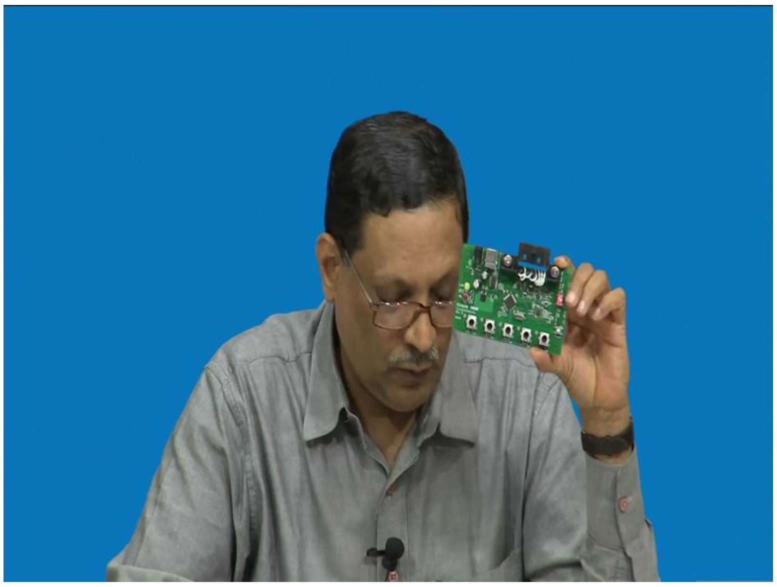
Advanced IOT Applications

Dr. T V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science, Bangalore

Lecture – 22
Real time event processing and Anomaly detection

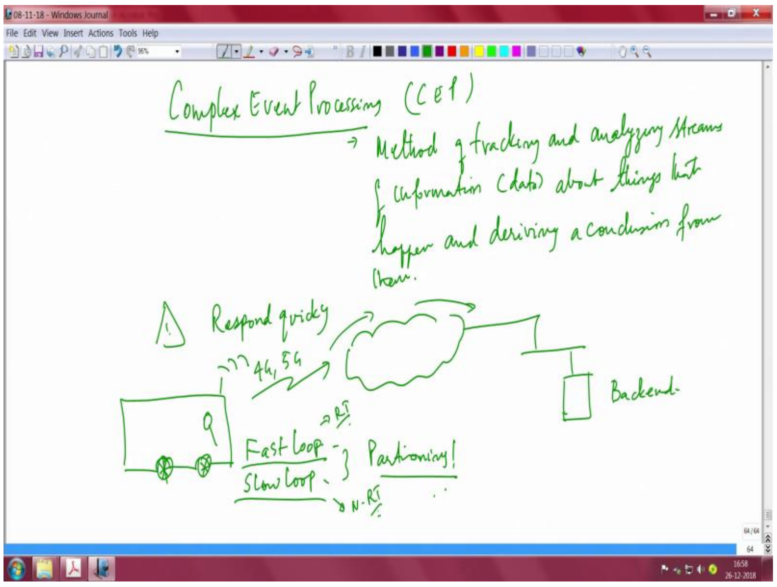
(Refer Slide Time: 00:27)



This is our piece of hardware that we have. We mentioned about the OBD II and so on. And, these are the several sensors which are available as part of the engine ECU. Now, where does IoT come in here? That is your immediate question: where do we fit in our IoT related part into this course? To explain that, it is useful to know that these 5 sensors give you 5 different outputs. You should make some sense out of those outputs which are given, and you should be able to say something is malfunctioning, something is incorrect or everything is ok.

Even it can also be that when everything is fine you do not have to do anything. So, even to get to a point where everything is normal or something is anomalous you must be able to process all these signals, combine them in some way, interpret them in some way, and then arrive at something useful. All of that essentially means that you need a framework under which such anomaly detections or event processing actually happens.

(Refer Slide Time: 01:44)

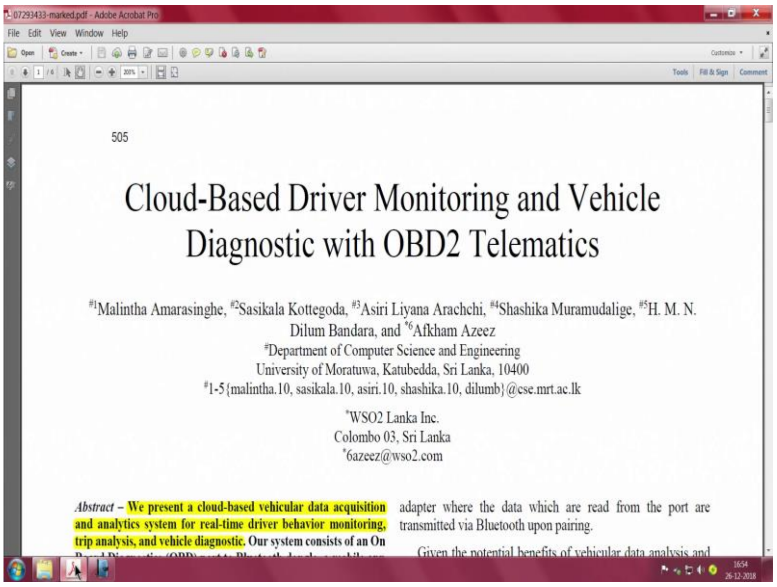

08-11-18 - Windows Journal
File Edit View Insert Actions Tools Help

Complex Event Processing (CEP)

→ Method of tracking and analyzing streams of information (data) about things that happen and deriving a conclusion from them.

For that what they normally do is they use what is known as a Complex Event Processor it is called CEP. Complex event processing is a method of tracking and analysing streams of data or streams of information. In other words, it is mostly data which is coming from the sensors about things that happen and deriving a conclusion from them. So, you need a CEP system, a framework under which CEP is possible.

(Refer Slide Time: 02:50)


07293433-marked.pdf - Adobe Acrobat Pro
File Edit View Window Help

505

Cloud-Based Driver Monitoring and Vehicle Diagnostic with OBD2 Telematics

#1Malintha Amarasinghe, #2Sasikala Kottegoda, #3Asiri Liyana Arachchi, #4Shashika Muramudalige, #5H. M. N. Dilum Bandara, and *6Afkham Azeez

#Department of Computer Science and Engineering
University of Moratuwa, Katubedda, Sri Lanka, 10400
#1-5{malintha.10, sasikala.10, asiri.10, shashika.10, dilumb}@cse.mrt.ac.lk

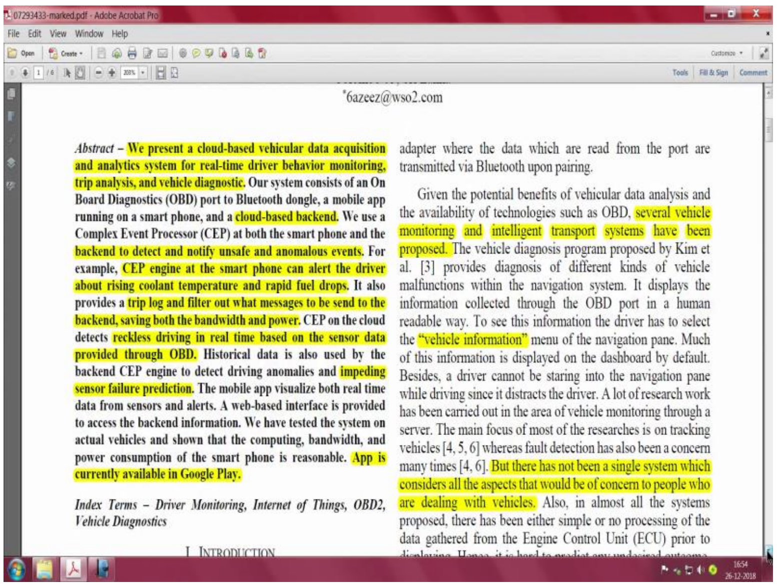
*WSO2 Lanka Inc.
Colombo 03, Sri Lanka
*6azeez@wso2.com

Abstract – **We present a cloud-based vehicular data acquisition and analytics system for real-time driver behavior monitoring, trip analysis, and vehicle diagnostic. Our system consists of an On**

adapter where the data which are read from the port are transmitted via Bluetooth upon pairing.

I will go through this paper for you and this is a nice paper which sort of summarises where IoT can be a big driver. So, this is a paper on “Cloud based driver monitoring and vehicle diagnostic with OBD II telematics". As you know the title is quite obvious, you know everything about OBD II now.

(Refer Slide Time: 03:12)


07293433-marked.pdf - Adobe Acrobat Pro
File Edit View Window Help
Open
Create
Customize
Tools
Fill & Sign
Comment
16:54
26-12-2018

*6azeez@wso2.com

Abstract – We present a cloud-based vehicular data acquisition and analytics system for real-time driver behavior monitoring, trip analysis, and vehicle diagnostic. Our system consists of an On Board Diagnostics (OBD) port to Bluetooth dongle, a mobile app running on a smart phone, and a cloud-based backend. We use a Complex Event Processor (CEP) at both the smart phone and the backend to detect and notify unsafe and anomalous events. For example, CEP engine at the smart phone can alert the driver about rising coolant temperature and rapid fuel drops. It also provides a trip log and filter out what messages to be send to the backend, saving both the bandwidth and power. CEP on the cloud detects reckless driving in real time based on the sensor data provided through OBD. Historical data is also used by the backend CEP engine to detect driving anomalies and impeding sensor failure prediction. The mobile app visualize both real time data from sensors and alerts. A web-based interface is provided to access the backend information. We have tested the system on actual vehicles and shown that the computing, bandwidth, and power consumption of the smart phone is reasonable. App is currently available in Google Play.

Index Terms – Driver Monitoring, Internet of Things, OBD2, Vehicle Diagnostics

I. INTRODUCTION

adapter where the data which are read from the port are transmitted via Bluetooth upon pairing.

