AN END-TO-END COMMUNICATION ARCHITECTURE FOR INTELLIGENT TRANSPORTATION SYSTEMS: DESIGN, IMPLEMENTATION AND LATENCY ANALYSIS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÇAĞATAY BAĞCI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

FEBRUARY 2018

Approval of the thesis:

AN END-TO-END COMMUNICATION ARCHITECTURE FOR INTELLIGENT TRANSPORTATION SYSTEMS: DESIGN, IMPLEMENTATION AND LATENCY ANALYSIS

submitted by **ÇAĞATAY BAĞCI** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Tolga Çiloğlu
Head of Department, **Electrical and Electronics Eng.** _____

Prof. Dr. Ece Güran Schmidt
Supervisor, **Electrical Electronics Engineering Dept., METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Cüneyt Fehmi Bazlamaçcı
Electrical and Electronics Engineering Department, METU _____

Prof. Dr. Ece Güran Schmidt
Electrical and Electronics Engineering Department, METU _____

Prof. Dr. İlkay Ulusoy
Electrical and Electronics Engineering Department, METU _____

Assoc. Prof. Ertan Onur
Computer Engineering Department, METU _____

Assist. Prof. Ulaş Beldek
Mechatronics Engineering Department, Çankaya Uni. _____

Date: **06.02.2018**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ÇAĞATAY BAĞCI

Signature :

ABSTRACT

AN END-TO-END COMMUNICATION ARCHITECTURE FOR INTELLIGENT TRANSPORTATION SYSTEMS: DESIGN, IMPLEMENTATION AND LATENCY ANALYSIS

BAĞCI, Çağatay
M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Ece Güran Schmidt

February 2018, 93 pages

Vehicle to anything (V2X) communication is a very significant component of Intelligent Transport Systems (ITS) applications.

This thesis proposes an application layer communication architecture, ITSVeCon for V2X communications which enables communication among the end-hosts which can be vehicle Electronic Control Units (ECU)'s, Road Side Units (RSU)s, computers, smart phones or third party service providers. All these end-hosts are bi-directionally connected to the ITSVeCon Server where this server carries out application layer switching realizing unicast or multicast communication. The architecture consists of a layered software and network protocol stack with message formats and rules, which are implemented in the end-hosts and the ITSVeCon server.

To this end, this thesis presents the ITSVeCon realization on the vehicle On Board Unit (OBU) and the ITSVeCon server. The OBU realization further fulfills the gate-

way functionality between the in-vehicle CAN network and the Internet. The ITSVeCon implementation features WebSockets carrying messages in JSON format, Publish and Subscribe pattern and NTP synchronization to enable V2X communications for real-time ITS applications. To this end, the proposed architecture allows the seamless running on different ITS applications on different types of host devices.

This thesis proposes the cellular communications as the wireless communication technology for the vehicle. To this end, the end-to-end communication path in ITSVeCon consists of cellular access and IP core network over multiple nodes and network segments. A very important contribution of the thesis is the measurement set-up, detailed experiment scenarios and measurement results of the end-to-end delay components. The measured end to end delay values are close to 100 ms with embedded component delays under 4 ms and large cellular network access delay under 3G network. Hence, a complementary short range wireless interface is proposed in this thesis as the second option to improve communication and functional tests of this option is carried out. With the improvements in technology, around 10 ms end to end delay value can be achieved with 4G and 1 ms end to end delay value is expected to be accomplished with 5G.

Keywords: V2X Communication, Intelligent Transportation Systems, On Board Units, Publish and Subscribe

ÖZ

AKILLI ULAŞIM SİSTEMLERİ İÇİN UÇTAN UCA BİR HABERLEŞME MİMARİSİ: TASARIM, GERÇEKLEŞTİRİM VE GECİKME ANALİZİ

BAĞCI, Çağatay

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Ece Güran Schmidt

Şubat 2018 , 93 sayfa

Araçtan Her Şeye (AHŞ) haberleşme, Akıllı Ulaşım Sistemleri (AUS) uygulamalarının çok önemli bir bileşenidir.

Bu tez, elektronik kontrol üniteleri, yol kenarı üniteleri, bigisayarlar, akıllı telefonlar veya üçüncü parti servis sağlayıcıları gibi uçta bulunan sistemler arasındaki iletişimi sağlayan Akıllı Ulaşım Sistemleri ve Araçlar Arasındaki Bağlantı (AUSAAB) iletişim mimarisini içermektedir. Bütün bu uç sistemler, uygulama katmanında anahtarlama yaparak tek noktaya veya çok noktaya iletişimi gerçekleştirebilen ITSVeCon sunucusuna bağlıdır. Bu mimari, uç sistemlerde ve AUSAAB sunucusunda uygulanmış olan katmanlı bir yazılım, ağ protokol yığını, ileti format ve kurallarından oluşmaktadır.

Bu amaçla bu tez, Araca Takılı Ünite (ATÜ) ve AUSAAB sunucusu üzerindeki AUSAAB

mimarisinini gerçekleştirimini sunmaktadır. ATÜ gerçekleştirimi, araç içi CAN ağı ve internet arasındaki geçit işlevselliğini göstermektedir. AUSAAB uygulaması, gerçek zamanlı AUS uygulamaları için gerçekleştirilen AHŞ iletişimleri için NTP senkronizasyonunu, JSON formatında mesaj iletimini sağlayan Websocket alt yapısını ve "Yayınla ve Abone Deseni"ni içermektedir. Bu amaçla, önerilen mimari, farklı türdeki ana cihazlarda farklı AUS uygulamalarının kesintisiz çalışmasını sağlar.

Bu tez, araç için kablosuz iletişim teknolojisi olarak hücresel ağ iletişimini sağlamaktadır. Bu nedenle uçtan uca iletişim yolu çoklu düğümler ve ağ kesimleri üzerinden gerçekleştirilen IP çekirdek ağı ve hücresel ağ erişiminden oluşmaktadır. Tezin çok önemli katkıları olarak ölçüm kurulumu, detaylı deney senaryoları ve uçtan uca gecikme bileşenlerinin ölçüm sonuçları gösterilmektedir. Ölçülen uçtan uca gecikme değerleri, 3G şebekesi altında gömülü bileşenlerdeki gecikmenin 4 ms'nin altında olmakla birlikte büyük bir hücresel ağ erişim gecikmesi altında 100 ms'ye yakındır. Dolayısıyla, iletişimin iyileştirilmesi için ikinci seçenek olarak, tamamlayıcı bir kısa menzilli kablosuz arayüz önerilmiştir ve bu şekilde fonksiyonel testler yapılmıştır. Teknolojideki gelişmelerle birlikte, 4G teknolojisiyle yaklaşık 10 ms uçtan uca gecikme değeri ve 5G teknolojisi ile 1 ms uçtan uca gecikme değerinin elde edilmesi beklenmektedir.

Anahtar Kelimeler: Araçtan Her Şeye Haberleşme, Akıllı Ulaşım Sistemleri, Araca Takılı Ünite, Yayınla ve Abone

To my family and my wife

ACKNOWLEDGMENTS

First of all, I sincerely thank to my advisor Prof. Dr. Ece Güran Schmidt for her guidance and motivation throughout the study. This thesis work was supported by the Middle East Technical University as a Scientific Research Project with contract number of BAP-03-01-2016-001.

I am grateful to my family for their love, support, encouragement and guidance throughout my entire life.

I would like to thank my employer, ASELSAN, for supporting this study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

3GPP	3rd Generation Partnership Project
API	Application Programming Interface
BSP	Board Support Package
CAN	Controller Area Network
CDMA	Code Division Multiple Access
D2D	Device to Device
DSRC	Dedicated Short Range Communication
ECU	Eloctronic Control Unit
EPS	Evolved Packet System
E-UTRAN	volved Universal Terrestrial Access Network
HSPA	High Speed Packet Access Protocol
HTTP	Hypertext Markup Language
IEEE	Institute of Electrical and Electronics Engineers
EP	End Point
ITS	Intelligent Transportation Systems
ITSVeCon	Intelligent Transportation Systems and Vehicular Connectivity
JSON	JavaScript Object Notation
LIN	Local Interconnect Network
LON	Line of Sight
LTE	Long Term Evolution
LTE-A	Long Term Evolution Advanced
MAC	Media Access Control
MMBS	Multimedia Broadcast/Multicast Service
MOST	Media Oriented Systems Transport
NAT	Network Address Translation
NTP	Network Time Protocol
OBU	On Board Unit
OFDMA	Orthogonal Frequency Division Multiple Access
RAN	Radio Access Network

RSU	Road Side Unit
SABRE-AI	Smart Application Blueprint for Rapid Engineering for Automotive Infotainment
SAE	Society for Automotive Engineers
SRWI	Short Range Wireless Interface
TMC	Traffic Management Center
UTC	Coordinated Universal Time
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle
V2X	Vehicle to X
VANET	Vehicular Adhoc Network
VM	Virtual Machine
WAVE	Wireless Access in Vehicular Environment
WiFi	Wireless Fidelity

CHAPTER 1

INTRODUCTION

Intelligent Transportation Systems (ITS) support transportation of goods and humans with information and communication technologies in order to efficiently and safely use the transport infrastructure and transport means [1]. ITS exploit technologies from multiple disciplines to improve transportation systems in all aspects by increasing traffic information, reducing driving loads and enhancing route management [2].

Vehicle to Vehicle (V2V) or Vehicle to Infrastructure (V2I) communication are very significant components of ITS. These different types of communications involving the vehicle are called V2X communications. The ITS applications with V2X communications incorporate Road Side Units (RSU), vehicles, user devices and service providers. Applications serving different purposes have been developed for each platform.

These ITS applications are classified according to being safety related or not. While emergency electronic brake lights, traffic condition warning and pre-crash sensing warning are the examples of safety related ITS applications, media download, intersection management and traffic light optimal speed advisory can be given as examples to non-safety applications [3]. Frequency of safety and non-safety related messages ranges from 1 Hz to 10 Hz. Maximum latency of safety related messages is generally 100 ms except for pre-crash sensing warning which requires 50 ms latency. In non-safety applications, latency up to 500 ms is acceptable.

V2X wireless communications are enabled by mostly DSRC (Dedicated Short Range Communications)/WAVE that is derived from IEEE 802.11p and cellular communication standards 3G and LTE. Although DSRC/ WAVE are very widely promoted

for V2X communications particularly in the US, there are many drawbacks which degrade the overall performance. The main issues are that packet collisions and medium access collisions occur at high rates when the vehicle population is dense. Packet reception ratio decreases drastically when distance between vehicles increases. These issues make it hard to adopt these protocols to ITS safety applications which requires high packet reception ratios with low latencies. In addition to that, obstacles such as buildings result in coverage problems affecting transmission performance. As for D2D communication, it is a very new proposal and is not widely deployed yet. Furthermore, as it shares the same frequency band with cellular networks which may cause interference problems.

Cellular access network offers the best packet reception ratios with higher latencies compared to wireless communication types. However, with the improvements in the technology, especially latest releases of LTE towards 5G mobile communication network is considered to take away many disadvantages. Low access network latency down to 1 msec, high bandwidth up to 10Gbps, reliability and ubiquitous coverage can all be achieved with 5G which makes it the most promising choice for ITS communication. In addition to these, today backbone networks can support 100 Gbps connection speed supporting the usage of 5G.

In-vehicle communication is also a part of ITS applications. Electronic control units located in the car are connected with various interconnection mechanisms such as Controller Area Network (CAN, CAN-FD), FlexRay and Ethernet. To use messages generated in vehicles by these ECUs in V2V or V2I communication, there should be an additional component having interfaces suitable for in-vehicle, inter-vehicle and vehicle to infrastructure. A common practice is to use commercial off-shelf unit having flexible interfaces called On Board Unit (OBU) [4].

In this thesis, we propose an ITS communication architecture which achieves V2X communications through cellular access network and IP-backbone network. In this architecture, the end hosts can be ECUs, RSUs, third party servers, computers and smartphones. All these end hosts are are always connected to a Vehicular Connectivity (ITSVeCon) Server via WebSockets. The ITSVeCon Server processes the data collected from the end hosts and other ITS sources of information. Accordingly,

ITSVeCon Server provides end-to-end unicast or multicast communication among the end-hosts by application layer switching. The vehicle connectivity is achieved by OBUs located in vehicles with cellular network interfaces. In addition to that, the OBUs are connected to In-Vehicle CAN Network, fulfilling CAN-to-Internet gateway functionality. Furthermore, the vehicles can directly connect with each other through short range wireless interfaces without the need of ITSVeCon Server to support connectivity and improve the latency. Communication with other end points such as smart phones or RSUs are processed over ITSVeCon Server.

The first contributions of this thesis is the implementation of ITSVeCon together with the layered protocol architecture, OBU software and ITSVeCon Server software. The second and a very significant contribution of the thesis is the measurement set-up and the measured values for end to end latency values for V2V communication using ITSVeCon. The transmission path starts with an ECU transmitting a CAN frame and ends in another ECU receiving the same CAN frame in a different vehicle. In this thesis, this transmission path is deeply studied and delay values between each components located in this path are measured. An additional application is implemented for a device capable of transmitting CAN frames to simulate ECUs. Time synchronization infrastructure is established for measurements. Usage of short range wireless interface together with 3G to achieve better latency is realized on OBU ITSVeCon Applications.

This thesis is organized as follows:

First, relevant background information about ITS applications, performance metrics, communication types of ITS applications, network application layer technologies and time synchronization is given in Chapter 2. In addition to that, ITS communication architectures in the literature are investigated.

Chapter 3 explains the proposed ITSVeCon architecture, communication type and implemented applications. This chapter includes a comparison of ITSVeCon to the previous work of CarCode presented in [5].

Chapter 4 gives evaluation of the proposed ITSVeCon architecture, measurement method and results.

Finally, thesis is concluded in Chapter 5.

CHAPTER 2

BACKGROUND AND PREVIOUS WORK

2.1 ITS Applications and Communication

A contemporary vehicle has a large number of electronic components including Electronic Control Units (ECUs), sensors and actuators to realize distributed applications with the help of in-vehicle communication among these components. It is important to note that some ITS applications are distributed over the components of a single vehicle and require in-vehicle communication only whereas other ITS applications involve geographically separate end nodes. To this end, V2X communication between the vehicle and other vehicles, Road Side Units (RSU) and Internet end-nodes take place. Main ITS applications about safety services and their maximum tolerable latency values are stated in Table 2.1. Some non-safety related ITS applications are given in Table 2.2.

2.2 Performance Metrics for V2X Communication for ITS

ITS applications have different communication requirements that are quantified by the following metrics [6]. Here we assume that the application is running over multiple ITS nodes that are not on the same vehicle.

- **End to end latency:** It is the maximum tolerable time from the time message is generated at source application until it is received by destination application. For short range wireless communication, mostly the air interface latency defines end to end latency. For cellular network communication, sum of uplink, routing and downlink time defines end to end latency. It is the most critical design factor

Table 2.1: ITS safety services and maximum latency values [3]

Application	Communication Mode	Min. Frequency of Per. Messages	Maximum Latency
Emergency electronic brake lights	Time limited periodic broadcast on event	10 Hz	100 ms
Abnormal condition warning	Time limited periodic broadcast on event	1 Hz	100 ms
Slow vehicle warning	Periodic triggered by vehicle mode	2 Hz	100 ms
Wrong way driving warning	Time limited periodic broadcast on event	10 Hz	100 ms
Roadwork warning	Temporary messages broadcasting on event	2 Hz	100 ms
Lane change assistance	V2X co-operative awareness	10 Hz	100 ms
Pre-crash sensing warning	Broadcast of pre-crash state	10 Hz	50 ms

especially for safety related applications.

- **Reliability:** It is the maximum tolerable packet loss rate of ITS application. If destination point does not obtain the generated packet, it is counted in lost packets. For example, big trucks, tunnels and buildings lead to packet loss for communications over short range wireless interfaces such as DSRC.

- **Data rate:** It defines minimum bit rate necessary for application to work properly.

- **Communication range:** It is the maximum distance between source and destination points which can provide required reliability with respect to relevant ITS application. Communication range is also closely related to end to end latency.

- **Node mobility:** It is the maximum speed which can satisfy reliability conditions. Node mobility is an important issue for short range wireless interfaces because of limited mobility support. On the other hand, communication over cellular network supports higher speeds.

- **Network density:** Node density defines maximum number of vehicles that can be present at a specified area without affecting reliability.

Table 2.2: ITS non-safety services and maximum latency values [3]

Application	Communication Mode	Min. Frequency of Per. Messages	Maximum Latency
Traffic light optimal speed advisory	Periodic, permanent messages broadcasting	2 Hz	100 ms
Intersection management	Periodic, permanent messages broadcasting	1 Hz	100 ms
Electronic toll collect	I2V broadcasting and unicast full duplex session	1 Hz	500 ms
Local electronic commerce	Duplex commun. between RSU and vehicles	1 Hz	500 ms
Media download	User access to internet for multimedia download	1 Hz	500 ms

- **Positioning accuracy:** It defines the maximum location error that can be tolerated by the application.

- **Security:** Different security features are required for ITS applications such as authentication or user privacy.

2.3 Communication Types in ITS Applications

2.3.1 In-Vehicle Communication

In-vehicle communication is an indispensable part of overall ITS architecture, playing a key role to access ECUs located in vehicles. Vehicles include complex intelligent electronic and mechanical systems in today's technology. Engine, transmission, ABS control mechanisms, power locks and entertainment systems can be given as an example to electronic systems. There are many communication types among these ECUs differing according to the application. In-vehicle network connects all of this electronic systems in order to add flexibility, to avoid wiring and to have better control. Society for Automotive Engineers (SAE) has classified in-vehicle networks into four classes according to network speed [7] [8]. Table 2.3 illustrates these classes. Mostly used in-vehicle networking protocols and their applications are listed in Table 2.4 [9].

Table 2.3: In-vehicle network classes

Class Number	Specification
Class A	Low speed (< 10Kbits /Second) Seat control, door lock
Class B	Medium Speed (10Kbitps to 125Kbps) Vehicle speed, general message transferring
Class C	High speed (125 Kbitsps to 1M) Real time applications
Class D	Speeds higher than 1Mb/sec Internet, X by Wire

Table 2.4: In-vehicle networking protocols

Protocol	Applications
LIN	Door Locks, Climate Control, Seat Belts, Sunroof, Lighting,Window Lift, Mirror Control
CAN	Body Systems, Engine Management, Transmission
FlexRay	Drive-by-Wire, Brake-by-Wire, Advanced Safety and Collision Avoidance Systems, Steer-by-Wire, Stability Control, Camera-Based Monitoring Systems
RF	Remote Keyless Entry, Vehicle Immobilization, Passive Entry, Tire Pressure Monitoring Systems

2.3.1.1 CAN Bus

According to [10], Controller Area Network is a serial communications protocol which efficiently supports distributed real time control with a very high level of security. CAN supports bitrates up to 1 Mbps. Prioritization of messages, configuration flexibility and error detection are main capabilities of this communication type. CAN 2.0A and CAN2.0B are types of CAN protocol. While CAN 2.0A uses 11 bit identifier, CAN2.0B uses 29 bit identifier. Bit fields of standard CAN frame is given in Figure 2.1. Each CAN frame starts with Start of Frame (SOF) bit used to synchro-

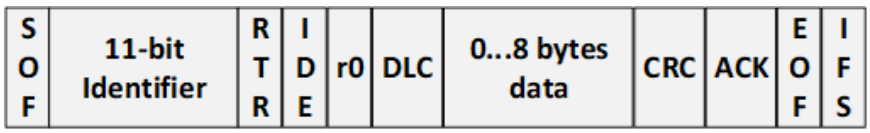

Figure 2.1: CAN frame structure

nize the nodes on the bus after idle period. After that identifier bits are placed which are 11 bits for standard CAN frame and 29 bits for extended CAN frames. Single remote transmission request is used to request data transmission from the node defined in identifier field. IDE bit is used to indicate whether the frame is extended or stan-

dard frame. R0 is reserved bit. DLC field defines the number of data bytes. Cyclic redundancy check bits come after these field for error correction. ACK bits are used for nodes to acknowledge received data. End of frame (EOF) marks the end of CAN frame. Interframe space (IFS) bits are used for controller to move the received frame to buffer.

CAN physical layer implements logical AND operation on the bus. '0' logic level is dominant and '1' is recessive. If a node transmits a dominant bit level on the bus, it is in dominant state regardless of any conditions. Bus arbitration is done according to ID bits. A frame with low ID field has higher priority over other frames, so this frame wins arbitration. If nodes detect any other node that is transmitting a higher priority frame, they stop transmitting and wait for end of the frame.

CAN frames are divided into two according to schedule. Periodic CAN frames are sent with predefined intervals. Sporadic frames can be transmitted in any time. A CAN frame that is transmitted by emergency brake can be given as an example to sporadic frames. These frames have higher priority over other frames because of strict latency requirements. CAN schedule analysis is done by calculating worst case response time which should be below deadline to perform as intended. Worst case response time has three elements. If m represents a frame with priority m, J_m represents queuing delay which is the time between initiation and the time the frame is ready to be transmitted on the bus. w_m is the maximum queuing delay for the frame. C_m is transmission time of of the frame. Worst case response time R_m is given by the equation below.

$$R_m = J_m + w_m + C_m$$

Queuing delay w_m includes two components. First one is blocking B_m which is due to lower priority messages being in transmission process when frame m is queued. Second component is interference caused by higher priority messages' winning arbitration. In [11], w_m is given by the equation below.

$$w_m^{n+1} = B_m + qC_m + \sum_{\forall k \epsilon hp(m)} \lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \rceil C_k$$

In this equation, B_m is the maximum queuing delay represented with maximum blocking time. Worst case response time measurements are done in various platforms to see if CAN frames can meet deadline requirements. SAE benchmark test uses seven different subsystems transmitting 53 different CAN frames [12]. CAN frames include both sporadic and periodic messages. Sporadic messages generally have 20 ms deadline, periodic messages have deadlines between 5 ms and 1000 ms. While deadline requirements can be satisfied with 250 Kbps bus rate, 125 Kbps fails to meet sporadic messages' latency requirements. This shows the importance of selecting the right bus speed according to the platform. In [13], network calculus method, a method used to determine upper and lower bounds of end to end delays, is used for in-vehicle CAN network. According to the data sets obtained from Audi, maximum end to end delay varies from 1 ms to 14 ms.

2.3.2 Vehicle to X (V2X) Communication

2.3.2.1 Short Range Communication

- **DSRC/WAVE:** Dedicated Short Range Communication (DSRC) is a set of standards mainly developed by IEEE for wireless communication. Wireless Access in Vehicular Environment (WAVE) term defines core standards of this communication type based on IEEE P1609.X [14]. IEEE 1609.1 core system standard, IEEE 1609.2 security standard, IEEE 1609.3 network services and IEEE 1609.4 channel management standards focus on MAC and network layers constituting WAVE term. IEEE 802.11p, which is an improved standard compared to IEEE 802.11a is used in PHY layer. Whereas WAVE defines the core standards of DSRC, generally DSRC and WAVE terms are used together. US allocated 75 MHz spectrum in 5.9 GHz frequency band for DSRC/WAVE [15]. Seven 10 MHz bands and one 5 MHz guard band are defined in this spectrum interval.

  DSRC/WAVE network uses two components which are On Board Units (OBU) and Road Side Units (RSU). OBUs are located in vehicles having wireless interface to communicate with RSUs. RSUs are the units connecting vehicles to access network which is intermediate network before core network. Each

Table 2.5: Differences of IEEE 802.11a and IEEE 802.11p Protocols [16]

Parameters	IEEE 802.11a	IEEE 802.11p
Bit Rate	6, 9, 12, 18, 24, 36, 48, 54	6, 9, 12, 18, 24, 36, 48, 54
Code Rate	1/2, 2/3, 3/4	1/2, 2/3, 3/4
Symbol Duration	4 μs	8 μs
Guard Time	0.8 μs	1.6 μs
FFT Period	3.2 μs	6.4 μs
Preamble Duration	16 μs	32 μs
Subcarrier Spacing	0.3125 MHz	0.15625 MHz

RSU has a limited communication zone. IEEE 802.11p standards are used for the communication between OBU and RSU. IEEE 802.11p brings many advantages compared to IEEE 802.11a. Table 2.5 illustrates the differences of these two protocols. In [16], IEEE 802.11a and IEEE 802.11p are compared in contact duration and loss comparison aspects. For contact duration test, car is started its movement outside connection range of RSU and maintains the same speed at 200 meter proximity to RSU. Results show that at 20 Km/h speed, contact time of IEEE 802.11a is 4.5 seconds, while IEEE 802.11p is 38.5 seconds. Contact duration is 0 for 802.11a for 40 Km/h and 60 Km/h speeds. IEEE 802.11p provides a contact duration of 19 seconds and 14 seconds for 40 Km/h and 60 Km/h speeds respectively. For loss comparison, losses of 802.11p are close to zero in line of sight (LON) environment while it is 2.68% in non-line of sight (NLON) environment. Losses can reach to 11% in NLON environment for 802.11a protocol.

