Dynamic Computation of Connectivity Data

Jon Arild Ekberg Meyer

Master of Science in Communication Technology
Submission date: February 2019
Supervisor: Peter Herrmann, IIK
Co-supervisor: Ergys Puka, IIK

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

Title: Dynamic Computation of Connectivity Data

Student: Jon Arild Ekberg Meyer

Problem description:

In the Intelligent Transport Systems (ITS) domain, the connectivity in certain places is important since, of course, data can only be exchanged between vehicles and their infrastructure (Vehicle-to-Infrastructure) if the cars have a wireless connection. To make decisions in certain ITS applications and data forwarding strategies, it is important with in-depth knowledge about the connectivity on the road network. With this information, cars can for instance make decisions whether to use V2I communication or use other cars as a relay (V2V – Vehicle-to-Vehicle) in areas of low connectivity. This knowledge must be up-to-date since connectivity may change dynamically and unpredictably.

In this master thesis we will develop a system in which cars regularly transmit connectivity data to an external server that constantly aggregates the connectivity status in certain areas. On request, the data for an area shall be transmitted to the cars. This will then allow cars to make decisions about the best way to handle communication in areas of low connectivity, e.g. dead spots.

As a platform, Android devices will be used in the cars. An app will be developed for the Android platform that can perform various measurements regarding the current connectivity (e.g. signal strength, round trip delay and bandwidth). Different metrics to express the connectivity will be tested and chosen based on what is feasible for the available equipment and appropriate for the application domain.

A server application will be connected to the Android app (the client) using socket programming. It will store measurements received from clients in a relational database (spatial database tools/extensions might be used since clients will collect location data) and aggregate these measurements into simpler information based on the chosen metrics. This information can then be utilized by the clients for communication decision making as described above.

Responsible professor: Peter Herrmann, IIK

Supervisor: Peter Herrmann, IIK

Co-supervisor: Ergys Puka, IIK

Abstract

The performance of the underlying communication technology plays an important role in Vehicular Networks (VNs). Strict requirements must be met in order to ensure necessary safety and robustness, as well as to provide user satisfaction. In reality, however, the available connectivity performance often does not meet these requirements. For instance, in rural areas the mobile network may still be using 2G or 3G technology, and dead spots can be present. In this thesis we focus on the mobile network as the mode of communication in VNs, with emphasis on such dead spots.

A client-server solution where connectivity data from sampling clients is processed has been developed. Connectivity data is collected using Android devices and uploaded to a central server. The devices may be used in different means of transport, such as cars or bicycles as well as by pedestrians. In real-time, the server processes and aggregates the data into structured data formats, which is also presented as graphs and geographical maps showing dead spots and other areas of low connectivity. These aggregated forms of data can then be requested by the means of transport, which can use the data to decide where to communicate and with which quality of service.

To build and test the system, we sampled data from two different geographical areas of Norway: Lofoten and Trondheim. Over 130 000 samples were collected from various types of networks. An in-depth analysis of these samples is provided. The resulting system is a scalable solution for handling large volumes of connectivity data from a multitude of sources.

Sammendrag

Ytelsen til den underliggende kommunikasjonsteknologien spiller en viktig rolle i Vehicular Networks (VNs). Strenge krav må oppfylles for å sikre nødvendig sikkerhet og robusthet, i tillegg til å gi brukertilfredshet. I virkeligheten er det derimot ofte slik at tilgjengelig ytelse ikke oppfyller disse kravene. For eksempel, i distriktene kan mobilnettet fortsatt være basert på 2G- eller 3G-teknologi, og dødsoner kan forekomme. I denne oppgaven fokuserer vi på mobilnettet som kommunikasjonsmodus i VNs, med vekt på slike dødsoner.

En klient-server-løsning der mobilnettverksdata leveres av målende klienter har blitt utviklet. Nettverksdata blir samlet inn ved bruk av Android-enheter og lastet opp til en sentral server. Enhetene kan brukes i forskjellige transportmidler, som biler og sykler, men også av fotgjengere. I sanntid prosesserer og aggregerer serveren dataene om til strukturerte dataformat, som også kan presenteres som grafer og geografiske kart som viser dødsoner og andre områder med dårlig forbindelse. Transportmiddel kan deretter forespørre disse aggregerte formene av data, og bruke dem til å avgjøre hvordan det skal kommuniseres og med hvilken tjenestekvalitet.

For å bygge og teste systemet ble det gjort målinger fra to forskjellige geografiske områder i Norge: Lofoten og Trondheim. Over 130 000 målinger ble samlet fra forskjellige typer nettverk. En dybdeanalyse av disse målingene er gitt. Det resulterende systemet er en skalerbar løsning for å håndtere store volum av måledata fra forskjellige typer kilder.

Preface

This master thesis was carried out by Jon Arild Ekberg Meyer during the fall semester of 2018 into the beginning of 2019. It is the final work of the Master of Science program in Communication Technology at the Department of Information Security and Communication Technology at Norwegian University of Science and Technology. Peter Herrmann acted as responsible professor and supervisor of the project. Ergys Puka acted as co-supervisor of the project.

I would like to thank both Peter and Ergys for all feedback for the entire duration of the project, both to practical work and to writing. Our meetings, described as “longer than usual”, have been intriguing and of great value to my work. I’d also like to thank my mother, Anne Ekberg Meyer, for familiarizing herself with the client application I developed in the project, and then providing more than half of the sample data of the entire project.

Contents

List of Figures

List of Tables

List of Acronyms

3GPP The 3rd Generation Partnership Project.

ASU Arbitrary Strength Unit.

CDMA Code-division multiple access

DBMS Database Management System.

DSRC Dedicated Short Range Communication.

EDGE Enhanced Data rates for GSM Evolution.

GPS Global Positioning System.

GSM Global System for Mobile communications.

HSDPA High Speed Downlink Packet Access.

HSUPA High Speed Uplink Packet Access.

HSPA High Speed Packet Access.

HSPA+ Evolved High Speed Packet Access.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

IIK Department of Information Security and Communication Technology.

IPDV IP Packet Delay Variation.

ITS Intelligent Transportation Systems.

JSON JavaScript Object Notation.

RAN Radio Access Network.

REST Representational State Transfer.

RTT Round-trip time.

SQL Structured Query Language.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

V2I Vehicle-to-Infrastructure.

V2V Vehicle-to-Vehicle.

V2X Vehicle-to-Everything.

VANET Vehicular ad-hoc network.

VN Vehicular Network.

Chapter 1 – Introduction

1.1 Problem area

The work of this thesis falls within the domain of Intelligent Transportation Systems (ITS). ITS combines information and communications technology (ICT) with the conventional world of transportation infrastructure. It has been developed and deployed over the past few decades to improve transportation safety, reduce environmental impact, promote sustainability and enhance productivity [1]. The rate of advances in technology of today in combination with an increasing concern for the environment of the planet makes ITS a highly relevant field of research.

1.2 Problem description and focus

With the development of ITS, infrastructure plays an increasing role in communication with vehicles. In a sophisticated scenario, routing, velocities of vehicles and distances between vehicles could be suggested by the infrastructure based on traffic conditions. This requires vehicles to have a wireless connection to the infrastructure, e.g. through the mobile network or with Dedicated Short Range Communication (DSRC). DSRC provides wireless communication capabilities for transportation applications within a 1000 m range in highway speeds [2]. The work of this thesis is concerned with the former: the mobile network.

Throughout the mobile network, network performance will vary depending on coverage, but also the condition of the underlying network. Poor performance can still be experienced with excellent signal strength if the network is congested. Knowledge and mapping of such poor (and good) network conditions is, of course, of great value to vehicle communication. Additionally, coverage mapping of the mobile network alongside the road could be useful for other interest groups, such as telecom companies. This information must be up-to-date, since network conditions can change dynamically and unpredictably. The focus of this thesis is as such developing a system that can provide vehicles with real-time connectivity information and identify areas with poor network performance.

There are many applications in ITS where vehicles communicate with each other, the infrastructure or a combination of both. Examples are intersection management, remote diagnosis/repair warnings, road conditions and media downloads. In the following sub sections, we present a few examples of how the work of this thesis could be used in real-world scenarios.

1.2.1 Improving routing decisions and safety

With knowledge about a poor connection area in advance, systems for ad-hoc communication can be selected for use earlier. A car does not need to sense the network state itself before deciding whether to communicate with infrastructure or directly with other vehicles. Information about potential hazards on the road, such as a traffic accident or an icy road, could then potentially be delivered faster.

If the mobile network is unavailable, being aware that direct communication with other vehicles is the only usable option could have other implications to safety. Without information also delivered from the infrastructure network, a car should operate safely within the latency constraints implied by ad-hoc communication. For instance, a self-driving car could adjust its speed accordingly.

1.2.2 Smarter media buffering

Adaptive streaming protocols (MPEG-DASH, HLS, ...) are used by many media content providers today to deliver audio and video. In adaptive streaming the quality of for instance a video will be lowered if the client has a low bit rate. Users are more willing to accept a few minutes of bad quality video rather than an interrupted experience [3]. Also, to conserve bandwidth, only a certain amount of content will be buffered at a time. If the user decides to close a video before finishing watching it, downloading the entire video would be a waste. Adaptive streaming is illustrated in the following figure.

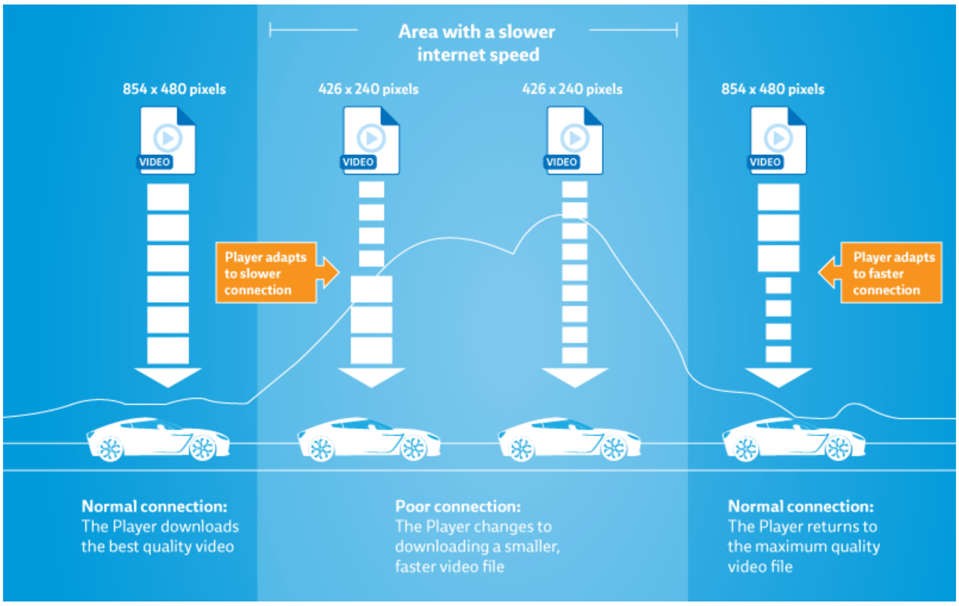
Area with a slower internet speed
854 x 480 pixels
426 x 240 pixels
426 x 240 pixels
854 x 480 pixels
VIDEO
Player adapts to slower connection
Player adapts to faster connection
Normal connection: The Player downloads the best quality video
Poor connection: The Player changes to downloading a smaller, faster video file
Normal connection: The Player returns to the maximum quality video file

Figure 1.1: Adaptive streaming [3]

In an area of very poor or no connectivity at all, even worst quality of content will not be delivered to the user. This is the case with mobile network dead spots. With prior knowledge of a dead spot, bandwidth can be utilized in a smarter manner to ensure a richer user experience with no interruptions. For instance, part of the bandwidth before the dead spot can be used for buffering more content.

1.3 Methodology

A client-server software architecture is used. Cars are equipped with Android devices sensing the network performance of the current mobile network, measuring various performance metrics such as signal strength, round-trip time and packet loss. These measurements are uploaded to a server that processes them. Upon request, the server provides network performance information of specified geographical areas, by using the earlier measurements provided by other cars.

1.4 Report structure

Chapter 2 presents background material of relevance to studying the rest of the chapters.

Chapter 3 presents and summarizes related work.

Chapter 4 describes the design and implementation of the developed software. A sampling client application is first presented, then the server application that receives and processes these samples.

Chapter 5 contain results and discussion. Issues faced during practical work are first shown. Then, a statistical analysis of results is performed and discussed. Application scalability is also addressed here.

Chapter 6 is the conclusion of the thesis work.

Chapter 7 contains proposed future improvements to our work.

Chapter 2 – Background

This section provides background knowledge for the problem area. As we throughout the thesis perform network measurements in mobile networks, knowledge about different types of mobile networks and how they have developed is useful. Vehicular Networks (VNs) are an important part of network communication in the ITS domain and are also shortly introduced. Mobile network dead spots are then discussed, as we have a need of identifying them in this thesis. Lastly, different techniques for measuring network performance are introduced.

2.1 Mobile network types

Global System for Mobile communications (GSM) and Code-division multiple access (CDMA) are different channel access methods in mobile networks, where GSM is dominating the market share. Measurements performed in this thesis are done purely in GSM networks, and CDMA technology is therefore not addressed. However, note that a variant of CDMA (WCDMA) is used in Universal Mobile Telecommunications System (UMTS), a GSM 3G technology.

The 3rd Generation Partnership Project (3GPP) is a collaboration between several telecommunication standard associations. The scope of 3GPP [4] includes:

- GSM and related 2G and 2.5G standards
- UMTS and related 3G (and 3.5G) standards
- Long-Term Evolution (LTE) and related 4G standards
- Next generation and 5G standards

2G technologies became available in the 1990s, being enhanced with packet-data capabilities into so-called 2.5G. Enhanced Data rates for GSM Evolution (EDGE) is considered a 2.5G standard. 3G requirements were specified by the International Telecommunication Union (ITU) as part of the International Mobile Telephone 2000 project. Enhancements to 3G standard were later made, being 3.5G technologies. The first enhancement was High Speed Downlink Packet Access (HSDPA). It was introduced in 3GPP Release 5, followed by High Speed Uplink Packet Access (HSUPA) in 3GPP Release 6. The combination of HSDPA and HSUPA is referred to as High Speed Packet Access (HSPA). HSPA+ (HSPA evolution, evolved HSPA) came in Release 7 with further improvements in later releases.

LTE was developed to ensure continuity and competitiveness of the 3G system, not being part of the fourth-generation mobile systems (4G). Further enhancements to LTE led to LTE-Advanced, that completely fulfilled the requirements set by ITU for 4G. Further enhancements to LTE-Advanced came in 3GPP Release 13 and 14, known as LTE-Advanced Pro.

3GPP Release 15 saw the delivery of the first 5G specifications, including new work as well as the maturing of the LTE-Advanced Pro specifications. Release 16 will be the second phase of 5G and was completed in December 2019 [5].

2.2 Vehicular Networks

ITS involve both traffic management and vehicle communication capabilities. Regarding communications, Vehicular Networks (VNs) are wireless networks that support transportation safety, transportation efficiency, and user services delivered to the vehicle. VNs are divided into two forms of communication, Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I). The combination of V2V and V2I is commonly referred to as Vehicle-to-Everything (V2X) [6]. The solution discussed in this thesis uses V2I communication, but also seeks to aid in V2V communication, as cars can for instance switch to V2V if the connection to the infrastructure is poor.