Given the potential benefits of vehicular data analysis and the availability of technologies such as OBD, several vehicle monitoring and intelligent transport systems have been proposed. The vehicle diagnosis program proposed by Kim et al. [3] provides diagnosis of different kinds of vehicle malfunctions within the navigation system. It displays the information collected through the OBD port in a human readable way. To see this information the driver has to select the "vehicle information" menu of the navigation pane. Much of this information is displayed on the dashboard by default. Besides, a driver cannot be staring into the navigation pane while driving since it distracts the driver. A lot of research work has been carried out in the area of vehicle monitoring through a server. The main focus of most of the researches is on tracking vehicles [4, 5, 6] whereas fault detection has also been a concern many times [4, 6]. But there has not been a single system which considers all the aspects that would be of concern to people who are dealing with vehicles. Also, in almost all the systems proposed, there has been either simple or no processing of the data gathered from the Engine Control Unit (ECU) prior to

So, let us look up this paper. They present a cloud based vehicular data acquisition and analytics for real time driver behaviour monitoring trip analysis. What is important here is which you must note, you have a vehicle (quickly) this is the vehicle connecting to the cloud and there is a server which is in the backend.

Whatever decisions that have to be taken quickly are taken in what is known as fast loop. For example, the fact that you want to check an anomaly of a sensor which can be done over a longer time period, that you can push it to the backend if you want to do something there. But, if you have to take a decision with respect to some driver behaviour. Driver behaviour, which can be disastrous to the vehicle and occupants who was sitting in the vehicle at that point in time, that cannot happen over a backend system, that would not work.

So, what should be taken into fast loop and what should be taken into slow loop is the question. Data is being generated inside this vehicle and what should be processed right then and there, and what can be pushed to the cloud for processing and the backend is really a question. And, this slow loop - fast loop partitioning is very critical. And, you must learn the trick of thinking of what should be done in fast loop or slow loop.

So, please note algorithms that you design, the systems that you put in place, will all be with respect to the fast loop and the slow loop algorithms, which will have to be done. Fast loop means indeed real time and if you say slow loop it can be offline. So, it need not be real time. So, you can say real time and near real time. It is something that you can take an action when the bus or the vehicle reaches the nearest service station you may have to show it up, it is not urgent at the moment, but it is required to be done immediately on arriving at a particular spot.

Think of what a pilot and the first officer would have to do when there is smoke inside a plane: that data that is gathered by the set of sensors inside a plane. They cannot be waiting for a command from the backend what should be done after analysing the data from the sensors. The pilots have to take an action, they have to either land somewhere in nearest airport get the passengers out from the plane.

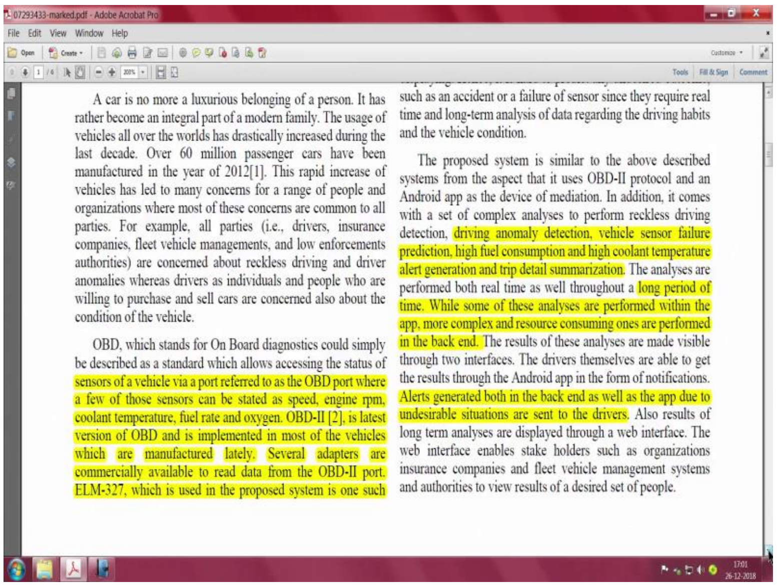
So, that is a fast action and a fast loop action which humans have to themselves have to perform. But if you look at autonomous vehicles, there are not going to be any more humans in the loop, which means data and sensors and the processors and the intelligence and the CPUs have to take this decision as though they were like humans taking the decision at high speed and in real time. So, therefore, real timeliness is an important thing.

So, this paper is actually focusing on that aspect of doing something in real time and putting something back in slow loop. So, CEP is split into 2 parts: there is a CEP running on the vehicle like for instance, you see CEP engine at the smart phone like it is inside the vehicle. And, then there is a CEP on the cloud as well. CEP on the cloud detects reckless driving in real time, based on sensor data provided through OBD. And, CEP engine at the smart phone can alert the driver about rising coolant temperature and rapid fuel drops, which is something that has to be done right then and there.

If a driver is reckless, you can report about a driver's bad behaviour to the owner of your taxi, after you get off. But this is something that you cannot afford right: rising coolant temperature and rapid fuel drop. So, you can see it is all about fast loops and slow loops. Also the fact that sensors failure prediction is an important aspect of this whole paper and they talk about IOT sensors failing and therefore; predicting that there is a failure is also an important thing.

But, you know failure prediction is typically done through take data and do some form of regression analysis and that is what this paper is also talking about. And, interestingly this there is an app which you can download from Google play and I did play with it.

(Refer Slide Time: 09:56)


07293433-marked.pdf - Adobe Acrobat Pro
File Edit View Window Help
Open Create
Customize
Tools Fill & Sign Comment
17:01
26-12-2018

A car is no more a luxurious belonging of a person. It has rather become an integral part of a modern family. The usage of vehicles all over the worlds has drastically increased during the last decade. Over 60 million passenger cars have been manufactured in the year of 2012[1]. This rapid increase of vehicles has led to many concerns for a range of people and organizations where most of these concerns are common to all parties. For example, all parties (i.e., drivers, insurance companies, fleet vehicle managements, and low enforcements authorities) are concerned about reckless driving and driver anomalies whereas drivers as individuals and people who are willing to purchase and sell cars are concerned also about the condition of the vehicle.

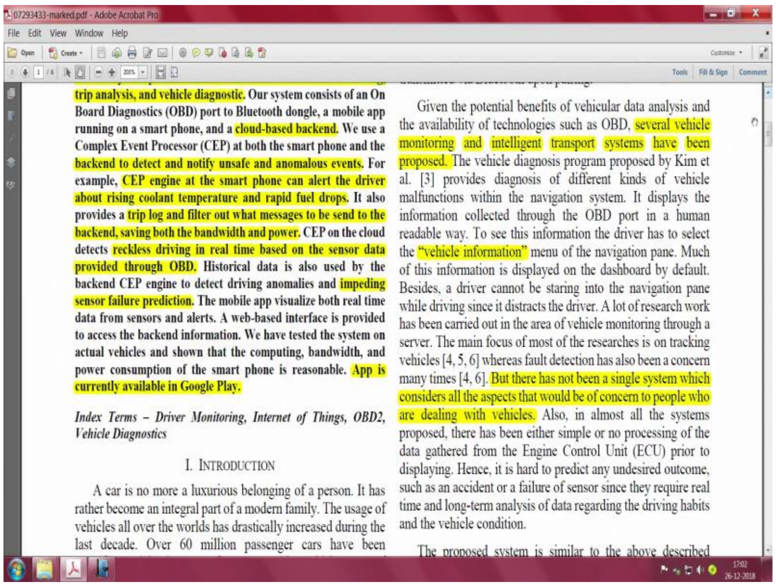
OBD, which stands for On Board diagnostics could simply be described as a standard which allows accessing the status of sensors of a vehicle via a port referred to as the OBD port where a few of those sensors can be stated as speed, engine rpm, coolant temperature, fuel rate and oxygen. OBD-II [2], is latest version of OBD and is implemented in most of the vehicles which are manufactured lately. Several adapters are commercially available to read data from the OBD-II port. ELM-327, which is used in the proposed system is one such

such as an accident or a failure of sensor since they require real time and long-term analysis of data regarding the driving habits and the vehicle condition.

The proposed system is similar to the above described systems from the aspect that it uses OBD-II protocol and an Android app as the device of mediation. In addition, it comes with a set of complex analyses to perform reckless driving detection, driving anomaly detection, vehicle sensor failure prediction, high fuel consumption and high coolant temperature alert generation and trip detail summarization. The analyses are performed both real time as well throughout a long period of time. While some of these analyses are performed within the app, more complex and resource consuming ones are performed in the back end. The results of these analyses are made visible through two interfaces. The drivers themselves are able to get the results through the Android app in the form of notifications. Alerts generated both in the back end as well as the app due to undesirable situations are sent to the drivers. Also results of long term analyses are displayed through a web interface. The web interface enables stake holders such as organizations insurance companies and fleet vehicle management systems and authorities to view results of a desired set of people.

So, now let us look at what this paper is saying in detail. OBD II is giving you ECU related parameters. This whole paper is about ECU related. So, it is all about speed, engine RPM, coolant temperature, fuel rate, and oxygen sensor. Almost the same set of sensors that we have with us in our lab here, the ECU Sim hardware that we have. And, we can skip this part because this is talking about OBD II protocol and all that. So, we can leave that.

(Refer Slide Time: 09:43)


07293433-marked.pdf - Adobe Acrobat Pro
File Edit View Window Help
Open
Create
Customize
Tools
Fill & Sign
Comment
17:02
26-12-2018

trip analysis, and vehicle diagnostic. Our system consists of an On Board Diagnostics (OBD) port to Bluetooth dongle, a mobile app running on a smart phone, and a cloud-based backend. We use a Complex Event Processor (CEP) at both the smart phone and the backend to detect and notify unsafe and anomalous events. For example, CEP engine at the smart phone can alert the driver about rising coolant temperature and rapid fuel drops. It also provides a trip log and filter out what messages to be send to the backend, saving both the bandwidth and power. CEP on the cloud detects reckless driving in real time based on the sensor data provided through OBD. Historical data is also used by the backend CEP engine to detect driving anomalies and impeding sensor failure prediction. The mobile app visualize both real time data from sensors and alerts. A web-based interface is provided to access the backend information. We have tested the system on actual vehicles and shown that the computing, bandwidth, and power consumption of the smart phone is reasonable. App is currently available in Google Play.

