RTP compatible: Two models of video streaming over VANETs

by

Zhifei Fang

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Computer science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Because Vehicular Ad Hoc Networks (VANETs) often have a high packet loss rate, the old protocol for video streaming like Real-time Transport Protocol (RTP) is no longer fitting for this specific environment. Previous research has offered many new protocols to solve this problem, however most of them cannot make full use of the existing Internet video streaming resources like RTP servers.

Our works propose two models to solve this compatibility issue. The first model we call it converter model. Based on this model, we first modify RTP using Erasure Coding (EC) technique in order to adapt it to high packet loss rate of VANETs. This new protocol we called EC-RTP. And then, we develop two converters. The first converter stands on the boundary between Internet and VANETs. It receives the RTP packets which sent from Internet. And then it translates them to the EC-RTP packets. These packets are transported over the VANETs. The second converter receives these EC-RTP packets, translates them back to the RTP packets. And then it sends them to the RTP player. So that the RTP player can play these packets. To make EC-RTP can carry more kinds of video stream other than RTP, we propose the second model. The second model we call it redundancy tunnel. Based on this model, we let the protocol between the two converters carry RTP protocol as its payload. We use the same technique as we have used to modify RTP. At last we do some experiments with Android tablets. Experiment results show our solution can use the same player to play the same video resources as RTP does. However, it can reduce packet loss rate other than RTP.

Acknowledgements

I would like to thank all the little people who made this possible.

My deepest gratitude goes first and foremost to Professor Azzedine Boukerche, my supervisor, for his constant encouragement and guidance. He has walked me through all the stages of the writing of this thesis. Maybe a lot of people think he is a tough guy. However, without his consistent and illuminating instruction, and his push, this thesis could not have reached its present form.

Second, I would like to express my heartfelt gratitude to Dr.Cristiano Rezende, who led me into the world of video streaming. And Dr.Abdelhamid Mammeri,who helps me to finish my experiments and edit this thesis.

Third,I feel grateful to my lab colleagues. We often argue some topic during the lunch. The more the truth is debated, the clearer it becomes.

Last my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years. I also owe my sincere gratitude to my friends and my fellow classmates who gave me their help and time in listening to me and helping me work out my problems during the difficult course of the thesis.

Dedication

This is dedicated to the one I love.

My girlfriend Jingwen Feng, she made coffee for me when I was tired.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Vehicular Ad Hoc Networks (VANETs) often have a high packet loss rate. The traditional Real-time Transport Protocol (RTP) have no features to handle packets loss. Actually if there is high packet loss rate, the receiver even cannot play the video stream. However, the RTP is widely used on the traditional IP network, there are a lot of solutions used RTP as their transport protocol to provide video streaming services. Nowadays, many researchers offer many new protocols to solve the packet loss issue, but most of them are lack the capacity to communicate with the RTP protocol. Therefore, we cannot deploy RTP directly to the VANETs, making the VANETs even more expensive.

In our work, we aim to modify the RTP so that it can be used in the VANETs environment. In addition our solution can also communicate with the RTP server at the same time. To resist to the packets loss, redundancy is one of several possible strategies. According to [2], we decide to use the Ensuring Coding (EC) technique to develop our new version of RTP, which will be called the Ensuring Coding Real-time Transport Protocol (EC-RTP). To let the EC-RTP easily communicate with RTP, we design a EC-RTP to RTP converter. In addition, to make our converter can support more protocols other than RTP, we designed EC Generic Routing Encapsulation (EC-GRE).

The contributions in this work are divided in mainly 4 parts: i) Conduct a study on

VANET's topology and some use cases. And then we formulate our problem based on the study. Besides, we give out 2 model to solve this problem after we formulating it. ii) Design a EC model, based on it, we design our EC-RTP and EC-GRE. iii) Build a demonstration based on Android system. It can support EC-GRE, EC-RTP, RTP and EC-RTP to RTP converter. iv) Conduct experiments based on this demonstration to verify the effectiveness of the EC-RTP and EC-GRE. Comparing with RTP, we verify the redundancy if it improves the quality of video streaming.

Chapter 2 will show a survey about the video streaming packet recover techniques. Chapter 3 will outline the basic VANETs topology, based on the topology we will find out where the packet loss happens, determine the boundaries of the VANETs protocol and the Internet protocol, and propose 2 models that can easily increase the VANETs' compatibility with the Internet system. Chapter 4 shows the design of the EC-RTP and EC-RTP to RTP tunnel. Chapter 5 shows a demonstration based on the Android system, with the model we talked about in Chapter 3 and protocols we talked about in Chapter 4. Chapter 6 describes some experiments. Comparing with the original RTP, the experiment can verify if our EC-RTP can provide a better quality video. Our conclusions and future work are presented in Chapter 7.

Chapter 2

Related work

VANETs is a very lossy network, since its topology changes very frequently. Therefore, we need to find an effective way to decrease the packet loss rate. In this chapter, we will introduce some techniques people use to reduce packet loss rate. Therefore we can find a adapted way for our solution.

At the beginning, people design the IP network for transmitting files other than some real-time media streaming. Since the transmitting files do not have any real-time requirement, the IP network allowed delay and packets lost. What's more, because VANETs is a lossy network, the high packets loss rate may cause poor video quality. Transmission Control Protocol (TCP) designed to solve this packets loss problem; however,because TCP waits every packet's feedback, for the real-time video streaming, TCP may cause an unacceptable delay. That's the reason why people decide to choose User Datagram Protocol (UDP) for real-time video streaming. However, UDP is not a reliable protocol. There is no guarantee of reliability or ordering of packets, they may arrive out of order, be duplicated, or not arrive at all. When the network jams, the UDP may also drop some packets. Nowadays, people have specifically performed research to solve this packet loss problem.

Packet loss compensation technology can be classified into two generic groups; compensation based on the sender, and compensation based on the receiver. The technology from

the sender includes redundancy, interleaving, key frame retransmission and Dynamic Rate Adjustment. The technology from receiver includes multiple error concealment methods. Besides, there are still some other techniques for video streaming.

Sender	**Receiver**
• Redundancy. • Interleaving • Key frame retransmission • Dynamic Rate Adjustment	• Error concealment.

Table 2.1: Methods of Packet loss compensation

2.1 Redundancy

Redundancy is the most promising current technology to improve the quality of video streaming for now. The main idea is that it not only send out the main information, but also some redundancy information. When some packets lost, these redundancy information can recover these packets back. The advantage of this technology is that we no longer care about what kind of media we are now transmitting. The disadvantage, which is also very obvious, is that sending the redundancy information can increase bandwidth cost. And the processing of generating redundancy information can cause transmitting delay.

2.1.1 Forward Error Correction (FEC)

IEEE's RFC 5109 [3] gives out a FEC method for protecting against packet loss over packet-switched networks. This is a method designed for RTP. However, not all the player

can support this method. What's more, this method is not very flexible. It treats all packets with same priority. And It doesn't have enough ability to handle high packets loss environment. Figure 2.1 shows an example of FEC.

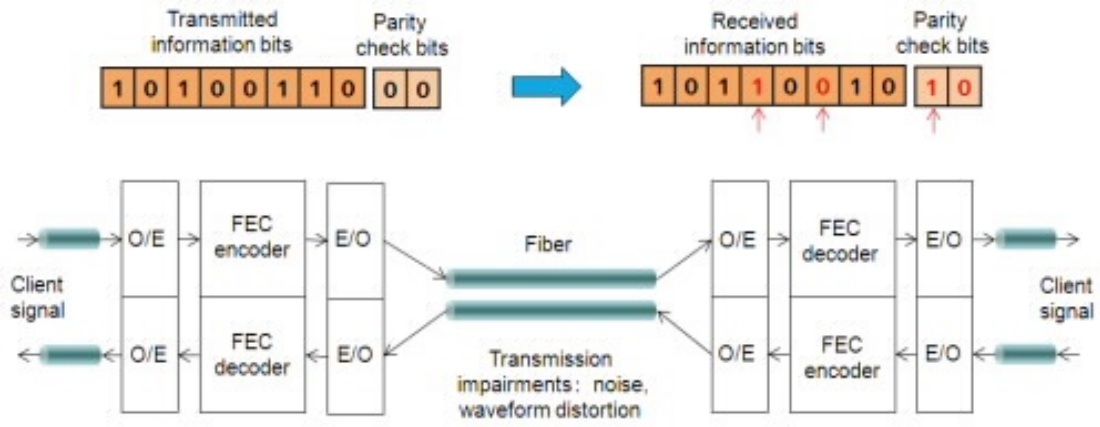

Figure 2.1: An example of FEC

Pasin et al.[4] suggest a packet level Forward Error Correction and interleaving algorithm. Their experiment shows their algorithm decrease the packet loss rate. It shows that, a feedback mechanism may be a good choice to decrease the redundancy's bandwidth consummation.

2.1.2 Erasure coding

Erasure coding is one of the FEC method. The idea behind Erasure Coding(EC) is that the loss of a part of transmitted data does not influence the ability of receivers to obtain the original content. In EC, the original packets are encoded into a bigger amount of packet at the sender side; and, intermediary nodes may decode received packets for their own benefit (i.e. in the case of video dissemination) relaying exactly the same received packets. Figure 2.2 shows an example of erasure coding.

Although Erasure coding is first designed for the storage system, it wildly used in open

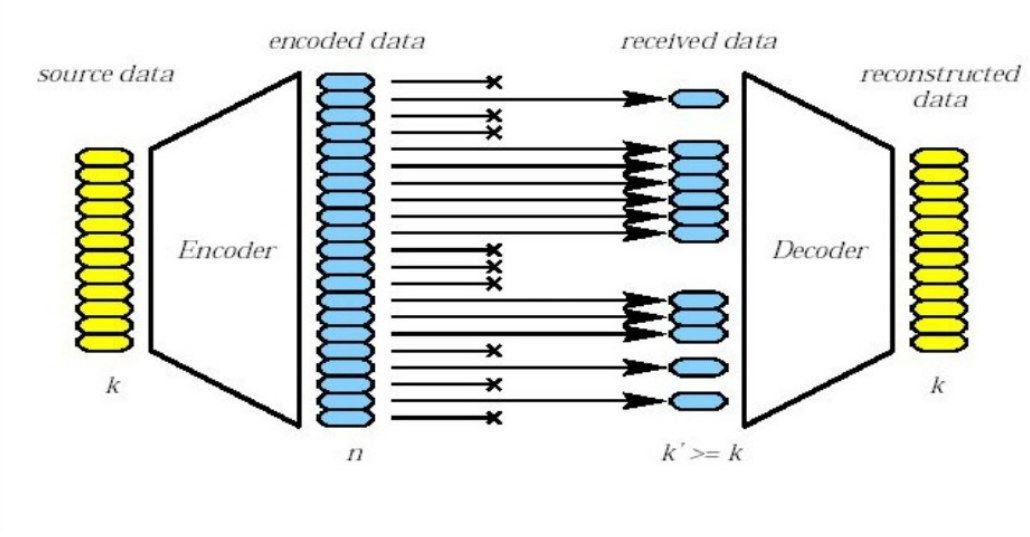

Figure 2.2: An example of erasure coding

source software, cloud system [5][6][7][8][9], now a lot of research show it can eliminate the bad effort of packet[10]. There are kinds of EC techniques, but few of them can using in the real-time video streaming. Because the storage system don't have the real-time demands.

Among them, XOR-based-coding is one worthy of consideration [11][12].There are a lot of XOR based coding techniques, for example, Krohn et al.[13] outlines an erasure coding technique for efficient multicast transfers, Luby[14] [15] introduced a rateless erasure coding technique that are very efficient as the data length grows.

Random Linear Coding [16] (RLC) is another, which has been created to optimize the distribution of information in a multicast scenario but it can be adapted to add redundancy in a network in order to handle packet loss.

Rezende et al.[10] have observed that XOR-based Coding has achieved higher delivery ratios than RLC at similar amounts of additional redundancy.

Sardari et al., [18] and [17] suggest using Erasure Coding for dissemination of multimedia content. Their work takes into consideration the scenario in which roadside infrastructure is available to assist the dissemination process.

2.1.3 Network coding

Network Coding (NC) [19] in VANETs is aimed specifically at making more efficient use of the shared medium by requesting that intermediary nodes transmit newly encoded packets Through this manner, transmissions observed by neighbouring nodes are always valuable towards gathering the original content transmitted, instead of gathering simply duplicates. In NC, intermediary nodes wait for the reception of enough packets before decoding a block, and re-encoding the original content into newly encoded packets which are then transmitted further. NC techniques have to offer great diversity in the packets newly encoded by intermediary nodes and to make sure that the reception of such packets is relevant to gathering the original information at receivers.

Figure 2.3 shows a classical example of network coding.

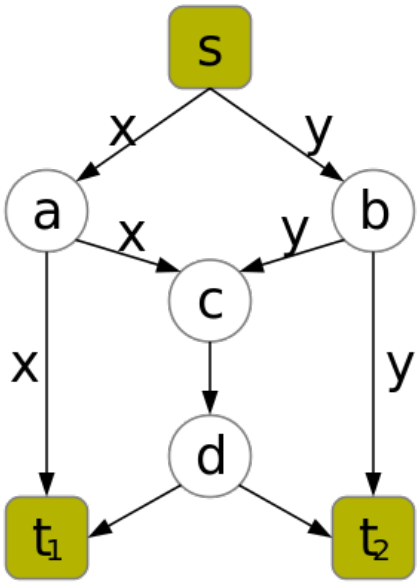

Figure 2.3: An example of network coding

Lee et al.[20] analysis of implementation issues of network coding in vehicular networks , their results show that network coding parameters must be carefully configured by taking resource constraints into account. This paper give us a hint, that NC will consume a lot

of resource in the network.

Lee et al.[21] propose a network coding based file swarming protocol targeting vehicular ad hoc net-works (VANET).

Ahmed and Kanhere[22] purpose a novel network coding based co-operative content distribution scheme called VANETCODE. The randomization introduced by the coding scheme makes distribution efficient.

2.1.4 Selective of different coding technology

Zhang et al.[23] propose a solution that combines both Erasure and Network Coding using Random Linear Coding for lossy network. In their work, the source node and the intermediary nodes send a number of encoded packets that is greater than the size of a block.They suggest that the redundant information should be used to handle packet loss while Network Coding to improve throughput efficiency.

Rezende et al. [2] outlines how to use different redundancy methods suitable for VANETs. Based on this paper, in our work, we try to apply erasure coding (EC) as our coding technology. Because our work is focus on end-to-end scenario[24].

2.2 Interleaving

When a lot of packets have been lost suddenly, the redundancy packets may not enough to recover all of the lost packets. To resist this situation, interleaving is one of the choices. The main idea behind interleaving is that, we divide our original data into small units, they are smaller than a packet. Before sending, reorder these units to let every packet have data from different frames. Then at the receiver side, make the unit to the original order. When a packet lost, since the units inside the packet come from different frames, we lost these units rather than lost a whole frame.

Actually, interleaving cannot recover the lost packets back. At the same time, the reordering progress consume a lot of time. That may cause an unacceptable delay for real-time video streaming. However, it can help to reduce the impact of packet loss. Figure 2.4 shows the processing of interleaving.