While DSRC/WAVE is considered a feasible solution to ITS applications, it has many drawbacks. According to [17], these drawbacks results from PHY layer, MAC layer and multi-channel operations. First of all as vehicles moves very fast and there are a lot of obstacles such as tunnels, bridges and intersections in the environment, this can affect the radio performance badly. As for MAC layer, packet collision ratio can reach to remarkable value when the density of vehicles in one area is high. Since IEEE 802.11p protocol is a multi-channel protocol, a rule is required for vehicles to use different channels for different applications. Although some solutions are proposed in [17] for these disadvantages, there are still challenges and unresolvable problems in DSRC/WAVE

protocol.

- **D2D:** In cellular networks, users should always communicate with cellular receivers. Even if two users are in close distance with each other, data goes through additional path when compared to DSRC/WAVE. D2D is proposed as a solution to this problem. Device to Device (D2D) is defined as the communication between two users directly in close distance to eliminate RSU usage [18]. D2D is designed to use cellular network resources and can work in four modes. In silent mode (no D2D) all resources are allocated for cellular network so D2D usage is restricted. Underlay D2D uses the resources at the same time with cellular network. This can cause interference problems between D2D and cellular network. In overlay D2D some resources of cellular network is allocated for D2D communication. Unique mode (D2D only) uses all resources of cellular network. In [19], D2D communication is applied in LTE-A cellular network standard which brings many innovations compared to DSRC/WAVE. 3 ms delay value are obtained from simulation results for D2D communication when number of vehicles are 30 in coverage area. However, there are no practical ways to implement D2D communication in ITS applications yet, so only simulation results can be used for analysis. The main disadvantages of D2D are described in [20] as follows. In underlay D2D, interference management and power control is hard to realize between D2D and cellular network. Despite allocated resources, overlay D2D communication has low spectral efficiency compared to underlay. Last words that can be said for D2D communication is that, it has a long way ahead to improve and develop itself to adapt to real life conditions.

2.3.2.2 Access Network

- **Cellular Networks (3G/4G/5G):** Cellular network is mostly used in daily life and it is becoming more popular in ITS applications. The trend of using cellular network in ITS applications started with 3G solution which offers data rates up to 5 Mbps and latencies between 100 ms and 500 ms [21]. 3G uses an upgraded version of Code Division Multiple Access (CDMA) which is Wideband CDMA (WCDMA). High Speed Packet Access Protocol (HSPA) is used to obtain better

download speeds at the cost of upload speed.

LTE/4G technology can reach 1-50 Mbps data rates and latency below 100 ms [21]. LTE network architecture is consisted of user equipment, Evolved Universal Terrestrial Access Network (E-UTRAN) and Evolved Packet System (EPS) [22]. E-UTRAN contains base stations which are called eNodeB or eNB. eNB communicates with user equipment in its coverage area. These base stations are connected to EPS which can be also called as core network of LTE. LTE uses Orthogonal Frequency Division Multiple Access (OFDMA) modulation and uses wide bandwidths up to 20 MHz. LTE-Advanced (LTE-A) is an upgraded version of LTE, offering increased peak data rate, higher spectrum efficiency and better performance at cell edges [23]. In spite of having such features, LTE still is not considered an efficient way. According to [24], when vehicles sends periodic messages to eNB at every 100 ms, these nodes becomes overloaded even if an idealistic assumption is used. Even 802.11p beaconing outperforms LTE. Because of that, D2D approach in LTE-A is considered to be a better solution. However, as explained in D2D section, it has several problems too.

In today's technology 5G cellular network is being developed day by day. It has many features already declared, it is expected to be a milestone in ITS applications. 5G uses Beam Division Multiple Access (BDMA) and Filter Bank multi carrier (FBMC) access technologies [25]. With applying this technologies, each user equipment obtains an orthogonal beam and this beam is divided according to locations of mobile stations. 5G has more throughput, substantial amount of bandwidth, higher mobility and low latency values [6]. 5G is expected to solve all coverage problems addressed in other V2X communication types. Mobility up to 500 Km/h, 10-5 packet loss ratio and 1 ms end to end latency are supported with 5G solution. In addition to that, D2D communication is expected to be handled better.

2.3.2.3 Overall Evaluation of V2X Communication Types

There are other solutions for vehicular communication such as Wifi, Zigbee, and Bluetooth which are lagging behind other communication types due to disadvantages.

Table 2.6 illustrates some of the features of 802.11p, LTE-A, Wifi-Direct, Zigbee and Bluetooth. Table 2.7 compares advantages and disadvantages of 802.11p and LTE communication.

Table 2.6: Comparison of 802.11p, LTE-A, Wifi-Direct, Zigbee and Bluetooth [19]

Feature	LTE-A	802.11p	Wifi Direct	Zigbee	Bluetooth
Frequency Band	Licensed band	5.86-5.92 GHz	2.4, 5 GHz	2.4 GHz	2.4 GHz
Max. Trans. Distance	1000 m	200 m	200 m	10-100 m	10-100 m
Max Data Rate	1 Gb/s	27 Mb/s	250 Mb/s	250 kbps	24 Mb/s
Mobility Support	Up to 350 Km/h	Up to 60 Km/h	Low	Low	Low

Many V2X communication types are explained in previous section. They all have their own disadvantages and advantages except 5G. Considering the fact that 5G is still in development and there is no practical implementation with 5G related to vehicular communication, it can still has drawbacks. The best choice for V2X communication is open to discussion. Further work is needed to apply these communication types to real world.

Table 2.7: Comparison of 802.11p and LTE [6]

Features	IEEE 802.11p	3GPP LTE
Traffic Bottleneck	No (fully distributed)	Yes (eNB)
Spectral Efficiency	Low (throughput performance degrades under high load due to backoff procedure)	High (channel dependent scheduling in frequency selective channels)
Qos Guarantees	Not guaranteed (due to probabilistic nature of CSMA/CA backoff procedure).	Guaranteed after connection establishment

2.4 Network Application Layer Technologies

2.4.1 Websocket Protocol

Websocket protocol is designed to provide a full-duplex communication over a single TCP connection [26]. In HTTP protocol, many TCP connections need to be opened in order for the server and the client to communicate. Because of that, server becomes overloaded. In fact HTTP is not designed for full-duplex communications. Websocket is designed to address these issues by enabling bi-directional communication over a single connection.

In Websocket protocol, at first, client sends a handshake message to the server. Server responds back by sending a different handshake message. If handshake is successful, data transfer starts. Each side can send data at any time which is the most important feature of the protocol. Websocket frame format is shown in Figure 2.2. FIN field defines if the message is final fragment or not. RSV1, RSV2 and RSV3 bit fields are 0 unless there is an extension. Opcode defines if payload data is a continuation of frame, text frame, binary frame, connection close, ping or pong. MASK bit is used to specify if payload data is masked. Payload length defines the length of payload data. Masking key is 32 bit value which is used if MASK bit is 1. Payload data is sent after all these fields compatible with the format defined in opcode field.

HTTP polling, long polling and Websocket protocol are explained and compared in [27]. In HTTP polling, after client sends a request to the server, server responds to the client with new message if it exists or an empty response if there is no new message. Server does not just immediately sends empty response if there is not any new messages in HTTP long polling. Server waits until a new message is available for the client and then it sends the message to the server. Timeout is used to define a maximum interval of time that client can wait to get a response. To compare these three protocols, one way latency is measured between server and client at different times in [27]. In this test, messages are obtained from a wind sensor at 4 Hz rate by the server and the server handles HTTP and Websocket requests made by clients. The results show that Websocket protocol has the lowest latency values among all of them. Long polling HTTP performs the second after Websocket protocol. Polling

HTTP is the worst option having latencies 3 times more than Websocket protocol. ZeroMQ is an asynchronous messaging library, aimed at use in distributed or concurrent applications [28]. For Websocket protocol, client and server have fixed roles, while in ZeroMQ these roles can be changeable. In addition to that, ZeroMQ is used for much complex messaging pattern by using many sockets on the same node. The architecture is not centralized and administrated by one node. Websocket protocol is selected for being more suitable to the centralized architecture proposed in this thesis.

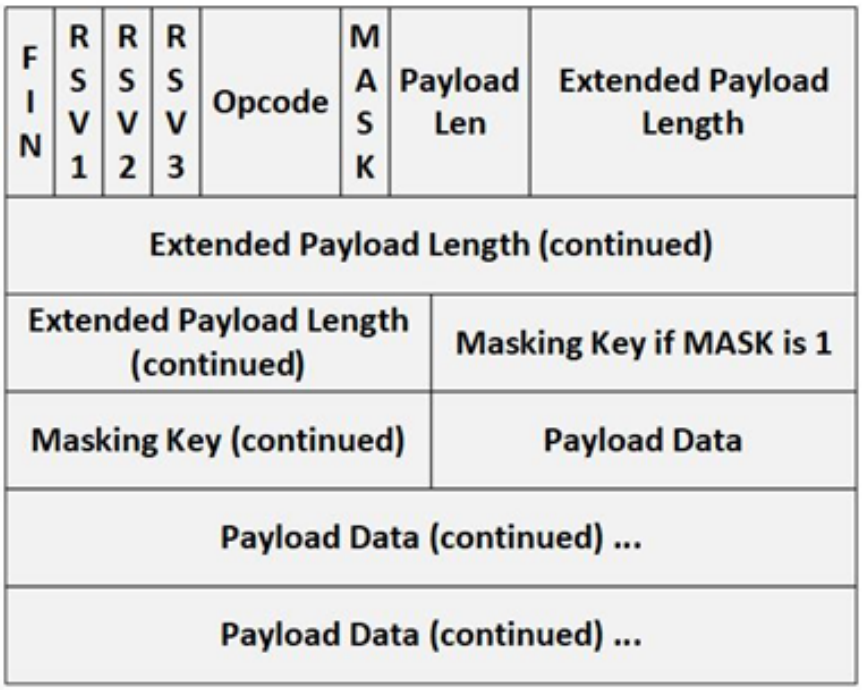

Figure 2.2: Websocket Frame Format [26]

2.4.2 Publish and Subscribe Pattern

Publish-Subscribe pattern is an asynchronous communication pattern, where the subscribers are the special clients showing their interest in form of subscriptions, and the publishers are the clients sending information to subscribers [29]. In Publish-Subscribe pattern, subscribers first subscribe to the topics according to their interest. These topics can also be called channels. To give an example about ITS application, a vehicle may subscribe to the "Weather" channel if it wants to get information about the weather conditions. On the other hand, publishers are the ones that send informa-

tion to these channels. A RSU can send information about weather conditions to the channel named “Weather”. When a publisher sends information to a channel, all of the subscribers in this channel get this information. The process is shown in Figure 2.3. For our example, after RSU sends information about the weather conditions to “Weather” channel, all of the vehicles in this channel get this information and see it on their dashboards. To publish information to a channel, it is necessary to be subscribed first.

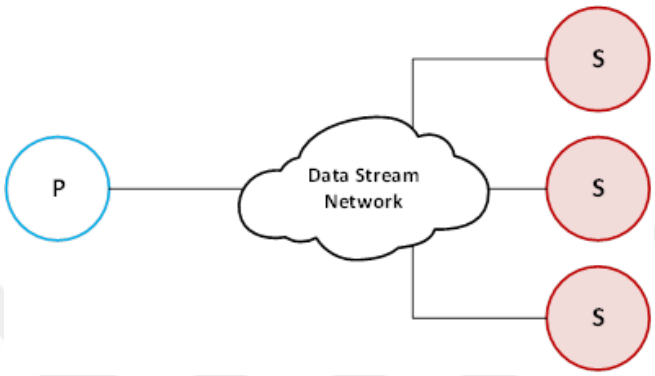

Figure 2.3: Publish and Subscribe Pattern

In Publish and Subscribe pattern, there is decoupling between publishers and subscribers which can be examined in three different sections [29]. First of all, publishers and subscribers do not need to know each other which is space decoupling. Subscribers does not contain a list of publishers, publishers does not know any information about subscribers similarly. For time decoupling, subscribers can receive events after some period of time if they are disconnected from server at the time when publisher sends an information. In addition to that, subscribers can get notified asynchronously while carrying out another activity.

2.5 Time Synchronization

Time synchronization is an important part for delay measurement purposes. Especially for ITS, end to end latency is important to decide whether the communication structure allows real time ITS applications. If delay measurement is to be carried out between two systems that are located at different points, time synchronization issue arises. This case perfectly fits to ITS applications because of the distance between

vehicles.

Network Time Protocol (NTP) is the biggest solution to time synchronization problem. Coordinated Universal Time (UTC) represents solar time defined by national standards laboratories [30]. System time is the time set by hardware and operating system. The aim of NTP protocol is to minimize the time difference between UTC and system clock with the help of time servers. Time servers are special computers obtaining real time information from reference clocks and distributing time information to its clients using NTP protocol. The accuracy of time servers are defined according to stratum level. Stratum 0 devices are very high precision devices such as GPS clocks or atomic clocks. Stratum 1 computers obtain their time information from stratum 0 devices and stratum 2 computers are synchronized with stratum 1 computers. Therefore computers with the lower stratum numbers have the higher accuracy. There are some terms that is used to define synchronization performance of clients to time servers [30]. Offset is calculated using root mean squares showing the time difference between time server and client. Delay is the round trip delay between server and client. Jitter indicates root mean square (RMS) average of the most recent offset differences.

A sophisticated algorithm is needed to synchronize clocks using NTP protocol. NTPD is an operating system daemon used to synchronize system clocks with time servers [31]. NTPD program exchanges messages with one or more NTP servers at specified intervals. NTPD adjusts system clock in small steps to prevent discontinuities. Offsets higher than 128 ms is discarded by NTPD. NTPD is a Linux based program having some configuration options for operation. Configuration options are kept in "ntp.conf" file which is required for NTPD at startup. In "ntp.conf" file "server" name is used to define NTP servers. "minpoll" and "maxpoll" options are used to define minimum and maximum poll intervals for messages, in seconds to the power of two. "iburst" option is used to send a burst of eight packets to server when client cannot reach server. "ntpq –p" command is used to obtain offset, jitter and delay values created by NTPD. A sample output of this command is given in Figure 2.4. "remote" section shows time server. "reach" value shows the failure rate of connecting with the time server. 377 is the highest value meaning that last eight communications with time server is successful. "st" means the stratum of the server. "t" is the type of the

connection which can be unicast (u) or broadcast (b). “ntpq –p” output can be used to define the quality of synchronization.

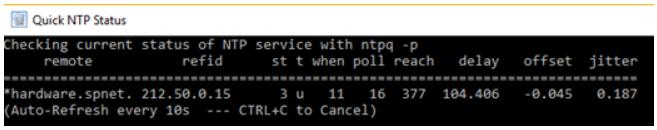

Figure 2.4: NTPQ Output

Since NTPD is a Linux based daemon, it cannot be used in Windows OS. Meinberg program is a different version of NTPD, which can run in Windows operating system [32]. The configuration file and commands are the same with NTPD program. These two programs are widely used for synchronization purposes. In the scope of this thesis, NTPD and Meinberg programs are both used for delay measurement.

2.6 Previous Work on ITS Connectivity

Many research and work have been done in ITS connectivity area. Some of them offer simulation results without practical usage, some of them implement the work in real life but there is still disagreement about selecting the best communication type. In the literature, works about measuring actual end to end latencies are lagging behind with respect to simulation results. In this chapter, different ITS applications, communication types and architectures existed in the literature are analyzed.

ITS applications can be used for various purposes. In [33] ITS application about controlling the traffic at intersection points to reduce CO2 emission is proposed. Virtual Traffic Light (VTL) is a presented concept to improve traffic flow using VANET. Vehicles always transmits their position information with beacon messages. When a vehicle approaches to an intersection point it checks for other vehicles’ beaconing messages. If it does not receive any other beaconing messages from vehicles coming from other directions then VTL is not needed to be created. If there is another vehicle coming from other directions and close to intersection point, then one of the vehicles at the front of the lane is selected as leader. The leader stops at the intersection point and it is responsible for controlling VTL. Green light is available for the vehicles coming from other direction. After a predefined period, the leader is assigned

with green light. Before moving, the leader selects the new leader among the vehicles waiting at the intersection point. Leader selection is not done when there are not any vehicles waiting at red light. Simulation results are obtained using traffic density and road information of Porto in Portugal. The results show that fuel consumption is reduced by 25%. In addition to that, average vehicle velocity increases between 26% and 41%.

Latency analysis between OBU and Traffic Management Center (TMC) in Sarubaya city using 3G and LTE cellular network is realized in [34]. Messages contain information of vehicle location, speed, route and passenger. Firstly OBU remains stable and latency is measured with 3G and LTE. Then latency is measured again for mobile OBU using 3G. OBU sends and receives data packets limited below 1500 bytes every minute. OBU application runs on the laptop connected to internet with 3G or LTE modem. Time stamp values are recorded in laptop for end to end delay measurement. T_{start} is the time when laptop sends HTTP.GET packet. Laptop sends HTTP.GET command to the PC with ITS server. It waits until HTTP.GET transaction is completed. T_{stop} is the time when laptop saves timestamp. Then time difference is recorded and this procedure is repeated for several times. According to results, 90% of the data can be delivered in 4 seconds using 3G connection in static location. When using LTE network, 90% of the data can be transferred in 0.25 seconds. As for mobile condition, 90% of the data is transferred in 5 seconds with 3G connection. Average latency values are not tolerable for safety related applications but it can be useful for infotainment purposes such as ticketing application.

Another latency analysis using LTE networks is presented in [35]. Round-trip time of packets that is sent from a moving vehicle is measured with three different mobile carriers. Test routes are chosen from Federal University of Pernambuco which is located in Recife. Smart phone with Android OS is used to access LTE network. ICMP messages are sent to web server and round-trip times are measured. Figure 2.5 illustrates an overview of measurement method and the components of LTE network. Throughput of the messages changes from 1 to 10 Hz and the length of the messages

are 800 bytes. Total delay can be expressed in the following equation.

$$L_{E2E} = 2 * (L_{E-UTRAN} + L_{EPC} + L_{Internet})$$

End to end delay is measured as 132 ms with carrier "C". This value is 147% lower than measured delay value with 3G usage. Secondly carrier "T" is used and 244,5 ms average end to end latency value is obtained. Finally average end to end latency is 122 ms when carrier "V" is used. This is just 2% lower with respect to 3G usage of carrier "V". When all results are analyzed, it is seen that these latency values do not fit the requirements of real time ITS applications. However, if network architecture and service providers are improved, then average latency can be reduced below 100 ms.

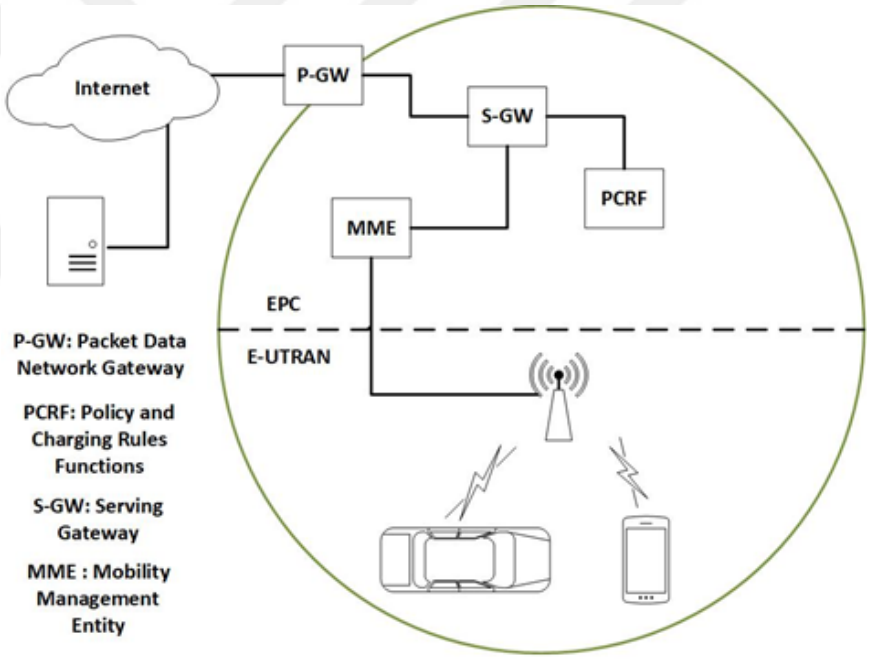

Figure 2.5: LTE architecture used in measurement method

Another work that focuses on improving LTE performance for ITS applications is presented in [36]. LTE can be used for the areas that DSRC infrastructure is not enough but messages has to be transmitted to server first across Radio Access Network (RAN) and this can add extra delay compared to DSRC. Secondly when large number of devices communicate over eNBs, scalability issues arise. This paper proposes that the disadvantages of LTE network can be eliminated by selecting servers close to vehi-

cles and by using broadcast communication. Data freshness concept is defined as the time difference between the generation and consumption of information. Freshness can be 1 ms for event-based messages using DSRC communication because end to end delay is low. For periodic messages transmitted at 10 Hz, freshness is 100 ms. In LTE networks, this latency value can be achievable by reducing message period. For example, if the round trip delay between cloud server and vehicle is 50 ms and message period is 100ms, the worst case freshness is 100+50=150 ms. If the vehicles transmit location update messages at every 50 ms, then the worst case freshness is 50 + 50 = 100 ms which is same with DSRC communication. Disadvantage of increasing frequency of messages is that network utilization becomes higher. Server location has significant impact on network usage. For example, when RTT is lowered from 80 ms to 60 ms for the messages with 100 ms target freshness, the network usage is reduced by half. Therefore placing servers close to eNB or EPC is important for network utilization. LTE supports Multimedia Broadcast/Multicast Service (MBMS) which enables server to send broadcast messages. MMBS and non-MMBS cases are shown in Figure 2.6.

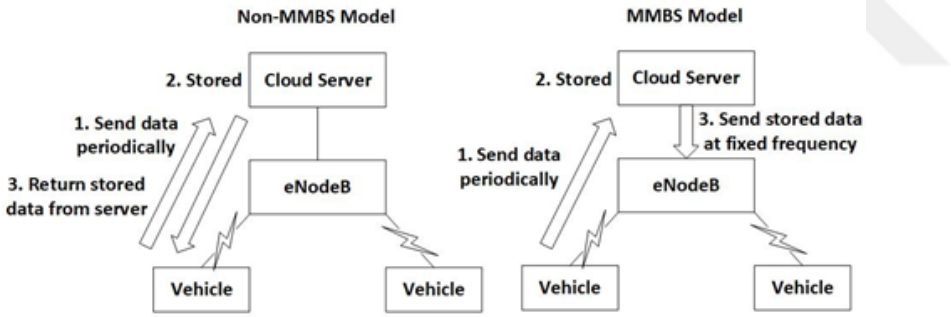

Figure 2.6: MMBS and Non-MMBS architectures of LTE

Carriers do not support MMBS service at the moment but it is expected to be accomplished soon. Server obtains location updates from vehicles and sends combined broadcast messages to all vehicles periodically. On the other hand, one message including the location updates of nearby vehicles is sent to only one vehicle by server in non-MMBS case. Experiments are done for non-MMBS case by sending UDP data packets to server periodically. RTT is measured as approximately 50 ms when two vehicles are present in the architecture. RTT level stays the same until twelve vehicles. After this number, RTT starts to increase linearly. Simulation is done for MMBS

and non-MMBS cases. When number of vehicles increases, freshness is increased drastically for non-MMBS architecture. For example, when freshness is around 100 ms for MMBS architecture with 150 vehicles present, it is approximately 200 ms for non-MMBS case. In conclusion, by using MMBS architecture and by placing servers close to eNB, end to end LTE network latency is reduced significantly.

In [37], publish and subscribe system is applied to "Real-time Public Transit Tracking" ITS application. Applying Pub/Sub pattern to ITS applications has three challenges. First of all, vehicles continuously publish messages which creates a large amount of real time data. It can create scalability issues. Secondly, context aware messages including location updates have to be sent from each vehicle to a large number of subscribers. Final issue is that Pub/Sub system should be fault tolerant in case of drastic workload changes. This paper claims to resolve three issues resulting from Publish and Subscribe pattern. To evaluate Pub/Sub system, "Real-time Public Transit Tracking" ITS application is developed on Mobile Pub/Sub System (MoPS) and it is built over OpenStack which is an open source cloud platform. Architecture is consisted of Publishers/Subscribers, MoPS Broker and OpenStack. Brokers provide communication of publishers and subscribers. These brokers are deployed over virtual machines (VM) at physical nodes. Cloud network based on OpenStack houses VMs. In experiment, each vehicle sends current location at predefined intervals. Total of 10 VMs are used in experiment. Vehicles publish their position according to lines. For example vehicle in line number 1 sends messages to broker 1. In the first scenario, 1000 subscribers are subscribed to the same broker all the time but they change their subscriptions in the middle of the experiment. In the second scenario, 1000 subscribers reconnect to a new broker in the middle of the experiment. These subscription changes are done to simulate mobility of vehicles. In addition to that, messages are published per 60, 90 or 120 seconds in three cases to evaluate publication rate. The results show that end to end latency is around 175 ms when total number of publishers is 1000 in scenario 1 with 2500 subscribers. Publication rate does not change the latency for this case. End to end latency is around 225 ms for 5000 publishers when messages are published per 90 seconds. Latency increases to 275 ms when messages are published per 60 seconds. When the number of subscribers increases, end to end delay increases drastically. For example latency is 800

ms for 10000 subscribers when total number of publishers is 1000. It is very high compared to 175 ms latency for previous case with 2500 subscribers. As for scenario 2, around 325 ms latency is achieved with 2500 subscribers and 1000 publishers when messages are published per 90 seconds. Again, it is very higher than scenario 1 case. 1300 ms end to end latency is calculated when the number of subscribers is increased to 10000. The results show that proposed MoPS is able to handle ITS applications when the number of subscribers and publishers are large. Average end to end latency values show that as the number of subscribers and publishers increases, it is becoming hard to implement real time ITS applications. Secondly, this paper proves that MoPS can be deployed in OpenStack.