2.3 Dead spots

Places where mobile networks fail to provide successful communication is often referred to as *dead spots* in literature [7] [8]. These dead spots may occur due to, for example, mobile devices being out of range of any cellular towers or the wireless signal being blocked by some material in the surroundings of the device (e.g. in a tunnel or a building). Mobile and Wi-Fi networks are often arranged to primarily serve inhabited areas but not the roads in between, particularly, when these are minor [9]. Dead spots or poor mobile signal strength more frequently occurs in rural areas.

A different way to think of dead spots is also based on the area of application. As we will see in the next chapter, certain security related applications of vehicles have very strict latency requirements. The delay of the Radio Access Network (RAN) is much larger in for instance 2G networks than 4G networks [10] [11], and this delay alone may already exceed the requirements of such security related applications. That means, an area with only 2G access could be considered a dead spot even if the signal strength is satisfactory in that case.

It is also important to note the dynamic nature of dead spots. A breakdown in infrastructure because of a natural disaster could cause connectivity to change drastically. More commonly though, in the case of packet-switched networks, performance may vary because of network congestion. The real performance of the network will vary by network configuration, the radio environment of the location, the device in use, and number of active users [12].

2.4 Network measurements

Measuring network performance is not a new phenomenon and is a wide area of research. Several research projects are available with different purposes regarding this subject [13] [14]. Network performance measuring can be divided into two categories: active and passive measurements [15] [16]. Passive measurements are carried out by observing network traffic flows and do not add to network traffic. It is well suited for traffic engineering, showing flow dynamics and distribution. But obtaining end-to-end measurements with passive measurements is hard as presence of traffic between to measurement points cannot be ensured and matching measurements from different nodes is challenging.

Active measurements are on the other hand well suited for end-to-end measurements in a network and more feasible to implement in a client/server architecture (as we have in this thesis). As opposed to passive measurements, with active measurements probe packets are sent into the network and thus adds to the network traffic, potentially affecting the results.

Common metrics measured actively are:

- One-way delay
- Round-trip time
- Delay variation (commonly referred to as jitter)
- Packet re-ordering
- Packet loss
- Bandwidth

One-way delay measuring requires clock synchronization, as do most techniques for measuring delay variation. This is a complex problem that can be solved by using Global Positioning System (GPS) cards, radio clock receivers, or with Network Time Protocol (NTP) servers. Other metrics listed here generally do not imply constraints to the clocks of the end-systems.

For Transmission Control Protocol/Internet Protocol (TCP, TCP/IP) networks, TCP throughput is in [8] mentioned as a key bandwidth-related metric. TCP throughput is of interest to end users as TCP carries most of Internet traffic. Defining TCP throughput is however difficult, as several factors may influence it. This includes transfer size, number of competing TCP connections, socket buffer sizes, congestion along the reverse path, as well as variations in TCP implementations. Bulk transfer capacity defines a metric that represents the achievable throughput by a TCP connection, and is described by RFC 3148 [17].

Chapter 3 – Related work

There are multiple theses written at the Department of Information Security and Communication Technology (IIK), NTNU regarding communication in Vehicular Networks. Achieving V2V communication using peer-to-peer communication such as Wi-Fi Direct has been explored [18], among other topics. Solutions for measuring mobile network performance already exist, some also developed at IIK [9]. There are several mature, existing projects available where the mapping of network performance is the main objective. Identifying network performance constraints is an important subject in Vehicular Communications for the security of autonomous cars.

3.1 A Way to Measure and Analyze Cellular Network Connectivity on the Norwegian Road System

This paper [9] was written in 2017 at NTNU in collaboration with the Norwegian Public Road Administration (NPRA). The authors present a possible solution to collecting connectivity-data on the Norwegian road network on a big scale. Vehicles are equipped with Android devices that measure different network performance metrics, for instance round-trip time and signal strength. This data is then stored locally on the device together with GPS data of the location where measurements were performed. After completing measurements, the device can then be connected to a computer where the measurements can be uploaded and analyzed.

3.2 Message Forwarding between Vehicles in Dead Spots

As stated in [9], hybrid networks are a highly relevant research and development topic for academia, governments, automotive companies, and other stakeholders. They will combine mesh and cellular networks to utilize bandwidth efficiently. This master thesis by Jakobsen [18] (also written at IIK) uses such hybrid networks by showing how vehicles can switch to using vehicular ad-hoc networks (VANETs) in dead spots, to maintain the ability to forward important security messages between vehicles. The outlined solution uses Wi-Fi Direct in Android devices to accomplish this. It also discusses how a good power saving algorithm is essential to such a solution, as it is meant to be a full-time working system.

3.3 Vehicular Communication Systems: Enabling Technologies, Applications, and Future Outlook on Intelligent Transportation

In [19], several practical applications of VNs are presented along with potential latency requirements for each application. A small excerpt of the table presented in the article is given below. Note how safety-oriented applications like pre-crash sensing much lower latency requirements (50 ms) than user services like media downloads (500 ms).

Application name	**Communication**	**Latency**	**Other requirements**
Hazardous Location Warning	Ad hoc, infrastructure I2V, V2V	100 ms	High priority
Pre-Crash Sensing	AD hoc V2V	50 ms	Range: 50 m
Media Download	Infrastructure; cellular, other broadcast network	500 ms	Internet access Server availability

Table 3.1: VC-enabled applications and their characteristics (excerpt) [19]

In "5GCAR Scenarios, Use Cases, Requirements and KPIs" [20] several safety-oriented use-cases are also identified, with very strict latency and reliability requirements (less than 30 ms for all use-cases). The goal of the 5GCar project is to contribute to the specification of 5G to become a true enabler of V2X applications that are not realizable of today due to shortcomings of current communication networks.

3.4 NorNet project

NorNet is a platform for experimental networking research. It is built and operated by the Simula Research Laboratory and financed by the Research Council of Norway [21]. NorNet Edge consists of a large number of nodes geographically spread across Norway, able to perform complex measurement experiments. Simultaneous access to mobile broadband connections from multiple operators is provided. Real-time visualizations from NorNet Edge are publicly available and provide, among other metrics, round-trip time and packet loss rate across multiple ISPs. On their website, several publications are listed where the NorNet platform has been used for coverage testing while driving. Simula's Center for Resilient Networks and Applications (CRNA) publishes annual reports about the current state of Norwegian mobile networks based on measurements obtained using the NorNet infrastructure [22].

3.5 Crowdsourced projects

There are several publicly available crowdsourced projects for mapping mobile network coverage. The main issues with such crowdsourced solutions are the lack of repeatability and having no control or knowledge about the measurement equipment. On the other hand, they are generally end-user-friendly which means more people can contribute measurements and thus offers very large amounts of data.

Two examples of such projects are OpenSignal and Mobiperf. As opposed to drive-testing with expensive test equipment, users' smartphones are used for performing measurements. OpenSignal claims to receive data from more than 100 million devices worldwide, and capture over 3 billion measurements per day [23]. Network operators have criticized OpenSignal for being limited and non-scientific, not reflecting geographical coverage and only providing large amounts of data for downtown areas [24].

3.6 Summary of related work

ITS systems must work under tough latency constraints and ensure both security and user satisfaction. Studying network performance is a well-established area of research, and how measurements should be conducted is well-defined. Despite the increased interest in VNs and the ITS field in general, there seems to be little focus on the role of low connectivity areas in a future ITS world. This also applies to the dynamic nature of networks, and that cars need updated information in real-time. The authors were not able to find related work with a similar methodology in the same problem area.

Chapter 4 – Design and implementation

Our centralized solution for handling connectivity information for vehicles are based on a client-server architecture. The clients perform measurements using Android devices and transmit them to a server that processes the data. Cars on the road network with an internet connection can then request connectivity information for a given location from the server. While the intended user of the system is a car, practically any device with an Internet connection can utilize it. The general architecture is illustrated in the following figure.

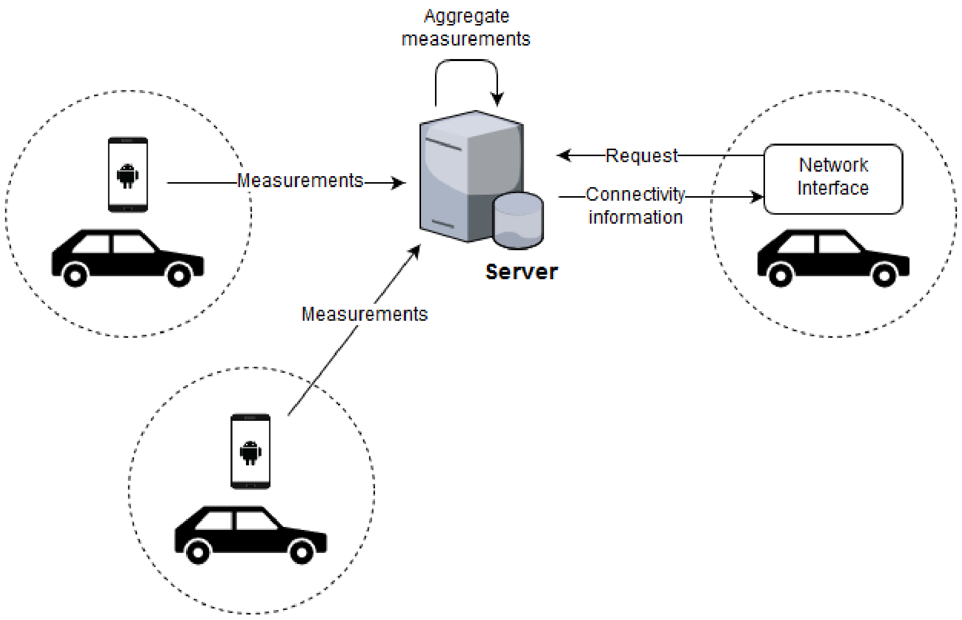

Figure 4.1: Architectural overview

Cars have their own complex computer systems, and most modern ones have their own SIM-cards. However, building the system around the computer systems of cars requires special insight, and could also limit the areas of use of the system. Android is a mature and widespread platform, that already in 2015 had over 1 billion devices using it [25]. With using Android devices, measurements can also be obtained by other mediums than cars, such as trains or bicycles.

Android equipment was handily available, and a similar app which could be used as a basis for development already existed from the work in [9]. Based on the authors existing experience with app development on both Android and iOS (iPhone) devices, iOS would not be well suited for this project due to APIs being more closed and that distributing an unsigned app only for research purposes is timelier and more difficult. However, it should be entirely possible to develop a similar app for the iOS platform if necessary.

Our centralized server approach provides more accurate dead spot information than the Vehicle-to-Vehicle methods we have investigated. Samples from several cars over a long period of time can be utilized, as opposed to only recent data from cars in proximity. Note how with our approach every vehicle potentially benefits from every measurement ever made for a certain location.

However, this architecture comes with the cost of running a server and a cost of data traffic flowing to the server using the available mobile network. As the server must handle large amounts of data from potentially many cars, this is not a negligible cost. A general problem of the client-server architecture is also having a single point of failure. Server reliability and availability are however not the focus of the thesis.

The centralized data may be of value to other interest groups. As mentioned in chapter 3, IIK has for instance already co-operated with the Norwegian Public Road Administration for measuring connectivity on the Norwegian road system. Telecom companies have access to the infrastructure to perform their own measurements for connectivity, but the data from our application could perhaps also be of interest if reaching a certain scale.

4.1 Client application

The task of the client application is to perform measurements of the performance of the mobile network and upload them to the server application. As mentioned, the chosen approach for the client is developing it for Android platform. In section 2.4, we introduced metrics that are commonly measured actively, such as round-trip time and packet loss. A subset of these metrics has been implemented in the application. As will be explained, some metrics required the development of a measurement server to which probe packets could be sent to. With this additional component, the architecture from a client perspective is presented in the following figure.

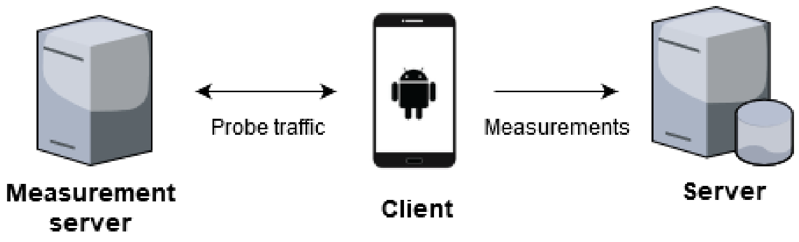

Figure 4.2: Architecture from client point of view

Parameters measured by the client application are:

- Signal strength
- Location
- Bandwidth
- Round-trip time (RTT)
- IP Packet Delay Variation (IPDV)
- Packet loss

Further details about parameters, their motivation and how they are implemented, and the use of the measurement server, are presented in the following sub-chapters. Lastly the interface of the application and how the application is used is shown.

4.1.1 Measuring signal strength

Data communication in a mobile network is done wirelessly, and the signal strength may therefore affect the achieved performance.

Android offers access to detailed information about the current mobile connection, such as network type (e.g. EDGE, HSPA and LTE) and signal strength. The Arbitrary Strength Unit (ASU) expresses signal strength as an integer. In GSM networks, the Received Signal Strength Indicator (RSSI) is used for calculating ASU. While in LTE networks, the Reference Signal Received Power (RSRP) is used. In the first case the ASU ranges from 0 to 31, and in the second from 0 to 97. In both cases a value of 99 indicates the signal strength could not be determined. This information can be found in the relevant Android source code [26] [27], where it is also indicated that this is based on the technical specifications of 3GPP. We have implemented measuring of both the GSM ASU level and the LTE ASU level. CDMA mobile telephony standards are not covered.

4.1.2 Measuring location

The measurements obtained for the other listed parameters must of course be paired with the current location of the device. Determining location is done using the Fused Location Provider API, which provides simple methods of obtaining an Android device's position given a specified quality [28]. In our case we request the highest accuracy possible and a frequent updating of position at a 1000 ms interval. GPS is the underlying technology used by this Android API in our case.

4.1.3 Measuring bandwidth

Bandwidth is notably more important when transferring larger amounts of data, such as when streaming media content. The available bandwidth limits the quality of the content the user receives. The Android API offers reading of total number of bytes received over all interfaces through "Android TrafficStats". Bytes received are measured at the network layer, including both TCP and User Datagram Protocol (UDP) usage [29].

Bulk transfer capacity, as described in section 2.4, is not possible to measure correctly using only this data, as we would need to know of retransmissions and TCP traffic only. What we can measure with the Android API though, is bytes received (as described above) over time. Facebook have developed a library, "Network Connection Class", for classifying user's internet connection and this is also what they use in their implementation [30]. They refer to this measurement as the "downstream bandwidth". To obtain a more accurate measurement, there needs to be some traffic to the smart phone during the sampling period. To achieve this, we issue downloads of a fixed-size file of 10 MBs from an external web server.

4.1.4 Probe traffic measurements

Measurements for RTT, IPDV and packet loss are obtained by sending probe packets to a separate measurement server. Note that it is purely implemented for performing client measurements. When the term *client-server* is used for our general system architecture, the measurement server is not the indicated server.

RTT limits how fast a response can be generated for a request and is therefore important for establishing connections. This makes round-trip time a key metric in VN applications. In related work in section 3.4, we saw some of the strict latency requirements of VNs.