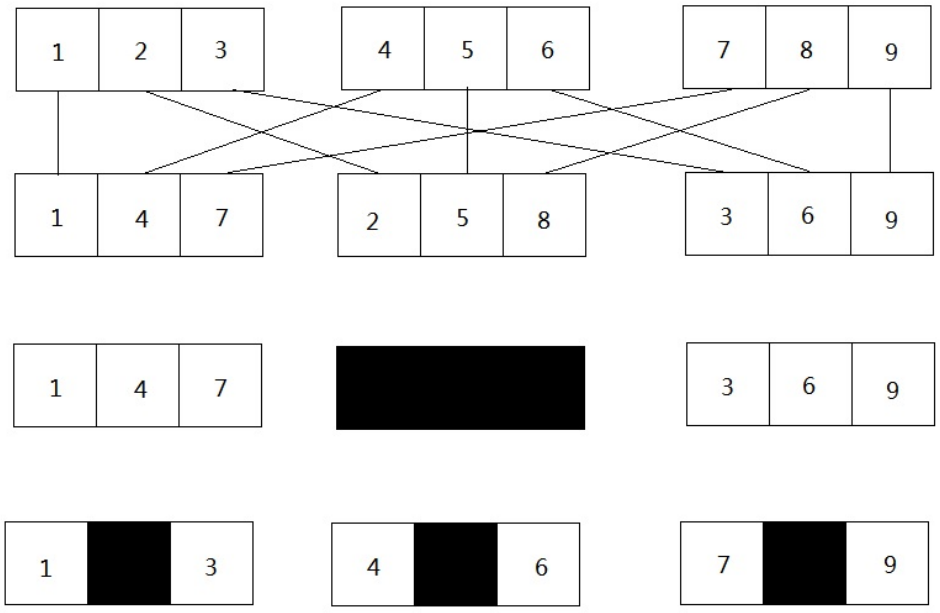

Figure 2.4: An example of interleaving

Lambert et al.[25] purposed test the new Flexible Macroblock Ordering(FMO) in H.264/AVC digital video compression standard. FMO is one kind of Interleaving. It splits a frame into some slices. Using FMO itself maybe not enough, so Qadri et al.[26][27] and Razzaq and Mehaoua[28] try to combine the this method with different technology such as we have mentioned above FEC, and the one we will introduce later Error concealment.

2.3 Key frame retransmission

According to the H.264/AVC standard, a video will divide into frames with different types. Among these frames, some plays the key role. They are the key frames, and other frames

only record the changes of the key frame. Therefore, the key frames' quality is the important factor of video quality. In other words, what we need to do is just retransmit these key frames, instead of retransmitting all frames. Therefore, bandwidth is saved. However, when we meet a network jam, retransmission is often failed. Figure 2.5 shows an example of frame retransmission.

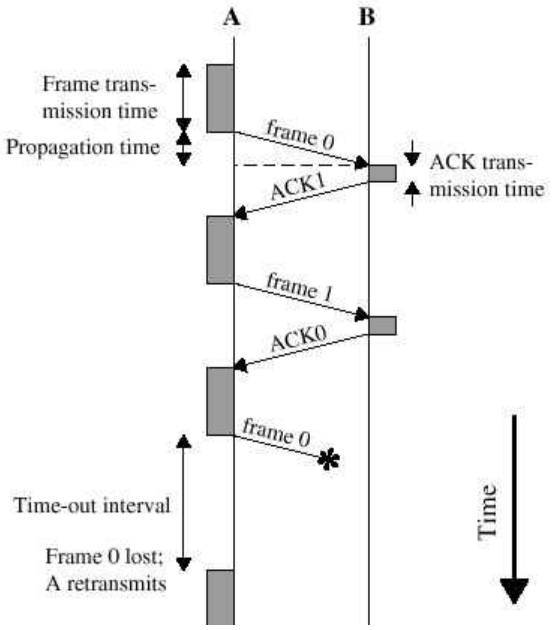

Figure 2.5: An example of frame retransmission

IEEE's RFC 3366[29] provides a frame retransmission technique called Automatic Repeat Request (ARQ). In the ARQ, when the receiver side receives data correctly, it must give feedback to the sender side. Otherwise, sender side stops sending. ARQ can make the transmission reliable. However, the retransmission delay makes ARQ doesn't adapt to real-time video system.

2.4 Dynamic Rate Adjustment

The main idea of this technique is to change the bit rate of a video stream dynamically, let the video bit rate adapt to the network bandwidth. For example, if the network is jammed,

and since a high bit rate video have bigger size than a low bit rate video, we don't have enough bandwidth to send out high bit rate video. At the same time, the receiver side often has a bad video playing experience. A proper way to avoid the bad experience is to decrease the video bit rate.

Xing and Cai [30] describe a technique using dynamic rate adjustment to VANETs. And Krasic et al. [31] describe a technique using priority data dropping. In this paper, the authors also show how to express adaptation policies and how to deal with priority-mapping.

2.5 Error concealment

Error concealment is a technique which is used by sender side. Here are some classical error concealments.

1. Insertion

   The receiver side reorders the received packets, because these packets may not in the right order. After this progress, there may be some empty positions, because the packets lost in those positions. To conceal these errors, a simple way is to insert zeros to these positions. Or, we can copy the previous packets which before the lost packets to fill these empty positions.

2. Interpolation

   The main idea behind Interpolation is that after the reorder progress, we use the packet near the lost packet to compute a similar packet to the lost packet. Then use this packet to replace the lost packet, we may reduce the bad effect that caused by packets lost . Figure 2.6 shows an example of frame interpolation. We can find 1a is generated by frame 1 and 2.

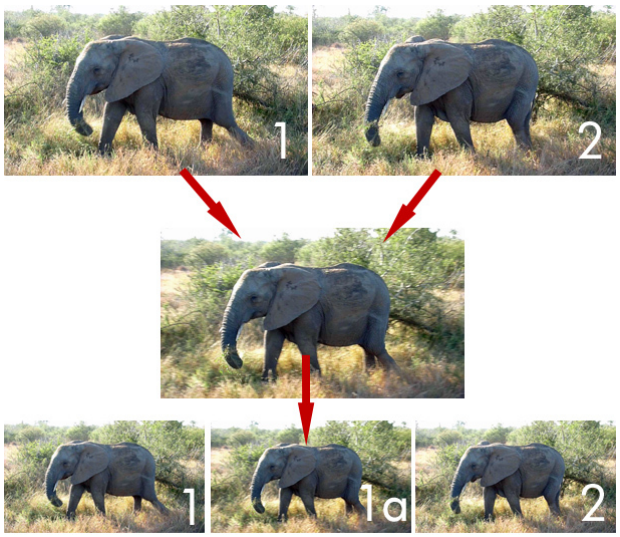

Figure 2.6: An example of frame Interpolation

The shortcoming of error concealment is that it is passive. Only when the packets lost, it starts to work. So it is often combined with other techniques. Ho et al.[32]intend to enhance video streaming quality at the receiver side by recovering lost packets with error resilience. In addition, choose more stable routing paths to make inter-vehicle data transmission more reliable.

2.6 Lower layer technology

The packet loss often occurs when the VANETs topology changed. Assume we have 3 vehicles A, B and C. They are running on a road. B is in the middle of A and C. Now A is leaving the communication range of B, C is going to enter that range. B want to send packets out, at first it connects to A. However, when the A leaves out the communication range, at the same time the new connection between B and C haven't ready yet; thus, the packets send out from B will be lost. Reducing the time of establishing new connection is one way to solve this problem.

IEEE's 802.11p [33] is a new protocol which is designed for VANETs. Eichler [34] evalu-

ate its performance. The result shows that 802.11p shall provide a multi-channel dedicated short-range communications solution with high performance for multiple application types.

Asefi et al.provide an adaptive Medium Access Control (MAC) retransmission limit selection scheme just like IEEE 802.11p. This technique can establish connections among nodes fast.

2.7 The Next Generation codec

Nowadays, the new emerging video coding standard High Efficiency Video Coding (HEVC) which is also known as H.265 is coming out. If this new coding standard has the ability to solve the packets loss issue, we can use this new coding standard as our standard.

However, Piol et al. [36] evaluate it and the result shows that even if the next generation coding standard is still very sensitive to packet loss.

Chapter 3

Problem formulation

During this chapters, we firstly study several possible video streaming scenarios to locate the compatibility issue. And then we develop our two models to solve the compatibility issue.

3.1 VANETs basic Topology

According to [1], the basic VANETs have two basic parts; one is the vehicle to vehicle network, the other one is vehicle to roadside infrastructure network. VANETs enable a vehicle to communicate with other vehicles which are out of sight or even out of radio transmission range. They also enable vehicles to communicate with roadside infrastructure. In addition, the roadside infrastructure lets the VANETs connect to the Internet.

3.2 Several Scenarios for video streaming

3.2.1 The user in a car plays the video from Internet

In this scenario, as Figure 3.2 shows, there is a car in the VANETs; inside the car, there is a small Local Area Network (LAN), one laptop and one cell phone connected to a vehicle-

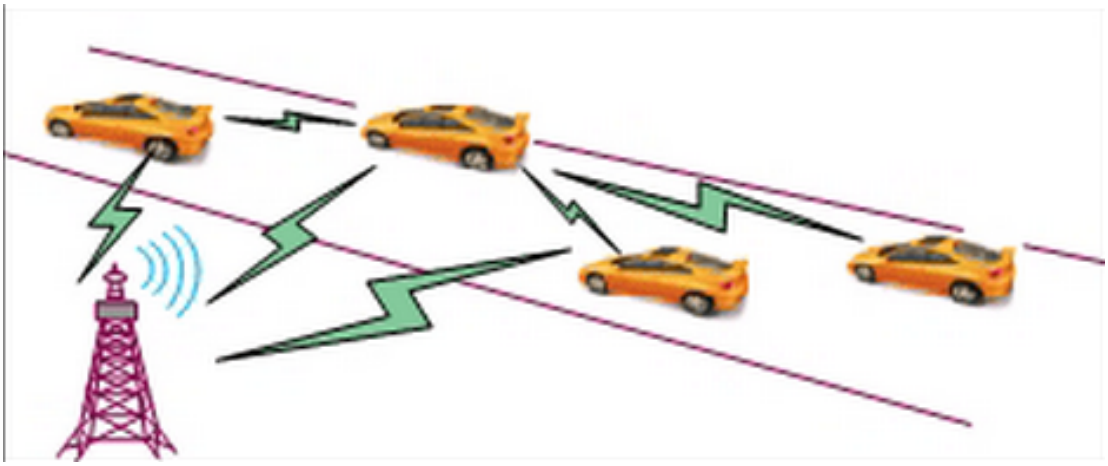

Figure 3.1: Basic VANETs topology[1]

mounted wireless router. The user of the laptop wants to see the video from the Internet; his request will first be sent to the vehicle-mounted wireless router. It will then forward to the VANETs, the VANETs will communicate with Internet by the roadside infrastructure. And the internet will retrieve the video from the video streaming server.

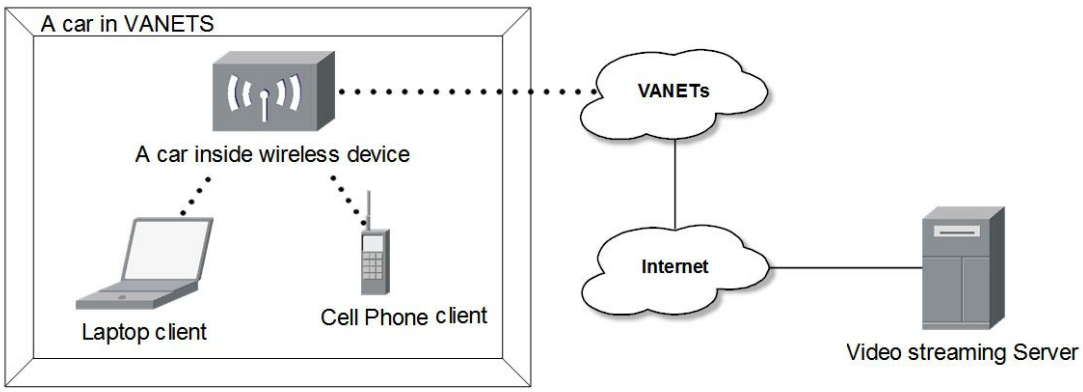

Figure 3.2: The user in a car plays the video from Internet

3.2.2 The user on the internet plays a real-time video captured from vehicle's camera

In this scenario, as Figure 3.2 shows, the user of the PC wants to see the video captured from the camera on Car 1. The stream will flow exactly like 3.2.1 but in the opposite direction.

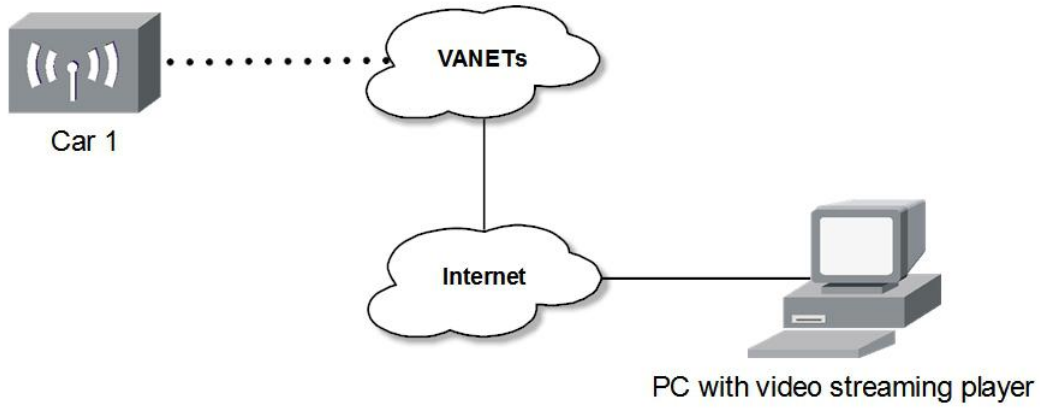

Figure 3.3: A user on the internet plays a real-time video captured from vehicle's camera

3.2.3 Car to Car video share

In this scenario, as Figure 3.4 shows, cell phone 1 and cell phone 2 want to share a video with each other. Because they are in two different cars, these cars are connected to each other through VANETs. In each car, there is a wireless router, which is connected to each cell phone. The router in the vehicle plays a gateway role to VANETs.

3.3 Packets loss

From the 3 scenarios, the main area of packet loss in the topology will occur in the VANETs. When the Internet protocols want to go through VANETs, the risk of packet loss will

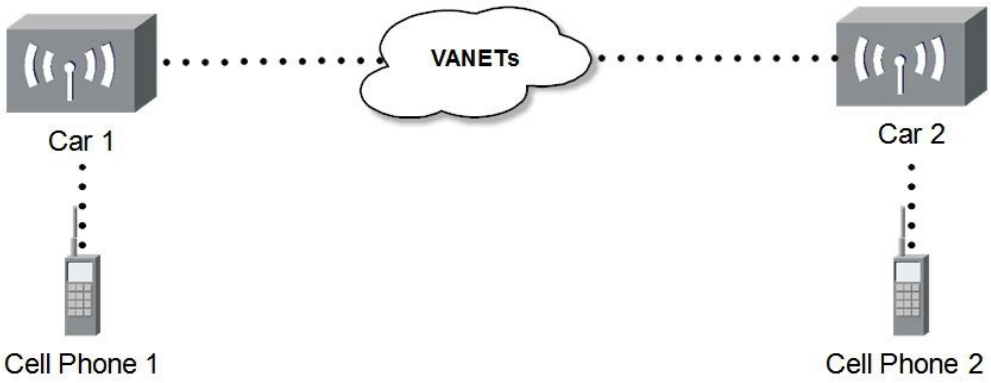

Figure 3.4: A car to car video sharing scenario

increase dramatically. The main reason is that: the Ad-hoc connection is easily interfered by environment, while the topology in the VANETs changed frequently; for example, in 3.2.1 the car connects to the roadside infrastructure, and when the car runs at a high speed, it will switch to different roadside infrastructures. This switching will cause a lot of packet loss.

3.4 Models

From the cases we have discussed above, we can draw a conclusion: when the Internet protocols go through the VANETs, they may meet the high packet loss rate. When this situation happens, a lot of these protocols cannot work properly, especially for those real-time video streaming protocols. Therefore, the user will obtain an unsuccessful experience. At last, our problem can be describe as below.

Assume we have 2 different networks, one is traditional network like Internet, the other one is new network like VANETs. They are both packet-switched network, but there are some differences with features like packet loss rate, delay, sending rate, etc. In the traditional network, there are some protocols used widely. However, there are no similar

protocols in the new network. In the mean time, because of different features, if we directly bring these widely used protocols from the traditional network to the new network, the user experience may be bad. The related research mainly focus on designing new protocols which can achieve same goal as these widely used protocols in the new network. However, because these newly designed protocols often have different working principle, they cannot communicate with these protocols running in the traditional networks, that cause resources wasting.

We have two goals to achieve. The first one is we use some techniques to improve the user experience with video streaming. At the same time, we want to keep use the same player and server as people used in Internet. In this work, we use the RTP player and server. According to Chapter 2, we decide to use EC as our technique to reduce packet loss rate. Since the VANETs is the network with high packet loss rate, we need firstly to determine VANETs and Internet boundary.