Numerous works exist in the literature about ITS communication. In [38], 3G is used with VANET to reduce average delay and overall performance. Some techniques to enhance the capabilities of LTE network for vehicular communication is proposed in [39]. USA and Japan are the leading countries focusing on research about ITS architectures. Although there are a lot of proposed architectures in the literature, it is still unclear which architecture works best for ITS applications. With improvement of technology and increasing interest of ITS all over the world, it can be seen that practical implementations will be carried out soon.

CHAPTER 3

PROPOSED ITS ARCHITECTURE

In this section, the general ITSVeCon architecture and its elements are explained in different aspects. After this explanation, OBU and server architectures and their software implementations are given.

3.1 Overview of Proposed ITS Architecture

Intelligent Transportation Systems (ITS) consist of many different components. ITS communication architecture defines the communication requirements and protocols for the distributed application components. The main aim is to provide integrity, coordination and reliability while maintaining the communication between these components.

A flexible architecture that we call ITSVeCon which maintains the communication between different end points while being suitable to real life conditions is introduced in this thesis. The architecture enables end-to-end communication of host devices through the dedicated ITSVeCon server which provides application layer switching of TCP connections. ITSVeCon is an IP-based architecture which describes the application layer protocols and message structures. We propose carrying out the link layer communication over cellular access and IP core network. Main units and the communication pattern are illustrated in Figure 3.1.

The host devices can be in-vehicle units such as Electronic control units (ECU) and On Board Units (OBU), infrastructure units such as Road side units (RSU), user devices such as computers and smart phones, and servers of third party service

providers. Electronic control units which are illustrated as EP1 and EP2 are located in vehicles. They are connected to in vehicle CAN Bus Network. Their main mission is to collect information from sensors located in different parts of vehicle and to control vehicle by sending commands to electrical subsystems. Electric Parking Brake Control Unit, Electric Vacuum Pump Control Unit, Speed Control Unit, Door Control Unit are main examples of electronic control units. On board units (OBU) are also

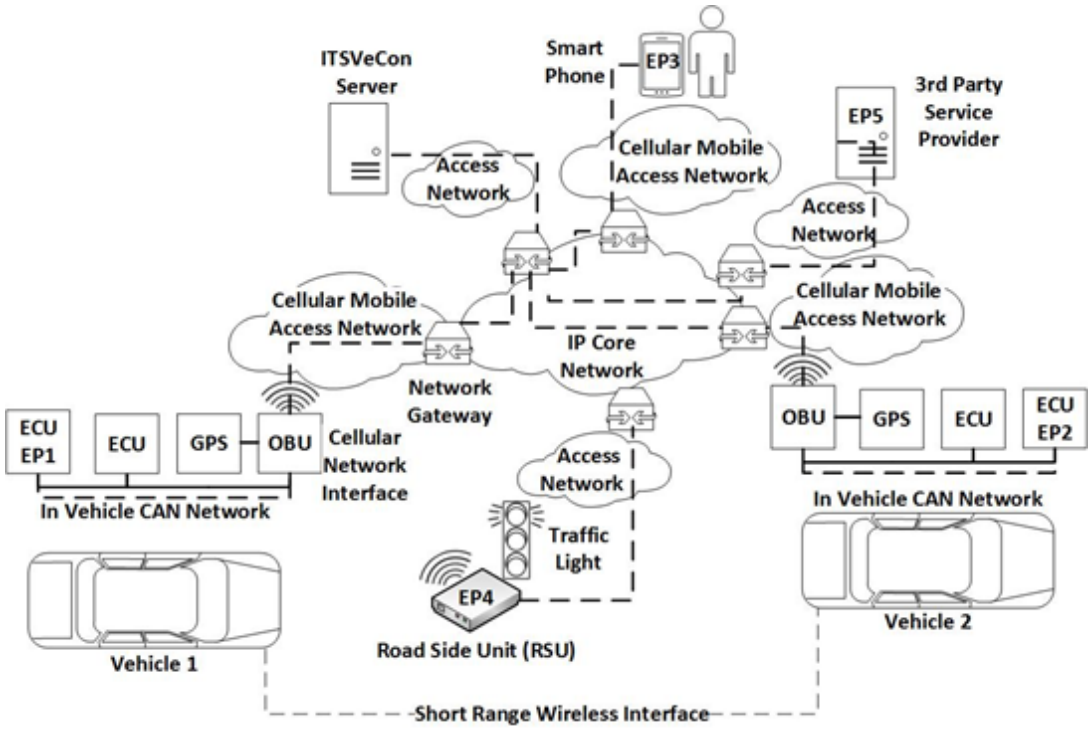

Figure 3.1: ITSVeCon Communication Architecture

located in vehicles. They are also connected to in vehicle CAN network. The responsibility of these units is to provide cellular network interface, to collect information from electronic control units and GPS and to run ITS applications on top level. It can also send commands to ECUs located in vehicle according to the application type. It is the gateway of the vehicle to the outside network. Communication of the vehicle with other end points are done by this component.

Roadside Units (RSU), 3rd Party Service Provider and smart phone are the other main components presented in this architecture. Communication is done over cellular access network and IP Core Network. IP Core Network provides path for exchanging messages over different sub-networks. In this architecture, these sub-networks are

access networks used by RSU, vehicles, smart phones, 3rd party service provider and ITSVeCon server.

In this proposed architecture, OBUs, RSU, smart phone and 3rd party service provider are always connected the ITSVeCon server via TCP/IP layer. Full duplex communication is established between the server and the end-hosts by Websocket protocol over a single TCP connection. This is crucial for real time applications because of the ability of instantaneous bi-directional communication. The messages can be sent to only 1 destination point (unicast) or they can be sent to multiple destination points (multicast). For real time applications, it is estimated that all of the components are in the same city which enables low communication delay. Here we note that short range wireless interfaces (SRWI) such as D2D or DSRC can be used together with ITSVeCon communication. In this case, this wireless communication is only realized between vehicles. Messages are transmitted directly without the use of the server which enables lower delay values. However due to coverage issues, sometimes messages may not be delivered to the destination point. For reliable communication, vehicles send messages from both cellular and short range wireless interface. This guarantees higher rates of packet reception ratio as well as best delay values. This type of communication is handled in this thesis as a functional test. SRWI is simulated as a second interface to show that OBUs can get data from multiple interfaces and make a selection. In order to give examples of communication types, end points (EP) are numbered as shown in Figure 3.1. Examples of communication types are given below.

- Communication between EP1 and EP2 (EP12): In this application, the ECU named as EP1 sends speed or brake information to actuator via CAN Bus. Since OBU in Vehicle-1 listens CAN Bus, it can obtain the CAN frame. After the CAN frame is obtained by OBU, if short range wireless interface is not enabled, the message is sent to only ITSVeCon Server. ITSVeCon Server examines the destination point of this message and sends this message to Vehicle-2. OBU in Vehicle-2 obtains this message and transmits it to CAN Bus. Finally, ECU named as EP2 in Vehicle-2 receives the message and read the speed and brake information transmitted by EP1 of Vehicle-1. If short range wireless interface is enabled, the message is also sent from wireless interface. Vehicle-2 uses

the first message that is obtained and ignores the other one. This application can be used as controlling the speed of Vehicle-2 in accordance with Vehicle-1. This communication can be classified in Security and Traffic Administration Applications.

- Communication of EP4, EP1 and EP2 (EP412): In this application, the frequency of traffic light EP4 is transmitted to the ECUs in Vehicle-1 and Vehicle-2 with the help of ITSVeCon Server and OBUs located in the vehicles. The color of the traffic light can then be displayed in the driver's dashboard. This application is an example of Security and Traffic Administration Applications.
- Communication of EP5, EP1 and EP2 (EP512): 3rd party service provider sends the information of weather and traffic conditions which was previously provided from the related RSUs. The information coming from the ITSVeCon server is displayed in driver's dashboard after obtained from OBU. Communication is broadcast so both EP1 and EP2 can obtain the information. This is classified in Automotive Infotainment Applications.
- Communication of EP1 and EP3 (EP13): In this application, speed and GPS location of Vehicle-1 can be displayed in user's smart phone with the help of OBU and ITSVeCon Server. Similarly a message that is sent from smart phone can be seen by the driver of the Vehicle-1 in dash board. This can also be listed in Automotive Infotainment Applications.

The main aim of this proposed architecture is to enable the communication between every component and to establish control over them. ITSVeCon server is the main control and switching mechanism while the other clients are responsible for sending and receiving necessary messages and taking actions accordingly. It is possible to increase the number of servers to provide better quality of service and to take security measures in case of any unexpected issues.

3.2 ITSVeCon Server Architecture and Software

ITSVeCon server is the main message switching center in ITSVeCon architecture. Every component is always connected to the ITSVeCon Server using Websocket pro-

tocol. The messages are not changed by server, server just adds the timestamp values of the time when the message is received and sent. Server is also responsible of authentication of clients. Unauthorized clients cannot communicate with other clients. In this proposed architecture, authentication is done by looking clients' user information IDs. This ID is license plate for vehicles, IMEI number for smart phones, and a predefined keyword for RSUs. There is a list containing all of the authorized clients' user information. This list is checked by the server when a new client wants to connect.

ITSVeCon server should always send the message to the desired end points to maintain reliable communication. In some ITS applications such as speed and brake control of the vehicle according to another vehicle, the transmission of the message should be unicast. However, in some cases, the message should be transmitted to more than just one vehicle. For example; a RSU may send the weather condition of the area to the nearest vehicles. In order to provide both unicast and broadcast messaging, the communication between the clients are carried out in Publish-Subscribe pattern by ITSVeCon Server. In this proposed architecture, all of publish and subscribe requests are sent to the ITSVeCon Server and server handles creating the channels and sending the messages.

For the communication in this architecture, the messages are formed in JSON format before sending to the ITSVeCon Server. JavaScript Object Notation (JSON) format is a collection of the human-readable name-value pairs which is used when transmitting data objects [40]. In this name-value pairs, value represents the content of the name. It can be an object, number, array, string, true, false or null. One or more of these name-value pairs constitute objects. JSON format is different from XML format and it is more practical to use because of its' readability. JSON format is a language independent format meaning that it can be implemented in various platforms. Considering the fact that there are many components in ITS architecture, this makes JSON format a perfect option for exchanging messages. In this proposed ITS architecture, the clients generate messages in JSON format first and then send them to the ITSVeCon Server. When receiving messages, clients parse the received message to obtain name-value pairs. Similarly, ITSVeCon Server parses the received message first and then it regenerates the message in JSON format be-

fore sending it to the clients. An example of the JSON format is given below:

```
{
“Folder”: “windows”,
“Number”: 20,
}
```

In this example; “Folder” is a name and its value is “windows” which is a string. “Number” is a name whose value is an integer 20.

3.2.1 Structure of JSON Messages

The communication between clients and the ITSVeCon Server is done by Websocket protocol, which creates a permanent full duplex communication over a single TCP connection. An example JSON message format of this proposed architecture is given below:

```
{
“Action”: “Publish”,
“UserID”: “06FJ6392”
“DestUserID”: “06AA06”
“ChannelName”:”Diagnosis”,
“ApplicationName”:”DiagnosisApp”,
“DestApplicationName”:”MaintenanceApp”,
“CANFrame”:”5A1#11.22.33.FF”,
“Lat”: 34.6543
“Lng”: 33.9543
“Counter”:1
“Info”: “Server”
}
```

Since Publish-Subscribe pattern is used, in addition to the context of the message, the clients need to define which type of action they want to perform. In order for the server to keep record of the clients in the channels, every client should have user ID.If the message is broadcast, channel name definition is enough for the server to send the message to the subscribers. However, if the message is unicast, then the “Destination

User ID" should be defined. Then the server just sends the message to the destination with destination user ID. It is still necessary that destination user is subscribed to the channel stated in message. More than just one application can run on the clients, so the clients also define the name of the application sending the message and the name of the application that should receive the message. In addition to these fields, more name-value pairs can be added to the JSON message depending on the application. For example; for sending the CAN messages, "CANFrame" name can be added to the JSON message and a real CAN message is added as a string in the value part. All of this name-value pairs are parsed by the ITSVeCon server and the necessary fields are obtained. Detailed explanation of name-value pairs is given below.

- **"Action":** Its' values constitute a command set for the server to accomplish. This name can take 6 string values. "AuthenRequest" value is firstly used by clients for authentication request. If the client is suitable for making a connection and if it is defined as a reliable source, the server puts "AuthGranted" value in the "Action" value and then sends it back to the client. For "AuthenRequest" value, other names except "UserID" is not important. Because of that, when client requests for authentication it just fills the "UserID" and "Action" names. Clients which want to subscribe to a channel, fill this name's value with "Subscribe". The server then looks for the value of the "ChannelName" name and performs subscription to the desired channel. If the channel does not exist, the server creates channel and then adds user to the subscription list. "Unsubscribe" value is used together with the value of "ChannelName" name to indicate that the client wants to unsubscribe from the defined channel. Then the server removes the user from the list of subscriptions of the related channel. "UnsubscribeAll" value is used for client to unsubscribe from all subscribed channels. This is generally used before the client disconnects from the server. "Publish" value is used to send necessary information to the channel defined by the value of "ChannelName" name. The information can be changed from application to application and it can be added as different name-value pairs. Since the communication can both support unicast and broadcast, after "Publish" value is received from the server, the server looks for the "DestUserID" name's value. If it is null, then server sends the message to all of the subscribers

in the channel. If it is not null and destination user is subscribed to the channel, the server just sends the message to the user defined in “DestUserID” name. To sum up, the values of the name “Action” and their meanings are illustrated in Figure 3.2.

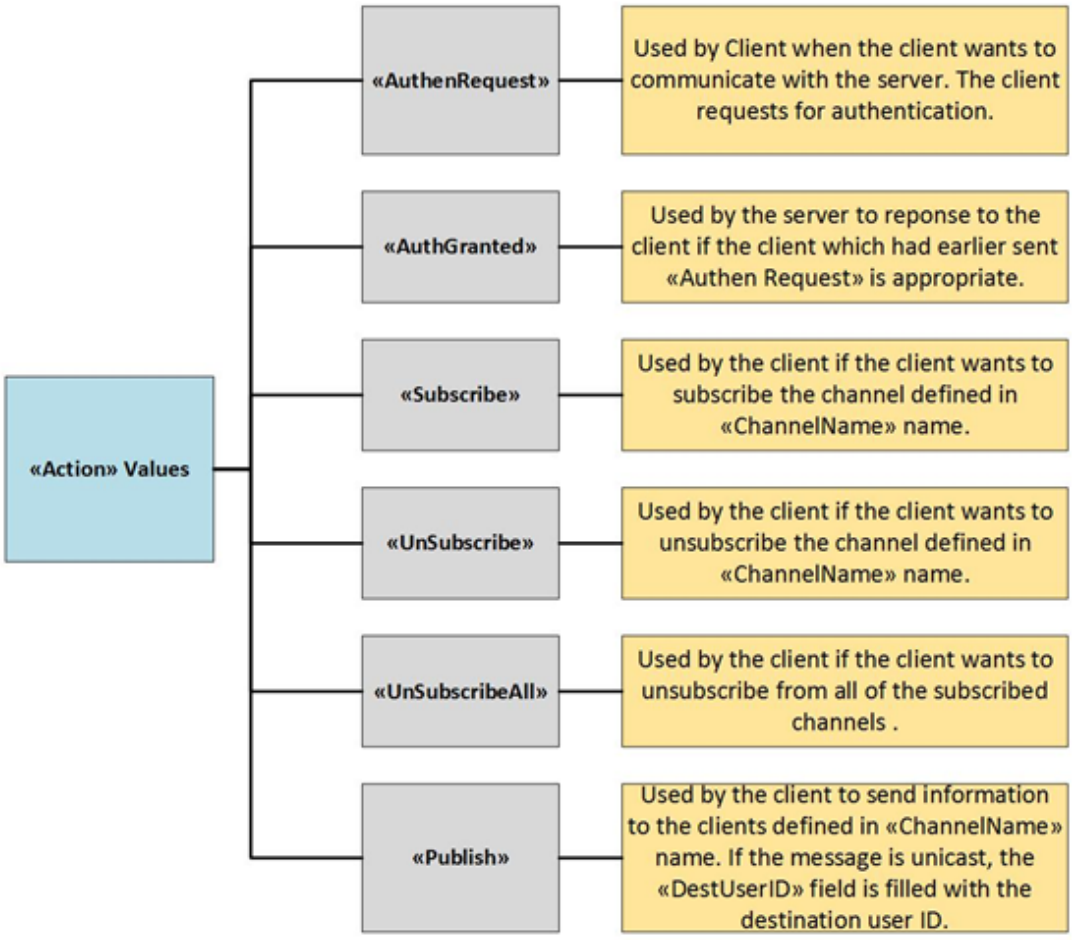

Figure 3.2: The Values of “Action” Name in JSON Messages

- **“UserID”:** The name “UserID” is used for the client to identify itself. In this proposed ITS architecture, every client should have user ID. Vehicles use their license plates and smart phones use their IMEI number as user ID.
- **“DestUserID”:** If the message is aimed to be sent to only one client, then the value of “DestUserID” name is filled with the user ID of the client which is expected to obtain the message.
- **“ChannelName”:** When the client makes a subscription or publish request to

the server, channel name should be defined. If subscription request is made, the server creates a channel if it does not exists and then adds the client to the subscription list. If the channel exists, the server just adds the client to the subscription list. Channel name is a string and it may be application dependent.

- **“ApplicationName”:** Since more than one application can run in each client, when sending messages, clients define the application which sends the message in “ApplicationName” field. For example, an application named “RoadInfo” may request road conditions from RSU and it is necessary for RSU to obtain the application information before preparing response message.

- **“DestApplicationName”:** This field is filled with the name of application running on destination client which is supposed to receive the message. Applications can send messages to other applications with the help of this name field.

- **“CANMessage”:** On Board Units which are located in the vehicles can send CAN messages obtained from ECUs in this field. CAN message format is a string. ’#’ character is used to separate CAN ID from the CAN data. For example; “5A1#11.22” means a CAN message with ID 0x5A1 whose data is 0x11 and 0x22 in hexadecimal form. If CAN message is not wanted to be transmitted, this field is left as null.

- **“Lat”:** This name is used to define the latitude of the client. It can be left as null if it is not used.

- **“Lng”:** This name is used to define the longitude of the client. It can be left as null if it is not used.

- **“Counter”:** It is used to count the number of specific messages sent to the server in some applications. For example in real time CAN frame transmission between vehicles, this value is increased by 1 for each CAN frame transmitted.

- **“Info”:** It is used to distinguish short range wireless interface and server messages in some of the applications. In ITS architecture, this area is filled with the value of “Server” when the communication is done over server. However for the communications done by short range wireless interface, this area is filled with the value of “ShortRangeWirelessInterface”.

3.2.2 Server's Algorithm

According to these JSON name-value pairs, ITSVeCon Server creates and deletes channels, adds clients to the channel or removes clients from channels, sends the message to appropriate destination points and maintain always-open connection between its' clients. The algorithm of the ITSVeCon server is represented in Figure 3.3 in flowchart format. As it can be seen from the figure, algorithm almost depends on name-value pairs encapsulated in JSON format. ITSVeCon server is implemented in Microsoft Visual Studio development environment using C# programming language. To parse encapsulated JSON messages, "Newtonsoft-JSON" external library designed for .NET framework is used. Newtonsoft-JSON is an open-source project offering libraries of serializing and de-serializing JSON objects [41]. When a message is received by the server, it is first de-serialized and value of the "Action" name is obtained. Then message is serialized again after adding necessary time-stamps and editing some necessary name-value pairs. Since there is not default Websocket implementation in C#, an external library "Websocket-sharp" is used to provide Websocket protocol. "Websocket-sharp" is an open-source library aiming to provide Websocket protocol to .NET framework [42].

ITSVeCon Server runs on port 80, which is default TCP/IP port number. The general server address starts with "ws://" indicating that it is using Websocket protocol. ITSVeCon Server has infrastructure of maintaining more than just one service. For example, while one service can serve for the ITS, other service can serve just for chat purposes. In this proposed ITS architecture, ITSVeCon Server has only one service and it is fully reserved to ITS algorithm. The service name is added to the total server address which makes total server address as "ws://IP_Address:Port_Number/Name_of Websocket_Service". When clients want to connect to the ITSVeCon server, they should know the IP address of the server as well as the service name. In this architecture, it is assumed that all clients know these information about server.

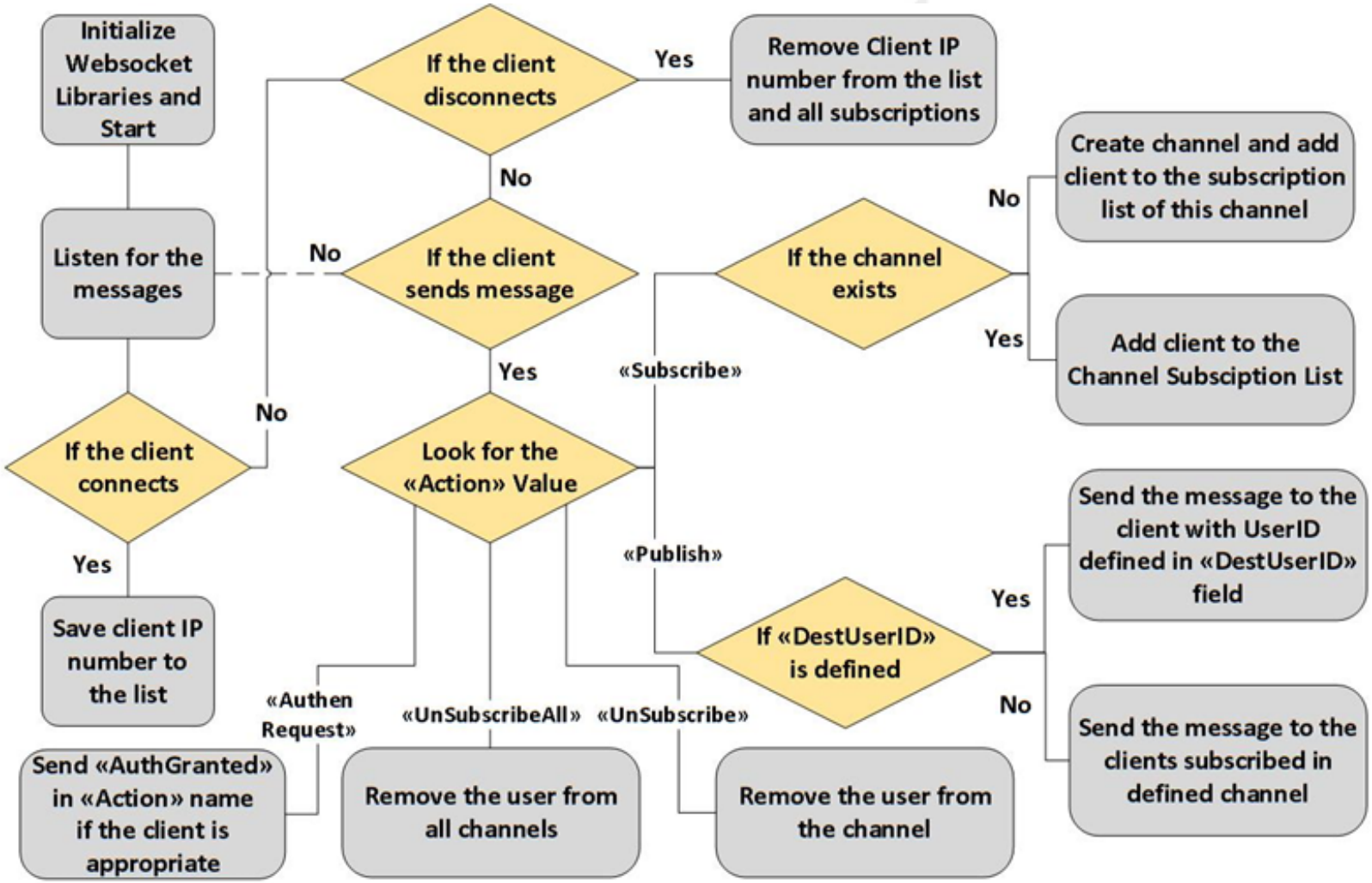

Figure 3.3: Flowchart of ITSVeCon Server

3.2.3 Server Timestamp Functionality and Synchronization

Meinberg program runs together with ITSVeCon server to maintain synchronization with NTP server. Meinberg is Windows version of NTPD and working algorithm is the same. ITSVeCon server has the ability to execute "ntpq -p" command to obtain current offset value. ITSVeCon server can log time stamps at reception and transmission of any Websocket messages. These time stamp values are recorded as text file when server ITSVeCon application stops. These time values can be used for measurement and prioritization purposes.