Index Terms – Driver Monitoring, Internet of Things, OBD2, Vehicle Diagnostics

I. INTRODUCTION

A car is no more a luxurious belonging of a person. It has rather become an integral part of a modern family. The usage of vehicles all over the worlds has drastically increased during the last decade. Over 60 million passenger cars have been

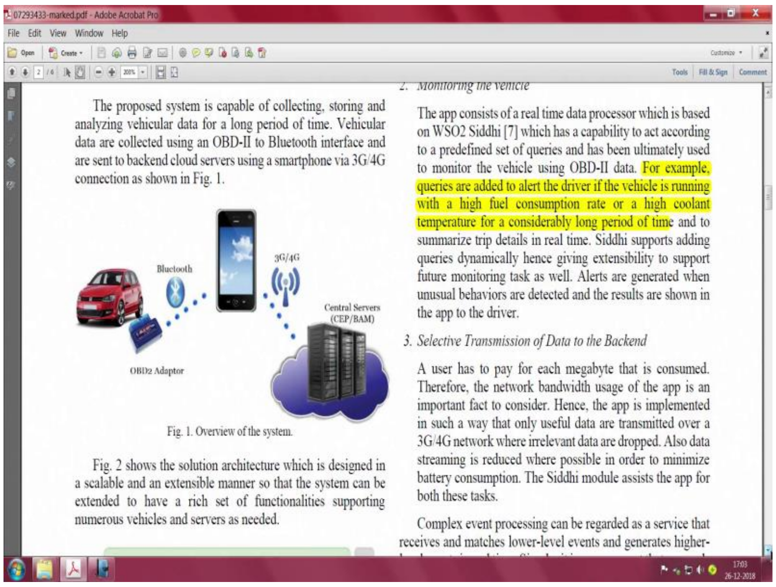
Given the potential benefits of vehicular data analysis and the availability of technologies such as OBD, several vehicle monitoring and intelligent transport systems have been proposed. The vehicle diagnosis program proposed by Kim et al. [3] provides diagnosis of different kinds of vehicle malfunctions within the navigation system. It displays the information collected through the OBD port in a human readable way. To see this information the driver has to select the "vehicle information" menu of the navigation pane. Much of this information is displayed on the dashboard by default. Besides, a driver cannot be staring into the navigation pane while driving since it distracts the driver. A lot of research work has been carried out in the area of vehicle monitoring through a server. The main focus of most of the researches is on tracking vehicles [4, 5, 6] whereas fault detection has also been a concern many times [4, 6]. But there has not been a single system which considers all the aspects that would be of concern to people who are dealing with vehicles. Also, in almost all the systems proposed, there has been either simple or no processing of the data gathered from the Engine Control Unit (ECU) prior to displaying. Hence, it is hard to predict any undesired outcome, such as an accident or a failure of sensor since they require real time and long-term analysis of data regarding the driving habits and the vehicle condition.

The proposed system is similar to the above described

So, now let us go down to the actual part. This paper is actually trying to say that there is a gap in existing literature and this is a gap that they are actually trying to fill. Where there is information about vehicle, there is some scattered information about several aspects. Research is being carried out in the area of vehicle monitoring, but fault detection is something that people have not really looked at. So, this is a single system which not only does vehicle monitoring, but also can do fault detection.

Apart from that, this is also talking about reckless driving; driving anomaly, vehicle sensor failure prediction, high fuel consumption, high coolant temperature alert generation and trip detail and so on. So, if you read this paper you will see that these are handled separately each one has a section by itself. And, alerts are generated both in the back end as well as in the app which is held by the driver or some occupant inside the vehicle.

(Refer Slide Time: 10:38)


07293433-marked.pdf - Adobe Acrobat Pro
File Edit View Window Help
Open
Create
Customize
Tools
Fill & Sign
Comment
Bluetooth
3G/4G
Central Servers (CEP/BAM)
OBD2 Adaptor
17:03
26-12-2018

The proposed system is capable of collecting, storing and analyzing vehicular data for a long period of time. Vehicular data are collected using an OBD-II to Bluetooth interface and are sent to backend cloud servers using a smartphone via 3G/4G connection as shown in Fig. 1.

Fig. 1. Overview of the system.

Fig. 2 shows the solution architecture which is designed in a scalable and an extensible manner so that the system can be extended to have a rich set of functionalities supporting numerous vehicles and servers as needed.

2. *Monitoring the Vehicle*

The app consists of a real time data processor which is based on WSO2 Siddhi [7] which has a capability to act according to a predefined set of queries and has been ultimately used to monitor the vehicle using OBD-II data. For example, queries are added to alert the driver if the vehicle is running with a high fuel consumption rate or a high coolant temperature for a considerably long period of time and to summarize trip details in real time. Siddhi supports adding queries dynamically hence giving extensibility to support future monitoring task as well. Alerts are generated when unusual behaviors are detected and the results are shown in the app to the driver.

3. *Selective Transmission of Data to the Backend*

A user has to pay for each megabyte that is consumed. Therefore, the network bandwidth usage of the app is an important fact to consider. Hence, the app is implemented in such a way that only useful data are transmitted over a 3G/4G network where irrelevant data are dropped. Also data streaming is reduced where possible in order to minimize battery consumption. The Siddhi module assists the app for both these tasks.

Complex event processing can be regarded as a service that receives and matches lower-level events and generates higher-

So, that is the key thing and they have an architectural here. This is nice, is simple to understand: your data coming in from OBD II and then you upload it to use this as a sort of a gateway system, where also the CEP algorithms are running.

CEP can run here on this phone, right. And, whatever your partition does something that should go to the back end, user a 3G link from the phone and then put it to the CEP BAM servers; Business Activity Monitor as it is called so. This is a nice framework under which the authors have put this framework in place, alright.

(Refer Slide Time: 11:16)

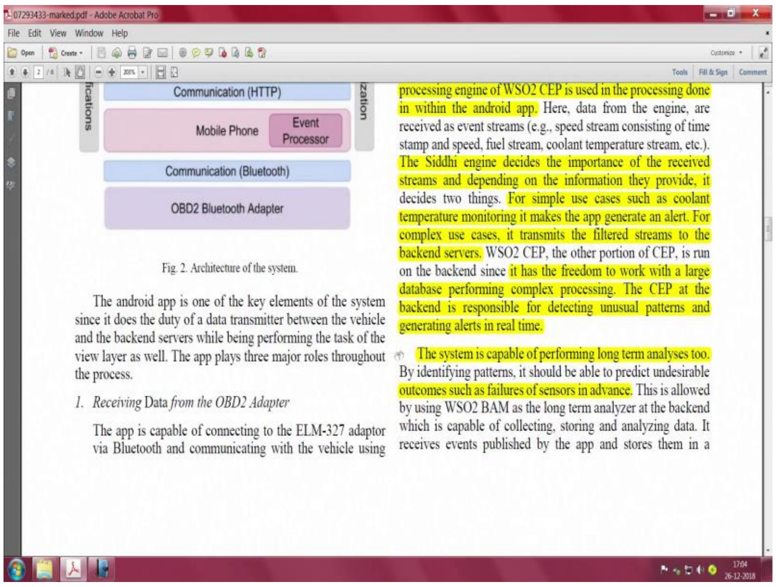


Fig. 2. Architecture of the system.

The android app is one of the key elements of the system since it does the duty of a data transmitter between the vehicle and the backend servers while being performing the task of the view layer as well. The app plays three major roles throughout the process.

1. *Receiving* Data *from the OBD2 Adapter*

The app is capable of connecting to the ELM-327 adaptor via Bluetooth and communicating with the vehicle using

processing engine of WSO2 CEP is used in the processing done in within the android app. Here, data from the engine, are received as event streams (e.g., speed stream consisting of time stamp and speed, fuel stream, coolant temperature stream, etc.). The Siddhi engine decides the importance of the received streams and depending on the information they provide, it decides two things. For simple use cases such as coolant temperature monitoring it makes the app generate an alert. For complex use cases, it transmits the filtered streams to the backend servers. WSO2 CEP, the other portion of CEP, is run on the backend since it has the freedom to work with a large database performing complex processing. The CEP at the backend is responsible for detecting unusual patterns and generating alerts in real time.

The system is capable of performing long term analyses too. By identifying patterns, it should be able to predict undesirable outcomes such as failures of sensors in advance. This is allowed by using WSO2 BAM as the long term analyzer at the backend which is capable of collecting, storing and analyzing data. It receives events published by the app and stores them in a

So, how do you receive data from the OBD II: this is mentioned here, parameter id is we already know, you want to monitor the vehicle now. So, what you do is the app consists of real time data processor, which is based on some framework called Siddhi, WSO2 Siddhi, which has the capability to act according to predefined set of queries and has been ultimately used to monitor the vehicle using OBD II.

You can give queries which are added to alert the driver if the vehicle is running with high fuel consumption or high coolant temperature for a considerably long period of time. I mentioned to you many times that getting this information in isolation is not going to help, but if you really feed this information to the cloud along with the location information of the vehicle, then it is a lot more valuable. Time, date, and location information along with this high fuel consumption, and high coolant temperature adds a lot more value.

You actually know where the vehicle is, what the terrain is and what the condition is under which the system has this problem. In the airplanes also it is a same: a bunch of sensors which are connected inside the plane are all recorded on a flight data recorder.