3.4.1 Internet boundary

As we discussed above, when the data wants to go towards the internet, they need to go through the roadside infrastructure first. Therefore, it is easy to give the boundary of the internet. Figure 3.5 shows this kind of boundary.

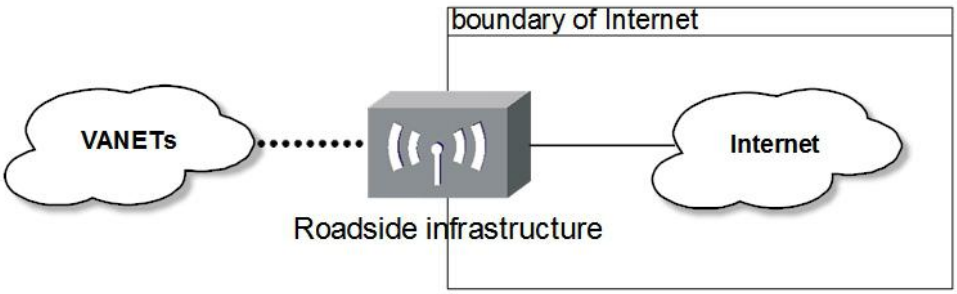

Figure 3.5: Internet boundary

3.4.2 VANETs boundary

Besides the boundary on the roadside infrastructure, inside a vehicle, the LAN is different with VANETs, the vehicle-mounted wireless router is at relative rest with its clients, with no Doppler effect; the clients do not need to roam, so the packet loss rate is lower than it in VANETs. Therefore, there is an other boundary for VANETs. Figure 3.6 shows the boundary.

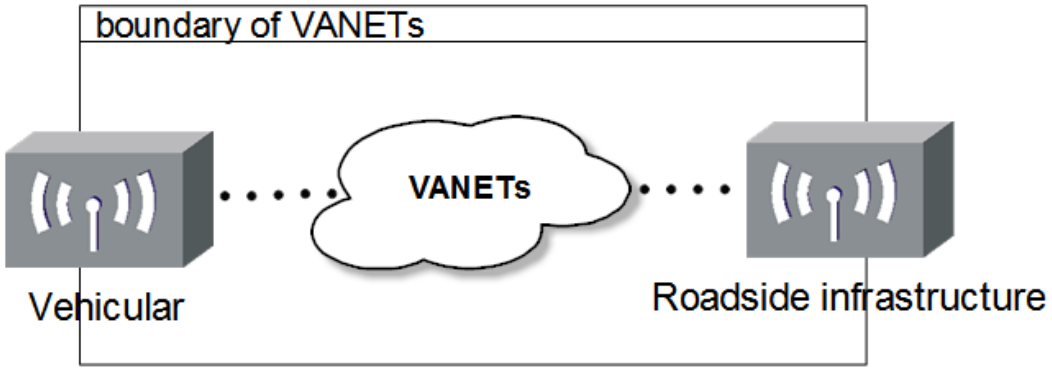

Figure 3.6: VANETs boundary

Converter model

Figure 3.7 gives out the topology of this model. In this figure, we add a converter between the traditional network (A) and the new network (B), this converter takes charge of translating A adaptive protocol to B adaptive protocol. To reduce the delay caused by the translating progress, and the load of the converter, the A adaptive protocol and B adaptive protocol should be one same protocol but in different versions. The B adaptive protocol should be one version modified from the original protocol running in A. In addition, this modification should not affect the working principle of the original protocol.

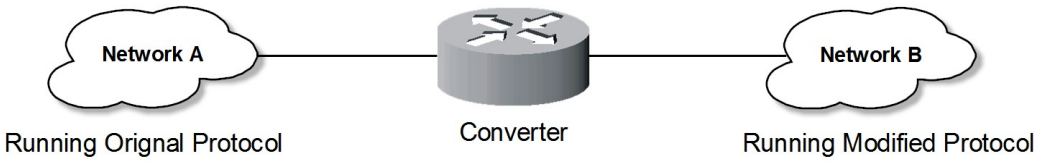

Figure 3.7: The topology of converter model

In our work, A is the internet. B is the VANETs. The A adaptive protocol is RTP. Therefore, we need to design a modification version of RTP. We call this modification version EC-RTP. In addition, because we want to use RTP server and player at the same time, we need to add another converter to translate EC-RTP back to RTP. Figure 3.8 shows the topology of our solution.

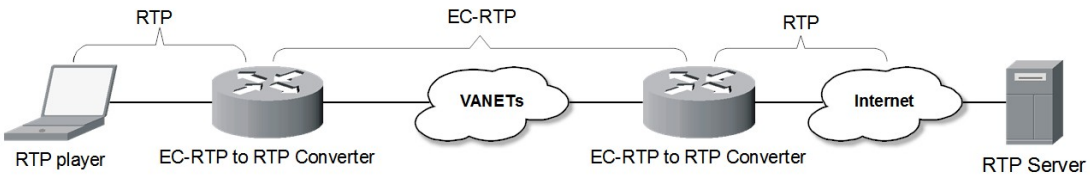

Figure 3.8: The topology of our converter model solution

The benefits of a modified protocol which works on the Internet are that we can add the some additional features to the protocol. However, modifying a specific protocol and developing a converter are still a lot of work. Though we achieve our goals, the cost of VANETs may not be reduced. This solution may more suitable for those networks that have certain requirements. Therefore, we start to design a new model. This model will not care about what kind of server and player we are using now. It will build a tunnel between the two converters.

Generic Tunnel

The Generic Tunnel has its own header, and just take the whole packets of protocol want to go through VANETs as its body. It will act like a lower level protocol. Just add its own header before the one it will carriedIt will act like a lower level protocol, but will add its own header before the one it will carry. The figure 3.9 shows us the position of the header of the Generic Tunnel. In fact, using this tunnel we can treat the Generic Tunnel as a new protocol. We can now use any technology of the VANETs to give this tunnel have strong ability to deal with high packets loss rate.

At the same time, different protocols may have different requirements. Some real-time protocol request low packet loss, some request low delay. One simple generic tunnel may not be able to afford these specific requirements at the same time.

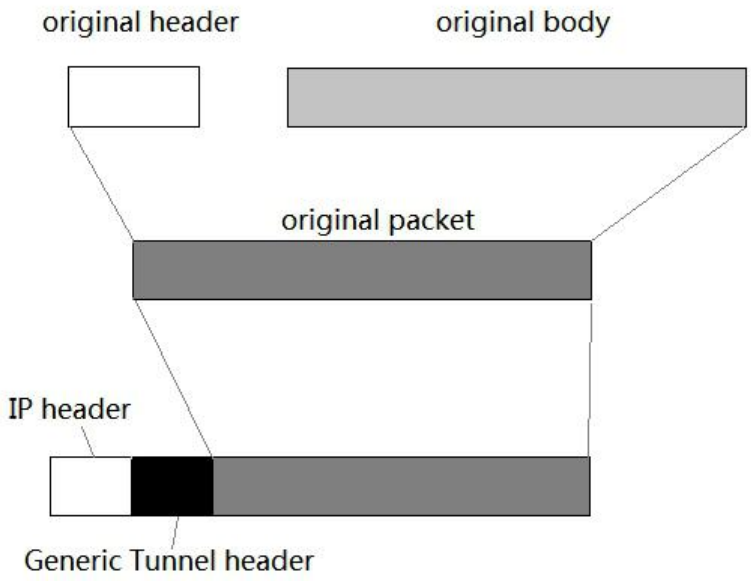

Figure 3.9: Generic Tunnel working principle

Redundancy Tunnel model

The Redundancy tunnel is one kind of Generic Tunnel. This tunnel can help the protocol which is designed for the Internet to gain the ability to resist the packet loss when it needs to go through the VANETs. The Figure 3.10, demonstrates how this kind of tunnel works. Assume that we need to send some packets from Network A through the VANETs to the Network B. The protocol we use is designed for the normal Internet, for example User Datagram Protocol (UDP). When the packet reaches the Redundancy Tunnel Gateway (RTG) A, the RTG will generate the redundancy packet, re-encapsulate the destination of the IP address to the RTG B and some additional information which will be needed when the other side of tunnel (RTG B) tries to recover the lost packets; after that, these packets will be sent through the VANETs. When the RTG B receives the packets, it will try its best to recover as many packets as it can, perform decapsulation to get the original protocol header, drop the duplicate packets, and resend the original protocol packets. The whole process for the protocol we use will be transparent; in other words, the protocol we use here will not be affected.

Besides, using this tunnel, we do not need the redundancy packets to travel from the very beginning to the end; these redundancy packets will only affect the load of VANETs. That will decrease the effect of speed of these redundancy packets.

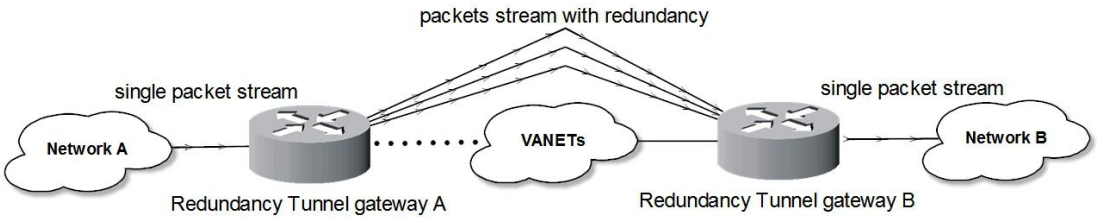

Figure 3.10: Redundancy tunnel working principle

3.4.3 Models Comparison

The table 3.1 shows below, outlines the advantages and disadvantages of these two models.

	Advantages	**Disadvantages**
Converter model	Can add different features based on the needs of different protocols	The tunnel cannot be reused for other protocols. Cannot have a standard for each protocol's tunnel
Generic Tunnel	Simply configured once, can be use for many protocols	The tunnel cannot be reused for other protocols. Cannot have a standard for each protocol's tunnel

Table 3.1: Advantages and disadvantages of different Models

Chapter 4

Protocols

4.1 RTP

4.1.1 Overview

RTP was developed by the Audio-Video Transport Working Group of the Internet Engineering Task Force (IETF) and first published in 1996 as RFC 1889, superseded by RFC 3550 [37] in 2003. RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. RTP does not address resource reservation. RTP is based on the UDP, so that when the packet loss occurs, we cannot ask the resource to re-transfer the lost packet. This does not guarantee quality-of-service for real-time services.

RTP packets are composed of a fixed-length header and variable-length body. Figure 4.1 shows the RTP's header. To let the RTP communicate from end to end, the header need not be changed; it carries a lot of useful information, such as the type of packets, time stamp, etc.

RTP's body can carry different format streams with different methods. There are a lot of standards made by IEEE, to make it easy to narrate, we will use H.264 as our payload

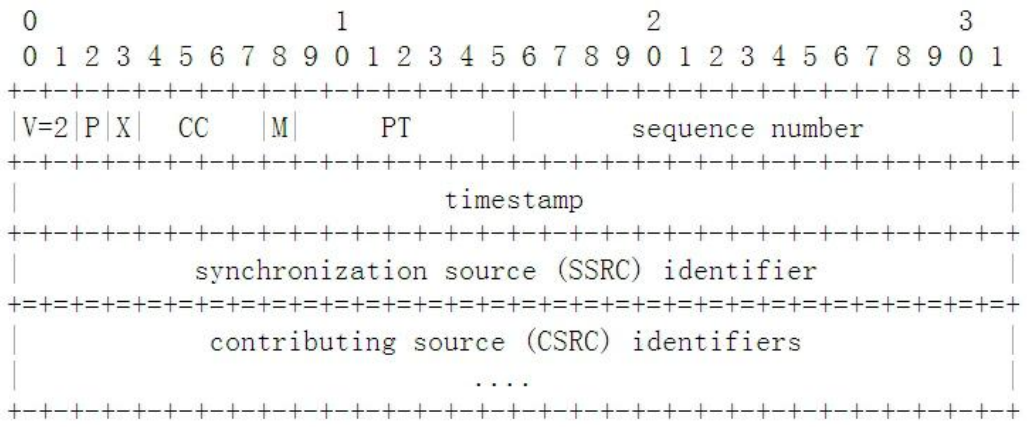

Figure 4.1: RTP header

format.

Because the RTP itself doesn't have any methods to control the packets loss and according to the H.264 codec is packet loss sensitive, real-time data packet loss, for a H.264 sequence of real-time video data is very detrimental.

4.1.2 RTP Payload Format for H.264 Video

H.264 is currently one of the most commonly used formats for the recording, compression and distribution of video content. The final draft work on the first version of the standard was completed in May 2003.

H.264 is now widely used in a lot of systems. For example, Apple has official adopted H.264 as the format for Quick Time [61]. It is also one of the formats chosen to be supported by both high definition DVD standards. It is also destined to be the future standard format for Blu-ray. H.264 is also used in camcorders and Blu-ray recorders.

The RTP payload format allows for packetization of one or more Network Abstraction Layer Units (NALUs). These NALUs produced by an H.264 video encoder. The payload format has wide applicability, as it supports applications from simple low bit rate conver-

sational usage, to Internet video streaming with interleaved transmission, to high bit rate video-on-demand [38].

The Figure 4.2 shows how the H.264 video is mapped into RTP packets. Because every frame in the video don't have the same size, and the NALUs can contain one frame or a part of frame, the NALUs also don't have the same size. Some NALUs is bigger than Max Transfer Unit(MTU). And Some NALUs is smaller than it. Because each RTP packet should be smaller than the MTU, for the bigger one, we cannot use it as payload of a RTP packet directly. So that we need use more than one packet to carry this bigger NALU. And for those smaller ones, we can use one RTP packet to carry several NALUs.

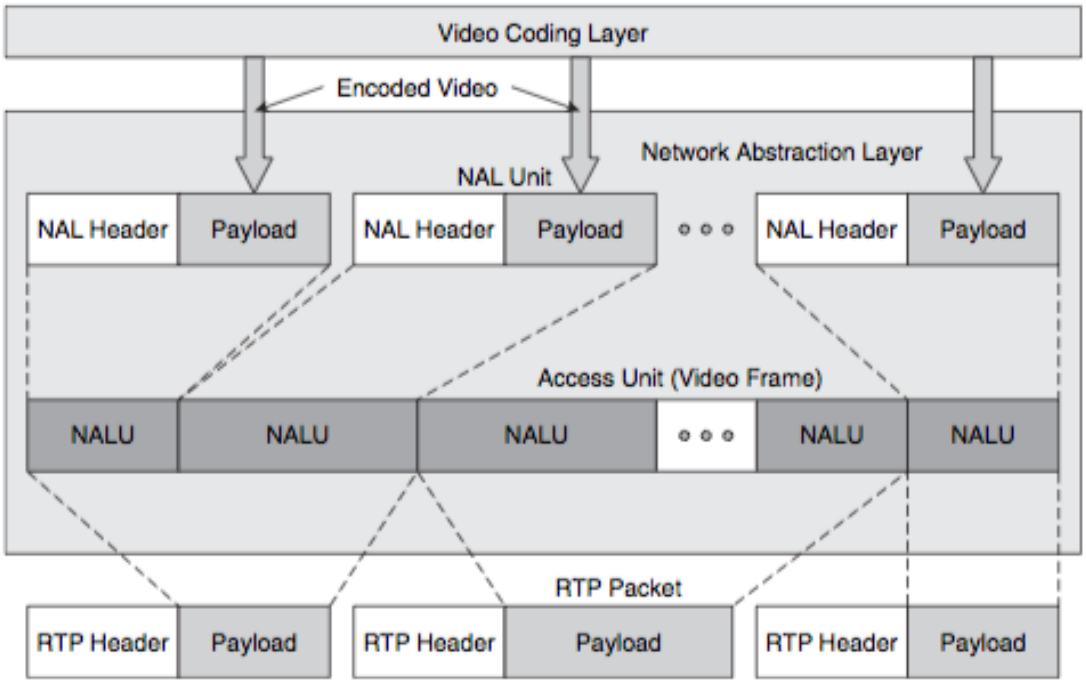

Figure 4.2: H.264 video map RTP packets

4.2 REDUNDANCY

In chapter 2, we decide to use the Erasure Coding technique as our transmitting strategy. The following section will introduce our Erasure Coding model. This model is based on

XOR.

