The figure displays one test from Telia which behaved out of the normal. See figure, 5.15 on page 77, or visit, webapp [7], for more details.

With this in mind we will show a similar presentation of a proper transmit procedure, from Telia and Telenor. Looking at figure, 5.2 on page 63 and 5.3 on page 63, the process is totally different and is looking more like a graph representation of the specifications. Looking first at the graph

from UiO with Telia we see two transmit periods of around 20 seconds, following a period of eDRX and then PSM. We were not using RAI in this test either, hence the RRC period of 20 seconds. Notice that the device is not pulling for data all the time within the RRC period. It usually receives a lot of data when initiating the uplink transmit, but after the transmit it only pulls for data around 20% of the time, hence reducing the power usage. When asking Telia about this they state that this is most likely related to DRX process which kicks in because there are not any data to receive[15]. However, we did not manage to figure out why the device performed this way when connected to Telia and not Telenor. It is also worth noting that when the device is not in RRC connected mode the statistics from **NUESTATS** become stale. This is an indication of deep sleep mode, and in eDRX the device should sleep between each eDRX period. We can see that after two eDRX periods the device enters PSM with permission from the network.

Moving focus over to Telenor we can also see some distinct differences from Telia's transmit process. The first thing we notice is that Telenor is pulling for data close to 100% of the time spent in RRC connected mode. This results in an increase of 5 in terms of pure power usage compared to Telia's solution. We can see that the current at this stage is comparable, at around $60 - 70mA$, a little higher than Ublox specification on page 16[47]. If you look at the graphs from the website you are able to zoom in on these periods, revealing that Telia's RRC period is mostly using $6 - 7mA$, only flickering up to $60 - 70dBm$ when actually receiving. Telenor's RRC period however is mostly at $50 - 60mA$ and it is likelty that there are different configurations in the networks resulting in actual receive time. A part from this, the graphs have the same characteristics. One thing worth noticing is the adjusted transmit power because of the excellent reception at Q-Free. The device has $-55dBm$ signal power resulting in $-24dBm$ transmit power, almost minimum. This is clearly reflected in the power usage graph where Telenor only uses at most $70mA$ when transmitting, while Telia at $+9dBm$ transmit power is reaching $110mA$ at transmit time. Notice that this is close to what Ublox self is stating in their specifications. It is however lower than our calculations and we will discuss this in section about deviations, 6 on page 93.

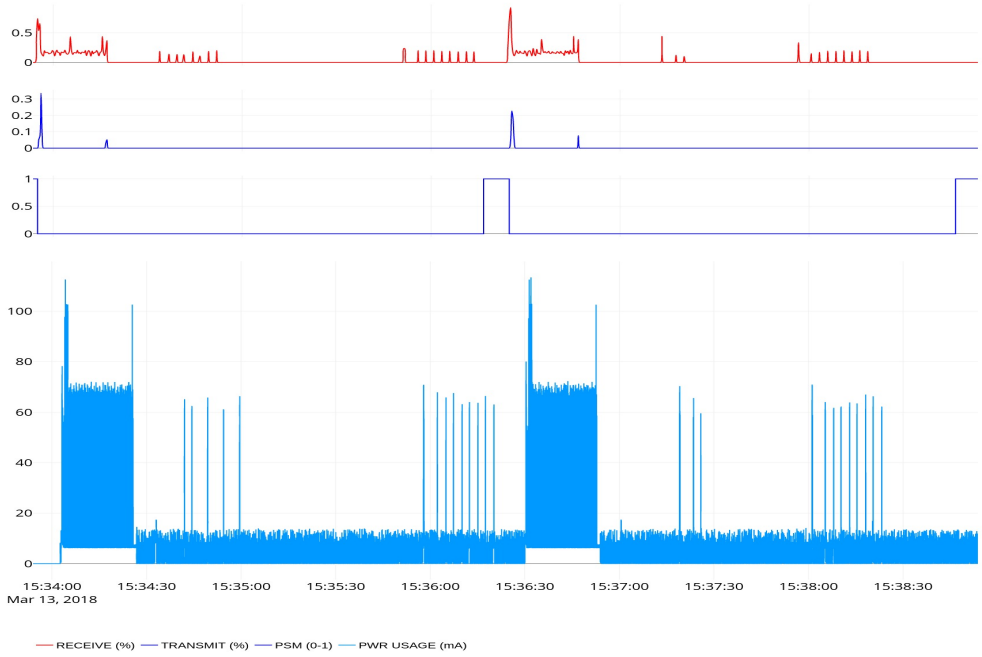

Figure 5.2: The figure displays normal behavior with two transmits over 5 minutes at UiO with Telia. See figure, 5.16 on page 78, or visit, webapp [8], for more details.

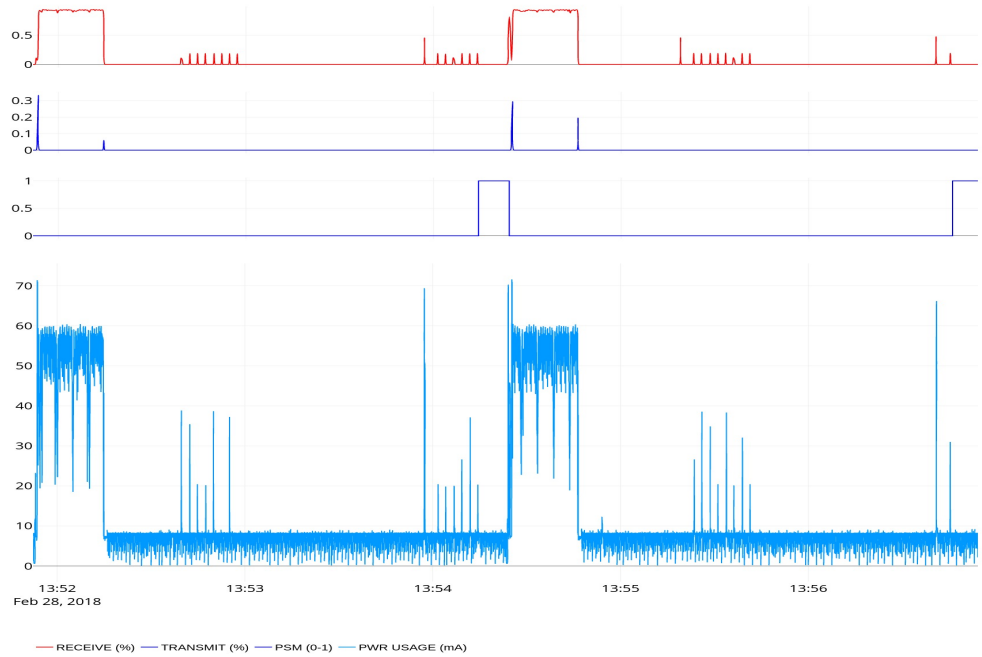

Figure 5.3: The figure displays normal behavior with two transmits over 5 minutes at Q-Free with Telenor. See figure, 5.17 on page 79, or visit, webapp [6], for more details.

5.1.2 Transmit power spike

Disregarding coverage level, one thing we noticed while doing most of the tests was that there was often a spike of power usage up to $230mA$. We have not included all graphs in this paper, but visiting henninghaakonsen.me you can view results tagged with 60x1 for Telenor for example. You will see this spike in many of the graphs and it is an interesting fund we would probably not find without the precision multimeter. The context of the occurrences is when the device is in RRC connected mode, hence in active communication with the network - either UL or DL. As explained in paragraph, 4.3.2 on page 56, a NACK is sent with $+23dBm$ transmit power, hence we have reason to believe that this is what is happening at these spikes. This behavior is not mentioned in the specification, and at the moment of the tests there were instances of this behavior than without. We also saw this behavior with Telia's network, so it is likely that this is normal procedure in a transmit process. It is however somewhat strange that this happens with transmits when the signal power is excellent, since we would presume that this practice would only be in case of bad reception and signaling faults.

5.1.3 Loss of connection prior to transmit

We performed some tests on Telenor's network at Q-Free with interval at 40 seconds with and without RAI. At many of these transmits we saw that the device indicated network disconnection directly before transmitting the packet. This resulted in a period with ECL at 255 and higher power usage for a short period where the device reconnected to the network. We have included two figures, 5.4 on the facing page and 5.5 on page 66, representing this behavior. In both these tests, there is a period of reconnection to the network, followed by the actual transmit process. This pre-procedure of reconnecting to the network operates at a current of $45 - 55mA$ and is comparable to a RRC period or transmitting with very good reception. In these tests the reconnection time has been rather low, but you will see in the long-term tests, 5.3 on page 83, that this is not always the case. This behavior is regardless of the RAI flag, hence the power usage is a lot higher compared to the actual transmit process when using the RAI flag. This is a root problem users and developers would have a hard time finding, and could cause battery lifetime issues. Using **power_calculator.py** we have calculated the power usage over the pre transmit period, see result in equation 5.1 on the next page. This is actually $0.1764/0.034 = 5.2$ times higher than a normal transmit(see webapp [37]) with RAI set and excellent signal power. In terms of battery lifetime this behavior shortens the battery lifetime with approximately 5 hours given

that the application transmits every hour.

$$\begin{aligned} From &: 2018-03-07 \quad 13:36:14.415420 \\ To &: 2018-03-07 \quad 13:36:19.022218 \\ ((41.793363329583805mA * 3.3V/60/60) * 4.606798S) &= 0.176489mWh \end{aligned} \tag{5.1}$$

Figure 5.4: The figure displays one transmit over 40 seconds at Q-FREE with Telenor, with device logging. See figure, 5.19 on page 81, or visit, webapp [1], for more details.

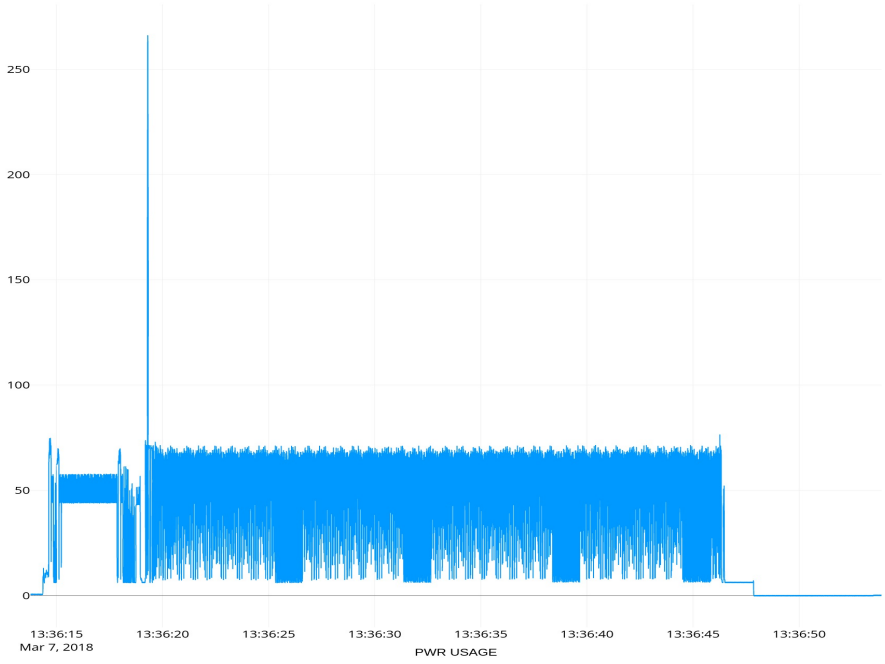

Figure 5.5: The figure displays one transmit over 40 seconds at Q-FREE with Telenor, without device logging. Visit, webapp [2], for more details.

Because of this bug we had some problems related to the long-term tests so we had to investigate what actually happens when doing a longer test. The device would indicate network disconnection and reconnection, resulting in ECL being 255 and receive/transmit counters were reset. We looked into the issue by monitoring the transmit process towards the server and most of the time the device did not try to reconnect to the network. The behavior is not expected and is most likely related to the hardware and software of the development kit. We conclude that the bug does introduce reconnection some times, but we believe it is hardware and software related and not network related. This is based on what we talked about in section, 2.4 on page 25, that the chips produced at the point of the tests were not production ready, but good enough for us to test the networks.