This is called the FDR system. Then, you have another system which is called the cockpit voice recorder. This is just capturing the conversations between the captain and the first officer, or any announcements. I mean whatever is happening in the cockpit: if the pilot is making an announcement to the cab inside that part is also captured.

So, there is conversations part and then there is actual sensor data part, both are separately recorded on separate systems and you can have redundant boxes. So, that even if you know for some reason the some part of the plane, you cannot get it from the wreckage, the wreckage at least from the other part of the plane may be somewhere it is kept and then you can pull it out.

So, it simultaneously recorded on more than one system. So, that it is easy for recovery and for diagnosis and so on. Same the thing here you will have to add that part of the system of location information. So, that it adds value to whatever is being said.

(Refer Slide Time: 13:40)

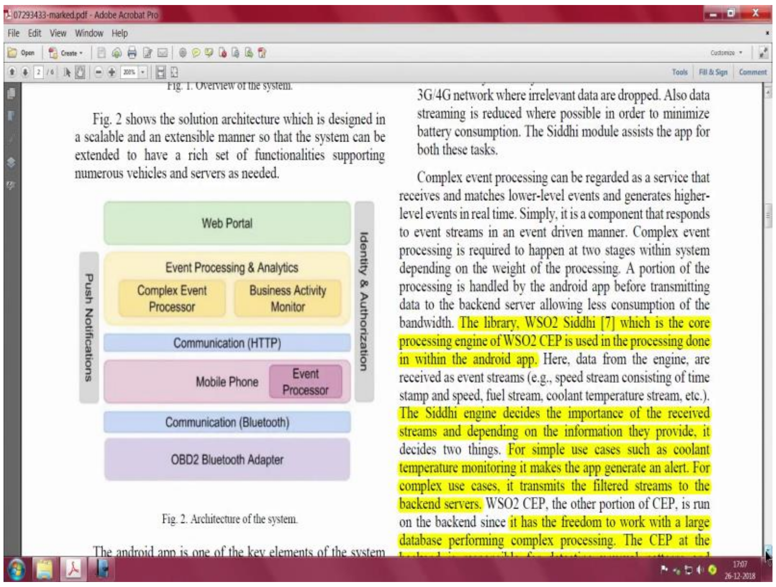


Fig. 1. Overview of the system.

Fig. 2 shows the solution architecture which is designed in a scalable and an extensible manner so that the system can be extended to have a rich set of functionalities supporting numerous vehicles and servers as needed.

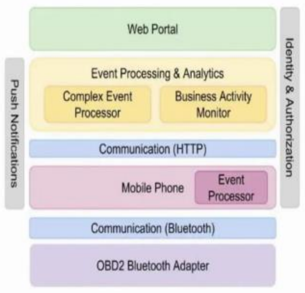


Fig. 2. Architecture of the system.

The android app is one of the key elements of the system

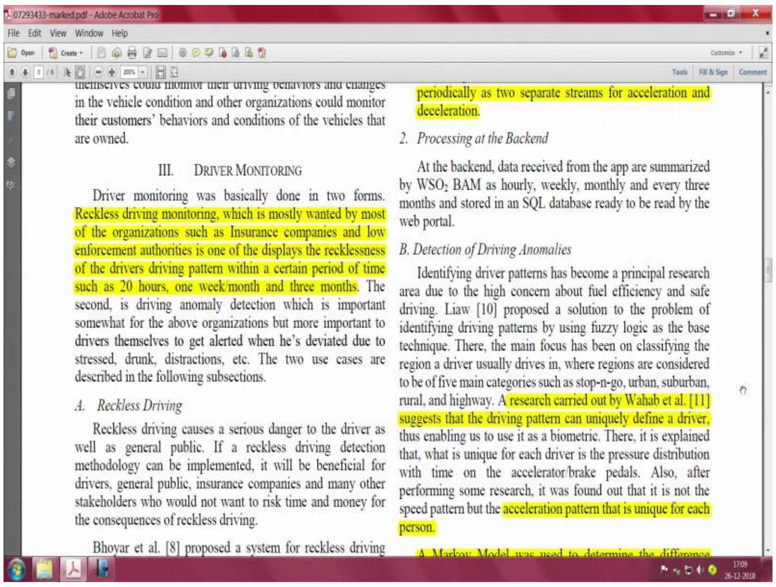
3G/4G network where irrelevant data are dropped. Also data streaming is reduced where possible in order to minimize battery consumption. The Siddhi module assists the app for both these tasks.

Complex event processing can be regarded as a service that receives and matches lower-level events and generates higher-level events in real time. Simply, it is a component that responds to event streams in an event driven manner. Complex event processing is required to happen at two stages within system depending on the weight of the processing. A portion of the processing is handled by the android app before transmitting data to the backend server allowing less consumption of the bandwidth. The library, WSO2 Siddhi [7] which is the core processing engine of WSO2 CEP is used in the processing done in within the android app. Here, data from the engine, are received as event streams (e.g., speed stream consisting of time stamp and speed, fuel stream, coolant temperature stream, etc.). The Siddhi engine decides the importance of the received streams and depending on the information they provide, it decides two things. For simple use cases such as coolant temperature monitoring it makes the app generate an alert. For complex use cases, it transmits the filtered streams to the backend servers. WSO2 CEP, the other portion of CEP, is run on the backend since it has the freedom to work with a large database performing complex processing. The CEP at the

So, complex event processing can be regarded as a service and you can see that the portion of processing is handled by the Android app. And, before transmitting data; that means, fast looping is done first and then the backend server allowing less consumption of bandwidth. There is a library WSO2 Siddhi which is the core processing engine and used for processing this. The Siddhi engine decides the importance of the received streams and depending on the information they provide it decides two things. So, simple cases: coolant temperature monitoring, it generates an alert right on the app, and for complex cases it seems that there are many more events happening, then you transmit the filter streams to the backend servers. And, let the backend figure out, because it does not appear to be based on just one single isolated sensor measurement, which is just coolant temperature; as remaining parameters are all fine then you have no way of concluding what the problem is. So, it has to give it to the backend.

So, the other portion of the CEP running on the backend has the freedom to work with large database. Performing complex processing may be a simple. Analysis that is happening on the phone is insufficient, it look at some unusual patterns and then generate alerts. So, all of that is captured in the backend part. So, this is one thing, then the whole framework also has the ability to do this very critical thing of performing long term analysis. It should be able to predict undesirable outcome such as failures of sensors well in advance and so, this paper goes into those details.

(Refer Slide Time: 15:53)



themselves could monitor their driving behaviors and changes in the vehicle condition and other organizations could monitor their customers' behaviors and conditions of the vehicles that are owned.

III. DRIVER MONITORING

Driver monitoring was basically done in two forms. Reckless driving monitoring, which is mostly wanted by most of the organizations such as Insurance companies and low enforcement authorities is one of the displays the recklessness of the drivers driving pattern within a certain period of time such as 20 hours, one week/month and three months. The second, is driving anomaly detection which is important somewhat for the above organizations but more important to drivers themselves to get alerted when he's deviated due to stressed, drunk, distractions, etc. The two use cases are described in the following subsections.

A. Reckless Driving

Reckless driving causes a serious danger to the driver as well as general public. If a reckless driving detection methodology can be implemented, it will be beneficial for drivers, general public, insurance companies and many other stakeholders who would not want to risk time and money for the consequences of reckless driving.

Bhoyar et al. [8] proposed a system for reckless driving

periodically as two separate streams for acceleration and deceleration.

2. Processing at the Backend

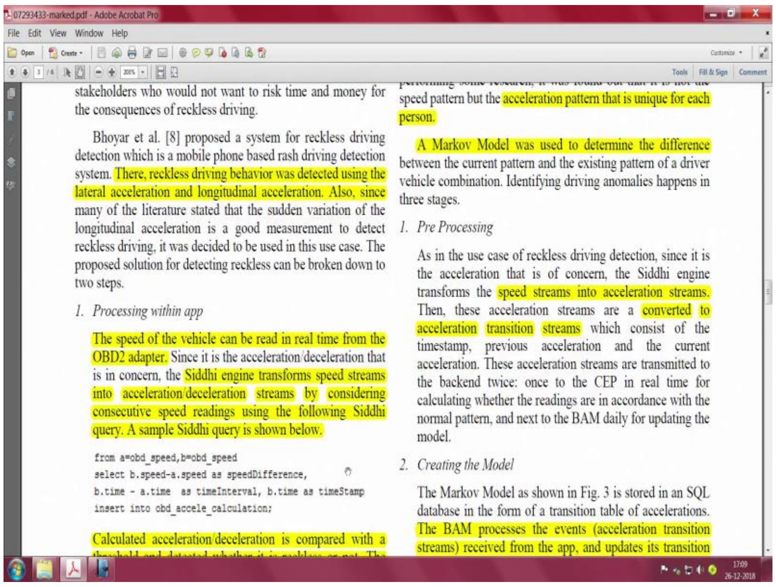
At the backend, data received from the app are summarized by WSO$_2$ BAM as hourly, weekly, monthly and every three months and stored in an SQL database ready to be read by the web portal.

B. Detection of Driving Anomalies

Identifying driver patterns has become a principal research area due to the high concern about fuel efficiency and safe driving. Liaw [10] proposed a solution to the problem of identifying driving patterns by using fuzzy logic as the base technique. There, the main focus has been on classifying the region a driver usually drives in, where regions are considered to be of five main categories such as stop-n-go, urban, suburban, rural, and highway. A research carried out by Wahab et al. [11] suggests that the driving pattern can uniquely define a driver, thus enabling us to use it as a biometric. There, it is explained that, what is unique for each driver is the pressure distribution with time on the accelerator/brake pedals. Also, after performing some research, it was found out that it is not the speed pattern but the acceleration pattern that is unique for each person.

It does something about driver monitoring which is an important thing.

(Refer Slide Time: 15:28)



stakeholders who would not want to risk time and money for the consequences of reckless driving.