4.2.1 Erasure Coding Model

The figure 4.3 shows our Erasure Coding model.

At first, we divide the sending data into blocks. These blocks have same size. And each block will be divided into 2 size isometric parts. We call the first part of data A, the second part of data B. We call the redundancy part C.

According to the character of XOR, it's easy to find these formulas below. We can generate C by using Formula 4.1.

$$C = A \oplus B \tag{4.1}$$

$$A = C \oplus B \tag{4.2}$$

$$B = C \oplus A \tag{4.3}$$

According to Formula 4.2 and 4.3, if we have any two of A,B and C, we can recover the missing one easily. This model can decrease the packet loss rate. At the same time, because the formulas have low computing complexity. Therefore, this model has a fast encoding and decoding speed.

4.3 EC-RTP

4.3.1 Prototype

First, we divide data into packets like normal RTP does. And then we group the packets as blocks, each block will have same even numbers of packets. We call this number S. Now the question is how we generate C in a proper way. We define the progress of generating C

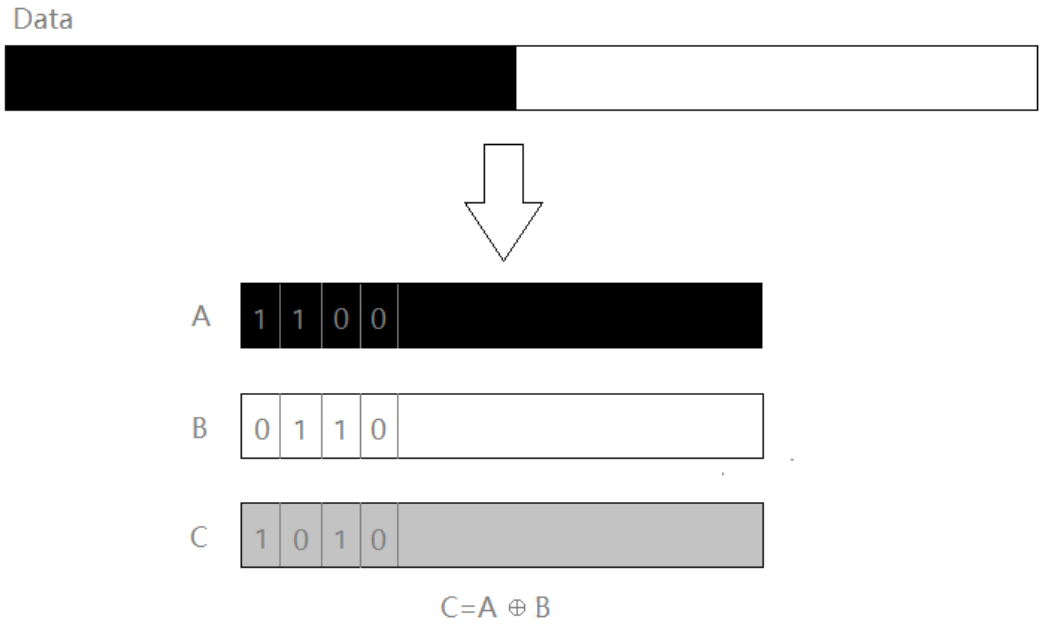

Figure 4.3: Our erasure coding model

packets as encoding progress, and we define the progress of recovering a packet as decoding progress. There are a lot of ways to generate C. We make it as Figure 4.4 shows. After applying our Erasure Coding model, for one block, the first half of the block is A. The other half of the block is B. And then we can have three new formulas. A_i means i^{th} packet in A. B_j means j^{th} packet in B. C_{ij} means C packet generated by A_i and B_j. All the operation will computed bit by bit.

$$C_{ij} = A_i \oplus B_j \tag{4.4}$$

$$A_i = C_{ij} \oplus B_j \tag{4.5}$$

$$B_j = C_{ij} \oplus A_i \tag{4.6}$$

Now we can generate C packets by using Formula 4.4. More C means low packet loss rate. However, it means more bandwidth consuming at the same time. So that we let users define how many C packets they want. For example, the first $S/2$ C are generated by $A_1 \oplus B_1, A_2 \oplus B_2 ... A_{S/2} \oplus B_{S/2}$; the second $S/2$ C are generated by $A_1 \oplus B_2, A_2 \oplus B_3 ... A_{S/2} \oplus B_1$; the third $S/2$ C are generated by $A_1 \oplus B_3, A_2 \oplus B_4 ... A_{S/2-1} \oplus B_1, A_{S/2} \oplus B_2$.

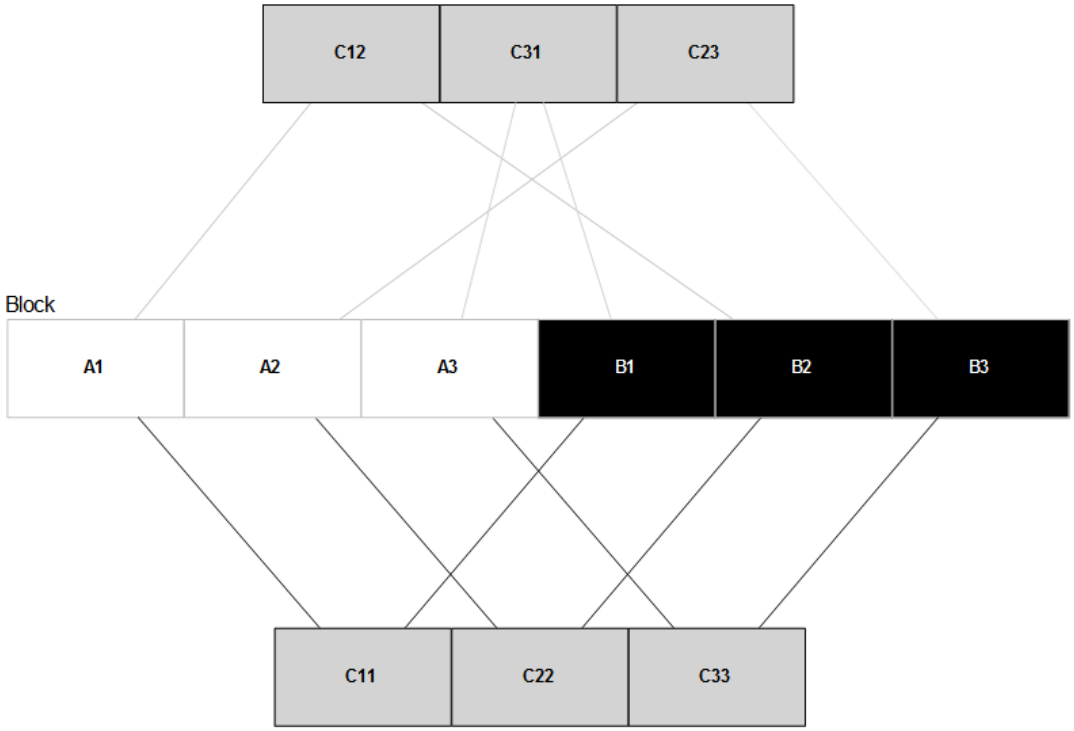

Figure 4.4: The strategy of generating redundancy packets

The reason why we use the model is that VANETs often loss packets in consecutive way. Imagine that we didn't use the model we talked above, we use a new way called alternate grouping instead: we pick up the first packet of the block, assume it is the first packet in A. We pick up the second packet of the block, assume it is the first packet in B. We pick up the third packet of the block, assume it is the second packet in A. Continue doing this until all the packets in the block have their group. Once the packet loss occurs, we lost 3 packets consecutive at the beginning of a block. If we use the alternate grouping

way described above, we will lost A_1, B_1 and A_2. Even we have received C_1, we cannot recover A_1 or B_1, because we need two of A_1, B_1 and C_1. However, in our original model, although we lost 3 packets of A, we can easily recover all lost packet back by using Formula 4.5.

4.3.2 Header design

In order to add the redundancy support to RTP, based on the Erasure Coding model we have introduced above, we need add some useful information into the header.

Since we have A packets, B packets and C packets, identifying a packet is very important. According to Formula 4.4, Formula 4.5 and Formula 4.6, i and j are also very important. For example, in encoding progress, we need them to generate C packets. In decoding progress, when a packet lost, we can use them to find which packet we need to recover the lost packet. We call i and j index number. To record index number in a packet, we add a new field in the header. We call this field index field.

As Figure 4.5 shows, we first use 2 or 1 byte for storing the index field. To make the index field more flexible, we design the 2 modes for the index field. They are normal mode and compressed mode. They are designed for adapting different situation. We use the first bit of the filed as a flag bit. If the flag bit is setted to 0, the index field is working in normal mode. Otherwise, if the flag bit is setted to 1, the index field is working in compressed mode. In the normal mode we use 2 bytes for storing the index field. The higher byte is use to store the A's index number, and the lower byte we use to store B's index number. If a packet is a C packet, the two bytes will store the A and B packet's index number which generate C out. For example, we generate $C_1 1$ by using $C_1 1 = A_1 \oplus B_1$. For the A_1 packet, the index field in the A_1 packet header is setted to 0000 0001 0000 0000. In the its index field, the higher byte stores A's index number. Because the packet is an A packet, the lower byte is all setted to zero. For the B_1 packet, the index field is setted to 0000 0000 0000 0001, the higher byte is setted to all zero, the lower is used to store the index number.

Because the first bit of the field is taking for record the mode and A packets and B packets should have same quantity, we can have $2^7 = 128$ A packets and B packets. Therefore, the maximum block size is $128 + 128 = 256$. However, in some situations, we do not need such a big block size. Therefore, we need the compressed mode. In the compressed mode, we use only one byte for storing both index number of A and B. The 3 bits after the flag bit is used for storing A's index number, the following 4 bits is used for storing B packets' index number. Depending on we need the A packets and B packets have same quantity, in fact only the last 3 bits is used for storing B packets' index number. In the compressed mode, the protocol will support small blocks with a size less than $2^3 + 2^3 = 16$. In Section 4.2.1

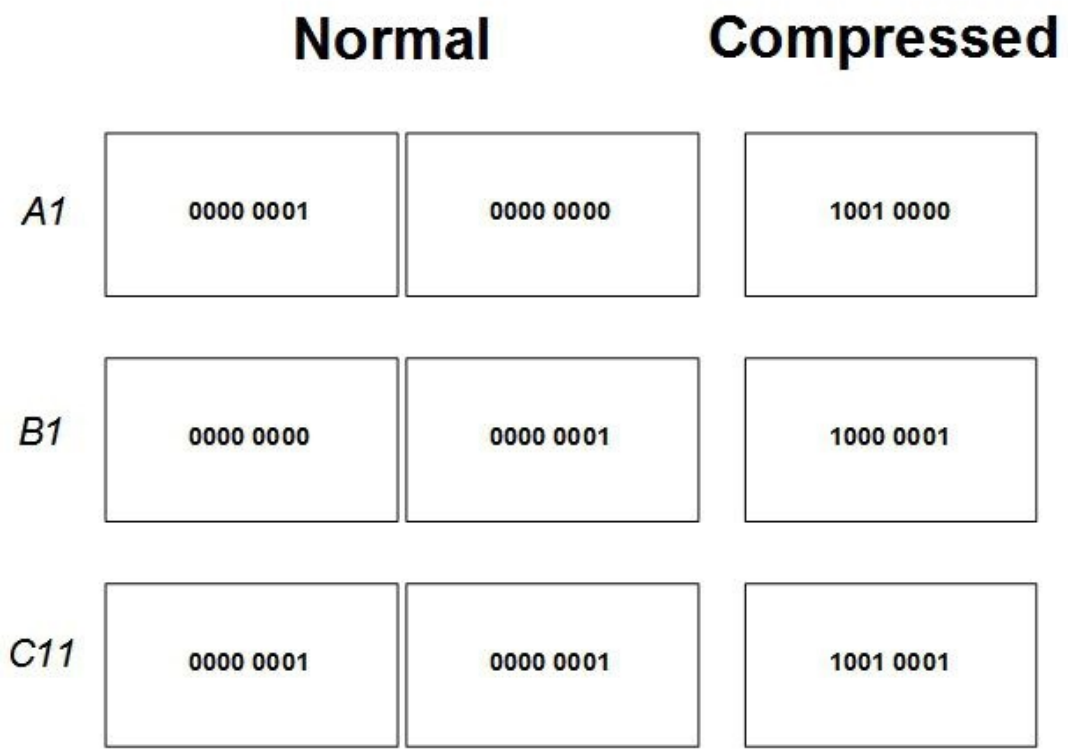

Figure 4.5: An example of Index field in different modes

we discussed about the our Erasure Coding model, doing XOR bit by bit needs the packets have the same size. However, in Section 4.1.2 we know about that the H.264 payload RTP packets don't have same packet size. It depends on the original video's frame size. Figure

4.6 describes this situation.

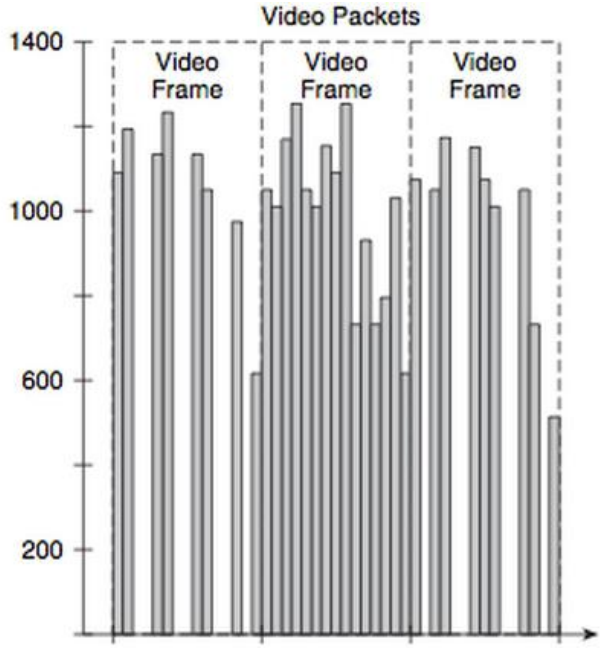

Figure 4.6: Packets will not have same length

According to [3], the big NALUs can be split into several packets, While the small NALUs can be aggregated in one packet. With this feature, some packets will reach the maximum packet size, but some will not. It can help us decrease the influence of the different sizes, but this is not enough; what we need is for the packets to have exactly the same size. To achieve this goal, we firstly defined a maximum packet size. Choosing the MTU as the maximum packet size is an effective way. And then we simply add 0 to the tail of a packet which is smaller than the maximum packet size. Make the smaller packet reach the maximum packet size. Figure 4.7 shows how to make packets to have same size.

After this progress, there may have some useless zeros at the end of a packets. Since they are just located in the end of a packet, we add a field to the header to record the original packet size. In the decoding progress, we cut off the packet with the size in this field. MTU is set ordinarily around from 1400 to 1500. Therefore, we use two bytes for

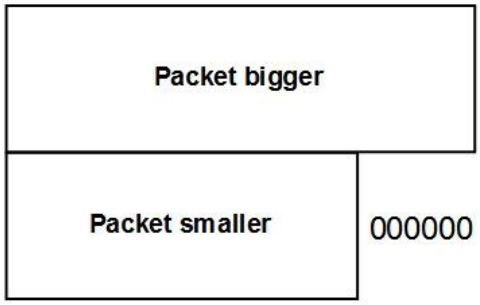

Figure 4.7: Make packets to have same size

storing this field. At last we can support the packet size from 0 to 65535.

Out of all the previously discussed designs, the modified RTP header will be like Figure 4.8.