3.3 OBU Hardware and Operating System

In the scope of this thesis, “SABRE for Automotive Infotainment Based on the i.MX6 Series” development board is selected and used. The board and its’ interfaces can be seen in Figure 3.4. SABRE for Automotive Infotainment Development Board is

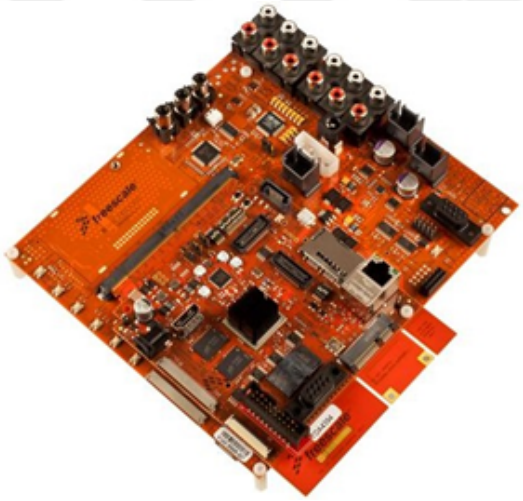

Figure 3.4: SABRE for Automotive Infotainment Development Board

designed to include the features necessary for developing automotive applications. Board support package (BSP) and demo images of both Linux and Android can be obtained via NXP’s website. This enables customization of the selected operating system as well as the developed applications running on top of it. For example initialization parameters of some interfaces, booting logo or user interface can be changed by changing BSP. The main features of SABRE-AI development board is as follows

[43]:

- i.MX 6 QUAD processor running up to 1GHz,
- 2 GB x 64 DDR3 running up to 532 MHz,
- 32 MB 16-bit parallel NOR flash,
- SD card interface, NAND flash socket,
- 1.5 Gb/s SATA interface,
- Ethernet interface,
- JTAG and UART interfaces,
- High and low speed CAN interfaces,
- High-Speed USB (OTG) interface,
- LVDS and HDMI interfaces,
- SPDIF receive interface,
- I2C module connector

Main lack of this board that it does not have cellular mobile interface and GPS. To establish 3G interface, external components such as "3G to Ethernet Modem" should be used. GPS can also be added if an external GPS board is connected to the UART interface.

For operating system, both Linux and Android operating systems can run on this board. Linux images are obtained in accordance with Yocto Project, a project that aims to create customized Linux operating systems [44]. The libraries and features have recipes which can be added or removed when creating the desired custom image. Developing Java applications in this customized Linux operating system is difficult and adding visual graphics to applications requires using extra libraries such as Wayland or X11. However developing Java applications and adding graphics features are practical in Android OS. Considering the fact that Android OS is a widely used operating system in Intelligent Transportation Systems, it is reasonable to select this OS.

It enables more customization compared to any other operating systems. Therefore in the scope of this thesis, Android 4.3 OS demo image which is given in NXP's website is used. This image contains base Android operating system without any additional features. Necessary developer settings have to be arranged first to install applications. ITSVeCon Application is implemented on top of this image in Android Studio, a development platform for Android applications.

3.3.1 OBU ITSVeCon Application

The application on OBU is a multi-layered application designed to run on Android operating system. This application is implemented mostly with Java libraries that are developed by Google for Android. This Android development framework comes with the Android Studio, which is an Android application compiler [45]. External Java and native (C/C++) libraries are also used to add some features that normally do not exist in Android development framework.

ITSVeCon Application aims to provide an infrastructure for the OBU that is located inside the vehicle while being connected to the CAN Bus, 3G cellular network and GPS. It is designed in such a way that if OBU is connected to a vehicle in real life, it can be easily modified and be ready to work as desired. In order to do so, a flexible structure is implemented allowing future developers to add as many applications as necessary. ITSVeCon Application is capable of sending and receiving CAN messages, communicating ITSVeCon Server using Websocket protocol, providing short range wireless interface protocol, logging message reception and transmission timings, running super user commands and allowing developers to add many applications that can run using the application's existing libraries. ITSVeCon OBU Application's being able to provide real CAN and Ethernet interfaces makes it usable in any vehicle. The layers in ITSVeCon Application is arranged such that every layer can only talk with upper or lower layer of itself. The architecture of the ITSVeCon application is given in Figure 3.5. The user is only interested with the "MainActivity" and "SettingsActivity" which are placed on top the ITSVeCon architecture. The layers at the bottom are responsible of controlling interfaces and simulations. The communication of the lowest layers with the top layers is done by "V2X Application Interface" and

"Distributer Starter" layers.

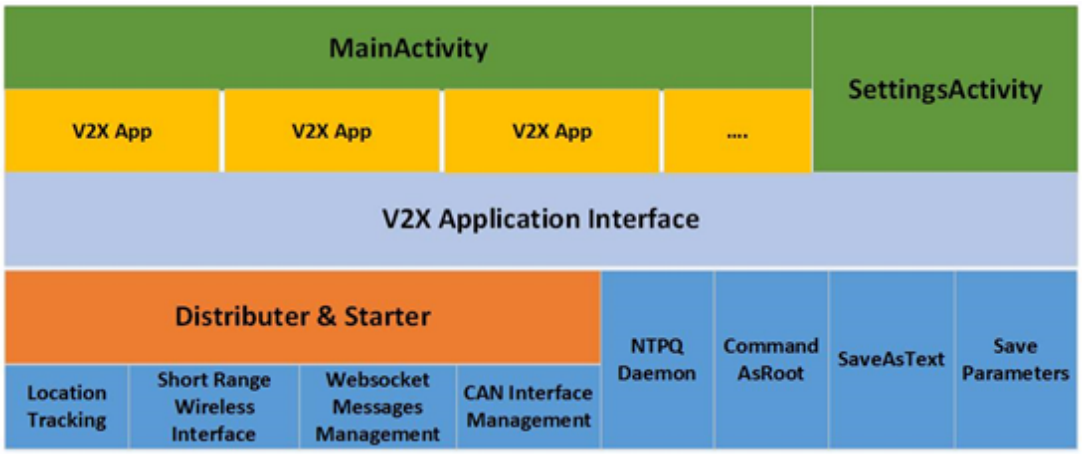

Figure 3.5: ITSVeCon application layers

Websocket Messages Management: This layer is used to communicate with ITSVeCon server. Since Websocket protocol and JSON message format are used in ITSVeCon algorithm, this layer makes sure that the message transmission requests coming from V2X Applications are converted to the right format. When a V2X application requests to send a message to the ITSVeCon Server, this layer gets the request with the necessary parameters and creates JSON message accordingly. When the server sends a message to the OBU, first this layer gets the message, and it directly passes the message to an upper layer, which is "Distributer & Starter". Application name, destination application name, user ID, destination user ID, channel name and "Action" values are the ones that should be included in every message. Extra name-value pairs are added to JSON messages according to V2X Application which requested to send message.

Android SDK normally does not provide Websocket library. In order to use Websocket protocol, an open-source library, Autobahn|Android is used. Bidirectional real-time messaging in Android applications using Websocket protocol are realized with the help of Autobahn|Android library [46]. When the application begins, this layer first tries to connect to the server. If it is successful, the authentication message is sent immediately. After authentication granted message is received, normal operation depending on callback functions start. These callback functions trigger events if

disconnection or message reception occurs. When V2X applications requests to send messages to server, “Action” value and timestamps are added after creating JSON message. After formation, this message is sent to the server. Figure 3.6 illustrates the algorithm.

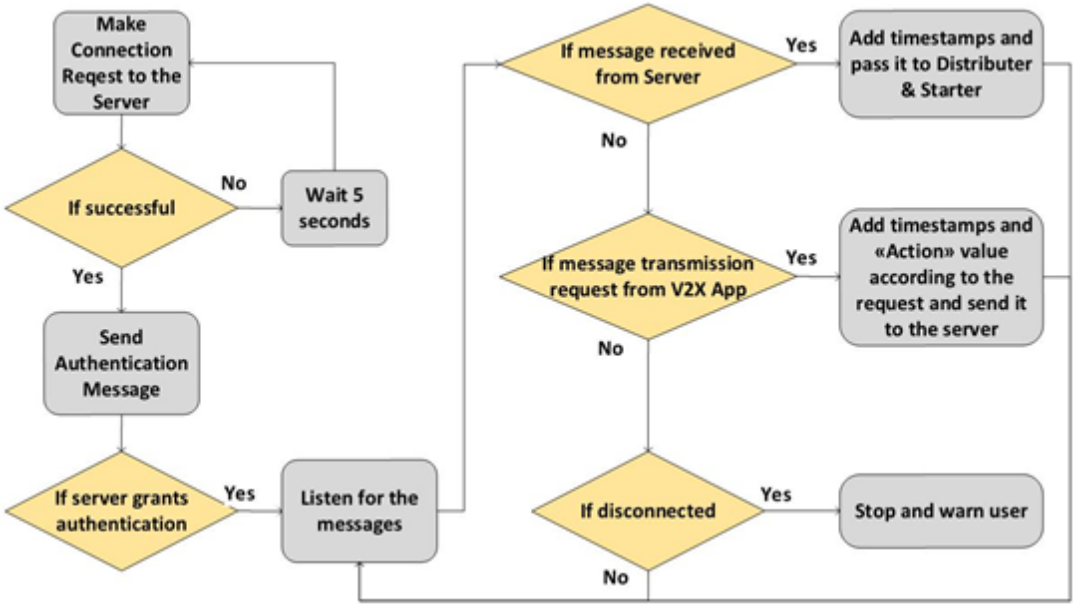

Figure 3.6: Websocket Messages Management Flowchart

CAN Interface Management: In ITSVeCon Application, CAN messages are transmitted and received using this layer. In NXP’s Android image, Linux kernel beneath the Android operating system is patched with SocketCAN open source CAN drivers to control CAN interface. Berkeley socket API, which is a Linux network stack is used by SocketCAN to implement CAN drivers as if it is a network interface [47]. SocketCAN supports extended, remote, error and normal CAN frames, local loopback, filtering, triple-sampling and CAN FD frames. All bitrates up to 1 Mbps can be selected using this driver. SocketCAN makes using C code mandatory because controlling this CAN interface requires opening, listening and closing sockets which is not included in Java programming. In order to use C code, Java Native Interface (JNI) programming is adapted to android coding, which enables using C/C++ coding in Java programming language [48]. Initialization, reading and transmitting CAN frames are written in C code. In this layer’s Java part, the functions written in C code is called by using wrapper functions.

Linux-can-utils which is a collection of open source tools developed by Volkswagen and Bosch to control SocketCAN drivers using iproute2 tools [49]. The commands listed below are used to initialize, start and stop CAN interface.

- Before starting the CAN interface, the bitrate must be defined. “ip link set can0 type can bitrate 125000” is used in command line to define bitrate.
- CAN interface is started by using the command “ip link set can0 up”.
- CAN interface is stopped by using the command “ip link set can0 down”.

These commands are called by ITSVeCon Application with the super user privilege. Triple sampling, loopback and listen-only mode are disabled by default, so it is not necessary to call the commands related to them before starting CAN interface.

When a CAN frame is read by JNI layer, it is automatically transferred to the Java layer and “Distributer & Starter” layer respectively. When a V2X Application wants to send a CAN frame, it calls the wrapper function in Java layer. Afterwards, CAN frame is transmitted by the JNI layer. If CAN filtering is used, JNI layer does not receive the CAN frame unless the received frame’s CAN ID matches with the filtered CAN ID. ITSVeCon Application supports filtering CAN frames according to their ID’s, however in the scope of this thesis, filtering is not used in JNI layer. CAN frame filtering happens at “Distributer & Starter” which is an upper layer of “CAN Interface Management”. Initializing, reading and sending CAN frames in ITSVeCon Application’s layered structure is illustrated in Figure 3.7.

Location Tracking: This layer is used to earn location detection ability to ITSVeCon Application. SABRE for Automotive Infotainment Board which is used for OBU hardware, does not have default GPS. Because of that, a mechanism getting mock location updates is realized in this layer. When vehicle’s location changes by 10 meters, an event is triggered by Android Location API. This layer listens to this event and informs “Distributer & Starter” layer when the event is triggered.

Short Range Wireless Interface: This layer is used for functional testing to prove that OBU ITSVeCon application can receive data from multiple interfaces. SRWI is simulated in this layer just like a second arbitrary interface. By doing this, the effects

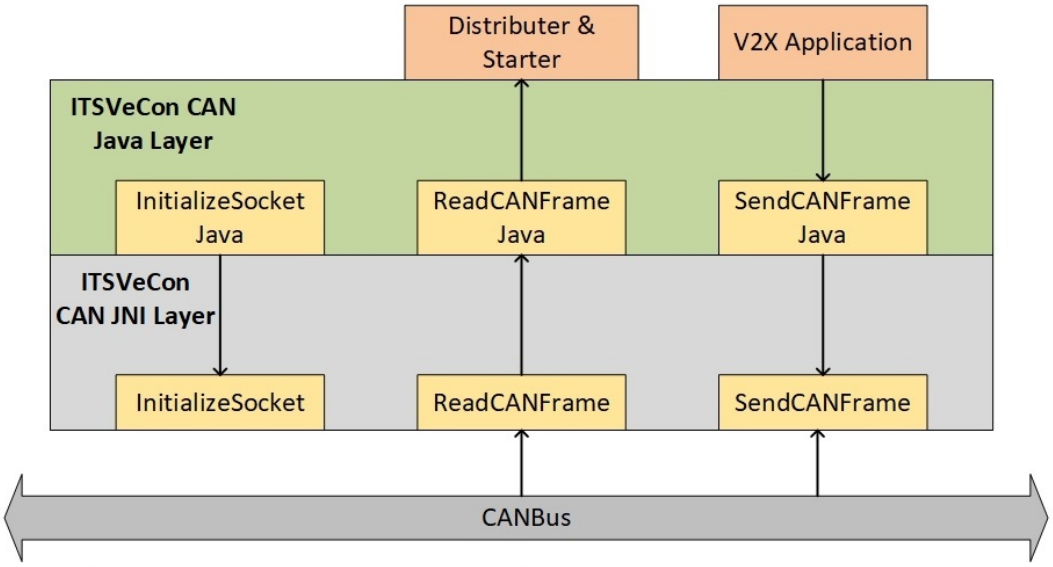

Figure 3.7: ITSVeCon Application CAN Interface Management Structure

of implementing a short range wireless communication such as DSRC/WAVE or D2D and the functionality of making them to work together with cellular interface can be seen. In order make these interfaces comparable, delay values of the second interface should be selected close to the cellular network interface. The aim is to think about the future when short range wireless interface delay values and cellular network delay values such as 5G are comparable.

Normally ITSVeCon Application sends all Websocket messages to ITSVeCon Server when this interface is disabled, so all communication are done over ITSVeCon Server. In "Settings Activity" layer, there is an option to enable short range wireless interface. If it is enabled, this layer starts working. As it can be seen from Figure 3.8, this layer generates JSON messages like normal ones except "Info" field and sends them with predefined interval. V2X Applications understand that if the message comes from short range wireless interface by looking to the value of "Info" name in JSON message. When a Websocket message is received from server, onMessageReceived() function is called in "Websocket Messages Management". This layer calls the same function for generated virtual messages. However, for wireless communication, sometimes packets cannot be delivered to destination. Because of that, this function is called with a certain probability to adapt this situation. 3 differ-

ent probabilities are selected as %30, %60 and %100. These three probabilities are used for every measurement when Short Range Wireless Interface (SRWI) is enabled. With this configuration, all cases can be examined in one measurement experiment. Predefined time interval is be selected according to measured end to end delay values.

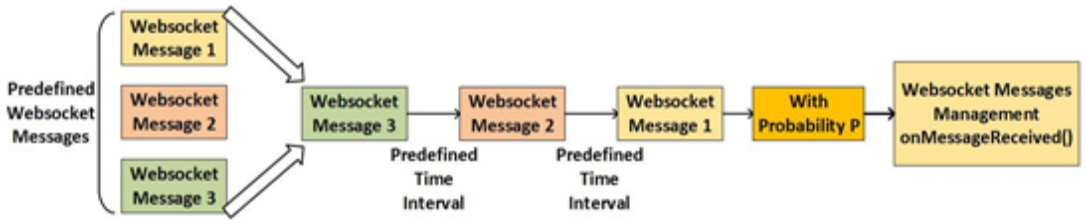

Figure 3.8: Short Range Wireless Interface Packet Generation

CommandAsRoot: This layer is used to execute commands necessary for the "CAN Interface Management" layer.

SaveParameters: ITSVeCon Server address, port number, CAN bitrate and other settings are saved by this layer.

SaveAsText: Timestamp values at reception and transmission of CAN frames and Websocket messages are saved in text format by this layer.

NTPQ Daemon: : The responsibility of this layer is to get offset values from NTPD daemon running in the background by executing "ntpq –p" command.

Distributer & Starter: The link between the layers located at the bottom and the "V2X Application Interface" layer is established with the help of this layer. Received CAN frames and Websocket messages are distributed to the correct V2X application here. A subscription mechanism is used to determine the application which should get the message. The application which is designed to get a specific CAN frame, subscribe to this CAN frame's ID at initialization. Distinguishing CAN frames are done by looking to their ID's. "Distributer & Starter" layer keeps a table of every application that subscribes to specific CAN ID's. After CAN frame is received by "ITSVeCon CAN Interface Management" layer, this table is used to convey the frame to the correct application. The path of CAN frames starting from CAN Bus ending

in V2X applications is show in Figure 3.9. On the other hand, Websocket messages

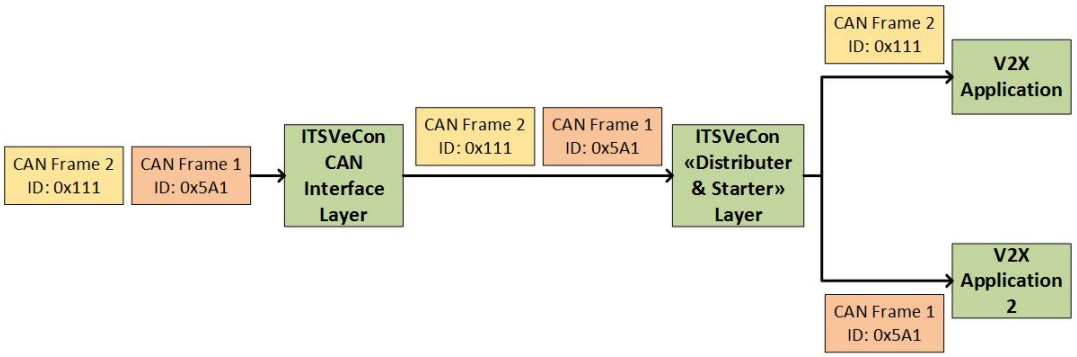

Figure 3.9: Distributing CAN Frames

are distributed by looking to the value of "ApplicationName" name which is defined in JSON format. Since every application has a name in string type, this layer parses the received JSON message, looks "ApplicationName" field and then decides which application gets which message. A matching table like CAN frames is used for Websocket messages. The process is show in Figure 3.10.

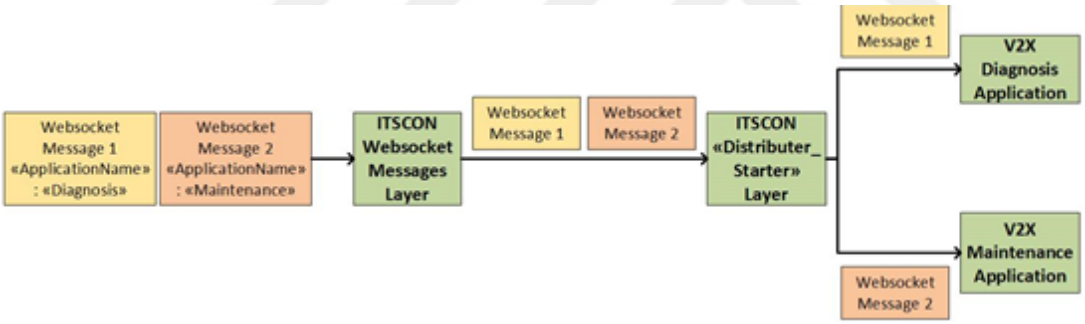

Figure 3.10: Distributing Websocket Frames

V2X Application Interface: This layer presents abstract functions for V2X applications to implement. Initialization, reception of CAN frames and Websocket messages and an application specific loop are provided to develop V2X applications. As it is shown in Figure 3.11, initialization, start and stop functions are the same for all V2X applications. Reception of CAN frame and Websocket message functions, location updates as well as application specific loop function are different for each V2X application. Applications running on the top are developed by implementing these specific functions.

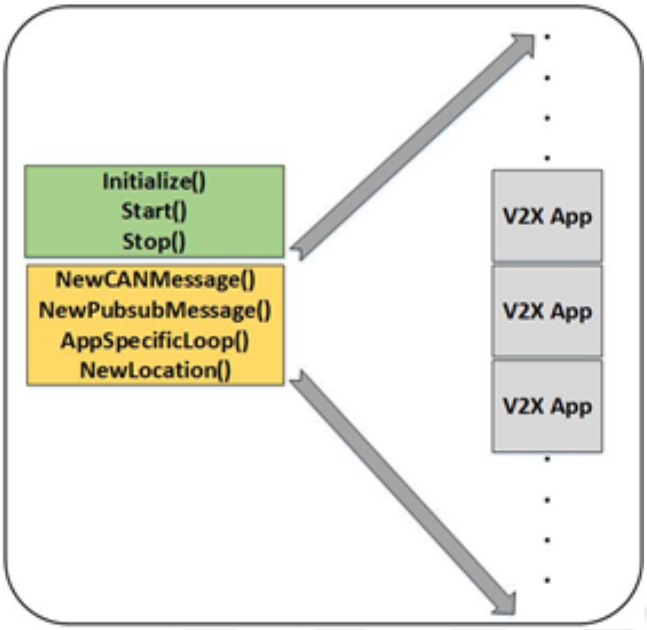

Figure 3.11: V2X application base functions

V2X Applications: In this layer, many applications can run simultaneously in separate threads. According to CPU's performance, the number of applications may be higher in some platforms. V2X applications are developed by extending "V2X Application Interface" layer. Initialization, start and stop functions are used from this interface. Necessary initialization steps can be added separately for each application, which allows flexible usage. In this layer, reception of CAN frames and Websocket messages, application specific loops and location updates are implemented. It is not obligatory to implement all of the functions defined in "V2X Application Interface". For example, some applications may not need location updates. Then in order not to receive location updates, it is enough not to implement the related function.

Application name, subscription of CAN frames and location updates and application specific loop delay are defined at initialization stage. Applications may work independently or they can communicate with each other over "Distributer & Starter" layer.

Main Activity: It is the layer that provides user interface together with the Settings Activity. This layer provides the buttons for starting and stopping application.

Settings Activity: This layer presents the settings necessary at ITSVeCon Applica-

tion's initialization step to the users. ITSVeCon Server's IP address, selection of V2X applications to run in application, CAN bitrate and user ID (license plate) are defined and saved in this layer.

3.3.2 Development of V2X Applications and Operation of SRWI

Since the main aim of this thesis is to provide end to end message delivery between two ECUs in different vehicles, necessary applications are developed for this purpose. As OBU ITSVeCon Application allows usage of many V2X applications running together, two different V2X applications are implemented for transmitting and receiving messages. These applications are used for real time CAN frame delivery, so they can be used for any ITS application related to safety such as vehicle tracking. When implementing applications, it is assumed that ECUs in both vehicles transmit and receive speed information so that the architecture presented in Figure 3.1 can be achieved.

"Speed Controller" is used to obtain the CAN message, encapsulate it in JSON format and send it to ITSVeCon Server. First of all this application subscribes to related CAN frame IDs that is transmitted by Ixxat's first CAN interface. After authentication, "Speed Controller" subscribes to channel with channel name "RealTime". After that, it starts to listen CAN frames. If a CAN frame is received and its ID matches with one of the subscribed CAN IDs, this CAN frame is delivered to this application by "Distributer & Starter" layer. The time and offset values of reception are recorded as text file with the help of "SaveAsText" layer. Application, destination application, destination user and channel names are added to created JSON message. CAN message is added as string to JSON message. For example "5A1#11.22" means a standard CAN frame with 0x5A1 ID and 2 bytes of data which are 0x11 and 0x22. "Action" name's value is "Publish" since message is aimed to be delivered to the destination user. Unicast message transmission is realised by ITSVeCon server because destination user name field is filled. The value of "Info" is "Server" meaning that this message will be delivered to the server. This name-value pair is necessary for the application runs on OBU2 to identify whether the message is coming from server or short range wireless interface. Counter starts with 0 and increases by 1 for every

subscribed CAN message transmitted. After JSON message is created, it is sent to ITSVeCon server. Timestamp and offset values of transmission time is saved by calling function from “SaveAsText” layer. Flow diagram of this application is shown in Figure 3.12.