Bhoyar et al. [8] proposed a system for reckless driving detection which is a mobile phone based rash driving detection system. There, reckless driving behavior was detected using the lateral acceleration and longitudinal acceleration. Also, since many of the literature stated that the sudden variation of the longitudinal acceleration is a good measurement to detect reckless driving, it was decided to be used in this use case. The proposed solution for detecting reckless can be broken down to two steps.

1. Processing within app

The speed of the vehicle can be read in real time from the OBD2 adapter. Since it is the acceleration/deceleration that is in concern, the Siddhi engine transforms speed streams into acceleration/deceleration streams by considering consecutive speed readings using the following Siddhi query. A sample Siddhi query is shown below.

```
from a=obd_speed,b=obd_speed
select b.speed-a.speed as speedDifference,
b.time - a.time  as timeInterval, b.time as timeStamp
insert into obd_accele_calculation;
```

Calculated acceleration/deceleration is compared with a

speed pattern but the acceleration pattern that is unique for each person.

A Markov Model was used to determine the difference between the current pattern and the existing pattern of a driver vehicle combination. Identifying driving anomalies happens in three stages.

1. Pre Processing

As in the use case of reckless driving detection, since it is the acceleration that is of concern, the Siddhi engine transforms the speed streams into acceleration streams. Then, these acceleration streams are a converted to acceleration transition streams which consist of the timestamp, previous acceleration and the current acceleration. These acceleration streams are transmitted to the backend twice: once to the CEP in real time for calculating whether the readings are in accordance with the normal pattern, and next to the BAM daily for updating the model.

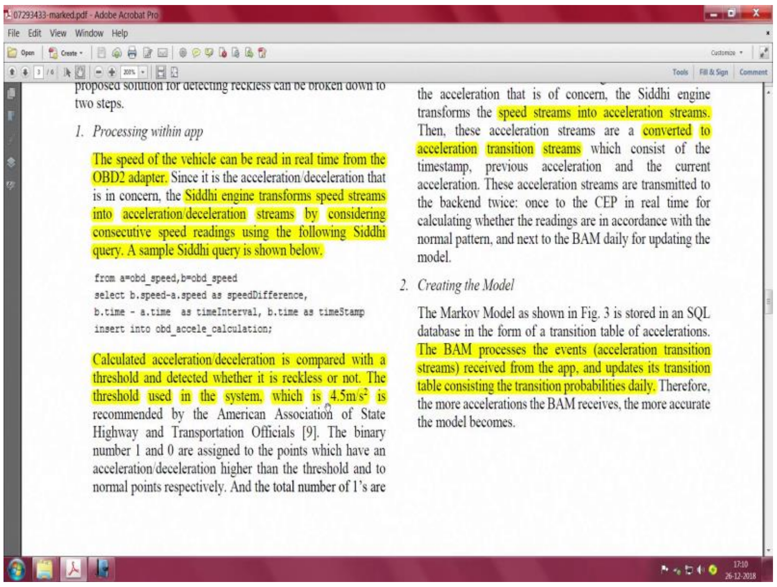
2. Creating the Model

The Markov Model as shown in Fig. 3 is stored in an SQL database in the form of a transition table of accelerations. The BAM processes the events (acceleration transition streams) received from the app, and updates its transition

And, it captures reckless driving by using lateral acceleration and longitudinal acceleration. Any sudden variation of the longitudinal acceleration is a good measurement to detect reckless driving and it was decided to use in this case.

How to set it up processing with within the app? This is mentioned here. How do you get acceleration lateral acceleration and longitudinal acceleration? This you can get from the speed itself, you see Siddhi engine transform speed streams into acceleration deceleration streams, by considering consecutive speed readings using the following Siddhi query. So, you actually generate reckless driving behaviour from the speed measurement itself.

(Refer Slide Time: 16:18)


07293433-marked.pdf - Adobe Acrobat Pro
File Edit View Window Help
Open
Create
Customize
Tools
Fill & Sign
Comment

proposed solution for detecting reckless can be broken down to two steps.

1. Processing within app

The speed of the vehicle can be read in real time from the OBD2 adapter. Since it is the acceleration/deceleration that is in concern, the Siddhi engine transforms speed streams into acceleration/deceleration streams by considering consecutive speed readings using the following Siddhi query. A sample Siddhi query is shown below.

```
from a=obd_speed,b=obd_speed
select b.speed-a.speed as speedDifference,
b.time - a.time  as timeInterval, b.time as timeStamp
insert into obd_accele_calculation;
```

Calculated acceleration/deceleration is compared with a threshold and detected whether it is reckless or not. The threshold used in the system, which is $4.5 m/s^2$ is recommended by the American Association of State Highway and Transportation Officials [9]. The binary number 1 and 0 are assigned to the points which have an acceleration/deceleration higher than the threshold and to normal points respectively. And the total number of 1's are

the acceleration that is of concern, the Siddhi engine transforms the speed streams into acceleration streams. Then, these acceleration streams are a converted to acceleration transition streams which consist of the timestamp, previous acceleration and the current acceleration. These acceleration streams are transmitted to the backend twice: once to the CEP in real time for calculating whether the readings are in accordance with the normal pattern, and next to the BAM daily for updating the model.

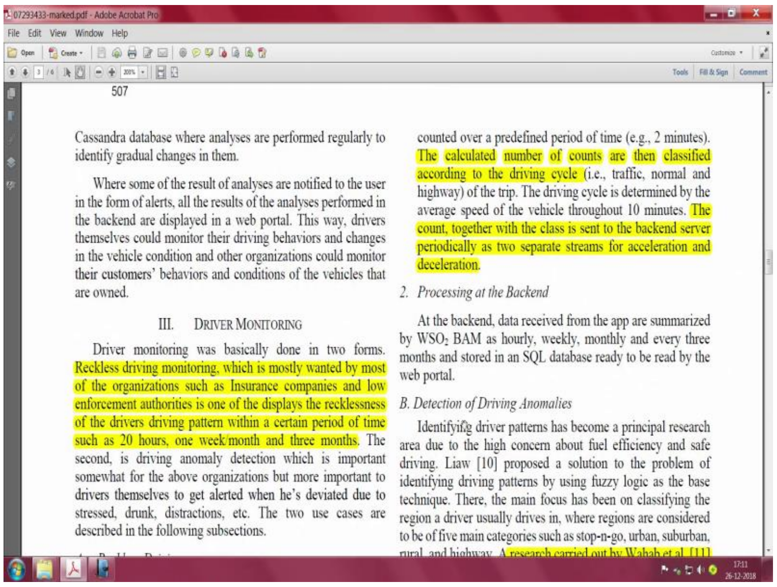
2. Creating the Model

The Markov Model as shown in Fig. 3 is stored in an SQL database in the form of a transition table of accelerations. The BAM processes the events (acceleration transition streams) received from the app, and updates its transition table consisting the transition probabilities daily. Therefore, the more accelerations the BAM receives, the more accurate the model becomes.

What should be the acceleration, deceleration and what should be the threshold? Where should you set it? When the threshold should be set it 4.5 m/s^2 and this is the standard, you cannot accelerate greater than this.

So, if you do that, you have actually violated the safety norm. So, what is a clear indicator, that the driver is really reckless and he is going higher than this. So, what you do is you get it over a period of time and you count the number of 1 and 0 assigned to the points, which have an acceleration or deceleration. And, apply the threshold and have acceleration higher than the normal threshold and to normal points respectively and total number of 1's are counted over a predefined time period.

(Refer Slide Time: 17:06)


07293433-marked.pdf - Adobe Acrobat Pro
File Edit View Window Help
Open
Create
Customize
Tools
Fill & Sign
Comment
17:11
26-12-2018

507

Cassandra database where analyses are performed regularly to identify gradual changes in them.

Where some of the result of analyses are notified to the user in the form of alerts, all the results of the analyses performed in the backend are displayed in a web portal. This way, drivers themselves could monitor their driving behaviors and changes in the vehicle condition and other organizations could monitor their customers' behaviors and conditions of the vehicles that are owned.

III. DRIVER MONITORING

Driver monitoring was basically done in two forms. Reckless driving monitoring, which is mostly wanted by most of the organizations such as Insurance companies and low enforcement authorities is one of the displays the recklessness of the drivers driving pattern within a certain period of time such as 20 hours, one week/month and three months. The second, is driving anomaly detection which is important somewhat for the above organizations but more important to drivers themselves to get alerted when he's deviated due to stressed, drunk, distractions, etc. The two use cases are described in the following subsections.

counted over a predefined period of time (e.g., 2 minutes). The calculated number of counts are then classified according to the driving cycle (i.e., traffic, normal and highway) of the trip. The driving cycle is determined by the average speed of the vehicle throughout 10 minutes. The count, together with the class is sent to the backend server periodically as two separate streams for acceleration and deceleration.

2. *Processing at the Backend*

At the backend, data received from the app are summarized by WSO$_2$ BAM as hourly, weekly, monthly and every three months and stored in an SQL database ready to be read by the web portal.