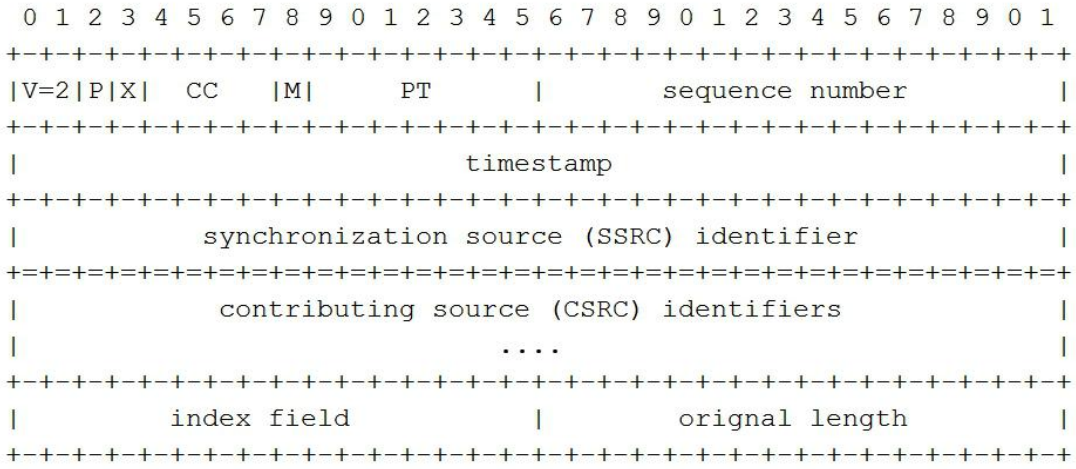

Figure 4.8: EC-RTP header

4.3.3 EC-RTP server

It is easy to build an EC-RTP server based on a RTP server, because of the similarity of the two protocol. The video encoding part will be the same. What we need to do is

generating the redundancy packets. And then modify every packet header and then send them out.

4.3.4 Performance influence by redundancy packets

More redundancy packets means there will be more data need to be sent out. When network congestion happened, the bandwidth may not be enough to send out these packets. Therefore, the redundancy packets could impact the experience of playing a real-time video.

A proper way to solve this problem is that we can use the idea behind bit rate adjustment technique. We can change our redundancy packets number based on current sending rate. The detail design will show in the chapter 5.

4.4 EC-RTP to RTP Converter

4.4.1 Design

Figure 4.9 shows the basic topology of the EC-RTP to RTP converter. One converter locates on the boundary between Internet and VANET. The other converter locates between VANETs and the RTP player. And they act as a pair. They know each other's IP address, and they will do the following things: when one receives the RTP traffic, it will translate these traffics to EC-RTP traffic, and then it relays the traffics to the other converter. It will translate the packets back and send them to the right destination.

The detail design and implementation will be showed in Chapter 6.

4.5 Generic Routing Encapsulation (GRE)

Generic Routing Encapsulation (GRE) is a tunneling protocol developed by Cisco Systems that can encapsulate a wide variety of network layer protocols inside virtual point-to-point

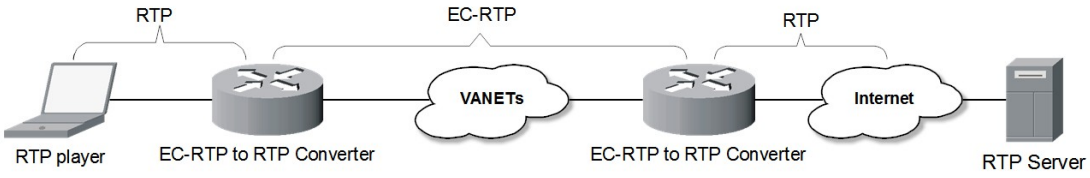

Figure 4.9: EC-RTP to RTP Converter

links over an Internet Protocol inter-network [39].

The following Figure 4.10 shows how the GRE works. GRE carries other protocols as its payload. Therefore, GRE can carry any kind of packets without knowing what they really are.

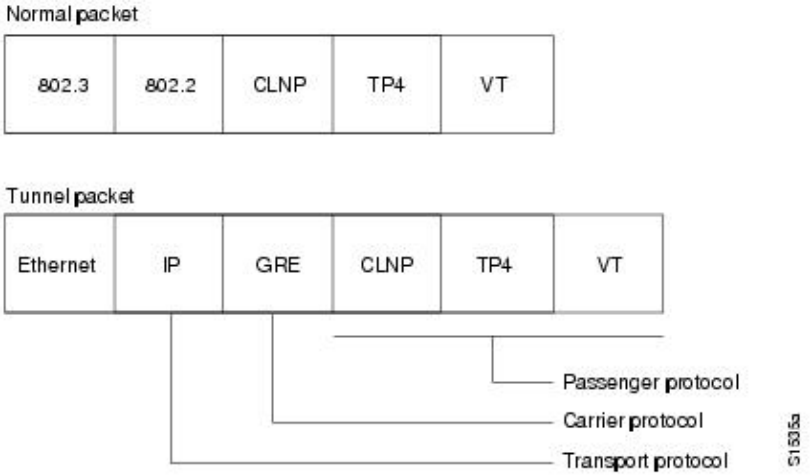

Figure 4.10: GRE header

4.6 EC Generic Routing Encapsulation (EC-GRE)

In the Section 3.4.2, we have design a model of redundancy tunnel, which can carry many different kinds of protocols. In Section 4.5 we have introduced the GRE, we can modified it to a redundancy tunnel with same Erasure Coding model we have introduced above,

which we have called it EC-GRE.

The only difference between the EC-GRE and EC-RTP is that EC-GRE carries RTP as its payload. Therefore, we can use the same modification we do to the RTP to the GRE. Add the index field and the packet size field to the GRE's header. At last, we run the EC-GRE between the converters, now the converters become the tunnel gateways.

Chapter 5

Implementation and Demonstration

5.1 Overview

Nowadays, most research about VANETs is based on the Network Simulator Version 2 (NS2) instead of the real devices. Though NS2 has its own advantages, the simulated result may different than the result on the real devices. After we have designed our protocols, we will implement them on the real devices.

Recently, Google and a number of automakers have been planning to bring Android to cars with the launch of a new group called the Open Automotive Alliance. The alliance consists of Google, GM, Honda, Audi, Hyundai, and chipmaker Nvidia, and will focus on bringing the successful mobile operating system to in-car entertainment systems "in a way that is purpose-built for cars." The first cars with Android integration are planned for launch by the end of 2014. Because the Android system will be a potential system launched on the cars, we will focus on this system.

We will use the newest Android tablet Google's Nexus 10 as our platforms. Here is the summary of this device:

- CPU: Dual-core ARM Cortex-A15

- GPU: Mali-T604

- Memory: 2 GB RAM

- Wireless: Wi-Fi 802.11 b/g/n (MIMO+HT40)

- Android system: 4.2.1

- Cameras: 5 MP (main) ,1.9 MP (front)

Because Nexus 10 is made by Google, we can have a pure Android device to run our demonstration. In addition, the Wi-Fi connection of Nexus 10 is more stable.

We will use 3 Nexus 10 devices, each of them playing a different role in the system. The first one is the video streaming server. The second one is the video player. And the last one is a simulator. It can simulate packets loss . The video streaming server have 2 different modes. One is providing the video which captured from the tablet's camera, the other one is providing the video file which is inside the video server.

Figure 5.1: Google's Nexus 10

5.2 Topology

Figure 5.2 shows the whole demonstration's topology. We use 3 tablets, one is The Real-Time Streaming Protocol (RTSP) server, one is the RTSP player, and both of them are connecting to the VANETs routing simulator.

Since we want to show that our solution are compatible with Internet protocols, we will use the RTSP protocols, which will be introduced in the following section.

- RTSP server: Here is the place where we store the video file or capture the camera and provide the video streaming service.

- VANETs Routing simulator: the server and player will connect to this tablet, like a wireless access point. What's more, here the tablet will drop some packets randomly, so we can simulate the situation often occurs in VANETs, transparent route changed, packets lost.

- RTSP player: This is the client side of the system. This tablet will contact the video streaming service and play the stream which the server sends back.

5.3 RTSP server

5.3.1 Why use RTSP

RTSP is a very famous protocol. Here is the list of reasons why we have chosen it.

- RTSP is one of the IETF standards, the document is RFC 2336. The standard helps the protocol maintain consistent behaviour, no matter what company implements it.

- It will use RTP as its transport layer protocol. We can easily convert it to our own EC-RTP.

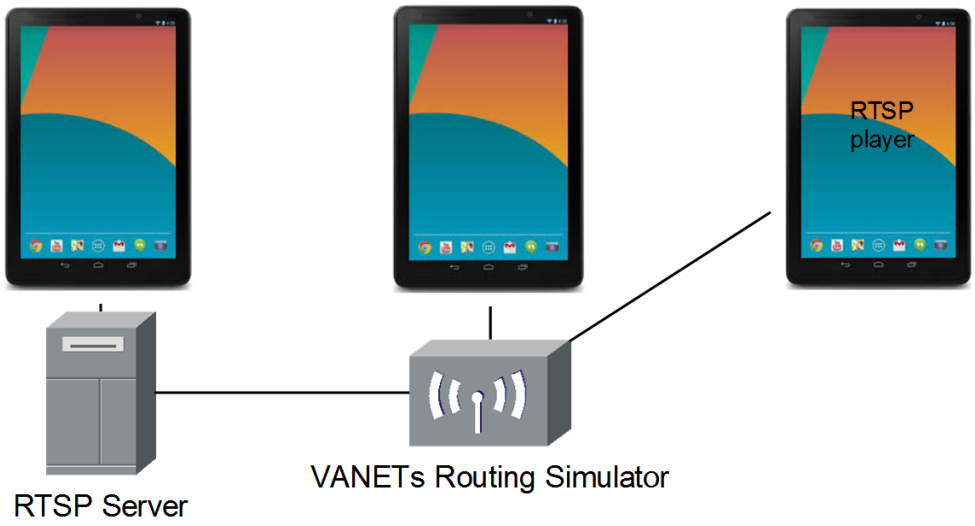

Figure 5.2: Topology

5.3.2 RTSP protocol

For now, there are few RTSP servers based on the Android system. One aspect of our work is to implement a proper RTSP server on the Android system. Before we do this, we need to know how this protocol works.

RTSP overview

The Real-Time Streaming Protocol (RTSP), developed by RealNetworks and Netscape, is a application layer protocol; it establishes and controls either a single or several time-synchronized streams of continuous media such as audio and video.The protocol is intentionally similar in syntax and operation to HTTP/1.1. RTSP server uses Uniform Resource Locator (URL) [40] to identify resources.

When client requests a URL, the server will first build a session for the client, so that it can track every client. And then use the Session Description Protocol (SDP), which will be introduce in the following section, to give the streaming information to the client; the server will then negotiate with the client about the transport protocol (RTP) and the port

number for each streaming. After this progress is done, the server will start to send video streaming to the client with the protocol which they have negotiated.

RTSP can be based on TCP or UDP, in our implementation, because of the high packet loss rate on VANETs, we use TCP to carry RTSP's packets.

RTSP URL

RTSP URL is used to locate the resources on the server, we can give URL to a single file, or we can give URL to the device such like video camera, microphone, etc. URL parser is one of key parts of our implementation, we use Android's API to implement this part [41].

RTSP URL grammar:

$RTSP\_URL = (\text{"}rtsp : \text{"}|\text{"}rtspu : \text{"}|\text{"}rtsps : \text{"})\text{"}//\text{"}host[\text{" : "}port][abs_path]$

Some examples:

$rtsp : //192.168.20.136 : 5000/$

$rtsp : //www.example.com/media.mp4$

RTSP methods

RTSP have 2 types of methods; one is the request method, and the other is the response method. They have different formats.

Here are some useful methods we need to implement [42].

1. OPTION

   OPTION method is a request method; the client uses it to query how many kinds of methods are available on the server.

2. DESCRIBE

Client use the DESCRIBE method to ask the server for information about the media data which it requested. In addition, client can use the Accept header to tell the server can be understood on the client side.

3. SETUP

   The SETUP method could let clients on the server side tell which kind of transport protocols and port numbers it can use for receiving streaming data. Even for the stream already built up, the client can also use the SETUP method tell whether the server uses other protocol (If serverdoesn't support it, it will respond with the error message: "45x Method not valid in this state").

4. PLAY

   Client use the PLAY method to tell the server that it is ready to receive stream data with the protocol and port number confirmed by the SETUP method. Therefore, if the client does not receive the acknowledgement of the server for the SETUP method, the client cannot send PLAY's request.

   PLAY method can use a Range header determine the time region of data it needs from the server side, which means the Range header will give the start and end position of a stream. Besides, the PLAY request could be pipelined (queued), server should handle the PLAY requests in the arrived order.

   One PLAY request without a Range header is also valid. It represents a stream which will be played from the very beginning to the end. Or the stream will be played form the very beginning to the time when it get paused. If a stream has been paused, the stream's start point will be the point when it get paused.

5. TEARDOWN

   The client use TEARDOWN method to tell the server that it doesn't need any more data. The server will close the session and stop the data transmission.

6. GET_PARAMETER

The client use this method as a heartbeat packet to determine if the server is still alive.

Here is an example of commands between the RTSP Server(S) and Client(C). this progress shows how a client connects to the server.

1. $S \leftarrow C$:

   OPTIONS rtsp://www.example.com/media.mp4 RTSP/1.0

   CSeq: 1

   Require: implicit-play

   Proxy-Require: gzipped-messages

2. $C \leftarrow S$:

   RTSP/1.0 200 OK

   CSeq: 1

   Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE

3. $S \leftarrow C$:

   DESCRIBE rtsp://audio.example.com/twister/audio.en RTSP/1.0

   CSeq: 2

4. $C \leftarrow S$:

   RTSP/1.0 200 OK

   CSeq: 2

   v=0

   o=- 2890844526 2890842807 IN IP4 192.16.24.202

   s=RTSP Session

m=audio 0 RTP/AVP 0

a=control:rtsp://audio.example.com/twister/audio.en

m=video 0 RTP/AVP 31

a=control:rtsp://video.example.com/twister/video

5. $S \leftarrow C$:

   SETUP rtsp://audio.example.com/twister/audio.en RTSP/1.0

   CSeq: 1

   Transport: RTP/AVP/UDP;unicast;client_port=3056-3057

6. $C \leftarrow S$:

   RTSP/1.0 200 OK

   CSeq: 1

   Session: 12345678

   Transport: RTP/AVP/UDP;unicast;client_port=3056-3057;server_port=5000-5001

7. $S \leftarrow C$:

   SETUP rtsp://video.example.com/twister/video RTSP/1.0

   CSeq: 1

   Transport: RTP/AVP/UDP;unicast;client_port=3058-3059

8. $C \leftarrow S$:

   RTSP/1.0 200 OK

   CSeq: 1

   Session: 23456789

   Transport: RTP/AVP/UDP;unicast;client_port=3058-3059;server_port=5002-5003

9. $S \leftarrow C$:

   PLAY rtsp://video.example.com/twister/video RTSP/1.0

   CSeq: 2

   Session: 23456789

   Range: smpte=0:10:00-

10. $C \leftarrow S$:

    RTSP/1.0 200 OK

    CSeq: 2

    Session: 23456789

    Range: smpte=0:10:00-0:20:00

    RTP-Info: url=rtsp://video.example.com/twister/video;seq=12312232;rtptime=78712811

11. $S \leftarrow C$:

    PLAY rtsp://audio.example.com/twister/audio.en RTSP/1.0

    CSeq: 2

    Session: 12345678

    Range: smpte=0:10:00-

12. $C \leftarrow S$:

    RTSP/1.0 200 OK

    CSeq: 2

    Session: 12345678

    Range: smpte=0:10:00-0:20:00

    RTP-Info: url=rtsp://audio.example.com/twister/audio.en;seq=876655;rtptime=1032181

Now that we have obtained our RTSP server, because the negotiation progress is very important and short, we decide to use TCP to transfer them. In addition, we need the SDP protocol in the negotiation progress.

5.3.3 MP4 file

Based on two modes we want to implement (file and real-time video), before we continue to introduce the SDP protocol, we firstly analysis the MP4 file. With gathering information about the MP4 file, we can let our RTSP protocol give the SDP feedback.

ISO/IEC 14496-14:2003 specifies the MP4 file format as derived from ISO/IEC 14496-12 and ISO/IEC 15444-12, the ISO base media file format. It revises and completely replaces Clause 13 of ISO/IEC 14496-1, in which the file format was previously specified.