Since “Sabre for Automotive Infotainment” board does not have wireless interface, the messages sent through wireless interface are simulated. Short Range Wireless Interface layer presents at the bottom of the layered structure of OBU ITSVeCon application and implements this simulation. However, in order to receive and use these messages, a V2X application needs to be developed on top of this layered architecture. Enabling SRWI interface is done for functional testing.

The second application “Tracker” is used to obtain the received JSON message, extract “CANMessage” name’s value and send this CAN message through CAN interface together with the ability to receive SRWI messages. This application can run in two modes. In the first mode, JSON messages are received only from ITSVeCon Server which is normal operation. In the second functional mode, short range wireless interface (virtual second interface) is enabled. Therefore both ITSVeCon Server and “Short Range Wireless Interface” layer sends JSON messages including the same CAN messages in the same order to this application. "Tracker" application starts thread after receiving the first CAN message. SRWI packet generation is done with the algorithm explained in Figure 3.8. “Tracker” decides which one to receive by investigating “Counter” value. While the message with lower counter value is received, the other message is discarded. Mode selection is done in “SettingsActivity” layer by user. Figure 3.13 illustrates the architecture of “Tracker”.

While enabling SRWI is done in "Settings" layer, starting thread in "SRWI layer" is done by "Tracker" application. Therefore SRWI is enabled in the OBU running "Tracker" application.

3.3.3 OBU Timestamp Functionality and Synchronization

Normally Freescale Android image does not support NTP synchronization. Synchronization can be done with the help of NTPD application. First of all Android image is

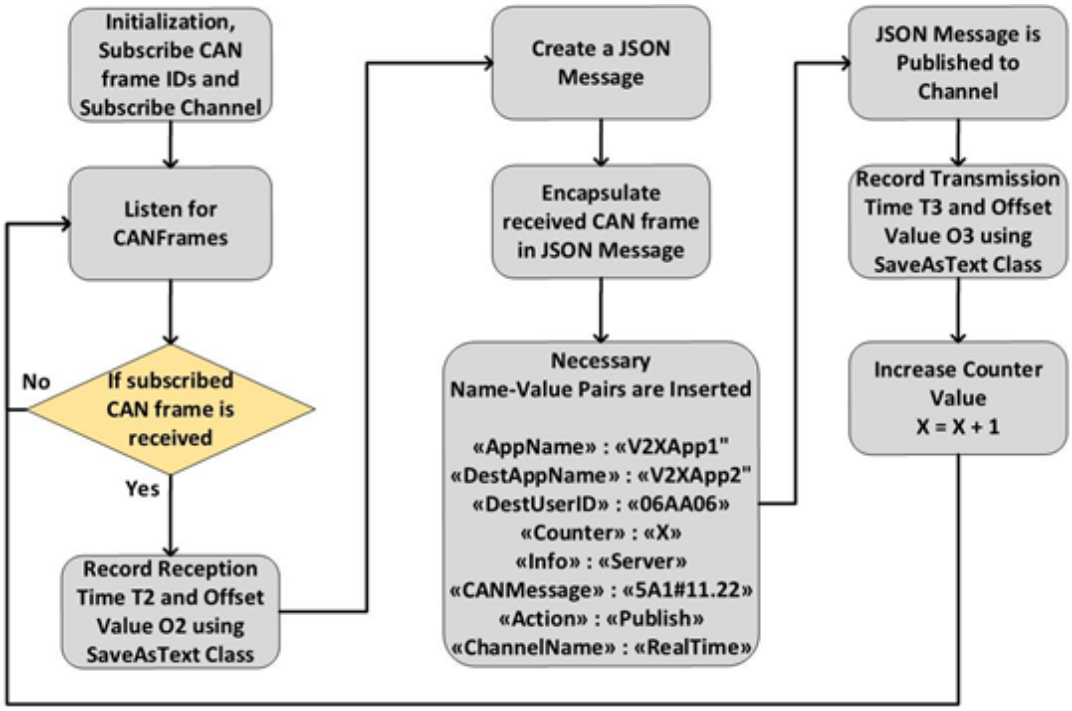

Figure 3.12: Speed controller flow diagram

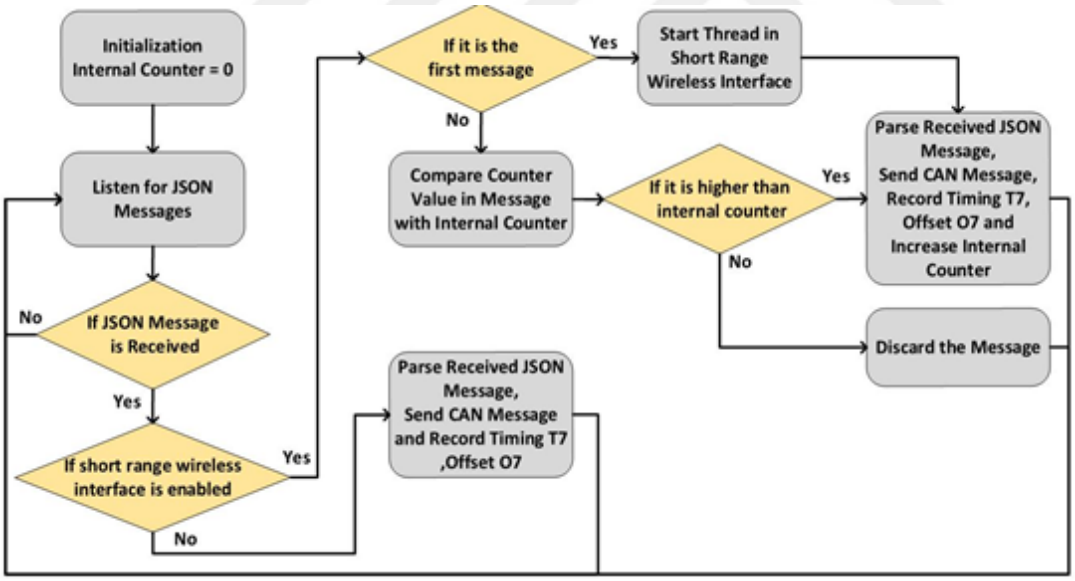

Figure 3.13: Tracker flow diagram

patched with NTPD source code to be able to synchronize OBU time with the selected NTP server. NTP synchronization process is started with "./ntpd -g -N -c ntp.conf" command. NTPD usage is described at "Section 2.5 Time Synchronization".

OBU ITSVeCon application has the ability to record timings at reception and transmission of CAN frames and Websocket messages. Timestamp values are obtained and recorded whenever a Websocket or CAN message is received or transmitted. In addition to that, the difference of OBU time with respect to NTP server is recorded as offset time. "NTPQ Daemon" layer runs "ntpq -p" command to obtain offset values. Then these offset values are conveyed to V2X applications running on top of ITSVeCon application. These timings and offset values are saved with the help of "SaveAsText" layer. "Speed Controller" saves time stamps at CAN frame reception and Websocket message transmission. "Tracker" application saves time stamps at CAN frame transmission and Websocket message reception.

3.4 Contributions Compared to CarCode ITS Architecture

Kaan Çetinkaya's M. Sc. Study [5] was the first step towards ITS Communication Architecture which utilized the same hardware platform, WebSocket protocol and application level switching. To this end, we clearly state the contributions of this thesis with respect to the proposed CarCoDe architecture in [5].

CarCoDe was designed as a proof of concept work to show that server OBU communication can be done local area network and it provides an infrastructure for adding new CarCoDe applications. This thesis however, focuses on measuring the real time end to end delay in real life while realizing real interfaces. The novel contributions of this thesis with respect to [5] are summarized below and explained in detail in related sections.

Using Real CAN Interface: In CarCoDe application, sending and receiving CAN messages are implemented with BCM server, which uses network stack as CAN interface. Therefore the effects of receiving and sending real CAN messages cannot be observed. Virtual CAN interface is controlled by sending commands over local Ethernet interface, which increases message traffic between server and OBU. Because of controlling virtual CAN interface by sending commands from server, delay measurement of messages is not possible. BCM server is always running on the background using the command line, which makes CPU usage inefficient. When CAN frames are

received, these frames are obtained by parsing the command line, which adds extra time when reading and parsing command line strings. Sending of CAN messages are just like receiving which requires executing commands. In Android operating system, executing commands in command line takes long time. That is the reason why it is not recommended to control fast interfaces like CAN.

In the scope of this thesis, CAN interface is controlled such that OBU can work in a real environment like a vehicle. Sending and receiving real CAN frames are supported. Socket type programming in JNI is used which decreases command time significantly compared to executing commands in command line. Transmission and reception of RTR, extended and normal CAN frames are supported. The bitrate can be adjusted in Settings Activity. ITSVeCon Application CAN Interface Management Layer is converting the CAN frames into strings when received which makes it easier to parse their ID in “Distributer & Starter”.

Publish & Subscribe Pattern: The server conveys the messages by looking just destination ID fields in CarCode architecture. This enables unicast messaging and limits the use of broadcast messages. OBU has to send a lot of messages to the server when broadcast messaging is intended. In addition to that, OBU has to know all of the destination user ID’s which makes it impossible and insecure. In ITS applications, only authorized controlling mechanisms should know every user’s information.

In proposed ITS architecture, publish and subscribe pattern is used in communication between components. Publish and subscribe pattern is the best choice for ITS applications. There are many free and paid services which provides publish and subscribe infrastructure and API in real world. Publish and subscribe enables unicast and broadcast messaging. In the scope of this thesis, this pattern is used for communication of end points. When user subscribe to specific channels, ITSVeCon Server creates a matching table of which user is subscribed to which channel. Only ITSVeCon Server knows the user ID’s which makes the architecture secure. When there are more than 2 users in the channel and if "DestID" field is null, ITSVeCon Server sends the message to every user and this makes the message broadcast. When the destination ID is specified in the message, server just sends the message to the user with that ID if the destination user is in the same channel. This makes the communication

unicast. ITSVeCon Server sends the incoming messages to the subscribers respectively. However there are different server solutions which can send the message to the subscribers simultaneously. Crossbar, an open source networking platform, uses Web Application Messaging Protocol (WAMP) which is a sub-protocol of Websocket together with Publish and Subscribe pattern [50]. Crossbar is capable of sending the message to the subscribers simultaneously. Since ITSVeCon architecture uses publish and subscribe pattern, end to end delays can become even smaller if the server is replaced with the one like Crossbar.

Synchronization of Time: In proposed ITS architecture, every user that is connected to server is also connected to the NTP server. The main motivation for the synchronization in the scope of this thesis was to collect timing measurements. However, such capability can be used in time stamping of messages for real-time applications. End points can have different time clocks which can cause problems. Every user in ITS architecture add timestamp values of the time when the message is received and transmitted. If there is a considerable amount of time difference between users, it may result inconsistency in some applications. For example, user U1 sends a message to user U2. The time when user U1 sends the message is T1. The time when the user U2 receives the message is T2. In real life, T1 must be smaller than T2. However, if user U2's system clock is wrong, T2 may be smaller than T1. This causes problems, especially with the applications requiring sensitive timing measurements. In the future, it is expected that 5G will be widely used in ITS applications. It means that even 10 ms time interval can be a big deal soon.

In the scope of this thesis, it is proposed that every end user is also connected to the NTP server. If the connection of end user is stable and the delay between NTP server and the user is low, time difference between NTP server and the user becomes very small. Considering the fact that 1 ms offset value can be achieved with 8 Mbps ADSL connection in Ankara, it will not be hard to obtain the same values when 4G or 5G is used in On Board Units located in vehicles. In [51], it is proposed that OBUs in each vehicles need to synchronize perfectly by using a GPS receiver to achieve low jitter TDMA implementations in ITS-G5 standard.

Virtual Second Interface (SRWI) working with the server communication: In

the scope of this thesis, a preliminary implementation of short range communication is simulated for V2V applications to support server communication. In CarCode architecture, every message is conveyed by the server. Delay values can be high if the connection between the server and the end point is not good. Therefore to see the effects of adding a second interface (wireless communication) to the architecture, short range wireless interface should also be implemented. OBU can understand if the message comes from the server or short range wireless interface. Selection between these messages are done by looking to counter value. This enables to compensate for the packets lost in wireless communication. In this architecture, it is guaranteed that the messages are received by the destination end point. Change of delay values are recorded and compared with the normal operation. This is a functional test aiming to show that OBU application can work both with cellular network and a second interface such as DSRC, D2D or another SRWI.

Configuration of OBU Application: In CarCode application, the configuration of the application is done by changing XML file. XML file editing is hard and requires an editor. In ITSVeCon Application Settings Activity that has a user interface is used to configure application. This makes it easy for standard users to edit.

Reducing Internal Delay in OBU Application: OBU ITSVeCon Application coding is changed effectively to reduce the internal delay. V2X applications running on top can send Websocket and CAN frames faster compared to the old architecture. Application loop delay is reduced in V2X applications and some polling functions are converted to event triggered functions which is more suitable for real time applications.

V2X Applications Running on Top: In OBU ITSVeCon application, the applications running on top are implemented to measure end to end delay. In addition to that, applications serve different purposes and they can be used in real life.

Real Communication over Internet: In CarCode application, only local area network is used for communication between OBU and the server. In ITSVeCon Application, OBU and server are connected to internet using different access networks. On Board Units connect to the server over internet which enables to observe real life usage and delay measurements. Port forwarding is used in ADSL router to forward the

message coming from internet to ITSVeCon Server in private local area network by using router's Network Address Translation (NAT) feature.

CHAPTER 4

EVALUATION OF ITSVECON ARCHITECTURE

This thesis aims for exploring the feasibility of communication over cellular access network and IP core network to achieve V2X communication. To this end, an important research question is "Is it possible to use such an architecture for real-time vehicle applications". This section explores different test scenarios to answer this question.

4.1 Usage of Developed V2X Applications

The communication can both be unidirectional or bidirectional and measurements will be carried out accordingly. For unidirectional communication, only ECU in vehicle 1 sends speed information to ECU in vehicle 2 presented in Figure 3.1. For bidirectional communication, both ECUs simultaneously transmit speed information. “Speed Controller” is the name of the application that transmits speed information as Websocket message to ITSVeCon Server. “Tracker” is the application that gets this speed information from ITSVeCon Server. Therefore “Tracker” runs in the vehicle which follows other vehicle. These both applications run in each vehicle if the communication is bidirectional.

To sum up, "Speed Controller" is used in OBU1 and "Tracker" runs in OBU2 if the communication is unidirectional. Both "Speed Controller" and "Tracker" runs in each vehicle if communication is bidirectional.

4.2 Experimental Setup

Some ITS applications, especially the ones concerning about safety, require low end to end delay. Because of that, delay should be measured accurately and it should represent real life conditions. Since the most important safety communication in ITS applications is between vehicles, the delay between two electronic control units (ECU) located in different vehicles is measured in this experiment setup. These electronic control units are shown in the previous figure, Figure 3.1, which shows the entire architecture. Here it is important to note that the mobility effects are not present as this is a laboratory experiment.

In this experimental setup, delay between the time when EP1 sends a CAN frame and the time when EP2 receives the same CAN frame is measured. For example, EP1 in vehicle 1 sends the CAN frame which commands the brake control module to reduce the speed of the vehicle. This CAN frame is transmitted to the entire CAN network in vehicle 1. OBU which is also connected to the same CAN network in vehicle 1, receives this frame and sends it to the ITSVeCon Server after converting it in JSON format. ITSVeCon Server transmits this message to OBU in vehicle 2. OBU parses the received JSON message and obtains CAN frame. OBU sends the same CAN frame to the CAN network in vehicle 2. Finally EP2 in vehicle 2 receives this CAN frame and commands the brake control module accordingly. The path of the message transmission between two end points is shown in Figure 4.1. As it can be seen, end to end communication includes many components. This message transmission shown in this figure is unidirectional. It is also possible that EP2's transmitted CAN frames are also delivered to EP1 in vehicle 1. Therefore this delivery can also be bidirectional.

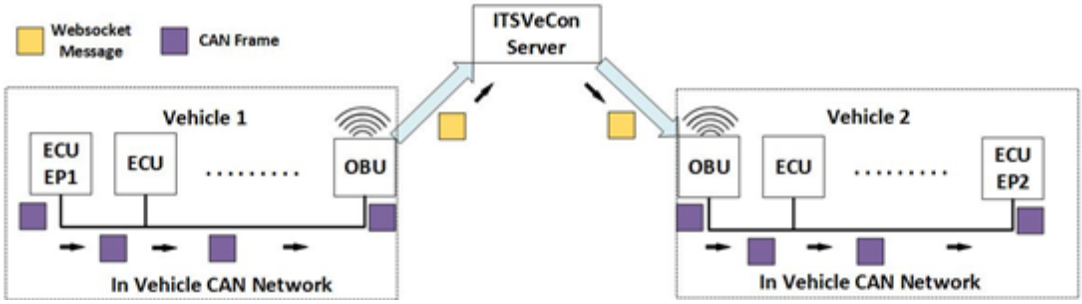

Figure 4.1: CAN frame transmission unidirectional path

Secondly, functional testing is carried out by enabling short range wireless interface. This is done with the help of "Tracker" application and "SRWI" layer in OBU ITSVe-Con application.

In experimental setup, to simulate electronic control units (ECU), Ixxat USB-to CAN converter is used. Ixxat USB-to-CAN V2 product supports two high speed CAN interface and USB 2.0 interface that is used to connect to host computer [52]. The product is shown in Figure 4.2. A software is developed for the USB to CAN converter to send and receive CAN messages and to record timings.

Figure 4.2: Ixxat USB to CAN converter

On Board Units have Ethernet interface which can be used to connect to internet by cable. Since OBUs are used in vehicles, internet access can only be realised by using cellular mobile access like 3G, 4G or 5G. To gain cellular mobile interface to OBUs, TP-Link's 3G/4G USB modem to Ethernet/Wireless converter and Turkcell VINN USB modem are used together. TP-Link TL-MR3020 is capable of providing 150 Mbps wireless and Ethernet speed while using 3G/4G USB Modem as source [53]. TL-MR3020, Turkcell VINN USB Modem and their connection to OBU are shown in Figure 4.3.

In ITS architecture, all of the components should have the same time source. To achieve this, every component is connected to the same NTP server. In this experimental setup, NTPD daemon runs at background in both On Board Units. Similarly, Meinberg program runs at background in Server computer. If the number of components using NTP increases, it would be difficult to decrease time difference between

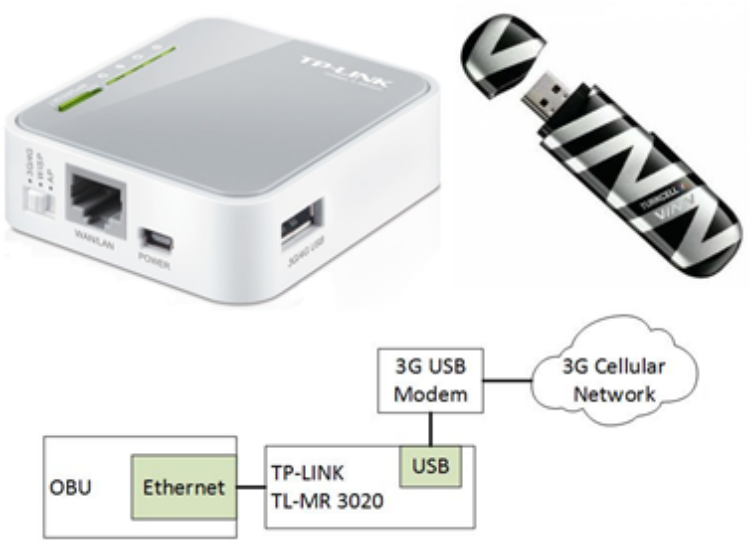

Figure 4.3: Providing 3G interface to OBU

them. To prevent that, Ixxat USB to CAN converter is connected to the ITSVeCon Server machine. This reduces the number of components using NTP and makes time domain of ITSVeCon Server and Ixxat USB to CAN Converter the same. As a result of ITSVeCon computer's high speed internet connection, end to end delay measurement is very accurate. Experimental setup is shown in Figure 4.4. For simplicity, TP-Link 3G USB Modem to Ethernet Converter and USB modem are not shown in this figure.

There are many timestamp values that need to be recorded for this measurement. All timestamp values are explained below.

- **T1:** The time when Ixxat's first CAN interface sends CAN frame representing ECU in vehicle 1.
- **T2:** OBU1 is going to obtain this CAN frame at $T2$. Then OBU1 is going to encapsulate this CAN frame in JSON format.
- **T3:** This is the time when OBU1 sends JSON message to ITSVeCon Server over Websocket protocol.
- **T4:** It is the time when ITSVeCon Server obtains JSON message.
- **T5:** ITSVeCon Server sends this JSON message to OBU2 at time $T5$.

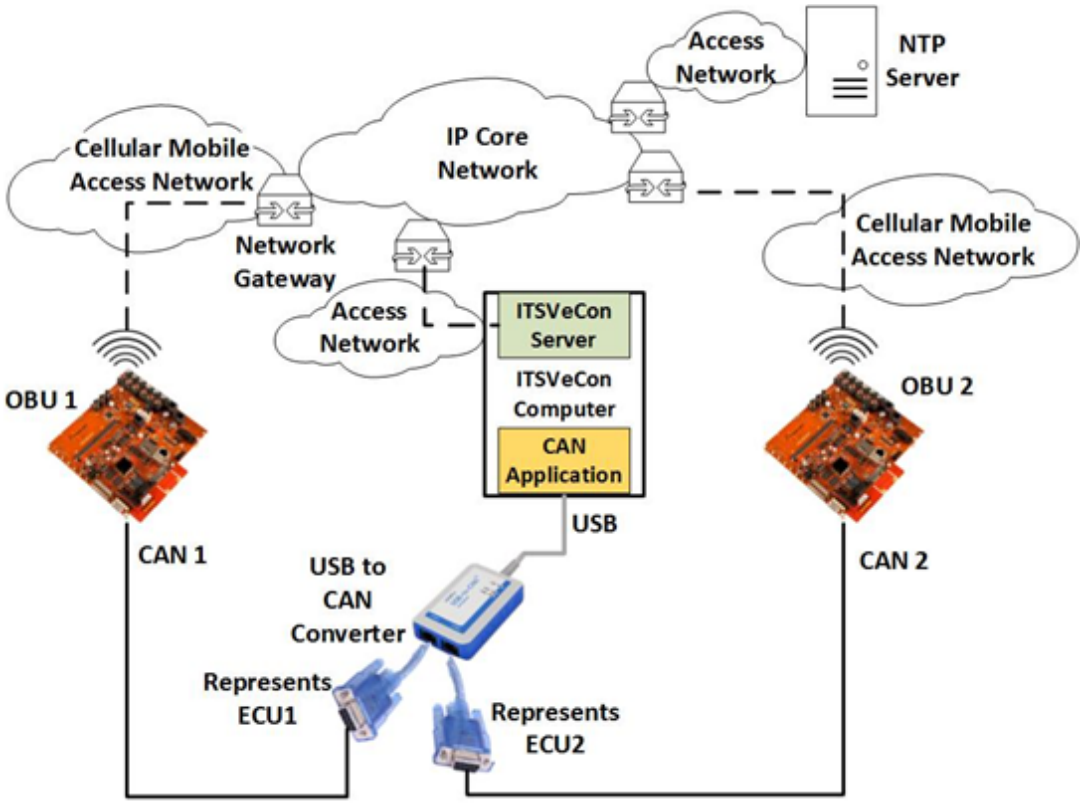

Figure 4.4: Experimental Setup with Communication Over 3G

- **T6:** OBU2 obtains JSON message from ITSVeCon Server at time $T6$. After parsing this message, OBU2 is going to extract the CAN frame.
- **T7:** OBU2 sends CAN frame at time $T7$.
- **T8:** Ixxat's second CAN interface representing ECU in vehicle 2 obtains CAN frame at time $T8$.

According to these values, total delay is $T8 - T1$. Delay in OBU1 is $T3 - T2$ while delay in OBU2 is $T7 - T6$. ITSVeCon Server delay is $T5 - T4$. $T4 - T3$ and $T6 - T5$ are network delays. $T2 - T1$ and $T8 - T7$ are CAN frame transmission delays. All timings are illustrated in Figure 4.5.

Since Ixxat is connected to ITSVeCon Server computer, there are total of 3 time domains. End to end delay and the delay between OBU and ITSVeCon Server do not need synchronization because necessary timestamp values are obtained from the same computer. Similarly internal delays of OBUs and ITSVeCon Server do not need syn-

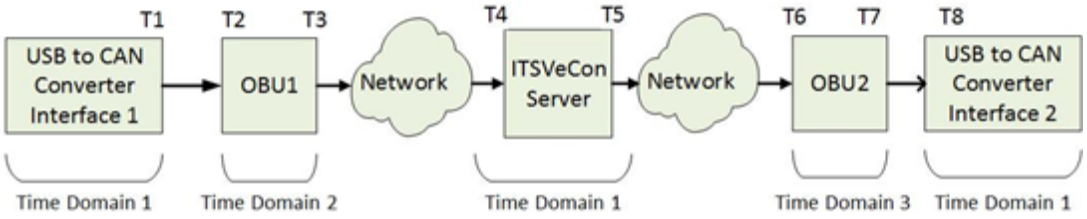

Figure 4.5: End to End Delay Components

chronization. NTPD maintains synchronization with NTP server well when internet connection speed is sufficient. However, if the connection speed is not good enough, offset values become large resulting in poor synchronization with NTP server. $T2 - T1$, $T4 - T3$, $T6 - T5$ and $T8 - T7$ delays are the most difficult ones to measure. For $T2 - T1$, time domain 1 and time domain 2 need to be synchronized perfectly. Same condition is valid for time domain 3 and time domain 1 when measuring T8-T7. When OBUs are connected to internet by 3G, offset values become too large for sensitive measurement because of weak connection. GSM Operators in Turkey do not provide stable 3G connection with high speed. Efficient synchronization with any NTP Server is not possible with these connections. Even 50 ms offset values can be observed, which is not suitable for measurement.

