Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project - Final Report 2017



Project Title: **Distributed Road Traffic Speed Monitoring**

Student: **Jeremy Chan**

CID: **00818433**

Course: **4T**

Project Supervisor: **Dr. Ed Stott**

Second Marker: **Dr. Christos Papavassiliou**

# Abstract

Speed cameras have been shown to slow traffic down and prevent accidents. Specifically, average speed cameras are more effective than fixed position cameras as they prevent speeding after the motorist has passed the camera. However, around 84% [1] of the UK’s average speed cameras are placed on busy trunk routes as it is not economically feasible to place them in rural areas. Moreover, the design of traditional speed cameras means their widespread deployment on residential roads would be detrimental to the streetscape.

This project therefore enables the use of average speed cameras in any area. It is a low-cost, distributed vehicle speed monitoring system that communities can install themselves within private property next to a street. The system enables civilians to use their own personal cameras to detect and share license plate information by leveraging computer vision algorithms such as LBP and OCR. Plates, along with time and location data, are then directly broadcasted to other peers in the network without the need for a central server. Violations are detected based on the average speed method using public mapping data, and users notified if an infraction occurs. Simulations, along with real world testing, show positive results under multiple lighting and camera conditions.

# Acknowledgements

I would like to express my heartfelt gratitude to my project supervisor, Dr Ed Stott, for his enthusiastic guidance, encouragement and useful critiques of this final year project. I would also like to thank Dr. \*\*\*, for her advice and assistance in keeping my progress on schedule. My grateful thanks are also extended to Mr. \*\*\* for his help in doing the meteorological data analysis, to Ms \*\*\*, who helped me calculate the wind pressure coefficient and to Ms \*\*\* and Mr. \*\*\* for their support in the site measurement.

I would also like to extend my thanks to the technicians of the laboratory of the \*\*\* department for their help in offering me the resources in running the program.

Finally, I wish to thank my parents for their support and encouragement throughout my study.

Table of Contents

[Abstract 2](#_Toc484441833)

[Acknowledgements 3](#_Toc484441834)

[1 Introduction 5](#_Toc484441835)

[2 Related Work 6](#_Toc484441836)

[2.1 License plate recognition 6](#_Toc484441837)

[2.1.1 Hardware 6](#_Toc484441838)

[2.1.2 Software 6](#_Toc484441839)

[2.2 Peer to peer network 7](#_Toc484441840)

[2.3 Photo evidence publication 7](#_Toc484441841)

[2.4 Overall Feasibility 7](#_Toc484441842)

[3 Project Specification 9](#_Toc484441843)

[3.1 Project goals 9](#_Toc484441844)

[3.2 Advanced project goals 9](#_Toc484441845)

[3.3 Clarifications 9](#_Toc484441846)

[4 Background Reading and Literature Review 11](#_Toc484441847)

[4.1 License plate recognition 11](#_Toc484441848)

[4.2 Peer to peer network 13](#_Toc484441849)

[4.3 Photo evidence publication 14](#_Toc484441850)

[5 Design 15](#_Toc484441851)

[6 Build and Implementation 16](#_Toc484441852)

[7 Testing 17](#_Toc484441853)

[8 Evaluation 18](#_Toc484441854)

[9 Deployment 19](#_Toc484441855)

[10 Conclusion and Future Work 20](#_Toc484441856)

[11 References 21](#_Toc484441857)

[12 Appendix 24](#_Toc484441858)

# Introduction

The number of people speeding on the roads in the UK have actually decreased over the past 2 years [2] as a result of there being more speed cameras being installed across the country [3]. From [4], it is clear that ‘the higher the speed, the greater the probability of a crash’. Hence, it is in the best interest of the authorities to monitor speeding convictions using average speed check cameras.

However, most of the average speed check cameras across the UK are installed in busy trunk routes, i.e. motorways. They use automatic license plate recognition technologies to track the entry and exit times. This data is not accessible for the general public and is kept for two years [5]. This project aims to remedy that, given that many countries have supported the idea of using license plate readers and that there should not be any restrictions surrounding who and when can license plates be collected [6].

There are also very little amounts of speed cameras in rural and local roads. By using a decentralised network that can find out peers in its vicinity, anyone can download, compile, and use this project as a pseudo-speed camera in their local area, removing the need for expensive cameras and reliance on a closed database of license plates that only the police have access to. Anyone caught speeding will have their license plate posted onto a social network. Given that there is a set procedure that police use in issuing speeding tickets, the end goal is just to have the evidence on the social networks and not to actually ticket the offender.