The MP4 file format defines the storage of MPEG-4 content in files. It is a flexible format, permitting a wide variety of usages, such as editing, display, interchange and streaming.

The basic unit of MP4 file is Box. These Boxes can contain either data or metadata. MP4 standard permits us to use different data Box and metadata Box. Normally, putting the metadata before data, we can read more information before playing this video (audio). Figure 5.3 shows the MP4 file's structure.

Box

First, the order of bytes in a Box is the network byte order known as Big-Endian. Simply, that means, high byte is in the low end of the memory. The Box is composed by the header and body. The header will tell which the type and size of the box, and the body will have a different format and meaning based on the type.

For our task, the biggest priority is to convey information about the MP4 file, width, height, fps, codec, etc. These information is located in MOOV box. In the MOOV box,

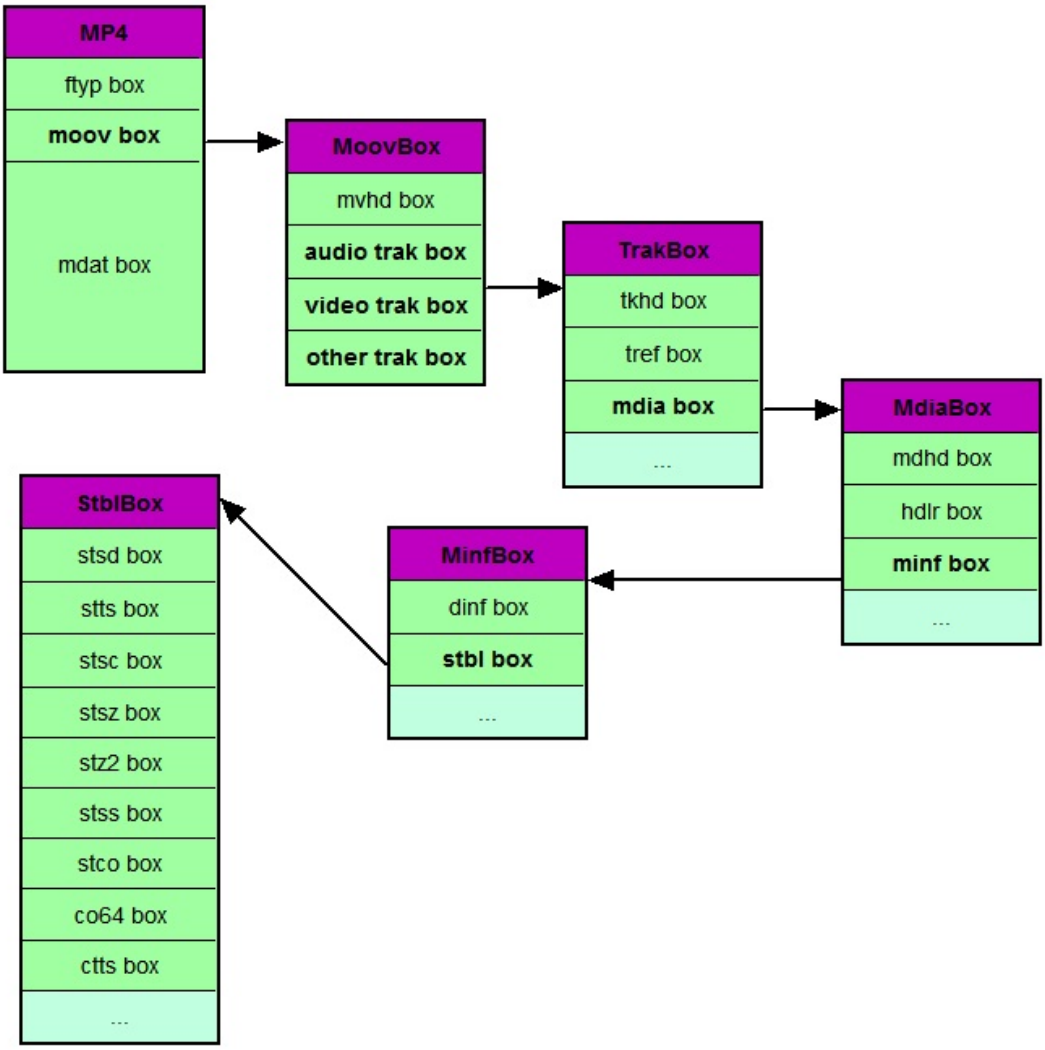

Figure 5.3: MP4 file structure

there are some track boxes, Which boxes are used to describe the media information about the video and audio. So first, we need go to the track box, determine whether it is a video track or not, and then we need to know if the video is H.264 video or not. In the track box, we have mdia box, which is used to record the information about the media. In the mdia box, we can find minf box, in minf we can find stdl, and in stdl, we can find stsd, Which is our final location. The H.264 video will have a avc1 box in the stsd. The avc1 is also named AVCDecoderConfigurationRecord. What we need in the avc1 box, is SPS and PPS.

SPS and PPS

SPS is also named Sequence Parameter Set. It includes all necessary information about one sequence. PPS is also named Picture Parameter Set. It includes information about a picture such as the type and sequence number. They are very important for the player to play the video successfully.

To get these two parameters, we need to read the avc1 box. Figure 5.4 shows its structure, as well as the method to get it perfectly.

5.3.4 SDP protocol

The Session Description Protocol (SDP) provides a standard representation for the information such as: media details, transport addresses, and other session description metadata to the participants. It is purely a format for session description – it does not incorporate a transport protocol. We will use SDP in RTSP.

Table 5.1 shows the main description of SDP. For us, the relevant information is the SPS and PPS. Figure 5.1 shows an example about how to describe a video stream. Notice in the sprop-parameter-sets parameter, we attached the SPS (the first one) and PPS(the Second one) with this parameter. For the file mode, we read the file, find the avc1 box,

unsigned int(8) configurationVersion = 1;	8bit
unsigned int(8) AVCProfileIndication;	8bit
unsigned int(8) profile_compatibility;	8bit
unsigned int(8) AVCLevelIndication;	8bit
bit(6) reserved = '111111'b;	6bit
unsigned int(2) lengthSizeMinusOne;	2bit
bit(3) reserved = '111'b;	3bit
unsigned int(5) numOfSequenceParameterSets;	5bit
for (i=0; i< numOfSequenceParameterSets; i++) {	
unsigned int(16) sequenceParameterSetLength ;	16bit
bit(8*sequenceParameterSetLength) sequenceParameterSetNALUnit;}	
unsigned int(8) numOfPictureParameterSets;	8bit
for (i=0; i< numOfPictureParameterSets; i++) {	
unsigned int(16) pictureParameterSetLength;	16bit
bit(8*pictureParameterSetLength) pictureParameterSetNALUnit;}}	

Figure 5.4: AVCDecoderConfigurationRecord structure

Parameter	Purpose
v =	Protocol version
o =	Session owner / Session ID
s =	Session name
*i =	Session information
*u =	Session URI
*e =	E-mail address
*p =	Phone number
*c =	Connection information
*b =	Bandwidth information
*z =	Time zone
*k =	Encryption key
*a =	Session attribute
t =	Session activity time
r =	Repeat times
m =	Media name and transport address

Table 5.1: SDP session description (* is optional)

Type	Name	Purpose
200	SR	Sender Report
201	RR	Receiver Report
202	SDES	Source Description Items
203	BYE	Stop transmitting
204	APP	Used for a specific application

Table 5.2: 5 types of RTCP packet

and read the SPS and PPS. If we are now in real-time video mode, we can record the video on the server for one second, use a Android API, save a temporary MP4 file, get the parameter, and then delete that file.

```
m=video 0 RTP/AVP 96
b=AS:13953
a=rtpmap:96 H264/90000
a=mpeg4-esid:201
a=control:trackID=65736
a=fmtp:96 profile-level-id=428015; packetization-mode=1; sprop-parameter-sets=Z00AH41oBQBbkA==,aO4G8g==
a=framesize:96 1280-720
```

Figure 5.5: A SDP example

5.3.5 RTCP protocol

RTP is defined for transmitting audio, video and other real-time data transmitting protocol. Compared with the traditional transport level protocols which focuses on reliability, it focus on real-time performance. It often works with RTCP. RTCP can help RTP to manage the session, through flow control, congestion control, etc.

Table 5.2 shows 5 types of RTCP packets. We use RTCP to transport the packet loss rate information between the server and client side. This information either can adjust our redundancy packets number.

5.3.6 Architecture of RTSP server

We now have many protocols, so our next step is to implement them and make things work together. We divide the our RTSP server into four layers. The first layer is the UI layer. It takes charge of communicating with user, accepting the request url and some parameters. The second layer is the RTSP layer. It has some functions including parsing RTSP command and RTSP URL, and controlling RTSP sessions. These functions let the server can negotiate with the client. And then they can set up the proper transport protocol between them. The third layer is the streaming abstract layer. It helps the system manage streams. When we have decided to send out a video stream with negotiated protocol, the server will choose a proper packetizer of that protocol to packet this stream. The last layer is the function layer, which will have many useful classes of function, such as RTP packetizer, MP4 file parser, some high efficiency data structures, etc. Figure 5.6 shows all of them.

Basically, when the server received a request, firstly URL parser will try to locate the resource on the server. And then, the session controller will then help the client build up a session. Then they will communicate with each other using the RTSP commands. In the DESCRIBE comand, session controller will use the MP4 file parser to generate the SDP information. In the SETUP command, the session controller will give a stream to the client session. This stream is setted up with proper transporting protocol. The packetizer of the chosen transporting protocol will start to generate packets.

5.4 RTSP player

Google's Android Software Development Kit (SDK) has a VideoView class which can display a video file. The VideoView class can load images from various sources (such as resources or content providers), takes care of computing its measurement from the video so that it can be used in any layout manager, and provides various display options such as

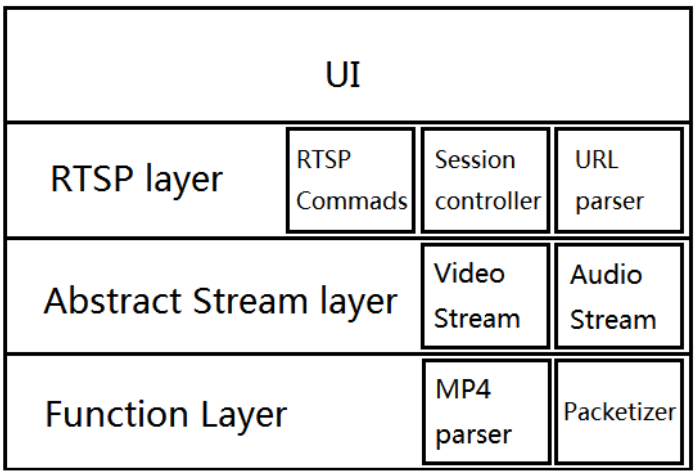

Figure 5.6: Architecture of RTSP server

scaling and tinting. And it can play the RTSP URL directly. [43] To validate our system compatibility, the player will not be change;, we will use exactly the same thing in Google's Android SDK. Actually, we will use this client as a black box.

5.5 Converter

5.5.1 Architecture of Converter

The converter will have two roles; one is the sender and the other one is the receiver. Both of them can use RTCP to send each other some key information like packet loss rate. Each of them will have 3 threads per session. One is in charge of receivings packets, sorting them to right order, one is in charge of decoding/encoding the redundancy packets, and one is in charge of sending packets out. Each thread can work independently, and the thread all have their own memory. Therefore, we can decrease the delay caused by sorting, decoding, encoding. Figure 5.7 shows the architecture of the converter.

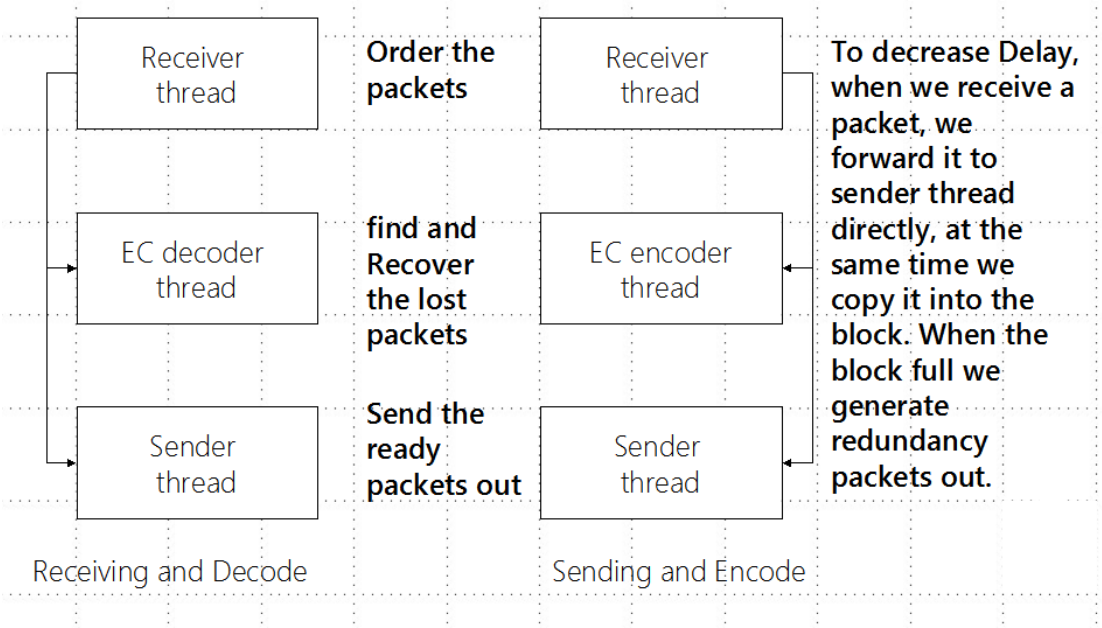

Figure 5.7: Architecture of Converter

5.5.2 Encoding

The strategy of encoding will follow what we have discussed above in chapter 4. However, in implementation level, there still one thing that needs to be considered. If the converter receives and stores the RTP packets until it has a full block. And then it generates the redundancy packets, then send out all the packets. Because it need to wait for receiving all block of packets, that will cause unwanted delay. In this case, we use a cache to copy the receiving packets, and then forward these packets out. Once we have a whole block, we can generate the redundancy packets and send them out.

5.5.3 An EC-RTP/RTP packets sort algorithm

When receiving EC-RTP/RTP packets, one difficult problem lies in how to determine whether a packet have not reached its destination yet because of the network delay, or it just have been lost during the transport. Because each packet's delay maybe different, the packets may not reach their destination in the order they were sent. Thus, an efficient

algorithm to sort the EC-RTP/RTP packets is required.

Considering this problem, we can easily find that we cannot store all the packets, and then sort them based on the sequence number field in the packet. The reason why we cannot do this is mainly because, first, we cannot save all the packets in the memory; and second, even if we use the quick sort method to handle it, the time complexity will be $\mathcal{O}(n \lg n)$ which will cause an unacceptable delay. One method that would likely be successful would be to sort the packets as soon as we received them.

As we mentioned above, there will still have problem about the determine a packet delayed or lost. According to the RTP, we do not have any retransmission mechanisms, so a sensible method is to set up a timer and a time-out time. When a packet exceed exceeds the allotted time, we treat this packet as a lost packet.

A high efficiency data structure: Bitmap

According to [44], a Bit-map uses a bit to mark the value of an element, and the key could be the element itself. The bitmap actually is just an array, which we can use to map to other elements. Table 5.3 shows the advantages and disadvantages of this data structure.

An example of using Bitmap to sort elements:

Image we have these numbers: 2,6,4,3,1. We need to sort them in ascending order. The following is the progress of using a bitmap to sort them.

Because we only have numbers less than 7, we only need 8 bits (1 byte) to represent all the elements. Firstly, we initialize this byte with all 0 (shown in Figure 5.8). Now, each digit represents a number. 1 means there is a element in that position. 0 means there are no elements in that position. For example, the first digit represent 0, the second digit represents 1, the third digit represents 2, etc. Also, we can use the first digit represents 1, but here we use the first digit represent 0 in the example.

In step two, we traverse all the 5 elements. The first element is 2, we just change the third bit from 0 to 1. Figure 5.9 shows the result.