5.1.4 Coverage and ECL level

Depending on the devices signal power the ECL level should adjust accordingly. As discussed in section, ECL - 2.3.6 on page 24, the network provider configures thresholds for ECL levels and our results have mostly followed these thresholds as expected. We have included two figures to describe the behavior of different ECL, 5.6 on page 68 and 5.7 on page 68. The two figures are showing results from one transmit of 50 bytes over

a period of 60 seconds with RAI set. You should focus on the two tops at the beginning of the figures, please visit the webapp(link is supplied in the figures) to zoom into the specific periods. If we begin looking at the first figure we can see that the signal power is good, but at transmit the power is bumped up from $+9dBm$ to $+23dBm$, maybe because of a NACK. Looking at the transmit graph we can see that the transmit process is relatively short, only covering approximately 1.5 seconds. In addition we see that the device was actually only active for 15-20% of that time, resulting in actual transmit time of $\tilde{0}.3$ seconds. We have used the program **power_calculator.py** to calculate the power usage for the transmits and the following equation, 5.2, shows us the power usage of the transmit with ECL 0.

$$\begin{aligned} From &: 2018-03-1612:35:55.901531 \\ To &: 2018-03-1612:35:57.543329 \\ ((24.7694923857868mA * 3.3V/60/60) * 1.641798S) &= 0.037278mWh \end{aligned} \tag{5.2}$$

This next figure is showing a transmit where the signal power is poor, hence the ECL is 2. Not surprisingly the transmit power is at $+23dBm$ and we can see a clear difference in this result compared to the previous. First of all the transmit process endures longer, around 5 seconds, hence the device is retransmitting the packet to ensure that it will be received at the eNB. The following equation, 5.3, shows us the power usage of the transmit with ECL level 2.

$$\begin{aligned} From &: 2018-03-1613:13:42.283327 \\ To &: 2018-03-1613:13:47.912613 \\ ((80.83920204978038mA * 3.3V/60/60) * 5.629286S) &= 0.417145mWh \end{aligned} \tag{5.3}$$

If we compare the two transmit operations we see that the second transmit uses $\tilde{1}1.2$ times as much power as the first. In our experience the first transmit is the most accurate towards what actually is going on most of the time, but if the UE has bad reception the battery lifetime will be heavily affected. In section, 2.3.7 on page 24, we described the theoretical disadvantage of different ECL levels, however we think that this comparison is more accurate towards normal behavior. We have only included results from Telia for this observation since the results were very similar. We did not see the expected increase with very good reception at Q-Free with Telenor either, hence in practice the difference in power usage with "normal" reception is not millions of times higher.

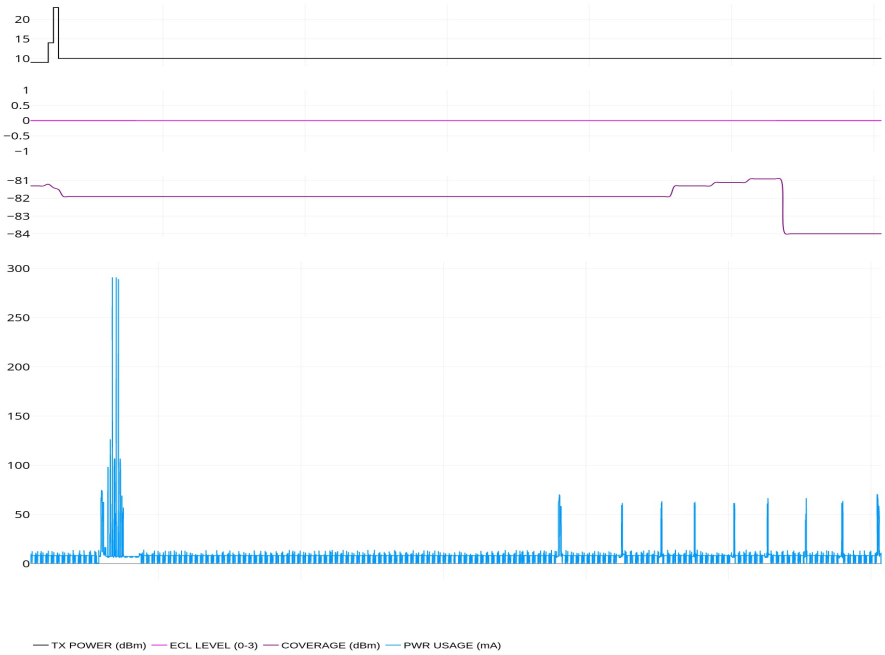

Figure 5.6: The figure displays one transmit over 60 seconds at UiO with Telia, with ECL 0. See figure, 5.20 on page 82, or visit, webapp [3], for more details.

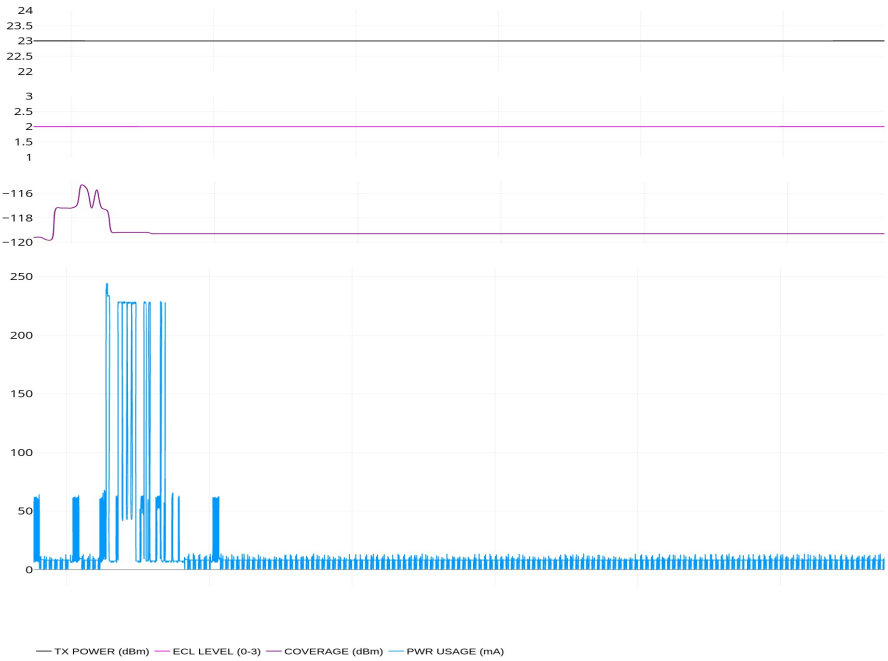

Figure 5.7: The figure displays one transmit over 60 seconds at UiO with Telia, with ECL 2. See figure, 5.21 on page 83, or visit, webapp [4], for more details.

5.1.5 Transmit comparison

Many of our tests were performed at Q-Free's office towards Telia and Telenor with different packets sizes and at different coverage levels. This was a good opportunity to collect the results and compare the results in a chart. All transmits used the same setup and the signal power was altered using attenuators. One thing to note is that since the tests were performed at Q-Free where the signal power of Telenor's network is exceptional, the device never entered ECL level 1 or 2 with Telenor's network. There was however an interesting result and we will begin looking at the results from the transmits without RAI set, 5.8 on the next page.

There are a few things to notice in this chart. First of all there are two groupings, one with Telenor's transmits and another with Telia's. As we have seen before Telia's transmits use less time in RRC connected mode, hence lowering the power usage. We see that Telia uses around 50% less power at the first two transmits, regardless of byte size. It is not until the last transmit where Telia's transmits uses more power, and this is because the device has entered ECL 2 and retransmits occurs. We will look closer into packet size in the next section, 5.1.6 on the following page, but we can see indications that packet sizes does influence the power usage to some degree.

Moving over to the chart with transmits with RAI set we see that the results are more similar between the two network providers when the device is in ECL 0. If we look closely at the first transmits from Telia we can see a bigger gap between packet sizes, and this is mainly because the RRC period is not influencing the results and we get more information about the actual transmit process. You can view the actual transmit graphs at henninghaakonsen.me where they are tagged with **60_1**.

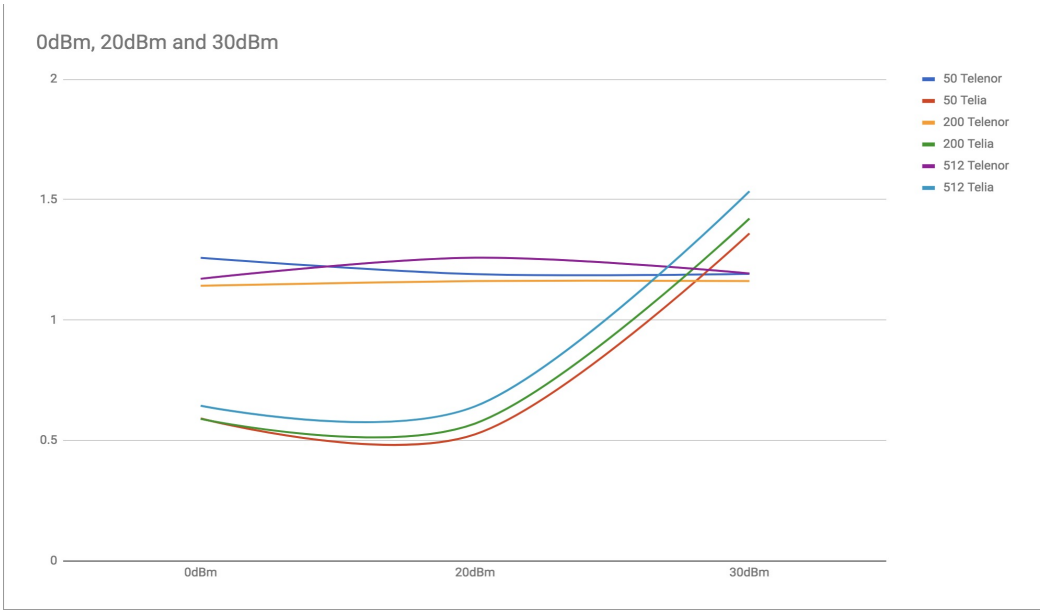

Figure 5.8: The figure compares transmits without RAI

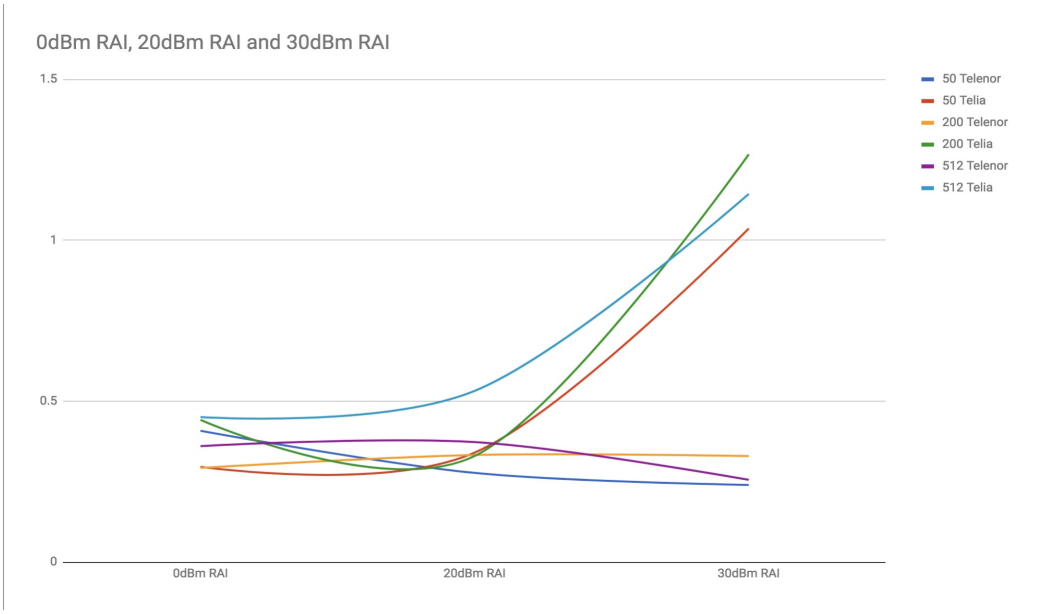

Figure 5.9: The figure compares transmits with RAI

5.1.6 Packet size

The typical packet size for an IoT device is often small, typically under 100 bytes, but for some cases bigger packets might be required and we have performed a specific test to see what the packet size does with a transmit. The test is performed with the program **nbiot_labtest_details.py** and we

have included two figures in section, 5.10 on the next page and 5.11 on the following page, which displays two executions of the program - one with RAI and one without. We needed to test both transmit modes to see how much the transmit process with different packet sizes costs in terms of power usage.