Packet loss is an important metric in several applications. In safety-oriented VN applications, such as a hazardous location warning, packets would have to be re-transmitted in case of loss, drastically increasing any delays. Cisco state the following regarding packet loss in VoIP [31]: "The default G.729 codec requires packet loss far less than 1 percent to avoid audible errors. Ideally, there should be no packet loss for VoIP."

IPDV, more commonly referred to as jitter, is described in RFC 3393 [32]. The importance of jitter highly depends on the application. In a Skype call for instance, the tolerance for jitter is very low and jitter is easily noticeable. In media streaming, the tolerance is higher since more content can be buffered.

Because of the connection-oriented mechanisms of TCP, such as slow start and flow control, interpreting TCP results require more careful inspection. For simplicity we instead use the UDP protocol. While today's versions of Hypertext Transfer Protocol (HTTP) are built on top of the TCP protocol, the next version of HTTP, namely HTTP/3, will be built on top of UDP and not TCP [33]. The structure of our UDP probe packets is shown in the following figure.

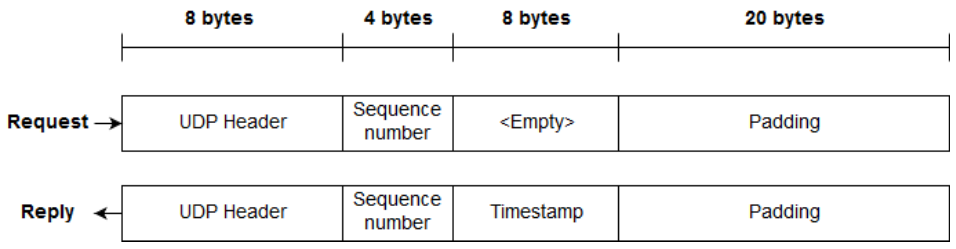

Figure 4.3: Probe packet structure

An “echo server” is a server that replies to a client with the same data as received. This is essentially what the measurement server does, except for the timestamp appended in reply packets. The timestamp indicates the time of which the server received the packet, and as will be explained shortly, is used for calculating IPDV. The “sequence number” is a number increased by one in the client each time a packet is sent.

Note that even though the reply packet from the server has additional data, we have used the same packet length for packets flowing in each direction. The packets are of the same fixed data length of 32 bytes (in addition to the UDP header of 8 bytes). 20 extra bytes were added in case more information would be required later. The sections illustrated as “<Empty>” or “Padding” are filled with 0s in the packet data.

Measuring RTT

For each sequence number, we store the corresponding send-time of the packet. When the client receives a reply packet, RTT can be calculated by using the sequence number to retrieve the send-time again and subtracting it from current time.

Measuring packet loss

Since the sequence number of all sent packets are stored, packet loss is easily determined by identifying packets that have received no reply.

Measuring IPDV

RFC 3393 [32] defines packet delay variation as "the difference between the one-way-delay of packets within a stream going from measurement point MP1 to measurement point MP2".

This implies that knowledge of the following is required for calculation:

- Transmit time of first packet in a pair
- Receive time of the first packet in a pair
- Transmit time of second packet in a pair
- Receive time of second packet in a pair

It also implies that the clocks of the two measurement points are synchronized. However, if packets are transmitted periodically, two of the needed parameters are pre-determined and clock synchronization is not necessary. With packets being transmitted periodically, only the inter-arrival time of received packets are measured. If the packets are sent periodically, then the difference between the *receiving times* for two consecutive packets should be equal to the *periodic send interval* if there is no packet delay variation [34].

An accuracy flaw of measuring packet delay variation in this way occurs when a packet is lost. The inter-arrival time between two packets before and after the dropped packet will be large and could skew the data. We solved this in our implementation by only looking at consecutive packets, i.e. not using lost packets when calculating IPDV.

As mentioned, when the measurement server receives a packet from a client, a timestamp for when the packet was received is appended to the reply packet. For two consecutive (using the sequence number) packets p_1 and p_2 with receiving timestamps t_1 and t_2, and a periodic send interval p the packet delay variation then is:

$$\mathrm{ipdv} = \mathrm{t}_2 - \mathrm{t}_1 - \mathrm{p}$$

An example is illustrated in the following figure.

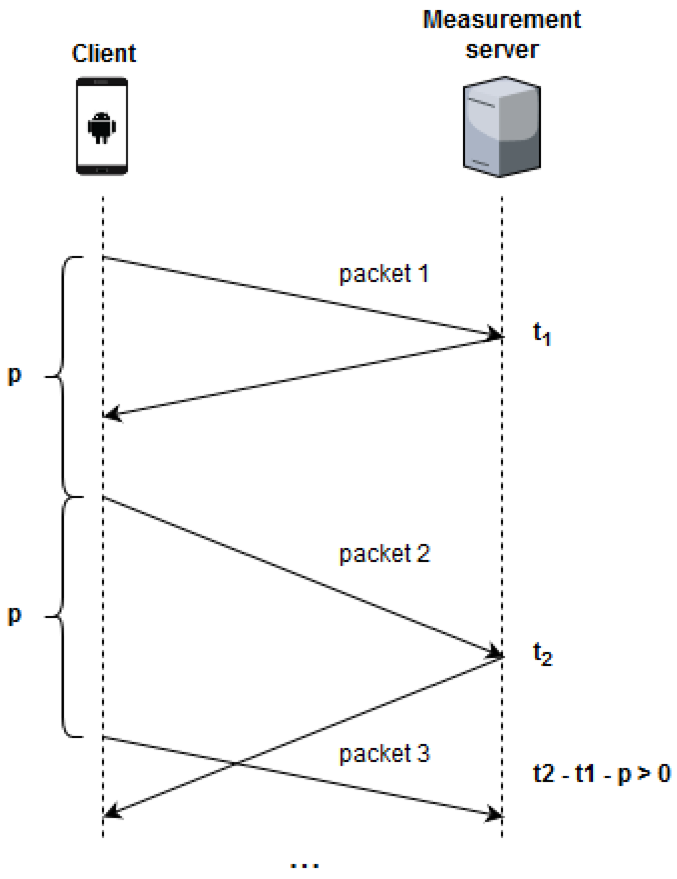

Figure 4.4: IPDV example

In this example, the second packet of the pair takes longer to arrive at its destination. The result is then a positive IPDV, since:

$t_2 - t_1 - p > 0$

A negative value of IPDV would occur if packet 2 had a shorter transmit time than packet 1. Note that this is the packet delay variation for traffic flowing from the Android device to the measurement server, and not the other way around. For measuring the IPDV of packets flowing from measurement server to the Android device, the measurement server would be required to periodically send packets to the Android device instead.

Periodic measurements

To able to calculate IPDV without clock synchronization, we use periodic sampling, i.e. packets are sent from the Android device to the measurement periodically. While periodic sampling provides a simpler way to calculate IPDV, it also comes with disadvantages. This is also discussed more in depth in RFC 2330 [35]. Periodic sampling is more susceptible to manipulation, and repeated periodic injection of traffic could drive the network into a state of synchronization. If the measured metric exhibits periodic behavior, there is a possibility that the sampling only observes the part of the periodic behavior the periods agrees with. Given the very small traffic volume that our application generates (approximately 160 byte/s), we assume these issues should not have a large impact on the measurements.

Different sampling rates were tested to see if the round-trip time was affected. Measurements for RTT with different rates were made on the same network, within the same area and within a short timeframe and are presented in the following table. A lower average round-trip time was achieved when using a higher sampling rate (to a certain extent).

Interval	**Average round-trip time**	**Sample count**
10 ms	74 ms	> 1000
25 ms	59 ms	500
250 ms	61 ms	600
500 ms	92 ms	400
1000 ms	103 ms	400
2000 ms	94 ms	400

Table 4.1: Average round-trip time for different periodic send intervals

There appears to be a sudden increase in round-trip time when sampling at a lower rate than about every 250 ms. The reason behind this has not been investigated thoroughly, but a possible explanation could be how intermediate routers on the path between client and server treat different traffic patterns. NorNet suggests this idea on its web page for showing NorNet Edge details: “The mode of a connection again depends on the traffic pattern. Hence, sending more traffic can result in a lower RTT.” [36]. As explained later, since we aggregate measurements based on proximity, the sample rate also determines how many measurements are obtained per aggregated area.

4.1.5 Using the application

The app provides a simple interface with 2 buttons, a “Start” and a “Stop” button, as shown in following figure. Sampling starts immediately when the “Start” button is pressed, and the user interface is continuously updated with location information and number of samples that are currently stored in memory. Once the “Stop” button is pressed, the sending of new probe packets stops immediately, but the app will continue to receive probe packets for 5 seconds before the measurement process is considered completed. This is to allow extra time for recently sent packets to be replied to, so the reply packet is not considered lost. Once the measurement process has completed, all measurements are reliably uploaded to the server using TCP.

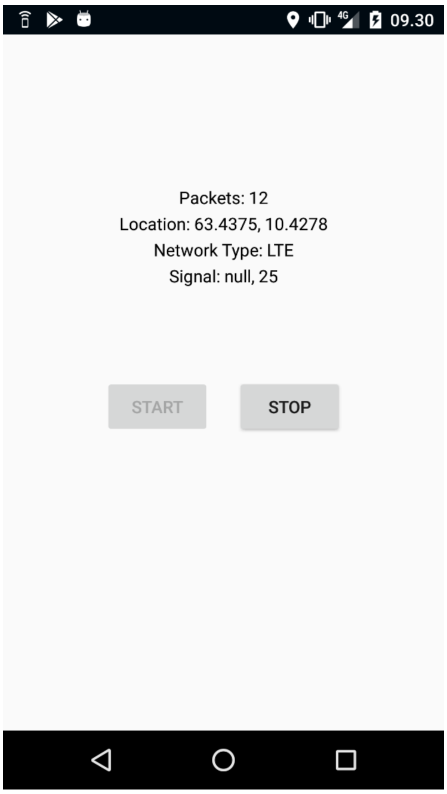
Packets: 12
Location: 63.4375, 10.4278
Network Type: LTE
Signal: null, 25

Figure 4.5: Android app interface

Uploading all measurements when the process is completed rather than continuously uploading them partly conflicts with the real-time aspect of the system. That is, cars would always have more recent information with a continuous upload. This was done to ensure that the uploading of samples did not affect measurements. As we will see in chapter 5, we found that using bandwidth affected for instance round-trip time measurements. Possible solutions to this problem could be either using a separate connection for uploading samples or making short pauses in measuring to upload samples. Sending the samples through the TCP socket usually took under 1 second, depending on the amount.

4.2 Server application

The server application has mainly three responsibilities:

1. Receiving measurements from clients
2. Storing measurements
3. Providing connectivity information

To receive measurements, a TCP socket listens to incoming data from clients. Measurements are stored using a database management system (DBMS). A connection and an interface between our application and the DBMS are required for storing and retrieving data. To provide connectivity information, the server application also acts as a web server, exposing a Representational State Transfer (REST) API. This makes the server able to respond to HTTP requests, for instance requesting a list of measurements or a live map. A simple overview of the server functionality is shown below.

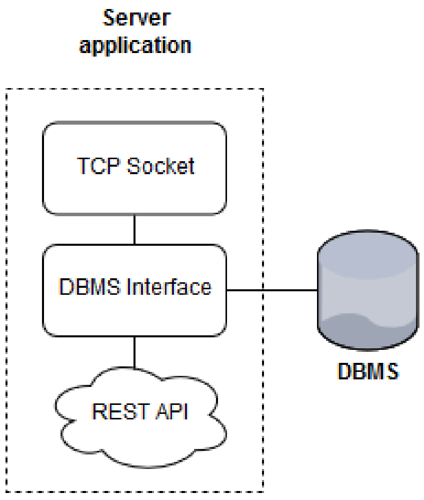

Figure 4.6: Server application overview

In the following sub-chapters, we will have a closer look at the different parts of the server application. First, how measurements are received and stored is discussed. We will then introduce how measurements can be aggregated and what types of aggregations are performed. Furthermore, we have developed a function for quantifying network performance based on aggregations. Using this function, dead spots in a mobile network can be identified. An algorithm for identifying these dead spots is then described in detail. Lastly, we show how the application is capable of presenting connectivity information in multiple ways, with both structured data formats and more human-readable formats like graphs and maps.

4.2.1 Receiving and storing measurements

A TCP socket listens for new connections from clients. For each new client, a new thread is spawned to support concurrent receiving of measurements from multiple clients. Each set of measurements received from a client is assigned a separate "trip ID", making it easier to separate data from different driving trips or sources. Note how the server application can potentially receive sample data from multiple types of clients. It is not dependent on the described Android application, only that received data is in expected format.

Measurements are aggregated based on location proximity – samples within a radius of 50 meters of each other are considered to be part of the same *area*. Measurements and areas are the two entities in the database. The Entity-Relationship model (ER model) of the database is shown in the figure below.

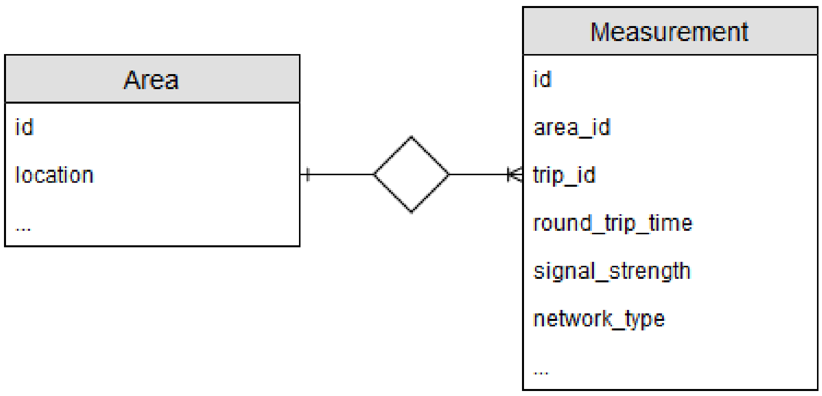

Figure 4.7: Database ER-model

Area radius and sample rate of the client application determines number of measurements per area on average. With an area radius of 50 meters, a sampling rate of 250 ms and a vehicle moving at 60 km/h, one should be able to obtain about 24 measurements for an area. Note that this only is true when the area is not yet created or when driving through the center of the area. The current balance between area size and sampling frequency should be sufficient for doing calculations without results being severely prone to outliers in the data. An area radius of 50 meters gives good locality in measurements, being able to state where the start and end point of a dead spot could be somewhat accurately. A more frequent sampling rate would give more data, but the variations in the data might be small between samples. The increased data volume could also affect performance.

For each measurement, the application checks whether an area for the measurement location exists, and if not, creates a new area. That is, the application needs to check whether an area within 50 meters of the measurement's location already exists. With our measurements we had several thousand areas, so it is important that this is done efficiently. This is solved using spatial indexes, a property of an extension library to our DBMS, described shortly in section 4.3. Scalability and performance of the server application are addressed in section 5.6.

4.2.2 Aggregating measurements

With multiple measurements for each area, aggregations can be performed. Depending on the type of request, all measurements or only a subset of measurements in an area can be used for performing aggregations. For instance, a request may specify that only measurements from a certain trip (trip ID) or a certain network type (e.g. LTE) is of interest. More information about different types of requests is provided in section 4.2.6. For the selected measurements, the following aggregations are calculated:

- Average round-trip time
- Median round-trip time
- Round-trip time quality*
- Jitter ratio*
- Average signal strength
- Packet loss ratio
- Most common network type (e.g. EDGE, HSPA, ...)