B. Detection of Driving Anomalies

Identifyiñg driver patterns has become a principal research area due to the high concern about fuel efficiency and safe driving. Liaw [10] proposed a solution to the problem of identifying driving patterns by using fuzzy logic as the base technique. There, the main focus has been on classifying the region a driver usually drives in, where regions are considered to be of five main categories such as stop-n-go, urban, suburban, rural and highway. A research carried out by Wahab et al. [11]

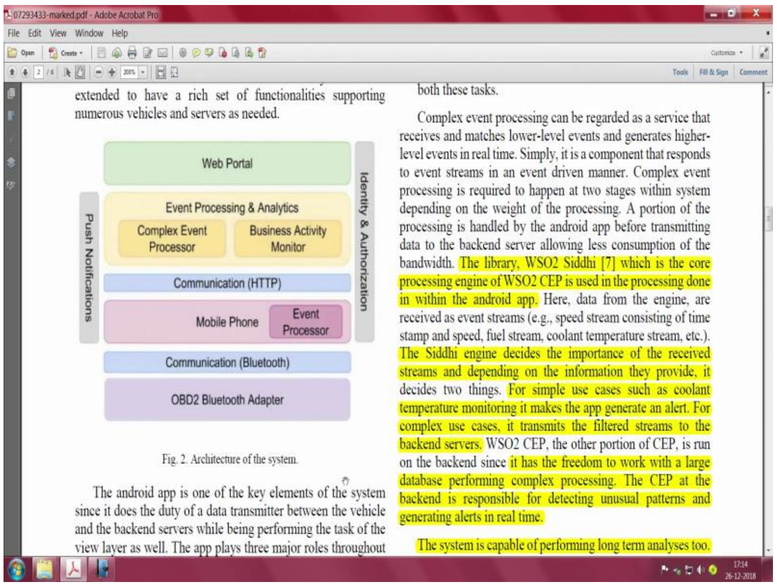
And, by this count you classify according to the driving cycle, whether it is traffic or normal highway. The driving cycle is determined by the average speed of the vehicle throughout 10 minutes: the count together with the class is sent to the backend server periodically to separate streams for acceleration and deceleration.

So, you can see lot of information is being already generated from just the speed related information. And so, very useful data is actually got directly from the speed itself, ok. This is the nice take away from this part of the discussion. Now, what does the backend do: you have to do some processing on the backend. So, backend does get the data and it is summarised. You have the business activity monitor which can be set to hourly, weekly, or monthly.

And there is an SQL database ready to be read by the web portal, it just takes the data and puts it there, alright. Next is a detection of driving anomalies. This is again referred to by another paper, and they give reference of that paper. It is a research area and you can investigate this all by yourselves by looking at several papers in that. The paper here is actually talking about creating a Markov model, which was used to determine the difference between the current pattern and the existing pattern of a driver.

So, you start identifying driving anomalies and put them into 3 different stages, there is a pre-processing stage. So, speed to acceleration streams: from acceleration streams you put it as acceleration transitions streams. So, that essentially means you get the timestamps of previous acceleration and the current acceleration. These streams are then communicated to the backend twice actually. Once to the CEP in real time for calculating whether the readings are in accordance with the normal pattern and next to the BAM daily for updating the model.

(Refer Slide Time: 19:46)



extended to have a rich set of functionalities supporting numerous vehicles and servers as needed.

Fig. 2. Architecture of the system.

The android app is one of the key elements of the system since it does the duty of a data transmitter between the vehicle and the backend servers while being performing the task of the view layer as well. The app plays three major roles throughout both these tasks.

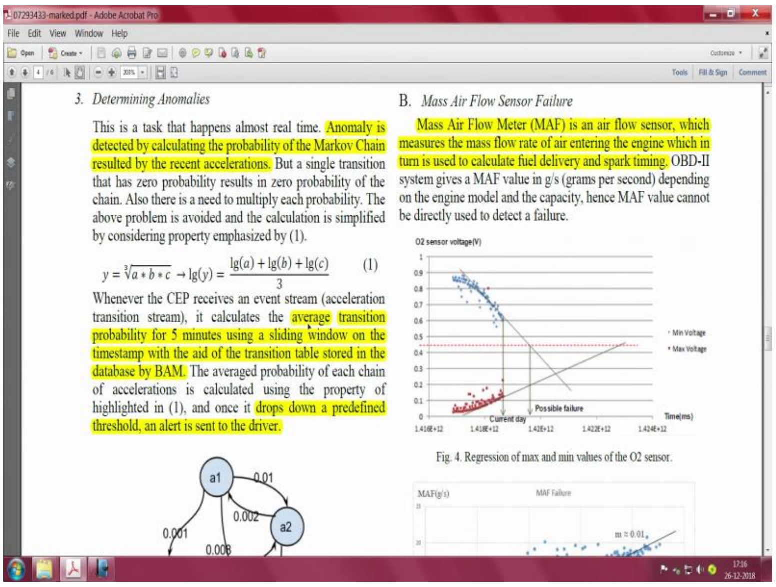
Complex event processing can be regarded as a service that receives and matches lower-level events and generates higher-level events in real time. Simply, it is a component that responds to event streams in an event driven manner. Complex event processing is required to happen at two stages within system depending on the weight of the processing. A portion of the processing is handled by the android app before transmitting data to the backend server allowing less consumption of the bandwidth. The library, WSO2 Siddhi [7] which is the core processing engine of WSO2 CEP is used in the processing done in within the android app. Here, data from the engine, are received as event streams (e.g., speed stream consisting of time stamp and speed, fuel stream, coolant temperature stream, etc.). The Siddhi engine decides the importance of the received streams and depending on the information they provide, it decides two things. For simple use cases such as coolant temperature monitoring it makes the app generate an alert. For complex use cases, it transmits the filtered streams to the backend servers. WSO2 CEP, the other portion of CEP, is run on the backend since it has the freedom to work with a large database performing complex processing. The CEP at the backend is responsible for detecting unusual patterns and generating alerts in real time.

The system is capable of performing long term analyses too.

So, you can see that one is to the CEP and there. So, you can see here that you may have to send it twice. For this model for this block and to that block as well. So, you from there you find out the driving anomaly you do twice transmission.

Once to the CEP in real time for calculating whether the readings are in accordance with the normal pattern, because it is the CEP that captures any anomalies and BAM for updating the model. Creating the Model: what is the model here? The model indeed is a Markov model and that model is stored in some sequential database. The BAM essentially processes the events: acceleration transition streams received from the app and update its transition table consisting of transition probabilities daily. Therefore, more accelerations of the BAM receives, the model becomes much more accurate.

(Refer Slide Time: 20:48)



3. *Determining Anomalies*

This is a task that happens almost real time. Anomaly is detected by calculating the probability of the Markov Chain resulted by the recent accelerations. But a single transition that has zero probability results in zero probability of the chain. Also there is a need to multiply each probability. The above problem is avoided and the calculation is simplified by considering property emphasized by (1).

$$y = \sqrt[3]{a*b*c} \rightarrow \lg(y) = \frac{\lg(a)+\lg(b)+\lg(c)}{3} \qquad (1)$$

Whenever the CEP receives an event stream (acceleration transition stream), it calculates the average transition probability for 5 minutes using a sliding window on the timestamp with the aid of the transition table stored in the database by BAM. The averaged probability of each chain of accelerations is calculated using the property of highlighted in (1), and once it drops down a predefined threshold, an alert is sent to the driver.

B. *Mass Air Flow Sensor Failure*

Mass Air Flow Meter (MAF) is an air flow sensor, which measures the mass flow rate of air entering the engine which in turn is used to calculate fuel delivery and spark timing. OBD-II system gives a MAF value in g/s (grams per second) depending on the engine model and the capacity, hence MAF value cannot be directly used to detect a failure.

Fig. 4. Regression of max and min values of the O2 sensor.

You do the anomaly detection by calculating the probability of Markov chain, resulted by the recent accelerations appears to be quite straight forward.

And, then CEP: after it receives an event that is the acceleration transition stream, it calculates the average transition probability for 5 minutes using a sliding window on the timestamp, with the aid of transition probabilities table stored in the database by BAM. The average probability of each chain of accelerations is calculated using the property highlighted in this expression here. And, once it drops down to a predefined threshold an alert is sent.

(Refer Slide Time: 21:36)

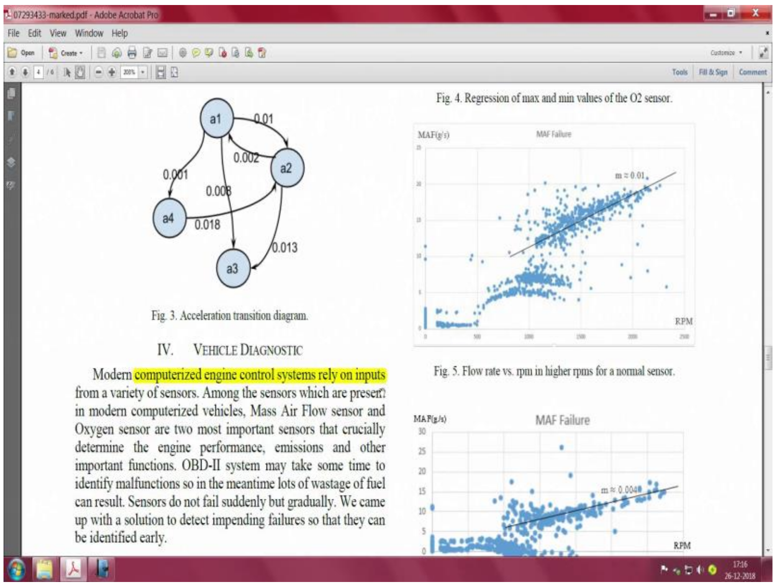


Fig. 4. Regression of max and min values of the O2 sensor.

Fig. 3. Acceleration transition diagram.

IV. VEHICLE DIAGNOSTIC

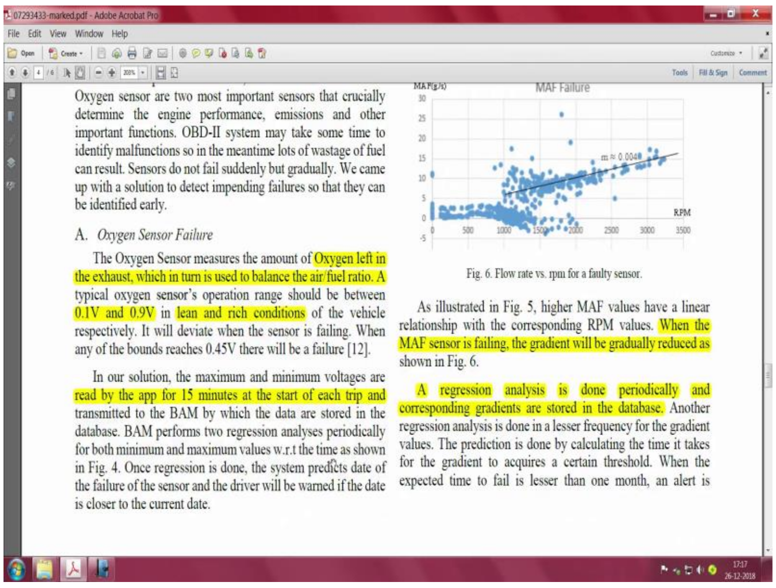
Modern computerized engine control systems rely on inputs from a variety of sensors. Among the sensors which are present in modern computerized vehicles, Mass Air Flow sensor and Oxygen sensor are two most important sensors that crucially determine the engine performance, emissions and other important functions. OBD-II system may take some time to identify malfunctions so in the meantime lots of wastage of fuel can result. Sensors do not fail suddenly but gradually. We came up with a solution to detect impending failures so that they can be identified early.