We begin looking at the first figure - transmits without RAI. The whole transmit process without RAI endures approximately 20-30 seconds, while the actual transmit process only takes 0.5-2 seconds depending on the coverage level. Looking at all aggregated lines it is easy to see that the bigger part of the power consumption comes from overhead of the RRC process. There is a slight increase in power usage while increasing the packet size, but it is clear that if your application transmits packets without RAI flag the packet size is not crucial to the power usage.

Moving over to the next graph we see a noticeable difference. Not surprisingly there is a general decrease in power usage when using RAI. If we look at the average aggregated value of packets transmitted with 25 bytes, there is a decrease of 240%. This results in less overhead from other processes related to the transmit, hence the power usage from the transmit has bigger influence on the graphs. Looking at the average line in pink there is a steady increase in power usage related to the packet size. Comparing 25 and 512 bytes, there is a power increase of 217% which sounds logical since a bigger payload required several data packets. It is interesting to look at the aggregated sum of each byte size category. According to, [5], the Maximum Packet Size (MPS) is 680 bits, which is 85 bytes. Looking at the aggregated sum line we see two spikes which is likely to be related to the MPS. We can see the start of a spike at 100 bytes, which is directly after the MPS. There is another spike from 400 to 512 bytes, which also may be related to the maximum transmit size of the chip which is 512 bytes.

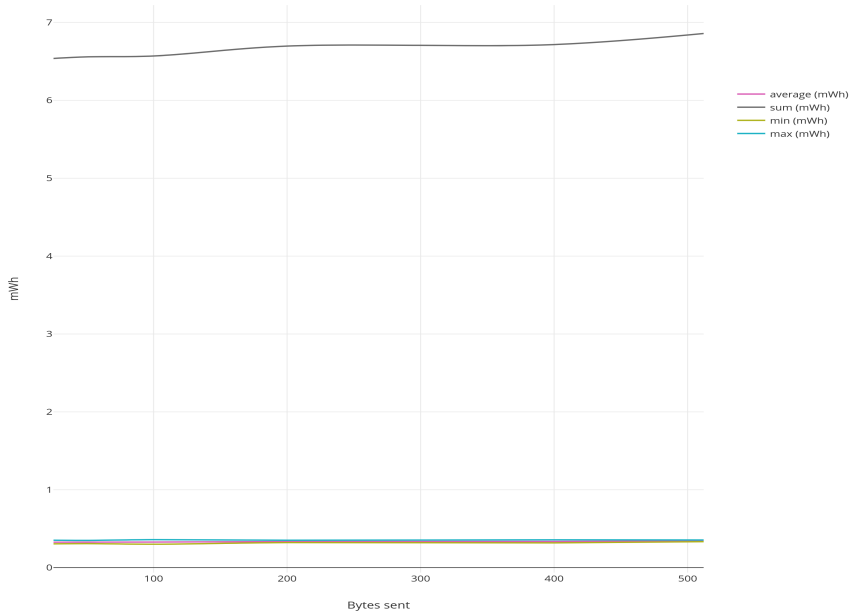

Figure 5.10: The figure compares different packet sizes without RAI set. Visit, webapp [14], for more details.

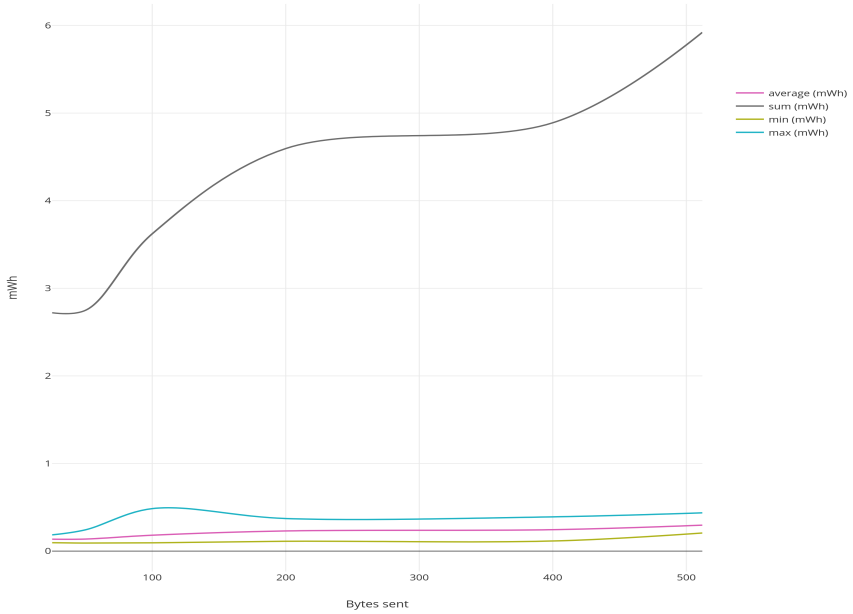

Figure 5.11: The figure compares different packet sizes with RAI set. Visit, webapp [13], for more details.

5.1.7 Sensor reboot

There are a number of reasons why the device would perform a reboot, especially if the application is operating over several years. One reason for a reboot is because of some software or hardware fault, which could occur even though it should not happen. Another reason is that the developer reboots the device if it has stalled or some logic kicks in. The main reason to investigate what happens in a reboot is to see how long to network connection takes. If this process is long and cumbersome it will affect the power usage, hence the battery lifetime decreases. We will elaborate why and when your application should consider rebooting the device in section, 7.1 on page 95.

We performed several reboot tests - with both network providers and with different coverage levels. We have included two results, one from Telenor and one from Telia. The results supplied are very similar to the other tests we did with less good reception. The only difference was a slight increase in power consumption. Since the device reboots we are not able to log the status of the device with **NUESTATS** so you will only be able to see the power usage in this test.

We will begin looking at Telenor's result without any attenuators, see figure 5.12 on the following page. The graph displays 120 seconds which includes the whole process of rebooting and connecting to the network. The first thing we see is a long period of what looks like RRC connected mode. Following is what looks like the actual connection process about half way into the graph and ending with a RRC period until the timer runs out and the device enters PSM mode. The whole process uses $5.1mWh$, which if we use the results from the test about packet sizes is the same as around 28 transmits with 100 bytes payload with RAI set. If we assume that we transmit a packet every hour this reboot process uses approximately the same amount of power as a days worth of uptime.

Moving over to Telia's result without any attenuators, see figure 5.13 on page 75, you might notice that this result only covers 60 seconds. This is because the reboot process using Telia's network was much shorter and since we only want to look at the reboot process the log duration is kept to the minimum. Directly after reboot the device performs a set of transmits, followed by a period with RRC connected mode. After this process the device enters PSM and the device has connected to the network. The whole process uses $0.55mWh$, which is around ten times less than with Telenor's network. It is interesting to see the differences the network provider introduces to the network connection process, since this is mainly due to network configurations. One explanation of the lower power consumption is likely related to the power consumption during the RRC period. We saw in section, 5.1.1 on page 60, that Telia used around 60%

less time in RRC connected mode, hence reducing the power consumption drastically. In addition we see that the duration of the connection process is twice as long with Telenor. The results are consistent throughout all reboot tests and gives an indication that Telia has used the configuration to reduce the power consumption on reboot and in these RRC periods.

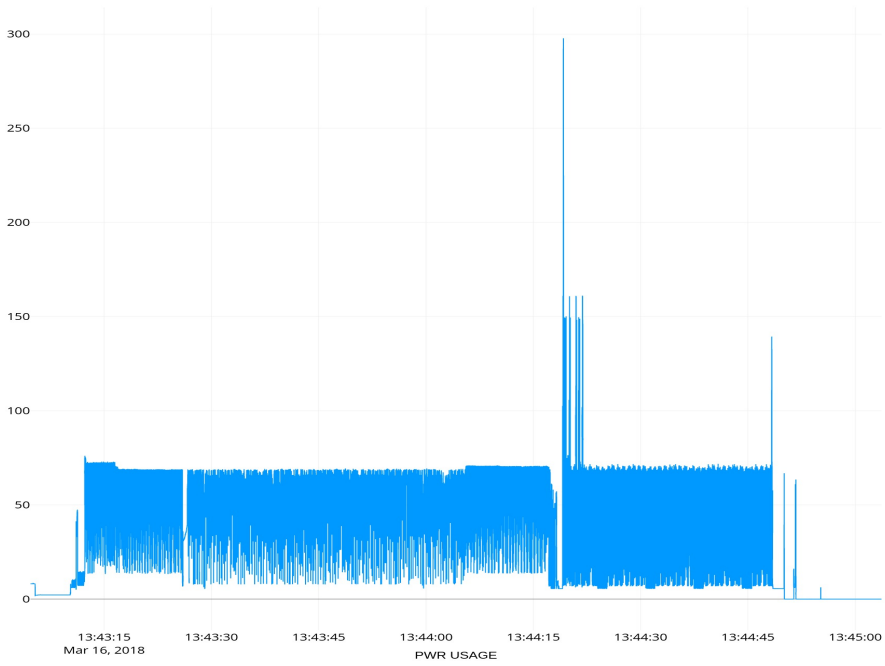

Figure 5.12: The figure shows a reboot of the development kit with Telenor. Visit, webapp [40], for more details.

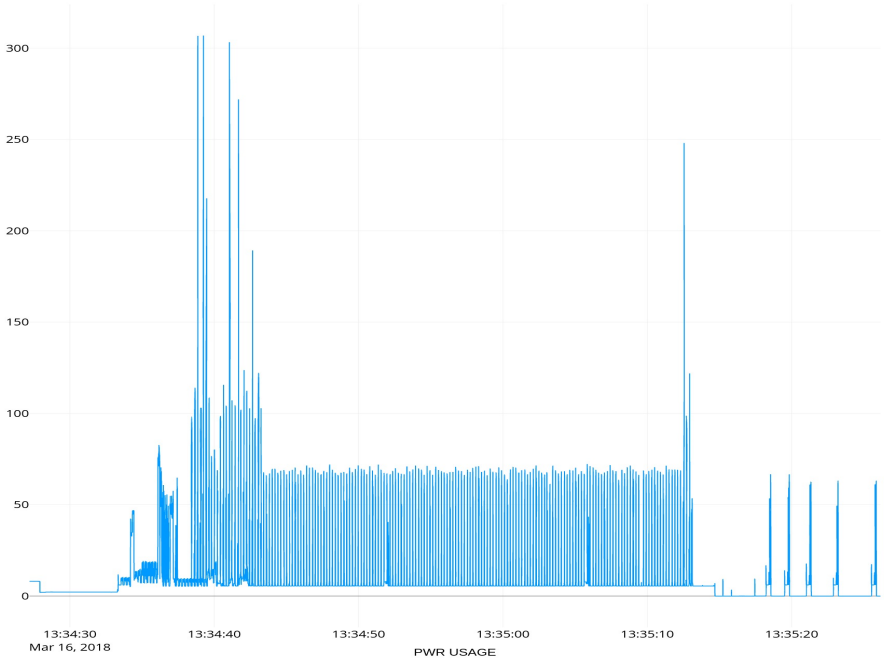

Figure 5.13: The figure shows a reboot of the development kit with Telia. Visit, webapp [42], for more details.

5.1.8 Downtime test

If the network is down because of a power outage or similar breakdowns the UE might be disconnected from the network if there are no other eNB's in the close proximity. We had a period where the device actually lost connection to the network, even though the signal strength showed otherwise. In figure, 5.14 on the next page, you can see a longer monitoring sequence. At first the device transmits data, but after a short while it is disconnected and the subsequent results gives us an indication of what happens if the device looses connection. We see that the receive/transmit timers does not change, hence we believe that the device does not use any effort trying to reconnect to the network. We will discuss different approaches to this problem in section, 7.1 on page 95.

