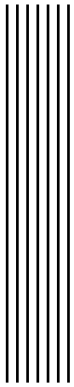
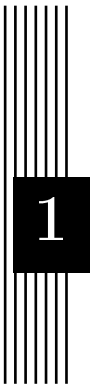


Contents

List of Corrections



1 Introduction

1.1 Chapter Desc

The following is a brief description of each chapter in the report

1.1.1 Design and Implementation

1.1.2 Test

1.1.3 Future Work

1.1.4 Conclusion

2 Problem Analysis

When using a mobile device on the go, a good network connection is not always available, but relevant and updated data may still be needed. This naturally raises the question:

How can a user be guaranteed to always have relevant information available, regardless of network connectivity?

In the following, we will present some possible scenarios, in which this problem can occur.

2.1 Different Needs in Different Contexts

In order to solve the problems described in the scenarios it becomes important for a single application to tell the different contexts apart, so it is able to solve the right problem at the right time. This is necessary because not all of the problems are relevant in all contexts, e.g. predicting when a connection might drop with the intention of downloading a map would not have much use if the user were traveling by bus.

The context of a user can be split into two categories: the locational and the activity context. The locational context refers to the location of the user, e.g. a user in a forest is likely to have different needs than a use in a city or at home. The activity context refers to the activity of a user, e.g. whether the user is traveling in a vehicle or by foot, or whether or not they have an appointment. The application must be able to determine the user context in order to adhere to the user's needs.

3 Prototypes

3.1 Handlers

3.1.1 Packaging

A solution to losing network services and therefore the possibility to download maps, are to detect when the device is losing network-connection in appropriate time. Hereby the device may start downloading maps over the nearby area and store for use once the network connection is lost. These maps should then be compartmentalized into groups with different prioritzation, based on a criteria of nearest-package-first.

Downloading the by nearest-package-first requires an indexation of map-packages for a given location of a device. Since our service are to provide maps even when network connection is lost, we intend to use a map-service like Google Maps and not develop our own. Google Maps already divided a map into small packages, which we could use for downloading and displaying upon request depending on simple GPS-coordinates. New packages will be downloaded and displayed when the user moves the map on a mobile-device. The problem that has to be dealt with, are which packages to download without knowing more than GPS-coordinates.

We put forward three solutions to this problem:

- Define a circular perimeter around the GPS-coordinates. Start downloading map-packages on the device GPS-coordinates, then prioritize after nearest-package-first until the perimeter is covered with downloaded packages.

- Define a rectangular perimeter around the GPS-coordinates and proceed as above.

- Process the GPS-coordinates over time, which will provide direction and velocity of the device. This should be used to download packages in the direction that the device is moving, and disregard the package behind the device. Considering the velocity, the width of required packages can be defined and as velocity increase packages further away can be downloaded. This solution

does not follow nearest-package-first defined on first set of GPS-coordinates but rather try to dynamically follow nearest-package-first based on time and movement.

Advantages	Disadvantages
• adv • adv	• disadv • disadv

Table 3.1: this is a dummy

3.2 Supplements

4 Architechture

5 Transmission Control Protocol

The Transmission Control Protocol (TCP) is a part of the Internet protocol suite. Together with the Internet Protocol (IP), TCP is so widely used that the entire Internet protocol suite, containing many protocols, is often just called TCP/IP.

In the following, the parts of TCP that are particularly important for this project are described. This chapter is based on [4]. [1]

5.1 Data Transfer

Because we are working with transfer of data to a device with an uncertain connection, it is relevant to consider how TCP handles transferring data. An important aspect of this is to ensure reliable transmission, i.e. to make sure all the data is transferred correctly. To ensure this, each byte of data gets a sequence number, allowing the destination host to reconstruct the data in case of for example packet loss. Additionally, when a packet is received, the receiver sends back an acknowledgment, and if such an acknowledgment is not received the packet will be sent again. To ensure correctness of the packet content, each packet has a checksum included which the packet's content can be compared to upon arrival at its destination.

This possibility of error checking comes in handy in our project. When a map is transmitted to a client, it ensures that the map data is not corrupted. This is very desirable since corrupted map data could make the map unusable in the best case, and show wrong map data in the worst case.