Round-trip time quality

As mentioned in section 2.3, in 2G/3G data networks, usually the RAN latency dominates the overall end-to-end delay. The real performance of every network will vary by provider, their network configuration, active users within a given cell, the radio environment in the location etc. While there are no guarantees for latency in real environments, a simple strategy is to assume a higher bound for packet latency for every generation [12]. The following table shows latency for different network generations.

Generation	**Latency**
2G	300-1000 ms
3G	100-500 ms
4G	$<$ 100 ms

Table 4.2: Latency for different network generations [12]

Round-trip time quality simply means that we treat a measured round-trip time differently depending on the network type. We assign a value between 0 to 1, where 1 is the best quality. We use information from the table above, and the distributions of our actual RTT measurements (presented in chapter 5) to determine the value. The assignment of a value between 0 and 1 is done quite coarsely, as the variations in RTT can be large, especially for 2G networks. The correctness of this and what exact values should be used is arguable, but the point is that "how good" a measured RTT value is may depend on the network type.

Jitter ratio

As described in the client application section, we measure the IPDV of consecutive packets. This value can be both negative or positive. Jitter can be represented relative to the mean period [37]:

$$\text{jitter} = \frac{\text{mean jitter}}{\text{mean period}}$$

In our case this means: how much is the mean jitter compared to the periodic send interval of 250 ms? Since IPDV can be both positive and negative, the absolute value of IPDV is used.

$$\text{mean jitter} = \frac{\sum \text{abs(IPDV)}}{\text{n}}$$

This is then divided by the periodic send interval of 250 ms.

Aggregations are calculated when relevant data is requested, rather than as an ongoing process and stored in the database. This means that all aggregations described above must be calculated for each area that is contained within a reply. Calculating aggregations on request comes with a flexibility advantage. New types of aggregations can easily be added without any structural changes to the database.

If we wanted to store calculated aggregations in the database, we would have to store many aggregations for different input parameters. Storing the aggregations should give increased performance, but so far performance has not been observed to be an issue regarding this. A simpler approach could be using a cache, either persistent storage or in memory. The cache would store aggregations for given input parameters for a limited period of time.

4.2.3 Network performance model

Based on the different aggregations discussed in the previous section, we have defined a function which uses these aggregations for generating a value between 0 and 1. This value indicates the general network performance of an area, where a greater value means greater performance. The color of areas in any map figures to be shown are based on the output value of this function, ranging from red (value 0) to green (value 1).

Given a packet loss ratio PLR, a jitter ratio JTR, a round-trip time quality RTTQ and an average signal strength of AVG_SIG, the defined function is as follows:

$$p = (1 - \text{PLR})(1 - \text{JTR}) \cdot \text{RTTQ} \cdot \text{AVG\_SIG}$$

Since signal strength comes in the form of a GSM or LTE ASU level value (see section 4.1.1), each value is normalized into a number between 0 and 1 and the average of these values is then taken. This function models network performance in a simple manner, with each metric being able to affect the output value. It is rather meant as a tentative starting point of a proper model than a good, polished model of network performance. It is for instance theoretically possible to score a performance value of 0.5 with 50% packet loss with this function, where in practice most applications would be unusable under such conditions. Parametrizing, evaluating and testing such a function is rather encouraged as future work, and also a bit further addressed in chapter 7.

4.2.4 Defining dead spots

Being able to quantify the network performance of each aggregated area, a threshold value for a required performance can be specified. When the performance value falls under this threshold value for an area, that area can then be considered a dead spot. An example of this is described below.

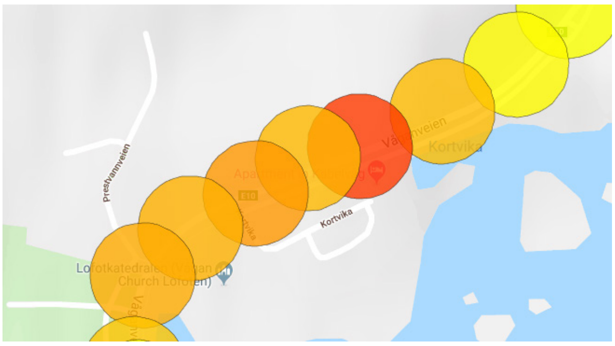

Figure 4.8: Performance threshold example

The red area in the figure has the worst performance of all areas shown with a value of 0.1. The surrounding yellow areas all have a value of at least 0.3. If a threshold value of 0.2 is specified, this red area would then be considered a dead spot, and the rest of the areas would be considered to have sufficient performance.

The other way a dead spot may occur is if there is a gap between areas. This happens when the client is unable to identify its current location, for instance in tunnels where GPS services could be unavailable. This is a weakness of the client sampling application, as the mobile network could still be available even though GPS services are not. As we do not know if only GPS services, or both GPS services and the mobile network that are unavailable, it is treated as a dead spot. Positioning issues and potential solutions are also addressed in section 5.1 and chapter 7. An example of a gap dead spot is shown in the following figure.

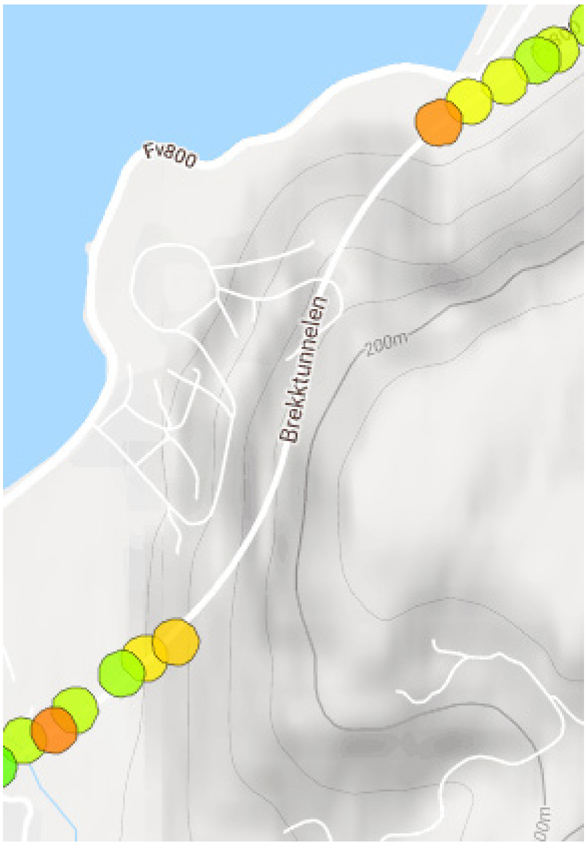

Figure 4.9: Gap dead spot, a GPS (and possibly mobile network) dead spot in a tunnel

4.2.5 Algorithm for finding dead spots

To provide vehicles on the road with up to date information about dead spots, an algorithm to identify them based on our network data has been developed. The input to the algorithm is a route from a specified origin point (latitude, longitude) to a specified destination point and a minimum desired performance value. The route must be described by a *polyline*, which is a continuous line composed of one or more segments.

To obtain such a polyline route, we utilize the Google Directions API in our implementation. This API is a service that calculates directions between locations. It can be used for directions search for several modes of transportation, including driving, walking and cycling. Based on origins, destinations and waypoints it returns multi-part directions using a series of waypoints [38].

Figure 4.8 below is used to describe the algorithm. A route polyline is illustrated in (1). First, the polyline is segmented so that no line segment is longer than a maximum length as shown in (2). The smaller the segments, the more accurate the algorithm is, but the costlier the computations. An area with a performance value less than the specified minimum performance value is considered a dead spot. In (3) in Figure y we see corresponding hypothetical sample data for the polyline route, with an indicated dead spot area.

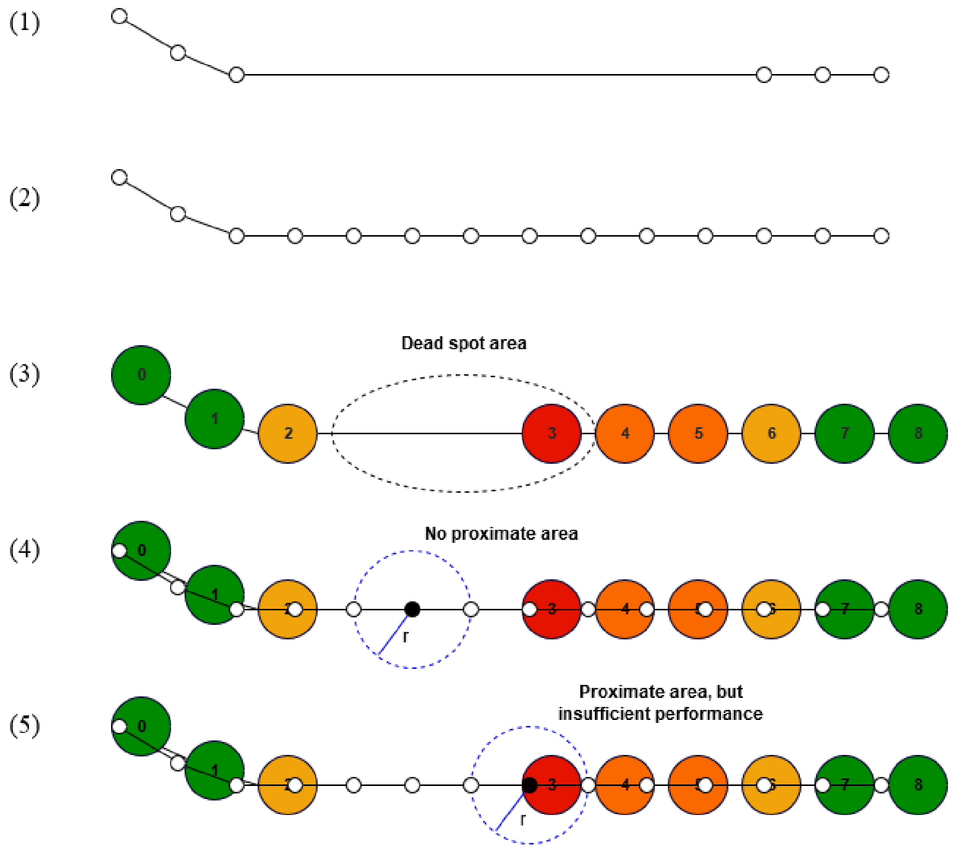

Figure 4.10: Different components in dead spot finding algorithm

A *proximate area* is an area within the distance *r* from a vertex in (2). For each vertex on the polyline in (2) we check if there are no proximate areas, or if the performance of proximate areas is insufficient. The first is illustrated in (4), and the latter in (5). As seen in (3), the last area before the dead spot starts will be area 2, and the first one found with sufficient performance after the dead spot began will be area 4. The dead spot is then said to be between area 2 and area 4. The distance *r* must be chosen with the line segment length chosen for (2) in mind. Testing and outputs of the algorithm is presented in chapter 5.

4.2.6 Providing connectivity information

The application provides connectivity information in a variety of ways. As mentioned, this is enabled by the application acting as a web server and exposing a REST API. The following types of requests are supported:

1. Show measurement data
2. Show aggregated area data
3. List all dead spots on a route given a start and destination point
4. Show statistical correlations in aggregated data
5. List all trips and their most common network type
6. Show live map of measurements
7. A variety of (live) graphs depending on input parameters

Note that every response is based on live data, including maps and graphs. For instance, if viewing a map in a web browser and new measurements are received from a client, the new measurements are reflected in the map when refreshing the page. This conforms to the real-time requirement of the application.

With this wide array of functionality, users of the application have large flexibility in how they choose to interpret information. 6 and 7 are mainly for human readability, while the output for other types of requests can be used as input to other information systems. For instance, listing of dead spots can be directly used by a video streaming application to schedule its bandwidth usage. The structured data format generally used is JavaScript Object Notation (JSON).

Maps and graphs are very helpful to studying the sample data, for instance looking at the statistical distributions for different metrics. As we will see, we use these tools extensively in chapter 5 for analyzing results. More details for each request type and some examples are provided on the following 3 pages.

Show measurement data

Outputs raw measurement data in JSON format for given parameters.

Supported parameters: Area ID, network type, trip ID(s).

Example: http://address:port/measurements?tripId=59,60&networkType=LTE

Show aggregated area data

Performs aggregations (average signal strength, round-trip time quality etc., see section 4.2.2) for each relevant area and outputs result in JSON format.

Supported parameters: Area ID, network type, trip ID(s).

Example: http://address:port/areas?tripId=59,60&networkType=LTE

List all dead spots given a start and destination point

Given an origin point (latitude, longitude) and a destination point (latitude, longitude), outputs all dead spots on a route between the two points and the dead spots respective lengths in meters.

Example:
http://address:port/dead-spots?origin=63.3975,10.5480&destination=63.3774,10.
6019

Show statistical correlations in aggregated data

Calculates and outputs correlations between different metrics for given parameters.

Supported parameters: Network type, trip ID(s).

Example: http://address:port/correlations?tripId=43,45&networkType=EDGE

List all trips and their most common network type

Outputs a list of all trips, their IDs and the most common network type for each trip.

Example request: http://address:port/trips

Show map of measurements

Shows a geographical map of areas for the given parameters. The color of an area is based on the calculated performance for the area using the function described in section 4.2.3. Uses Google JavaScript Maps API for geographical data. Example output shown below.

Supported parameters: Area ID, network type, trip ID(s).

Example: http://address:port/map?tripId=16

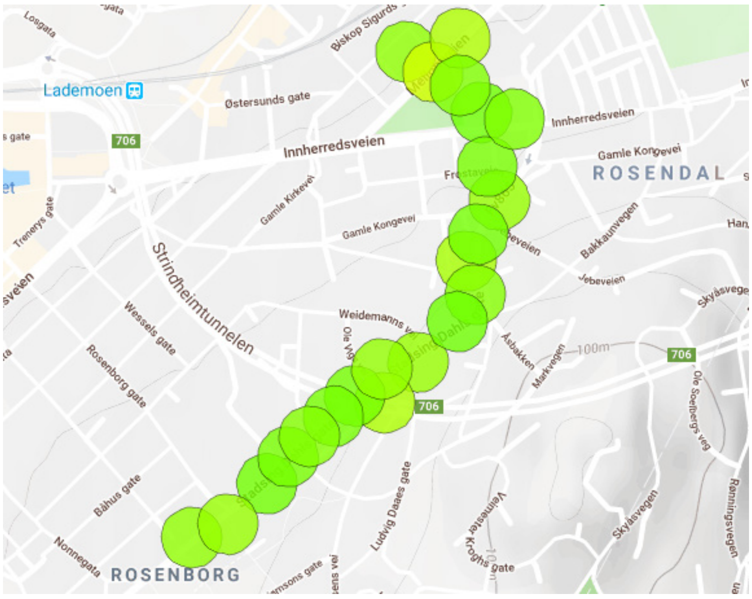

Figure 4.11: Example output for a map of a specified trip id

Graphs

A variety of different graphs of the measurement data or aggregations of them based on the input parameters can be displayed. This includes statistical distributions of metrics, showing a metric Y as a function of metric X and variations in a metric over time for a specified trip. When calculating a metric Y as a function of metric X, the average or the median value for metric Y is calculated. Supported metrics are round-trip time, signal strength (GSM ASU level or LTE ASU level) and IPDV. Example output of a graph showing median round-trip time as a function of LTE ASU level for a specified set of trips in LTE networks is shown below. Statistical analysis is further explored in chapter 5.

