

Certificate Revocation List Distribution in Vehicular Communication Systems

MANI AMOOZADEH



KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden November 2012

XR-EE-LCN 2012:014



KTH Electrical Engineering

Master of Science Thesis

Certificate Revocation List Distribution in Vehicular Communication Systems

Mani Amoozadeh

Supervisor and Examiner

Professor Panos Papadimitratos

Lab of Communication Networks (LCN)
School of Electrical Engineering
Kungliga Tekniska Högskolan (KTH)
Stockholm, Sweden

Abstract

Message exchange in VANETs should be secured. Researchers have designed many methods to meet this goal. One of the ways agreed upon by most researchers, is through the use of a public-key infrastructure (PKI). An important part of any PKI system is certificate revocation. The revocation is usually done by periodically issuing a Certificate Revocation List (CRL) by the Certification Authority (CA).

After the creation of a CRL by CA, the CRL should be distributed in the VC system. The important question is how we can distribute the CRL efficiently and in a timely manner throughout the system in a way that all vehicles receive a genuine copy of it. A couple of researches considered CRL distribution in the past and proposed different methods like RSU-only [1], C2C Epidemic [2], and Most Pieces Broadcast (MPB) [3].

We implement the aforementioned CRL distribution methods and evaluate them using a common framework. With this approach, we can compare these methods accurately and point out the limitations of each. Due to the fact that C2C Epidemic did not provide any packet-level implementation, we propose an implementation for it.

We also propose a new method for CRL distribution called ICE (Intelligent CRL Exchange). This method uses V2V and I2V communication to distribute the CRL pieces to vehicles. ICE is an enhanced version of the MPB method and it uses semi-incremental CRL exchange. With this approach, the number of duplicate received pieces decreases in comparison to the MPB method. Moreover, ICE uses a simple approach to decrease the number of unnecessary broadcasts by RSUs.

The evaluation is done through simulations. OMNET++ [4] and the MiXiM framework are used for detailed packet-level simulation. The simulation is done for both small and large scale scenarios. For the large scale simulation, we use SUMO [5] to generate mobility traces of vehicle nodes. Different criteria are defined so that we can compare CRL distribution methods.

According to the simulation results, vehicles in C2C Epidemic, MPB and ICE receive all the required CRL pieces in less time in comparison to RSU-only, because vehicles use both I2V and V2V communications. MPB shows a better performance than C2C Epidemic, but the number of duplicate received pieces increases substantially. ICE tries to alleviate this by incorporating semi-incremental CRL exchange. Furthermore, the number of broadcasts by RSUs in the ICE method shows reduction.

Acknowledgement

This thesis has been done in LCN (Lab of Communication Networks) lab in the school of electrical engineering at KTH University, Stockholm, Sweden.

Special gratitude goes to my supervisor, Professor *Panos Papadimitratos* for providing me the opportunity to work with him. He patiently guided me during this thesis and provided valuable input and knowledge.

Moreover, I would like to thank *Christoph Sommer* for helping me with problems I encountered using the Veins framework. I am also grateful to *Stylianos Gisdakis* for his useful hints, and *Marcello Laganà* for giving me access to the server for running large scale simulation. Furthermore, I would like to thank all my Professors at Haninge Campus and also LCN lab who taught me during my master's program at KTH.

Finally, I appreciate my parents and my brother for all their supports and encouragements. I would not be able to complete my master's degree without their true support. I dedicate this thesis to them.

Mani Amoozadeh
Stockholm November 2012

To my lovely parents

Contents

1	Introduction	1
1.1	Problem Definition	2
1.2	Method	2
1.3	Contribution	3
1.4	Thesis Outline	3
2	Background	5
2.1	Vehicular Ad-hoc Networks (VANETs)	5
2.2	Secure Vehicular Communication (VC) Systems	6
2.2.1	Simplified Operation of a Secure VC System	7
2.2.2	Using Short-term Identification	8
2.2.3	Obtaining Pseudonyms	8
2.2.4	Certificate Renewal and Certificate Revocation	9
2.2.5	Certificate Revocation Schemes	9
2.2.6	Revocation of the Trusted Component	11
2.3	Beaconing	11
3	Current CRL Distribution Methods	13
3.1	RSU-only CRL Distribution	13
3.2	C2C Epidemic CRL Distribution	15
3.3	Most Pieces Broadcast (MPB) CRL Distribution	16
4	Implementation of Current CRL Distribution Methods	19
4.1	Group 1: Scenarios without Erasure Code	19
4.2	Group 2: Scenarios with Erasure Code	21
4.2.1	Rabin's Algorithm	21
4.3	Using Special Vehicles to receive CRL Magically	24
4.4	Implementing C2C Epidemic CRL Distribution	25
4.5	Implementing MPB CRL Distribution	27
4.6	Hidden Station Problem	28
5	Intelligent CRL Exchange (ICE)	31
5.1	Beacon Message Format in ICE	31
5.2	The Need for Incremental CRL Exchange	31
5.3	Start, End Index Fields: Semi-incremental CRL Exchange	32

5.4	Prioritizing the CRL Pieces	33
5.5	Efficiency of ICE	33
5.6	Reducing the Number of Broadcasts	34
6	Simulation Setup	37
6.1	OMNET++ Simulator and MiXiM Framework	37
6.2	Topology of VC Systems	38
6.3	CRL Distribution (In nutshell)	39
6.4	Protocol Stack of Each Element	39
6.5	Cross-layer Communication	41
6.6	Application Layer Parameters	42
6.7	Mobility of Vehicles	44
6.8	SUMO Simulator	46
6.8.1	TraCI: Online Interaction with SUMO	46
6.8.2	Generating the Network File from OpenStreetMap	47
6.8.3	Generating the Traffic-demand File	48
7	Evaluation Results	51
7.1	Evaluation Criteria	51
7.1.1	Convergence time	51
7.1.2	Number of Vehicles that Received the Full set of CRL Pieces	52
7.1.3	Number of Received Pieces	52
7.1.4	Number of Frames Received with Error	52
7.1.5	Number of Broadcasts by RSUs	52
7.2	RandomDirection Mobility	53
7.2.1	Comparing CRL Distribution Methods	53
7.2.2	Using Shuffling	56
7.2.3	Using Erasure Code	58
7.2.4	Using "Magic Cars"	59
7.2.5	Number of Broadcasts by RSUs	60
7.3	SUMO Mobility	61
7.3.1	Comparing CRL Distribution Methods	62
8	Conclusion and Future Works	65
A	Calculation of Interference Distance	67
B	Simulation Platform	69
	Bibliography	70

List of Figures

2.1	Simple Illustration of VANETs	6
2.2	Long-term Identification of Vehicle X	7
2.3	Message Format in Secure VC System	7
2.4	Certificate and CRL Format in X.509 Standard	10
3.1	V2I Communication in RSU-Only Approach	14
3.2	V2I and V2V Communications in C2C Epidemic	15
3.3	Example of Most Pieces Broadcast	17
4.1	Message Format Sent from CA to RSUs	20
4.2	Overview of Rabin’s Algorithm	22
4.3	Applying Rabin’s Algorithm on CRL Message to Produce CRL Pieces	24
4.4	Operation of C2C Epidemic Method	26
4.5	FSM of C2C Epidemic Method for Vehicles (IEEE 802.11g)	27
4.6	FSM of MPB Method for Vehicles (IEEE 802.11g)	28
4.7	Hidden Station Problem in Wireless Communication	29
5.1	Hypothetical View of the List of CRL Pieces a Vehicle Node has Received So Far	32
5.2	Multiple Vehicle Nodes in the Same Radio Range of Each Other	33
5.3	Worst Case Scenario in ICE	34
6.1	Topology of a Simple VC System	38
6.2	Protocol Stack for Different Elements According to .ned File	40
6.3	80211MultiChannel in MiXiM	41
6.4	Cross-layer Communication	42
6.5	Inheritance Hierarchy of Different Applications in Different Elements	44
6.6	Broadcasting of CRL Pieces by an RSU	44
6.7	Using BonnMotion for Mobile Scenario Generation	45
6.8	Using the “Export” tab in OpenStreetMap to Get the Map of an Area in Stockholm	47
6.9	Network Definition in SUMO	48
6.10	Traffic-demand Definition in SUMO	49
6.11	Number of RSUs each Vehicle Encounters	50
7.1	Convergence Time in Each Scenario	53
7.2	Number of Received CRL Pieces in I2V and V2V Communication	55
7.3	Number of Total Frames Received with Error in Each Scenario	56

7.4	List of Possessed CRL Pieces by a Selected Node When Shuffling Is Off	57
7.5	List of Possessed CRL Pieces by a Selected Node when Shuffling Is On	57
7.6	Convergence Time in each Scenario When Shuffling is on/off	58
7.7	Effect of Erasure Coding on Convergence Time	59
7.8	Effect of Erasure Coding on Received CRL Pieces	59
7.9	Convergence Time in Each Scenario When Using Magic Cars	60
7.10	Number of Broadcasts in All RSUs	61
7.11	Number of Vehicles that Have Received All the CRL Pieces Over Time	62
7.12	Number of Received CRL Pieces in I2V and V2V Communication	63
7.13	Number of Total Frames Received with Error in Each Scenario	64
7.14	Number of Broadcasts in All RSUs	64

List of Tables

4.1	Beacon message format in C2C Epidemic.	25
4.2	PList message format in C2C Epidemic.	26
4.3	Beacon message format in MPB.	27
5.1	Beacon Message Format in ICE	31
6.1	Protocol stack of each element within the network	40
6.2	Application Layer Parameters for Different Elements.	43

List of Acronyms

AP	Access Point
CA	Certificate Authority
CCH	Control Channel
CIDR	Classless Inter-Domain Routing
Cmdenv	Command-line Interface
CRL	Certificate Revocation List
EEBL	Emergency Electronic Brake Light
EVs	Emergency Vehicles
FCW	Forward Collision Warning
FEC	Forward Error Correction
FSM	Finite State Machine
GLOSA	Green Light Optimal Speed Advisory
HSM	Hardware Security Module
ICE	Intelligent CRL Exchange
LEAVE	Local Eviction of Attackers by Voting Evaluators
MANETs	Mobile Ad-hoc Networks
MDS	Misbehavior Detection System
MPB	Most Pieces Broadcast
MT	Mersenne Twister
OBU _s	On-board Units
PKI	Public-key Infrastructure
RDS	Radio Data System
RHSM	Revocation of the HSM

RSU	Roadside Unit
RTC	Revocation of the Trusted Component
SCH	Service Channel
SMT	Secure Message Transmission
SUMO	Simulation of Urban Mobility
TC	Trusted Component
TraNS	Traffic and Network Simulation Environment
TrIaC	Traffic Control Interface
V	Vehicle
V2I	Vehicle-to-Infrastructure Communication
V2V	Vehicle-to-Vehicle Communication
VANET	Vehicular Ad-hoc Network
VC	Vehicular Communication
Veins	Vehicles in Network Simulation

Chapter 1

Introduction

A Vehicular Ad-hoc Network (VANET) is a communication network for vehicles on the road. The vehicles are equipped with the required hardware/software that gives them the intelligence to communicate with each other. This feature enables a range of applications to improve transportation safety and efficiency.

Transportation safety applications include location warning (notifying about hazardous conditions), motorcycle warning, warning of road works, blind spot warning, Forward Collision Warning (FCW), Emergency Electronic Brake Light (EEBL), etc. According to the U.S. Census Bureau report [6], in 2009, there were 10.8 million motor vehicle accidents and 35.9 thousand motor vehicle deaths. A great number of these fatalities could have been prevented by exploiting vehicle-to-vehicle safety messages.

Transportation efficiency applications include traffic re-routing, making way for Emergency Vehicles (EVs) like ambulances or police cars, Green Light Optimal Speed Advisory (GLOSA), etc. Message exchange in VANETs should be secured. Penetration of an adversary to this system can change it into a dangerous weapon to make criminal behavior easier. For example, the adversary can send false information to the network. This false information propagates throughout the system and it can be misleading or hazardous to drivers. Thus, paying attention to security aspects in these systems is really crucial.

Researchers have designed many methods to meet this goal. One of the ways, agreed by most researchers, is through the use of a public-key infrastructure (PKI) that creates, distributes, and revokes digital certificates based on the X.509 standard. Moreover, one important requirement in a secure VANET is privacy. Using short-term certificates or Pseudonyms is one way to achieve privacy.

Each vehicle node is equipped with a private/public key pair, a long-term certificate and multiple short-term certificates called pseudonyms that do not reveal the true identity of the vehicle. Each vehicle uses these pseudonyms alternately, each for a short period of time. The vehicle signs its own messages using the private key corresponding to a pseudonym and attaches the pseudonym to the message. By using the pseudonym, other vehicles can verify if the received message is coming from a legitimate vehicle without revealing the identity of the vehicle.

An important part of any PKI system is certificate revocation. There are some cases that a certificate must be revoked before its expiration. For example a vehicle might misbehave, or a private-key of a vehicle might have been compromised. In these cases, the long-term certificate belonging to the vehicle along with all its short-term certificates must be revoked. Once revoked, messages from that vehicle will be ignored by other vehicles. The revocation is usually done by periodically issuing a certificate revocation list (CRL) by the Certificate Authority (CA).

1.1 Problem Definition

After the creation of a CRL by a CA, the CRL should be distributed in the VC system. In ideal conditions all vehicles receive the CRL right after its publication by the CA; but in reality the CRL distribution takes some time. The important question is how we can distribute the CRL efficiently and in a timely manner throughout the system so that all vehicles receive a genuine copy of it.

The certificate revocation process, as well as the distribution of revocation information, is an open research problem for VC systems. A couple of research projects considered CRL distribution and different methods, such as the methods we term RSU-only [1], C2C Epidemic [2], and Most Pieces Broadcast (MPB) [3], were proposed.

This thesis implements the aforementioned CRL distribution methods and evaluates them using a common framework. With this approach we can compare these methods accurately and point out the limitations of each.

1.2 Method

We start with a general study of different CRL distribution methods. Then we try to create some scenarios to implement each method in packet-level detail.

We assume a VC network with one CA, multiple RSUs (located in predefined locations), and a set of vehicles. We also assume that in all scenarios, the CA generates a single CRL only once. Then one of the distribution methods is used to spread the CRL throughout the VC system. Moreover, the CA uses base CRL ie. It publishes the full CRL rather than using delta-CRL.

The evaluation is done using simulation. Cost and safety dictate that simulations are a necessary tool for doing research in the field of VC systems. Setting up a test bed with all required hardware/-software is really costly. The simulation results are compared to each other based on a couple of predefined criteria.

OMNET++ [4] and MiXiM framework are used for detailed packet-level simulation. BonnMotion [7] and SUMO [5] are also used for generating the vehicle mobility traces. The raw simulation results are fed into a MATLAB script to perform numerical computation.

1.3 Contribution

The C2C Epidemic approach [2] did not provide any implementation in packet-level detail. We design a specific protocol for the C2C Epidemic approach, and provide a packet-level implementation to be able to compare it with other CRL distribution methods.

We also propose a new method for CRL distribution called ICE (Intelligent CRL Exchange). This method uses V2V communication as well as I2V communication to distribute the CRL pieces to vehicles. Our method is similar to the Most Pieces Broadcast (MPB) one, but the vehicle with the highest number of pieces (say vehicle A) does not broadcast its pieces blindly. ICE uses a semi-incremental CRL exchange by taking the CRL pieces possessed by other vehicles in the same radio range into consideration. In other words, vehicle A tries to broadcast only those pieces that nearby vehicles do not have. With this approach, the number of duplicate received pieces decreases in comparison to the MPB method.

Moreover, ICE uses a simple approach to decrease the number of unnecessary broadcasts by RSUs. When a vehicle receives all the required pieces, it announces this in its beacon messages. The RSU receiving this beacon recognizes that the vehicle does not need any further CRL pieces and does not start broadcasting. The RSU starts broadcasting only when a vehicle asks it to do so. The performance of the ICE is compared to the existing methods and the results are presented.

1.4 Thesis Outline

This thesis is organized into eight chapters plus two appendices.

Chapter 2 provides an overview of VANETs, specifically discussing the VANET technology, applications, and security. In this chapter, we also talk about PKIs and certificate revocation in the context of VANETs.

Chapter 3 introduces the CRL distribution topic and presents the currently proposed methods. Each method is discussed with sufficient amount of details and its limitations are pointed out.

Chapter 4 provides the implementation details of the current CRL distribution methods. Eventually, multiple scenarios are defined to compare these method to each other.

Chapter 5 presents our new approach for CRL distribution called ICE.

Chapter 6 dives into the simulation and presents the different parameters for OMNET++ and SUMO simulators.

Chapter 7 provides the evaluation results from the simulation and compares different approaches to each other in respect to some criteria.

Chapter 8 introduces the conclusion and also the future works.

Appendix A presents the calculation for computing the Interference distance in OMNET++. This calculation is done for drawing circles around RSUs or vehicle nodes, showing the radio range of each.

Appendix B enumerates configurations of the machine used for running the simulations.

Chapter 2

Background

This chapter contains the basic principles behind the thesis and provides a brief overview of Vehicular Ad-hoc Networks (VANETs). We will also talk about secure vehicular systems and the concept of certificate renewal and revocation.

2.1 Vehicular Ad-hoc Networks (VANETs)

Vehicular Ad-hoc Network (VANET) is one of the fastest growing technologies in the area of communication technologies. It is a subset of Mobile Ad-hoc Networks (MANETs) in which the nodes are vehicles. Although node movements (mobility pattern) seem random in MANETs, vehicle movements in VANETs tend to be in organized fashion. In section 6.7 and 6.8, we will talk about mobility pattern in VANETs and how we can use a good mobility model to emulate movements of cars in real-life.

The vehicles are equipped with on-board units (OBUs) which give them capability to exchange data between each other.¹ This feature enables a range of applications to improve transportation safety and efficiency. The vehicles can communicate with each other and inform each other from different events.

As an example for transportation safety, a vehicle travelling may detect an environmental hazard (like road slipperiness) or a car accident and notify other approaching vehicles to reduce their speed. As an instance for transportation efficiency, a vehicle can inform about a traffic jam to assist other vehicles in choosing less congested routes.

Another key element in VANETs is the presence of infrastructure points that provide a connection to network services, similar to an access point (AP) in traditional wireless networks. The infrastructure points are normally referred to as Road-Side Units (RSUs) and this term is used in the rest of the

¹Cars are already equipped with sophisticated control system to bring different parts into harmony. Engine control unit (ECU) is an example of electronic control unit that regulates air/fuel mixture, ignition timing, etc. It is important to not confuse these control systems with the aforementioned OBU.

report. The RSUs are also equipped with OBUs to give them intelligence to communicate with other elements in the VANET. Figure 2.1 presents a simple illustration of VANETs.

A vehicle can communicate with other vehicles (in single-hop or multi-hop fashion) using vehicle-to-vehicle (V2V) communication. V2V communications are shown with bidirectional red dashed arrows. A vehicle can also communicate with RSUs using vehicle-to-infrastructure (V2I) communication. V2I communications are shown with bidirectional black solid arrows. For a good survey encompassing the communication and networking aspects of VANETs, refer to [8, 9].

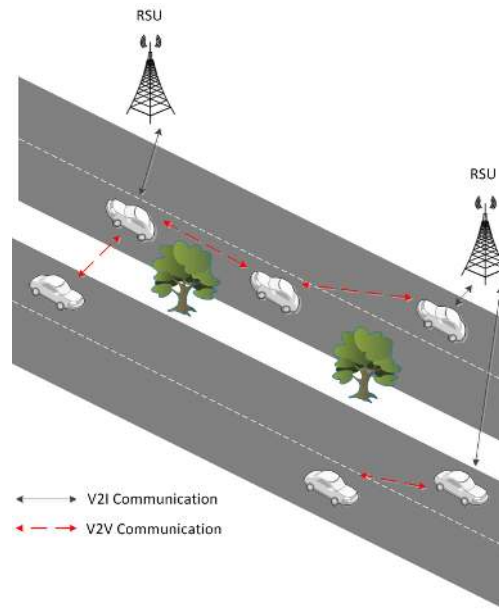


Figure 2.1: Simple Illustration of VANETs

2.2 Secure Vehicular Communication (VC) Systems

Although using VC systems can be beneficial, penetration of an adversary to this system can change it into a dangerous weapon to make criminal behaviour easier. For example the adversary can send false information to the network. A vehicle receiving this false information will notify other nearby vehicles and after sometime a large portion of the network will be contaminated with false information that can be misleading or hazardous to drivers.

As can be seen, security in vehicular communication systems is really important since it directly affects the human safety. Thus, a lot of researches are conducting to secure the VC system and make it immune to different types of attacks. One of the promising approaches which is agreed by most researchers is through the use of a public-key infrastructure (PKI) that creates, distributes, and revokes digital certificates based on X.509 standard.