Fig. 5. Flow rate vs. rpm in higher rpms for a normal sensor.

So, you can see this is the acceleration transition diagram. Similarly, vehicle diagnostic: if you run through you will see that, you can look at oxygen sensor, which is important aspect, which is more like an output from the whole engine manifold system.

Because, you know what is the amount of oxygen that is coming out. If it is a large quantity, then you know that it is unburnt, it is just efficiency problem. And, there is you need to do something about the air to fuel ratio mixing, right.

(Refer Slide Time: 22:21)


07293433-marked.pdf - Adobe Acrobat Pro
File Edit View Window Help
MAF(g/s)
MAF Failure
m ≈ 0.0048
RPM
17:17
26-12-2018

Oxygen sensor are two most important sensors that crucially determine the engine performance, emissions and other important functions. OBD-II system may take some time to identify malfunctions so in the meantime lots of wastage of fuel can result. Sensors do not fail suddenly but gradually. We came up with a solution to detect impending failures so that they can be identified early.

A. *Oxygen Sensor Failure*

The Oxygen Sensor measures the amount of Oxygen left in the exhaust, which in turn is used to balance the air/fuel ratio. A typical oxygen sensor's operation range should be between 0.1V and 0.9V in lean and rich conditions of the vehicle respectively. It will deviate when the sensor is failing. When any of the bounds reaches 0.45V there will be a failure [12].

In our solution, the maximum and minimum voltages are read by the app for 15 minutes at the start of each trip and transmitted to the BAM by which the data are stored in the database. BAM performs two regression analyses periodically for both minimum and maximum values w.r.t the time as shown in Fig. 4. Once regression is done, the system predicts date of the failure of the sensor and the driver will be warned if the date is closer to the current date.

Fig. 6. Flow rate vs. rpm for a faulty sensor.

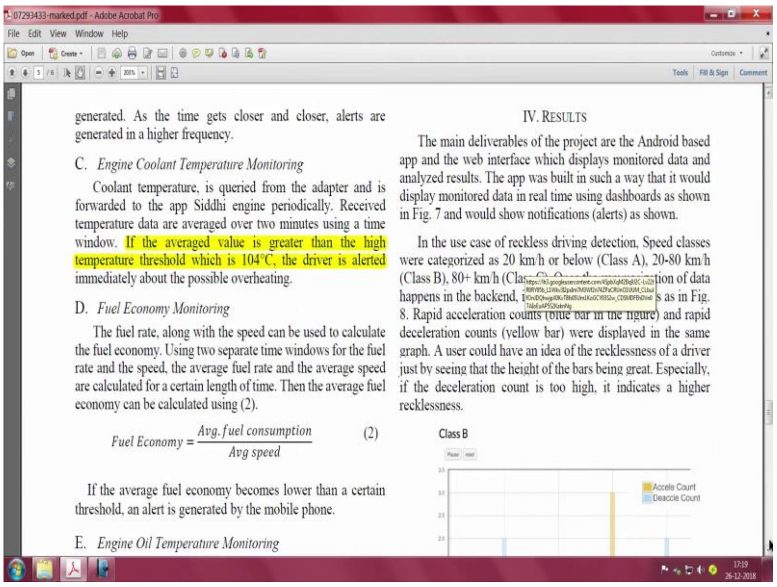
As illustrated in Fig. 5, higher MAF values have a linear relationship with the corresponding RPM values. When the MAF sensor is failing, the gradient will be gradually reduced as shown in Fig. 6.

A regression analysis is done periodically and corresponding gradients are stored in the database. Another regression analysis is done in a lesser frequency for the gradient values. The prediction is done by calculating the time it takes for the gradient to acquires a certain threshold. When the expected time to fail is lesser than one month, an alert is

So, you can do some sort of regression analysis of this oxygen sensor failure. And, you can see that this picture is actually capturing that. It shows what should be the threshold and where exactly the problem comes by simple thresholding. Similarly, you could do with mass air flow sensor as I mentioned to you the MAF sensor is placed between the air filter and the throttle body, and is typically measured in grams per second, and you can see that there is also the MAF: flow rate versus RPM.

In higher RPMs for a normal sensor, this is a normal value and then there is an anomaly, then you should also be able to say what is the faulty sensor :flow rate versus RPM for a faulty sensor. You would get like this pictures are pretty different as you can see. And regression analysis is done periodically and corresponding gradients are stored in the database. So, you can do some very simple things to get to understanding whether there is a problem with respect to the MAF sensor, whether there is a faulty MAF sensor. And, I mentioned to you all big companies have to do something very simple like to predict there is a problem with the MAF sensor.

(Refer Slide Time: 23:59)


07293433-marked.pdf - Adobe Acrobat Pro
File Edit View Window Help
Open
Create
Customize
Tools
Fill & Sign
Comment
17:19
26-12-2018

generated. As the time gets closer and closer, alerts are generated in a higher frequency.

C. *Engine Coolant Temperature Monitoring*

Coolant temperature, is queried from the adapter and is forwarded to the app Siddhi engine periodically. Received temperature data are averaged over two minutes using a time window. If the averaged value is greater than the high temperature threshold which is 104°C, the driver is alerted immediately about the possible overheating.

D. *Fuel Economy Monitoring*

The fuel rate, along with the speed can be used to calculate the fuel economy. Using two separate time windows for the fuel rate and the speed, the average fuel rate and the average speed are calculated for a certain length of time. Then the average fuel economy can be calculated using (2).

$$Fuel\ Economy = \frac{Avg.fuel\ consumption}{Avg\ speed} \qquad (2)$$

If the average fuel economy becomes lower than a certain threshold, an alert is generated by the mobile phone.

E. *Engine Oil Temperature Monitoring*

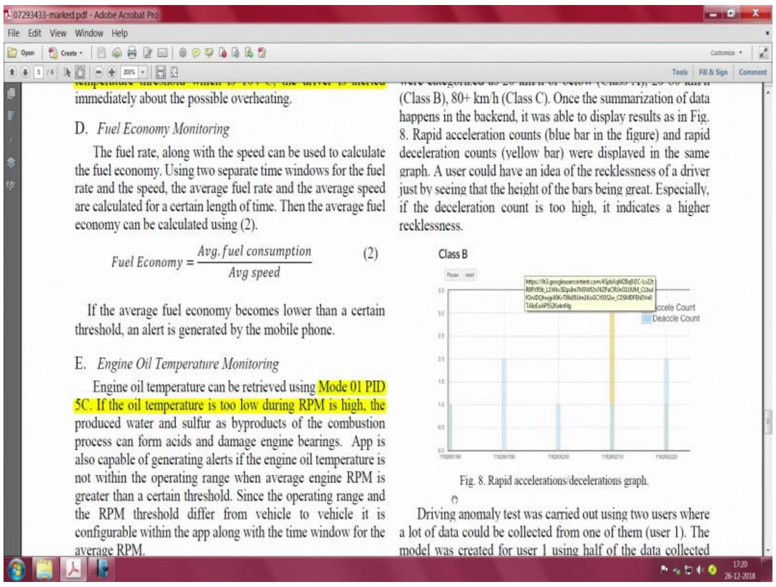
IV. RESULTS

The main deliverables of the project are the Android based app and the web interface which displays monitored data and analyzed results. The app was built in such a way that it would display monitored data in real time using dashboards as shown in Fig. 7 and would show notifications (alerts) as shown.

In the use case of reckless driving detection, Speed classes were categorized as 20 km/h or below (Class A), 20-80 km/h (Class B), 80+ km/h (Cla [illegible] tion of data happens in the backend, i [illegible] s as in Fig. 8. Rapid acceleration counts (blue bar in the figure) and rapid deceleration counts (yellow bar) were displayed in the same graph. A user could have an idea of the recklessness of a driver just by seeing that the height of the bars being great. Especially, if the deceleration count is too high, it indicates a higher recklessness.

Then you will go on to read this you will find that engine coolant is also something you can put a threshold, and then if it crosses the threshold, you mention that the engine coolant temperature has increased.

(Refer Slide Time: 24:18)


07293433-marked.pdf - Adobe Acrobat Pro
File Edit View Window Help
Open
Create
Customize
Tools
Fill & Sign
Comment

immediately about the possible overheating.

D. *Fuel Economy Monitoring*

The fuel rate, along with the speed can be used to calculate the fuel economy. Using two separate time windows for the fuel rate and the speed, the average fuel rate and the average speed are calculated for a certain length of time. Then the average fuel economy can be calculated using (2).

$$Fuel\ Economy = \frac{Avg.fuel\ consumption}{Avg\ speed} \quad (2)$$

If the average fuel economy becomes lower than a certain threshold, an alert is generated by the mobile phone.

E. *Engine Oil Temperature Monitoring*

Engine oil temperature can be retrieved using Mode 01 PID 5C. If the oil temperature is too low during RPM is high, the produced water and sulfur as byproducts of the combustion process can form acids and damage engine bearings. App is also capable of generating alerts if the engine oil temperature is not within the operating range when average engine RPM is greater than a certain threshold. Since the operating range and the RPM threshold differ from vehicle to vehicle it is configurable within the app along with the time window for the average RPM.