# Related Work

Research into already existing products was done on the three main goals of the project, to judge the market feasibility of the project, and to prevent overlap with existing work where possible.

## License plate recognition

### Hardware

There are many license plate recognition systems available on the market, with many of them being used in speed cameras around the country [7]. Traffic cameras can generally be separated into radar based and optical based systems, with radar based cameras checking the instantaneous speed of the vehicle as it moves past, and optical based systems checking the average speed of the vehicle as it passes between two points. There has been a steady increase in average speed check cameras over the years [3], mostly along motorways but also by local councils [8]. These average speed check cameras often include multiple enticing features like 24/7 operation and 4G connectivity, as such in Jenoptik’s VECTOR cameras [9].

### Software

Most of the license plate recognition software available is proprietary and closed source. Some area only available for commercial use. These systems generally have features such as video stream processing, ability to detect plates from multiple points of view, and cloud services to take in a video stream and output detected license plates along with their timestamps [10]–[12]. However, the price of the commercial systems (if available for purchase) are definitely out of budget for this project, as shown in Table 1. ARES (<http://platesmart.com/>) did not respond to an enquiry regarding cost of usage, so is not shown.

As the project specification specifies that ‘existing computer vision algorithms’ should be used, the choice was made to use the consumer version of OpenALPR. The main reasons for this was that it was open source, and very feature rich. Additionally, a choice was made to instead replicate the commercial features of OpenALPR using computer vision processing libraries instead of ignoring the extra features or paying for them as they were standard vision processing techniques, e.g. background subtraction, motion detection.

Table 1: Prices of different license plate detection systems

|  |  |  |
| --- | --- | --- |
| System | Price (one-off) | Price (per month) |
| DTK LDR SDK | 190 – 1430 EUR | N/A |
| OpenALPR | 1000 USD | 50 USD |

## Peer to peer network

Peer to peer networks (P2P) have existed for a long time and there are numerous implementations available for use on the Internet. The main uses for peer to peer networks are web, messaging, and file sharing.

Bitmessage (<https://bitmessage.org>) is used for encrypted messaging, and is decentralised. It also uses strong authentication to prevent spoofing of the sender of a message [13]; however, it removes information about the sender and receiver. It is therefore not suitable for this project as this system needs to know information about the sender to accurately work out an average speed.

Telehash (<https://github.com/telehash/telehash.github.io>) is open source, fully end-to-end encrypted, enforces strict privacy rules, and cross platform. It also supports JSON message sending with unique sender and receiver ID’s. Unfortunately, the development is focused around using Node.JS and not Python or C++ (the Python binding repository is still empty at time of writing). Therefore, it is not suitable as it would take too much time to implement.

## Photo evidence publication

There are official API’s for most social networks on uploading images to their respective social networks so there is no need to reinvent the wheel, no is there a need to use a third party service to upload images.

## Overall Feasibility

Overall, the project is very feasible as there has been a lot of work in all three areas, proving that the ideas in the project are not a dead end. Moreover, there is a free and open source implementation for the license plate recognition system which is immensely useful in providing a solid head start in license plate detection, a primary goal of the project. As most advanced features are behind a commercial paywall, a novel aspect of the project is to implement new algorithms / research to better improve the detection rate of the license plate detection.

The network is also a very feasible part of the project. P2P networks have extensive research and Python has libraries which support development using web sockets and different communication protocols. However, since the license plate detection is done using existing libraries, there has to be extra emphasis placed on the networking side to make sure this is the primary novel aspect to the project. At the time of writing, there has not been any projects tying license plate detection to a decentralised network which seeks out peers in its immediate vicinity.

Lastly, there are extensive documentation on uploading images to social networks – no foreseeable problem there unless the API access is revoked.