5.1.1 Flow Control

It is possible for the server to sent more data than the receiver can process, and to accommodate this a flow control protocol is used. When data is sent, the receiver

[1] FiXme Fatal: Find better source.

answers back with a *window size*, telling the sender how much more data it is able to process. If the window size reaches 0, a persist timer will be set to account for the possibility that the updated window size was simply lost. When the timer runs out the server will send a small packet, probing the receiver for an updated window size.

5.1.2 TCP/IP versus UDP

We would like to consider two aspects of TCP/IP versus UDP. The first aspect is which protocol is cheapest in space usage when transferring from one unit to another. The second aspect is which protocol offers the best suited services for our application.

These questions are related to the problem we try to solve. We want to try to download data on an endangered connection or download within a limited time-span before the device goes offline. Therefore we investigate TCP/IP versus UDP further.

When transferring data from one unit to another, the size of data to send varies depending on the program. However, the header has a minimum or fixed size it always uses, without considering the size of the data the user wants to send. In TCP/IP, the size of the header is 21 octets[1]. In UDP, the size of the header is 8 octets[2]. In comparison, UDP uses 38% of what TCP/IP uses for each exchange.

The services provided by TCP/IP are reliability through error checking and delivery validation whereas UDP emphasizes low-overhead operation and reduced latency[4]. Both protocols provide valid services to be used when solving our problem. TCP/IP will ensure we have correct, consistent and working data. UDP could be used for transferring GPS-coordinates because they will be transferred extensively when the application is running. Losing an UDP transaction will not break the application. If a few data is lost or inconsistent, new data will be sent in the very near future.

5.2 Location Detection and Predicting Signal Loss

5.2.1 Determining Location

As pointed out in Section 2.1 it is important to know the user's locational context, to make sure to supply the most relevant data. It is particularly important to determine whether the user is in an urban area or not. This can be done by comparing the user's current GPS coordinates to the map data, to determine if the coordinates are within the boundaries of a city. This method has a number of disadvantages: If it is a small city, there may not be many cell towers nearby, and the problem may be similar to the one encountered in non-urban areas. Conversely, a well connected area might not be within a city border. On top of that it can be hard to determine if a certain GPS coordinate is within a city or not.

A simpler and more dynamic method is to measure the connectivity and number of nearby cell towers to determine how likely it is for a user to keep an internet connection. In most cases, if there are many cell towers nearby it means the user is in or near a city. Using this method it is hard to be completely sure without facing

the same problems as with the first method, but it is a straight-forward solution, which allows the development to focus on more import areas of the problem, and which can relatively easily be improved later.

5.2.2 Predicting Signal Loss

To predict when a device might suffer a signal loss and become disconnected from the internet, two different scenarios must be considered. The scenarios are as follows

- The device is moving into the country side
- The device is in the city[2]

In order to predict when the signal may be at risk of being lost in the first scenario, it makes sense to look at measurements from the antenna of the device. A number of measurements, five for example, can then be stored on the device. These values should be error corrected (e.g. using standard deviation) to make sure sudden fluctuations will not trigger an alarm, telling the device that it has lost its signal. Since coverage is usually worse outside of big cities compared to in the cities, the signal will be lost gradually and not suddenly. Therefore it makes sense to measure a number of antenna measurements, and decide whether they are falling at a rate which would indicate heading towards the country side.

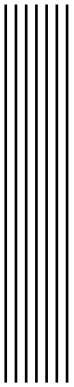
This functionality uses the moving average, which has the purpose of smoothing out temporary fluctuations in measurements. [3]

In the city, it is very hard to accurately predict signal loss, even with precise heat maps. This is partly because phones and their antennas vary in quality and partly because signal loss in cities is sporadic and the signal usually returns quickly. Therefore signal loss is not predicted in the city, but only outside.

[2] FiXme Warning: should this be removed? is the scenario not needed anymore?

6 Implementation

7 Conclusion



Bibliography

[1] Information Sciences Institute, U. o. S. C. (2014, October 23). *DARPA Internet Program Protocal Specification*. http://tools.ietf.org/pdf/rfc791.pdf.

[2] J. Postel, I. (2014, October 23). *User Datagram Protocol*. http://tools.ietf.org/rfc/rfc768.txt.

[3] Wikipedia (2014a, October 20). *Moving average*. http://en.wikipedia.org/wiki/Moving_average.

[4] Wikipedia (2014b, October 12). *Transmission Control Protocol*. http://en.wikipedia.org/wiki/Transmission_Control_Protocol.