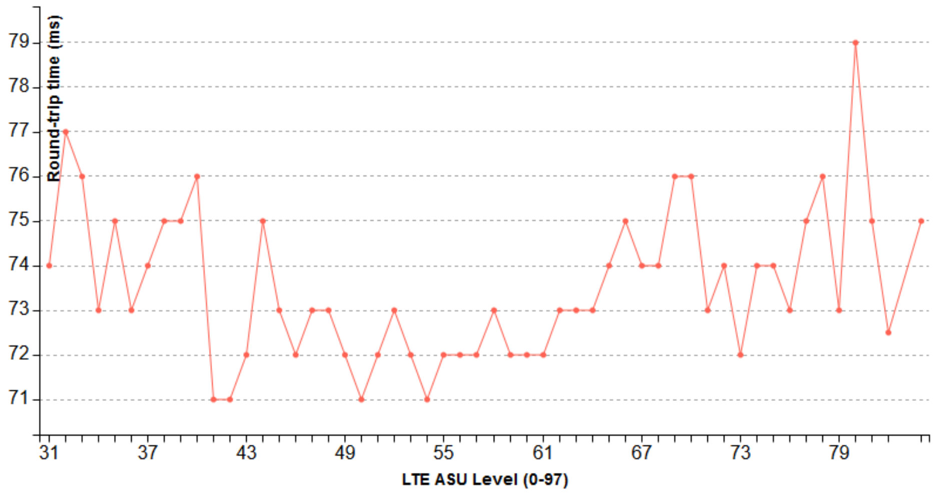

Figure 4.12: Example output of a graph

Example:

http://address:port/chart?tripId=24,27,28&metric=roundTripTime&networkType=LTE&type=distribution

4.3 Tools and libraries

Except for the HTML and JavaScript usage in the server application, Kotlin has been the programming language of choice for both client and server. Kotlin is a new (first appearance 2011) statically typed programming language fully interoperable with Java. Since the release of Android Studio 3.0, Kotlin has also been fully supported by Google for use with their Android mobile phone operating system [39]. In comparison to Java, Kotlin offers null safety, more concise syntax and less boilerplate code. However, Java popularity and its community is still much larger than of Kotlin.

While the Android application only uses the Android framework, the server application uses a variety of different libraries. Multiple programming languages and script languages were used to create its different components.

The underlying database management system (DBMS) chosen is PostgreSQL (Postgres). Postgres has many extensions, one of them being PostGIS. PostGIS is a spatial database extender for Postgres adding support for geographic objects. This allows location queries to be run in Structured Query Language (SQL), and tree-structured indexes (which are common in any DBMS) can be utilized for location queries improving the performance significantly. For instance, with over 3000 areas in the database, finding if a given pair of coordinates (latitude, longitude) belongs to an existing area is done within the range of 10 ms. PostGIS as an extension and its good documentation was the main motivating factor for choosing Postgres as a DBMS for the project.

Exposed (see https://github.com/JetBrains/Exposed), a lightweight SQL framework written over the JDBC (Java Database Connectivity) driver acts as the interface between the application and the DBMS. With its small requirements for configuration, *Exposed* is quick to set up. In addition to taking advantage of the library functions of this library some traditional SQL has also been used.

Opening and creating new connections to a database can be expensive, especially if the database is hosted on a different computer than which the application is running on. Connection pooling allows database connections to be reused by multiple clients, improving performance. In the case of our project the database was actually hosted on the same machine as the running application, but to ensure scalability and in case the database would later be moved to a remote server, connection pooling was implemented. For this, the library HikariCP (see https://github.com/brettwooldridge/HikariCP) was used.

To enable the application as a web server, the library Ktor (see https://github.com/ktorio/ktor) was used. Ktor is a framework for creating web applications in Kotlin programming language and runs an embedded web server on the host machine.

As mentioned in section 4.2.6, most response data the application generates to received HTTP requests is in JSON format. But there are a few exceptions, namely the maps and graphs. Responses to these requests are web pages, containing both HTML and JavaScript. Maps use the Google JavaScript Maps API, and graphs functionality is provided by the JavaScript library C3.js. The maps web page also fetches JSON data from the REST API itself, and for this jQuery, a JavaScript library for simplifying syntax, was used.

Chapter 5 – Results and discussion

In this chapter we first address issues faced during the development and sampling process, since they influence what results we are able to present. We then have an in-depth look at how our measurements are distributed across different metrics. We proceed to calculate and discuss relationships between different metrics. Lastly, we address the scalability of the application and provide examples that show that the implemented dead spot identification algorithm can handle a variety of cases.

When using the term “correlation” or “correlation coefficient”, the Pearson correlation coefficient (PCC) is implied. Its value r lies between -1 and +1 and measures the strength of the relationship between the relative movements between variables.

Note how all figures and calculations we present here are based on the live database, as opposed to for instance outputting a subset of data to Comma-separated values (CSV) format and using additional software to interpret it. We simply access the REST API through the web browser and provide the desired parameters to the application. The application uses our parameters to extract the relevant information from the DBMS and presents it.

5.1 Issues

Challenges faced during the development and sampling are addressed here. Some of the issues presented have rendered small parts of our samples difficult to use, such as GPS positioning issues. We have also not been able to take advantage of the bandwidth metric.

5.1.1 Bandwidth measuring issues

In the process of developing and testing the client application, it was found that bandwidth measurements had a significant impact on the other metrics while being performed. While downloading test files, the round-trip times would more than double in most cases. An attempt to solve this was made by pausing measurements for other metrics while bandwidth measurements were being made. This however resulted in very few measurements for some areas, as these bandwidth measurements take several seconds to perform. Decreasing the file size would help with this issue since the bandwidth measurement time would then be shorter, but this would make the bandwidth measurements more inaccurate.

Another issue with measuring bandwidth is cost-efficiency. The subscription for the test equipment SIM-card quickly ran out of allocated mobile data traffic (this is the model most operators use today). Because of these issues, bandwidth measurements were unfortunately abandoned in the development and testing process. Ideally one would want a SIM-card with no data volume limit and use a separate connection for sampling bandwidth data to not interfere with other metrics.

5.1.2 Software library issues

The SQL framework library, *Exposed*, was found to have a few bugs regarding batch insertion to the database. The developers of the library were notified and luckily the issues were fixed in less than 24 hours (see https://github.com/JetBrains/Exposed/issues/388). Because of the quick reaction, this did not halt the development process, but it suggests perhaps a more stable SQL framework should have been utilized. The framework also proved to be a bit limited for taking advantage of SQL functions of extensions like PostGIS. In these cases, raw SQL was used instead of the library functions.

5.1.3 GPS positioning issues

Any measurement taken with very poor positioning accuracy or without any positioning information at all is difficult to use. One could possibly use positioning data from earlier samples with a better positioning accuracy to accommodate for this, but it would require a more sophisticated implementation. On the road network, GPS is unavailable in most tunnels. Gaps in the measurement data happens because of this, as also shown in chapter 4.2.4.

For one Android device the positioning accuracy was occasionally very poor. This could be an issue with the device itself or with the used Fused Location API, but more testing would be required to identify the actual cause. Data from trips with such poor accuracy are filtered out when further studying the data. An example of such a trip is shown below. Addressing positioning issues is further discussed in future work.

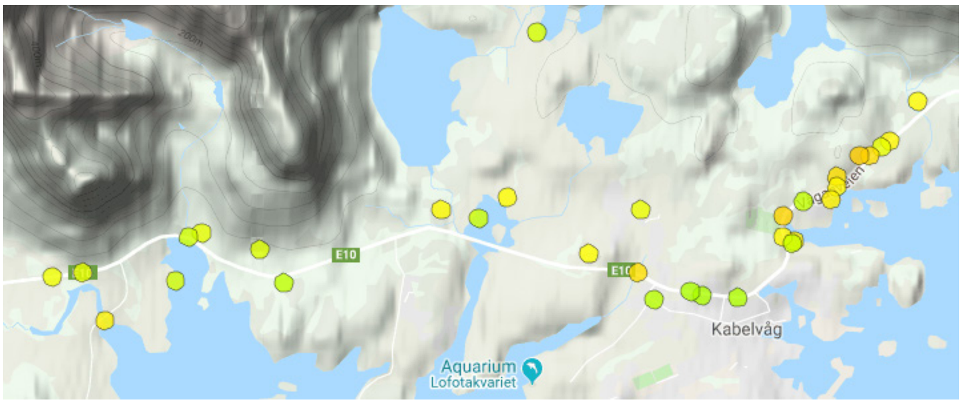

Figure 5.1: Example of trip with poor positioning accuracy

5.2 Collected data

Throughout this thesis we have collected more than 130 000 samples. More than half of these samples are from vehicle trips in Lofoten. This area is rather rural with very mountainous terrain. Samples from the Trondheim city area have also been collected, where most of the trips where done by walking or by bicycle. A vehicle trip from Trondheim to rural areas north of Lensvik has also been sampled. In addition to samples made during this thesis, datasets used in [9] from the Jonsvatnet area were imported to aid in developing performance models and the dead spot identification algorithm. These datasets do however not contain data for all metrics since a different application was used for sampling.

Figure 5.2 shows sampled data in the Lofoten area. The colored line consists of circle areas with a radius of 50 meters, where each of these areas on average have 90 measurements. Over 80 000 samples are represented in this map. The map is produced by the discussed REST API and this data is returned in less than 1 second - this also includes performing all aggregations discussed in chapter 4.2.2 for each area in the dataset.

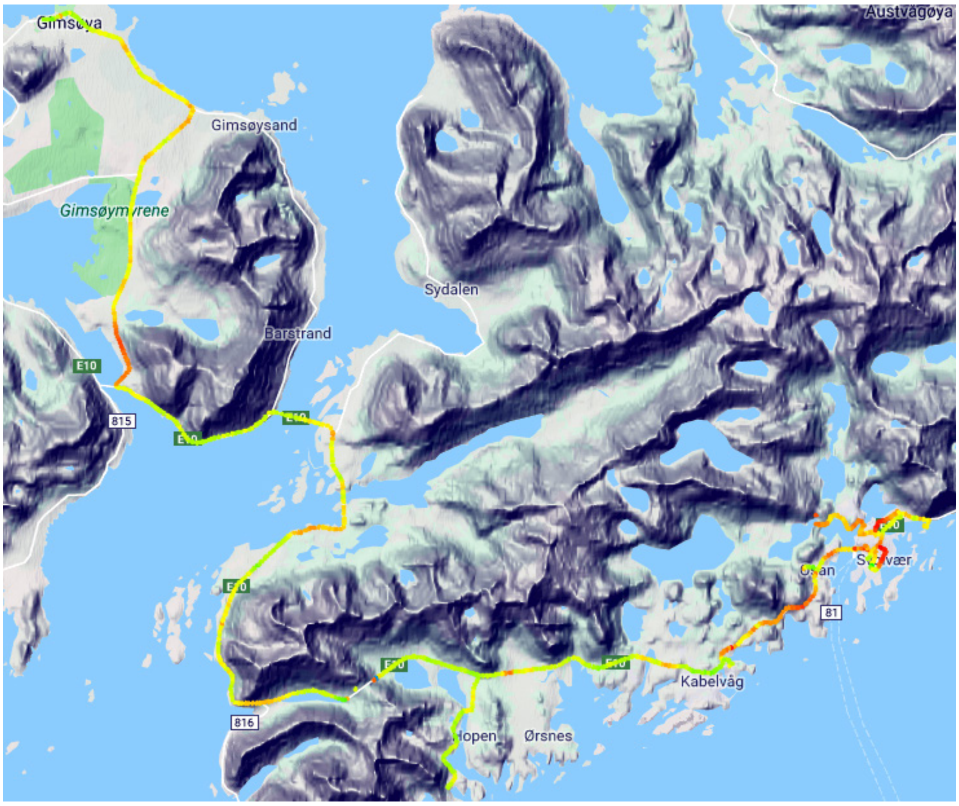

Figure 5.2: Map of sample data from Lofoten area

5.2.1 Sample distribution between network types

The distribution of measurements by network type are shown in the following table.

Network type	**% of total**
LTE	72 %
EDGE	19 %
HSPA	8 %
HSPA+	< 1 %
HSUPA	< 1 %

Table 5.1: Sample distribution between network types

The majority of measurements are from LTE networks. In Lofoten, no other network type was observed. Some non-LTE network type measurements in Trondheim originate from the imported datasets from the work in [9], sampled in 2017. In these datasets all measurements were in fact from EDGE, HSUPA and HSPA+ networks, even the samples from non-rural areas such as university campus. 4G coverage in Trondheim appears to have improved since then, as we throughout this thesis did not observe any central area without 4G coverage.

Expanding access to LTE has been the focus of the global mobile industry for years, and OpenSignal with over 58 billion measurements from all over the world claims 4G availability is over 90 % in several countries [40]. Note that these numbers only mean users have access to 4G networks 90% of the time, and do not show what is the geographical coverage.

In [41], the authors conclude that it may be economically unfeasible for network operators to deploy LTE networks in rural areas in Spain and that the results might be applicable to other big European countries with similar characteristics. These rural areas are where intuitively one would find dead spots, so measurements from older generation network types could be more relevant to consider regarding dead spots.

On Android devices, the user can select a preferred network type, being 2G, 3G or 4G. Using this feature, measurements for 2G and 3G network types were obtained in areas with 4G coverage. This was done to get more measurements from non-LTE network types. Telenor, the largest telecom company in Norway, has however started shutting down 3G networks [42], so this method may not work in the future. Shutting down 2G networks is also a long-term goal, but this can be more complicated. 2G networks are in many places a rigid part of infrastructure, being part of for instance card terminals or car systems where emergency services are automatically contacted in case of collision [43].

5.3 Metric distributions

Metric distribution reveals coverage for different values of a metric. We mainly address round-trip time and signal strength here. IPDV (jitter) and packet loss are more interesting to see in relation to other metrics and will be further discussed in the next section, section 5.4. We will show the mean (average) and median values together with the distributions. Median values are of interest since they are not as affected by extreme values as the mean.

5.3.1 Round-trip time

In statistics, the term *right-skewed* (positive skew) is often used for distributions having a lower median value than mean value. These distributions have a long tail on the positive direction on the number line. RTT values can be expected to be right-skewed, because RTT is lower bound by the underlying network technology but have no upper bound. High values occasionally occur, but most values are concentrated around a smaller range.

We found the high values of RTTs to occur in bursts. An example of this in an LTE network is shown in the figure below. The represented data is a selection of 500 consecutive packets from one specific trip, with RTT on the y-axis, and the packet index (the i-th packet sent on that trip) on the x-axis. The high RTTs in this example occur over 16 around seconds.