To overcome this situation, a simple method is developed. First of all, these delays are CAN transmission and reception delays meaning that they does not change according to the type of OBUs' internet connection. Therefore it is assumed that these delay values stay the same. Second experimental setup is used to measure CAN transmission and reception delays between OBUs and Ixxat CAN interfaces. In this setup, ITSVeCon Server and OBUs are connected to the same ADSL modem which is connected to internet. Offset values can become lower than 1 ms since it is a stable connection with 16 Mbps speed. Network delays stated as $T4 - T3$ and $T6 - T5$ are measured over local area network (LAN). Delay measurement applications running on OBUs are started when all components' offset values are less than 1 millisecond. This setup is used for measuring $T2 - T1$ and $T8 - T7$. These delay values are used later when OBUs are connected to internet with 3G. By using these values, $T4 - T3$ and $T6 - T5$ can also be calculated. Experimental setup for measuring CAN frame transmission and reception delays is shown in Figure 4.6.

If short range wireless interface is enabled, then only $T4$ and $T5$ values are not used because the message is generated directly in OBU2 and ITSVeCon Server is not used. $T8-T1$ value is the most critical one because it shows the effects of using short range wireless interface together with 3G communication.

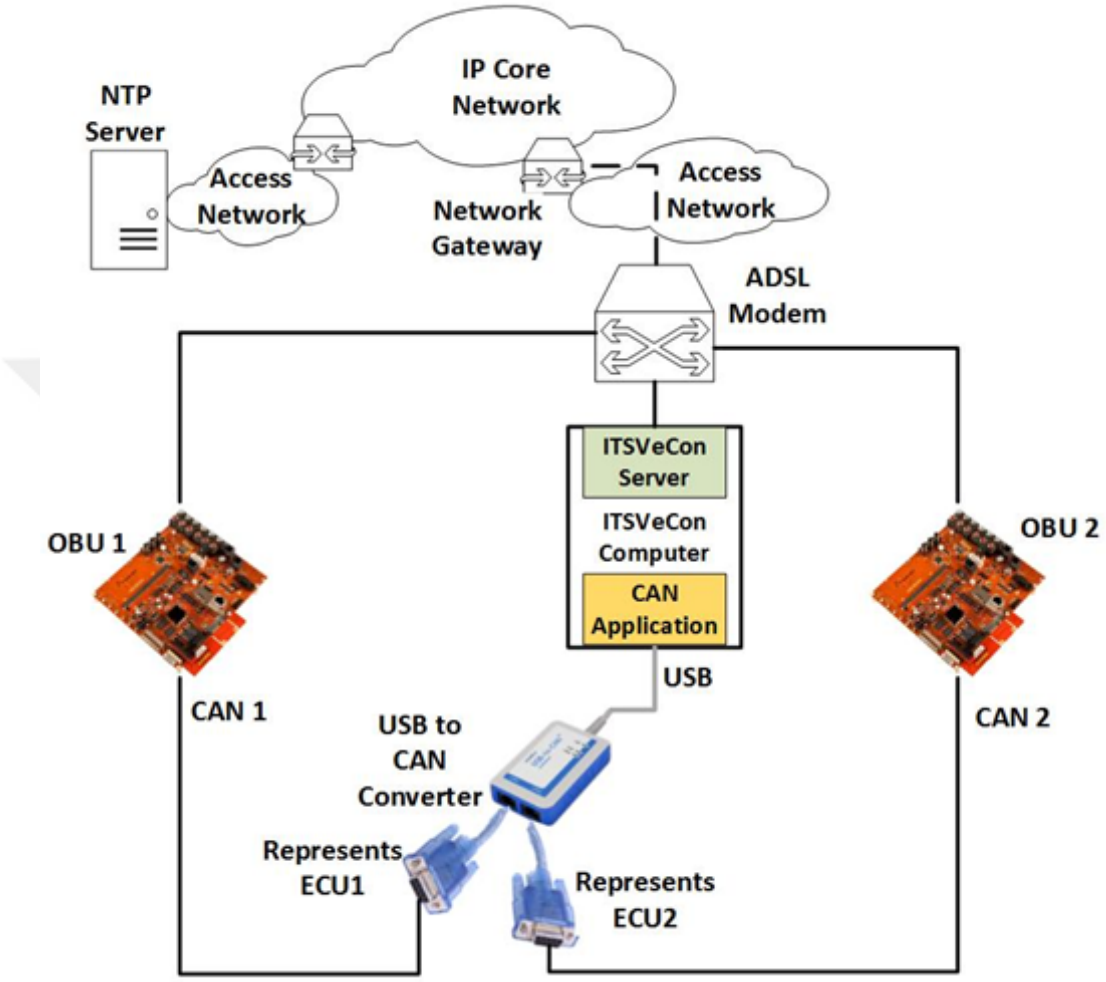

Figure 4.6: Experimental Setup with Communication Over LAN

4.3 Ixxat CAN Application

Controller Area Network (CAN) application also runs on ITSVeCon Server machine. This application sends a series of CAN messages from their CAN interfaces. If unidirectional communication delay is to be measured, then CAN messages are sent from only 1 CAN interface. For full duplex communication delay, same CAN messages

are sent from both CAN interfaces simultaneously. Total count of messages are stated in a text file which can be changed. Similarly, time interval between transmission of 2 successive CAN frames can be changed in a text file. Timestamp and offset values of transmitted and received CAN messages are recorded. This time values can then be parsed and all of the necessary values can be obtained. Different bitrate values as well as normal, remote, extended CAN frames are supported in this application.

Total count of frames, the frames that are going to be sent from channel 1 and interval between frames are obtained from text files. The frames defined in "CANTxMessages" text file are sent in a loop. When total frame count equals to the total CAN frames that have been sent from interface 1, application stops. For example, 2 CAN frames with ID's 0x111 and 0x222 are defined in "CANTxMessages" text file and total count of frames is defined as 100 in "CANTxTotalCount" text file. 50 CAN frames with ID 0x111 and 50 CAN frames with ID 0x222 will be sent from interface 1. When sending CAN frames, receive thread runs in background. When unidirectional communication is selected, CAN frames are received from interface 2. When a CAN frame is received, callback function is called and timestamp and offset values are recorded with the received CAN frame in text format. Similarly, timestamp and offset values are recorded when each CAN frame is sent.

If bidirectional communication is selected, CAN frames are sent from both interfaces simultaneously. Reception thread runs for both interfaces. CAN frames received from interface 1 and interface 2 are recorded in different text files. Similarly, CAN frames transmitted are recorded in different text files according to interface number. Flow diagram of Ixxat CAN Application's transmission process is shown in Figure 4.7.

4.4 Sample Log Outputs

Sample record of a transmitted and received CAN frames measured in second experimental setup where all components are connected to the same ADSL modem is explained in this section. Sample record from Ixxat CAN Application is given below. Time intervals between each component in transmission path is calculated in this section.

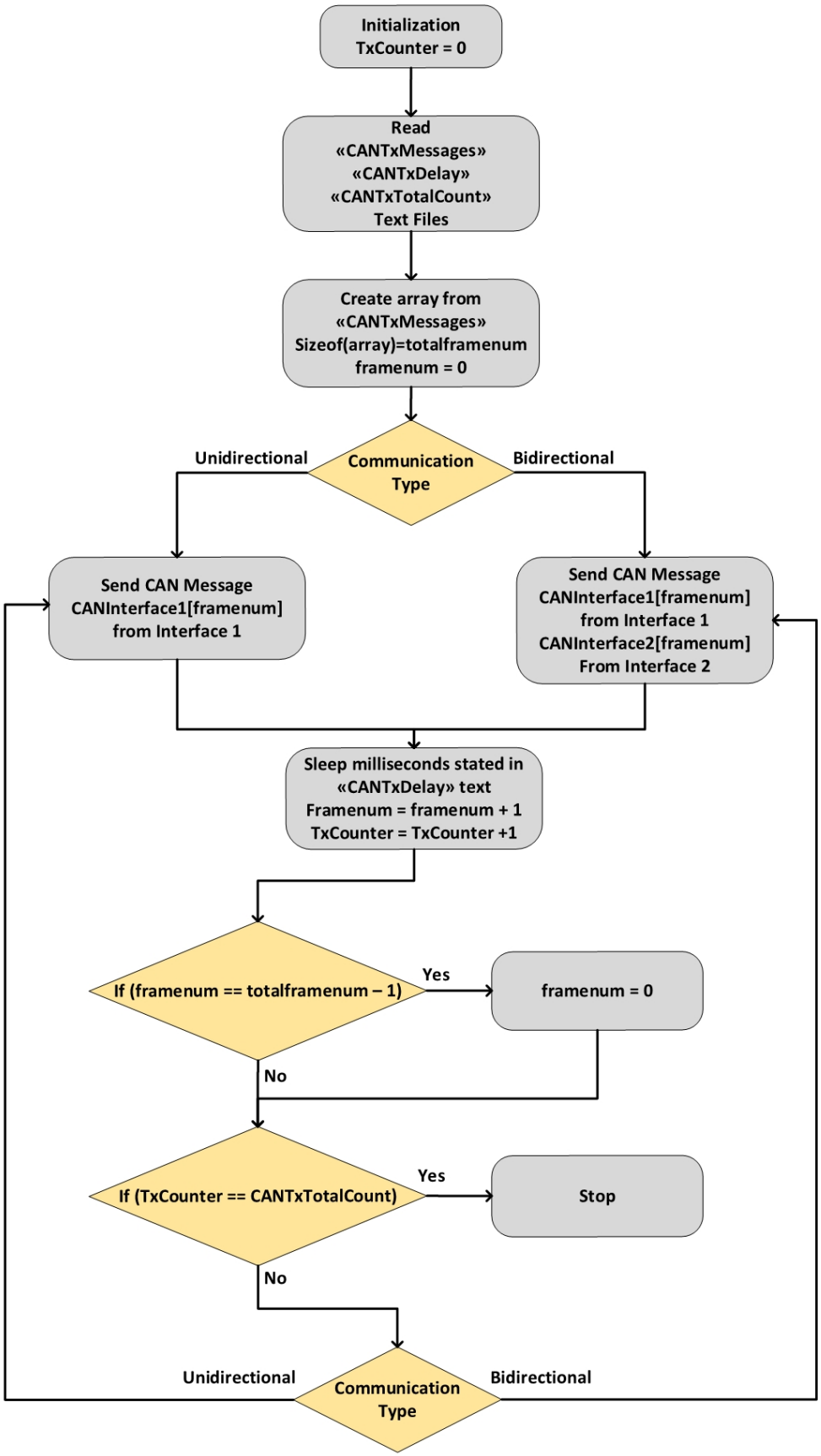

Figure 4.7: CAN Application Transmission Flow Diagram

CANTx_Chan1_Time1: 1491589748586 Offset: 0,074 Counter: 0 Message: Data [5A1] Dlc=4 11 22 33 44

CANRx_Chan2_Time8: 1491589748600 Offset8: 0,074 Count: 0 Message: Data [5A1] Dlc=4 11 22 33 44

The first record indicates time T1 which is the time when Ixxat's first CAN interface sends CAN frame with 0x5A1 ID and 0x11, 0x22, 0x33, 0x44 data. 1491589748586 represents milliseconds since 01.01.1970. Offset value is 0,074 milliseconds which is very low. Counter's being 0 means that it is the first message transmitted. Second record shows the reception time of same CAN frame from Ixxat's second CAN interface. 1491589748600 shows the time in milliseconds since 01.01.1970 when CAN frame is received.

"Speed Controller" saves timings as follows. Time T2 is the time when CAN frame is received. Time T3 is the time when JSON message is sent to the server.

Time2: 1491589748588 Offset2: -0.411 Time3: 1491589748590 Offset3: -0.411 Count: 0 CANMes: 5A1#11.22.33.44

Server's sample time record is given below. Time T4 is the time when JSON message is received. Server sends JSON message to the destination user at time T5.

Time4: 1491589748592 Offset4: -0,116 Time5: 1491589748593 Offset5: -0,116 Count: 0 CANMes: 5A1#11.22.33.44

"Tracker" receives JSON message at time T6. After CAN frame is obtained from encapsulated JSON message, it is sent to CAN Bus at time T7.

Time6: 1491589748595 Offset6: -0.426 Time7: 1491589748598 Offset7: -0.426 Count: 0 CANMes: 5A1#11.22.33.44

Real time value is found by summing offset and time value. All values are calculated below.

$$T1 = (1491589748586 + 0,074) = 1491589748586,0744$$
$$T2 = (1491589748588 - 0,411) = 1491589748587,589$$
$$T3 = (1491589748590\check{}0,411) = 1491589748589,589$$
$$T4 = (1491589748592\check{}0,116) = 1491589748591,884$$
$$T5 = (1491589748593\check{}0,116) = 1491589748592,884$$
$$T6 = (1491589748595\check{}0,426) = 1491589748594,574$$
$$T7 = (1491589748598\check{}0,426) = 1491589748597,574$$
$$T8 = (1491589748600 + 0,074) = 1491589748600,074$$

Interval calculations are listed below.

$$T2 - T1 = 1.515$$
$$T3 - T2 = 2$$
$$T4 - T3 = 2,295$$
$$T5 - T4 = 1$$
$$T6 - T5 = 1.69$$
$$T7 - T6 = 3$$
$$T8 - T7 = 2.5$$

This is the calculation method of timing intervals. Total end to end delay is 14 ms represented by $T8 - T1$. Since it is measured over local area network, this value is normal.

4.5 NTPD Configuration

It is important that NTP server is located at a close position because of low delay. Because of that servers included in “tr.pool.ntp.org” are considered to be selected. Since there are many servers in this project ping values are measured for each one of them. It is found that the server with IP address “212.50.1.11” gives the lowest ping delay. Therefore this server is selected as main NTP server for all components in the experimental setup. Same “Ntp.conf” file is used for both NTPD program running in OBUs and Meinberg program running on NTP Server machine. With “iburst” keyword, burst of 8 packets are sent to NTP server when it becomes unreachable. Minpoll and maxpoll values are 3 which is the lowest values meaning that time value is requested from NTP server for every $2^3 = 8$ seconds. "Ntp.conf" file is given below.

```
statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable
server 212.50.1.11 iburst minpoll 3 maxpoll 3
restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery
restrict 127.0.0.1
restrict ::1
```

4.6 Detailed Measurement Method

In this section, detailed measurement method is explained. Delay measurement depends on time recordings. Timings are recorded for different experimental setups and different settings. First of all, delay is measured when both OBUs and ITSVeCon server machine are connected to the same local area network. $T2 - T1$ and $T8 - T7$ values are obtained from this setup. After that OBUs are connected to the internet with 3G cellular network. The values of $T2 - T1$ and $T8 - T7$ which were obtained earlier are used for this measurement.

Eight arbitrary CAN frames are selected to be sent from Ixxat's CAN interfaces. These CAN frames are written in "CANTxMessages" text file for Ixxat CAN application running on ITSVeCon Server machine. Ixxat's CAN interfaces represent ECUs in vehicles. It is assumed that these ECUs send CAN frames periodically for every 100 ms. Therefore these predefined 8 different CAN frames are sent with 100 ms interval successively. Since confidence interval is calculated after measurement, the number of delay samples should be high enough to reflect real results. Therefore total of 500 CAN frames are transmitted from Ixxat's one CAN interface for every measurement. This is arranged by writing 500 to "CANTxTotalCount" text file. Due to the type of communication, receive thread can run for only 1 interface or both interfaces. These 8 CAN frames are sent in a cycle loop until total count of transmitted messages reaches 500. If unidirectional communication delay is to be measured, CAN frames are sent from both interfaces. CAN frames and the algorithm

is illustrated in Figure 4.8.

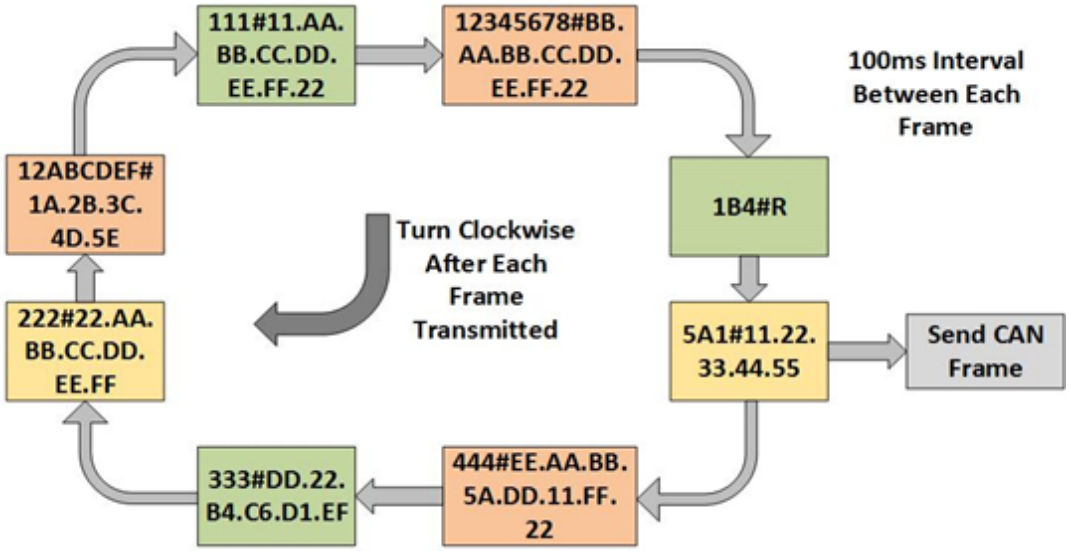

Figure 4.8: Transmitted CAN Frames from CAN Application

4.6.1 Unidirectional Measurement in Experimental Setup with Communication over LAN

Step by step measurement method is explained below.

1. First "Experimental setup with communication over LAN" is used. Both OBUs and ITSVeCon Server is connected to the same ADSL modem, so OBUs are connected to server by local area network. TP-Link TL-MR3020 and 3G USB Modem are not used.

2. NTPD daemon is started in both OBUs by “ntpd –g –N –c ntp.conf” command. “ntp.conf” file is given in previous section. After entering this command, clock of both OBUs are started to be synchronized with NTP server.

3. Meinberg program is started in ITSVeCon Server machine. Same “ntp.conf” file is used to connect to same NTP server.

4. After waiting 30 minutes, offset values of both OBUs and Server computer is checked. If they are all below 1 ms, applications can be started.

5. ITSVeCon Server Application and CAN Application is started in ITSVeCon Server machine. Server application is illustrated in Figure 4.9.

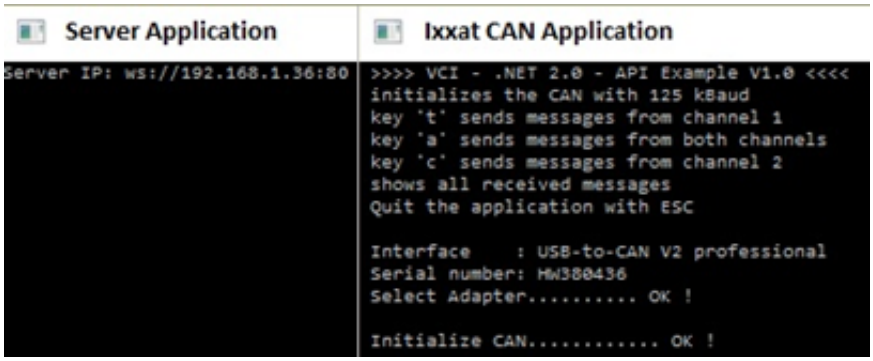

Figure 4.9: Server and Ixxat CAN Application

6. OBU application is started in both OBUs. "Speed Controller" is selected from settings in OBU1. "Tracker" is selected in OBU2. After starting, these applications are connected to ITSVeCon server. These applications are ready to receive, transmit CAN frames and Websocket messages. Server console and Ixxat CAN Application are shown in Figure 4.10 and Figure 4.11 respectively.

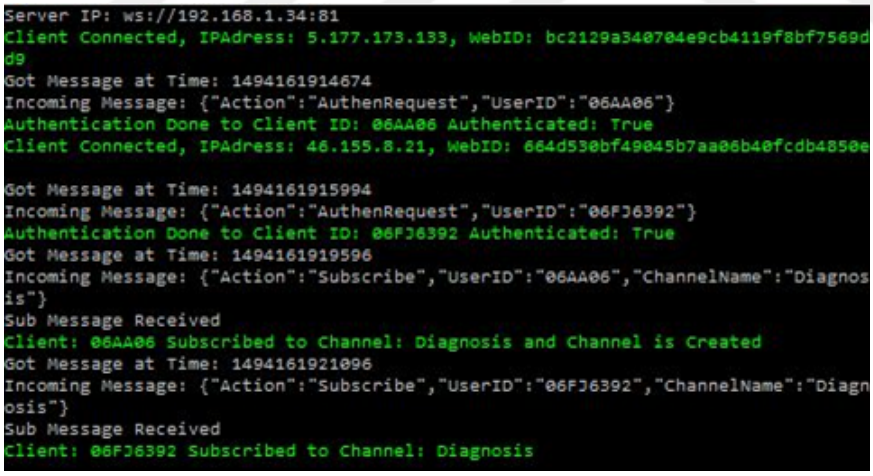

Figure 4.10: OBUs' Connection to Server

7. In Ixxat CAN Application, 't' key is pressed to send all 500 CAN frames from interface 1, so the measurement process begins here. All timing values are recorded for every component. After transmission of 500 CAN frames is finished, Ixxat CAN Application stops. Text files including records are obtained from OBUs and ITSVeCon Server machine.

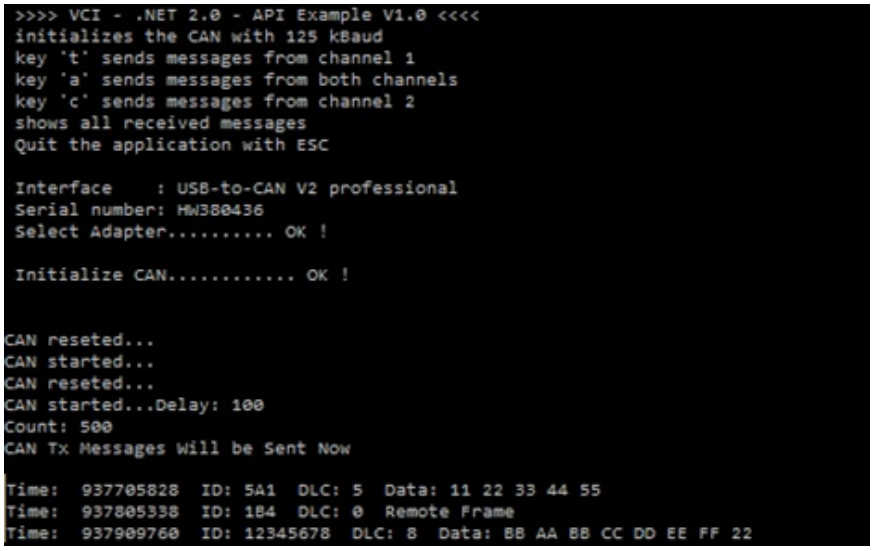

Figure 4.11: Ixxat CAN Application

8. $(T2 + Offset2)$ – $(T1 + Offset1)$ and $(T8 + Offset8)$ – $(T7 + Offset7)$ values are obtained for each CAN frame transmission. Average value of 500 CAN frames is obtained.

9. This process is repeated two times. Total average values of $T2-T1$ and $T8-T7$ are found. These values will be used for unidirectional measurement calculation when OBUs are connected to internet by 3G.

4.6.2 Bidirectional Measurement in Experimental Setup with Communication over LAN

This measurement is similar to unidirectional measurement. Only difference is CAN frames are sent from both interfaces of Ixxat. In order for OBUs to receive CAN frame and Websocket messages, both "Speed Controller" and "Tracker" applications are started in each OBUs. "Speed Controller" is capable of receiving CAN frame and converting it in JSON message, while "Tracker" is capable of receiving JSON message and extracting CAN frame. For unidirectional communication each OBU should be capable of these two conversions. This is why both applications run in each OBU. Ixxat CAN Application sends CAN frames from both interfaces simultaneously. After timings are recorded, $(T2 + Offset2)$ – $(T1 + Offset1)$ and $(T8 + Offset8)$ – $(T7 + Offset7)$ values are obtained for each CAN frame transmission for 500 CAN

frames. Process is repeated 2 times and total average values of $T2-T1$ and $T8-T7$ are obtained. These values will be used for bidirectional measurement calculation when OBUs are connected to internet by 3G.