Advantages	Disadvantages
• High operational efficiency, no need to compare and swap. • Occupy less memory, for example, if we have about 10000000 elements, we just need about: $10000000/8bit = 1250000Byte = 1.25MB$.	• All the element should not be duplicated.

Table 5.3: Advantages and disadvantages of using bitmap sort elements

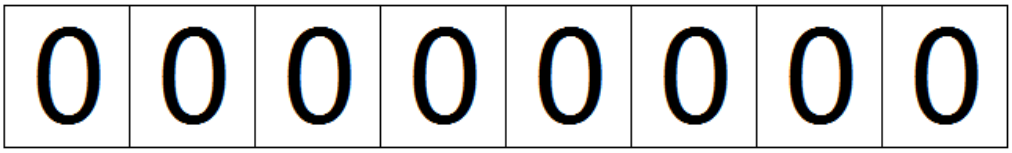

Figure 5.8: The state of the bitmap after initialization

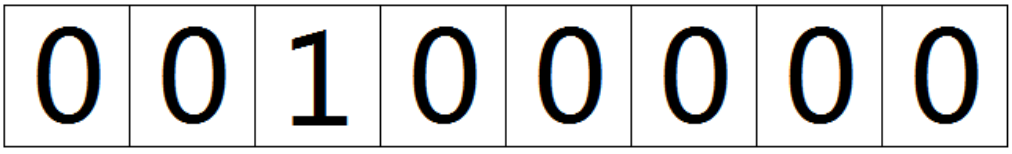

Figure 5.9: The sate of the bitmap after handling the first element

The following process will handle all the remaining elements and repeat step two. The next element is 6, we change the seventh bit to 1. The element after 6 is 4, we change the fifth bit to 1. We keep doing this, until we reach the last element. Figure 5.10 shows the final state of the bitmap.

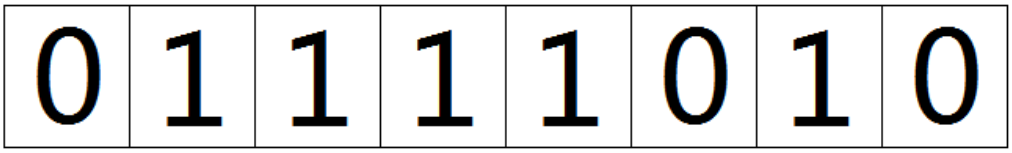

Figure 5.10: The final state of the bitmap after traversing all elements

When we finish, we can traverse the bits in the bitmap. When we find a bit that has been set to 1, we output its position in the bitmap, so we will get the result: 1,2,3,4,6.

Bitmap can save space, and also have a good speed to solve sorting problems. The time complexity will decrease to $\mathcal{O}(n)$, which is very suitable for the mobile device with limited CPU speed and limited memory. Additionally, all the packets for a session will have continuous sequence numbers, in other words, there will be no duplicate sequence number per session, only some missing. In a word, we can use a bitmap to sort these packets.

Design a data structure to solve this problem

Now that we can use a bitmap to design a high speed cache, we can find the a packet's right place by the $\mathcal{O}(1)$ time cost. First we can repeat the process as we described above. We use one bitmap and one array, using a bitmap map to the index of the array, just like Figure 5.11 shows.

let's assume the sequence number of a packet is Seq, the index of array is i, and the cache size is S. We can get the relation between them with this formula.

$$i = Seq \mod S \tag{5.1}$$

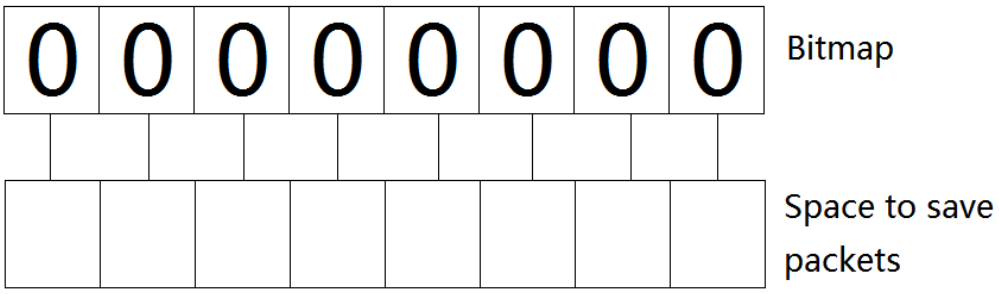

Figure 5.11: Basic data structure: one bitmap and one array

When we receive a packet, we read the sequence number of this packet. With this formula we can get the packet's memory space, copy this packet into the right space, and then we can set the relevant bit to 1.

However, we still have some problems. When should we stop this progress and send all the packets out to the decoding thread ? Just waiting for the packets fulfil the cache may be one simple way. However we need to solve the loss/delay problem first. Because, if there is a packet lost, the cache cannot be filled up. Besides, as we cannot store whole packets in the memory, we should send some packets already in right order to the decoding thread. Therefore, we must determine when we will send them out of the cache. If we don't send them out on time, the following packets may overwrite the previous packets because of the Formula 5.1 which we have used.

It seems like only one bitmap does not satisfy our requirements, so we add another bitmap help us to solve these problems.

Now we have 2 bitmaps. When we received one packet, we checked the first bitmap; if there is already one packet, we check the second bitmap, if there is already one packet, we forward whole packets in the cache to the next progress. If the packet do not reach in time, we simply drop it.

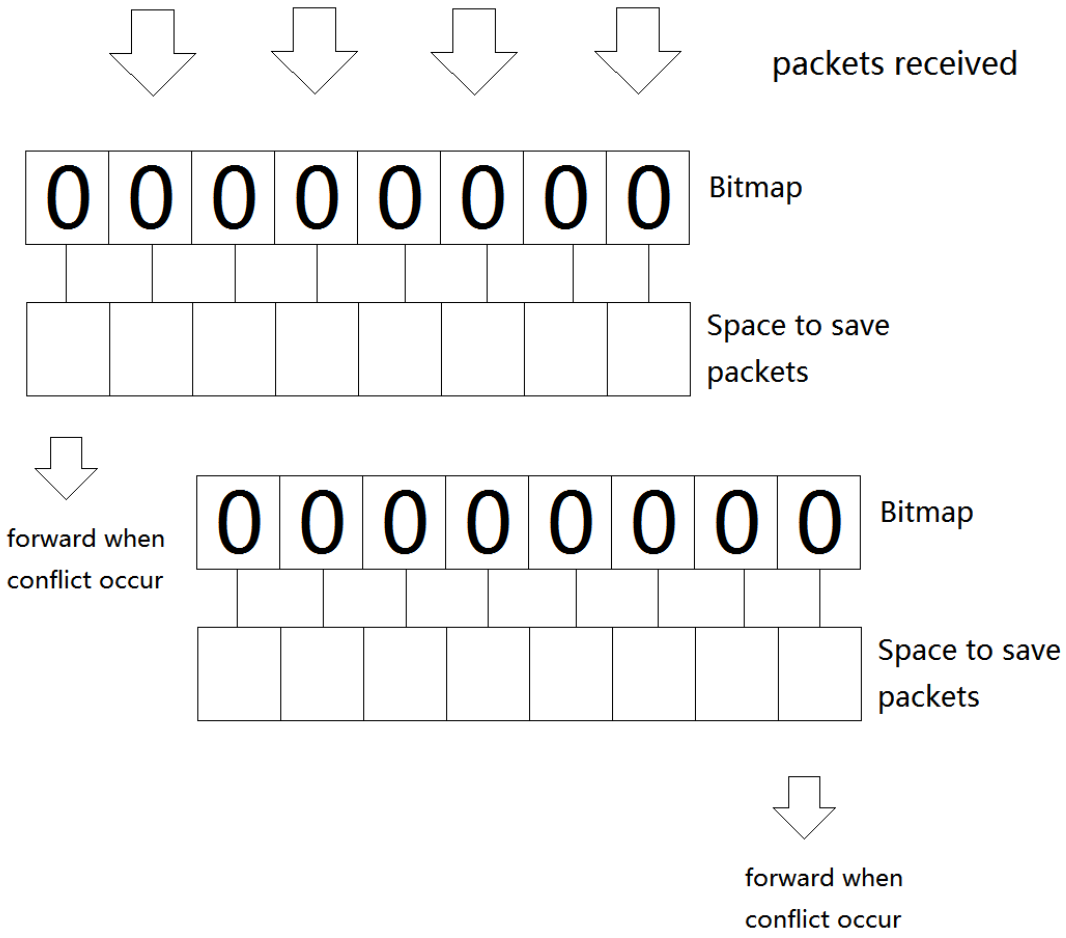

Figure 5.12: Advanced data structure: two bitmaps and two arrays

```
Two Cache Switch packets sort
 1  index ← packet.Seq mod cacheSize
 2  nowLoop ← packet.Seq/cacheSize
 3  if nowLoop == loop and isPacketCached[index] == 0
 4     then cache[index] ← packet
 5          cache[index] = 1
 6          ▷ save the packet in the cache and set the bitmap.
 7  elseif nowLoop > loop
 8     then
 9          if sendSet == ture
10            then cacheWaitToSend.sendOut()
11                 sendSet = false
12                 ▷ send out the second cache.
13          if sendSet == false
14            then cacheWaitToSend[index] ← packet
15                 sendSet = ture
16          loop + +
17          renew(isPacketCached)
18          renew(cache)
19  elseif nowLoop == loop − 1
20     then swap(cache, cacheWaitTosend)
21          cache[index] = 1
```

5.5.4 Redundancy rate adjustment

Because the redundancy packets consume more bandwidth, when network congestion occurs, sending more redundancy packets will make things worse. In other words, we need to adjust the number of redundancy packets based on the sending rate. If the sending rate is high enough, we can increase the number of redundancy packets; if it is not, we need to

decrease our redundancy rate. Figure 5.13 describes this progress. Basically, the sender and receiver will use RTCP packet to tell each other the packet loss rate.

First we need to determine if there are any packets lost or not. If there is no packets loss, we don't need to send any redundancy packets. If the packets loss occur the, we also need check the video streams bit rate, if the bit rate is too low, that means there may have a network congestion (RTCP will help our video stream to be adjusted with the sending rate). In this case, we cannot send more redundancy packets. Besides, if there are no more lost packets, that doesn't mean we will not have packet loss again later, so what we do is decrease the number of redundancy packets step by step instead of stopping sending the redundancy packets once for all.

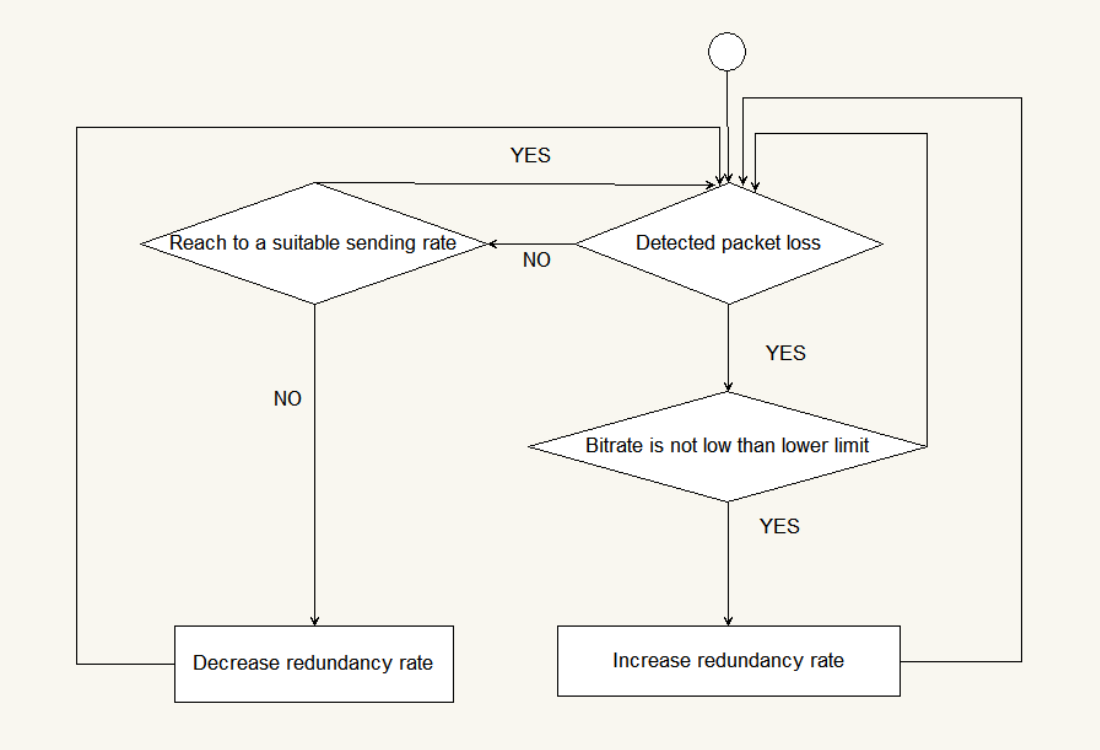

Figure 5.13: Redundancy Adjustment

5.6 Simulator

The main task of the simulator is to randomly drop packets. We need to make sure that each packets have an equal chance to be dropped. To achieve this goal, we first design a function which can generate the random value "drop" or not "drop" as input percentage.

LOSS OR SEND

```
1  loss ← configuredLossPercentage
2  if loss == 0
3      then exit
4  rad = newboolean[10000]
5  for i ← 0 to 10000
6          do
7              if i < loss
8                  then rad[i] = false
9                  else
10                         rad[i] = true
11 index = radom.nextInt(9999)
12 return rad[index]
```

The main idea is to generate some values in proportion, for example, if we want a 10% lost, we generate 1000 false and 9000 true (false represents loss, true represents send). We use a random generator generate 0 9999 equitably, so the probability of generator chose true will be 9000/10000 = 90% and the probability of generator chose false will be 1000/10000 = 10%.

Chapter 6

Experiment

6.0.1 Simulator Test

Before we begin our testing work, we must verify that the simulator is working perfectly. This test is built in the following way: we set up the packet loss rate to $10\%, 30\%, 40\%, 80\%$; at last the simulator give us a stable estimated loss rate. At the same time, we have test our 2 kinds of tunnels in each environment. 6.1 shows the results, we have obtained our proper loss rate. Everything works fine.

6.0.2 Running car Test

In this test, we put our system on a running car, to simulate a real scenario may happen in the VANETs.

Before we start, we need to know what is the maximum distance of the Wi-Fi communication between the 2 tablets we used, because we don't know the power of our tablets' wireless device. So we put the system in a stadium; with the help of the marks on the runway, we can get the general communication distance.

Usually, we can find the 60 meter and 100 meter' mark on the runway. Also, there are some marks on the runway like figure 6.2 shows, which will mark for 5 meters. Another

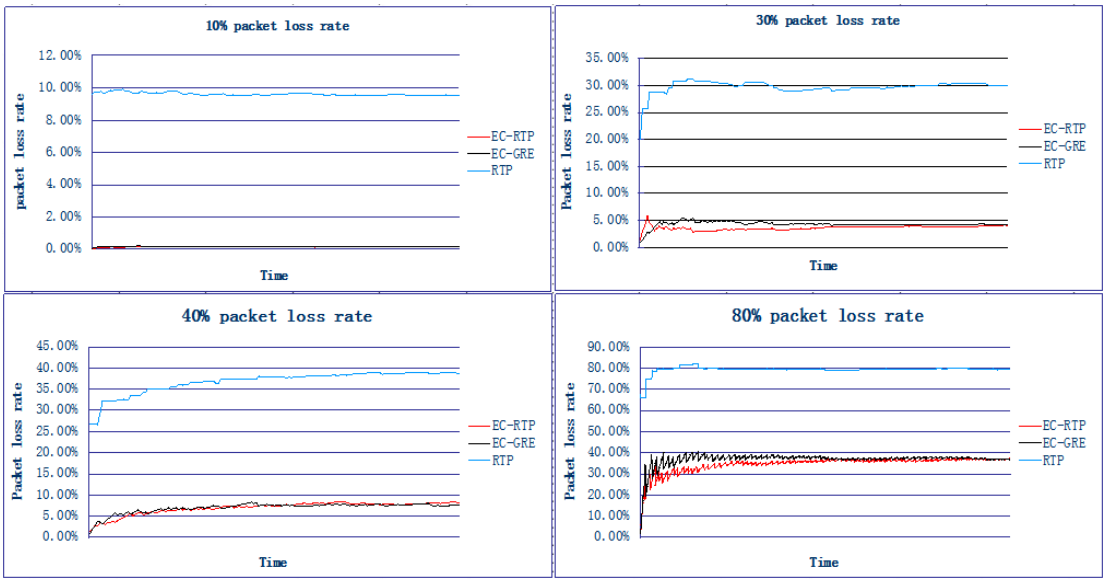

Figure 6.1: Result of testing simulator in different packet loss rate

important factor is signal disturbance. To make sure there will be no other Wi-Fi signal, we use a software to measure the environment, tthe results of which are shown in Figure 6.3.