One of the main components of the PKI is the certification authority (CA) which is responsible for issuing digital certificates. Dedicating only one CA is not enough for a big system like VC, and normally a lot of CAs are involved that each is responsible for a limited number of certificates. The

CAs are usually arranged in hierarchical fashion, and the interaction of vehicles with the CA is done through RSUs.

In the following section, we will present a simplified operation of a secure VC system. This discussion is important due to the fact that certificate revocation which is the focus of this thesis would be the main part of this system. For more detailed information refer to [10, 11].

2.2.1 Simplified Operation of a Secure VC System

As mentioned previously, secure VC systems need a large number of Certification Authorities (CAs) that each is responsible for a specific region (national wide, province, county, etc). Each node X (vehicle or RSU) is registered with only one CA and has a unique long-term identity ID_X . Furthermore, each identity is associated with a private/public key pair denoted by Pub_X and $Priv_X$, respectively.

The CA issues a long-term certificate that binds the identity to the public key. The CA signs this certificate by its private key ($Priv_{CA}$). Figure 2.2 shows this clearly. Note that we used a simple format for the certificate (public key, identity and lifetime). The X.509 standard specifies the format of public key certificates and we will present it later.

Long-term certificate along with public key and identity are stored on the OBU. Private key is stored on the Hardware Security Module (HSM) which is a tamper-proof device and physically separated from OBU.



Node X Identity: ID_X
 Public Key: Pub_X
 Private Key: $Priv_X$
 Long-term Certificate: $Cert_{Priv_{CA}}\{Pub_X, ID_X, lifetime\}$

Figure 2.2: Long-term Identification of Vehicle X

The nodes send different messages for different purposes. Each message includes the actual content m , a signature on m using nodes private key and also the certificate signed by the CA. Figure 2.3 shows the message format.



Figure 2.3: Message Format in Secure VC System

The aforementioned message is received by the nearby nodes. The receiving node first validates the certificate. The public key of the CA is used to decrypt the certificate and extract 'node identity' along with the corresponding 'public key'. By this the receiving node can be sure that the message is coming from a valid node. This process is shown below:

$$Pub_X, ID_X = D_{Pub_{CA}}(Cert)$$

At the next step, the extracted public key is used to verify the signature of the message. On successful verification, the message m is accepted and used by the node. Otherwise the message is discarded, because it has been changed during transmission.

2.2.2 Using Short-term Identification

The scenario described so far has a serious disadvantage. Each node uses its long-term certificate to send messages. This approach violates the privacy. An adversary can track the messages coming from vehicles and finds out the exact path taken by each. For example, the adversary can find out that a vehicle starts its journey from location x and goes to the parliament each day, and interpret that this vehicle maybe belongs to a Member of Parliament and location x is his/her home.

A proposed solution for the above problem is the concept of pseudonymity. Each vehicle is equipped with a couple of certified public keys (pseudonyms) that do not reveal the true identity of the vehicle. A vehicle uses these pseudonyms alternately, each for a short period of time. Thus, these pseudonyms are called short-term certificates.

The vehicle uses these pseudonyms for signing the messages instead of using its long-term certificate. By this the adversary would not be able to link the messages.²

2.2.3 Obtaining Pseudonyms

Researchers have suggested several methods for obtaining pseudonyms like pseudonym on demand [12]. Discussion about different ways to obtain pseudonyms is beyond the scope of this thesis, however, we will present a simple scenario in which a vehicle obtains multiple pseudonyms from the CA.

The vehicle V generates a couple of private/public key-pairs as below that each corresponds to a pseudonym:

$$\begin{aligned} &Priv_V^1, Pub_V^1 \\ &Priv_V^2, Pub_V^2 \\ &Priv_V^3, Pub_V^3 \end{aligned}$$

Then, the vehicle authenticates itself to the CA using its long-term identity and sends the public-keys to the CA using the following message (private keys are stored in HSM). Note that the vehicle signs each Pub_V^i to prove that it has the corresponding private keys.

$$PseuReq\{Pub_V^1, Pub_V^2, Pub_V^3, Pub_V\}, Sig_{Priv_V}\{PseuReq\}, Sig_{Priv_V^1}\{Pub_V^1\}, Sig_{Priv_V^2}\{Pub_V^2\}, Sig_{Priv_V^3}\{Pub_V^3\}$$

²The proposed pseudonym mechanism is not a definite way to provide privacy protection. It might still possible to track vehicles between pseudonym changes. It is obvious that increasing the pseudonym alternation can make life harder for an attacker, but it also increases the overhead. Hence, there is a need for new privacy protection mechanisms. Another technique is based on ‘group signatures’.

Finally, the CA sends the certificate for each pseudonym as below. As can be seen, the pseudonym does not contain the real identity of the vehicle. It contains the ID of pseudonym provider (CA), lifetime and public key. Note that the inclusion of pseudonym provider ID in the certificate implies that the vehicle belongs to a group registered with CA (This group defines the anonymity set of vehicle V).

$$PseuRep\{Pub_V^i, lifetime, CA\}, Sig_{Priv_{CA}}\{PseuRes\}$$

A vehicle needs to contact the CA, infrequently but regularly, to obtain a new set of pseudonyms.

2.2.4 Certificate Renewal and Certificate Revocation

An important part of any PKI system is the certificate renewal and certificate revocation. In this section we will talk about them briefly.

Each certificate (long-term or short-term) has a validity period (lifetime) which defines the period that the issued certificate is valid for use. If there is no problem with a certificate, the CA will issue a new certificate before the old certificate expires. This process is called certificate renewal.

There are some cases that a certificate must be revoked before its expiration. For example a vehicle might misbehave, or private-key of a vehicle might have been compromised. In these cases the long-term certificate belonging to the vehicle along with all its short-term certificates must be revoked. Once revoked, the messages from that vehicle will be ignored by other vehicles. The revocation is done by periodically issuing a certificate revocation list (CRL).

Figure 2.4a and Figure 2.4b show the format of certificate and CRL respectively in the X.509 standard. As can be seen, an entry is dedicated for each revoked certificate in CRL. Each entry consists of the serial number of a certificate along with revocation date for that certificate. Serial numbers are unique in the CA, thus they can be used to identify the certificates uniquely.

The CA signs the CRL and distributes it throughout the VC system. After reception of the CRL by a node, it is verified and then stored on the OBU.

When a vehicle receives a message, it first extracts the serial number of the certificate. Then it searches the CRL to see if the serial number is among the revoked certificates. If there is a match, the received message is dropped. Otherwise the message is accepted and public key of CA is used to get the public key of the sender. Then this public key is used to verify the message integrity.

In PKI system, a couple of factors determine the size of CRL including revocation rate, revocation scheme and number of nodes. In VC system, the number of pseudonyms used by each vehicle also effects the CRL size. The size of CRL can reach tenth of megabytes [3].

2.2.5 Certificate Revocation Schemes

CRL publishing by the CA obeys a particular revocation policy. It can be based on time interval (hourly, daily, monthly, etc.) or a certain number of revocations. If the revocation rate is very low, then the CRL size would be small and the CA can publish the full and complete CRL.

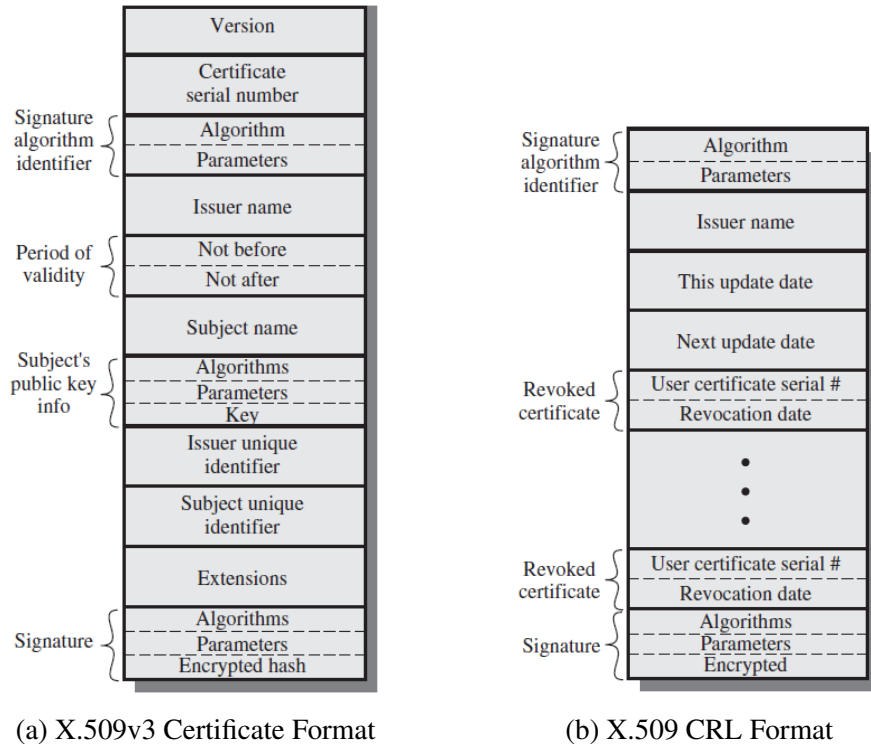


Figure 2.4: Certificate and CRL Format in X.509 Standard [13]

On the other hand, if the revocation rate is high, then the CRL would be big in size and changes more often. Thus transmission of the full CRL by CA is inefficient in terms of network resources. In this case it is much better to send the updated CRL rather than the full CRL. A couple of approaches have been proposed for doing this like Delta CRL, Over-Issued CRL.

Some master theses like [14, 15] provide a good survey and an analysis of different schemes for public key certificate revocation. Moreover, Partitioned CRL [16] and Compressed CRL using Bloom Filter [17] are also two different approaches proposed for reducing the CRL size.

CRL publishing in VC systems is rather infrequent (eg. once per day) [1] and this leaves a vulnerability window until a faulty node is revoked. For example suppose that vehicle V is identified as faulty just after the last CRL publishing. It takes a day for the new CRL to be distributed throughout the VC system and meanwhile the faulty node V can still communicate with other vehicles.

To address the aforementioned problem, [17] proposed that the vehicles themselves can detect the faulty node by locally voting off and excluding the misbehaving vehicle nodes. For doing this, two schemes were proposed by [17] called MDS (Misbehavior Detection System) and LEAVE (Local Eviction of Attackers by Voting Evaluators). Further discussion is outside the scope of this master thesis.

2.2.6 Revocation of the Trusted Component

[17] proposed RTC (Revocation of the Trusted Component), a method for revocation of a misbehaving node from VC system.³ When the CA decides to revoke a vehicle V, it generates a kill message that is encrypted with V's public key. Then the CA signs the message and sends it with the help of RSUs to the vehicle.

The kill message instructs the TC of vehicle V to erase everything from its memory including its own private keys. This action halts the operation of the vehicle and the vehicle would not be able to sign any messages or generate any new keys.

“The CA determines the location of the vehicle and sends the kill command via the nearest RSU(s). The TC has to confirm the reception of the kill command by sending a signed ACK before erasing the long-term signature. If communication via the RSUs fails (i.e., an ACK is not received after a timeout), the CA can broadcast the command via the RDS (Radio Data System).” [10]

This method assumes that the TC will follow the directions from the CA, however it is obvious that this assumption is not true in all cases. If the adversary controls the communication between CA-TC then RTC method would not work. In this case the CA will not receive any acknowledgments and thus uses CRL-based revocation.

2.3 Beaconing

Vehicle in the VC system periodically broadcast beacon messages. These beacons are single-hop broadcasts that are used for cooperative awareness applications. They contain some information about the vehicle like current position, speed, etc. In most cases, the interval of beaconing is expected to be in the range of 100ms to 1s. The vehicle uses the private key corresponding to the current pseudonym to sign the beacon.

³Trust component is another name for HSM, thus RTC is sometimes called RHSM (Revocation of the HSM).

Chapter 3

Current CRL Distribution Methods

After creation of CRL by CA, it should be distributed in the VC system. In ideal condition all vehicles receive the CRL instantaneously right after its publication by the CA, but in reality distribution of CRL takes some time. The important question is how we can distribute the CRL efficiently and in a timely manner throughout the system in a way that all vehicles receive a genuine copy of it.

A couple of researches considered CRL distribution in the past. We will present these researches here with sufficient details and also discuss the underlying assumptions of each.

3.1 RSU-only CRL Distribution

[1] is one of the earliest papers that talks about CRL distribution in VC system with very good details. Panos et al. designed a scalable and efficient approach for delivering the CRL to all nodes within a region in a reasonable amount of time (tenth of minutes). This paper presents an RSU-only approach in which only V2I communication is responsible for CRL distribution.

In its simplest form, the CA publishes the CRL and splits it into L segments. Then the CA adds header to each piece and then signs it to protect it from being modified. The CA distributes these pieces to the RSUs.

The RSU verifies each piece and then broadcasts the pieces wirelessly in specific intervals. RSU broadcasts the pieces with rate r_B pieces/sec which is chosen in a way that the bandwidth consumed by CRL pieces is much less than C (bandwidth the data-link can support). By this the RSU is not congested for sending time-critical data (such as safety application).

The vehicle within radio range of the RSU receives, validates and then stores each piece. The vehicle is in contact with several RSUs in its route to destination. If a vehicle receives all L pieces, it would be able to reconstruct the original CRL. Figure 3.1 shows this clearly.

As mentioned above, the CA divides the CRL into pieces instead of sending the whole CRL message. This is due to the fact that CRL message can be very big (containing a lot of certificates to be

revoked) and each RSU would be busy broadcasting the whole CRL.

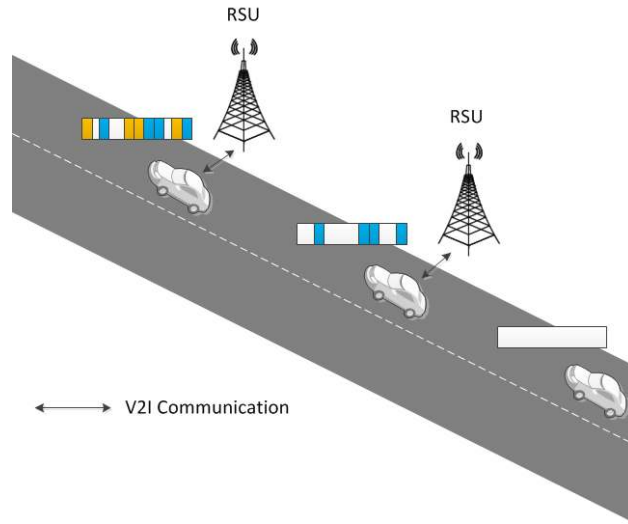


Figure 3.1: V2I Communication in RSU-Only Approach

They also proposed using an Erasure or Fountain code to add redundancy to the system. In this case the CA splits the CRL into L segments, and then uses an Erasure or Fountain code to transform L segments into N pieces. In this case reception of only a portion of CRL pieces in the vehicles would be enough for CRL reconstruction.

In simulation, the time to complete the CRL is measured by varying the range of RSUs, the distance between them, and the bandwidth allocated to the transmission. Some of the main settings of the simulation are as following:

- The network is an urban scenario in the shape of a grid (#) with dimension of 5×10 and 2 lanes per road.
- The SUMO simulator is used for realistic vehicle mobility. Flow rate is 100-400 vehicles per hour per lane and the simulation duration is 1000 seconds.
- IEEE 802.11a protocol is used with typical communication ranges up to 200m.
- Beaconsing rate is 10 beacons per seconds per vehicle

Few points regarding the aforementioned CRL distribution:

- They used a simplified model in simulation in which a simple urban scenario was used and high vehicle traffic densities have not been considered.
- They used a variant of IEEE 802.11 protocol (IEEE 802.11a) in data-link layer and did not consider IEEE 802.11p since it had not been extensively evaluated.
- They have not considered V2V communication as a way to spread CRL pieces. They only relied on RSUs to distribute the CRL pieces by broadcasting. This requires large number of RSUs to be deployed and maintained by the CA. In sparse infrastructure like rural areas,

many vehicles may rarely encounter an RSU, and they would not be able to collect all the required CRL pieces.

3.2 C2C Epidemic CRL Distribution

[2] proposed a C2C epidemic approach for CRL distribution. In this approach, V2V communication is used in conjunction with I2V communication to distribute the CRL throughout the VC system. In this case, as the vehicles come into radio range of each other they exchange their CRL. According to [2], only incremental updates are shared among vehicles, ie. A source vehicle should only share the part of the CRL which the recipient does not have but the source does.

By this, the CRL spreads quickly despite a minimal deployment of RSUs. Note that the RSUs are still needed to disseminate the CRL initially before V2V distribution can be used so that some vehicles have CRL to share. Figure 3.2 shows this clearly.

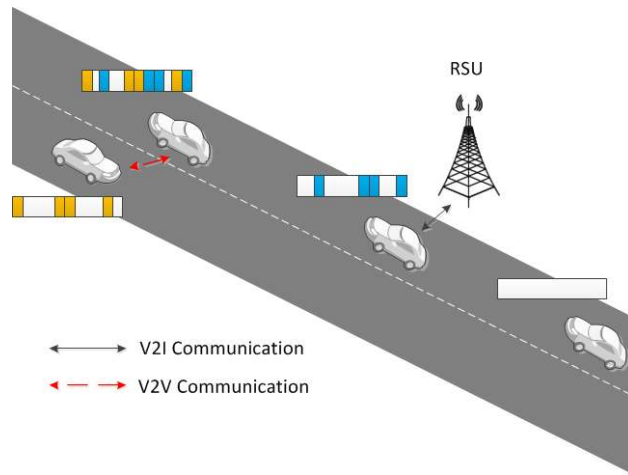


Figure 3.2: V2I and V2V Communications in C2C Epidemic

For evaluation, they used a large-scale simulation based on realistic mobility traces encompassing nearly 260,000 vehicles in an area surrounding Zurich, approximately $354 \text{ km} \times 263 \text{ km}$ in size. A simple infection model was used rather than using a detailed packet-level simulation. CRL update is exchanged if two nodes are within 100 meters of each other for at least X seconds. X is called *association time* and is varied between 0.1, 1 and 2 seconds in the simulation.

The results show that by using V2V communication and only a single RSU, the CRL update was distributed to more than 99 percent of the vehicles at the end of the 76000 seconds simulation. This compares to a completion rate of about 92 percent using 325 RSUs with no V2V communication and association time of 0.1 second (best case).

The simulation does not attempt to emulate packet-level communication. Their simulation model simply consists of a time-contact model that does not take radio properties into consideration. They simply ignore many implementation details.

3.3 Most Pieces Broadcast (MPB) CRL Distribution

[3] proposed Most Pieces Broadcast (MPB) approach for distribution of CRL pieces. In this method, the node (vehicle or RSU) with biggest number of pieces is selected to broadcast its CRL pieces. The rest of the nearby nodes remain silent and receive the pieces. This approach reduces the number of collisions in the wireless channel by reducing the number of broadcasting nodes to one.

MPB requires the nodes to send specific information to other nearby nodes. Instead of creating a custom message for this purpose, it uses the base format of the beacon message and adds some custom fields.

Three fields are added to the beacon messages which are CA identifier (8 bytes), CRL serial (4 bytes) and piece-count (2 bytes). The 2-tuple <CA identifier, CRL serial> uniquely identifies a specific CRL in the VC system and the piece-count shows the number of CRL pieces which the node has. With 2-byte piece-count, the CRL can be divided into maximum of 65,535 pieces which is enough for most situations.

Furthermore, each node (vehicle or RSU) has a counter variable that shows the number of nearby nodes that possess more pieces than the current node. This variable is initialized with zero and it is incremented by one, every time a beacon message received with higher piece-count value.

Note that RSUs also send out beacon messages. If an RSU is within radio range of a vehicle, the RSU will always increment the counter of vehicle since the RSU will always have the complete CRL.

IEEE 802.11p protocol is used and all nodes alternate between the control channel (CCH) and a service channel (SCH) every 50 milliseconds in synchronization. During the CCH interval, the nodes exchange beacon messages and update their counter variable appropriately.

If counter of a node is zero (the node has the most number of pieces within its listening range), then it will broadcast its pieces during the following SCH interval. Otherwise, if counter of a node is greater than zero (other nodes within its radio range have more CRL pieces), then it will remain silent and listens for broadcast pieces during the SCH interval. Nodes reset the counter variable at the beginning of every CCH interval.

If no pieces are received for a time period equal to T_{WAIT} , the silent vehicle begins to broadcast. T_{WAIT} is a fraction of the SCH interval and is equal to 576 us times the value of count variable (number of nodes with more pieces) plus the time required to transmit a packet.

[18] provides a good illustration of MPB method as shown in Figure 3.3. Numbers inside the diamonds show the node number and also the number of pieces possessed by that node. The table shows the counter value as well as T_{WAIT} for each node at the end of CCH interval.

For example, node 1 receives beacons from nodes 3, 5, 6 and 7 during the CCH interval, all of which have more CRL pieces than node 1; therefore, the counter of node 1, is incremented to 4. Thus, it will remain silent for the duration of the SCH interval and receives the pieces from node 7.