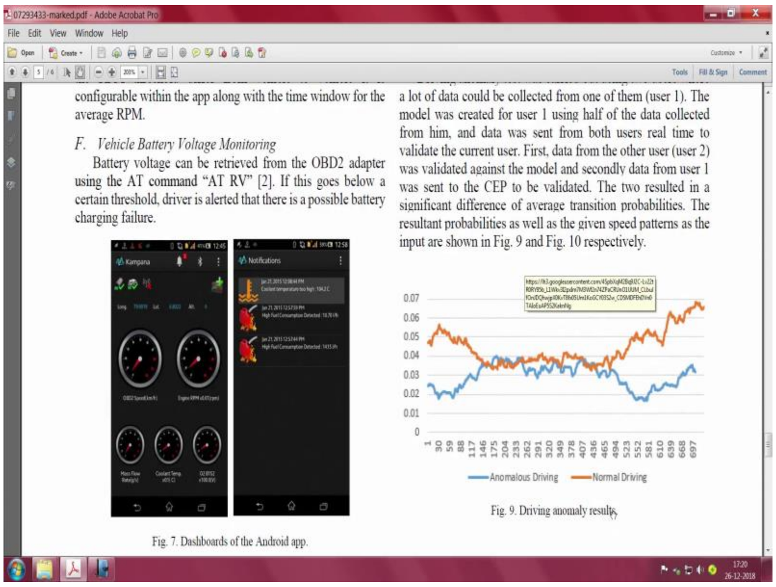
(Class B), 80+ km/h (Class C). Once the summarization of data happens in the backend, it was able to display results as in Fig. 8. Rapid acceleration counts (blue bar in the figure) and rapid deceleration counts (yellow bar) were displayed in the same graph. A user could have an idea of the recklessness of a driver just by seeing that the height of the bars being great. Especially, if the deceleration count is too high, it indicates a higher recklessness.

Fig. 8. Rapid accelerations/decelerations graph.

Driving anomaly test was carried out using two users where a lot of data could be collected from one of them (user 1). The model was created for user 1 using half of the data collected

So, one can go on with oil temperature monitoring you get you see, you get a mode 0 1 PID which is 5 C. So, you can clearly see that this is from the traction or the part when we talk about the power train. The PID is also from the power train and therefore, if oil temperature is low during high RPM, they produce water and sulphur as by-products. So, the combustion process can form acids and damage engine bearings.

(Refer Slide Time: 25:01)


07293433-marked.pdf - Adobe Acrobat Pro
File Edit View Window Help
Open
Create
Customize
Tools
Fill & Sign
Comment

configurable within the app along with the time window for the average RPM.

F. *Vehicle Battery Voltage Monitoring*

Battery voltage can be retrieved from the OBD2 adapter using the AT command "AT RV" [2]. If this goes below a certain threshold, driver is alerted that there is a possible battery charging failure.

Kampana
Notifications
High Fuel Consumption Detected
High Fuel Consumption Detected

Fig. 7. Dashboards of the Android app.

a lot of data could be collected from one of them (user 1). The model was created for user 1 using half of the data collected from him, and data was sent from both users real time to validate the current user. First, data from the other user (user 2) was validated against the model and secondly data from user 1 was sent to the CEP to be validated. The two resulted in a significant difference of average transition probabilities. The resultant probabilities as well as the given speed patterns as the input are shown in Fig. 9 and Fig. 10 respectively.

Fig. 9. Driving anomaly results.

So, I suggest and strongly recommend that you read this paper and also download Kampana tool.

You can also download this tool and get familiar with the tool. So, that you exactly know what is happening with the engine ECU related parameters. So, driving anomalies are shown here.

(Refer Slide Time: 25:31)

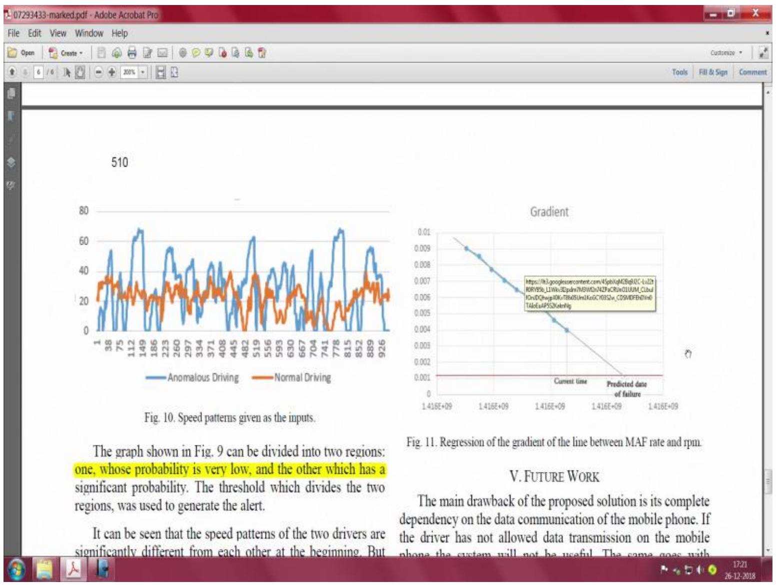


Fig. 10. Speed patterns given as the inputs.

Fig. 11. Regression of the gradient of the line between MAF rate and rpm.

The graph shown in Fig. 9 can be divided into two regions: one, whose probability is very low, and the other which has a significant probability. The threshold which divides the two regions, was used to generate the alert.

It can be seen that the speed patterns of the two drivers are significantly different from each other at the beginning. But

V. FUTURE WORK

The main drawback of the proposed solution is its complete dependency on the data communication of the mobile phone. If the driver has not allowed data transmission on the mobile

And, you can see that speed patterns are given as inputs. So, you know here that this one is anomalous driving and normal driving you can see how the results look.

(Refer Slide Time: 25:47)

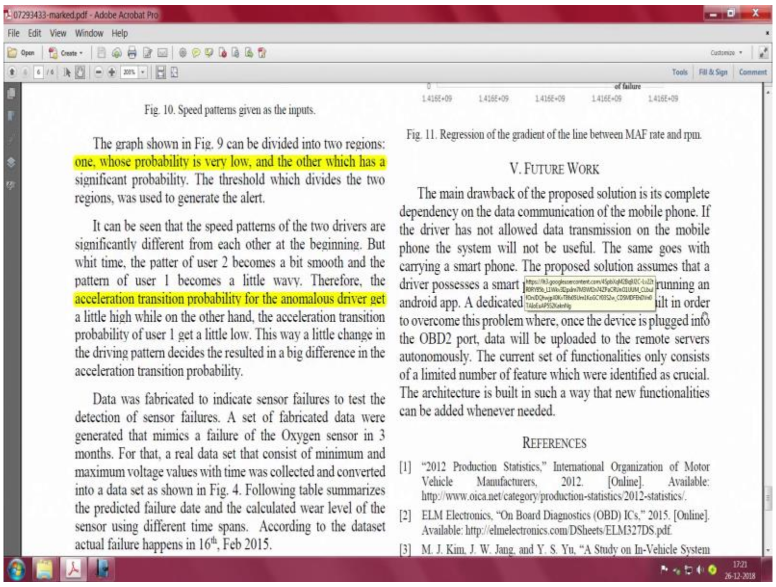

07293433-marked.pdf - Adobe Acrobat Pro
File Edit View Window Help
Open
Create
Customize
Tools
Fill & Sign
Comment
17:21
26-12-2018

Fig. 10. Speed patterns given as the inputs.

The graph shown in Fig. 9 can be divided into two regions: one, whose probability is very low, and the other which has a significant probability. The threshold which divides the two regions, was used to generate the alert.

It can be seen that the speed patterns of the two drivers are significantly different from each other at the beginning. But whit time, the patter of user 2 becomes a bit smooth and the pattern of user 1 becomes a little wavy. Therefore, the acceleration transition probability for the anomalous driver get a little high while on the other hand, the acceleration transition probability of user 1 get a little low. This way a little change in the driving pattern decides the resulted in a big difference in the acceleration transition probability.

Data was fabricated to indicate sensor failures to test the detection of sensor failures. A set of fabricated data were generated that mimics a failure of the Oxygen sensor in 3 months. For that, a real data set that consist of minimum and maximum voltage values with time was collected and converted into a data set as shown in Fig. 4. Following table summarizes the predicted failure date and the calculated wear level of the sensor using different time spans. According to the dataset actual failure happens in 16th, Feb 2015.

Fig. 11. Regression of the gradient of the line between MAF rate and rpm.

V. FUTURE WORK

The main drawback of the proposed solution is its complete dependency on the data communication of the mobile phone. If the driver has not allowed data transmission on the mobile phone the system will not be useful. The same goes with carrying a smart phone. The proposed solution assumes that a driver possesses a smart [illegible] running an android app. A dedicated [illegible] ilt in order to overcome this problem where, once the device is plugged into the OBD2 port, data will be uploaded to the remote servers autonomously. The current set of functionalities only consists of a limited number of feature which were identified as crucial. The architecture is built in such a way that new functionalities can be added whenever needed.

REFERENCES

[1] "2012 Production Statistics," International Organization of Motor Vehicle Manufacturers, 2012. [Online]. Available: http://www.oica.net/category/production-statistics/2012-statistics/.

[2] ELM Electronics, "On Board Diagnostics (OBD) ICs," 2015. [Online]. Available: http://elmelectronics.com/DSheets/ELM327DS.pdf.

[3] M. J. Kim, J. W. Jang, and Y. S. Yu, "A Study on In-Vehicle System

And, that will give you an idea of what is capable from IoT person to improve safety, to improve reliability of these vehicles. So, that is about this paper, let us move on to see a demonstration of this complete system to understand this better.