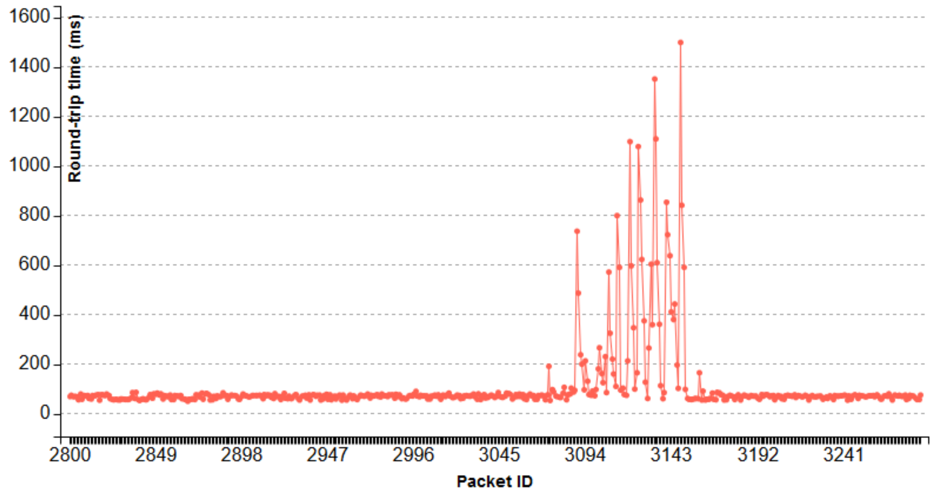

Figure 5.3: High RTT in short timeframe

The corresponding area data of the figure above is shown below. The bursts in RTTs result in our performance function outputting a lower value for the corresponding areas, as can be seen by the red areas in the figure.

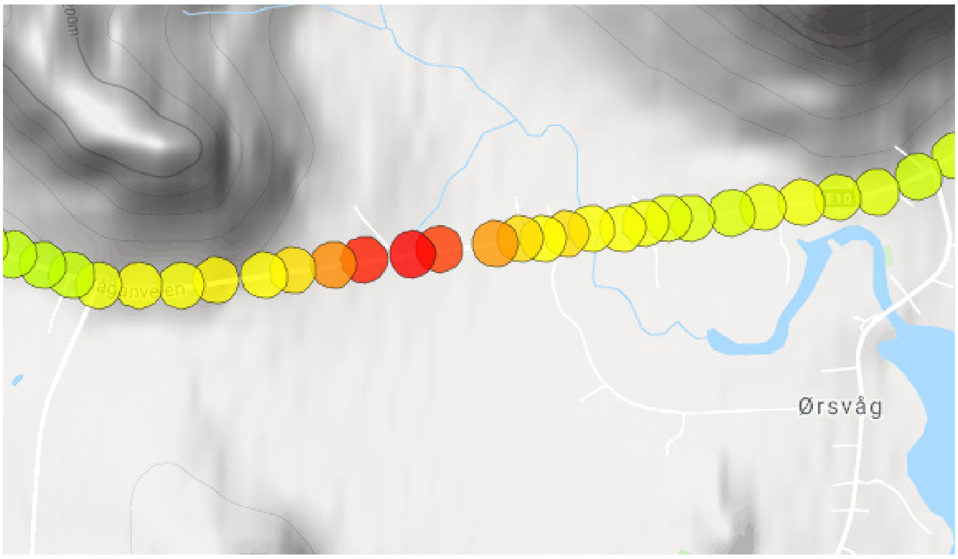

Figure 5.4: High RTTs reflected in performance function

In the following three figures, selections of the distributions of RTTs are shown for EDGE, HSPA and LTE networks. That is, occurred values are on the x-axis and number of occurrences for each value are on the y-axis. The mean and median values are indicated in each figure. The selections are made around where the mass of the distribution is concentrated, but mean and median values are calculated using all values. In addition to what is shown there are values with very few occurrences (generally one occurrence) on the far right of the distribution, which is what makes the mean higher than the median.

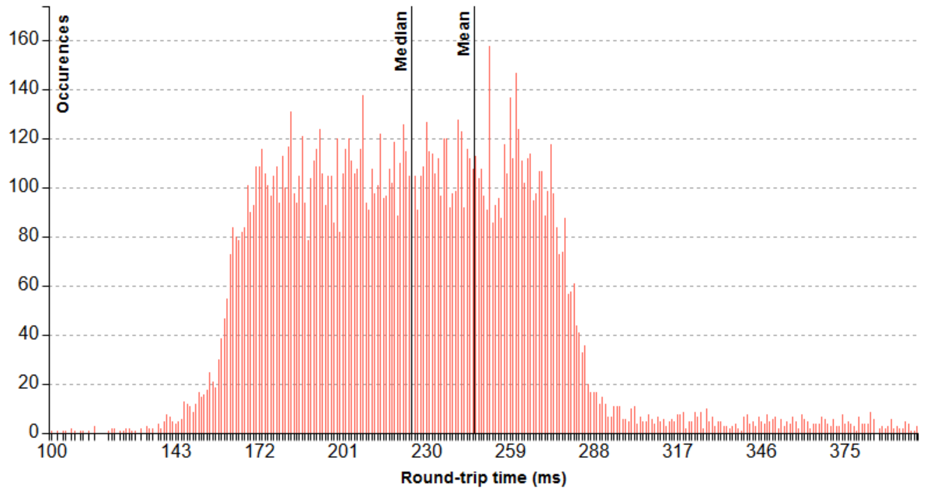

Figure 5.5: RTT distribution for EDGE networks

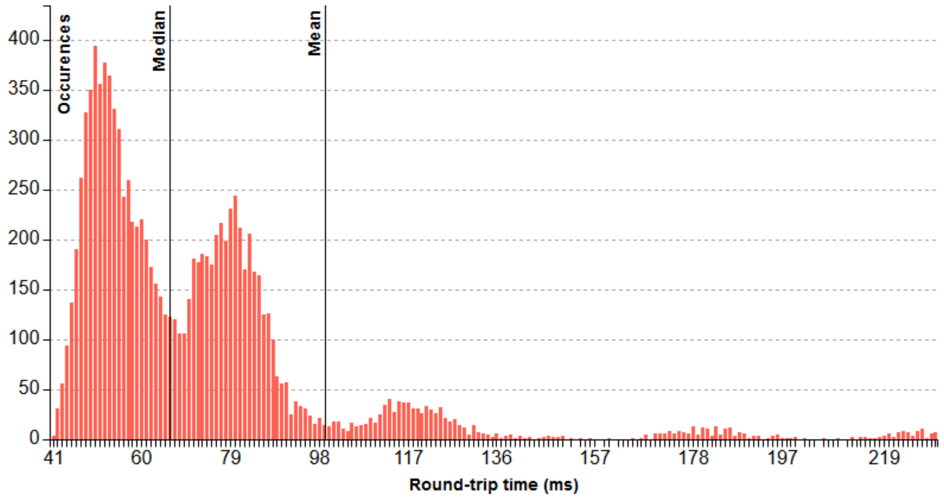

Figure 5.6: RTT distribution for HSPA networks

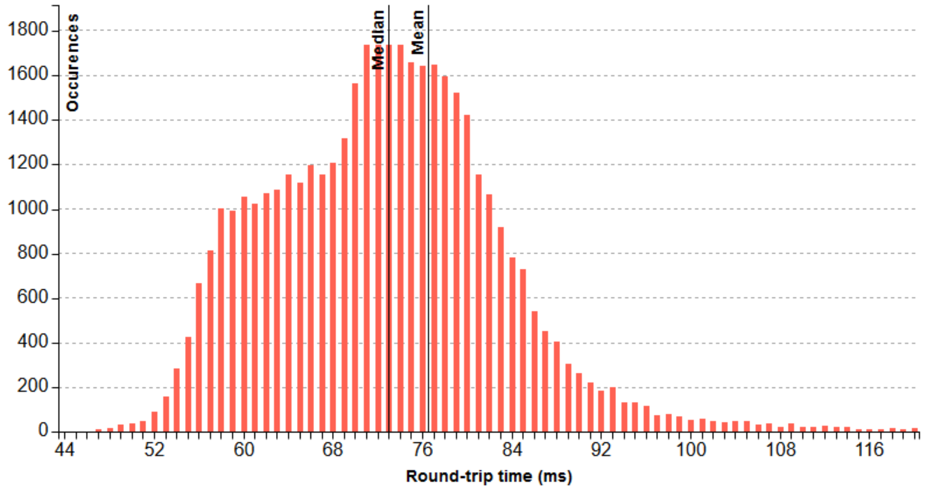

Figure 5.7: RTT distribution for LTE networks

As expected, all distributions are right-skewed. HSPA networks have the least amount of samples of the three, and we can see in the figure that the distribution shape is different from the other two. We believe this indicates that more samples from HSPA networks should be gathered, rather than the network behavior being different. On the other hand, the count for each occurrence are higher than of EDGE networks, perhaps suggesting more stability in HSPA than in EDGE.

5.3.2 Signal strength

First off, when connected to an LTE mobile network, the Android device can provide the LTE ASU level instead of the GSM ASU level, and sometimes both GSM and LTE ASU levels (see section 4.1.1). This could be dependent on the operator network configuration, as we found that when connected to LTE in the Trondheim area one would only get the LTE ASU level, while in the Lofoten area one gets both. The LTE ASU has been chosen as a basis for signal strength levels when available in an LTE network. From the dataset we find a strong correlation between LTE and GSM ASU levels (r = 0.86), so perhaps either one can be used.

Since we are trying to identify and learn more about mobile network dead spots, having a fair number of samples with low signal strength is ideal. In the case of 2G or 3G networks, we are unfortunately lacking measurements on the lower end of the scale. GSM ASU level distributions for EDGE and HSPA networks, and LTE ASU level distributions for LTE networks are shown in the following three figures. Recall from chapter 4.1.1 that GSM ASU levels range from 0 to 31, and LTE ASU levels from 0 to 97.

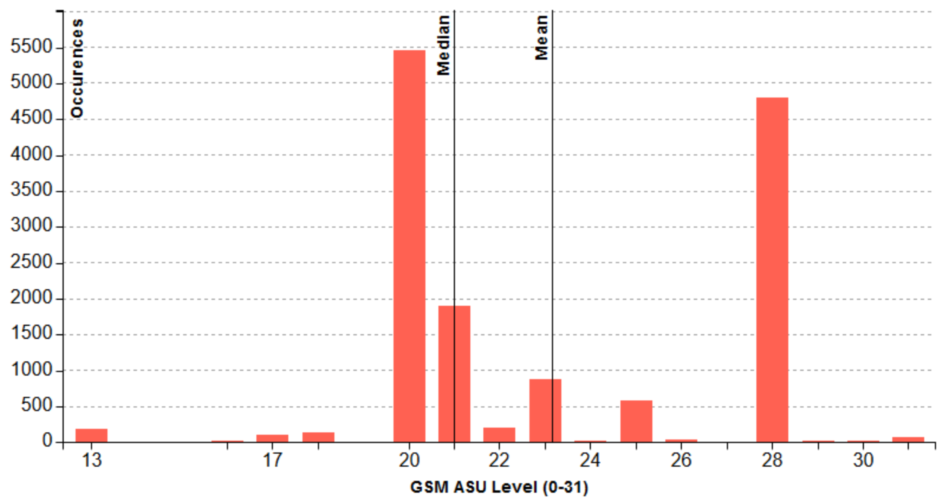

Figure 5.8: GSM ASU level distribution (EDGE)

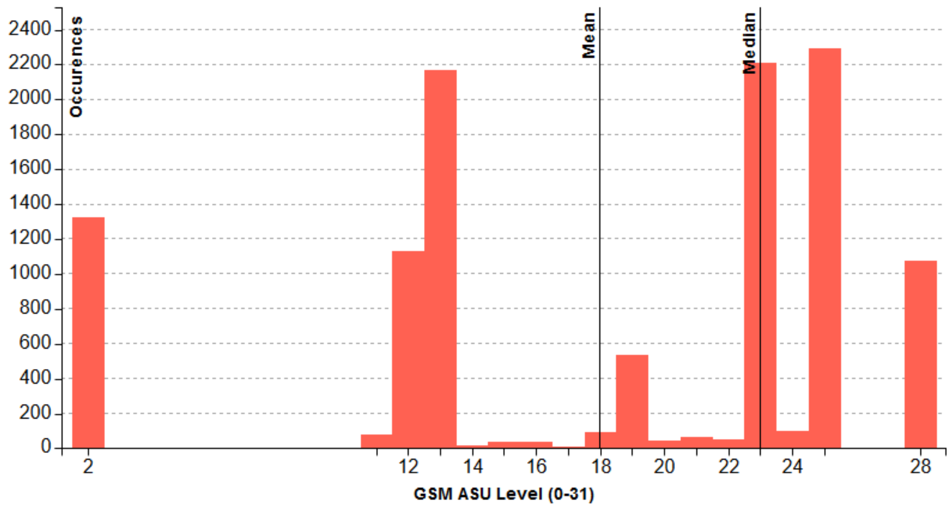

Figure 5.9: GSM ASU level distribution (HSPA)

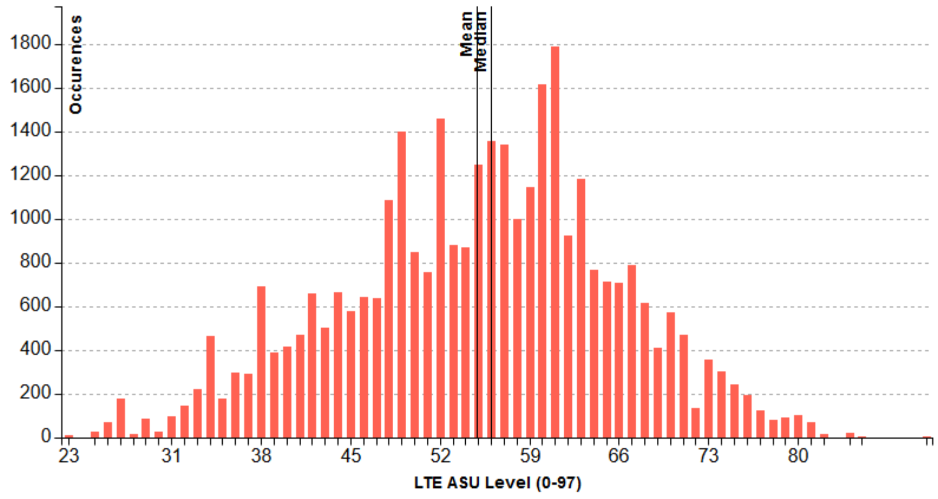

Figure 5.10: LTE ASU level distribution (LTE)

For GSM ASU levels, it is indeed evident that more measurements are needed, especially from poor connection areas. In EDGE networks, we have no measurements with a lower ASU level than 13. In HSPA networks we have measurements with a value as low as 2, but no measurements between values 2 and 11. As most of these measurements are from a small area of Trondheim this does not come as a surprise. We further inspected the dataset for HSPA and found that the GSM ASU value of 2 in HSPA networks originates from mainly one specific area from one trip. For LTE though, our coverage is better. The LTE ASU level distribution resembles a normal distribution, with samples both in the low and high end of the spectrum.

5.4 Metric correlations

For the network types discussed above (i.e. EDGE, HSPA and LTE), we present the correlation coefficients for:

- RTT and signal strength
- RTT time and IPDV
- Average RTT and packet loss

Signal strength is either calculated from the GSM ASU or LTE ASU levels. IPDV can be both positive or negative and is unwanted in both cases. Since we here are interested in correlations, we take the absolute value of the IPDV (hereby referred to as jitter). Packet loss has been low in our measurements overall; $< 1\%$ with about 500 packets per 100 000. The behavior of packet loss is similar to high round-trip times, where it occurs in short bursts. Per sample, packet loss is either true or false (the packet is either lost or not). Point-biserial correlation can be used for finding correlations between a numeric and a boolean value, but instead, we treat packet loss as an aggregated metric, calculating packet loss ratio per area. We therefore also aggregate the round-trip times of the corresponding areas into averages to obtain the correlations.