Note that node 5 will not receive any pieces since node 6, as well as nodes 1 and 3, is silent. Thus, node 5 will wait for T_{WAIT} and then begin to broadcast pieces.

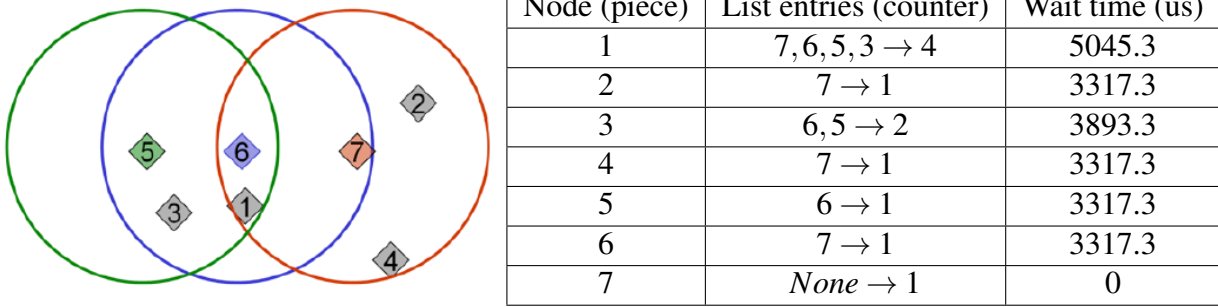


Figure 3.3: Example of Most Pieces Broadcast [18]

The NS-3 simulator was used for evaluation. Vehicles use RandomDirection model and two different scenarios were examined, one with five RSUs, each 450 meters apart with overlapping radio range (dense infrastructure), and one with two RSUs 1650 meters apart (sparse infrastructure).

Simulation uses Raptor code for generating redundancy. The 1 MB CRL is divided into 2000 pieces, each 500 bytes in size. Raptor code with coding rate of 200%, and coding overhead of 5% is applied, and 4000 pieces were generated. Reception of 2100 unique pieces in vehicles is enough for CRL reconstruction.

Few points regarding the aforementioned CRL distribution:

- They incorporate IEEE 802.11p protocol. All nodes alternate between the control channel and a service channel every 50 milliseconds in synchronization! Thus it needs global knowledge of the network.
- Mobility pattern of nodes is naïve (RandomDirection model), and no actual mobility trace is used. We will talk about RandomDirection model in section 6.7.

Chapter 4

Implementation of Current CRL Distribution Methods

The current CRL distribution methods were presented in the previous chapter. Each of these methods should be implemented first and then a couple of scenarios should be defined to be able to compare them to each other. The different scenarios will be presented in chapter 7.

We can divide the scenarios into two broad categories: scenarios that do not use erasure code and scenarios that use erasure code. First, we will talk about these two groups separately. Then we will present the implementation details, and how different CRL distributions are implemented in this project.

4.1 Group 1: Scenarios without Erasure Code

CA Node Operation

The CA publishes a CRL message at time t . This CRL message consists of the revoked certificates. The format of the certificate is very simple and has five fields including the CA name that issues this certificate and also the name of the vehicle that its certificate should be revoked.

The CA treats the CRL message as a raw data (sequence of bytes)¹, and splits the CRL message into segments. Then, a header is added to each of the segments to form CRL pieces. Figure 4.1 shows this process clearly. The header fields are version, timestamp, sequence number, CA ID. Now CA has a couple of CRL pieces and can send them separately to each RSUs.

Note that the CA can shuffle the CRL pieces before sending them to each RSU. This is done to introduce randomness into the CRL distribution. Assume that we have 5 RSUs and the CA transforms CRL message into 5 CRL pieces (p1 to p5). With shuffling, the CRL pieces are received in different order in each RSU, as shown below:

¹Boost Serialization C++ library can be used for serialization of CRL object.

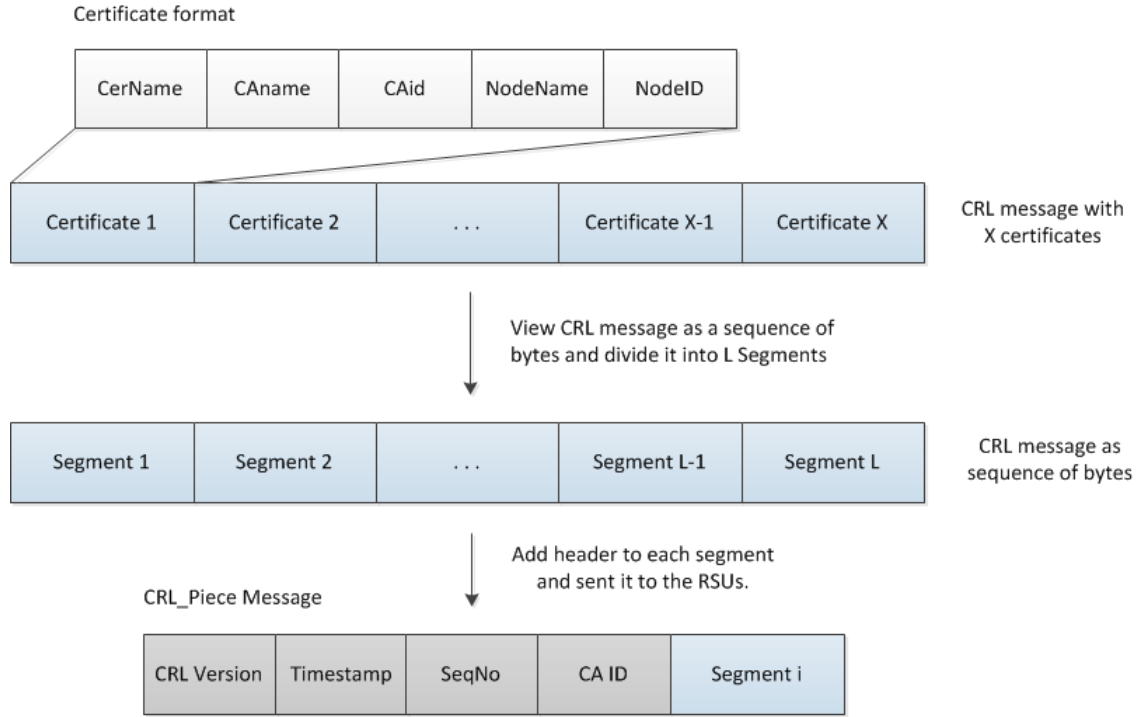


Figure 4.1: Message Format Sent from CA to RSUs

$RSU[0] \rightarrow p1, p2, p3, p4, p5$
 $RSU[1] \rightarrow p3, p5, p4, p1, p2$
 $RSU[2] \rightarrow p4, p1, p5, p2, p3$
 $RSU[3] \rightarrow p2, p4, p1, p5, p3$
 $RSU[4] \rightarrow p5, p3, p2, p1, p4$

RSU Nodes Operation

The RSU receives each CRL pieces from CA and stores a copy of it for future broadcasting (without any modification). We have more than one RSU in the VC system. These RSUs do not request any service from CA. Similarly, no communication is done between RSUs with respect to CRL distribution.

In the RSU-only and C2C Epidemic methods, every RSU broadcasts the CRL pieces periodically. In other words, an RSU always broadcasts the CRL pieces in specific intervals even if no vehicle is in its radio range. In contrast, the RSUs do not broadcast CRL pieces periodically in MPB. RSUs send beacon messages like vehicles, and when a vehicle enters into the radio range of an RSU, they both will be notified about each other, and the RSU can broadcast CRL pieces if needed.

Note that RSUs are not only responsible for broadcasting CRL pieces. They are also responsible for sending time-critical information to the network (such as safety applications). Thus, an RSU must not be kept busy sending big CRL for a long time.

We defined a parameter called `I2V_tho` that restricts the number of CRL pieces that an RSU can broadcast. After `I2V_tho`, the RSU stops sending the CRL pieces, and resumes sending at next broadcast. By changing the `I2V_tho` parameter, we can control the number of CRL pieces which are sent in each broadcast by RSUs.

Vehicle Nodes Operation

Depending on the CRL distribution algorithm, a vehicle can receive pieces from RSUs or nearby vehicles. A vehicle should receive all the CRL pieces to be able to reconstruct the original CRL message. Later on, we will incorporate erasure coding in which even one or more missing piece will not prevent vehicles from reconstructing the original CRL message.

Also note that a vehicle can receive a duplicate piece. In other words, a vehicle can receive a piece which has been received earlier. In this case, the vehicle simply drops the newly received piece.

4.2 Group 2: Scenarios with Erasure Code

As we said in the previous section, a vehicle should receive all the CRL pieces to reconstruct the original CRL message. If each of the pieces is received with error, the MAC layer simply discards it. It is also possible that a CRL piece collides with other messages and be destroyed. These cases are common in wireless communications.

To enhance the robustness of CRL distribution, erasure codes can be exploited. Erasure code is a type of forward error correction (FEC) code that can be used to reconstruct the message at receiver even in case of error. In nutshell, an erasure code transforms a message of k pieces into a code word with n pieces ($n > k$) in a way that the original message can be reconstructed from a subset of the n pieces.

In this thesis, the Rabin's algorithm [19] is used as an erasure code and we will talk about it shortly.

4.2.1 Rabin's Algorithm

The operation of Rabin's algorithm as an erasure code is depicted in Figure 4.2. We have a message with length F and Rabin's algorithm is applied on it. Later on, we will show how this technique can be applied on a CRL.

Step A) The algorithm treats the message as raw data and views it as a stream of integers, m bits each. The value of each integer will be in the range of $[0, 2^m - 1]$.

Step B) The message of length F is divided into L segments. Each segment contains M integers values. It is obvious that if F is not multiple of M , then we need a padding for the last segment. In other words, if F is divisible by M , then we have $L = F/M$ pieces, otherwise we would have $L = (F/M) + 1$ pieces, with some zero integers added as padding in the last segment.

Step C) Some mathematical calculations are done on these segments which results in N pieces. Each piece contains L integers. These are the CRL pieces that can be sent to the receiver.

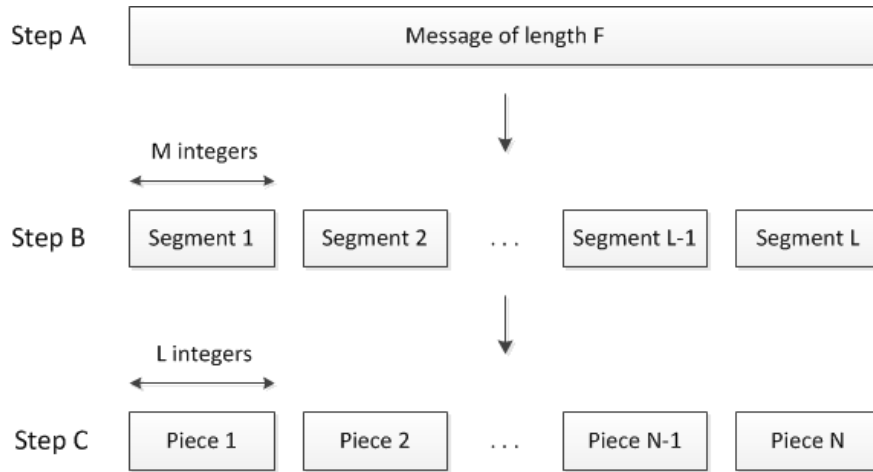


Figure 4.2: Overview of Rabin's Algorithm

Reception of any M out of N transmitted pieces results in reconstruction of the original message. For example, if $M = 3$ and $N = 4$ then reception of any three pieces out of four is sufficient for reconstruction of the message.

Values of M and N can be adjusted according to our needs. They are chosen to be large and more specially $N \gg M$. Large M along with $N \gg M$ increases resilience to errors and less RSU connection time. Furthermore, it decreases the probability of reception of duplicate pieces (as shown in simulation results). It is obvious that selecting a large value for M and N increases coding/decoding complexity.

Coding rate and *Coding overhead* are two parameters that are normally used for describing the coding technique. In the case of erasure codes, coding rate simply defines the ratio of additional pieces encoded, and coding overhead defines the ratio of additional pieces (redundant) needed for successful reconstruction of the message.

In the aforementioned Rabin's algorithm, coding rate is L/N meaning that for each L pieces, N pieces are generated. The coding overhead is $(N - M)/N$.

Mathematics behind Rabin's Algorithm

In this section we will present the mathematical calculations behind Rabin's algorithm. We used the same approach that SMT² protocol uses for message dispersion (refer to section 3.2 in [20]).

We assume that m is 8 ($= 1$ byte), and thus integers are in the range of $[0, 255]$. In this case, the message is split into L segments, M bytes each. Then, the segments are organized as columns of matrix B . Matrix B would be M by L in size.

²Secure Message Transmission

Matrix A is formed as a set of rows a_i in which a_i vectors are constructed by selecting N different elements u_i of “finite field mode $p = 256$ ” denoted by $GF(2^8)$ and set a_i is as following:

$$a_i = [1, u_i, \dots, u_i^{M-1}] \quad 1 \leq i \leq N, \text{ and } N < p$$

As illustrated, matrix A is N by M. First column is always 1. Elements in the second column (u_i) are generated randomly in the range of $[0, 255]$. The remaining columns are different powers of u_i . All calculations are done in $GF(2^8)$.

Having calculated matrix A and B, Matrix W is generated by multiplication of these two matrixes as identified below. Each row of matrix W would be one of the N pieces. As can be seen, the sender should calculate matrix A, then, form matrix B from the message, and finally calculate matrix W to get the N pieces.

$$A_{N \times M} \times B_{M \times L} = W_{N \times L} \quad \text{eq1}$$

If the receiver receives any M pieces out of N, it can reconstruct the original message. In this case, the matrix calculation is done in opposite direction. The receiver forms matrix W' from received pieces. The rows in matrix W' , corresponding to missing pieces, are set to zero.

The receiver creates matrix A' . This matrix is same as A, however the rows which are corresponding to the missing pieces are zero. Note that the receiver should know matrix A. A mechanism is needed for the sender and receiver to agree on the same matrix. This can be done through negotiation or defining matrix A globally to be reachable by all stations.

The receiver is looking for matrix B.

$$A'_{N \times M} \times B_{M \times L} = W'_{N \times L} \quad \text{eq2}$$

If we assume that matrix A is square, then we can find matrix B easily using the following equation:

$$B_{M \times L} = [A'_{N \times M}]^{-1} \times W'_{N \times L} \quad \text{eq3}$$

Unfortunately, we cannot assume that matrix A is square in all cases; thus, we should use a general solution for solving. For instance, QR decomposition of matrix with full pivoting can be used. Discussion about this method is out of the scope of this thesis.

Applying Rabin’s Algorithm on CRL

We used Rabin’s algorithm to add redundancy to CRL messages. All the elements in the network use the same version of matrix A. M and N are initialized and matrix A is calculated globally.

Each time the CA wants to send the CRL message, it divides the CRL message into L segments, M bytes each. Then, it creates N number of CRL pieces using $A.B = W$ equation and sends the CRL pieces to RSUs. This process is shown in figure 4.3.

Any M out of N pieces is sufficient for reconstructing the original CRL message. When a node receives M pieces, it forms equation $A'.B = W'$. Matrix A' and Matrix W' are known and we should calculate matrix B. QR decomposition of matrix with full pivoting can be used to find matrix B.³

³Eigen C++ library can be used for matrix calculation.

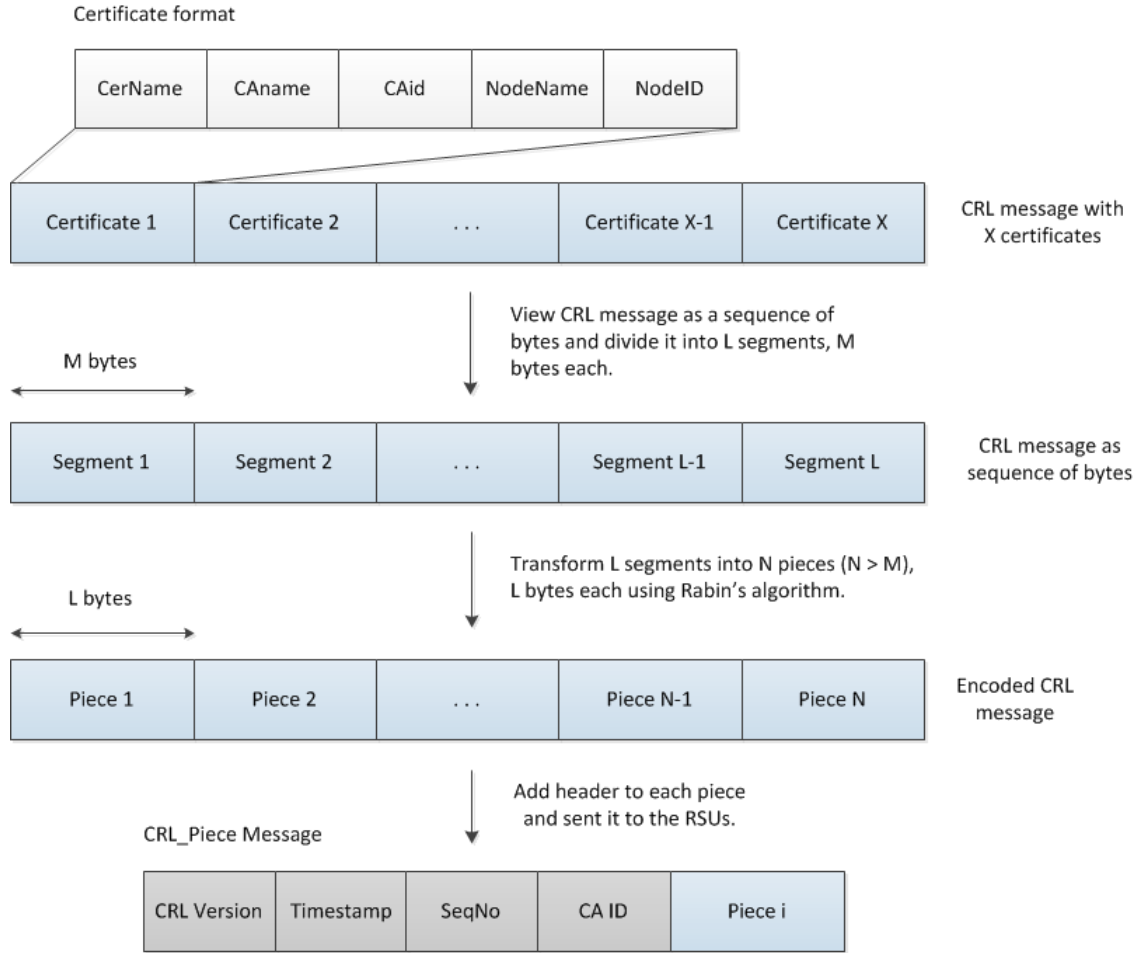


Figure 4.3: Applying Rabin's Algorithm on CRL Message to Produce CRL Pieces

4.3 Using Special Vehicles to receive CRL Magically

As mentioned earlier, vehicles do not have any CRL pieces at first, however they gradually receive CRL pieces from RSUs (I2V communication) or nearby vehicles (V2V communication).

Another method to obtain the CRL pieces is using cellular networks to connect to the CA directly. Cellular networks are virtually everywhere and have a broad coverage. A vehicle can be equipped with required hardware and software to be connected to cellular network. In this case, the vehicle can receive the CRL magically without going into the coverage area of an RSU.

This approach, on the other hand, improves the V2V communication. In fact a vehicle that receives the CRL from cellular network can now actively participate in V2V communication and share its pieces with nearby vehicles. This leads to faster propagation of pieces in VC system.

The main problem with this approach is cost. Connecting to cellular network and exchanging data is not free. The owner of a private vehicle should pay money to be able to use this feature. So all vehicles are not expected to be equipped with this feature. Vehicle owners who can afford this money or some public vehicles can use this. We use the term "Magic Cars" to refer to these kinds

of vehicles.

The number of magic cars in the network is determined by the `MagicCarsRatio` parameter. This parameter determines the ratio of magic cars in the network in percent. Suppose 10 vehicles are in the network and `MagicCarsRatio` is 20. As a result, 20 percent of the 10 vehicles (meaning 2) can connect to the CA directly. These magic vehicles are chosen randomly in the network. The time in which the magic vehicle sends a request to CA for getting the whole CRL is controlled by `MagicReqTime` parameters.

4.4 Implementing C2C Epidemic CRL Distribution

C2C epidemic approach (as proposed in [2]) uses V2V communication in conjunction with I2V communication to distribute the CRL updates. [2] provides no details on how these updates can be exchanged in packet-level. In this section, we will present how the C2C Epidemic is implemented in our scenarios.