# Project Specification

From the project description, the goals of the project can be separated into the following:

## Project goals

* Implement a number plate recognition system using existing computer vision algorithms on a low-cost, readily available hardware platform.
* Set up a peer-to-peer network to share vehicle passing times and detect violations without the need for a central server.
* Publish photo evidence of any violations

## Advanced project goals

* Use the changes in the number plate geometry as the vehicle passes to detect the instantaneous speed of a vehicle. This provides a stand-alone mode that will aid adoption in areas where there isn't already an established network.
* Implement automatic peer discovery so that each device can find its neighbours and calculate the minimum legal transit time between them using a public mapping database.
* Add an encryption layer so that a hacker or rogue peer cannot use the network to track the movements of law-abiding vehicles.
* Package the system so that it can be easily installed in a home by an inexperienced user.

## Clarifications

After discussions with Dr. Stott, the following clarifications were made:

1. The system should target license plates and deployment in the United Kingdom, and license plates of the UK format [14]. Custom license plates will not be in the scope of the project for simplicity.
2. The number plate recognition system should be targeted at an off the shelf package, so there should be minimal setup and calibration done. This also means anyone, with the right equipment, should be able to download and compile the system if they have existing hardware.
3. The low-cost, readily available hardware platform will be a Raspberry Pi (RPi), with a camera attached to it. Using an RPi combines the best of cost (~£40 at time of writing), power (quad core CPU [15]), flexibility (camera can be any USB webcam or RPi’s official cameras), and support (development work on the RPi is extensive and there are ample tutorials/information online).
4. The peer to peer network should ideally be fully decentralised, so the system should be able to find peers without the help of a central server.
5. Publishing photo evidence will most likely be done onto a social network.
6. The public mapping database will be one accessible to most people – Google Maps API. There is a speed limit API but it is only open to premium users (<https://developers.google.com/maps/documentation/roads/speed-limits>).
7. A hacker or rogue peer should not be able to extract license plates from the system remotely, nor should they be able to spoof detect license plates as another user to avoid framing an innocent driver.
8. If possible, there should be a whitelist of emergency vehicles in the case of there being an emergency vehicle over the speed limit.
9. The final package should be a software package uploaded onto GitHub, with clear instructions on how to compile and run the program with examples. As this project involves testing on hardware as well, a list of recommended hardware should also be provided (after successful testing), so the project can be easily replicated in the future.

# Background Reading and Literature Review

## License plate recognition

Most license plate recognition systems operate on a similar basis. Pre-processing, detection, and character recognition [16]. However, image processing and detection can used interchangeably out of order, as in the case of OpenALPR [17].

In the pre-processing stage, a combination of techniques can be used to make the image more recognisable to the following pipeline stages. Azad et al. [18] utilises the HSV colour space instead of the RGB colour space to better determine the location of the plate as the plate is assumed to be of a certain colour. Some implementations implement both the RGB and HSV colour space to get saturation and intensity maps [19]. Duan et Al. [20] first converts the image to greyscale, then normalises the histogram to perform histogram equalisation to get a picture with better contrast. This is especially important during the night when there are a lot of dark areas from shadows. An example is shown below in Figure 2.

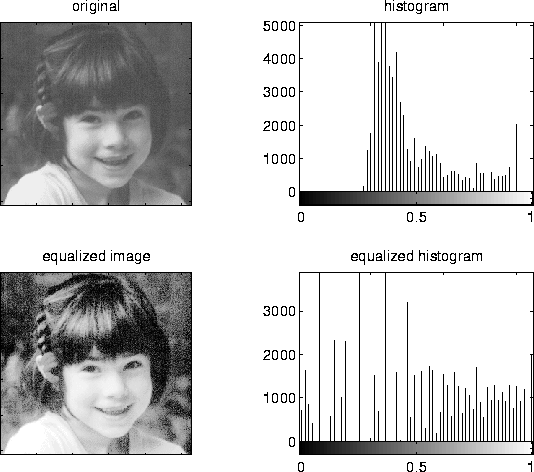


Figure 2: Histogram Equalisation (<http://electronicsinourhands.blogspot.co.uk/2012/10/histogram-equalization-in-image.html>)

In the detection phase, several algorithms are prominent. Edge detection is used extensively in Badr et al. [21], where the Sobel edge detection method is used in conjunction with thresholding techniques to determine the plate’s location in areas with high vertical lines. However, these methods often fail if the assumption that the license plate is captured from a fixed face-on angle is false as the plate can no longer be guaranteed to have perfectly vertical lines. Moreover, using edge detection as the first step often brings false positives, where buildings and road signs can also exhibit perfectly vertical lines – however they are not an area of interest for the license plate detection system.