4.6.3 Unidirectional and Bidirectional Measurement in Experimental Setup with Communication over 3G

This measurement is done in experimental setup with communication over 3G. After all timing values are obtained as explained in previous sections, $T2-T1$ and $T8-T7$ values are ignored because of high offsets. These values are obtained from previous sections. For unidirectional communication, average $T2-T1$ and $T8-T7$ values obtained from unidirectional communication measurement in "Experimental setup with communication over LAN" are used. For bidirectional communication, it is the same except the values obtained in bidirectional communication measurement in "Experimental setup with communication over LAN" are used. $T4-T3$ is found by $(T4-T1)$ – $(T2-T1)$ – $(T3-T2)$ equation. $T6-T5$ value is found by $(T8-T5)$ – $(T8-T7)$ – $(T7-T6)$ equation. All other interval values are calculated as the same with "Experimental setup with communication over LAN".

4.7 Results

In this section the results of detailed measurement are investigated. First, unidirectional and bidirectional communication end to end delay values are measured in "Experimental setup with communication over LAN". After calculating average $T2-T1$ and $T8-T7$ delay values, measurements are obtained for "Experimental setup with communication over 3G" using 3G cellular network. Finally, short range wireless interface is enabled and its' effects on measurement are investigated.

4.7.1 Unidirectional Measurement in Experimental Setup with Communication over LAN

Communication is set to unidirectional for this measurement. To achieve this, "Speed Controller" runs in OBU1 while "Tracker" runs in OBU2. "NTPD" daemon is started for OBU1, OBU2 and Server computer. Before running applications, NTP daemon synchronizes the time with NTP server for 30 minutes. Offset values can be seen in Figure 4.12, Figure 4.13 and Figure 4.14. Considering that all of offset values are under 1 ms, applications are started and Ixxat CAN application starts to send CAN frames from interface 1. Total of 2 measurements are done and 500 CAN frames are sent for each measurement.

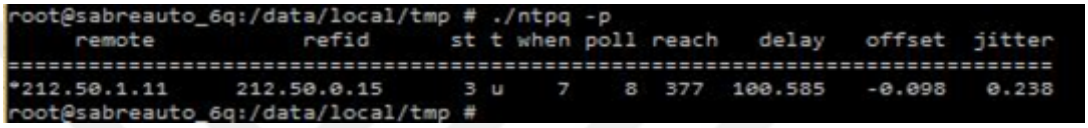

Figure 4.12: NTP and offset status of OBU1

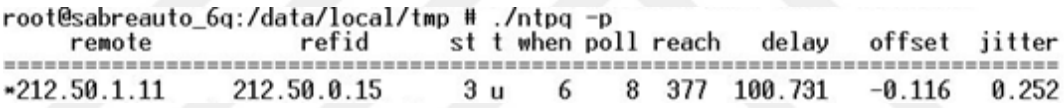

Figure 4.13: NTP and offset status of OBU2

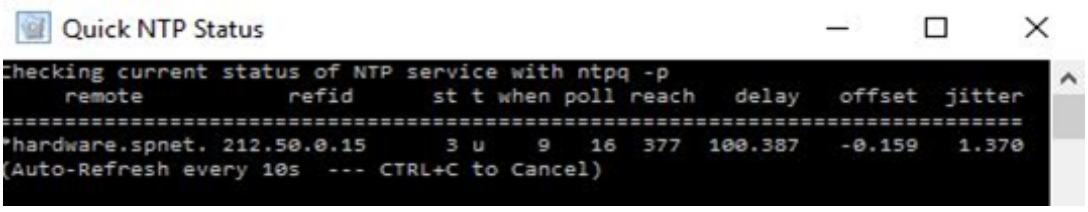

Figure 4.14: NTP and offset status of server computer

First measurement results are given in Table 4.1. According to these results, it is seen that average of total delay $T8 - T1$ is 12,248 ms which is very ideal. These results also show that if the delay between OBUs located in vehicles and server was less than 4 ms in real life, 12.248 ms end to end communication delay would be really sufficient enough to enable almost any kind of ITS applications.

Table 4.1: Unidirectional delay values for experimental setup with communication over LAN

T2-T1 (ms)	T3-T2 (ms)	T4-T3 (ms)	T5-T4 (ms)	T6-T5 (ms)	T7-T6 (ms)	T8-T7 (ms)
2.0651	1.4320	3.1149	1.074	1.2229	0.552	2.7871

Confidence interval computation is realized for above delay measurements. Since T2-T1 and T8-T7 delay values will be used when OBUs are connected to the internet by 3G, these values' confidence interval should be as small as possible. For $T2-T1$, 3.3% and 4.3% confidence intervals or 95% and 99% confidence levels show that the results are reliable. Similar confidence interval results are also obtained for $T8-T7$ delay values. Table 4.2 summarizes confidence interval computation results.

Table 4.2: Confidence interval computation for unidirectional communication over LAN

Delay	Mean (ms)	Std. Dev. (ms)	Confidence Interval($\pm\%\Delta$) Conf. Level: 95%	Confidence Interval($\pm\%\Delta$) Conf. Level: 99%
$T8-T1$	12.248	1.64879	1.18	1.553
$T2-T1$	2.0651	0.7787	3.3	4.3
$T8-T7$	2.7871	0.59635	1.8755	2.468

Second batch of 500 CAN frames are sent from Ixxat USB-CAN converter's interface 1. Delay values which is shown in Table 4.3 are calculated again. These delay values are very close to previous values. Confidence interval computation is shown in Table 4.4. The results are reliable and their confidence intervals are all below 4%.

Table 4.3: Unidirectional delay values for experimental setup with communication over LAN

T2-T1 (ms)	T3-T2 (ms)	T4-T3 (ms)	T5-T4 (ms)	T6-T5 (ms)	T7-T6 (ms)	T8-T7 (ms)
2.4623	1.468	2.7028	1.08	1.607	0.548	2.502

Average $T2-T1$ delay for these 2 measurements is 2.2637 ms. Average $T8-T7$ delay is 2.64455 ms. These values are obtained by realizing two measurements each of which includes transmission of 500 CAN frames.

Table 4.4: Confidence interval computation for unidirectional communication over LAN

Delay	Mean (ms)	Std. Dev. (ms)	Confidence Interval($\pm\%\Delta$) Conf. Level: 95%	Confidence Interval($\pm\%\Delta$) Conf. Level: 99%
$T8-T1$	$12,37$	$1,87539$	$2,65$	$3,49$
$T2-T1$	2.4623	0.84	2.99	3.93
$T8-T7$	2.502	0.5299	1.8567	2.444

4.7.2 Bidirectional Measurement in Experimental Setup with Communication over LAN

Both "Speed Controller" and "Tracker" run on each OBU to achieve bidirectional communication. Applications run and delay values are started to be recorded when offset values become less than 1 ms. Similar to unidirectional measurement, 2 measurements each transmitting 500 CAN frames are carried out. Results of first measurement can be seen in Table 4.5.

Table 4.5: Bidirectional delay values for experimental setup with communication over LAN

T2-T1 (ms)	T3-T2 (ms)	T4-T3 (ms)	T5-T4 (ms)	T6-T5 (ms)	T7-T6 (ms)	T8-T7 (ms)
2.4544	0.7480	2.6996	1.066	1.3954	0.596	2.7376

Confidence interval computation is shown in Table 4.6. All results are below 5% confidence interval.

Table 4.6: Confidence interval computation for bidirectional communication over LAN

Delay	Mean (ms)	Std. Dev. (ms)	Confidence Interval($\pm\%\Delta$) Conf. Level: 95%	Confidence Interval($\pm\%\Delta$) Conf. Level: 99%
$T8-T1$	11.696	2.249	3.371	4.43
$T2-T1$	2.4544	0.632	2.26	2.975
$T8-T7$	2.7376	0.5552	1.778	2.341

Second measurement results and confidence interval computations are given in Table 4.7 and Table 4.8.

Table 4.7: Bidirectional delay values for experimental setup with communication over LAN

T2-T1 (ms)	T3-T2 (ms)	T4-T3 (ms)	T5-T4 (ms)	T6-T5 (ms)	T7-T6 (ms)	T8-T7 (ms)
2.258	0.804	3.3035	1.072	1.5093	0.698	2.5952

Table 4.8: Confidence interval computation for bidirectional communication over LAN

Delay	Mean (ms)	Std. Dev. (ms)	Confidence Interval($\pm\%\Delta$) Conf. Level: 95%	Confidence Interval($\pm\%\Delta$) Conf. Level: 99%
$T8-T1$	12.24	3.0689	4.395	5.78
$T2-T1$	2.258	0.7489	2.907	3.826
$T8-T7$	2.5952	1.1196	3.7815	4.9778

Average $T2-T1$ delay for these 2 measurements is 2.3562 ms while average $T8-T7$ delay is 2.666 ms. These values will be used for delay measurements in "Experimental setup with communication over 3G" with bidirectional communication.

4.7.3 Unidirectional Measurement in Experimental Setup with Communication over 3G

For this experimental setup, both OBUs are connected to internet by 3G with the help of Turkcell 3G USB modem and TP-Link TL-MR3020. $T2-T1$ and $T8-T7$ values are obtained from "Experimental setup with communication over LAN" results with unidirectional communication. $T4-T3$ is found by $(T4-T1)$ – $(T2-T1)$ – $(T3-T2)$ equation. $T6-T5$ value is found by $(T8-T5)$ – $(T8-T7)$ – $(T7-T6)$ equation. Delay values and confidence interval computations are given in Table 4.9 and Table 4.10 respectively. Average end to end delay is 110.44 ms which is sufficient for many ITS applications. Confidence intervals are 2.732% and 3.596% for 95% and 99% confidence levels.

$T8-T1$ delay values are given in Figure 4.15 for all CAN frames transmitted. If the graph is examined, some spikes resulting from excessive 3G cellular access network delay can be seen. This is due to the 3G connection quality of selected carrier in

Table 4.9: Unidirectional delay values for experimental setup with communication over 3G

T2-T1 (ms)	T3-T2 (ms)	T4-T3 (ms)	T5-T4 (ms)	T6-T5 (ms)	T7-T6 (ms)	T8-T7 (ms)
2.2637	1.636	55.5163	1.082	46.70345	0.594	2.64455

Table 4.10: Confidence interval computation for unidirectional communication over 3G

Delay	Mean (ms)	Std. Dev. (ms)	Confidence Interval($\pm\%\Delta$) Conf. Level: 95%	Confidence Interval($\pm\%\Delta$) Conf. Level: 99%
$T8-T1$	110.44	34.417	2.732	3.596

Turkey. Better delay values can be obtained with 4G or realising resource allocation to ITS applications by GSM operators.

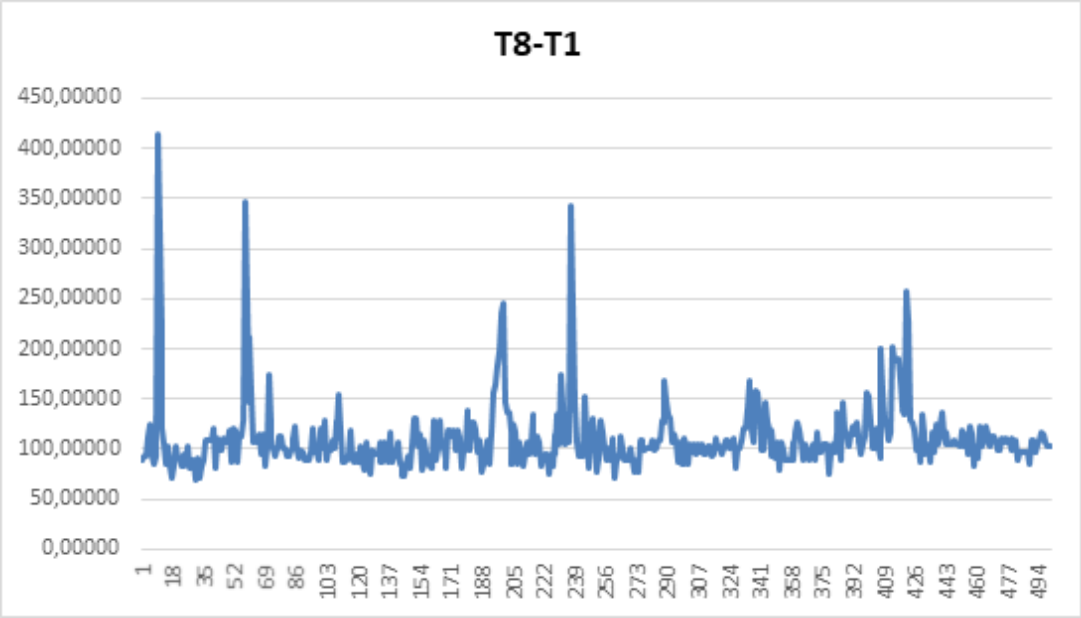

Figure 4.15: Unidirectional communication over 3G end to end delay values

Second measurement results and confidence interval computation are given in Table 4.11 and Table 4.12 respectively. Delay values and confidence intervals are close to the previous results.

$T8-T1$ values are given for all CAN frames transmitted in Figure 4.16. Again, some spikes resulted from 3G cellular access network. This shows that ITS applications

Table 4.11: Unidirectional delay values for experimental setup with communication over 3G

T2-T1 (ms)	T3-T2 (ms)	T4-T3 (ms)	T5-T4 (ms)	T6-T5 (ms)	T7-T6 (ms)	T8-T7 (ms)
2.2637	1.404	65.95485	1.086	47.0834	0.588	2.64455

Table 4.12: Confidence interval computation for unidirectional communication over 3G

Delay	Mean (ms)	Std. Dev. (ms)	Confidence Interval($\pm\%\Delta$) Conf. Level: 95%	Confidence Interval($\pm\%\Delta$) Conf. Level: 99%
$T8 - T1$	121.02	42.222	3.058	4.025

related to security cannot be realized with this connection speed.

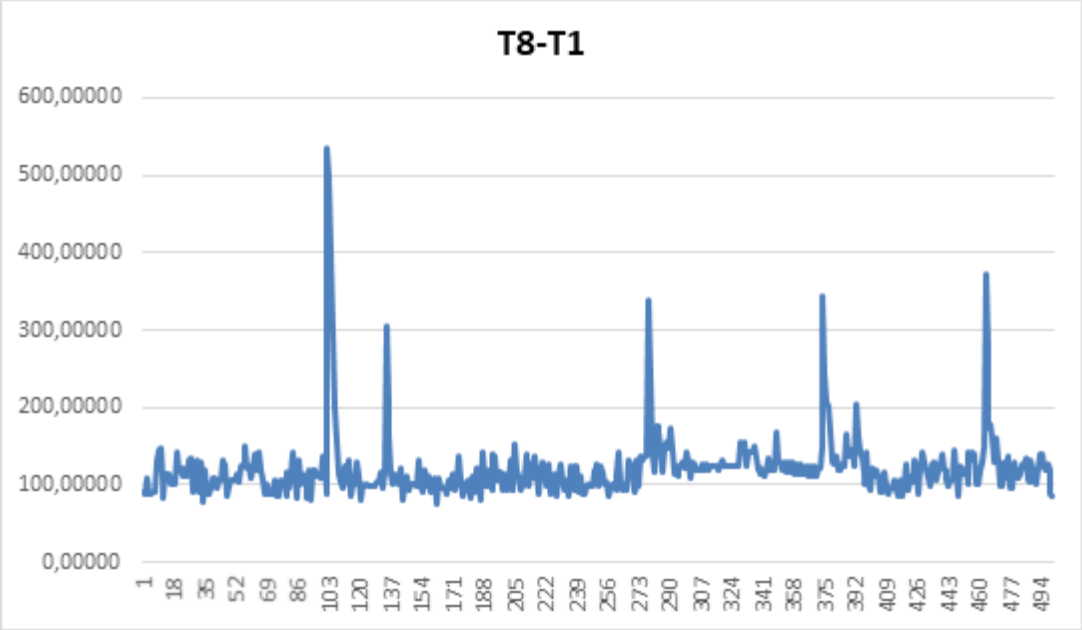

Figure 4.16: Unidirectional communication over 3G end to end delay values

4.7.4 Bidirectional Measurement in Experimental Setup with Communication over 3G

Bidirectional communication delay values are obtained in this experimental setup. Similar to unidirectional communication, $T2 - T1$ and $T8 - T7$ values are obtained

from "Experimental setup with communication over LAN" results with bidirectional communication. $T4-T3$ is found by $(T4-T1)$ – $(T2-T1)$ – $(T3-T2)$ equation. $T6-T5$ value is found by $(T8-T5)$ – $(T8-T7)$ – $(T7-T6)$ equation. Delay values and confidence interval computations are given in Table 4.13 and Table 4.14 respectively. Average end to end delay is 118.82 ms which is sufficient for many ITS applications. Confidence intervals are 4.49% and 5.916% for 95% and 99% confidence levels.

Table 4.13: Bidirectional delay values for experimental setup with communication over 3G

T2-T1 (ms)	T3-T2 (ms)	T4-T3 (ms)	T5-T4 (ms)	T6-T5 (ms)	T7-T6 (ms)	T8-T7 (ms)
2.3562	0.802	54.064	1.058	57.4834	0.508	2.666

Table 4.14: Confidence interval computation for bidirectional communication over 3G

Delay	Mean (ms)	Std. Dev. (ms)	Confidence Interval($\pm\%\Delta$) Conf. Level: 95%	Confidence Interval($\pm\%\Delta$) Conf. Level: 99%
$T8-T1$	118.82	60.92	4.49	5.916

$T8-T1$ delay values are given in Figure 4.17 for each of CAN frames transmitted. Spikes are observed because of 3G connection latency issues of the carrier.

Second measurement results and confidence interval computation are given in Table 4.15 and Table 4.16 respectively. Delay values and confidence intervals are close to the previous results.

Table 4.15: Bidirectional delay values for experimental setup with communication over 3G

T2-T1 (ms)	T3-T2 (ms)	T4-T3 (ms)	T5-T4 (ms)	T6-T5 (ms)	T7-T6 (ms)	T8-T7 (ms)
2.3562	0.696	41.076	1.05	61.721	0.63	2.666

T8-T1 delay values are shown in Figure 4.18.

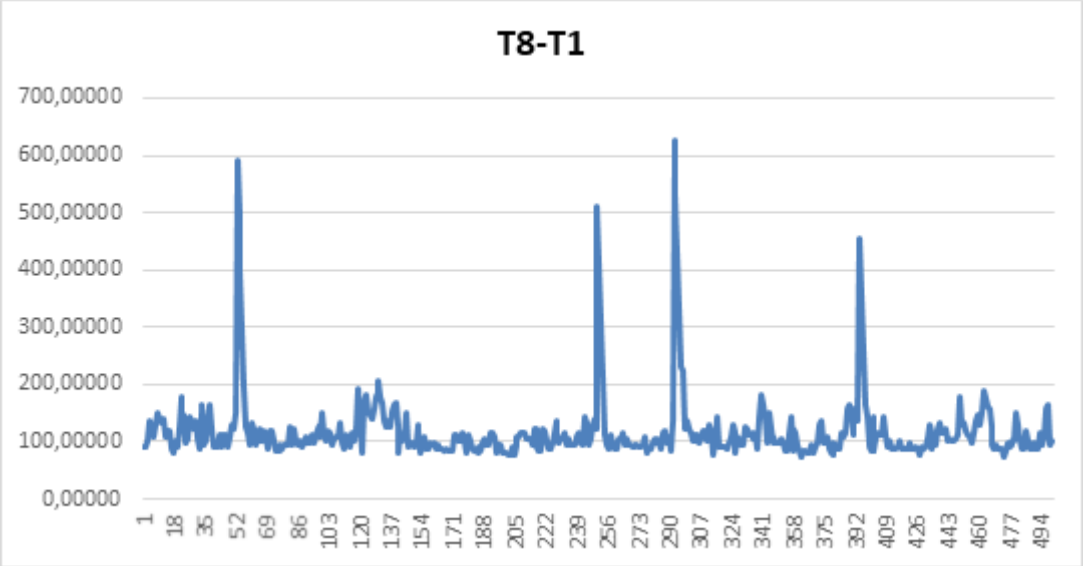

Figure 4.17: Bidirectional communication over 3G end to end delay values

Table 4.16: Confidence interval computation for bidirectional communication over 3G

Delay	Mean (ms)	Std. Dev. (ms)	Confidence Interval($\pm\%\Delta$) Conf. Level: 95%	Confidence Interval($\pm\%\Delta$) Conf. Level: 99%
$T8-T1$	110.08	57.455	4.575	6.022

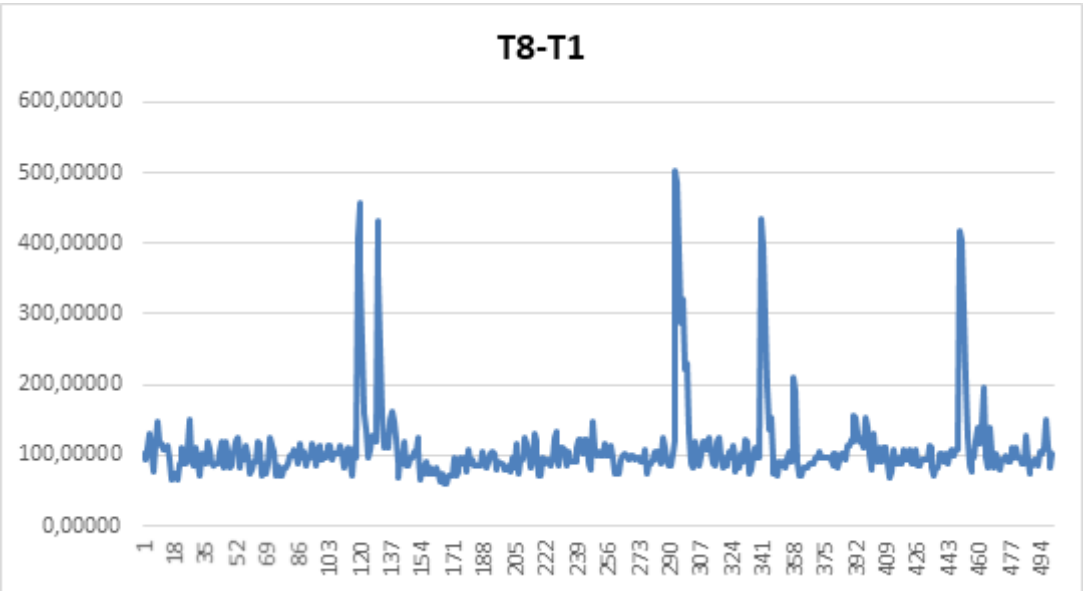

Figure 4.18: Bidirectional communication over 3G end to end delay values

4.7.5 Comparison of 3G and ADSL Ping Values

To see the effects of 3G access delay, ICMP packets are sent to server for two cases. In the first case, computer is connected to internet via ADSL with 16 Mbps connection speed. For the second case, it is connected to internet with the help of TP-Link TL-MR3020 and 3G USB modem. 40 ping packets are sent to server and RTT time is recorded. Figure 4.19 shows measured latency values for each packet. The time is divided into two to find one way latency. While average latency is measured as 4.775 ms with ADSL usage, it is measured as 39.325 ms with 3G connection. Latency from computer to ADSL modem is neglected because it is lower than 1 ms. Similarly, for 3G connection, latency from computer to TP-Link TL-MR3020 is neglected. Results show that, if stable connection could be achieved in mobile vehicles like ADSL, one way latency would reduce drastically enabling the use of safety related ITS applications.

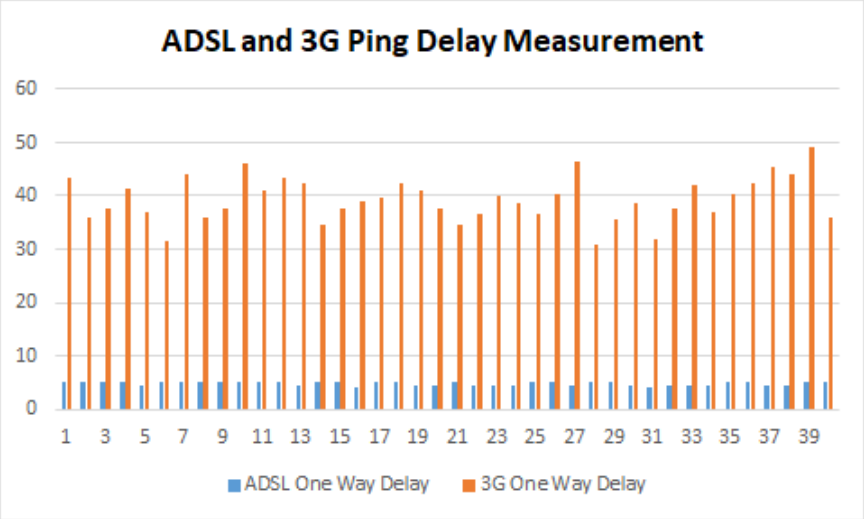

Figure 4.19: ADSL and 3G Ping delay measurement

4.7.6 Overall Assessment of Results

In "Unidirectional Measurement in Experimental Setup with Communication over 3G" experiment, total percentage of delays on embedded hosts are %2.93 and %2.48 respectively. In "Bidirectional Measurement in Experimental Setup with Communication over 3G" experiment, total percentage of delays on embedded hosts are %1.93

and %2 respectively. What we can get from these percentages is that internal embedded delays are negligible. Network delay is the main factor that is contributing to the total end-to-end delay. It can also be seen from Figure 4.19 that 3G network delay is much more than ADSL network delay. As a result of it, since embedded internal delays are very low, end-to-end delay values can reduce very much with the advances in technology.