We assume that the vehicles broadcast beacon messages periodically. A vehicle can receive beacon messages and be notified about the presence of other nearby vehicles. The format of beacons is depicted in Table 4.1. We are using IEEE 802.11g protocol in data-link layer and channel 1 is always used to send beacon messages.

double	double	double	bool	struct	struct
positionX	positionY	speed	need CRL	certificate	signature

Table 4.1: Beacon message format in C2C Epidemic.

A boolean field called `Need CRL` is added to the beacon message. A vehicle node sets this flag to notify nearby vehicles that it needs CRL pieces. The vehicles that have received all CRL pieces (for example they have received all the pieces directly from the CA using cellular network) can unset this flag. Figure 4.4 shows the operation of C2C Epidemic method in five steps.

- a. `V[0]` sends a broadcast beacon. The `Need CRL` flag in this beacon is set to true, which means that `V[0]` has not received all the required pieces and is looking for more pieces.
- b. `V[1]` receives this broadcast beacon, and is notified about the presence of `V[0]`. Because `Need CRL` flag is true, `V[1]` sends a `PList` message containing the sequence number of pieces that has received so far. Note that if `V[1]` itself needs CRL, it will send the `PList` message even though the `Need CRL` is false.
- c. `V[0]` does the same, and sends a reply `PList` message containing the sequence number of pieces that has received so far. By this approach, both vehicles become aware of the list of CRL pieces that the other vehicle possesses.
- d. `V[0]` starts a timer, and listens to the wireless medium. `V[1]` starts sending the CRL pieces that `V[0]` does not have. `V[0]` receives and stores the pieces.

- e. When the timer of V[0] expires, it starts broadcasting the CRL pieces that V[1] does not have. Next, V[1] receives and stores the pieces. At the end of this step, V[0] and V[1] exchange their CRL pieces in incremental fashion.

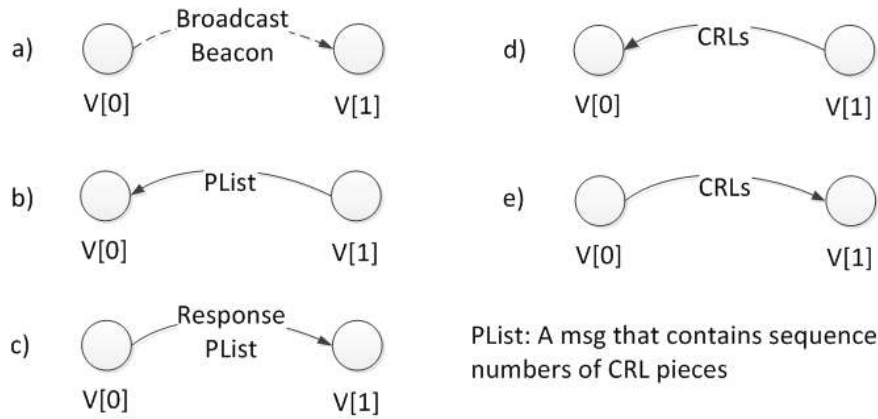


Figure 4.4: Operation of C2C Epidemic Method

In the C2C Epidemic method, the incremental CRL exchange is considered to be pair-wise. This means that the CRL exchange is done between two vehicles that are in the radio range of each other. For example if a third vehicle (V[2]) is present in figure 4.4, it will not participate in the incremental CRL exchange, but it can receive the broadcast pieces from V[0] or V[1]. This is possible due to the special format of the PList message, as shown in table 4.2.

PList message consists of two fields which are serial and list of sequence numbers of CRL pieces. V[1] generates a random number in the range of [0,32767) and stores it in the serial field of the PList message. V[0] echoes this number in the response PList message. Using this, the V[1] node recognizes that the arriving PList message is the corresponding response. This randomly generated number does not disclose the identity of the sending vehicle.

int	string
serial	list of Seq numbers

Table 4.2: PList message format in C2C Epidemic.

Figure 4.5 shows the Finite State Machine (FSM) of the C2C Epidemic method for vehicles. If a vehicle does not receive the corresponding PList message, timer-2 expires after 0.1 second, and the state goes back to IDLE. Timer-1 is initialized with the time it takes for transmission of all CRL pieces:

$$\text{Timer 1} = T_F \text{ of one frame} \times \text{totalPieces} = \frac{(18400 + 32 + 272 + 192) \text{bits}}{\text{bitrate}} \times \text{totalPieces}$$

Recall that if erasure coding is used, then the reception of any M out of N CRL pieces is enough for CRL reconstruction. Reception of M CRL pieces does not prevent the vehicle node from receiving more pieces. In other words, the vehicle can still receive pieces to help other nodes with their missing pieces in future broadcasting.

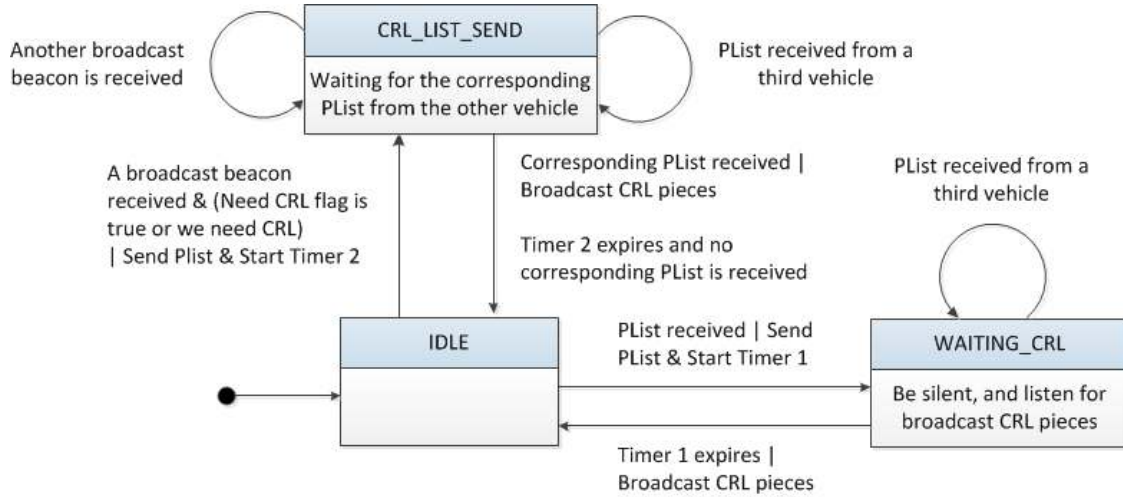


Figure 4.5: FSM of C2C Epidemic Method for Vehicles (IEEE 802.11g)

4.5 Implementing MPB CRL Distribution

MPB method proposed in [3] and we mentioned it earlier. In this section, we will show how this method is implemented in our project.

In MPB method, each node (vehicles or RSUs) broadcasts beacon messages periodically. Table 4.3 demonstrates the format of a beacon message in MPB. A field called piece-count is added to the beacon message that shows the number of CRL pieces which the node has (RSUs always set this field to total number pieces). Beacon messages are sent in channel 1 of IEEE 802.11g protocol.

double	double	double	int	struct	struct
positionX	positionY	speed	piece-count	certificate	signature

Table 4.3: Beacon message format in MPB.

When node A receives a broadcast beacon, it waits for a specific amount of time equal to T_W to receive broadcast beacons from any other nearby nodes (default value of T_W is 1 second). Each node has a counter variable that is initialized with zero. This variable is incremented by one, when a beacon is received with greater piece-count value than the number of A's pieces.

In T_W period, the nodes within the same radio range update their counter variable. After T_W the node with counter variable equal to 0 starts broadcasting all its CRL pieces, while the node with counter value greater than 0 is silent and only listens for broadcast CRL pieces. If no pieces are received for a time period equal to T_{WAIT} , the silent vehicle begins to broadcast.

Figure 4.6 shows the Finite State Machine (FSM) of MPB method for vehicles. Timer-1 is initialized with T_W and Timer-2 is initialized with T_{WAIT} .

The vehicles use channel 1 of IEEE 802.11g for sending beacon messages. They also use the same channel for sending the CRL messages. Thus, when a vehicle is busy sending CRL pieces, it cannot receive any safety beacons. If the vehicle owns a lot of CRL pieces, then broadcasting them all

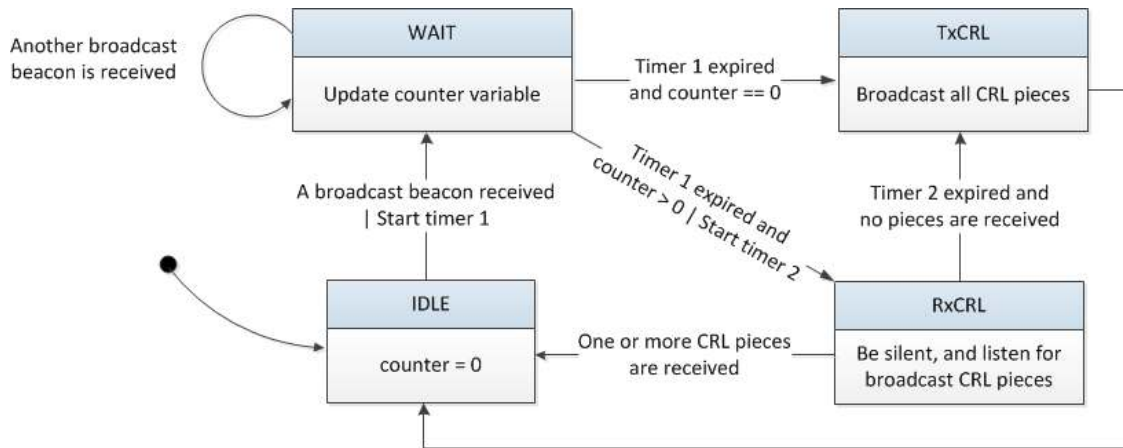


Figure 4.6: FSM of MPB Method for Vehicles (IEEE 802.11g)

is not a good idea, because it keeps the vehicle busy for a relatively long time. The solution is to define a parameter called $V2V_tho$. The purpose of this parameter is similar to $I2V_tho$ mentioned before. Each vehicle sends its CRL pieces in $V2V_tho$ time limit. After $V2V_tho$ the vehicle stops sending the CRL pieces.

Note that in contrast to the C2C Epidemic method, MPB method is not pair-wise. More than two vehicles can participate in the selection process and the vehicle with highest number of pieces is selected to start broadcasting.

4.6 Hidden Station Problem

Both C2C Epidemic and MPB methods suffer from the hidden station problem. The hidden station problem can be best described using figure 4.7. Station A and B are in the transmission range of each other, so are station A and C. Station A can hear any signal transmitted by B or C; but station B and C are hidden from each other with respect to A. Station B and C sense the wireless channel and find it idle; thus they both start sending a frame to station A. These frames collide in A and will be destroyed.

The solution to the hidden station problem in point-point data transmission is the use of optional handshake frames (RTS and CTS) in IEEE 802.11g. RTS/CTS frames are not used in broadcast mode. In broadcast mode a frame is sent to a broadcast address, and all nearby nodes receive it. C2C Epidemic and MPB methods rely on broadcasting frames, thus hidden station problem still exists.

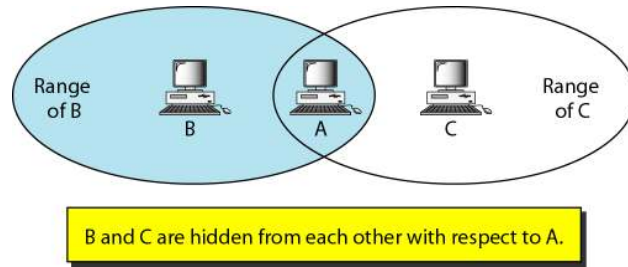


Figure 4.7: Hidden Station Problem in Wireless Communication [21]

Chapter 5

Intelligent CRL Exchange (ICE)

We propose Intelligent CRL Exchange (ICE), a new approach for distribution of CRL pieces in VC systems that uses V2V communication as well as I2V communication. ICE is an enhanced version of MPB. Here, we try to present this method with sufficient detail.

5.1 Beacon Message Format in ICE

Vehicles and RSUs broadcast beacon messages periodically. The nodes that are in the radio range of each other can receive beacon messages and be notified about the presence of each other. The format of a beacon message in ICE is shown in table 5.1. We are using IEEE 802.11g protocol in data-link layer and channel 1 is always used for sending beacon messages.

double	double	double	int	int	struct	struct
positionX	positionY	speed	start index	end index	certificate	signature

Table 5.1: Beacon Message Format in ICE

As can be seen, two new fields are added to the base beacon format which are Start index and End index. These fields are explained in details in the following sections.

5.2 The Need for Incremental CRL Exchange

In the MPB method, exchanging the CRL pieces among nodes is done blindly. In other words, the vehicle node with highest number of pieces starts to broadcast its CRL pieces, without paying attention to the pieces of other vehicles in the same radio range. Although the number of collisions is reduce, the number of duplicate received pieces increases substantially (the simulation results also confirm this statement).

In order to reduce the network load, an incremental CRL exchange can be exploited. In incremental update, a source vehicle should only share the part of CRL which the recipient does not have but the source does. A naïve approach is to include the list of possessed pieces by each node in the beacon messages. What makes this method infeasible is the potentially large number of pieces a node might have. Thus, the size of beacon messages can increase significantly.

We observed that the received CRL pieces by a node tend to be in chunks esp. when the shuffling is disabled (refer to Figure 7.4). We proposed a method based on sending the chunk interval rather than enumerating all the CRL pieces in beacon message. This method is used by ICE and we will talk about it in the next section.

5.3 Start, End Index Fields: Semi-incremental CRL Exchange

Each vehicle keeps track of all the received pieces and selects a range that includes all the missing pieces. The range is identified by two indexes: `Start index` and `End index`, which denote the start and end of the range, respectively. The start index shows the first missing piece and the end index shows the last missing piece.

These two indexes are stored in the beacon message. The nearby node receiving this beacon, checks the range to see if it has the pieces for filling it, and then broadcasts only the appropriate pieces. Next time the sender updates the range and includes the indexes in the beacon.

Figure 5.1 shows a hypothetical view of the CRL pieces that a vehicle has received so far. The received pieces are shaded. Assume that erasure coding is not used and the vehicle should receive all 20 pieces to be able to reconstruct the original CRL message. The node selects the 4-14 range, which includes all the missing pieces and stores 4 and 14 indexes in the beacon message.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----

Figure 5.1: Hypothetical View of the List of CRL Pieces a Vehicle Node has Received So Far

A nearby node receives this beacon and finds out that the sending node is looking for pieces 4 up to 14. Assume that a nearby node has all the pieces, and instead of broadcasting them all, it only broadcast pieces 4 to 14. In this case the number of duplicate received pieces is reduced from 11 to 2.

We can extend this concept to scenarios in which more than two nodes are in the radio range of each other. Vehicle exchange beacon messages and are notified about the ranges of each other. In this way, only the vehicle with smallest range -probably with highest number of pieces- can broadcast the pieces and other vehicles should remain silent while listening to the broadcast.

Assume a scenario in which three vehicles are in the same radio range of each other. The current list of possessed pieces for each vehicle is shown in figure 5.2. Vehicle 1 announces the range 15-19, vehicle 2 informs the range 5-19, and vehicle 3 announces the range 11-13 in their beacon messages. These vehicles exchange beacons and are notified about the range of each other. Vehicle 3 has the

lowest range, thus, it will broadcast the pieces, and the other two vehicles will be silent. Vehicle 3 only broadcasts pieces 5 to 8, and 14 to 19.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----

Vehicle 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----

Vehicle 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----

Vehicle 3

Figure 5.2: Multiple Vehicle Nodes in the Same Radio Range of Each Other

5.4 Prioritizing the CRL Pieces

The selected vehicle for broadcasting cannot broadcast large number of pieces. We cannot keep the radio chip busy sending large number of CRL pieces, since we will miss lots of safety beacon messages. This restriction is imposed by parameter 'tho' in the simulation that we have mentioned before.

Due to the fact that the broadcasting vehicle can send limited number of pieces in each broadcast, it is a good idea to priorities the pieces, and only broadcast the pieces that have higher priority. The question is how we can priorities the pieces. One approach is to give more priority to the pieces that none of the vehicles has received so far.

Based on figure 5.2, assume $tho = 5$. The broadcasting vehicle (vehicle 3) cannot broadcast all 5 to 8, and 14 to 19 pieces as before, and it should choose only 5 pieces. Vehicle 3 can give more priority to pieces 15 to 19, since neither vehicles 1 nor vehicle 2 have received them so far.

The same thing applies to RSUs. An RSU always has the full CRL pieces and can broadcast piece 'i' with higher probability, if none of the vehicles in its radio range has received piece i so far. The reason behind this approach is simple. When vehicles come into the radio range of an RSU, it is a good idea to give vehicles the pieces that none of them has received so far. We expect the vehicles to encounter each other later and exchange their pieces.

5.5 Efficiency of ICE

For ICE to work efficiently, the missing CRL pieces should be in continuous chunks. Figure 5.3 shows the worst case in which only the first and last pieces are missing. Although the vehicle only needs two pieces to be able to reconstruct the original CRL, it announces the range 0-19 which includes the whole CRL.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----

Figure 5.3: Worst Case Scenario in ICE

Fortunately, the above scenario rarely occurs, and has little effect on the performance. But to alleviate the aforementioned problem to some extent, two solutions can be used:

1. The first solution is to disable Shuffling. The shuffling introduces randomness to CRL distribution. The CA shuffles the pieces before sending them to each RSU. By this, the vehicles receive the pieces in random order and V2V communication spreads these pieces further. This randomness is not desirable in the ICE approach, due to the fact that it prevents formation of continuous chunks.
2. The second solution is to add a 'piece count' field to the beacon messages (similar to MPB method), that shows the number of pieces a vehicle has received so far. Although adding the 'piece count' field introduces more overhead, we can specify the possessed pieces of a vehicle more accurately.

In contrast to MPB method that uses one integer-field to announce piece count, the ICE method uses two integer-fields to announce a range covering the missing pieces. The power of ICE is in usage of only two extra fields in the beacon messages. It is obvious that we can introduce more fields to be able to announce more than one range with higher accuracy. This increases the overhead of beacon messages and is not desirable.

Moreover, the size of the integer fields can be reduced depending on the maximum number of pieces. In general, if the maximum number of pieces published by CA is n , then only $\lceil \log_2 n \rceil$ bits is enough for each field. For example, if the maximum number of pieces is 100, then a 7-bit field is sufficient.

5.6 Reducing the Number of Broadcasts

Recall that RSUs do not broadcast CRL pieces periodically in MPB/ICE method. RSUs send beacon messages like vehicles, and when a vehicle enters into the radio range of an RSU, they both will be notified about each other, and the RSU can broadcast CRL pieces. In ICE method, RSUs determine whether a vehicle needs more pieces or not by looking at its beacon messages. The RSUs broadcast CRL pieces if at least one vehicle in its radio range needs more pieces. Using this method, we prevent RSUs from unnecessary broadcasts.

For implementation, we used `Start index` and `End index` fields. A vehicle (say A) that has received all the pieces, set these two fields as bellow (all RSUs do the same because they always have the full pieces).

$$\text{Start index} = \text{End index} + 1 = \text{Number of pieces}$$

An RSU that receives the beacon message from A calculates *diff* as below. If *diff* = -1 then it recognizes that vehicle A does not need any CRL pieces. When an RSU does not receive any

beacon with $diff \geq 0$, then it will cancel the broadcast ¹.

$$diff = \text{End index} - \text{Start index}$$

Moreover, with the advent of Automotive Navigation Systems, the route of a vehicle is known at the start of the journey. We can exploit this information to reduce the number of broadcasts in RSUs. We know which RSUs a vehicle will encounter in its route to destination and instruct only those RSUs to broadcast the CRL pieces.

¹ Note that $diff = 0$ means the vehicle needs only one piece

Chapter 6

Simulation Setup

In this chapter, we will present the basic network design for the VC system in OMNET++ simulator. We also propose the exact configurations and parameters for each element in the network.

6.1 OMNET++ Simulator and MiXiM Framework

OMNET++ is an open-source discrete-event simulator based on C++ language that can be used for modeling communication networks. “OMNET++ attempts to fill the gap between open-source, research-oriented simulation software such as NS-2 and expensive commercial alternatives like OPNET”[4].

OMNET++ simulator only provides the basic tools, but itself does not have any components specifically for simulation of computer networks. For doing this, various simulation models and frameworks were developed like INET, MiXiM, Castalia, etc. The development of these models is completely independent of OMNET++ and they follow their own release cycles.

These models are also open-source. For a list of current OMNET++ models refer to [22]. Each of these models are for different purposes and can be integrated into the OMNET++ IDE.

We needed a model to provide support for mobile wireless networks. Two common modeling frameworks for doing this are MiXiM [23] and Castalia. We decided to use the MiXiM framework due to its detailed models for wireless communication.