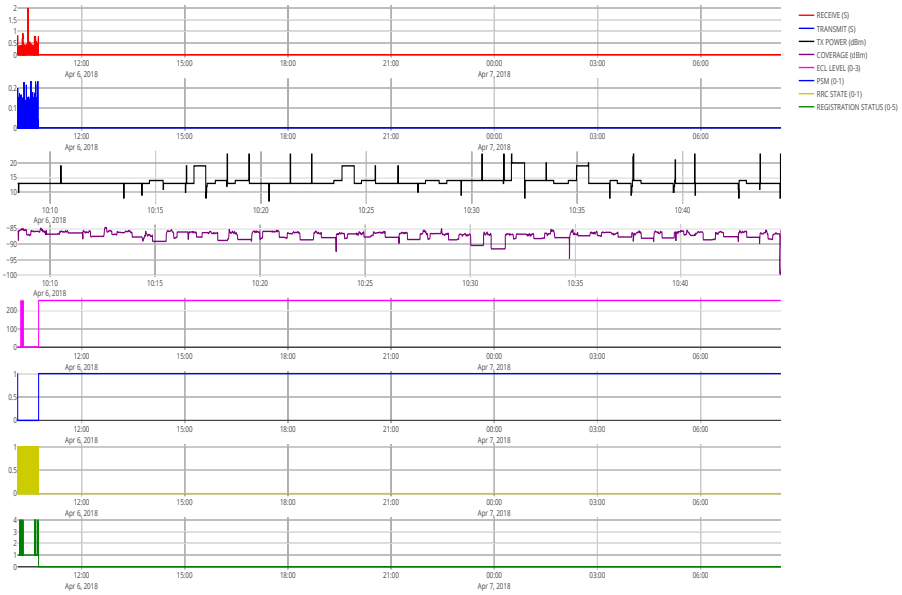

Figure 5.14: Example of disconnection from the network

5.2 Short-term figures

5.2.1 General

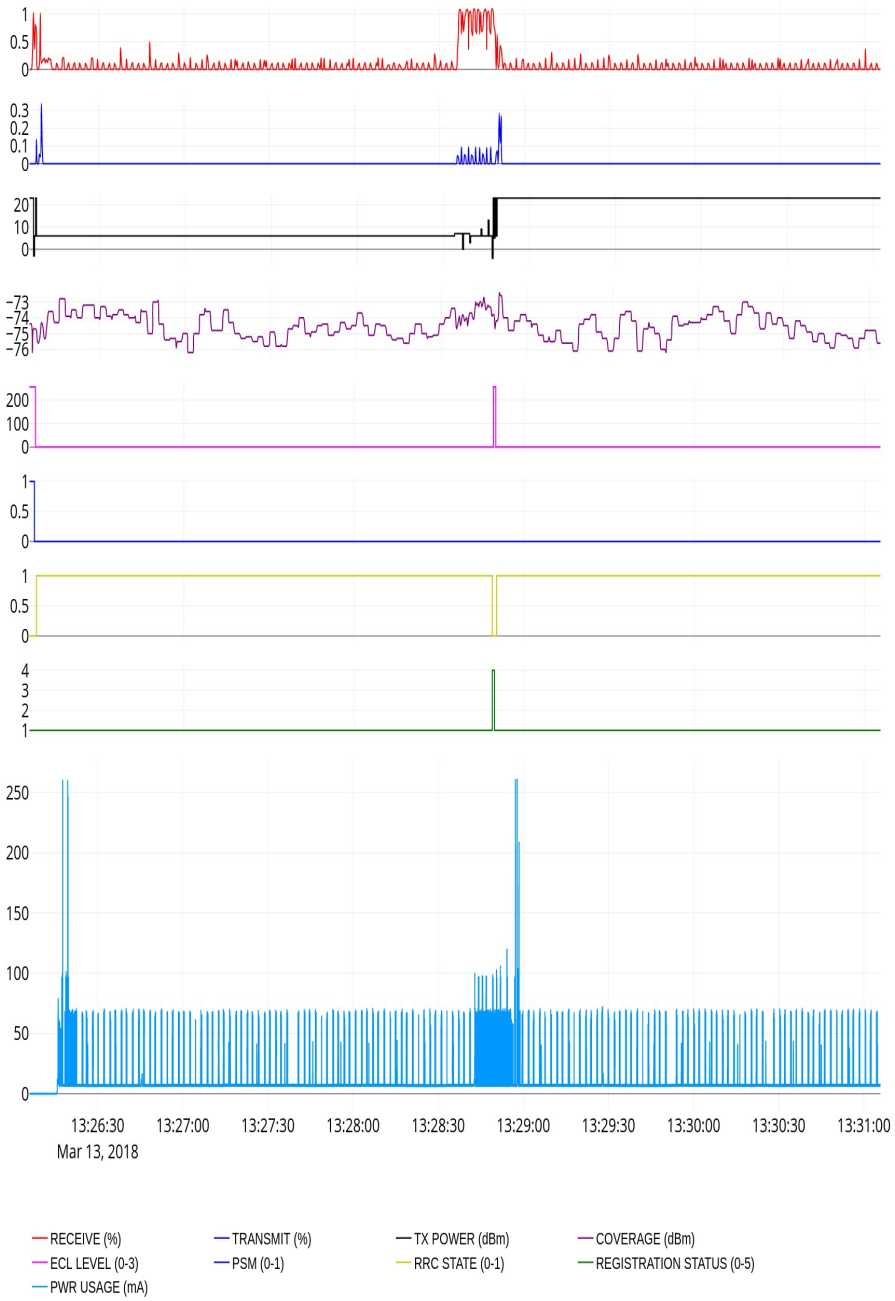

Figure 5.15: Short-term figure - unusual behavior, Telia

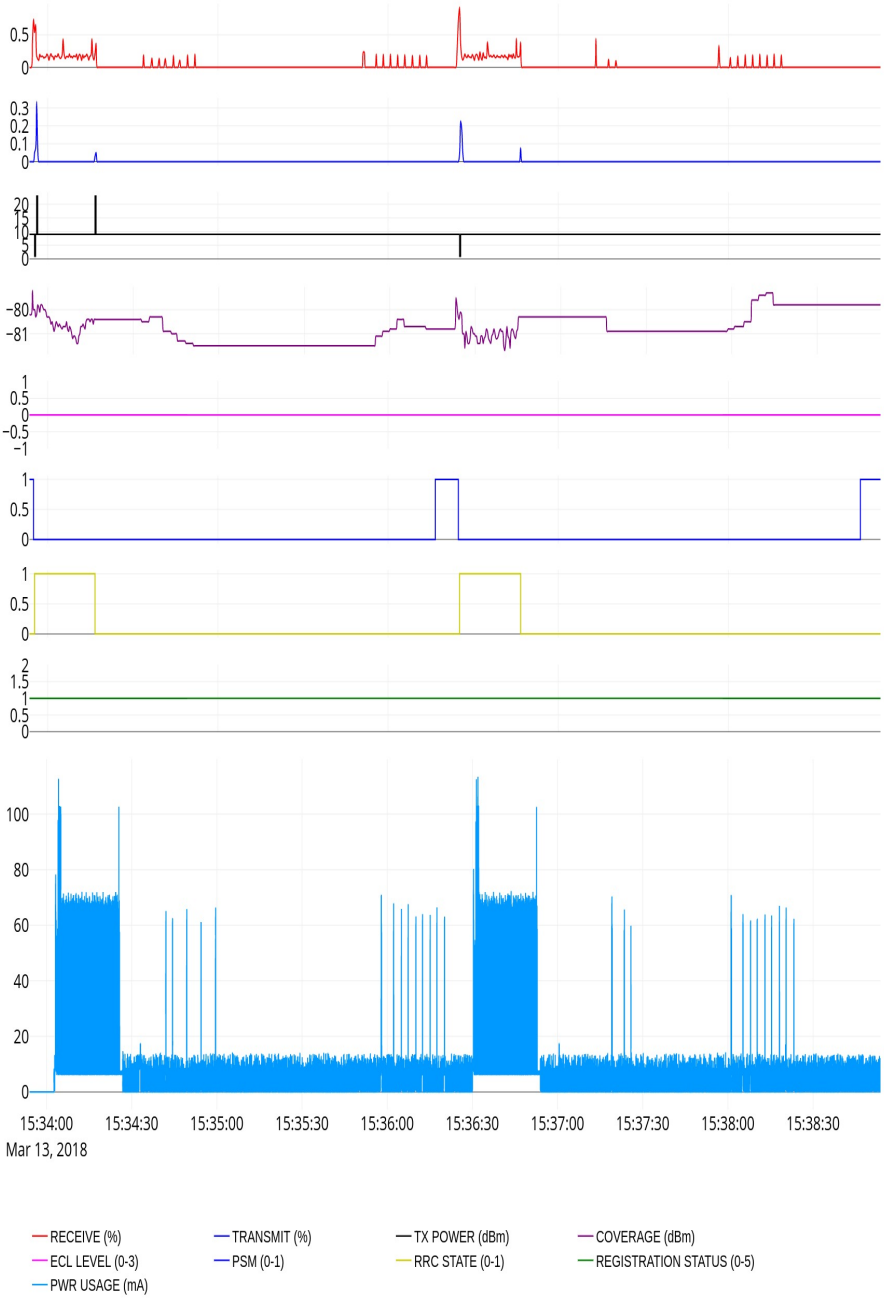

Figure 5.16: Short-term figure - normal behavior, Telia

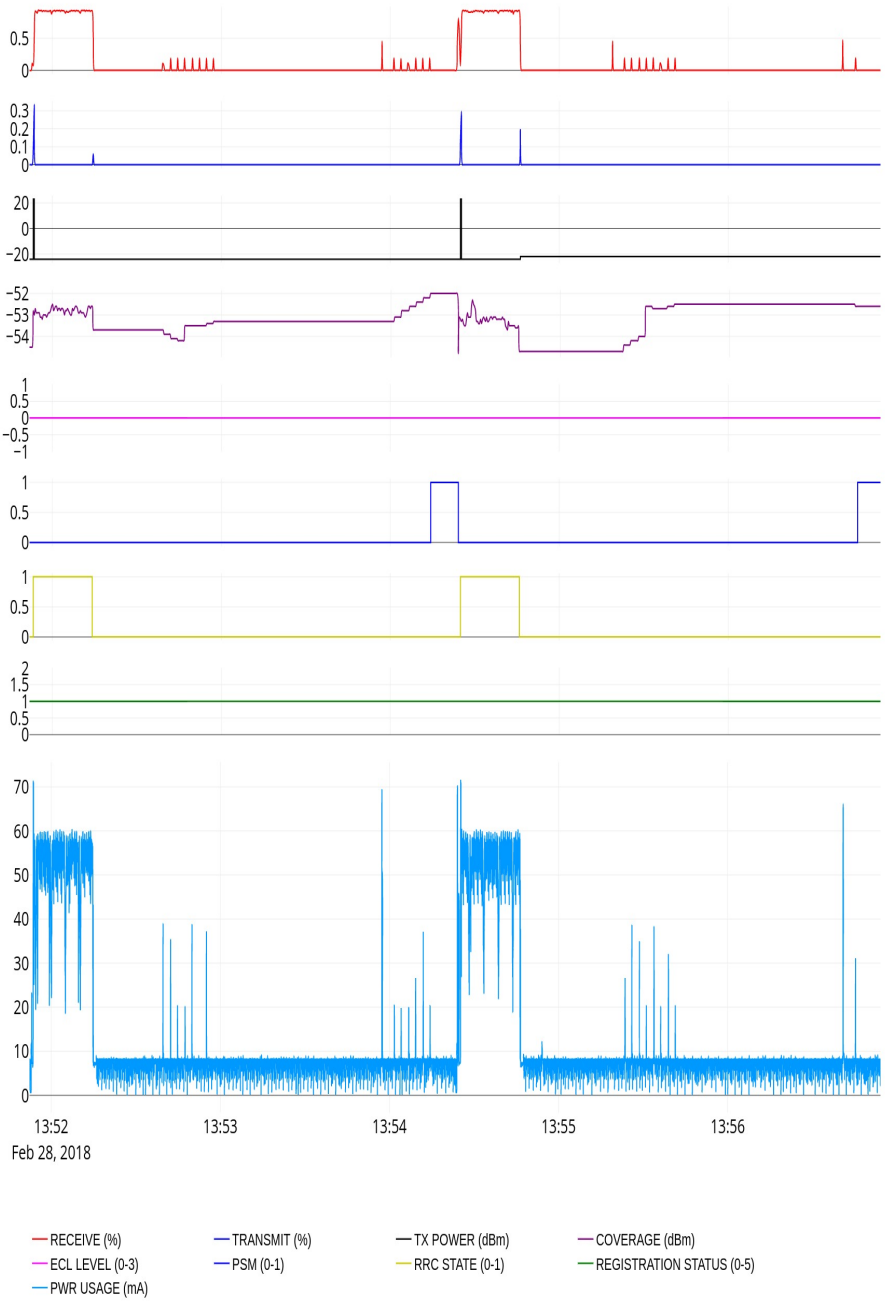

Figure 5.17: Short-term figure - normal behavior, Telenor

5.2.2 Downtime prior to transmit

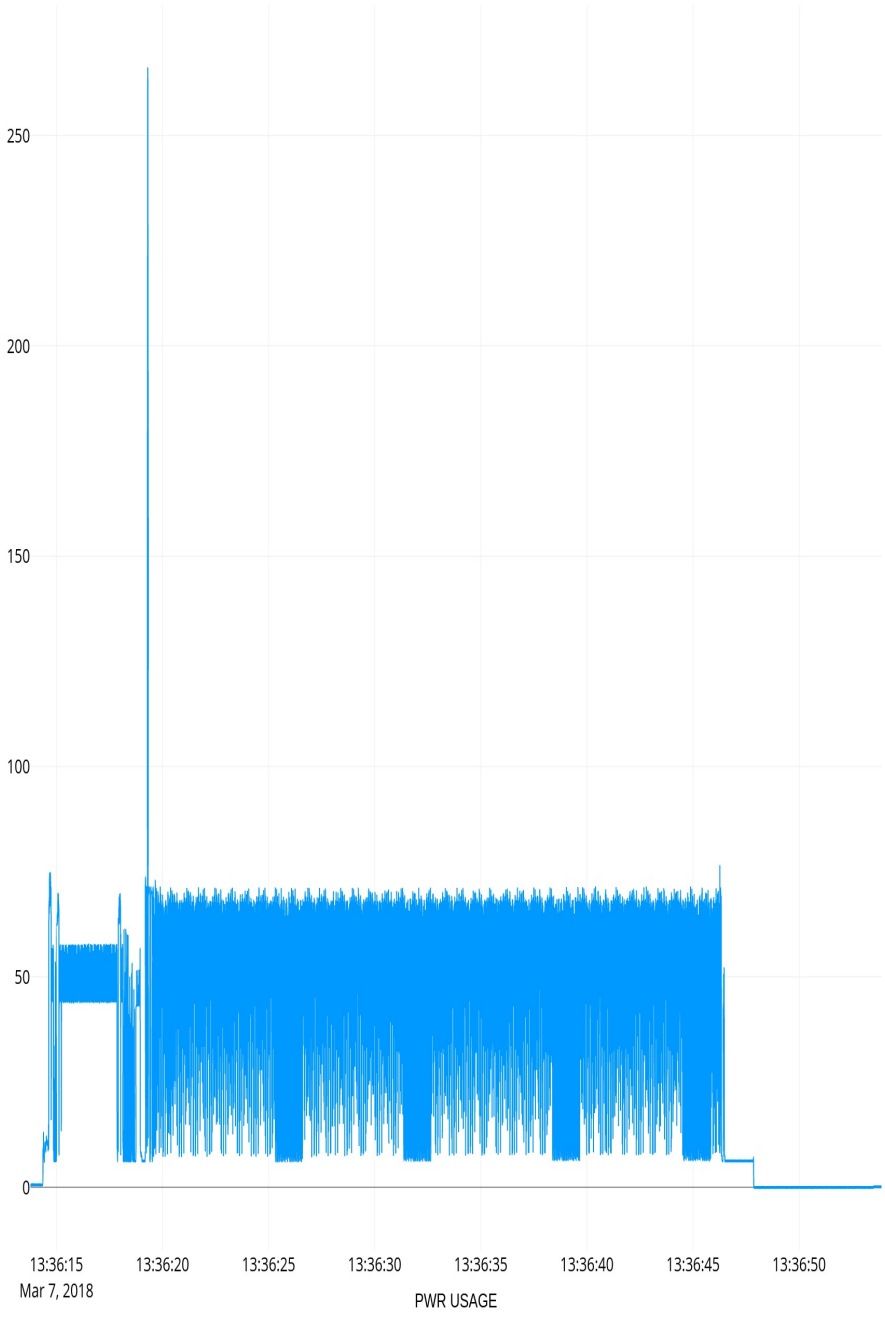

Figure 5.18: Short-term figure - loss of connection, without device logging

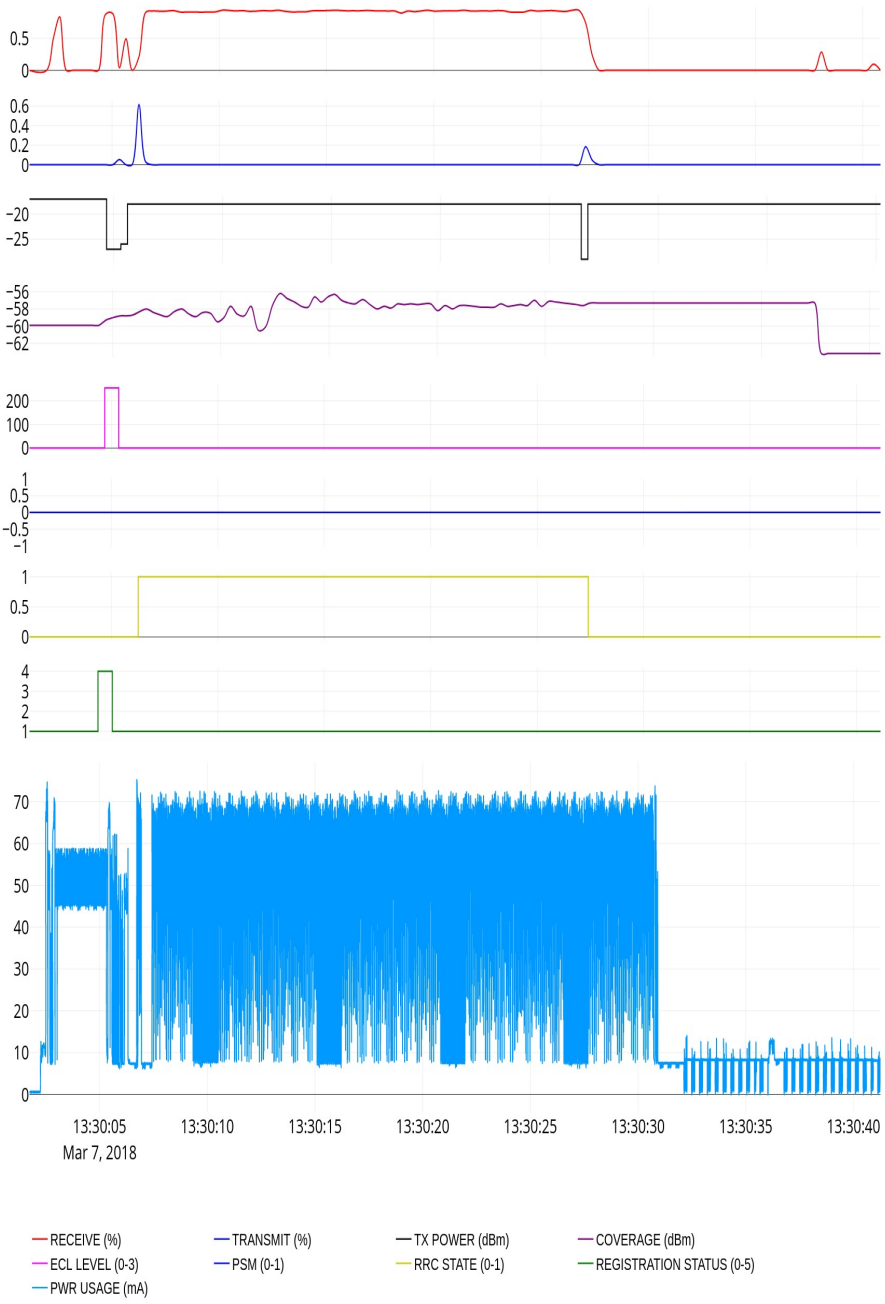

Figure 5.19: Short-term figure - loss of connection, with device logging

5.2.3 ECL level

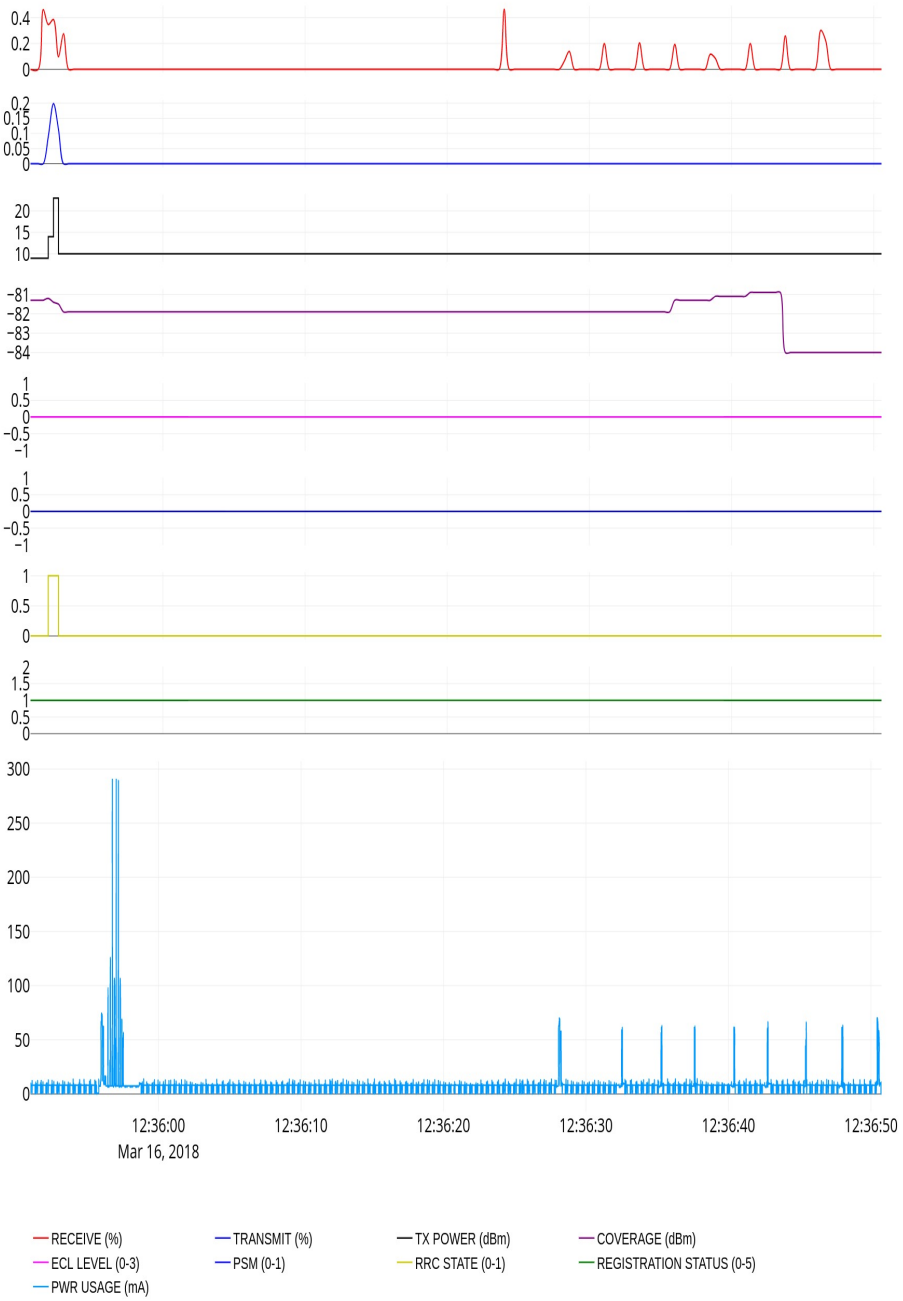

Figure 5.20: Short-term figure - ECL 0

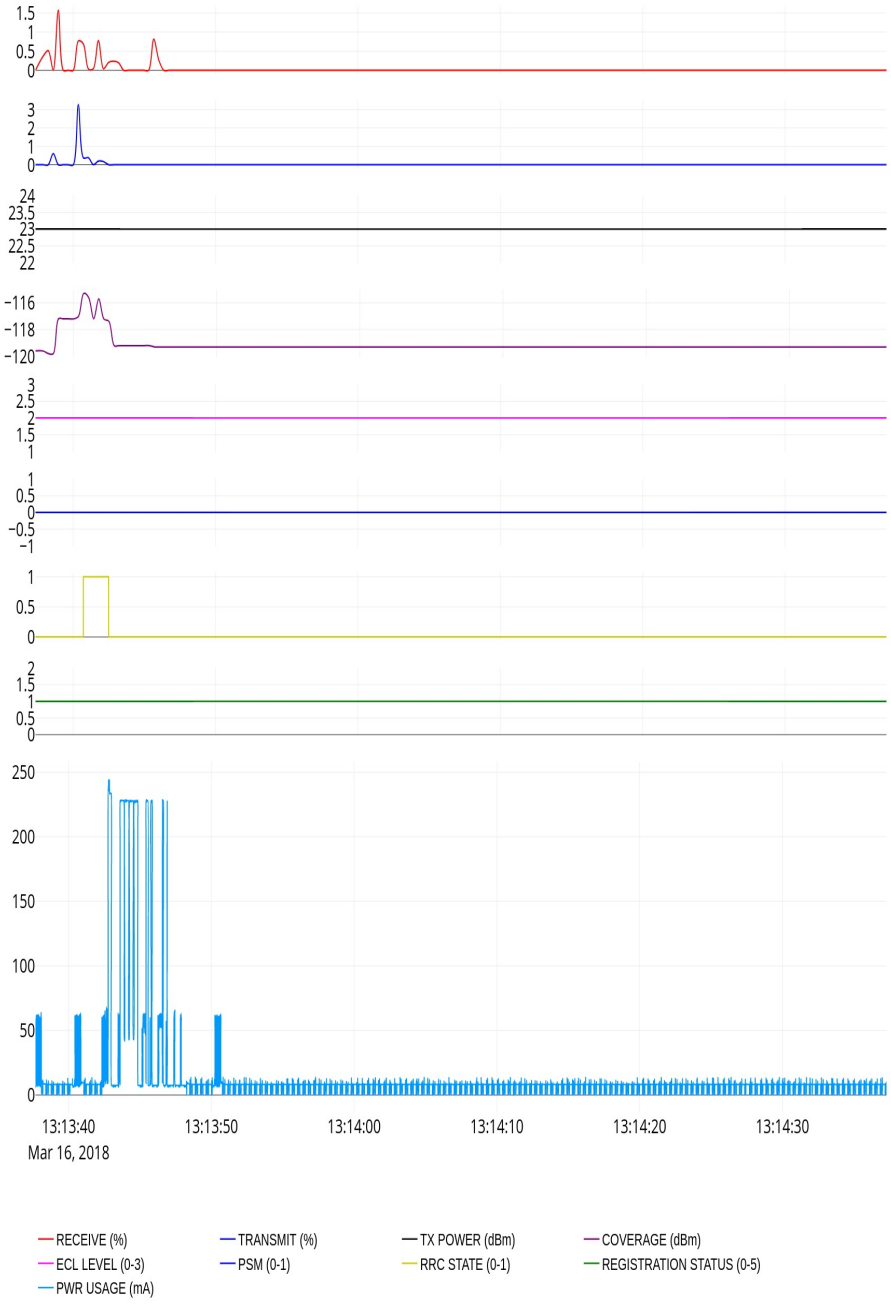

Figure 5.21: Short-term figure - ECL 2

5.3 Long-term tests

A device operating for a long time is prone to many bugs and software quality 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 the device managed to connect to the network. We define long-term test as a set of operations over a long period, typically one or several days. When trying to test many scenarios, the test period per network provider and location was 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 behavior of the chip over a longer period, which gives us statistics at a more abstract level, and can be combined 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. We have had some problems logging correct behavior of the device. It seems that at random points in time it looks as if the device looses connection to the network resulting in ECL set to 255, and receive/transmit timers are reset. It is at this stage unclear what evokes this behavior, but one reason might be related to PSM periods as the device shuts down all internal processes only keeping the clock and some logic alive. We had some progress to this problem in late March when we used the program, **nbiot_labtest.py**, for our long-term tests. This program constantly pulls the chip for statistical data, hence we might assume that the device is more active. Even with this program we saw signs of the same behavior and because of this abnormality some of the results are harder to make sense of, especially receive/transmit time. However the results we got were very much like the results from the short-term tests and we will use the good results to describe how the network performs.

Since the transmits towards the server contains all aspects of the long-term process we will try to include more information about the specific periods in this test, rather than to split the results into categories. We think that you will gain more knowledge by seeing relations between coverage, latency and other statistical perspectives at the same time. Most of the figures included in this section is taken from the web application so if you are unfamiliar with the format please refer to section, 4.3.1 on page 53, for a more detailed description of the graphs. For the best experience we advice you to use the web application to view the data, but we will include the most interesting findings in each apprioriate section.

5.3.1 Coverage and latency

One of the key features of NB IoT is the extended coverage. Devices should have good reception in normally bad coverage areas, such as tight city areas and far away mountain sides. The transmit power is $+23dBm$ which is very good, considering that a theoretical $+3dBm$ step is actually doubling in power output. The coverage is not only related to