4.8 Functional Test With Enabling SRWI

This experiment is carried out to show that OBUs are capable of handling messages coming from two different resources. SRWI messages are generated virtually in SRWI Layer creating an second interface in addition to 3G cellular network.

After enabling SRWI, same JSON messages with “Speed Controller” are generated by “Short Range Wireless Interface” layer in “Tracker” application except “Info” field. “Tracker” decides which JSON message to obtain by looking to counter value. In this measurement, overall delay $T8 - T1$ and the effects of enabling short range wireless interface are explained. This measurement is done with 3 different packet delivery ratios of this interface which are %30, %60 and %100. In this type of experiment, "Experimental setup with communication over 3G" with unidirectional communication configuration is selected together with enabling short range wireless interface. Delay of the short range wireless interface is selected as 115 ms which is the average of two end to end delay values obtained in "Experimental setup with communication over 3G" with unidirectional communication. Thus, delay of short range wireless interface becomes comparable to 3G interface. Messages are selected by looking at the counter values of Websocket messages coming from two interfaces which was explained in Figure 3.8. If 5G connection was used for communicating with server, real short range wireless interface delay values like 4 ms could be used for analysis.

Using SRWI together with 3G connection is examined in 3 different scenarios. At the first scenario, packet delivery ratio of SRWI interface is %100 which means that all of the Websocket messages generated by SRWI class in OBU application are conveyed to Websocket Messages class. Total of 1000 messages are delivered to "Tracker"

application in the form of 500 3G messages and 500 SRWI messages. Message reception is done by looking the counter values. Figure 4.20 compares this condition with using only 3G connection. It can easily be seen that using SRWI interface together with 3G inteface reduces end to end delay values significantly. It normalizes delay values between 100 ms and 120 ms. Whenever the connection of 3G interface is not stable, Websocket messages are received from SRWI which eliminates spikes. It is crucial for safety related applications. Usage of SRWI messages is shown in Figure 4.21. In this graph, the value of "1" represents the usage of SRWI messages. If two figures are compared, it is seen that when 3G connection makes peaks and has long delay values, SRWI compansates for these issues.

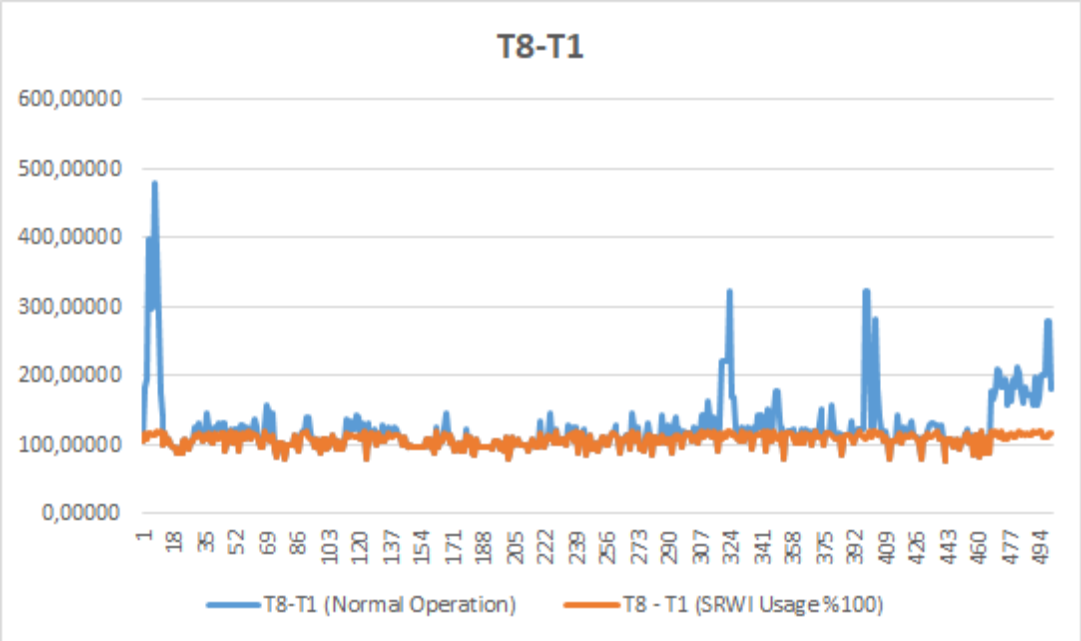

Figure 4.20: End to end delay after enabling SRWI with Probability %100

Figure 4.22 shows the same case when SRWI has %60 delivery ratio. In this case some of the peaks cannot be compansated by SRWI because of delivery ratio. The last scenario is shown by Figure 4.23 when SRWI delivery ratio is %30. The number of peaks and delay values are increased compared to other scenarios.

Confidence interval computation is done for 4 cases shown in Table 4.17. In the first case, SRWI is not used so only messages coming from 3G interface are used. Average of total end to end delay is computed as 123.83 ms. SRWI is activated for other cases

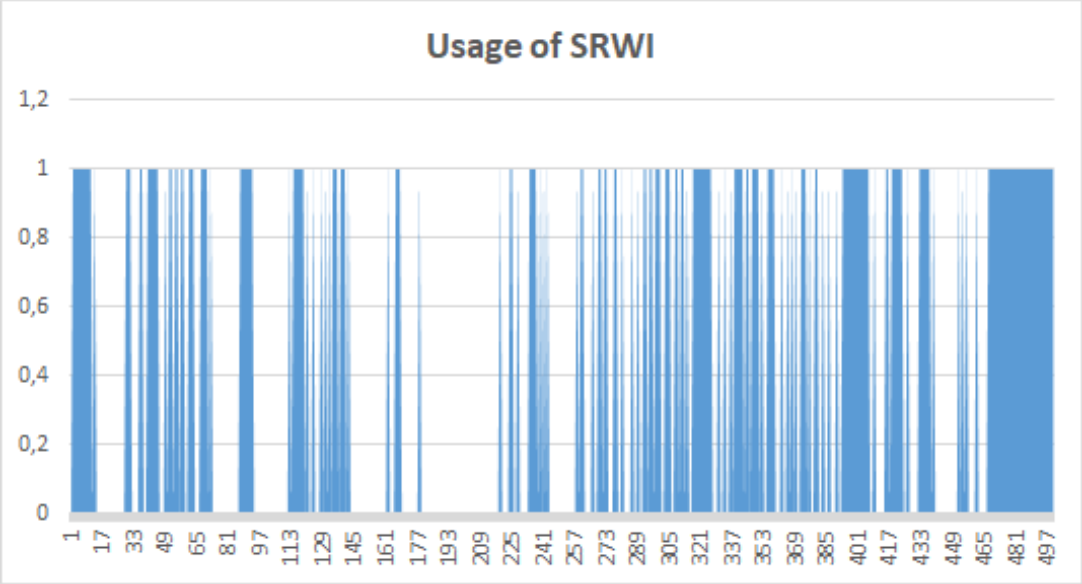

Figure 4.21: Usage of SRWI Messages %100

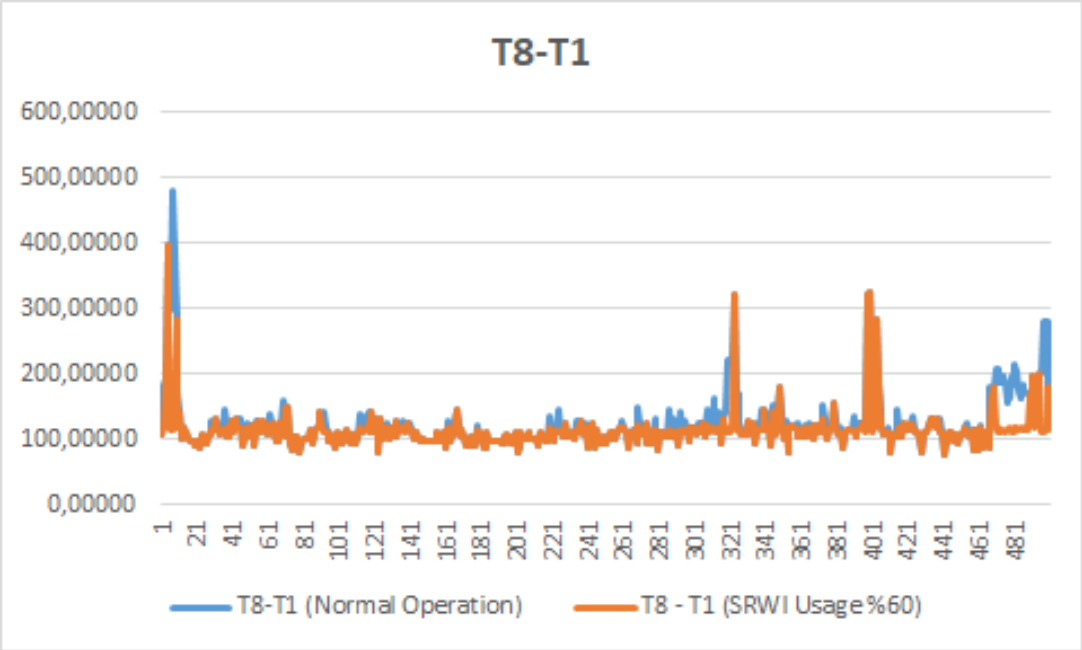

Figure 4.22: End to end delay after enabling SRWI with Probability %60

and it is seen that there is an improvement for end to end delay values. SRWI with %100 packet delivery ratio has the lowest end to end delay values, standard deviation and confidence interval.

These functional experiments are done to indicate the benefits of using SRWI together

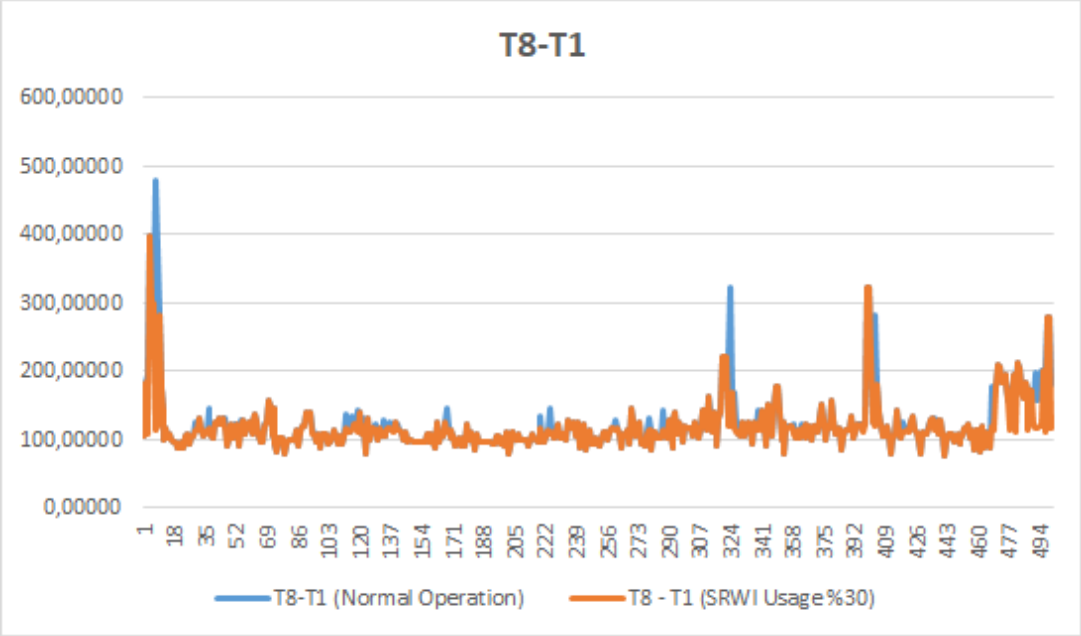

Figure 4.23: End to end delay after enabling SRWI with Probability %30

Table 4.17: End to end delay values for different SRWI delivery ratios

Delay DSRC Enabled	Mean (ms)	Std. Dev. (ms)	Confidence Interval($\pm\%\Delta$) Conf. Level: 95%	Confidence Interval($\pm\%\Delta$) Conf. Level: 99%
$T8-T1$ %100 Prob	107.758	9.2	0.74	0.98
$T8-T1$ %60 Prob	113.728	29.74	2.29	3.0
$T8-T1$ %30 Prob	118.314	34.03	2.5	3.3
$T8-T1$ Nor. Op.	123.83	43.16	3.0	4.0

with 3G interface. SRWI like DSRC has much lower end to end delay values in real life but the delivery ratio is the main problem. In order to solve this, a solution with using two interfaces is examined in this section. This experiment shows that if the problems of short range wireless interfaces regarding delivery ratio cannot be solved in the future, it can be used together with 4G or 5G standard. The advantages of this solution are low end to end delay values, standard deviation and confidence interval. Therefore it is very applicable for safety related ITS applications. Main disadvantage is that messages are sent from both interfaces meaning that usage of wireless band is

ineffective and it consumes more power.

CHAPTER 5

CONCLUSION

This thesis proposes an application layer communication architecture, ITSVeCon for V2X communications to facilitate Intelligent Transport Systems (ITS) Applications. ITSVeCon is an all IP application layer architecture enabling communication among the end-hosts which can be vehicle Electronic Control Units (ECU)'s, Road Side Units (RSU)s, computers, smart phones or third party service providers. All these end-hosts are bi-directionally connected to the ITSVeCon Server where this server carries out application layer switching realizing unicast or multicast communication among the end-hosts. The architecture consists of a layered software and network protocol stack with message formats and rules, which are implemented in the end-hosts and the ITSVeCon server.

To this end, this thesis presents the ITSVeCon realization on the vehicle On Board Unit (OBU) and the ITSVeCon server. The OBU realization further fulfills the gateway functionality between the in-vehicle CAN network and the Internet. This gateway enables end-to-end transmission of CAN messages between the ECU's of two distinct vehicles. The ITSVeCon implementation features WebSockets carrying messages in JSON format, Publish and Subscribe pattern and NTP synchronization to enable V2X communications for real-time ITS applications. OBU ITSVeCon application provides simple and flexible infrastructure. While layers at the bottom controls hardware components, layers in the middle maintain communication between lower and upper layers. Many ITS applications can be developed to run at top layer using features of lower layers. Architecture allows these ITS applications to run together or independently without any restrictions. Controlling real CAN and Ethernet interfaces, communication with short range wireless interface, location services, con-

trollable settings and editable user interface are most important features regarding ITSVeCon Application.

This thesis proposes the cellular communications as the wireless communication technology for the vehicle. To this end, the end-to-end communication path in ITSVeCon consists of cellular access and IP core network over multiple nodes and network segments. A very important contribution of the thesis is the measurement set-up, detailed experiment scenarios and measurement results of the end-to-end delay components. These results are collected for the communication of CAN data of two ECUs in two different vehicles which represent the most general scenario. By recording timestamp values at certain message transmission points with the help of NTP server and V2X applications, communication delay is measured along transmission path. Delay values inside and between the components are investigated. If results are examined, it is seen that for critical safety ITS applications, 3G usage is not enough because of low connection speed and unexpected connection issues. 3G connection speed in Turkey is less than average 3G speed in the world which causes higher delay values. Nevertheless measured end to end delay values are close to 100 ms showing that the proximity between server and clients is important. ITS applications related to infotainment can be realized with 3G connection if server is located in the same city with other components in ITS architecture. Using 3G interface together with short range wireless interface is proposed functionally in this thesis as a second option to improve communication. SRWI is seen as a second arbitrary interface and functional tests are carried out. Lower delay values can be achieved with this option because short range wireless interface compensates for the high latencies resulted from 3G cellular network. With the improvements in technology, around 10 ms end to end delay value can be achieved with 4G and 1 ms end to end delay value is expected to be accomplished with 5G.

An important contribution of this thesis is to show that embedded implementation overheads are negligible. The main factor determining end-to-end latency is the network latency. So for real time applications, reducing network delay should be the main focus.

As future work, it can be achieved that ITSVeCon server sends Websocket messages

simultaneously to the subscribers. This can improve end to end delay values for broadcast messages. Messages can be prioritized to make sure that safety related messages arrives to destination point at the right time. Geolocation based communication can be supported to relieve server load.

REFERENCES

[1] European Telecommunications Standards Institute. *Intelligent Transport Systems (ITS); Communications Architecture*. Standard ETSI EN 302 665 V1.1.1 edition, 2010.

[2] F.Qu, F.Wang, and L.Yang. Intelligent transportation spaces: vehicles, traffic, communications, and beyond. *IEEE Communications Magazine*, 48:136–142, November 2010.

[3] K. Zheng, Q. Zheng, P. Chatzimisios, W. Xiang, and Y. Zhou. Heterogeneous vehicular networking: A survey on architecture, challenges, and solutions. *IEEE Communications Surveys Tutorials*, 17:2377–2396, June 2015.

[4] P. Papadimitratos, A. L. Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza. Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation. *IEEE Communications Magazine*, 47:84–95, November 2009.

[5] Kaan Cetinkaya. A generic and extendable system architecture for intelligent transportation systems. Master's thesis, Middle East Technical University, 2015. Supervisor: Dr. Ece Guran Schmidt.

[6] 5GPP. 5G Automotive Vision. `https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-White-Paper-on-Automotive-Vertical-Sectors.pdf`. Accessed on May 2017.

[7] Society of Automative Engineers. J2056/1 class C application requirements classfications, SAE Handbook, 1994.

[8] Society of Automative Engineers. J2056/2 survey of known protocols, SAE Handbook, 1994.

[9] NXP. In-Vehicle Networking LIN, CAN, RF and Flexray Technology. `http://www.nxp.com/assets/documents/data/en/brochures/BRINVEHICLENET.pdf`. Accessed on May 2017.

[10] R. Bosch. CAN specification version 2.0, Stutgart, 1991.

[11] R. Davis, A. Burns, R. Bril, and J. Lukkien. Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised. *Real-Time Syst.*, 35:239–272, April 2007.

[12] K. Tindell and A. Burns. Guaranteering Message Latencies On Controller Area Network (CAN). Technical report, University of York, YO1 5DD, England, 1994.

[13] U. Klehmet, T. Herpel, K. Hielscher, and R. German. Delay Bounds for CAN Communication in Automotive Applications. In *Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2008 14th GI/ITG Conference*, 2008.

[14] IEEE. IEEE Standard 1609.1, IEEE trial-use standard for wireless access in vehicular environments (WAVE), 2006.

[15] Y. Li. An Overview of the DSRC/WAVE Technology. In *Quality, Reliability, Security and Robustness in Heterogeneous Networks*, volume 74, 2012.

[16] W. Lin, M. Li, K. Lan, and C. Hsu. A comparison of 802.11a and 802.11p for V-to-I communication: a measurement study. In *Quality, Reliability, Security and Robustness in Heterogeneous Networks*, volume 74, 2012.

[17] X. Wu, S. Subramanian, R. Guha, R. G. White, J. Li, K. W. Lu, A. Bucceri, and T. Zhang. Vehicular communications using dsrc: Challenges, enhancements, and evolution. *IEEE Journal on Selected Areas in Communications*, 31:399–408, September 2013.

[18] X. Cheng, L. Yang, and X. Shen. D2d for intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 16:1784–1793, August 2015.

[19] S. Mumtaz, K. Huq, M. Ashraf, J. Rodriguez, V. Monteiro, and C. Politis. Cognitive vehicular communication for 5g. *IEEE Communications Magazine*, 53:109–117, July 2015.

[20] A. Asadi, Q. Wang, and V. Mancuso. A survey on device-to-device communication in cellular networks. *IEEE Communications Surveys and Tutorials*, 16:1801–1819, April 2014.

[21] Ilya Grigorik. *High Performance Browser Networking*. O'REILLY, first edition, September 2013.

[22] 3GPP. LTE Overview. `http://www.3gpp.org/technologies/keywords-acronyms/98-lte`. Accessed on May 2017.

[23] 3GPP. LTE Advanced Overview. `http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced`. Accessed on May 2017.

[24] A. Vinel. 3gpp lte versus ieee 802.11p/wave: Which technology is able to support cooperative vehicular safety applications. *IEEE Wireless Communications Letters*, 1:125–128, April 2012.

[25] W. Jiang and V. K. Prasanna. The internet of vehicles based on 5g communications. In *Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2016 IEEE International Conference*, December 2016.

[26] I. Fette and A. Melnikov. The websocket protocol. RFC 6455, RFC Editor, December 2011. `http://www.rfc-editor.org/rfc/rfc6455.txt`.

[27] V. Pimentel and B. Nickerson. Communicating and displaying real-time data with websocket. *IEEE Internet Computing*, 16:45–53, July-Aug. 2012.

[28] Pieter Hintjens. ZeroMQ Protocol. `https://zeromq.org`. Accessed on December 2017.

[29] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35:114–131, June 2003.

[30] D. Mills, J. Martin, J. Burbank, and W. Kasch. Network time protocol version 4: Protocol and algorithms specification. RFC 5905, RFC Editor, June 2010. `http://www.rfc-editor.org/rfc/rfc5905.txt`.

[31] NTPD. Network Time Protocol (NTP) daemon. `http://doc.ntp.org/4.1.0/ntpd.htm`. Accessed on May 2017.

[32] About Meinberg NTP Program. `https://www.meinbergglobal.com/english/sw/ntp.htm`. Accessed on May 2017.

[33] M. Ferreira and P. d'Orey. On the impact of virtual traffic lights on carbon emissions mitigation. *IEEE Transactions on Intelligent Transportation Systems*, 13:284–295, March 2012.

[34] M. Ardita, Suwadi, A. Affandi, and Endroyono. Http communication latency via cellular network for intelligent transportation system applications. In *Information Communication Technology and Systems (ICTS), 2016 International Conference*, October 2016.

[35] H. Andrade, C. Ferreira, and A. Filho. Latency analysis in real lte networks for vehicular applications. In *Computing Systems Engineering (SBESC), 2016 VI Brazilian Symposium*, November 2016.

[36] S. Kato, M. Hiltunen, K. Joshi, and R. Schlichting. Enabling vehicular safety applications over lte networks. In *Connected Vehicles and Expo (ICCVE), 2013 International Conference*, December 2013.

[37] R. Nasim, A. J. Kassler, I. Podnar, and A. Antonic. Enabling vehicular safety applications over lte networks. In *Cloud and Autonomic Computing (ICCAC), 2014 International Conference*, September 2014.

[38] I. Lequerica, P. M. Ruiz, and V. Cabrera. Improvement of vehicular communications by using 3g capabilities to disseminate control information. *IEEE Network*, 24:32–38, Jan-Feb 2010.

[39] S. Zeadally, R. Hunt, Y. Chen, A. Irwin, and Aamir Hassan. Vehicular ad hoc networks (vanets): status, results,and challenges. In *Telecommun Syst*, December 2010.

[40] Introducing JSON. `http://www.json.org`. Accessed on May 2017.

[41] Newtonsoft JSON library. `http://www.newtonsoft.com/json`. Accessed on May 2017.

[42] Websocket sharp read-me file. `https://github.com/sta/websocket-sharp`. Accessed on May 2017.

[43] NXP. Sabre for automotive infotainment based on the i.mx 6. `http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/i.mx-applications-processors/i.mx-6-processors/sabre-for-automotive-infotainment-based-on-the-i.mx-6-series:RDIMX6SABREAUTO`. Accessed on May 2017.

[44] Yocto Project. `https://www.yoctoproject.org`. Accessed on May 2017.

[45] Android Official Site for Developers. `https://developer.android.com`. Accessed on May 2017.

[46] Autobahn Project. Open-source real-time framework for web, mobile and internet of things. `http://autobahn.ws/android`. Accessed on May 2017.

[47] Readme file for the Controller Area Network Protocol Family. `https://www.kernel.org/doc/Documentation/networking/can.txt`. Accessed on May 2017.

[48] JNI functions. `http://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/functions.html`. Accessed on May 2017.

[49] Linux can utils. CAN userspace utilities and tools. `https://gitorious.org/linux-can/can-utils`. Accessed on May 2017.

[50] Open source Web Application Messaging Platform. `http://crossbar.io`. Accessed on May 2017.

[51] Paolo Pagano. *Intelligent Transportation Systems: From Good Practices to Standards*. CRC Press, April 2016.

[52] Ixxat USB to CAN Converter V2. https://www.ixxat.com/products/productsindustrial/pc-interfaces/pc-can-interfaces/details-pc-can-interfaces/usbto-can-v2. Accessed on May 2017.

[53] TP-Link 3G USB Modem to Ethernet. http://www.tp-link.com/il/products/details/cat-14_TL-MR3020.html. Accessed on May 2017.