“MiXiM is an OMNET++ modeling framework created for mobile and fixed wireless networks (wireless sensor networks, body area networks, ad-hoc networks, vehicular networks, etc.). It offers detailed models of radio wave propagation, interference estimation, radio transceiver power consumption and wireless MAC protocols (e.g. Zigbee).” – From MiXiM Website

OMNET++ version 4.2.2 and MiXiM version 2.2 are used in this thesis (For a detailed list including the list of packages which were used in this thesis, refer to Appendix B).

6.2 Topology of VC Systems

A simplified topology of the VC systems is shown in Figure 6.1. The network is a two dimensional $2200\text{ m} \times 600\text{ m}$ region covered by a dense infrastructure. This network consists of:

- 1 CA which is called CA[0].
- 4 RSUs (RSU[0] to RSU[3]) 500m apart with overlapping radio range.
- 10 vehicles (V[0] to V[9]) with random initial position which are shown with filled circles.

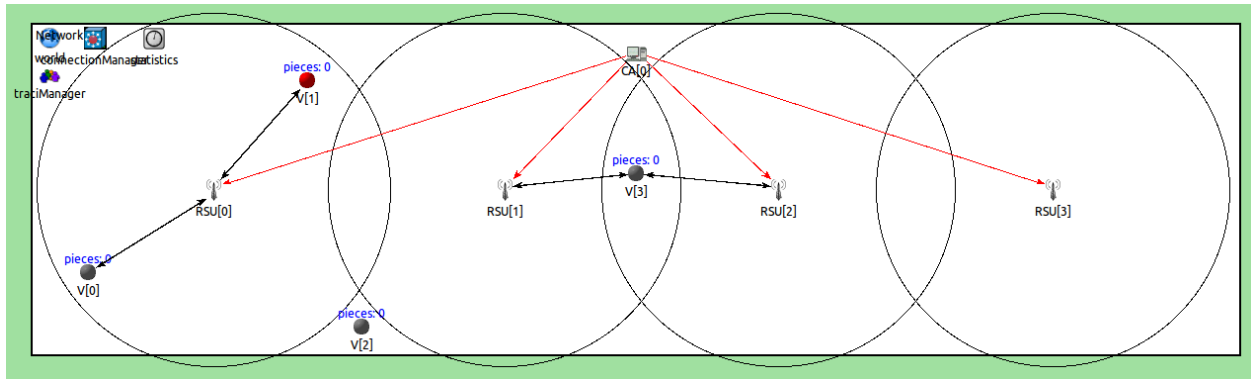


Figure 6.1: Topology of a Simple VC System

Vehicles are the only mobile elements in the network while RSUs have fixed positions. We will talk about the mobility pattern of vehicles in Section 6.7.

As illustrated in figure 6.1, the CA is connected directly to each RSU (red channels). Data rate and propagation delay of each channel is 100Mbps and 10us, respectively. The CA sends the CRL pieces through these channels to each RSU.

Note that these channels are point-to-point; thus, we have no concern for contention. Furthermore, the channels are uni-directional and only the CA can send data to RSUs. The only concern for the CA is to wait for the current frame to be received at the RSU before sending the next frame¹.

All nodes, RSUs and vehicles, use the IEEE802.11g protocol for communication. For illustration purposes, a circle is drawn around each RSU which shows its nominal radio range. If a vehicle is within this range, it can in principle communicate with the RSU at the data-link level. The circle around the vehicle was removed to prevent from a messy network. The radius of this circle is called interference distance and it is approximately 323 meters (refer to appendix A for calculation).

Four vehicles are shown in the picture. Vehicle V[0] in the far left of the picture is within range of RSU[0]; so a bi-directional line is drawn between them in order to show that they are connected in MAC layer. Thus, IEEE802.11g frames can be exchanged between them. Vehicle V[3] is in both coverage of RSU[1] and RSU[2]. V[2] is detached from the system at this moment and it has no connectivity to the RSUs.

¹ This is due to assigning data rate and propagation delay to each channel

OMNET++ simulator assigns a unique id number to each of the modules within the network. This id is used to uniquely identify the vehicles in the network. The id of a vehicle can be regarded as the vehicle registration plate.

“Each module in the network has a unique ID. The module ID is used internally by the simulation kernel to identify modules. Module IDs are guaranteed to be unique for the duration of the whole simulation, even when modules are created and destroyed dynamically; that is, IDs of deleted modules are not reused for newly created modules.”

– From OMNET++ Manual.

6.3 CRL Distribution (In nutshell)

The CA generates a CRL message which consists of revoked certificates. Then the CA creates CRL pieces and sends them all to each RSU through the dedicated red channel that we have discussed in section 6.2. Each RSU stores these CRL pieces and broadcasts them throughout the network.

The vehicles in the radio range of an RSU can receive the CRL pieces and store them internally. A blue text is displayed at the simulation on top of each vehicle which indicates the number of CRL pieces the node has received so far. Initially, this number is zero, however, it will increase gradually when the vehicle receives the CRL pieces.

Using V2V communication, the vehicles can receive the CRL pieces from nearby vehicles as well. Also note that a vehicle can receive a duplicate CRL piece. In other words, a vehicle can receive a CRL piece which has been received earlier. In this case, the vehicle simply drops the newly received piece.

Moreover, some vehicles can connect to the CA directly and get the whole CRL. These vehicles emulate cars with cellular network support. We call these vehicles, “magic cars” and their icon is changed to red to be recognizable from ordinary vehicles (V[1] in Figure 6.1). The number of magic cars in the network is determined by the `MagicCarsRatio` parameter.

6.4 Protocol Stack of Each Element

Each node (vehicle and RSU) consists of 3 layers including data-link layer, network layer and application layer. Transport Layer is not present and the Network layer performs simple logical addressing. For the sake of simplicity, CA only has application layer and sends the CRL directly to the RSUs without sending them through the protocol stack.

The protocol stack of a CA is pretty simple. It only has an application layer which has an output vector gate². The CA is connected to different RSUs through this output vector gate.

² The circle around the arrow shows that this gate is a vector gate

ApplCA	ApplRSU	ApplV
N/A	BaseNetwLayer	BaseNetwLayer
N/A	Nic80211MultiChannel	Nic80211MultiChannel

Table 6.1: Protocol stack of each element within the network

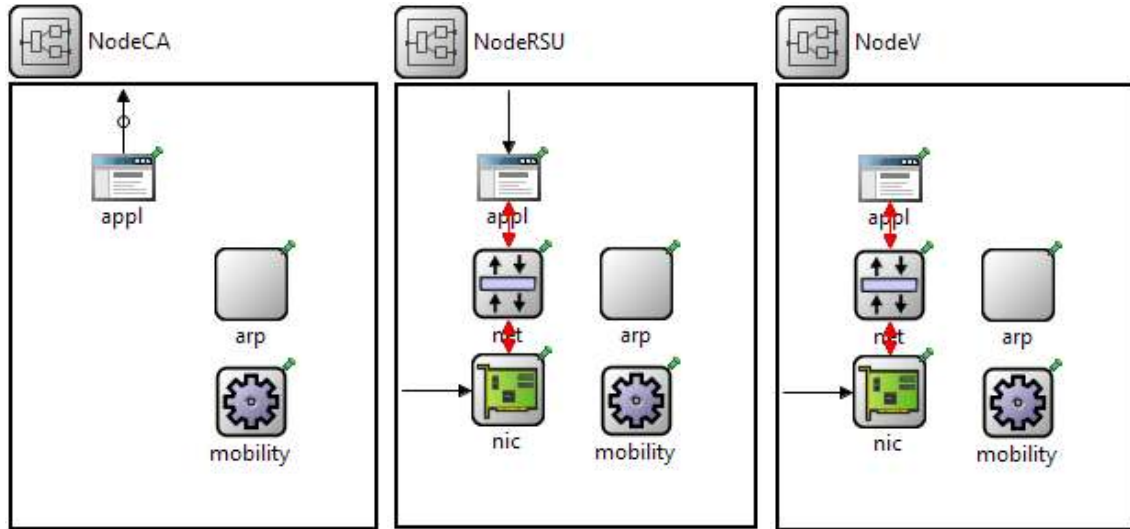


Figure 6.2: Protocol Stack for Different Elements According to .ned File

The protocol stack of RSU and V is very similar except a connection in application layer. Application layer in RSU has an input gate which CA connects to it. As can be seen the transaction between CA and RSUs is done directly from application layer without using the protocol stack.

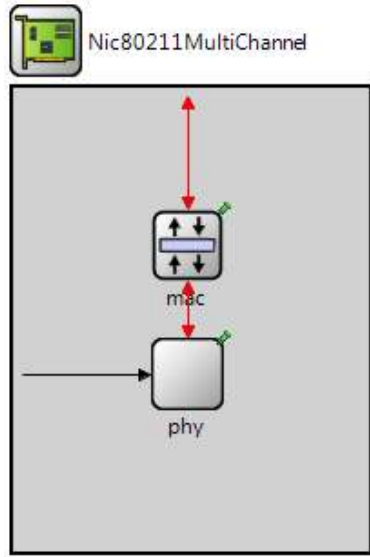
Now let's dig into the layers:

The Application layer is an important part which determines the behavior of each node in the network. Most of the work in this thesis is focused on writing an application layer code for different elements in the network. Each element in the network runs a different code in application layer. The CA node runs the code in ApplCA.cc file and the RSU node runs the code in ApplRSU.cc file. The vehicle nodes can run different algorithms depending on the CRL distribution method being used like ApplV, ApplV_C2C_Epidemic, and ApplV_MPB.

The network layer is very simple. It receives the message from application layer and encapsulates it into a packet by adding a 4 Bytes header. The application layer attaches control information to each message before sending it to the network layer. The control information provides extra information like the destination of the message.

Vehicles and RSUs have a data-link layer that is responsible for encapsulating the network layer packets into frame and sending it through wireless. The Data-link layer is implemented in the NIC³. We used a predefined module called Nic80211MultiChannel. This module implements the IEEE802.11g protocol and supports multi-channel. Figure 6.3a shows the structure of this module.

³Network Interface Card (NIC)



(a) 80211MultiChannel module

MAC layer	headerLength = 272 bit rtsCtsThreshold = 2000 queueLength = 20 bitrate = 2E+6bps autoBitrate = false snr2Mbit = 1.46dB snr5Mbit = 2.6dB snr11Mbit = 5.68dB neighborhoodCacheSize = 30 neighborhoodCacheMaxAge = 100s txPower = 110.11 mW
Physical layer	usePropagationDelay = true thermalNoise = -140dB useThermalNoise = true analogueModel = SimplePathloss maxTXPower = 110.11mW headerLength = 192bit sensitivity = -119.5dBm Decider = 80211MultiChannel nbRadioChannels = 15 initialRadioChannel = 1

(b) Configuration for PHY and MAC

Figure 6.3: 80211MultiChannel in MiXiM

Nic80211MultiChannel is a compound module consisting of mac and phy sub-modules. The configuration for phy and MAC sub-layers is presented in table 6.3b. Note that RTS/CTS feature in VC system is not needed. This feature is disabled by setting its threshold to a high value. Furthermore, thermal noise is enabled in physical sub-layer, but the thermal noise level is set to a low value to reduce the frame corruption. Number of radio channels are 15 and the initial channel is set to 1.

6.5 Cross-layer Communication

RSU and V often use the vertical communication in protocol stack. It means that the application layer generates a message and after adding some control information, passes it down to the network layer. Network layer adds a 4 Bytes header to the message and hands it over to the data-link layer.

Data-link layer consists of two sub-layers which are MAC and physical. IEEE 802.11 protocol receives the network layer message and encapsulates it in a frame and sends it wirelessly throughout the network.

In some cases, we need to have connection between the MAC sub-layer and the application layer. This forms a bi-directional cross-layer communication which is depicted in figure 6.4. Some of the CRL distribution algorithms need this cross-layer communication.

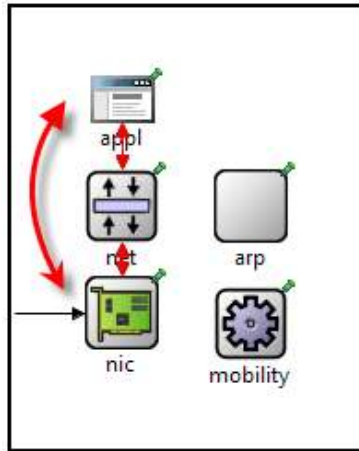


Figure 6.4: Cross-layer Communication

We used the signalling mechanism of the OMNET++ simulator to form this cross-layer communication from MAC to application layer (rather than using dedicated gates). MAC sub-layer generates signals and the application layer receives them in some specific conditions. For example, when a frame is sent successfully or when a frame is received with error. For doing this, we added some codes to the Nic80211MultiChannel module.

On the other hand, the application layer can also communicate with the MAC sub-layer. Application layer can request the current channel number which MAC layer is currently using, or change the channel number if it is desired.

6.6 Application Layer Parameters

Each node in the network has an application layer which determines its behavior. The different CRL distribution algorithms are implemented in this layer and we present the common simulation settings in Table 6.2.

The vehicles can run different algorithms for CRL distribution. This can be done by setting the `appType` parameter to one of the following: `ApplV`, `ApplV_C2C_Epidemic`, `ApplV_MPB`, and `ApplV_ICE`. Figure 6.5 shows the relationship between the different CRL distribution algorithms. `ApplV` class is the base class for all V2V algorithms. This class is responsible for all the common tasks like sending beacons, receiving CRL pieces and storing them internally, etc. `EraseCode` and `SumoMode` are Boolean parameters for turning on/off the respective features.

In the CA, `InitialWait` parameter determines when the CRL pieces should be distributed to the RSUs. The default value is 0, thus this process is done exactly at the start of simulation. `CRLsize` determines the number of certificates in the CRL that should be revoked from the system.

Global	CA
<code>applType = " "</code> <code>EraseCode = True/False</code> <code>SumoMode = True/False</code>	<code>InitialWait = 0</code> <code>CRLsize = 1000 certificates</code> <code>EnableShuffle = True/False</code> <code>NoSegments = 20</code> <code>M = 20</code> <code>N = 30</code>
RSU	Vehicle
<code>CRL_Interval = 100s</code> <code>I2V_tho = 0.1s</code> <code>Beacon_Interval = 1s</code>	<code>Beacon_Interval = 1s</code> <code>V2V_tho = 0.1s</code> <code>MagicCarsRatio = 30</code> <code>MagicReqTime = intuniform(0,60)</code> <code>CRLrecons = True/False</code>

Table 6.2: Application Layer Parameters for Different Elements.

`EnableShuffle` parameter determines if the CA should shuffle the CRL pieces before sending them to each RSU. If the erasure code is not used, `NoSegment` parameter determines the number of pieces the CRL is divided into. Otherwise, if the erasure code is active, `NoSegment` parameter is ignored. The CRL is divided into $N = 30$ pieces and reception of any $M = 20$ pieces is enough for CRL reconstruction.

In the RSUs, the `CRL_Interval` determines the time interval between broadcasts. The default value is 100 seconds meaning that each RSU broadcasts CRL pieces every 100 seconds. The `I2V_tho` parameter determines for how long an RSU is kept busy for broadcasting. The default value is 0.1 so that after 0.1 second the RSU stops broadcasting.

CRL pieces are assumed to have a fixed size of 18400 bits. A 32-bit header is added by the network layer and 272-bit + 192-bit header is attached by IEEE 802.11g protocol in data-link layer. Thus, each frame containing a CRL piece has 18896 bits. IEEE 802.11g with speed of 2 Mbps is used, so the transmission time for each frame is:

$$T_F \text{ for each frame} = \frac{(18400 + 32 + 272 + 192) \text{ bits}}{2 \times 10^6 \text{ Mbps}} = 0.0009448 \text{ second}$$

Assume that the CA produces 20 CRL pieces (p_1 to p_{20}) and it sends them to each RSU (without shuffling). If `I2V_tho` = 0.1, then at most 10 CRL pieces can be sent in each broadcast. As figure 6.6 shows, the RSU sends p_1 to p_{10} in the first broadcast, p_{11} to p_{20} in the second broadcast, and it keeps doing it in the remaining broadcasts.

Although each RSU broadcasts the pieces every 100 seconds, the time for the first broadcasting is chosen randomly according to the following formula. This is done to prevent RSUs from broadcasting the pieces simultaneously. `dblrand` generates a random double number from $[0, 1)$.

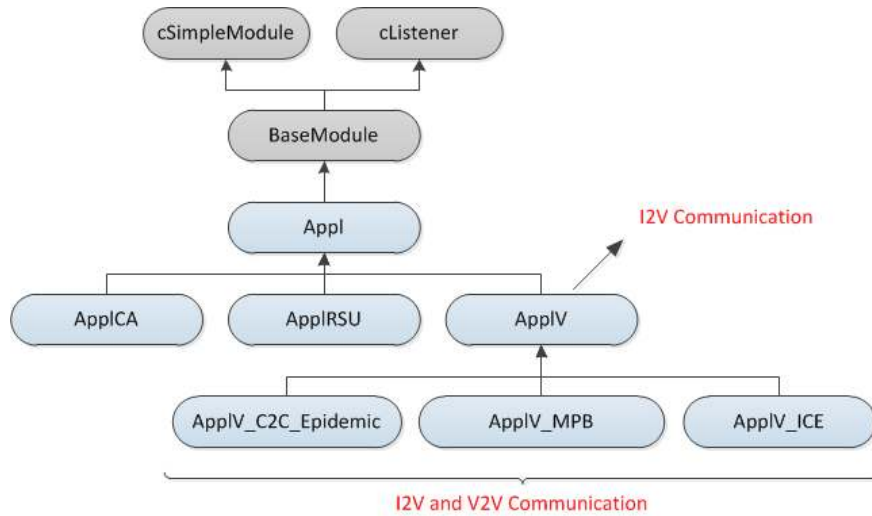


Figure 6.5: Inheritance Hierarchy of Different Applications in Different Elements

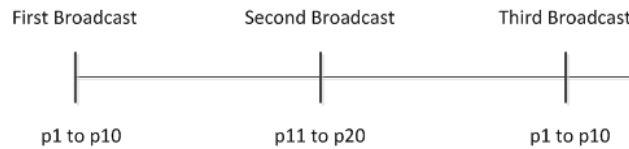


Figure 6.6: Broadcasting of CRL Pieces by an RSU

$$\text{Time of first broadcasting} = \text{dblrand}() \times 10$$

In MPB and ICE methods, the RSUs do not broadcast the CRL pieces periodically. They send beacon messages (like vehicles), while the beaconing rate is determined by `Beacon_Interval` parameter. The default value is 1 second meaning each RSU sends a beacon message every 1 second.

In vehicles, the `Beacon_Interval` determines the broadcast time interval of beacon messages. The default value is 1 second. `V2V_tho` is similar to `I2V_tho` and it prevents vehicles from broadcasting a large number of CRL pieces in V2V communication.

`MagicCarsRatio` shows the number of magic cars (cars that can connect to the CA and get the whole CRL directly) in percent. `MagicReqTime` determines the time in which the magic cars connect to the CA, and is chosen uniformly in the range $[0,60]$ for each vehicle.

6.7 Mobility of Vehicles

Using a good mobility model is crucial in VANET networks because the results of the simulation is highly dependent on the mobility of the vehicles. In the simulation, each vehicle moves according to a predefined mobility model.

At the initial phase of simulation, we used a rudimentary mobility model for the vehicles which is

called RandomDirection [24]. In this case, the vehicles do not leave the network, and they all move in the boundary of the network (white rectangle in Figure 6.1). According to this model, for each vehicle node:

Step 1: A random initial location is chosen.

Step 2: A random direction is chosen (from 0 to 2π in polar coordinate).

Step 3: A random speed is chosen from a range -defined by user previously - say 20 m/s.

Step 4: The node moves with constant speed of 20 m/s toward that direction until it reaches the edge of the network. Then, it bounces off the edge with an angle, determined by the incoming direction. The vehicle then continues along this new track.

The RandomDirection scenario is created with BonnMotion software [7] using the following command:

```
bm -f scenario2 RandomDirection -n 10 -x 2100 -y400 -d 1000 -h 20 -l 20 -p 0
```

-n <number of nodes>
 -x <width of simulation area>
 -y <height of simulation area>
 -d <scenario duration>
 -h <max. speed>
 -l <min. speed>
 -p <max. pause time>

Figure 6.7 shows the process very clearly. The command is fed into the BonnMotion and two files are generated as outputs. We are looking for the mobility traces which are in the compressed file with gz extension. This file is decompressed and used in MiXiM framework.

MiXiM framework has a similar mobility model called “Constant Speed Mobility” which is a simpler version of RandomDirection model, however we preferred to use BonnMotion instead.