the distance between the UE and the base station, but also the obstacles between. By looking at the map, 3.4 on page 34, we can see that the distance between the UE and base station at UiO and Q-Free is low, but the reception was quite different. At Q-Free, with Telenor, the reception was normally at $-60dBm$, while at UiO with Telia, the reception was normally at $-80dBm$. The main reason for the big difference is due to what lies between the sender and receiver. At Q-Free there are very few obstacles, hence the signal is passed directly to the base station. At UiO there are several buildings between the UE and the base station, resulting in lower reception. It would be interesting if Telenor also had a NB IoT enabled base station at the same location at UiO so that we could compare the two network providers more closely. We might assume that the two network providers use the same technology to offer NB IoT, but it is probably not from the same producer which might explain the big difference in reception as well.

In the following two sections we will give you insight in two longer transmit sequences. Because of the issues we have had with the long-term test we have focused mostly on one of the sequences with Telenor since their network is the most stable of the two. In section, 5.1.3 on page 64, we discussed what happened on the device at the occurrences of these reconnection periods.

Telenor

We will be looking closely at a transmit session over 5 days, 23.03-28.03, from UiO with Telenor's network. We have included two figures taken from the web application, 5.22 on page 87 and 5.23 on page 88, which we will use to point out certain aspects of the normal transmit process. Looking at the first section we see a steady line for both coverage and latency. The coverage is around $-95dBm$ and the latency ranges from 1.5 seconds to 3.5 seconds on average. We classify this as quite good coverage and resembles normal behavior. A thing we noticed is the flow of the latency line. We can see a clear interval between the high latency spikes and the low. We have not been able to find the origin of this behavior, but it most likely has something to do with the process in the core network since both the server and the client in this setup behaves the same way at all times. Another indication that it originates from the core network is that my supervisor, Q-Free's R&D manager, Ola Martin, also has seen this behavior in his tests[35]. Another interesting thing to view is receive/transmit time in the second figure. Looking at the transmit graph we can see that a normal transmit uses approximately 0.3 seconds which is relatable to the short-term tests. Also looking at the receive time graph the normal receive time between two transmits are around 21 seconds since

these transmits were not using the RAI flag. This is also very well in terms of what we saw in the short-term tests. This is useful information since we now know that this is not only a one time occurrence, but is actually the normal behavior of a receive/transmit process without RAI set.

Looking at the same period we can see three clear latency spikes up towards 10 seconds without any indication of bad reception. However at each latency spike we can see a similar spike in receive time and this indicates that something happened in the network which the device needed to listen in on, hence the transmit towards the server is delayed. The specification states a maximum latency of 10 seconds, so these transmits are on the edge of what is normal behavior. After a while, around 16:00 23.03, we see that the trend shifts. The coverage is the same, but the