5.4.1 Findings for EDGE networks

As mentioned, in addition to our own measurements, some of the datasets used in [9] have also been imported. These datasets do not contain information needed to calculate IPDV. They do however contain samples from actual areas where there is poor network performance and where older network types like EDGE are present. Excluding imported datasets, about 26 000 measurements from EDGE networks have been collected. These measurements are from the same small, general area of Trondheim and were obtained using a preferred network type on the Android device.

The correlations are as follows:

- RTT and signal strength: $r = 0$
- RTT and jitter: $r = 0.64$
- Average RTT and packet loss: $r = 0.51$

In the work of [9], a relationship between round-trip time and signal strength was present especially for GSM ASU levels less than 15. For our measurements, the correlation coefficient ($r = 0$) indicates that there is no such relationship. But as we saw in the previous sub-chapter, we lack measurements with poor signal strength in EDGE networks - our measurements had no GSM ASU value less than 13. It is therefore difficult to draw any conclusions for an RTT and signal strength correlation from our measurements. We do see that a high RTT is associated with both jitter and packet loss - a moderate positive relationship ($r > 0.5$) is present.

5.4.2 Findings for HSPA networks

As the number of measurements for HSPA+ and HSDPA is very low, we specifically look at the HSPA network type. As with measurements from 2G networks, these are also obtained using preferred network type on the Android device.

- RTT and signal strength: r = 0.05
- RTT and jitter: r = 0.51
- Average RTT and packet loss: r = 0.47

The correlation coefficients suggest a relationship between RTT and packet loss, and RTT and jitter. There appears to be no correlation between average round-trip time and signal strength (p = 0.05). But as with EDGE networks, no conclusions should be drawn for the latter relationship, as measurements from poor signal strength areas are lacking. Compared to EDGE networks, the span between minimum average round-trip time and maximum is however found to be much smaller, which suggest that HSPA networks are more stable (at least in terms of this metric).

5.4.3 Findings for LTE networks

The sample size for LTE networks is close to 100 000, all which were sampled with the Android device automatically selecting the best available network type. The LTE ASU level is used as an indicator of signal strength for this network type.

- RTT and signal strength: r = 0
- RTT and jitter: r = 0.60
- Average RTT and packet loss: r = 0.58

The findings for LTE networks are generally the same as the other types in terms of correlations between metrics: no relationship between RTT and signal strength, and a moderate positive correlation between RTT and jitter, and RTT and packet loss. With a much larger sample size than of previously discussed network types, we can with more certainty say there is no relationship between RTT and signal strength for LTE networks. This is also illustrated in Figure 5.11, with average round-trip time for each distinct LTE ASU value shown as a graph. The RTTs do not seem to be higher for lower ASU values. The high average RTT on the right of the figure was further investigated and found to originate from around the opening of a tunnel for a specific trip.

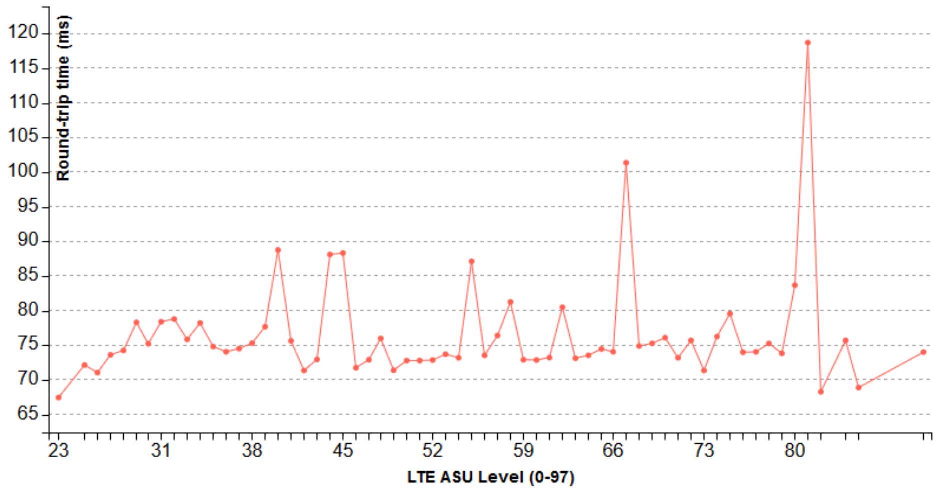

Figure 5.11: Round-trip time (average, ms) as a function of LTE ASU

Below the relationship between RTT and jitter (r = 0.60), using average round-trip time as a function of jitter, is shown.

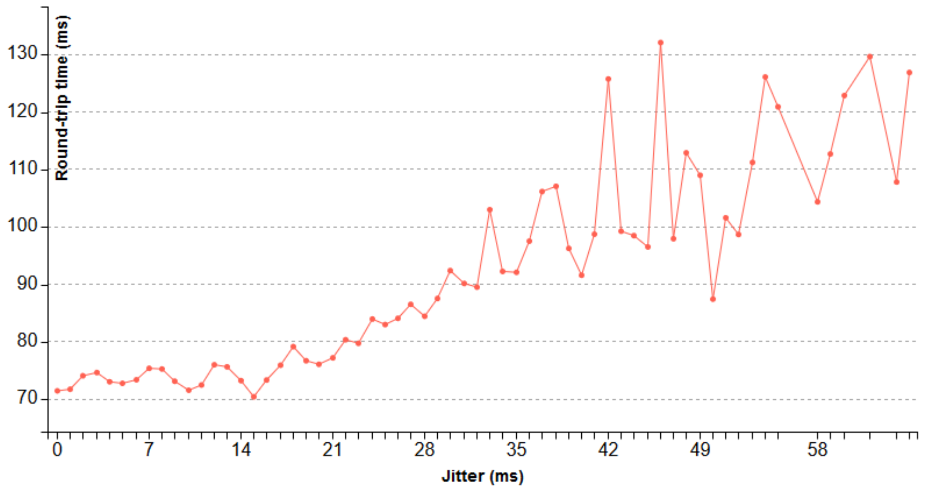

Figure 5.12: Round-trip time (average, ms) as a function of jitter (ms) (LTE)

5.4.4 Summary of metric correlations

Overall, we have not found a relationship between RTT and signal strength. We can say this with more certainty for LTE networks, but for EDGE and HSPA networks we should be more careful as we lack measurements on the low end of signal strength scale. Additionally, in the related work in [9], such a relationship was quite clear. The packet size for measuring RTT is fixed and is very small (40 bytes). Measuring with varying packet sizes could perhaps be a next step. We generally encourage obtaining more measurements from poor signal strength areas, preferably not using Android's preferred network type selection, but rather selecting best available to also get a better view of the actual coverage of different areas.

For all network types, relationships between RTT and jitter and average RTT and packet loss are evident. For RTT and jitter, the correlation coefficients are $r = 0.64$, $r = 0.51$ and $r = 0.60$ for EDGE, HSPA and LTE respectively. For average RTT and packet loss the values are $r = 0.51$, $r = 0.47$ and $r = 0.58$. As discussed, high values of RTT occur occasionally in bursts. This is generally when it gets coupled with packet loss and jitter. If this is not a result of poor signal strength, this could occur due to underlying temporary network performance degradation, such as network congestion.

5.5 Dead spot algorithm outputs

Recall that dead spots could occur in two different ways, either as insufficient performance or as a gap. Here we present various examples that prove that our application and algorithm can handle both cases. We also show computation times for the application to produce the outputs.

The input is an origin and a destination point in coordinates, and the output is a list of dead spots and their respective lengths in meters. Lengths are calculated using the PostGIS function ST_Distance. The specified start and end points of the dead spots are IDs of the corresponding areas and coordinates. The area IDs makes it easier to identify the location on the map, since our application logs area information to the web console when clicking on areas on the map.

5.5.1 Example with performance threshold

In this example there is a clear segment on the route from origin to destination that has worse performance. The performance of the yellow/green areas are around 0.5, and of the orange/red areas around 0.3. We specify a minimum performance of 0.4. The output then is:

```
Dead spot between
5151 (68.2682342, 14.1607782),
5176 (68.2815844, 14.1596285),
length: 1490 m
```

Area 5151 and 5176 are marked in the figure. The application used around 1 second to produce this output.

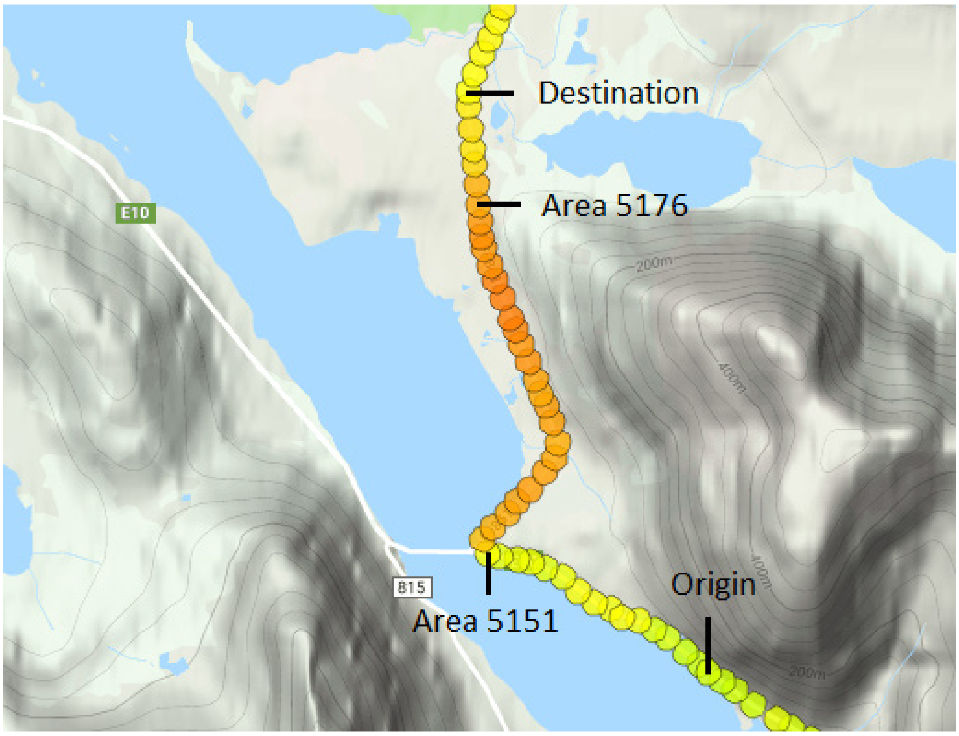

Figure 5.13: Dead spot algorithm output example 1

5.5.2 Example with gaps

In this example it is demonstrated how gaps are handled by the algorithm. To avoid any yellow/red areas to be identified as dead spots in this case, a very low minimum performance threshold of 0.01 has been specified. The output is:

```
Dead spot between
4072 (63.5874921, 9.6065109),
4073 (63.5830124, 9.5895948),
length: 977 m
```

The application used around 0.2 seconds to produce this output.

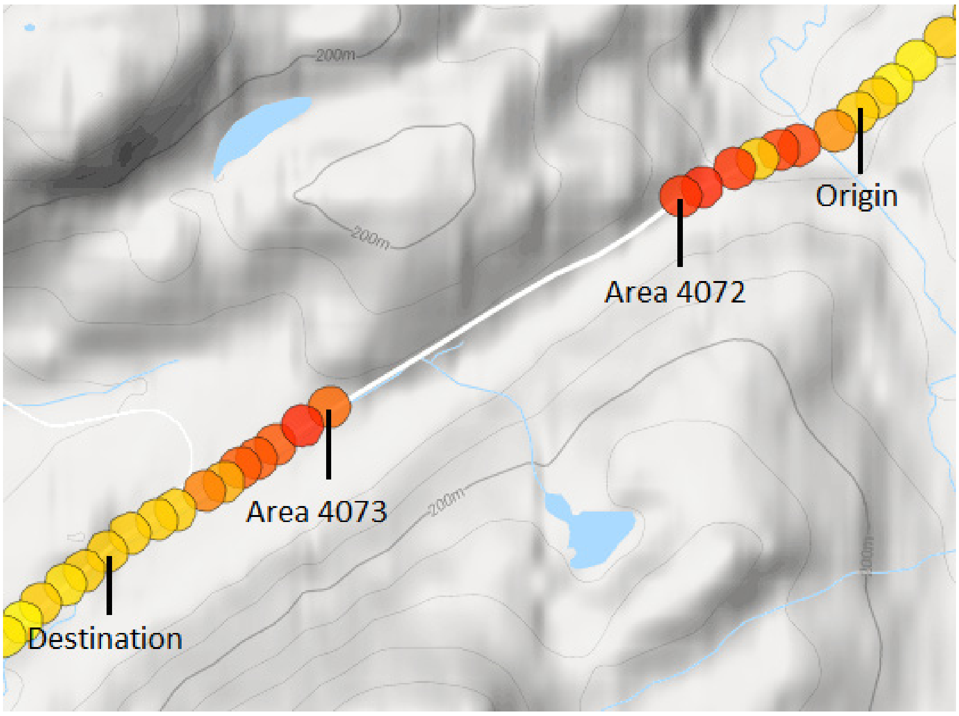

Figure 5.14: Dead spot algorithm output example 2

5.5.3 Example with performance threshold and gaps

In this last example there is a combination of poor performance areas and gaps. A minimum performance value of 0.3 is used, which excludes red and darker orange areas. The few orange areas between area 2494 and area 2518 happen to all have a performance value greater than 0.3, but the areas between 2518 and 2529 have a lower value. The output is:

```
Dead spot between
2488 (63.38390300050378, 10.572601994499564),
2494 (63.38578310329467, 10.585130006074905),
length: 661 m

Dead spot between
2518 (63.377765603363514, 10.601147320121527),
2529 (63.374154679477215, 10.61115799471736),
length: 642 m
```

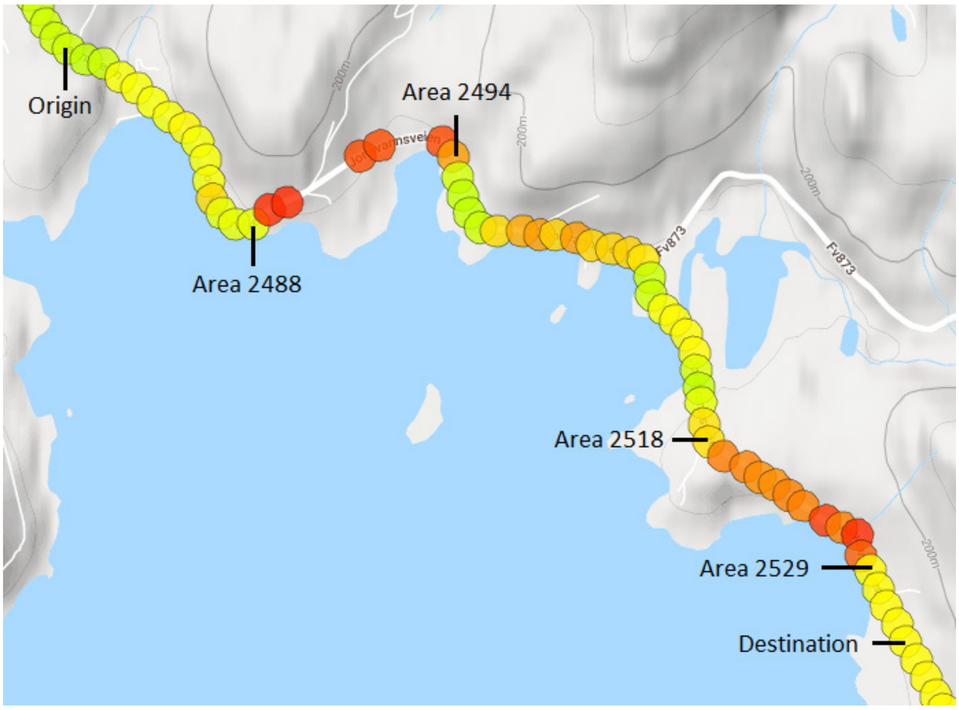

Figure 5.15: Dead spot algorithm output example 3

5.6 Server application scalability

Scalability has not been the main focus when developing this application but has still been addressed by several steps and seems to have paid off well. Database indexes, simultaneous client handling with threading and connection pooling are now essential parts of the application.