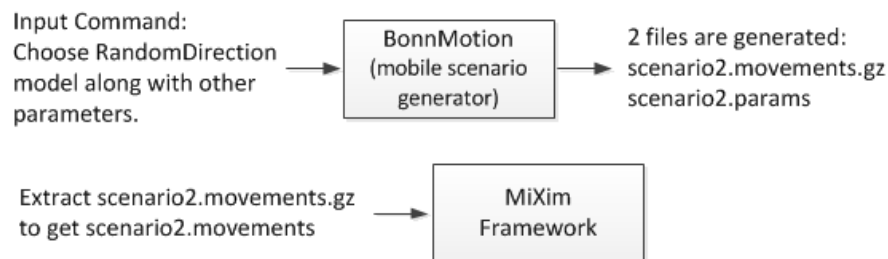


Figure 6.7: Using BonnMotion for Mobile Scenario Generation

Mobility scenarios generated by BonnMotion are often used in MANETs, and they are not suitable for VC systems. RandomDirection model can be used very easily in the simulator and is good for debugging purposes, but it does not resemble the real movement of vehicles in real life. A better solution is to use SUMO for generating mobile scenarios in vehicular communication systems.

6.8 SUMO Simulator

SUMO [5], which is an acronym for “Simulation of Urban Mobility”, is a road traffic simulator. It is mainly developed by Institute of Transportation System at the German Aerospace Center. SUMO is purely microscopic which means that each vehicle is modeled explicitly, has its own route, and moves individually through the network. It is designed to handle large road networks.

MiXiM framework inherently supports BonnMotion, however it cannot read the mobility traces generated by SUMO. We used the TraCI feature of SUMO and Veins framework to connect SUMO and OMNET++ to each other. In the next section we will talk about TraCI and Veins.

6.8.1 TraCI: Online Interaction with SUMO

TraCI which is an acronym for “Traffic Control Interface”, enables external applications to connect to SUMO. Using this approach, other applications can have online interaction with SUMO and send command to it.

The basic idea in TraCI is that a TCP connection is established between the external application and SUMO. SUMO acts as the server while external application as client. For doing this, SUMO should be started with additional command-line option `-remote-port <INT>`. After this, SUMO starts a TCP server process which listens to port `<INT>`, waiting for a client TCP.

Now the external application can initiate a TCP connection (as client process) and connects to the SUMO. External application normally selects a random ephemeral source port number and uses `<INT>` as the destination port number. Thus, the external application should know the port number of TCP server process in SUMO.

The external application that was mentioned above is often a network simulator. In other words the network simulators want to connect to SUMO using TraCI to be able to control it. Several tools have been developed for different simulators to ease this process. We will present three of these tools in the following:

- **TraNS (Traffic and Network Simulation Environment):** This tool [25] is developed to connect the SUMO to NS2 network simulator. TraNS was developed in laboratory for communications and applications (LCA) in EPFL University. According to the TraNS official website, its development is suspended and their developers will not provide any support.
- **iTETRIS:** This tool [26] connects the SUMO to NS3 network simulator.
- **Veins (Vehicles in Network Simulation):** Veins [27] is a framework in OMNET++ simulator that uses MiXiM for wireless communication. It provides some modules which can be used for TraCI interaction with SUMO.

Veins framework is used in this thesis. It provides a Python script called `sumo-launchd.py` that should be executed as below:

```
sumo-launchd.py -vv -c <PATH of SUMO>
```

This script instructs the SUMO simulator to start a TCP server process and listens to port 9999, waiting for a client TCP process. Now the Veins framework can initiate a TCP connection (as client process) and connect to the SUMO. Veins selects a random ephemeral source port number and uses 9999 as the destination port number. Note that all communications between Veins and SUMO take place in the local computer, thus a loopback IP address is used (127.0.0.0/8 in CIDR notation).

Veins framework has a class called “TraCIScenarioManager.cc” which is responsible for handling the TraCI. Establishing the TCP connection and sending/receiving TraCI commands are done via this class.

6.8.2 Generating the Network File from OpenStreetMap

Network file is an XML file which describes the junctions and the roads connecting the junctions together. The junctions are called nodes and the roads are called edges in the network file. These names came from the fact that the network of junctions/roads can be mapped to a graph in which the junctions are nodes and the roads are the edges of the graph.

One way to generate the network file is to import non-SUMO networks. OpenStreetMap format is also supported and can be converted to a compatible network file in SUMO. OpenStreetMap is like Google map and provides the map of the world.⁴ The huge difference is that the OpenStreetMap is free and editable.

In this thesis, we exported a small area in Stockholm as shown in figure 6.8. This gives us an .OSM file containing the map in XML format. We used netconvert application to convert it to a network file in SUMO as shown in figure 6.9.

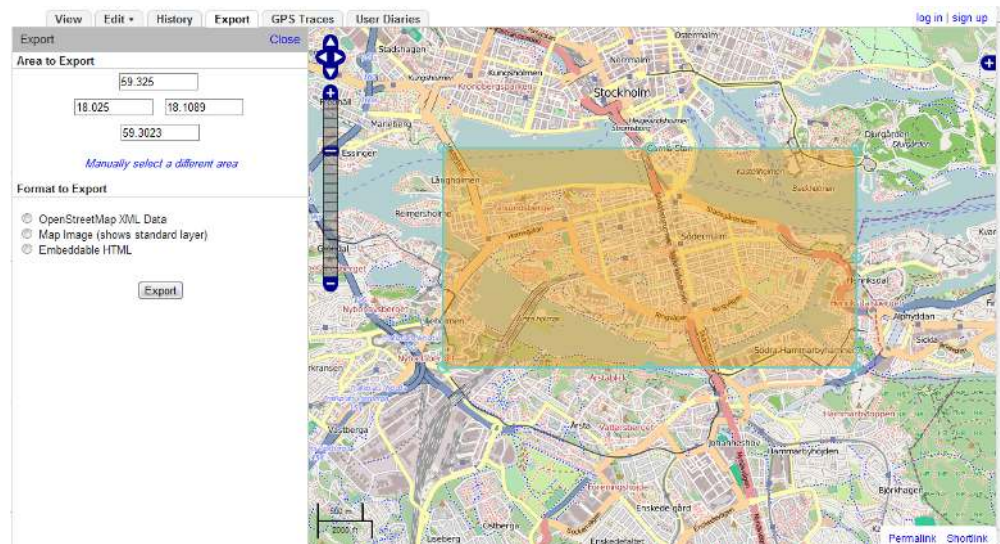


Figure 6.8: Using the “Export” tab in OpenStreetMap to Get the Map of an Area in Stockholm

The intersections we consider have no traffic lights.

⁴ <http://www.openstreetmap.org/>

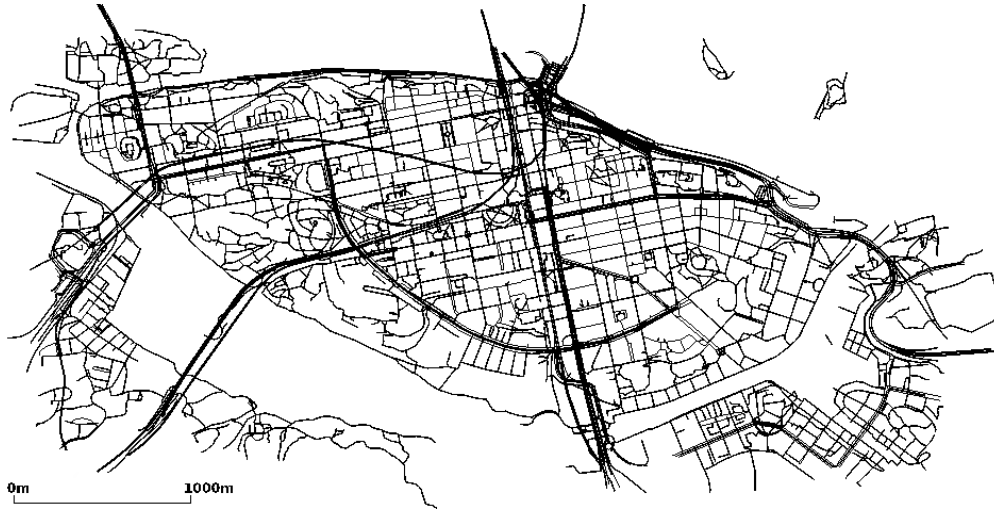


Figure 6.9: Network Definition in SUMO

6.8.3 Generating the Traffic-demand File

With network file, we are able to define the nodes and edges, but we should also define the vehicles and how they move in the network. Defining the vehicles and their movement is done in the traffic-demand file.

Four different types of cars are defined that each has different characteristics in terms of length (m), maximum speed ($\frac{m}{s}$), acceleration ($\frac{m}{s^2}$), deceleration ($\frac{m}{s^2}$), etc. Below is an XML code which shows different types of vehicles in the network.

```

<vType accel="3.0" decel="6.0" id="CarA" length="5.0" minGap="2.5"
  maxSpeed="25.0" sigma="0.5"/>
<vType accel="2.0" decel="5.0" id="CarB" length="5.0" minGap="2.5"
  maxSpeed="22.0" sigma="0.5"/>
<vType accel="1.0" decel="4.0" id="CarC" length="7.5" minGap="3" maxSpeed="17"
  sigma="0.5"/>
<vType accel="1.0" decel="4.0" id="CarD" length="7.5" minGap="3" maxSpeed="17"
  sigma="0.5"/>

```

8 end-points are selected in the network, and 6 of them are chosen for vehicles entrance. The end-points are denoted by a red triangle in figure 6.10. The rate of arrival is 14 vehicles per end-point, and the departure time of each vehicle is chosen randomly in the [0,300] range. As a result, $6 \times 14 = 84$ vehicles enter into the network.

Each vehicle selects one of the end-points randomly and start its journey toward that direction. The route⁵ of a vehicle is chosen according to the shortest-path routing. In order to do this, we used the DUAROUTER application which is provided in the SUMO package. This application calculates the shortest route using the Dijkstra algorithm.

⁵ A route definition contains the first and the last edge, as well as all edges the vehicle will pass.

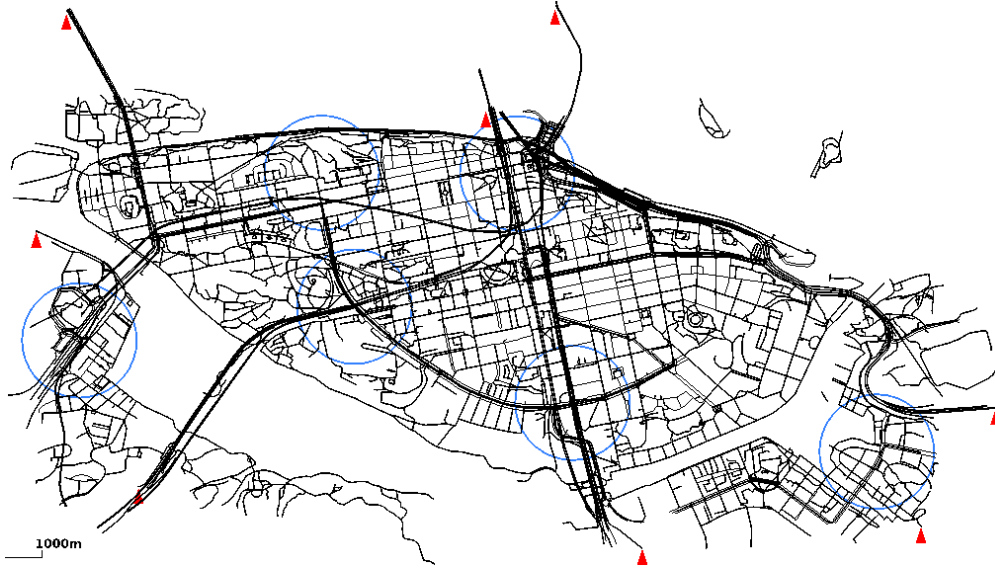


Figure 6.10: Traffic-demand Definition in SUMO

We placed six RSUs in specific locations in the network. The radio coverage of each RSU is shown with a blue circle in figure 6.10.

It is obvious that the more RSUs be placed on the network the better. In this case, it takes less time for a vehicle to receive all the required pieces due to the fact that it encounters more RSUs in its journey to the destination. Deploying large number of RSUs to achieve a dense infrastructure demands high cost, thus, we should pay close attention to where the RSUs should be established. The junctions with heavy traffic can be a good candidate.

Figure 6.11 shows how many RSUs the vehicles encounter in their journey to the destination. For example, 16 vehicles encounter 1 RSU, 30 vehicles encounter 2 RSUs, etc. The normal distribution curve is drawn as well in which $\mu = 2.4$ and $\sigma = 0.96$. It means that in average the vehicles encounter about 3 RSUs, and the dispersion from the average value is not high.

Number of RSUs encountered	0 RSU	1 RSU	2 RSUs	3 RSUs	4 RSUs	5 RSUs	6 RSUs
Number of vehicles	0	16	30	26	12	0	0

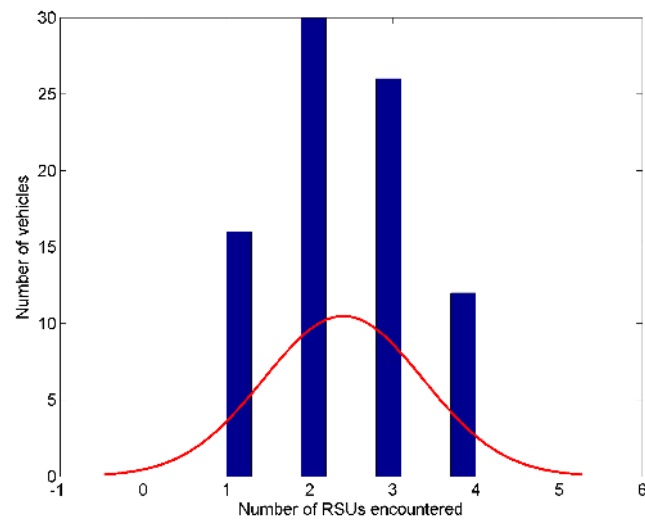


Figure 6.11: Number of RSUs each Vehicle Encounters

Chapter 7

Evaluation Results

This chapter provides the evaluation results from the simulation and it compares different approaches to each other in respect to some criteria.

We used command-line interface (Cmdenv) of OMNET++ to do the simulation. Cmdenv supports batch execution which allows us to run the simulation with different input parameters. Each scenario is repeated 20 times with different initial seed. The result of each run is written to a text file.

We pay close attention to the random number generation. Mersenne Twister (MT) algorithm is used for generating the pseudorandom numbers. Five random number sub-streams are created and each is assigned to a specific module (CA, RSUs, vehicles, etc). We do this to isolate the operation of modules from each other.

After completion of the simulation, the generated txt files are fed to a script in Matlab. Then the average, standard deviation and 95 percent confidence interval¹ are calculated for 20 runs belonging to the same scenario. At the end, a couple of graphs are generated.

7.1 Evaluation Criteria

7.1.1 Convergence time

In the RandomDirection mobility model, all vehicle nodes move within the network boundary. They gradually receive CRL pieces dictated by the CRL distribution method. The time in which a vehicle receives all the required pieces is recorded. When all vehicles in the network reconstruct the original CRL, the simulation is stopped and the current time is recorded as the convergence time. In other words, convergence time shows how long it takes for all the vehicles to receive all the required CRL pieces. It is obvious that lower convergence time for a method indicates a better performance.

¹The 95 percent confidence interval gives a good approximation of the range that is likely to include an unknown population parameter. In other words, if we do the simulation again, the next sample would be in this range with probability of 0.95.

7.1.2 Number of Vehicles that Received the Full set of CRL Pieces

In the SUMO generated scenarios, the vehicles are not bound to the network boundary. Each vehicle has its own path, time of arrival and departure. Vehicles enter the network and gradually receive the pieces as they travel across the network, and eventually leave the network. The simulation is stopped when all vehicles leave the network, thus the simulation time is fixed for all scenario.

When using SUMO, it is possible for a vehicle to only receive a portion of CRL pieces and leaves the network without being able to reconstruct the original CRL. In this case, we record the time in which a vehicle receives the last CRL piece before leaving the network. The number of vehicles that have received all required CRL pieces is used as evaluation criterion. It is obvious that a higher number of vehicles with full CRL pieces indicates a better performance.

7.1.3 Number of Received Pieces

Total number of received CRL pieces is recorded for each scenario. Depending on the CRL distribution method being used, the pieces are received from two different sources: RSUs in I2V communication or nearby vehicles in V2V communication. The number of received pieces from these sources is recorded separately.

Moreover, a vehicle might receive a duplicate piece. We record the total number of duplicate received CRL pieces in each scenario as well. We are looking for a CRL distribution method that reduces the number of duplicate received pieces.

7.1.4 Number of Frames Received with Error

The number of frames received with error in each scenario is recorded. This frame is IEEE 802.11g frame that may contain a beacon message or a CRL piece. When an RSU or a vehicle receives a frame with error, its MAC layer emits a signal and we record it appropriately.

These erroneous frames can be due to collisions in the wireless medium. If two or more different senders in the radio range of each other try to send their frame simultaneously on the shared wireless medium, their frames collide and will be destroyed.

On the other hand, erroneous frame can be due to the thermal noise in the wireless medium. As we mentioned in previous chapter, we also simulate the thermal noise. In this case by increasing the distance between sender and receiver, the probability of error in the received frame also increases.

7.1.5 Number of Broadcasts by RSUs

Recall that each RSU broadcasts the CRL pieces periodically. The number of total broadcasts by RSUs is also recorded. We use this criterion to compare the performance of ICE method with other CRL distribution methods. In the ICE method, if an RSU does not receive any beacon message asking for more pieces, it will stop the next broadcasting.

7.2 RandomDirection Mobility

We used the RandomDirection mobility only for debugging purposes. Although this model does not provide accurate or realistic vehicle movement, it can be used to simulate the interaction of nodes and the effectiveness of different CRL distribution methods.

10 vehicles are moving with RandomDirection mobility in the network depicted in Figure 6.1. This network represents a dense infrastructure. The common simulation parameters are enumerated in Table 6.2. We will present the simulation results here.

7.2.1 Comparing CRL Distribution Methods

We compared different CRL distribution methods in their simplest case possible, ie. shuffling is disabled, no erasure coding is used and without any magic car in the network. The CRL is divided into 20 pieces and each vehicle should receive all 20 pieces to be able to reconstruct the original CRL. Each scenario corresponds to a specific method as listed below:

Scenario 1: RSU-only

Scenario 2: C2C Epidemic

Scenario 3: MPB (Most Pieces Broadcast)

Scenario 4: ICE (Intelligent CRL Exchange)

Convergence Time

Figure 7.1 shows the average convergence time in each scenario with its respective confidence interval. Table 7.1 shows the exact value for each of the scenarios.

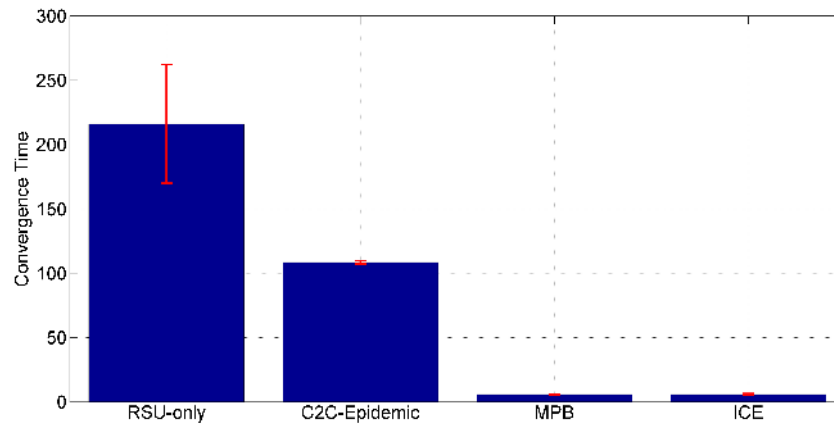


Figure 7.1: Convergence Time in Each Scenario

Scenario	RSU-only	C2C Epidemic	MPB	ICE
Average convergence time	216	108	5.79	5.88

Table 7.1: Result in Figure 7.1

Observation 1: C2C-Epidemic, MPB, and ICE have less convergence time than RSU-only which means it takes less time for all vehicles to receive all the pieces. This is obvious, because they use V2V communication as well as I2V communication for receiving the CRL pieces.

Observation 2: MPB and ICE show better performance in terms of convergence time than C2C Epidemic. This is due to the fact that in MPB and ICE, interaction of vehicles and RSUs is higher. In other words, RSUs do not periodically broadcast CRL pieces. Vehicles request CRL pieces as soon as they encounter an RSU.

Number of Received CRL Pieces

Figure 7.2 shows the number of received CRL pieces from different sources. In other words, illustrates how many pieces are received in I2V communication from RSUs (part a) and how many pieces are received in V2V communications from nearby vehicles (part b).

A “stacked bar” is used for plotting. In each scenario, the blue bar represents the number of duplicate received pieces and the brown bar at the top represents the number of new received pieces. Adding these two bars together in each scenario shows the total number of received pieces.

We have 10 vehicles in the network and the CRL is divided into 20 pieces; so the vehicles receive 200 pieces in total. Thus, the sum of brown bar in ‘figure a’ and ‘figure b’ in each scenario is 200.