latency suddenly fluctuates up towards 5-6 seconds and this happens with the same interval flow as previously. To try to see if the behavior is related to the test program we tried to restart the program at, 20:36 25.03, now with RAI flag set. The same behavior continued until I restarted the development kit at 14:00 26.03, where we can see that the coverage and latency normalizes to what we classify as normal behavior. A thing you might have noticed is the drop in receive/transmit time. There is a noticeable behavior change in receive time when RAI is set and this is very logical since the device goes to sleep directly after the transmit process has finished. Even though the transmit process only covers $\tilde{0}.3$ seconds there are still some overhead related to the transmit process so the typical receive time for Telenor is 2-3 seconds, which is approximately ten times lower than with RAI not set. It is hard to see from the figure, but the transmit time is also reduced to some extent while using the RAI flag. This is related to lowering the network communication overhead of a transmit. The average transmit time with RAI set is approximately 0.15-0.2 seconds, which is around 50% less than without RAI set. Since the transmit process involves higher power usage a decrease in time usage is valuable.

From 10:00 to 12:00 26.03 we can see three spikes in the receive time, which might indicate trouble with the connection towards the eNB. If the device detects loss of connection it will remain in RRC connected mode for a longer period, hence pulling data on a regular basis. This is not something we want happening since it consumes very much power.

At 12:14 27.03 we added a $-20dBm$ attenuator to the development kit hence the coverage was reduced to around $-120dBm$. The reason why we add attenuators is because it is interesting to see how the network performs in these conditions, even though it will hopefully not be the case of most applications. First of all we can see that the base latency is around the same, but the interval we saw earlier is not as clear as before and the overall latency is higher. More interesting is to watch the receive/transmit

time graphs, as they show drastic behavior changes. We begin looking at the graph for transmit time, where we can see that the device retransmits packets very often. We see that many transmits uses around 2 seconds, while others are between 4-8 seconds. This is close to what was described in section, 2.3.7 on page 24, meaning that in addition to time spent in transmit mode the transmit it self also uses maximum power for each transmit. This behavior is extremely battery deficient and will cause poor lifetime.

Moving over to the graph with receive time we can see that the minimum receive time is still 2-3 seconds, but now the line is not very stable. We see more tendencies to spikes as the device has to use more time communicating with the network because of the poor reception. However the receive time is not affected to the same degree as the transmit time and is kept under 10 seconds for most of the transmits. Looking at the two graphs the transmit time increased with approximately $2.5S/0.2S = 12.5$, while receive time only increased with approximately $5S/2.5S = 2$. This is not surprising since we know that at ECL 1 and 2 the device retransmits packets frequently to achieve higher receive rate at the eNB.

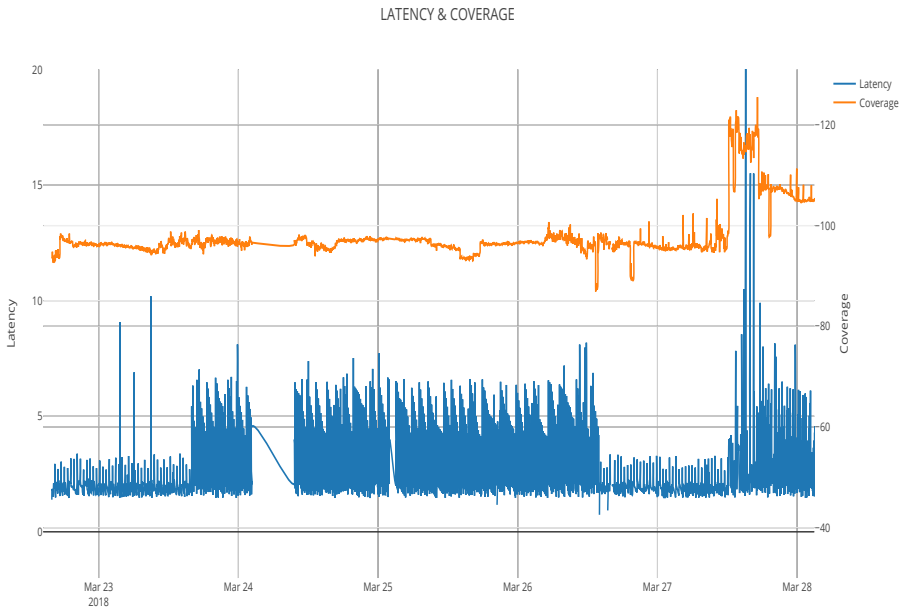

Figure 5.22: The figure displays a transmit session at UiO with Telenor over 5 days, 23.03-28.03, with the latency and coverage graph. Visit, webapp, for more details.