Hence, other implementations such as OpenALPR’s and Liu et al. [22] use an algorithm called Local Binary Patterns (LBP), generally used in face recognition, to get areas of the image where it thinks there is a license plate. However, since LBP uses a sliding window approach, the performance of the detection is quite CPU dependent [17] (can be accelerated using a CUDA but the RPi does not support this). This method has been proven by Nguyen and Nguyen [23], who successfully used LBP with extra classifiers and algorithms in a real time license plate detector.

Edge detection is not unused, however. The most often used combination is a mixture of edge detection algorithms (Sobel, Laplacian, Canny), the Hough transform, and the contour algorithm. The Hough transform is used in implementations like [20] and OpenALPR [17] to find the actual edges of the plate as detection only gives a rough area for the location of the plate, and may end up with an area that does not have a plate in, or slightly bigger than the plate.

Other implementations such as Kwaśnicka and Wawrzyniak [24], and Chang et al. [19] use a technique called connected component analysis before applying the Hough transform. This technique is applied to a binary image of the license plate, and rejects any connected components whose aspect ratio is outside a prescribed range (a license plate is rectangular). The reliability of each implementation has yet to be tested against each other, however.

Lastly, the plate area needs to be warped to be a rectangle for the Optical Character Recognition (OCR) algorithms to work their best. A perspective transformation is often used to warp the image – the final image looks like it has been taken from another perspective (Figure 3). Moreover, straight lines remain straight [25] after the transformation so the next stage, character recognition, is not compromised. OCR libraries such as tesseract (<https://github.com/tesseract-ocr>) are then used to get the alphanumeric characters from the plate.

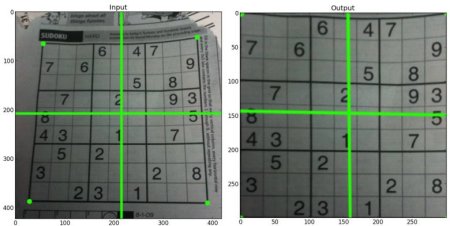


Figure 3: Using a perspective transformation [25]

## Peer to peer network

A P2P network is a network where many machines connect in an ad-hoc manner (they do not join and leave the network based on a schedule), and serve as a server for others. This means there is no need for files and messages or whatever the network is serving to pass through a central server, providing no one single point of failure.

P2P networks account for a significant portion of the web’s traffic [26], but its widespread nature means it comes with a few downsides as well. Making sure all copies of a file is free from corruption is one big challenge, as well as security. As all machines are servers of their own, a malicious hacker may attempt to distribute a virus or malware over the network to the many users using the network.

In this project, the main area of P2P networks that will be examined are decentralised networks. Centralised networks still have a central server that does routing, and peer finding, among other tasks [27], whereas decentralised networks do not.

Peer finding in a decentralised network is a challenge. As there is no central server, decentralised networks use techniques such as flooding the network with discover requests [27] and randomly pinging IP’s around the world to see if they are part of the network [28]. A well-known decentralised network, Gnutella, uses another technique where it finds new peers by expanding its search radius in its broadcasting phase. When a peer accepts this connection, it rebroadcasts the new peer’s address to its own peers, but decreases a counter called Time-to-Live (TTL) to ensure the message eventually dies out (when TTL=0) [29].

However, the broadcasting strategy in decentralised networks means as the search radius increases, there is an increasing amount of traffic in the network, and this may cause congestion in the network [29].

Other types of peer-finding techniques can be found in Mastroianni et al. [30].

## Photo evidence publication

Image upload API’s exist for many social networks, and the following table shows the documentation link for three popular image sharing sites.

Table 3: Photo upload API's

|  |  |
| --- | --- |
| Network | Link |
| Facebook | <https://developers.facebook.com/docs/php/howto/uploadphoto> |
| Twitter | <https://dev.twitter.com/rest/reference/post/media/upload> |
| Imgur | <https://api.imgur.com/endpoints/image> |

# Design

* Justify using Django
* Justify using HTTP requests
* Justify using ‘requests’ library
* Justify not having custom protocol, instead using HTTP verbs
* Justify using Python
* Justify using OpenALPR
* Justify add pre-processing
* Justify using JSON
* Justify using saved videos, and not real time detection (batch up operations, allow idle periods)
* Justify encryption using JWT
* Justify no HTTPS
* Justify trust and how it decays
* Justify bootstrap server
* Justify having peer/bootstrap projects (bootstrap modelled as peer, but separate dev environments)
* Justify broadcast as opposed to GET