Observation 1: In the RSU-only method, we only have I2V communications, so the number of received pieces in V2V communication in figure b is zero.

Observation 2: In C2C-Epidemic, MPB and ICE, by looking at the new received pieces in ‘figure a’ and ‘figure b’, we can conclude that the vehicles receive their CRL pieces more from the I2V communication rather than V2V communication. This shows the high density of RSUs in the network.

Observation 3: In C2C Epidemic method the number of duplicate received pieces in V2V communication is lower than MPB and ICE. This is due to the fact that it uses pair-wise incremental updates in CRL exchange.

Although C2C Epidemic method uses pair-wise incremental updates in V2V communication, the number of duplicate received pieces is not zero. This is because when two nearby vehicles (say A and B) exchange their pieces incrementally, a third vehicle (say C) can also receive the broadcast pieces. In this case, vehicle C can receive some duplicate pieces.

Observation 4: In MPB, the number of duplicate received pieces in V2V communication is high (3.4 times the C2C Epidemic). In average, 190 pieces are received in duplicate in V2V communication and only 42 pieces are new. This is because the node with highest number of pieces (say D) broadcasts blindly. In other words, it does not take the pieces of nearby vehicles

into consideration. Vehicle D might broadcast lots of pieces that the nearby vehicles have already received.

In contrast, this number is low in ICE. This method decreases the number of duplicate received pieces by adding the semi-incremental CRL exchange. As can be seen, 89.8 pieces in average are received in duplicate.

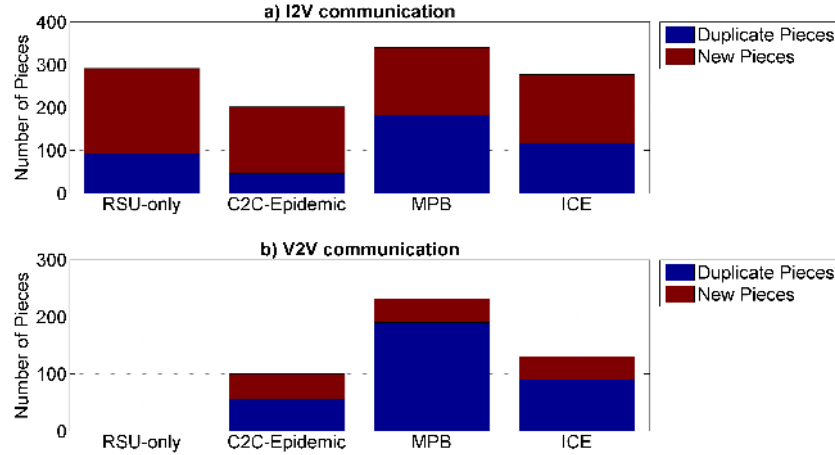


Figure 7.2: Number of Received CRL Pieces in I2V and V2V Communication

		RSU-only	C2C Epidemic	MPB	ICE
Figure a (I2V)	New	200	156	158	160
	Duplicate	91.8	46.3	182	118
Figure b (V2V)	New	0	44.5	42	40.2
	Duplicate	0	55.7	190	89.8

Table 7.2: Result in Figure 7.2

Frames Received with Error

Figure 7.3 demonstrates the average number of frames received with errors for each of the scenarios with its respective confidence interval.

Observation 1: In the RSU-only method, the number of frames received with error is very low, but not zero. The non-zero value is due to collisions of broadcast beacons with broadcast CRL piece frames.

Observation 2: In the C2C Epidemic, the number of frames received with error is high. This is because of the high interaction between vehicle nodes. Apart from the beacon messages, two PList messages should also be exchanged. This increases the probability of collisions.

Observation 3: The number of errors in MPB and ICE is low, and this is due to the fact that the number of broadcasting nodes is reduced to one. Only the node with the highest number of pieces starts broadcasting and the other nearby nodes remain silent.

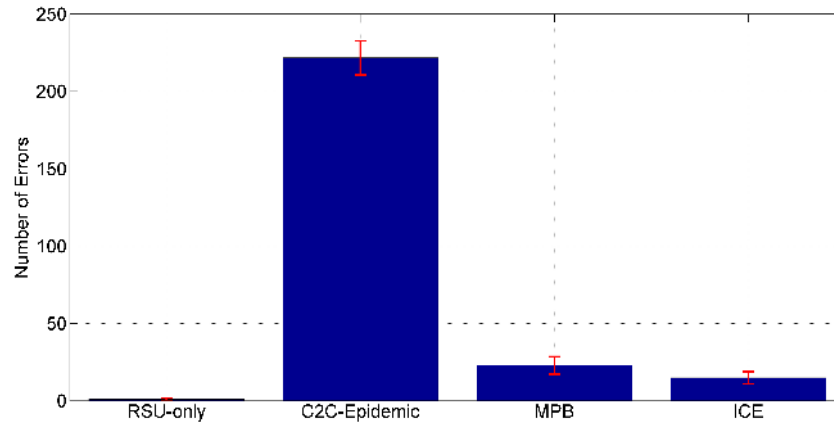


Figure 7.3: Number of Total Frames Received with Error in Each Scenario

Scenario	RSU-only	C2C Epidemic	MPB	ICE
Average # of frames with error	1	222	22.9	14.8

Table 7.3: Result in Figure 7.3

7.2.2 Using Shuffling

We compared different CRL distribution methods when shuffling is enabled i.e. The CA shuffles the CRL pieces before sending them to each RSU. By this approach, each RSU receives CRL pieces in different order and we introduce randomness into the CRL distribution.

To show this we ran a simulation and took a snapshot every 3 seconds from the CRL list of each vehicle in the VC network when MPB is used as the CRL distribution method. Figure 7.4 shows the CRL pieces of a selected vehicle node. The empty and filled rectangles represent the missing and possessed pieces respectively.

As can be seen, the vehicle has a sequential chunk of pieces from 0 to 9 at simulation time 30. This vehicle does not receive any new pieces until time 147. At 147, the rest of CRL pieces from 10 to 19 are received and the vehicle is able to reconstruct the original CRL.

Figure 7.5 illustrates the same scenario, however this time the shuffling is turned on. The vehicle receives 10 pieces at 30, but these pieces are not in sequential order. The vehicle continues to receive random pieces until all missing pieces are filled.

To investigate the effect of shuffling on CRL distribution, eight scenarios are defined as below, and the result is shown in figure 7.6.

Scenario 1: RSU-only, No shuffling
 Scenario 3: C2C Epidemic, No shuffling
 Scenario 5: MPB, No shuffling
 Scenario 7: ICE, No shuffling

Scenario 2: RSU-only, With shuffling
 Scenario 4: C2C Epidemic, With shuffling
 Scenario 6: MPB, With shuffling
 Scenario 8: ICE, With shuffling

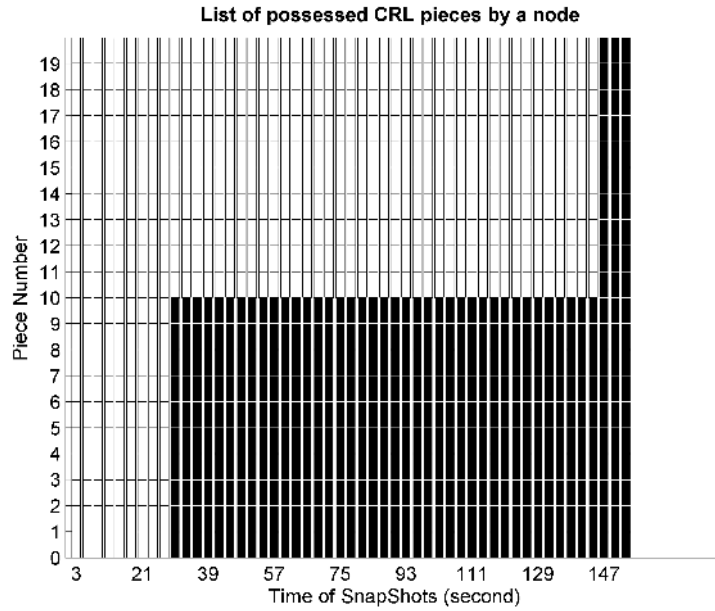


Figure 7.4: List of Possessed CRL Pieces by a Selected Node When Shuffling Is Off

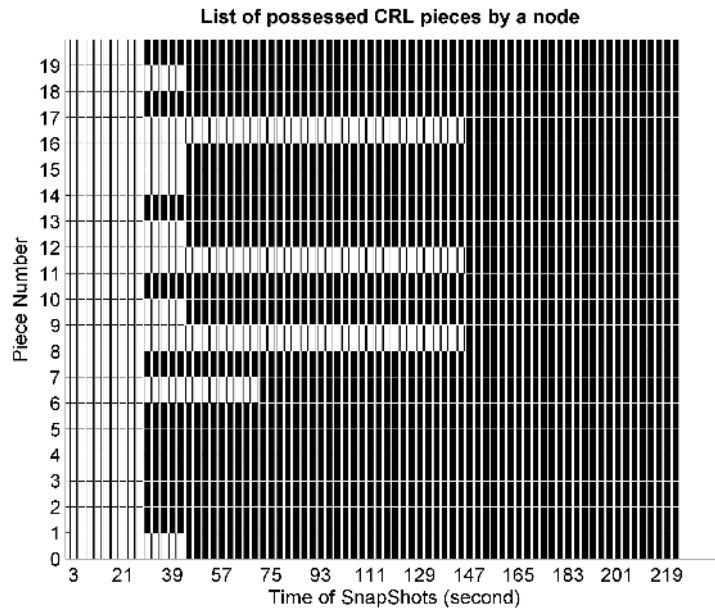


Figure 7.5: List of Possessed CRL Pieces by a Selected Node when Shuffling Is On

Shuffling has not a significant effect on convergence time in almost all the scenarios. The only exception is RSU-only approach where shuffling has a negative effect.

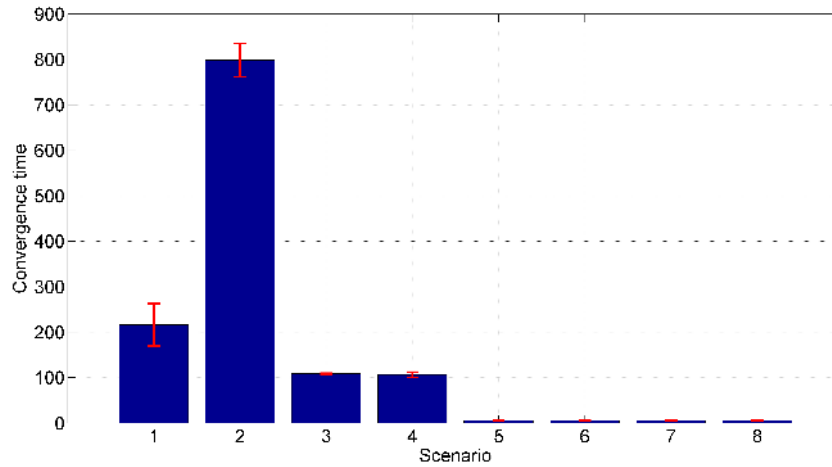


Figure 7.6: Convergence Time in each Scenario When Shuffling is on/off

Scenario	1	2	3	4	5	6	7	8
Average convergence time	216	798	108	106	5.79	5.68	5.88	5.88

Table 7.4: Result in Figure 7.6

7.2.3 Using Erasure Code

Until now the erasure code was disabled, and the CRL was divided into 20 pieces. Each vehicle must receive all the 20 pieces to be able to reconstruct the original CRL. In this section, we investigate the effects of using erasure code. The CRL is divided into 20 segments and is transformed into 30 pieces using Rabin's algorithm. In this case, reception of any 20 pieces is sufficient for CRL reconstruction.

To investigate the effect of erasure code on CRL distribution, eight scenarios are defined as below, and the results are shown in Figure 7.7 and Figure 7.8.

Scenario 1: RSU-only, No Erasure
 Scenario 2: C2C Epidemic, No Erasure
 Scenario 3: MPB, No Erasure
 Scenario 4: ICE, No Erasure

Scenario 5: RSU-only, With Erasure
 Scenario 6: C2C Epidemic, With Erasure
 Scenario 7: MPB, With Erasure
 Scenario 8: ICE, With Erasure

Like before any vehicle should receive 20 pieces, thus, the convergence time does not change significantly as can be seen in figure 7.7. The main effect of an erasure code on the CRL distribution is a decrease in the probability of reception of duplicate pieces. Figure 7.8 confirms this. By comparing the blue bar in the corresponding scenarios (for instance scenario 1 and 5), it can be seen that by using erasure coding, the number of duplicate received pieces in I2V and V2V communications has decreased.

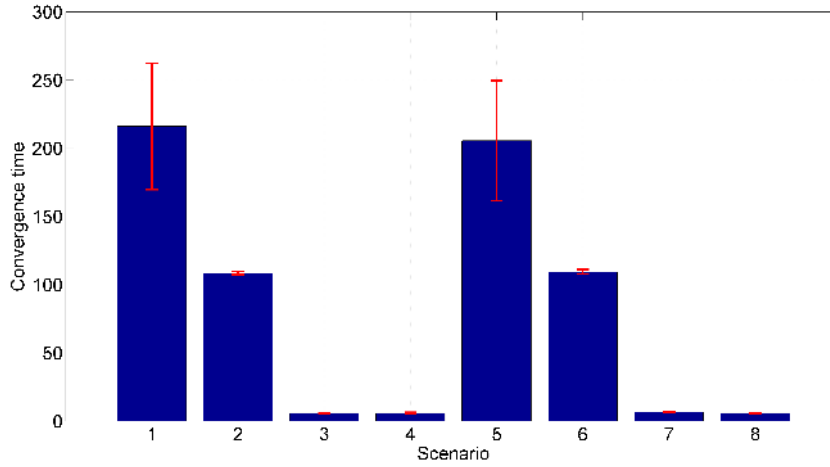


Figure 7.7: Effect of Erasure Coding on Convergence Time

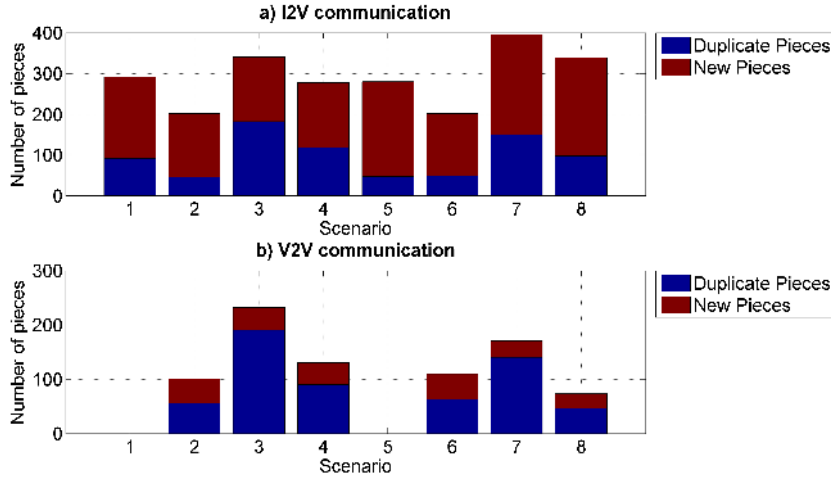


Figure 7.8: Effect of Erasure Coding on Received CRL Pieces

7.2.4 Using "Magic Cars"

A portion of the vehicles in the VC system can be equipped with cellular network support to connect directly to the CA and get all CRL pieces directly. We call these vehicles ‘magic cars’. Figure 7.9 shows the effect of using magic cars on convergence time. 30 percent of vehicles (3 out of 10 vehicles) are set to be magic in scenarios 5, 6, 7 and 8, and they request the CRL at a time which is chosen uniformly in the range of $[0,60]$.

Observation 1: RSU-only method does not take advantage of magic cars, because it only uses I2V communication for CRL distribution. Scenario 5 demonstrates a slight improvement in convergence time, but it is due to the 3 magic vehicles that do not need any CRL pieces, thus, the simulation stops in less time.

Observation 2: In contrast, the C2C Epidemic method shows a good improvement due to magic

Scenario 1: RSU-only, No Magic cars
 Scenario 2: C2C Epidemic, No Magic cars
 Scenario 3: MPB, No Magic cars
 Scenario 4: ICE, No Magic cars

Scenario 5: RSU-only, With Magic cars
 Scenario 6: C2C Epidemic, With Magic cars
 Scenario 7: MPB, With Magic cars
 Scenario 8: ICE, With Magic cars

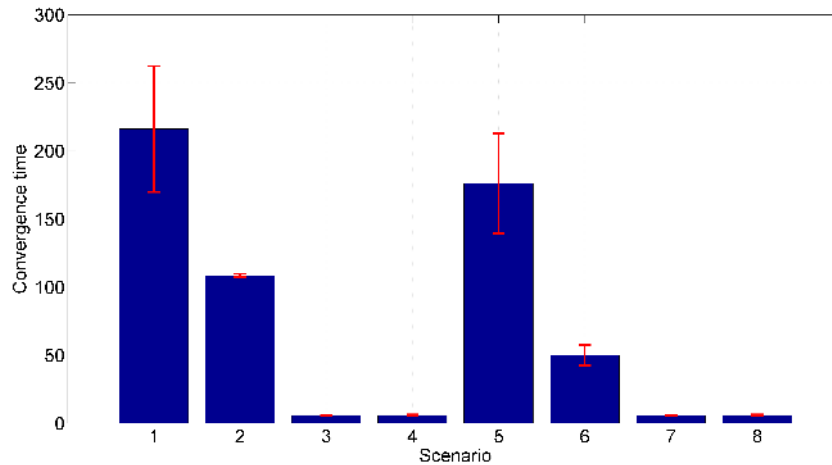


Figure 7.9: Convergence Time in Each Scenario When Using Magic Cars

Scenario	1	5	2	6	3	7	4	8
Average convergence time	216	176	108	49.8	5.79	5.74	5.88	5.9

Table 7.5: Result in Figure 7.9

cars. Magic cars increase the number of V2V communications and speed up the CRL distribution. This is due to the fact that magic cars always have the full set of CRL and can help other vehicles to receive the missing pieces. MPB and ICE do not take advantage of magic cars, because the simulation stops before appearance of magic cars in the network.

7.2.5 Number of Broadcasts by RSUs

Figure 7.10 illustrates the number of broadcasts by all RSUs during the simulation. Four scenarios are defined as following. These scenarios correspond to the simplest case in which shuffling and erasure coding is off, but magic cars are present in the network.

Scenario 1: RSU-only, No shuffling, No erasure, with magic cars

Scenario 2: C2C Epidemic, No shuffling, No erasure, with magic cars

Scenario 3: MPB (Most Pieces Broadcast), No shuffling, No erasure, with magic cars

Scenario 4: ICE (Intelligent CRL Exchange), No shuffling, No erasure, with magic cars

Observation 1: The number of broadcasts in RSU-only and C2C Epidemic is lower than the other two. This is because the RSUs always broadcast CRL pieces periodically. In contrast, the interaction of vehicles and RSUs is much higher in MPB or ICE. The vehicles request CRL pieces from RSUs

when they enter into their radio range.

Observation 2: The number of broadcasts in C2C Epidemic is lower than RSU-only, because the simulation is shorter (it uses V2V communication for CRL distribution and it takes less time for all vehicles to receive all pieces, hence the simulation time is lower).

Observation 3: average number of broadcasts by RSUs in ICE method is lower than MPB. This is due to the fact that ICE eliminates unnecessary broadcasts in RSUs. If no vehicles in the radio range of an RSU do not need any pieces, the RSU will cancel the broadcasting.

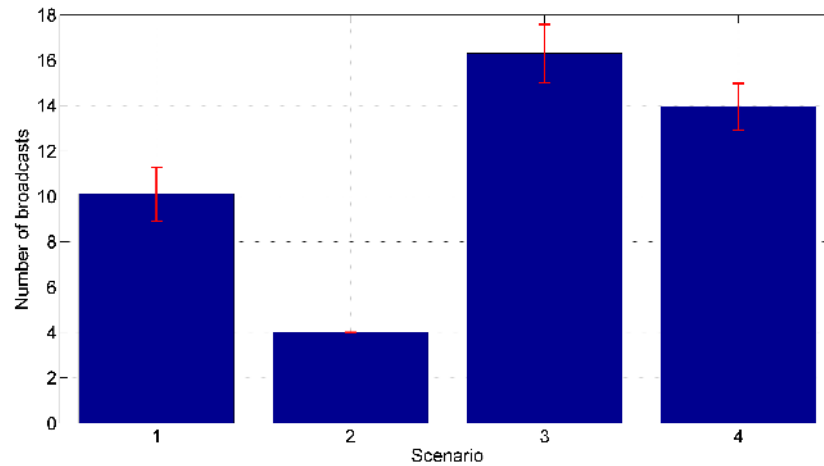


Figure 7.10: Number of Broadcasts in All RSUs

Scenario	1	2	3	4
Average # of broadcasts by RSUs	10.1	4	16.3	13.9

Table 7.6: Result in Figure 7.10

7.3 SUMO Mobility

SUMO was used to generate the mobility of vehicles which resembles the movement of vehicles in real life. By this, the simulation results tend to be more realistic. We performed a large scale evaluation consisting of 84 vehicles.

