KING'S College LONDON

University of London

7CCS4PRJ Final Year Individual Project

iBus Disruption Monitor

Real-Time Visualisation of Bus Delays in London using AVL Data

A project in collaboration with Transport for London

Final Project Report

Author: Konstantin Vladimirov Draganov

Supervisor: Dr Steffen Zschaler

Course: MSci Computer Science

Student ID: 1101314

September 2014 - April 2015

Abstract

Automatic Vehicle Location (AVL) systems for bus fleets have been deployed successfully in many cities. They have enabled improved bus fleet management and operation as well as wide range of information for the travelling public. However there is still a lot of area of utilisation of the data that these AVL systems generate and produce. This report explores and analyses the tools and applications currently available at Transport for London (TFL) bus operation unit. The report then proposes a prototypical tool for monitoring the bus delays in real time in the network. This tool offers objective source of processed information to bus operators and control room staff. However further work need to be done in order to place this tool in production environment as urban delay detection is very complex and unpredictable as will the report justify.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Konstantin Vladimirov Draganov

September 2014 - April 2015

Acknowledgements

First and foremost I offer my sincerest gratitude to my supervisor Dr Steffen Zschaler for the continuous support, guidance, encouragement, insightful comments and hard questions throughout the course of this exciting project.

I would also like to thank Andrew Highfield and Keith Elliot from TFL for providing me with all the needed data, information and feedback which has been immensely helpful.

Last but not the least, I would like to thank my parents Vladimir and Veselka also my sister Lilia and my partner Simona for their support and patience not only during the project, but throughout my life as well.

Contents

Chapter 1

Introduction

1.1 Motivation

London bus network is one of the largest and most advanced bus networks in the worlds. It is responsible for more than 2.4 billion passenger journeys a year [2]. The constant population growth of England's capital has been also driving the expansion and improvement of the transport networks across the city. Transport for London (TFL) is in charge of its operation and its bus network if recognised as one of the top in the world in terms or reliability, affordability and cost-effectiveness [2]. The capital's bus network is continually increasing along with the city's population [3]. Maintaining such a large scale network requires careful planing and monitoring. Being able to maintain such high reliability service 24/7 364 days in the year requires employing new technologies. This also helps keep costs down and thus keeping the service more affordable and accessible for the general travelling public. Each bus in the TFL network has been equipped with state of the art GPS enabled automatic vehicle location (AVL) system named iBus[1]. This AVL system has led to improved fleet management and has enabled the creation and improvement of multiple applications [4]. The system is generating large sets of data both in real-time as well as historical data. This helps, the bus operators and the emergency control room at TFL responsible for maintaining the bus network

(CentreComm), to better manage and maintain the smooth operation of the bus network.However there are currently problems for which CentreComm staff are required to manually analyse these data sets in order to discover in which parts of the networks problems are occurring. This is because there is lack of readily available preprocessed information. This is very impractical and time consuming and currently they ultimately rely on individual bus operators and drivers to notify them of possible problems. Once alerted of a possible disruption in the network they can start their own investigation into first verifying what they have been told by the bus drivers/operators and then into finding the cause and the actual severity of the problem. This often could lead to spending time and resources into investigating non existent problems. This is where this project comes in place to address this inefficiency and to propose, implement and evaluate a prototypical tool for real time monitoring of the bus network.

1.2 Scope

The scope of this project is to analyse the current work flow of CentreComm operators and their needs. This has to be followed by designing, implementing and evaluating a prototype which to aid the control room staff. This tool has to work and analyse the data available in real time. The main two problems that are in the scope of this project and need to be solved are:

- Analysing and identifying the disruptions in the network in real-time using the available data.
- Producing a prioritised list of the disruptions classified using rules gathered during meetings and discussions with the key stakeholders from TFL.
- Visualising the calculated list in an appropriate format.
- Evaluating the performance of the tool. This is essential as it need to be proven that the results could be trusted.

1.3 Aims

The main aim of this project is to design and implement a real-time visualisation tool which to highlight disrupted routes or parts of the TFL bus network where disruption might happen or have already happened even before the bus drivers or operators have noticed and alerted CentreComm. This could could be subdivided into two smaller aims:

- The first one which is independent of the other is to enable the processing of the data generated by the buses in the TFL's bus network. The tool need to be able to analyse the input data sets and calculate and output a list of the disruption that are observed in the network. It has to present information regarding the location (section) in the transport network and their severity.
- The second part of the main aim above is to visualise the generated output in an easy to use and understand way.