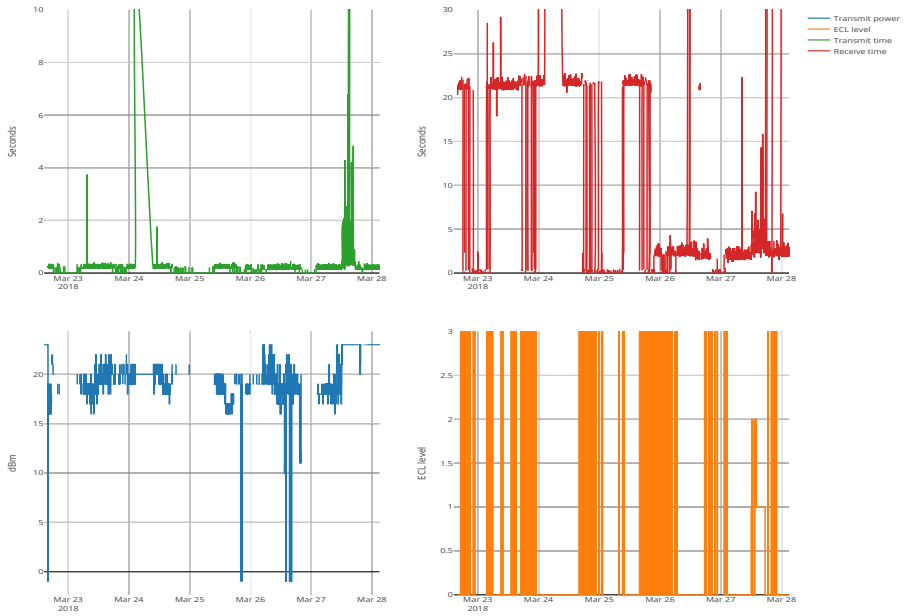

Figure 5.23: The figure displays a transmit session at UiO with Telenor over 5 days, 23.03-28.03, with the statistics graphs. Visit, webapp, for more details.

Telia

Transitioning over to Telia we will look closer at one of the sequences recorded with Telia's network at UiO, from 20.03 to 23.03. The setup is similar and there are obvious differences. We have included two figures, 5.24 on the next page and 5.25 on page 90, which illustrates the result. Looking at the beginning of the first figure we see that the graphs are comparable to Telenor's result. However, as you might have noticed most of the transmits are less stable, which affects the results. Since we don't get any feedback from the network it is hard to state the reason for this instability, but referring to section, 2.4 on page 25, we know that Telia deploys NB IoT in-band within their LTE deployments. We know that if there are many users in a certain area, this can cause poor reception and latency 2.2.1, hence our results reflect what might be this kind of behavior.

In addition the problem with reconnection is worse with Telia's network. This is probably related to the instabilities we have seen. Since the results are degraded we want you to focus on the results from the previous section as a kind of benchmark to what we believe is the closest to stabile behavior at this moment. However, the results from Telia shows an improvement in time spent in receive mode with RAI not set. Looking closer at the last part

of the graph we see a steady sequence without reconnections. This show us that when the transmit process behaves properly the device spends around 6 seconds in receive mode between each transmit. As stated in section, 5.1.1 on page 60, Telia uses approximately 60% less time in RRC connected mode resulting in lowered power consumption. At first we thought that this was the reason for the instabilities we saw, but we see the same behavior with RAI set. When RAI is set the two networks perform more or less identical if we only consider transmit and receive time.

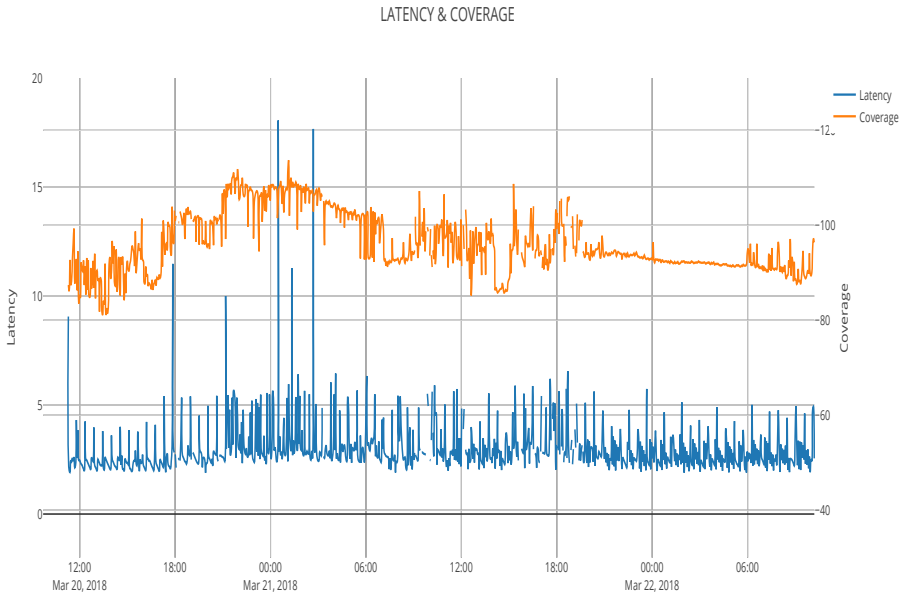

Figure 5.24: The figure displays a transmit session at UiO with Telia over 3 days, 20.03-23.03, with the latency and coverage graph. Visit, webapp, for more details.

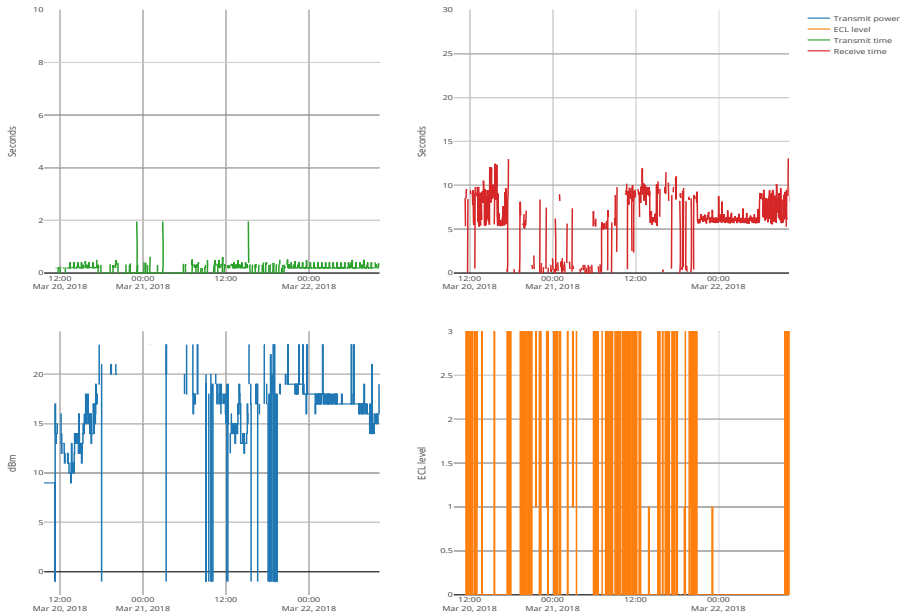

Figure 5.25: The figure displays a transmit session at UiO with Telia over 3 days, 20.03-23.03, with the statistics graphs. Visit, webapp, for more details.

5.3.2 Cell selection

When the device has bad reception it will try to reselect which cell it is connected to. We wanted to test how this affects the battery lifetime since this might happen frequently if the device is located equally far from two cells with approximately the same signal power. It might then jump back and forth between these cells which could drain the battery. However, because of the limited NB IoT enabled eNB's we were not able to test this hypothesis. We know that an attempt to connect to a network leads to higher power usage [5.1.7], hence reconnecting to a new eNB will probably use a similar amount of power.

5.3.3 Transmit Success Rate (TSR)

For an IoT application receive rate is not a priority, but it is important with stability and high uptime. All transmits towards the server includes a message id and we have used this to generate statistics about the transmit success rate. The program **uptime.py** takes in three parameters, node id(-c), start date(-start) and end date(-end). If you specify the start and end dates the program will only include data within the given period. This

option came handy due to issues related to the downtime on the server. The program calculates the TSR for sequences with a total number of transmits higher than 5 for the given period an prints a summary of the results. We have investigated all continuous sequences and used that to base our results upon. In listing, 5.1, you can see the output of one of the results, which displays data for id3 between **2018-03-13T15:00:00+02:00** and **2018-03-15T20:00:00+01:00**. Here we can see that in this period we found one sequence with a total transmits of 438 and actual received transmits of 426, which results in a TSR of 97.26%.

Listing 5.1 uptime.py example

```
1  Statistics about collection:  id3
2  Transmits:  438
3  Received transmits:  426
4  Uptime:  97.26027397260275
5  Total uptime:  [438. 426.  97.26027397]
```

We have looked at the sequences from UiO and can see that the TSR for Telenor and Telia is pretty similar to the listing example. We see that some packets are lost, which surprised us. Considering the number of devices connected to NB IoT at the point of the tests, we should have seen a higher TSR. A packet loss of ~3% is not very high, but if we imagine devices located in cities with device density of the technical specified limit of 50 thousand, the packet loss will probably increase resulting in degraded quality of service. We don't want the developers to have to implement retransmit logic in their applications so it will be interesting to see how the TSR changes as more devices connect to the network.

Chapter 6

Deviations

6.1 Imprecise clock

With the command **AT+CCLK?** we get the time from the network. The timestamp is normally precise, but we did see that after a while that the time skewed and the latency on the server became negative. The first hours of transmits were however precise and we saw that the time fetched from the network was good enough for some applications. With the hardware and software of our test chip it was not possible to sync the clock at specific intervals, but this would be necessary if you wanted to use the internal clock.

6.2 Imprecise NUESTATS

We believe that **NUESTATS** has given us great feedback about the behavior of NB IoT. As stated, the command is not 100% accurate and we did also see certain issues related to ECL, receive/transmit time, but in the end, when we combine the results from our test applications it is clear what the device actually does. As stated in sections, 2.4.1 on page 25 and 5.3 on page 83, the short-term tests gives the best picture of the state of the network at the time of the testing phase.

6.3 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.

6.4 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.

6.5 Theoretical vs. practical power usage

We have seen that the theory behind conversion between dBm, mA and mWh does not apply to the expected extent. We believe that calculating power usage based on the theory gives an indication of the power consumption, but it is only when using the appropriate tools that you will achieve the best understanding of power usage. In addition we know that if a transmit uses $+23dBm$ it only uses maximum power for a fraction of the actual transmit time.

Chapter 7

Conclusions and future work

In this chapter we will try to sum up the most important features of NB IoT and give a status report on the current situation. We will also give a short introduction to best practice guidelines, as well as combining the results we have.

7.1 Real world application guidelines

The testing phase has given us great overview of what we can expect of NB IoT. There are many possiblilites and with our results we will give you a short introduction to what we believe are best practive guidelines for NB IoT in regard to a real world application. We will use Q-Free's parking sensors specifications to estimate power usage. The sensor is equipped with two 3600mA batteries at 3.6 volts, which adds up to a maximum 25 920mWh.

Many developers will have a hard time deciding the transmit interval because it is difficult to predict the battery usage without doing any tests. Before calculating the power usage we need to specify what packet size we recommend. We saw in section, 5.1.6 on page 70, that there was a steady power usage increasement when we increased the packet size, which is logical. By lowering the packet size you will be able to increase the transmit interval if it is necessary. We recommend that your application normally transmits small packets, around 50 bytes, and if necessary have an alternate packet, between 100-200 bytes, which the device can transmit from time to time. If we use packets of 100-200 bytes and with RAI set as a basis, the device consumes around $0.2mWh$[13]. We have seen and tried different transmits intervals and have concluded that one transmit per hour is a good tradeoff. This results in a total power consumption of, $0.2mWh * 24 * 365 * 10 = 17\,520mWh$, and gives the device battery for

normal operations, transmits and edge case handling.

The alternate packet may contain a more detailed overview of the state of the device so that the server knows the state of each sensor. This is useful for maintenance and gives an indicator if something is wrong. When transmitting this larger packet you may also unset the RAI flag to allow for downlink communication. If your application only uses RAI it will never know if there is downlink data from the network, so by using the opertunity with the alternate packets your application will be able to receive downlink data which might be useful in some cases. A good example of a downlink message is for configuration purposes. Maybe you want to add data properties to your applications, or change the transmit interval. If so, you can do this if your application supports it and you disable RAI regularly. Keep in mind that the application can't be changed after it is installed, so you are better of with many safety mechanisms in case of errors.