The configuration of SUMO was presented in section 6.8.2 and 6.8.3. Ultimately, the movement of vehicles in the network is reflected into the OMNET++ and the simulation is done like before. The simulation is executed until all the vehicles leave the network. The common simulation parameters are enumerated in table 6.2. We will present the simulation results here.

7.3.1 Comparing CRL Distribution Methods

We compared different CRL distribution methods in their simplest case possible, i.e., with shuffling disabled, no erasure coding and without any "magic cars" in the network. The CRL is divided into 20 pieces and each vehicle must receive all 20 pieces to be able to reconstruct the original CRL. Each scenario corresponds to a specific method as listed below:

Scenario 1: RSU-only

Scenario 2: C2C Epidemic

Scenario 3: MPB (Most Pieces Broadcast)

Scenario 4: ICE (Intelligent CRL Exchange)

Number of Vehicles that Received the Full CRL Pieces

Figure 7.11 shows the number of vehicles that have received all the CRL pieces over time in each scenario. As can be seen the number of vehicles with full CRL pieces increases gradually over time, and eventually all vehicles receive all 20 pieces. MPB and ICE follow each other closely. They both show a better performance than RSU-only and C2C Epidemic, and vehicles in the network receive all the required CRL pieces in less time.

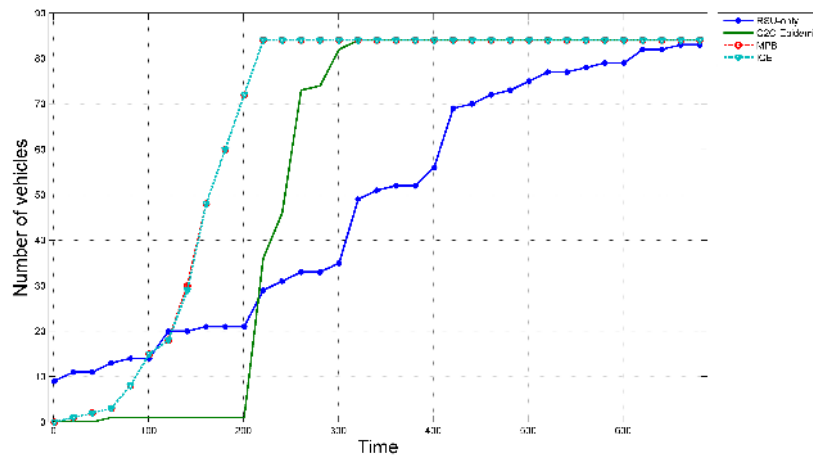


Figure 7.11: Number of Vehicles that Have Received All the CRL Pieces Over Time

Number of Received CRL Pieces

Figure 7.12 demonstrates the number of received CRL pieces from different sources. In MPB (scenario 3), the number of duplicate received pieces in V2V communication is high which corroborates the results in small-scale simulation. In contrast, this number is low in ICE (scenario 4) due to the semi-incremental CRL exchange.

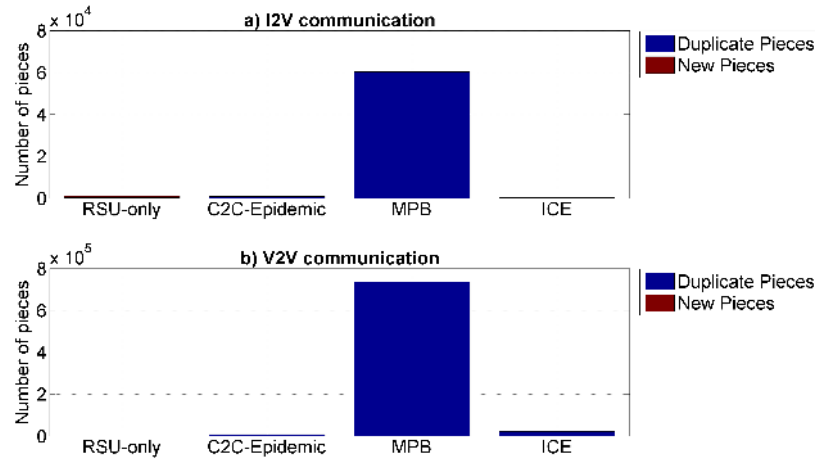


Figure 7.12: Number of Received CRL Pieces in I2V and V2V Communication

		RSU-only	C2C Epidemic	MPB	ICE
Figure a (I2V)	New	866	261	86.7	88.5
	Duplicate	147	722	60500	49.3
Figure b (V2V)	New	0	1400	1590	1590
	Duplicate	0	56020	734000	21800

Table 7.7: Result in Figure 7.12

Frames Received with Error

Figure 7.13 illustrates the average number of frames received with error for each of the scenarios with its respective confidence interval.

In the C2C Epidemic (scenario 2), the number of frames received with error is relatively high. This number is even higher in MPB which is due to the high number of unnecessary broadcasts in the vehicles. Number of errors shows a significant reduction in the ICE method (scenario 4).

Scenario	RSU-only	C2C Epidemic	MPB	ICE
Average # of frames with error	63.9	10300	68200	643

Table 7.8: Result in Figure 7.13

Number of Broadcasts by RSUs

Figure 7.14 shows the number of broadcasts by all RSUs during the simulation. Average number of broadcasts by RSUs in ICE method is significantly lower than MPB, because ICE eliminates unnecessary broadcasts in RSUs. If no vehicles within range of an RSU need any pieces, the RSU will cancel the broadcasting.

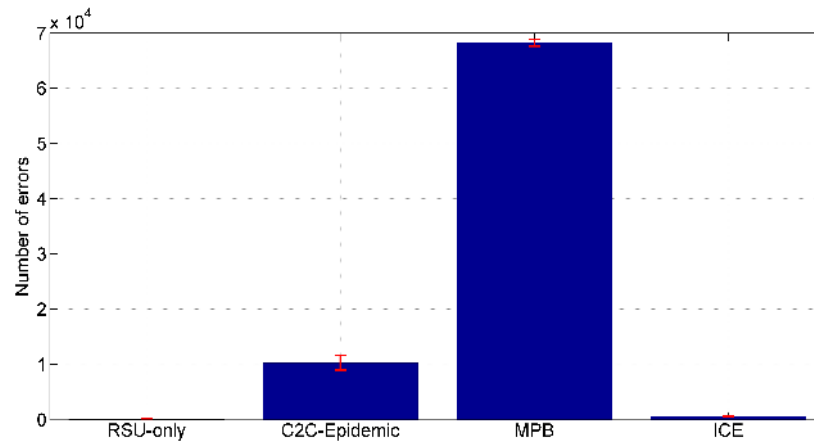


Figure 7.13: Number of Total Frames Received with Error in Each Scenario

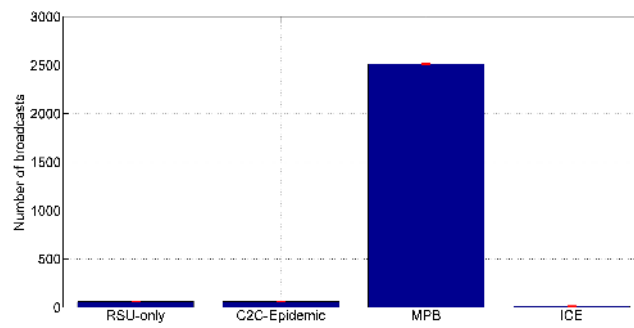


Figure 7.14: Number of Broadcasts in All RSUs

Scenario	RSU-only	C2C Epidemic	MPB	ICE
Average # of broadcasts by RSUs	60	60	2510	10.9

Table 7.9: Result in Figure 7.14

Chapter 8

Conclusion and Future Works

The purpose of this thesis was to implement the current CRL distribution methods in VANETs and compare their operations in small and large scale scenarios.

We considered the RSU-only, C2C Epidemic, and MPB methods. The C2C Epidemic and the MPB outperform the RSU-only method in terms of the time it takes for vehicles to receive all the required CRL pieces. This is due to the fact that they both use V2V communication in addition to I2V communication for receiving the pieces.

For MPB, the time it takes for vehicles to receive all the required pieces is less than that for C2C Epidemic. On the other hand, the number of duplicate received pieces is high for MPB. This is because the node with the highest number of pieces (say D) broadcasts blindly and it does not take the pieces of nearby vehicles into consideration. Vehicle D may broadcast lots of pieces that the nearby vehicles have already received. Broadcasting of unnecessary pieces increases the chance of collision in the crowded scenarios as well.

For ICE, the newly proposed method for CRL distribution, tries to alleviate the aforementioned problem by incorporating the semi-incremental CRL exchange. Simulation results show a significant reduction in the number of duplicate received pieces as well as the number of frames received with errors. Furthermore, RSUs broadcast CRL pieces only when the vehicles are in need. By this approach we prevent RSUs from unnecessary broadcasts, and the simulation results corroborate this clearly.

Although the time it takes for vehicles to receive all the required pieces in ICE is close to that for MPB, we find a significant improvement in the number of duplicate receive pieces, number of frames received in error, and also the number of broadcasts by RSUs in the network. The reduction in duplicate received pieces leads to lower channel load in the whole VANET.

The power of ICE is in using a low-overhead beacon message. It only uses two extra fields (start and end index) showing the range that encompasses all the missing pieces (MPB uses one extra field in beacon messages which shows number of pieces). The size of each field (start index or end index) is affected by the maximum number of pieces as discussed in section 5.5.

We also considered the effect of shuffling, using an erasure code, as well as using special-purpose

vehicles that have cellular connectivity and can get the full CRL directly from CA. The use of erasure code in CRL distribution decreases the probability of reception of duplicate pieces. Introducing special-purpose cars in the network decreases the convergence time, as they assist V2V communication.

Future Works

1. The nodes in the network do not have any transport layer, and they use a rudimentary protocol at the network layer. Future work can be integrating UDP and IP protocols into the project. Doing this does not have much effect on the simulation results, but makes it more realistic.
2. We used the IEEE 802.11g protocol in data-link layer due to the fact that it is tested extensively. Future work can use the IEEE 802.11p protocol. The Veins framework has this protocol implemented, but it is at an experimental stage; thus, we decided to not use it (although 802.11p and 802.11g are very similar in terms of operation).
3. Working more deeply with the OpenSSL library and the actual signing and verifying. Again this does not have any effect on the simulation results, but makes the experimentation more realistic. Moreover it can introduce explicitly computation overhead (cpu processing).
4. Integrating with other projects e.g., to obtain Pseudonyms. Each vehicle in the simulation can connect to the CA, and send the required commands.
5. Adding more features to the ICE protocol, such as multi-channel support. Until now, all message communication (beacons and CRL pieces) was done in a single channel (channel number 1 of IEEE 802.11g protocol). IEEE 802.11g provides 14 different channels, and we can exploit this to decrease the number of collisions especially in heavily crowded areas.

For example, beacon messages can be sent in channel number 1, and the CRL exchange between two or more nodes can be done in a separate channel between 2 and 14. Two or more vehicles that want to exchange CRL pieces should switch to a same channel. This channel is negotiated between the corresponding nodes, and all the nodes will be aware of the channel.

Appendix A

Calculation of Interference Distance

The connectionManager module is responsible for managing connections between different nodes in the network and also drawing the coverage circle around RSUs. It uses three parameters for calculating the radius of the coverage circle which are:

- carrierFrequency: minimum carrier frequency of the channel in Hz.
- pMax: maximum sending power used for this network in mW.
- sat: minimum signal attenuation threshold in dBm.
- alpha: minimum path loss coefficient.

$$\text{Radios of the circle is interfDistance} = \left(\frac{\text{waveLength}^2 \times \text{pMax}}{16 \times \pi^2 \times \text{minReceiverPower}} \right)^{\frac{1}{\text{alpha}}}$$

$$\text{in which the waveLength} = \frac{\text{speedOfLight}}{\text{carrierFrequency}}, \text{ and } \text{minReceiverPower} = 10^{\frac{\text{sat}}{10}}$$

(minReceiverPower is the minimum power level to be able to physically receive a signal).

With initial values for variables above, we can calculate the interfDistance.

carrierFrequency = 2.4 GHz
pMax = 110.11 mW
sat = -120 dbm
alpha = 4.0

And with simple calculation we have:

WaveLength = 0.124914
minReceiverPower = 10^{-12}
InterfDistance = **322.966 m**

Appendix B

Simulation Platform

The network simulation is done using OMNET++ version 4.2.2, Veins framework version 2.0 and MiXiM framework 2.2. An Intel Xeon (dual-core, 3.4 GHz) machine equipped with 8 GB RAM, running a 64-bit version of Linux Ubuntu 10.04 Server is used for the simulations. The simulation normally consists of hundreds of runs, and we execute two runs in parallel to exploit the dual-core CPU.

Installed Packages

The following packages were installed on the Ubuntu:

BonnMotion 2.0

The source code was downloaded from its official website and the "install" script was executed after setting the Java path.

Sumo 0.15.0

sumo	Installs sumo applications (sumo, sumo-gui, netconvert, etc)
sumo-tools	Installs sumo tools including the Python scripts
sumo-doc	Installs sumo documents including the Sumo tutorial and sample projects

Boost 1.46

Boost is a set of libraries that extend the functionality of the C++ programming language. We use the serialization library to convert C++ data structures into sequence of bytes, to be able to transmit them through the network. At the receiver we reconstruct the original data structure.

libboost1.46-dev	Installs the headers in usr/include/boost
libboost-serialization1.46-dev	Installs the serialization libraries in usr/lib

Eigen 3.0

Eigen is a C++ template library for linear algebra: vectors, matrices, and related algorithms. The source code was downloaded from its official website and the headers were used without any build (Eigen uses header-only libraries).

OpenSSL 1.0.1

OpenSSL is an open-source implementation of the SSL and TLS protocols. Basic cryptographic functions are also implemented in the core library and we are using them to sign the messages.

`libssl-dev` Installs headers in `/usr/include/openssl`, and libraries in `/usr/lib/i386-linux-gnu`

The Eclipse IDE was configured appropriately:

Project Properties Linker -> Libraries -> add option -> other option -> adding `-lcrypto` and `-lssl` separately.

Linux Bash Script

First we use the SSH protocol to connect remotely to the server. Then we execute a bash script which handles all the work.

To keep the terminal session we used the following command, and then detached the session (by pressing CTRL-A and then CTRL-D). By this we keep the simulation running even after closing the terminal window or breaking the SSH session. In the following command, `session1` is the session name and `scriptKTHServer.sh` is the name of the file that contains our bash script.

```
screen -S session1 -f ./scriptKTHServer.sh
```

To resume the session we issue the following command:

```
screen -r session1
```

The pseudo-code for bash script is shown in Algorithm B.1. First we check whether the OMNET++ is installed or not. Then We check if the bin directory of OMNT++ is included in the `$PATH`, because we need to use `opp_run` and `opp_runall` programs which reside in bin folder. Now we compile and link the whole project using `make`.

At the next step, we check if SUMO is installed or not, and run a Python script afterwards. This Python script starts a TCP server process for TraCI that listens to a specific port, waiting for a client process. Eventually we start the simulation by using the `opp_runall` script. Two runs are executed in parallel using `opp_runall`.

Algorithm B.1 Linux Bash script for running the simulation in remote server

```
if OMNET++ is not installed then
    echo "OMNET++ can not be found!"
    exit with error code -1
else
    echo "OMNET++ is installed."
end if

if OMNET++ bin folder is in $PATH then
    echo "OMNET++ bin folder is already in the path"
else
    export PATH=$PATH:/home/maniam/omnetpp-4.2.2/bin
end if
```

building the omnet++ project using make

```
if SUMO is not installed then
    echo "SUMO is not installed!"
else
    export "SUMO is installed."
    exit with error code -1
end if
```

Start a TCP server process for TraCI (by launching a python script in background).

Start the simulation (two runs are executed in parallel using opp_runall)

Bibliography

- [1] P. Papadimitratos, G. Mezzour, and J. Hubaux, “Certificate revocation list distribution in vehicular communication systems,” in *Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking*, pp. 86–87, ACM, 2008.
- [2] K. Laberteaux, J. Haas, and Y. Hu, “Security certificate revocation list distribution for vanet,” in *Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking*, pp. 88–89, ACM, 2008.
- [3] M. Nowatkowski and H. Owen, “Certificate revocation list distribution in vanets using most pieces broadcast,” in *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, pp. 238–241, IEEE, 2010.
- [4] A. Varga and R. Hornig, “An overview of the omnet++ simulation environment,” in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, p. 60, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.
- [5] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, “Sumo-simulation of urban mobility—an overview,” in *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pp. 63–68, 2011.
- [6] U. D. of Commerce, “Motor vehicle accidents—number and deaths,” tech. rep., United States Census Bureau, 2012.
- [7] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, “Bonnmotion: a mobility scenario generation and analysis tool,” in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, p. 51, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010.
- [8] P. Papadimitratos, A. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, “Vehicular communication systems: enabling technologies, applications, and future outlook on intelligent transportation,” *Communications Magazine, IEEE*, vol. 47, no. 11, pp. 84–95, 2009.
- [9] H. Hartenstein and K. Laberteaux, “A tutorial survey on vehicular ad hoc networks,” *Communications Magazine, IEEE*, vol. 46, no. 6, pp. 164–171, 2008.
- [10] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, and J. Hubaux, “Secure vehicular communication systems: design and architecture,” *Communications Magazine, IEEE*, vol. 46, no. 11, pp. 100–109, 2008.

- [11] F. Kargl, P. Papadimitratos, L. Buttyan, M. Muter, E. Schoch, B. Wiedersheim, T. Thong, G. Calandriello, A. Held, A. Kung, *et al.*, “Secure vehicular communication systems: implementation, performance, and research challenges,” *Communications Magazine, IEEE*, vol. 46, no. 11, pp. 110–118, 2008.
- [12] Z. Ma, F. Kargl, and M. Weber, “Pseudonym-on-demand: a new pseudonym refill strategy for vehicular communications,” in *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, pp. 1–5, IEEE, 2008.
- [13] W. Stallings, *Network Security Essentials: Applications and Standards*. Prentice Hall, 2010.
- [14] A. Arnes, “Public key certificate revocation schemes,” Master’s thesis, Norwegian University of Science and Technology, February 2000.
- [15] G. E. Nigusse, “Evaluating public key certificate revocation schemes,” Master’s thesis, Royal Institute of Technology, August 2007.
- [16] “Trial-use standard for wireless access in vehicular environments (wave),” tech. rep., IEEE, 2007.
- [17] M. Raya, P. Papadimitratos, I. Aad, D. Jungels, and J. Hubaux, “Eviction of misbehaving and faulty nodes in vehicular networks,” *Selected Areas in Communications, IEEE Journal on*, vol. 25, no. 8, pp. 1557–1568, 2007.
- [18] M. Nowatowski, “Certificate revocation list distribution in vehicular ad hoc networks,” 2010.
- [19] M. Rabin, “Efficient dispersal of information for security, load balancing, and fault tolerance,” *Journal of the ACM (JACM)*, vol. 36, no. 2, pp. 335–348, 1989.
- [20] P. Papadimitratos and Z. Haas, “Secure message transmission in mobile ad hoc networks,” *Ad Hoc Networks*, vol. 1, no. 1, pp. 193–209, 2003.
- [21] B. Forouzan, C. Coombs, and S. Fegan, *Data communications and networking*, vol. 4. McGraw-Hill, 2001.
- [22] OMNET++, “Simulation Models.” <http://www.omnetpp.org/models/>, 2008. [Online; accessed 15-June-2012].
- [23] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. Haneveld, T. Parker, O. Visser, H. Lichte, and S. Valentin, “Simulating wireless and mobile networks in omnet++ the mixim vision,” in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, p. 71, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.
- [24] T. Camp, J. Boleng, and V. Davies, “A survey of mobility models for ad hoc network research,” *Wireless communications and mobile computing*, vol. 2, no. 5, pp. 483–502, 2002.
- [25] M. Piorkowski, M. Raya, A. Lugo, P. Papadimitratos, M. Grossglauser, and J. Hubaux, “Trans: realistic joint traffic and network simulator for vanets,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 12, no. 1, pp. 31–33, 2008.

- [26] R. Bauza, J. Hernandez, J. Gozalvez, S. Vaz, D. Valiente, I. Aguado, and A. Gonzalez, “itetriss heterogeneous wireless communication platform for the large-scale evaluation of cooperative road traffic management policies,” -, 2009.
- [27] C. Sommer, R. German, and F. Dressler, “Bidirectionally coupled network and road traffic simulation for improved ivc analysis,” *Mobile Computing, IEEE Transactions on*, vol. 10, no. 1, pp. 3–15, 2011.