1.4 Objectives

1.5 Report Structure

In order to help the reader I have outlined the project structure here. The report would continue in the next chapter by providing the reader with all the background knowledge needed for the rest of the report. This would include brief of background on the current work-flow CentreComm operators follow and its inefficiencies. I will also give background on the iBus system and the data that the tool would need to operate with. I then explore related work that has already been done and how ours differs. This is followed by alternative approaches and models that could be utilised. Afterwards the report focusses on the specific requirements that have been identified and gathered from CentreComm. The report then goes on to discuss the design and the implementation of the proposed system. This is followed then by Chapter 5

and 6 which address testing and evaluation of the prototype. I conclude the report with a summary of what has been achieved and guidance how the work presented in this report could be further developed and improved.

Chapter 2

Background research

The background should set the project into context by motivating the subject matter and relating it to existing published work. The background will include a critical evaluation of the existing literature in the area in which your project work is based and should lead the reader to understand how your work is motivated by and related to existing work.

2.1 CentreComm

CentreComm is TFLâĂŹs emergency control room for all London buses. It is has been in operation for more than 30 years [12] and it employs a dedicated team who work 24/7 and are responsible for dealing with more than a 1000 calls each day. These are call either from bus drivers or bus company operators regarding incidents and disruptions in the bus network. CentreComm staff are responsible for maintaining the smooth, reliable and sage operation of London busy bus network consisting of more than 8000 buses [13]. Their job has been made easier with the introduction of a modern and innovative AVL system. CentreCommâĂŹs way of operation has been transformed beyond recognition since it first opened and today. It has started as few people and pen and paper to modern, electronic real-time GPS systems nowadays [12]. However there is still a lot of room for improvement in their way of operation.

2.2 iBus

AVL stands for Automatic Vehicle Location. iBus is an AVL system which has been developed by Trapeze ITS (http://www.trapezegroup.co.uk/) for TFLâĂŹs needs. The iBus system is complex and consists of a number of computer systems, sensors and transmitters as described in [8]. One of the key components of the system on-board unit (OBU) which mounted on each of buses in the TFL bus fleet and consists of a computational unit connected to sensors and GPS transmitters (see figure 1 below). This OBU is responsible for a number of tasks including a regular transmission of the bus location. This information is currently used by the different bus operators for fleet management as well as CentreComm. There are already modern and innovative fleet management and public information systems which incorporate iBus to improve the service which are not further discussed here as they are not related to the problem this project is focussed on. CentreComm is not responsible for the fleet management as this is contracted to the bus operators which are responsible for maintaining reliable service according to agreed contracts. The emergency control room comes in place when there are planned or unplanned events which disrupt the transport network. They are also responsible for helping the bus operators once they cannot maintain the service they are responsible for due to traffic congestion or other issues which are beyond their control. However currently CentreComm relies on the bus drivers and bus operators for letting them know of such cases as they do not have a system which to signal them about these issues. All the information they need is there and they have access to it however they do not have the resources to manually monitor each of the 8000+ buses.Currently CentreComm has system which displays information for all routes what is the expected and actual arrival time of a bus at stop. Staff use this tool for analysing and trying to find out if there is a problem once they have been alerted by bus drivers or operators. This is very inefficient and not in sync with 21st century. They also have access to a map which display each single bus and their status, but no overall network or route status.

2.3 Data

iBus system generate very short telegram messages with the bus location [8]. This information is then processed on a server and more information is derived. The information we are interested is the deviation from the schedule. This is calculated by knowing the expected arrival time of the bus at a stop from the schedule and is compared to the observed time. This value is calculated the same way for both low and high frequency buses(Low frequency buses are supposed to run according to a fixed schedule (e.g. a bus should arrive at stop at predefined time) and usually routes which have less than 5 buses an hour. High frequency bus routes should maintain headways - this meaning a bus should be arriving at stop at predefined intervals (e.g. each 2 minutes)). More about the supplied data for this project would be covered in the requirements section.