Add web interface for user

# Build and Implementation

* Django structure
* Where to store sensitive data
* Use of admin panels to change database data
* Wrapping in try/except
* Using python in built functions if possible
* Logic flow of modules
* sectoring plates based on string similarity

# Testing

* Initially test while developing
* Use logbook to mark down stuff
* Save useful links/compilation instructions
* Hard to find good test spots (portable battery, need wifi, hard to develop on the go, need roads, hard to find good footage online)
* Started off by still pictures
* Then transitioned to videos of me walking around with parked cars (moving license plates anyway)
* Then transition to recording videos at the ends of roads
* Hard to find sheltered places to put camera overnight (safety, etc)
  + Dissapointing
* Most sheltered places are not on ground floor – leads to warping, hard to compensate, resolution problems
* Measure performance, FPS, CPU usage, memory
* Automated testing for correct plates (manually type out plates)
* Testing suite for new software changes

# Evaluation

* User feedback on web interface
* Feel how useful this thing is
* What are main selling points
* What features would you be looking for?
* Is recording license plate moral?
* How effective is this system?
* How well can it detect plates? In what environment? In what weather? What camera angle?
* How fast can it share this data? How fast until notification to the user?
* Accurate sharing? To the right peers?
* How useful is the violation report shown to the user?

# Deployment

* Github
* Readme
* Startup scripts
* CRON jobs

# Conclusion and Future Work

# References

[1] ‘UK SPECS Speed Camera Locations including the latest sites’. [Online]. Available: https://www.speedcamerasuk.com/specs-speed-camera-locations.htm. [Accessed: 05-Jun-2017].

[2] ‘Number of speeding convictions from average speed cameras - a Freedom of Information request to Driver and Vehicle Licensing Agency’, *WhatDoTheyKnow*, 28-Jul-2015. [Online]. Available: https://www.whatdotheyknow.com/request/number\_of\_speeding\_convictions\_f. [Accessed: 22-Jan-2017].

[3] ‘Number of average speed cameras have doubled in three years’, *This is Money*, 31-May-2016. [Online]. Available: http://www.thisismoney.co.uk/money/cars/article-3617584/Number-average-speed-cameras-doubled-three-years.html. [Accessed: 23-Jan-2017].

[4] ‘FS\_Speed.pdf’. [Online]. Available: http://m.swov.nl/rapport/Factsheets/UK/FS\_Speed.pdf. [Accessed: 29-Jan-2017].

[5] ‘Automatic Number Plate Recognition - Police.uk’. [Online]. Available: https://www.police.uk/information-and-advice/automatic-number-plate-recognition/. [Accessed: 29-Jan-2017].

[6] ‘2014\_04\_29-Californians-Overwhelmingly-Support-Use-of-License-Plate-Readers-and-Their-Ability-to-Solve-Crimes.pdf’. [Online]. Available: https://vigilantsolutions.com/wp-content/uploads/2014/04/2014\_04\_29-Californians-Overwhelmingly-Support-Use-of-License-Plate-Readers-and-Their-Ability-to-Solve-Crimes.pdf. [Accessed: 23-Jan-2017].

[7] ‘The UK’s Speed Camera Types | Fixed and Mobile speed cameras explained’. [Online]. Available: https://www.speedcamerasuk.com/speed-camera-types.htm. [Accessed: 23-Jan-2017].

[8] Y. D. Silva, ‘Average speed enforcement (ASE) camera’. [Online]. Available: https://www.birmingham.gov.uk/info/20163/road\_safety/364/average\_speed\_enforcement\_ase\_camera. [Accessed: 23-Jan-2017].

[9] ‘specs3\_vector\_v1.1\_final.pdf’. [Online]. Available: http://www.jenoptik.co.uk/sites/vysionics.vmdrupal04.lablateral.com/files/specs3\_vector\_v1.1\_final.pdf. [Accessed: 23-Jan-2017].

[10] ‘ARES | Fixed ALPR’, *PlateSmart*, 18-Jun-2015. .

[11] ‘LPR/ANPR License Plate Recognition SDK’. [Online]. Available: http://www.dtksoft.com/dtkanpr.php. [Accessed: 23-Jan-2017].