As mentioned at the end of chapter 4.2.2, aggregations are calculated when data is requested, rather than pre-calculated. There is also no caching of aggregated data involved. One reason for this is the complexity involved in implementing this, as many types of aggregations with different parameters would have to be stored. Throughout the work of the thesis, we for instance often wanted to only view data for a specified range of trips or a certain network type. Caching would potentially involve storing all combinations of trips and network types, which there are many of.

As the database grew to over 130 000 samples, there was no noticeable increase in the time it took for retrieving data from the DBMS. As explained in chapter 4.2.1, for every single new measurement, the application needs to check whether an area with that location within 50 meters already exists. From going from 0 to more than 3000 such areas throughout sampling, this check is still done in about 10 ms. Since samples are received in batch (the client uploads all samples when the stop button is pressed), this time does however add up. Receiving a batch of 10 000 measurements, this would be around 10 seconds just for calculating what area ID a sample should have. The insertion process itself and creating potential areas that do not exist would also add to this time. The application is by no means unresponsive during this time, data can still be accessed. Read-write safety is of course handled by the underlying DBMS.

The general performance and the scalability of the application seems to be quite good for its current use. Recall that performing and presenting aggregations for the over 80 000 samples in the Lofoten area was done in less than 1 second. The most complex calculations are performed when requesting dead spot information. For practical reasons, we have not been able to test how the application performs several clients request such information simultaneously. Having implemented connection pooling to the database and proper threading, this should however not be expected to be an issue.

Chapter 6 – Conclusion

A substantial amount of work in this project has been design, implementation and sampling. Having a few years of experience with software development from the industry was very helpful. A server that receives, stores and presents connectivity data has been successfully developed. A sampling application for Android has also been developed. The intended use for this application is in cars, but it is not limited to that mode of transportation. Walking and bicycle trips were also used for sampling data in this project. Recall that the server application is not dependent on the developed Android application – any client using the expected format of data can potentially be used. Cars on the road network can request both aggregated or raw versions of data to improve their communication decisions. The information can also be used to strategize communication by the infrastructure on the reverse path, i.e. in Infrastructure-to-Vehicle communication.

The metrics provided by the sampling application, such as round-trip time, signal strength and packet delay variation, are incorporated into a simple model for calculating the overall "network performance" of an area. With a minimum desired value for network performance, our application can identify so-called dead spots on the road network with an accuracy down to about 50-100 meters.

Quantifying network performance with multiple parameters is not an easy task, and different models for different areas of applications should perhaps be used. Investigating such models further is encouraged as future work. Measuring bandwidth proved to be more challenging than what we initially thought, as downloading files influence measurements of other metrics such as round-trip time. We also faced issues regarding positioning with Android devices, and general limitations of GPS. Suggestions for improvements to positioning issues are also addressed in the next chapter.

With the work of this thesis, we have arrived at a scalable software system for measuring and analyzing mobile network performance. The system provides maps and various statistics and is able to output subsets of the stored data based on several input parameters, such as network type(s) and specified trips. The software can be useful for further works in the problem area, at least at IIK, NTNU to begin with. Source code for all developed applications will be passed on for research purposes. It is shortly discussed and provided in Appendix A.

Chapter 7 – Future work

7.1 Improving positioning

As mentioned in chapter 5.1.3, measurements with no positioning data or with poor accuracy are unwanted. There exist several off-the-shelf products for positioning that utilize for instance speed, bearing and last position to be able to still determine position accurately (for limited periods of time) when GPS services are unavailable. An example of such equipment is positioning chips provided by u-blox, a company delivering positioning and wireless communication technologies (see https://www.u-blox.com/en). Their equipment can be connected to another device using a USB-connection. If decided to use such equipment, it would perhaps be better to use a laptop computer for running client software instead of a smart phone, as such external equipment is then easier to connect.

7.2 Improving network performance models

In section 4.2.3, we introduced a "performance function" as a model for representing network performance of areas in an aggregated manner. In practice, perceived network performance depends on the application. Different applications imply different requirements for each type of metric. As such, different models should be developed for a wide array of applications. For instance, one could have one model for "vehicle safety critical information", with emphasis on low delay, and another model for "end user experience", focusing on bandwidth and jitter. Parametrizing these models properly would require extensive testing and good knowledge about the requirements of the relevant application. Machine learning techniques could potentially be used to evaluate which parameters are important.

7.3 Security and message prioritization

In the current solution, any client with the correct message format and the server address information can deliver data to the server. This type of security might suffice for small-scale research purposes such as a master thesis, but addressing security more explicitly on a larger scale is crucially important. Without proper authorization, server data could become contaminated.

Currently, all requests to the application for network performance data and dead spots are treated equally. In a real-world system, prioritization of delivering safety critical information to vehicles on the road would be an important aspect. Applications concerning for instance video streaming should have a lower priority. Avoiding malicious use of the priority system would then be an aspect of security. The topic of net neutrality could then potentially also play a part.

Third, depending on the source of measurements, privacy could become a concern. Data could potentially be misused to find detailed information about the whereabouts of vehicles and when they have visited certain places. With the increased focus on privacy in recent times, with for instance the introduction of the General Data Protection Regulation (GPDR), addressing this issue responsibly in further development is encouraged.

7.4 The future role of 5G

The fifth generation of mobile networks, 5G, will also shape the future of communication in VNs. Several telecom companies are currently doing trials for 5G networks, and 3GPP has entered the second phase of standardizing them [5]. The ultra-low latencies of these networks will help to meet the low latency requirements of some security applications of VNs and be of great value to ITS applications in general. However, the first LTE standards were released over 10 years ago, and even in a developed country like Norway, older network types are still prevalent in some rural areas. In these networks, the latency of the RAN alone can be larger than the latency requirements of VN applications for road safety. Knowledge of these poor performance areas will still be important in many years to come.

References

[1] Y. Lin, P. Wang and M. Ma, "Intelligent Transportation System(ITS): Concept, Challenge and Opportunity," *ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids)*, Beijing, 2017.

[2] C. M. Silva, B. M. Masini, G. Ferraria and I. Thibault, "A Survey on Infrastructure-Based Vehicular Networks," *Mobile Information Systems,* 2017.

[3] Bitmovin, "Adaptive Streaming," [Online]. Available: https://bitmovin.com/adaptive-streaming/. [Accessed 28 September 2018].

[4] 3GPP, "3GPP Scope and Objectives," 31 August 2007. [Online]. Available: http://www.3gpp.org/ftp/Inbox/2008_web_files/3GPP_Scopeando310807 .pdf. [Accessed 10 November 2018].

[5] 3GPP, "Releases," [Online]. Available: http://www.3gpp.org/specifications/releases. [Accessed 20 November 2018].

[6] J. Barrachina, J. A. Sanguesa, M. Fogue, P. Garrido, F. J. Martinez, J.-C. Cano, C. T. Calafate and P. Manzoni, "V2X-d: A vehicular density estimation system that combines V2V and V2I communications," *2013 IFIP Wireless Days (WD)*, Valencia, 2013, pp. 1-6.

[7] G. Neonakis Aggelou and R. Tafazolli, "On the relaying capability of next-generation GSM cellular networks," *IEEE Personal Communications,* vol. 8, no. 1, pp. 40-47, 2001.

[8] R. Prasad, C. Dovrolis, M. Murray and K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools," *IEEE Network,* vol. 17, no. 6, pp. 27-35, 2003.

[9] E. Puka, P. Herrmann, T. Levin and C. B. Skjetne, "A Way to Measure and Analyze Cellular Network Connectivity on the Norwegian Road System," *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, Bengaluru, 2018, pp. 595-600.

[10] J. Huang et al., "An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance," *ACM SIGCOMM 2013*, 2013.

[11] Q. Xu et al., "Cellular data network infrastructure characterization and implication on mobile content placement," *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, 2011.

[12] I. Grigorik, High Performance Browser Networking, O'Reilly Media, 2013, pp. 103-104.

[13] "Center for Applied internet Data Analysis," [Online]. Available: http://www.caida.org. [Accessed 10 October 2018].

[14] "RIPE NCC," [Online]. Available: https://www.ripe.net/. [Accessed 14 September 2018].

[15] J. Corral, G. Texier and L. Toutain, "End-to-end active measurement architecture in IP Networks (SATURNE)," *IP networks (SATURNE)," in PAM2003 - A workshop on Passive and Active Measurements*, 2003.

[16] F. Michaut and F. Lepage, "Application-oriented network metrology: metrics and active measurement tools," *IEEE Communications Surveys & Tutorials,* vol. 7, no. 2, pp. 2-24, 2005.

[17] M. Mathis and M. Allman, "A Framework for Defining Empirical Bulk Transfer Capacity Metrics," RFC 3148, July 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3148.

[18] R. H. Jakobsen, "Message Forwarding between Vehicles in Dead Spots," 2018.

[19] P. Papadimitratos, A. D. La Fortelle, K. Evenssen, R. Brignolo and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *IEEE Communications Magazine,* vol. 47, no. 11, pp. 84-95, November 2009.

[20] A. E. Fernandez et al., "5GCAR Scenarios, Use Cases, Requirements and KPIs," 2017.

[21] "NorNet," [Online]. Available: https://www.nntb.no/. [Accessed 10 January 2019].

[22] "Norwegian mobile broadband - fourth annual report from CRNA," Simula.no, 2017. [Online]. Available: https://www.simula.no/news/norwegian-mobile-broadband-fourth-annual-report-crna. [Accessed 3 February 2019].

[23] OpenSignal, "Methodology," [Online]. Available: https://opensignal.com/methodology. [Accessed 18 October 2018].

[24] K. Majithia, "Verizon slams latest OpenSignal study on 4G networks," Mobile World Live, 10 February 2017. [Online]. Available: https://www.mobileworldlive.com/featured-content/home-banner/verizon-slams-latest-opensignal-study-on-4g-networks/. [Accessed 12 January 2019].

[25] J. Callaham, "Google says there are now 1.4 billion active Android devices worldwide," Android Central, 2015. [Online]. Available: https://www.androidcentral.com/google-says-there-are-now-14-billion-active-android-devices-worldwide. [Accessed 1 February 2019].

[26] Google Git, "Android Source Code: SignalStrength.java," [Online]. Available: https://android.googlesource.com/platform/frameworks/base/+/master/telephony/java/android/telephony/SignalStrength.java. [Accessed 20 October 2018].

[27] Google Git, "Android Source Code: CellSignalStrengthLte.java," [Online]. Available: https://android.googlesource.com/platform/frameworks/base/+/master/telephony/java/android/telephony/CellSignalStrengthLte.java. [Accessed 20 October 2018].

[28] Google Developer, "Fused Location Provider API," [Online]. Available: https://developers.google.com/location-context/fused-location-provider/. [Accessed 15 September 2018].

[29] Android developer reference, "TrafficStats," [Online]. Available: https://developer.android.com/reference/android/net/TrafficStats.html. [Accessed 21 September 2018].

[30] "Facebook Network Connection Class on GitHub," [Online]. Available: https://github.com/facebook/network-connection-class/. [Accessed 8 September 2018].

[31] Cisco, "Quality of Service for Voice over IP," 2001. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/qos_solutions /QoSVoIP/QoSVoIP.pdf. [Accessed 20 January 2019].

[32] D. C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)," RFC 3393, November 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3393.

[33] H. Brombach, "Vraker TCP i neste versjon av HTTP," digi.no, [Online]. Available: https://www.digi.no/artikler/vraker-tcp-i-neste-versjon-av-http/451076. [Accessed 15 November 2018].

[34] J. Anuskiewicz, "Measuring jitter accurately," 2008. [Online]. Available: https://www.lightwaveonline.com/articles/2008/04/measuring-jitter-accurately-54886317.html. [Accessed 25 October 2018].

[35] V. Paxson et al., "Framework for IP Performance Metrics," RFC 2330, May 1998. [Online]. Available: https://www.rfc-editor.org/info/rfc2330.

[36] "Resilient Networks Mobile Broadband Measurements," [Online]. Available: http://robustenett.no/map. [Accessed 10 December 2018].

[37] J. M. Hillenbrand, "Lecture on jitter," [Online]. Available: https://homepages.wmich.edu/~hillenbr/501/PerturbationDemo.ppt.

[38] Google Maps Platform, "Directions API Developer Guide," [Online]. Available: https://developers.google.com/maps/documentation/directions/intro. [Accessed 16 September 2018].

[39] M. Shafirov, "Kotlin on Android. Now official," Kotlin Blog, 2017. [Online]. Available: https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official. [Accessed 2 September 2018].

[40] OpenSignal, "The State of LTE (February 2018)," [Online]. Available: https://opensignal.com/reports/2018/02/state-of-lte. [Accessed 20 November 2018].

[41] C. Ovando, J. Pérez and A. Moral, "LTE techno-economic assessment: The case of rural areas in Spain," *Telecommunications Policy,* vol. 39, no. 3, 4, pp. 269-283, 2015.

[42] J. Nyquist, "Nå er det snart slutt for 3G – dette betyr det for deg," online.no, December 2018. [Online]. Available: https://www.online.no/dekning/3g-fases-ut-4g-overtar. [Accessed 20 December 2018].

[43] M. Lorentzen, "Mobilgiganter stengte GSM-nettet for å gi plass til mer 4G: Det vil ta årevis før Norge følger etter," e24.no, 2017. [Online]. Available: https://e24.no/digital/telenor/mobilgiganter-stengte-gsm-nettet-for-aa-gi-plass-til-mer-4g-det-vil-ta-aarevis-foer-norge-foelger-etter/23910642. [Accessed 5 January 2018].

Appendix A – Source code

The source code of the Android application, the measurement server and the server application combined contain (only) about 2000 lines of code. Including the web pages with HTML and JavaScript replied by the web server, closer to 2500. Note that compared to for instance Java, Kotlin is quite concise in syntax, and saves a lot of code lines when it comes to boilerplate code like data classes. Functional programming idioms are used extensively in the code for handling measurement data and also saves a lot of space compared to more “traditional” methods. Consider for instance the following example that calculates the most common network type among result rows:

```
data.commonNetworkType = rows
        .map { it[Measurements.networkType] }
        .groupingBy { it }
        .eachCount() // gives Map<String, Int> with counts
                     // for each value.
        .maxBy { it.value }?.key
```

Since the measurement server and server application were hosted on the same machine during the thesis, they are contained in the same project file. They can be found at:

https://github.com/jameyer/connectivity-data-server

The source code for the Android application can be found at:

https://github.com/jameyer/connectivity-data-android-client