2.4 Approaches

2.4.1 Time series analysis

Simple moving average

Weighted moving average

Exponential moving average

2.4.2 Peak detection

2.4.3 Autoregressive moving average

2.4.4 Machine learning

2.4.5 Historical

2.5 Summary

Chapter 3

Requirements

In this subsection I have introduced and formalised the current requirements. This is an important step of any project as it formalises the problems that the project is trying to address. It also allows to be used as a measure for evaluating the success of the project once it has been completed. The requirements presented below could evolve during the progressing of the project however they are expected to remain mostly the the as defined.

3.1 User Requirements

The user requirements provide a list of the functionalities that the user is expecting to be able to see in the end product. The system is expected to be able carry out and achieve the following actions and tasks: âĂć The tool must be able to produce a prioritised list of the disruption in the bus network that it has knowledge of. âĂć The tool must be prioritising the disruptions according to the user defined rules (these are still discussed and gathered from the user) The tool must be updating this list of disruptions whenever there is more data. This should happen as real-time as possible. âĂć The tool must be able to provide more details regarding a disruption (e.g. which section and which routes are affected and what is the severity). âĂć The user must be able to interact with the system in order to lower or increase the priority of a given

disruption (even ignore one). Based on the above list of requirements the use case diagram below presents the way in which users (actors) interact with the tool (system).

3.2 Functional Requirements

System requirements are the specification of how the system behaves. These are requirements are built using the user requirements as an input. They are details of what the system (tool) should be able to accomplish technically. âĂć The system must be able to read and process CSV (comma-separated values) files.The system must be able to calculate and perform time-series analysis on the data producing list of the disruption in the network according to configurable threshold parameters (e.g. delay of more than 20 minutes in a given section of a route). âĂć The system must be able to update it self as near real-time as possible without user interaction or request. âĂć The system must be able to visualise the output as list. âĂć The system must be able to expand each list item and display further details. The functional requirement require us to deal with CSV file which are currently updated each 5 minutes and pushed each 15 minutes. The technical group at TFL has agreed that it would not be problem accessing the 5 minute snapshots. However going more real-time than 5 minutes probably would be left for post deployment as it would require a formal change request to be raised which costs time and resources. The system must be capable of updating itself as near real-time as possible even though for the moment we work with 5 to 15 minutes update intervals. The CSV files that the tool would have to work contains number of fields including: âĂć Unique vehicle identifier. âĂć Time when the data has been received. âĂć The type of the bus trip (e.g. normal trip with passengers, trip to depot etc.). âĂć The route number. âĂć The last stop. âĂć Deviation from the schedule as described in the background section above. 11 âĂć Longitude and latitude. There a number of other fields which are not interesting with respect to this project. The data is unordered and there are multiple files produced one for

each bus operator company. The tool should be able to aggregate all the data and analysis it and treat it as a single network.

Chapter 4

Design & Implementation

This section focuses on modelling the requirements defined in the previous section in an abstract level. This allows us to better organise and structure our problem. Below I have presented initial design decisions and diagrams related to the project. This however are expected to evolve during the course of the project.

4.1 Use cases

4.2 System architecture

4.3 Class organisation

4.4 Database

4.5 Software Platform

4.6 User Interface

Chapter 5

Testing

The disruption engine consists of a number of individual components and algorithms. These are tested using a number of unit tests. Due to the nature of the engine, stress tests were carried out to test it for any memory leaks and performance issues. Evaluation of the output of the disruption engine will be performed by first carrying out further literature review to find some guidance on what window for the data to consider to use and what weights to use for optimal results. This will be followed by running the system on a given set of data (e.g. a week worth of AVL data) and comparing the output with the actual state of the network during this period.

The user interface consists of a simple web application capable of displaying list of disruptions. Testing and evaluation of this web application will be performed as user testing. As I mentioned above, some feedback and problems were identified which will be addressed and fixed. Follow up user tests will be carried out by giving access to friends and family to the web site to use and give feedback on. This should give reasonable confidence in the correctness of the user interface as there is no complex logic incorporated in the web front end application.

5.1 Unit Testing

5.2 Functional Testing

5.3 Integration Testing

5.4 Stress Testing

Chapter 6

Evaluation

6.1 System Evaluation

6.2 Usability

6.2.1 Project Retrospective

A project post-mortem, also called a project retrospective, is a process for evaluating the success (or failure) of a project's ability to meet business goals.