[12] ‘OpenALPR Features’. [Online]. Available: http://www.openalpr.com/features.html. [Accessed: 23-Jan-2017].

[13] J. Warren, ‘Bitmessage: A peer-to-peer message authentication and delivery system’, *White Pap. 27 Novemb. 2012 Httpsbitmessage Orgbitmessage Pdf*, 2012.

[14] ‘inf104-vehicle-registration-numbers-and-number-plates.pdf’. [Online]. Available: https://www.gov.uk/government/uploads/system/uploads/attachment\_data/file/533255/inf104-vehicle-registration-numbers-and-number-plates.pdf. [Accessed: 29-Jan-2017].

[15] ‘Raspberry Pi 3 Model B’, *Raspberry Pi*. .

[16] V. Sharma, P. Mathpal, and A. Kaushik, ‘Automatic license plate recognition using optical character recognition and template matching on yellow color license plate’, *Int. J. Innov. Res. Sci. Eng. Technol.*, vol. 3, no. 5, 2014.

[17] ‘Accuracy Improvements — openalpr 2.2.0 documentation’. [Online]. Available: http://doc.openalpr.com/accuracy\_improvements.html#openalpr-design. [Accessed: 24-Jan-2017].

[18] R. Azad, F. Davami, and B. Azad, ‘A novel and robust method for automatic license plate recognition system based on pattern recognition’, *Adv. Comput. Sci. Int. J.*, vol. 2, no. 3, pp. 64–70, 2013.

[19] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, ‘Automatic License Plate Recognition’, *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 1, pp. 42–53, Mar. 2004.

[20] T. D. Duan, T. H. Du, T. V. Phuoc, and N. V. Hoang, ‘Building an automatic vehicle license plate recognition system’, in *Proc. Int. Conf. Comput. Sci. RIVF*, 2005, pp. 59–63.

[21] A. Badr, M. M. Abdelwahab, A. M. Thabet, and A. M. Abdelsadek, ‘Automatic number plate recognition system’, *Ann. Univ. Craiova-Math. Comput. Sci. Ser.*, vol. 38, no. 1, pp. 62–71, 2011.

[22] L. Liu, H. Zhang, A. Feng, X. Wan, and J. Guo, ‘Simplified Local Binary Pattern Descriptor for Character Recognition of Vehicle License Plate’, in *Imaging and Visualization 2010 Seventh International Conference on Computer Graphics*, 2010, pp. 157–161.

[23] T.-T. Nguyen and T. T. Nguyen, ‘A real time license plate detection system based on boosting learning algorithm’, in *Image and Signal Processing (CISP), 2012 5th International Congress on*, 2012, pp. 819–823.

[24] H. Kwaśnicka and B. Wawrzyniak, ‘License plate localization and recognition in camera pictures’, in *3rd Symposium on Methods of Artificial Intelligence*, 2002, pp. 243–246.

[25] ‘OpenCV: Geometric Transformations of Images’. [Online]. Available: http://docs.opencv.org/3.1.0/da/d6e/tutorial\_py\_geometric\_transformations.html. [Accessed: 25-Jan-2017].

[26] The Government of the Hong Kong Special Administrative Region, ‘PEER-TO-PEER NETWORK’. [Online]. Available: http://www.infosec.gov.hk/english/technical/files/peer.pdf. [Accessed: 25-Jan-2017].

[27] H. Park, R. I. Ratzin, and M. van der Schaar, ‘Peer-to-peer networksprotocols, cooperation and competition’, *Streaming Media Archit. Tech. Appl. Recent Adv.*, pp. 262–294, 2010.

[28] C. Grothoff and C. GauthierDickey, ‘Bootstrapping Peer-to-Peer Networks’. [Online]. Available: http://grothoff.org/christian/dasp2p.pdf. [Accessed: 21-Jan-2017].

[29] Q. H. Vu, M. Lupu, and B. C. Ooi, *Peer-to-Peer Computing: Principles and Applications*. Springer Science & Business Media, 2009.

[30] C. Mastroianni, D. Talia, and O. Verta, ‘Designing an information system for Grids: Comparing hierarchical, decentralized P2P and super-peer models’, *Parallel Comput.*, vol. 34, no. 10, pp. 593–611, Oct. 2008.

# Appendix