Another dilemma related to an application running for several years is network downtime. There will be periods where the device will loose connection to the network, either because of a power outage or because of unexpected behavior. Your application will need to handle these situations and we belive there are a couple of approaches to this problem. If the application detects network failure we believe it should go into a configured reconnection period. This period will cover one day and do a number of things. Firstly the device will try to reconnect to the network before a set of transmits. We believe it is wise to reconnect at an exponential rate, so you will reconnect at a rate equal to the power of 2. Say you loose network connection at 10:05, we recommend trying to reconnect to the network a set period prior to the following transmit. If the connection is not up at the time of the transmit we recommend trying to reconnect at the second following transmit, and then the fourth following transmit. In addition your application can reboot the device after a number of reconnection attempts. Rebooting the device will be a last resort option and should not be reattempted many times since it drains the battery. If the device does not manage to reinitiate connection towards the network after a day your application can restart the reconnection period, but you might want to reduce the number of reconnection attempts to prehibit more energy waste.

7.2 Combining the results

We have discussed many aspects of NB IoT and tried to test them with Telenor and Telia, which produced results indicating the state of NB IoT. In this section we will try to highlight the pros and cons of NB IoT in general

and give examples from our results. As stated in section, 2.5 on page 28, our long-term results are affected of early stage hardware and software, hence the main focus will still be at the short-term tests. We have split this section into subsections, each focusing on a specific NB IoT feature.

7.2.1 Transmit process

We have seen many transmit examples and in general all are very much like the specifications state. The transmit starts with a period of negotiation process, followed by the actual transmit and a optional RRC period. With RAI set, Telenor and Telia performed very similar and according to the specifications. However with RAI unset, Telia increases battery lifetime by using what we believe is DRX within the RRC period. This resolves in approximately 60% reduced power usage and can affect your application if the transmits do not use RAI. For most application this will not be a problem since it is recommended to use RAI for transmits unless really necessary.

In genera the results related to the transmit process are promising. We did see power spikes up to 250mA in many transmits when having good or excellent coverage. The spike was not present at every transmit, which leads us to believe that this is a network related issue. However the spike is very short and only happens once per transmit and will probably not be a major problem even though the chip uses more power than necessary.

7.2.2 Coverage and latency

Coverage and latency is closely related and our results show that NB IoT meets the requirements of good coverage and latency below ten seconds. The specification states a coverage up to 35 kilometers, but this is without any obsticles, hence the coverage rate falls in city areas.

7.2.3 Connection time

In section, 5.1.7 on page 73, we showed the results from the reboot tests. These tests showed us what happened during the connection period and we noticed that there were big differences between Telenor and Telia. Telenor used a lot longer time to connect to the network, hence increasing the power usage. We did not manage to find the reason for the long connection time, but it is worth noting the difference, since this might affect the battery lifetime in certain areas where reconnection or reboots

will occour often. This is neither a big concern, since the network should be stable and there is potentially no need to reboot the device.

7.2.4 Uptime

Through our long-term tests we have tested the stability of the networks and at that time the results were poor. The uptime was not as good as expected and the device needed to be rebooted to reinitiate normal behavior. We saw that Telenor's network was more stable than Telia, but not by much.

7.3 Final remarks

It has been a rewarding period working with NB IoT. We believe that this is a promising technology which will improve many applications. IoT is growing as we speak and there are companies waiting for good LPWAN's. As stated in section, 2.4 on page 25, there is a higher demand for LTE-M1 in USA which will postpone the production of NB IoT hardware to around 2019. The good thing is that Telenor and Telia hopefully will have their NB IoT implementations in production by 2018, meaning that the network is ahead of hardware production - which is unusual. We believe that there is one major drawback in the network today, which is the ability to investigate what happens when the device fails. The network is very much like a black box without any indication of the status, which can be frustrating for developers. Hopefully the network providers will have documentation related to NB IoT, thus the developers might have an easier time configuring their application to meet their requirements. Currently we believe there are to many unknown states and bugs in the network, and this needs to be handled before the network providers production set their implementations. Most companies will have to wait for the next generation hardware which increases the stability, and at that point we can start to create solid applications which are battery efficient and easy to maintain without the high level of logic currently needed.

Bibliography

[1] *1 x 40 Q-FREE, Telenor - device logging*. URL: http://158.39.77.97:9000/#/results/Q-FREE_TELENOR_5.02_2018-03-07_1_0x0_40_1_200.

[2] *1 x 40 Q-FREE, Telenor - no device logging*. URL: http://158.39.77.97:9000/#/results/Q-FREE_TELENOR_5.02_2018-03-07_0_0x0_40_1_200.

[3] *1 x 60 UiO, Telia - ECL 0*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_5.02_precision_2018-03-16_1_0x2_60_1_50.

[4] *1 x 60 UiO, Telia - ECL 2*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_5.02_precision_+30dBm_att_2018-03-16_1_0x2_60_1_50.

[5] *1MA266_0e_NB_IoT.pdf*. https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_application/application_notes/1ma266/1MA266_0e_NB_IoT.pdf. (Accessed on 03/26/2018).

[6] *2 x 150 Q-FREE, Telenor*. URL: http://158.39.77.97:9000/#/results/Q-FREE_TELENOR_2018-02-28_1_0x0_150_2_100.

[7] *2 x 150, Q-Free Telia, weird behavior*. URL: http://158.39.77.97:9000/#/results/Q-FREE_TELIA_5.02_precision_2018-03-13_1_0x0_150_2_100.

[8] *2 x 150 UiO, Telia*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_5.02_precision_2018-03-13_1_0x0_150_2_100.

[9] 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).

[10] *audreyt/node-webworker-threads: Lightweight Web Worker API implementation with native threads*. https://github.com/audreyt/node-webworker-threads. (Accessed on 04/11/2018).

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

[12] 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).

[13] *Details - RAI set*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_long_term_2018-03-22_0x2_30_20.

[14] *Details - RAI unset*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_long_term_2018-03-22_0x0_30_20.

[15] "Email correspondance."

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

[17] *expressjs/express: Fast, unopinionated, minimalist web framework for node.* https://github.com/expressjs/express. (Accessed on 04/11/2018).

[18] *Finnsenderen*. (Accessed on 04/01/2018).

[19] *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).

[20] 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).

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

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

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

[24] *LearnBoost/cluster: Node.JS multi-core server manager with plugins support.* https://github.com/LearnBoost/cluster. (Accessed on 04/11/2018).

[25] 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).

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

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

[28] *mcollina/node-coap: CoAP - Node.js style*. https://github.com/mcollina/node-coap. (Accessed on 04/11/2018).

[29] *moment/moment: Parse, validate, manipulate, and display dates in javascript*. https://github.com/moment/moment. (Accessed on 04/11/2018).

[30] *mongodb/node-mongodb-native: Mongo DB Native NodeJS Driver*. https://github.com/mongodb/node-mongodb-native. (Accessed on 04/11/2018).

[31] "NACK mail correspondance between Ola and Ublox."

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

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

[34] 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).

[35] "Ola Martin Lykkja."

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

[37] *Power comparison*. URL: http://158.39.77.97:9000/#/results/Q-FREE_TELENOR_SHORT_TEST_2018-02-28_0_0x2_5_1_100.

[38] *Power example*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_5.02_precision_2018-03-16_1_0x2_60_1_512.

[39] 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).

[40] *Telenor reboot*. URL: http://158.39.77.97:9000/#/results/UiO_TELENOR_5.02_precision_reboot_2018-03-16_0_0x0_120_1_0.

[41] *Telia mail thread*.

[42] *Telia reboot*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_5.02_precision_reboot_2018-03-16_0_0x0_60_1_0.

[43] *thesis/at_command_test.py at master · henninghaakonsen/thesis*. https://github.com/henninghaakonsen/thesis/blob/master/code/at_command_test.py. (Accessed on 04/11/2018).

[44] *thesis/nbiot_labtest_details.py at master · henninghaakonsen/thesis*. https://github.com/henninghaakonsen/thesis/blob/master/code/nbiot_labtest_details.py. (Accessed on 04/11/2018).

[45] *thesis/nbiot_labtest.py at master · henninghaakonsen/thesis*. https://github.com/henninghaakonsen/thesis/blob/master/code/nbiot_labtest.py. (Accessed on 04/11/2018).

[46] *thesis/power_calculator.py at master · henninghaakonsen/thesis*. https://github.com/henninghaakonsen/thesis/blob/master/code/power_calculator.py. (Accessed on 04/11/2018).

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

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

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

Acronyms

3GPP 3rd Generation Partnership Project. 1, 5, 12, 15

AuC Authentication Center. 6

CAT-M LTE Cat-M. 11

COAP Constrained Application Protocol. ii, v, 38, 39, 49

dBm Decibel / referenced to milliwatts. 20, 21, 24, 25, 36, 56, 57, 61, 62, 64, 67, 84–86, 94

DDoS Distributed Denial-of-Service. 17, 18

DL Down Link. 4, 5, 21, 64

DRX Discontinuous Reception. 19, 20, 22, 23, 60, 62, 97

EARFCN Evolved Absolute Radio Frequency Channel Number. 37

ECL Coverage Enhancement Level. i, ii, vii, viii, 24, 25, 37, 43, 44, 56, 59, 61, 64, 66–69, 82–84, 87, 93

eDRX Extended Discontinuous Reception. i, vii, 21–24, 38, 60, 62

eMTC enhanced Machine Type Communication. 12

eNB Evolved Node B. 6, 34, 67, 75, 86, 87, 90

EPC Evolved Packet Core. i, vii, 5, 6

ETSI European Telecommunications Standards Institute. 15

GSM Global System for Mobile communications. 12, 16, 19

GSMA GSM Association. 7, 10

HLR Home Location Register. 6

HSS Home Subscriber Server. 5, 6, 18

IoT Internet of Things. vii, 1–3, 6, 9–11, 13, 15, 16, 18, 27, 28, 70, 90, 98

IoT Platform IoT Platform. 18, 19

IP Internet Protocol. 42

IPv4 Internet Protocol version 4. 16

ITS Intelligent Transportation Systems. 7

Kbit/s Kilobit per second. 12

LoRa . vii, 13

LPWAN Low Power Wide Area Networks. vii, 2, 3, 6–15, 17, 20, 23, 25, 26, 98

LTE Long-term Evolution. i, 2–7, 9, 11–13, 15–20, 22, 26, 27, 88

LTE-M1 . 12, 13, 22, 25, 26, 98

mA Milliampere. 20, 21, 44, 60, 62, 64, 65, 67, 94, 95, 97

Mbit/s Megabit per second. 11, 12

MME Mobile Management Entity. 5, 6, 18, 23

MPS Maximum Packet Size. 71

MT Mobile Terminal. 37

mW Milliwatts. 20, 21

mW/s Milliwatts seconds. 44

mWh Milliwatt hour. 20, 21, 24, 44, 65, 67, 73, 94, 95

NACK negative-acknowledgement. 57, 64, 67

NAT Network Address Translation. 18

NB NarrowBand. 15

NB IoT NarrowBand IoT. i, ii, v, vii, ix, xiii, 1–5, 8, 9, 11–13, 15–28, 31, 32, 35–37, 39–43, 45, 55, 59, 84, 85, 88, 90, 91, 93–98

NPM Node Package Manager. 48

OECD ORGANIZATION FOR ECONOMIC CO-OPERATION AND DEVELOPMENT. 7

PCI Physical Cell ID. 37

P-GW Packet Data Network Gateway. 5, 6

PSM Power Saving Mode. i, 17, 20–23, 38, 39, 44, 56, 60, 62, 73, 84

QoS Quality of Service. 6

RAI Release Assistance Indicator. vii, viii, 21, 39, 43, 45, 56, 60, 62, 64, 67, 69–73, 86, 88, 89, 95–97

REPL Read-Eval-Print-Loop. 40, 42

RRC Radio Resource Connection. i, 20–22, 24, 39, 44, 60, 62, 64, 69, 71, 73, 74, 86, 89, 97

RSRQ Reference Signal Received Quality. 37

S-GW Serving Gateway. 5, 6

SNR Signal to noise ratio. 37, 60

T3324 Active Time. 22, 38, 39

T3412 Extended TAU Timer. 23, 38

TDMA Time division multiple access. 22

TSR Transmit Success Rate. ii, 24, 90, 91

UDP User Datagram Protocol. 38, 41

UE User Equipment. 2, 5, 6, 9, 20–22, 24, 38, 39, 60, 67, 75, 85

UL Up Link. 4, 5, 21, 64

WAN Wide Area Network. 17