Post-mortems can encompass both quantitative data and qualitative data. Quantitative data include the variance between the hours estimated for a project and the actual hours incurred. Qualitative data will often include stakeholder satisfaction, end-user satisfaction, team satisfaction, potential re usability and perceived quality of end-deliverables.

Chapter 7

Professional and Ethical Issues

Either in a seperate section or throughout the report demonstrate that you are aware of the **Code of Conduct & Code of Good Practice** issued by the British Computer Society and have applied their principles, where appropriate, as you carried out your project.

Chapter 8

Conclusion and Future Work

The project's conclusions should list the key things that have been learnt as a consequence of engaging in your project work. For example, "The use of overloading in C++ provides a very elegant mechanism for transparent parallelisation of sequential programs", or "The overheads of linear-time n-body algorithms makes them computationally less efficient than $O(n \log n)$ algorithms for systems with less than 100000 particles". Avoid tedious personal reflections like "I learned a lot about C++ programming...", or "Simulating colliding galaxies can be real fun...". It is common to finish the report by listing ways in which the project can be taken further. This might, for example, be a plan for turning a piece of software or hardware into a marketable product, or a set of ideas for possibly turning your project into an MPhil or PhD.

8.0.2 Conclusion

8.0.3 Future Work

It could use peak detection to make distinction between incidents and congestion. Data from other source could be used (taxis AVL, couriers services AVL) etc. Historical data could be employed in order to make further analysis and

correlations with weather data, time or the day/week/year etc. Increasing the frequency of the data means that we could make use of the actual geo location information in order to calculate and monitor the bus speeds rather than the preprocessed schedule deviation value.

References

[1] Bowen T. Clarke R. and Head J. Mass deployment of bus priority using real-time passenger information systems in london. pages 891–921, 2007.

[2] Transport for London Media. https://www.tfl.gov.uk/info-for/media/press-releases/2014/may/annual-passenger-journeys-on-london-s-buses-top-2-4-billion. Online, May 2014. Online; accessed 2-December-2014.

[3] BBC News. Extra 500 buses planned for growing capital before 2021. http://www.bbc.co.uk/news/uk-england-london-30285777, 2014. Online; accessed 2-December-2014.

[4] Alan Wong and Nick Hounsell. Using the ibus system to provide improved public transport information and applications for london. Paper 01753, July 2010.

Appendix A

Extra Information

A.1 Use Case Diagram

A.2 Architecture Diagram

A.3 Class Diagram

A.4 Relational Model

Appendix B

User Guide

You must provide an adequate user guide for your software. The guide should provide easily understood instructions on how to use your software. A particularly useful approach is to treat the user guide as a walk-through of a typical session, or set of sessions, which collectively display all of the features of your package. Technical details of how the package works are rarely required. Keep the guide concise and simple. The extensive use of diagrams, illustrating the package in action, can often be particularly helpful. The user guide is sometimes included as a chapter in the main body of the report, but is often better included in an appendix to the main report.

B.1 Installation

B.2 Execution

B.3 Dependencies

Appendix C

Source Code

Complete source code listings must be submitted as an appendix to the report. The project source codes are usually spread out over several files/units. You should try to help the reader to navigate through your source code by providing a "table of contents" (titles of these files/units and one line descriptions). The first page of the program listings folder must contain the following statement certifying the work as your own: "I verify that I am the sole author of the programs contained in this folder, except where explicitly stated to the contrary". Your (typed) signature and the date should follow this statement.

All work on programs must stop once the code is submitted. You are required to keep safely several copies of this version of the program - one copy must be kept on the departmental disk space - and you must use one of these copies in the project examination. Your examiners may ask to see the last-modified dates of your program files, and may ask you to demonstrate that the program files you use in the project examination are identical to the program files you had stored on the departmental disk space before you submitted the project. Any attempt to demonstrate code that is not included in your submitted source listings is an attempt to cheat; any such attempt will be reported to the KCL Misconduct Committee.

You may find it easier to firstly generate a PDF of your source code using a text editor and then merge it to the end of your report.

There are many free tools available that allow you to merge PDF files.

C.1 Engine

C.2 Web Front End

C.3 Database