We have tried to deliver a signal picture to determine the maximum distance between the two tablets. To reduce the test number, we use the binary search approach to let us find the proper distance fast. WThe starting point is 100 meters, which is halfway between 0 and the theoretical maximum distance of the Wi-Fi (200 meters). If it works perfectly, that means the maximum distance is between 100 meters and 200 meters. Otherwise, it means the maximum is between 0 and 100 meters, get the middle value again, and repeat the test.

Table 6.1 shows the result.

After we have finished the test, we know the maximum communicable distance is about 60 meters. Next, we need to find a location that has about 60 meters for our cars. Figure 6.4 shows the test location. The green mark is the starting point, the red mark is the end point.

No.	Distance	Result
1	100 m	Cannot find peers
2	50 m	found peers, connected, transfer completed
3	75 m	Cannot find peers
4	60 m	found peers, connected, transfer completed slowly
5	65 m	found peers, cannot connect

Table 6.1: Test results

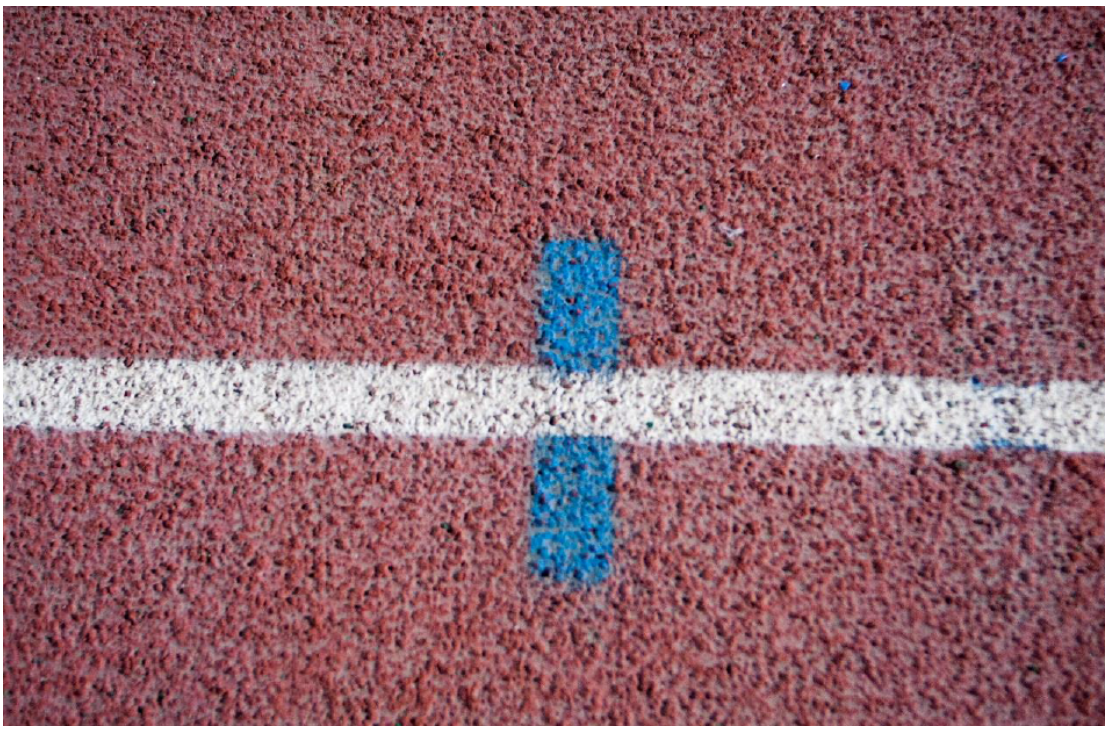

Figure 6.2: 5 meter mark on the runway

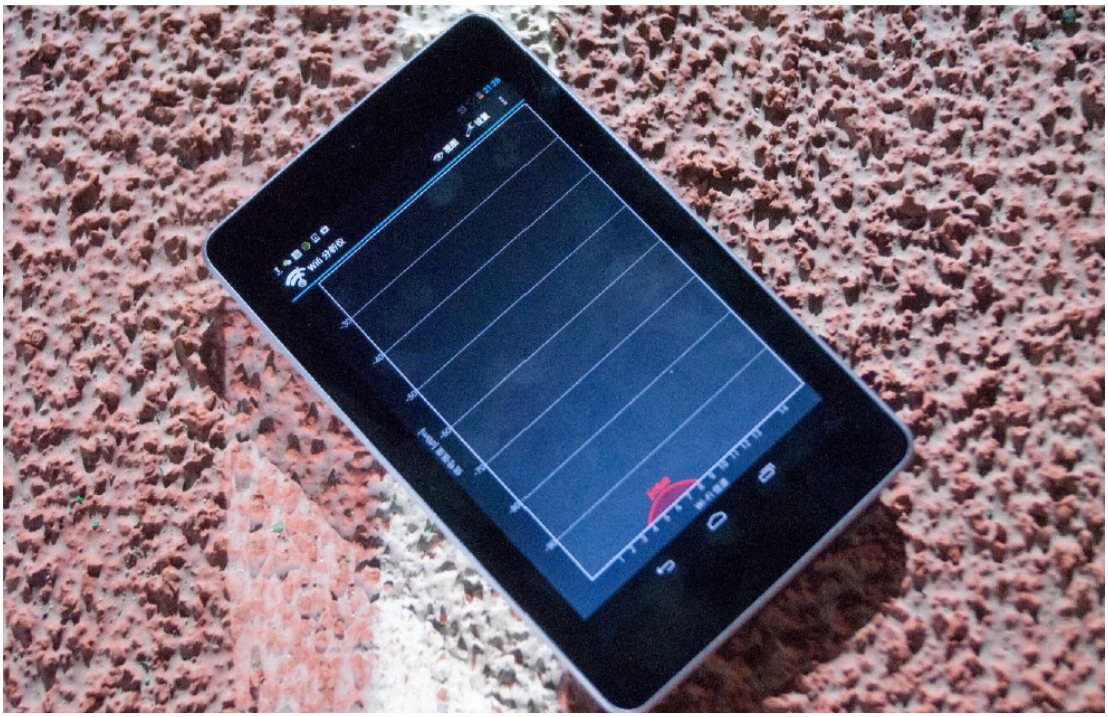

Figure 6.3: The other Wi-Fi signal on the stadium

The simulator and the RTSP server will be set on the car, and the left one will be set on the roadside. We let the car go through the line we have set up. Figure 6.5 shows the final result.

6.0.3 Delay Test

In this test, we want to make sure that our simulator, converter, and tunnel will not cause a lot of delay. The RTP without using any features, will be our control group. We continue to use the same test as we have done in section 6.0.1.

Figure 6.6 shows the final result; when there are a lot of packets lost, the solution we have applied may increase the delay by about 2 seconds. However, when the packet loss rate occurs in a reasonable region, the increment of the delay could be ignored.

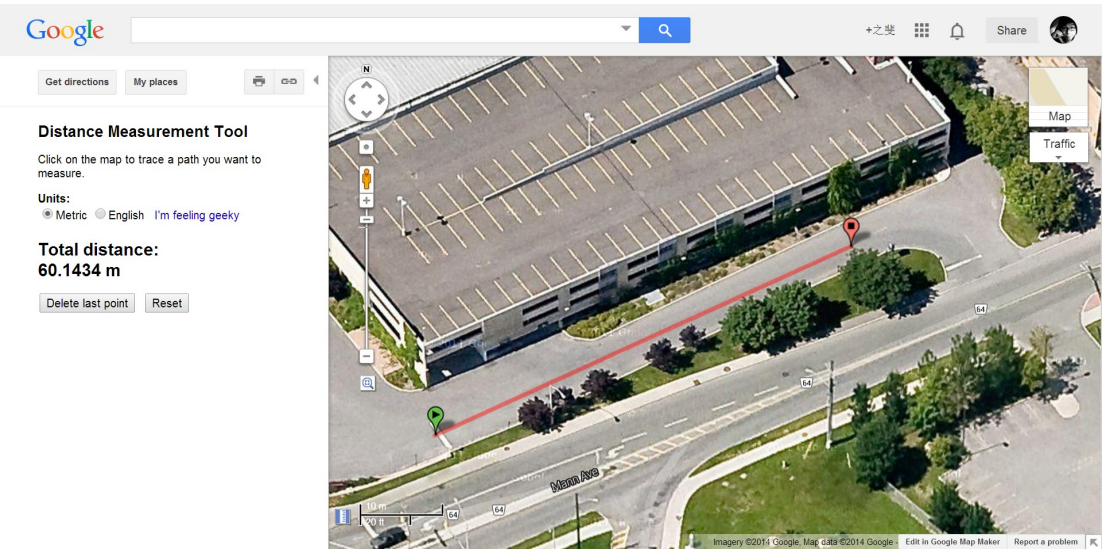

Figure 6.4: A map of the route where ran the tested on

6.0.4 CPU consumption Test

In this test, we want to make sure that our simulator, converter, and tunnel will not consume a lot of resources on the tablet. We continue to use the same test as we done in section 6.0.1. We used a software to record the CPU usage for 1 minute, and we compute the average the usage and the maximum usage.

Figure 6.7 shows the final result. It tells us our algorithm is working fine. Even in the 80% loss case, the maximum usage is not increased more than 20%.

6.0.5 Video quality Test

In this experiment, we set the simulator to the 10% packet loss. We use the file mode so that we can compare the received file and the original file easily.

Peak signal-to-noise ratio

We use Peak signal-to-noise ratio (PSNR) as our video quality measurements. PSNR is the most commonly used to measure the quality of reconstruction of lossy compression codecs.

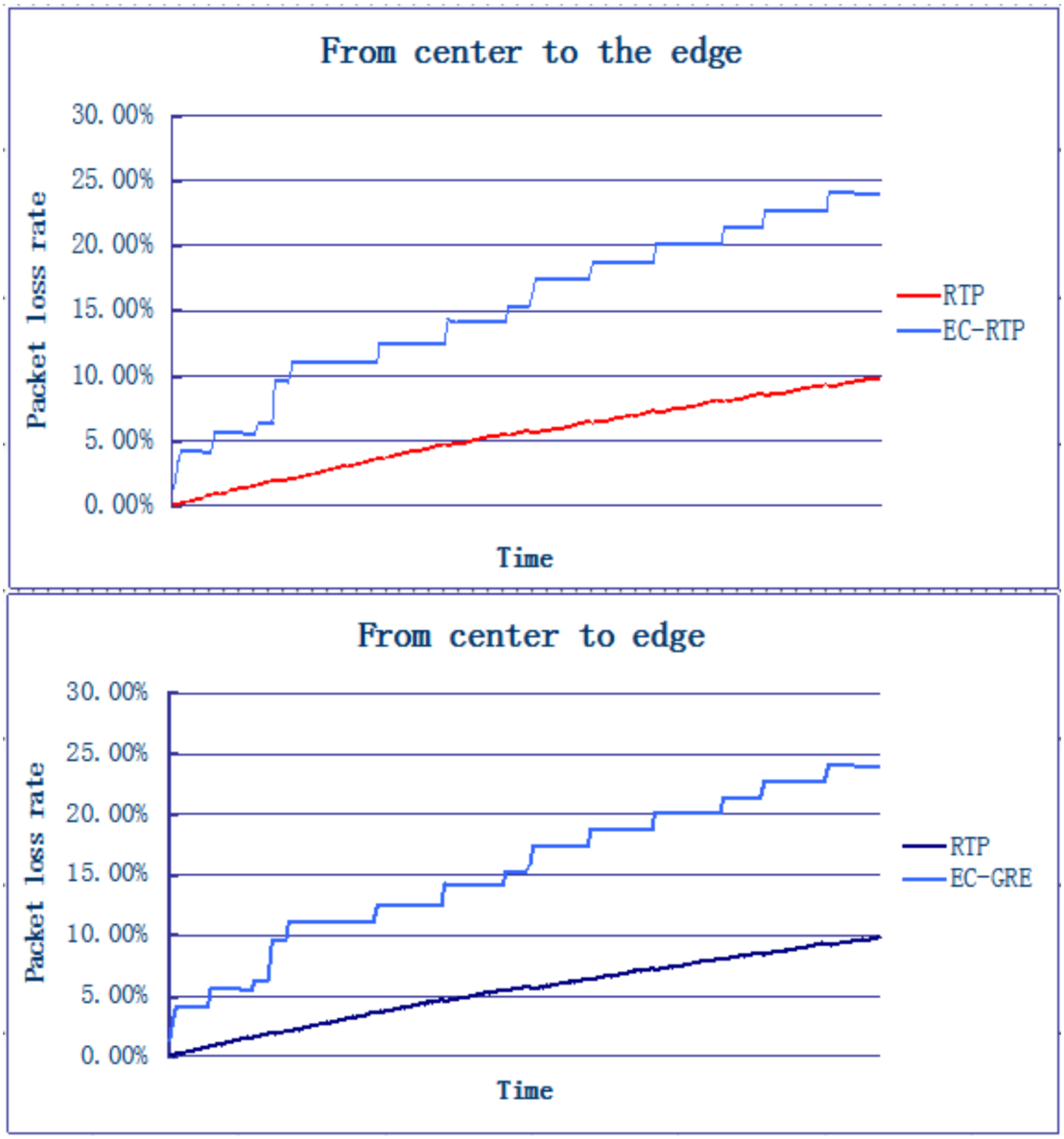

Figure 6.5: Result of testing a player in a car run away from the server

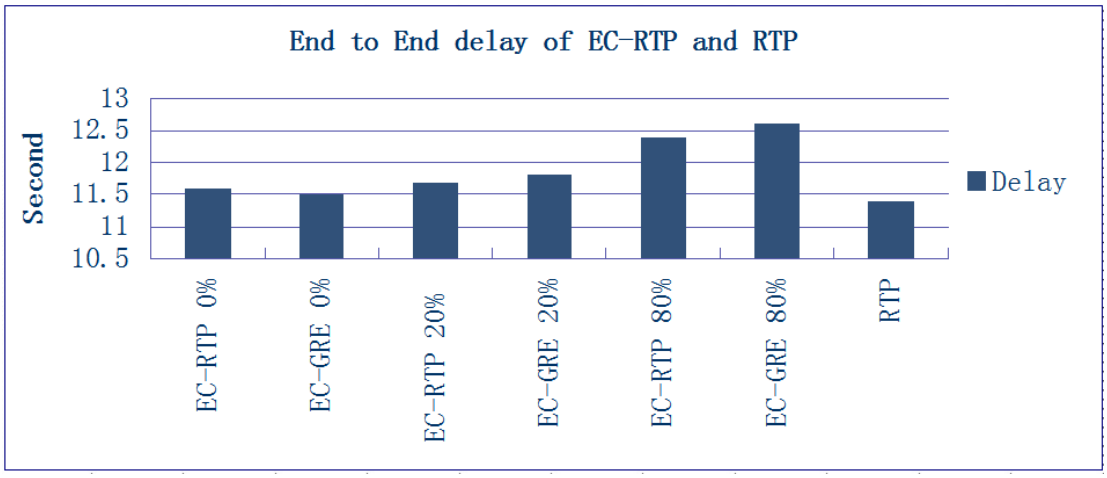

Figure 6.6: Result of testing delay in different packet loss rate

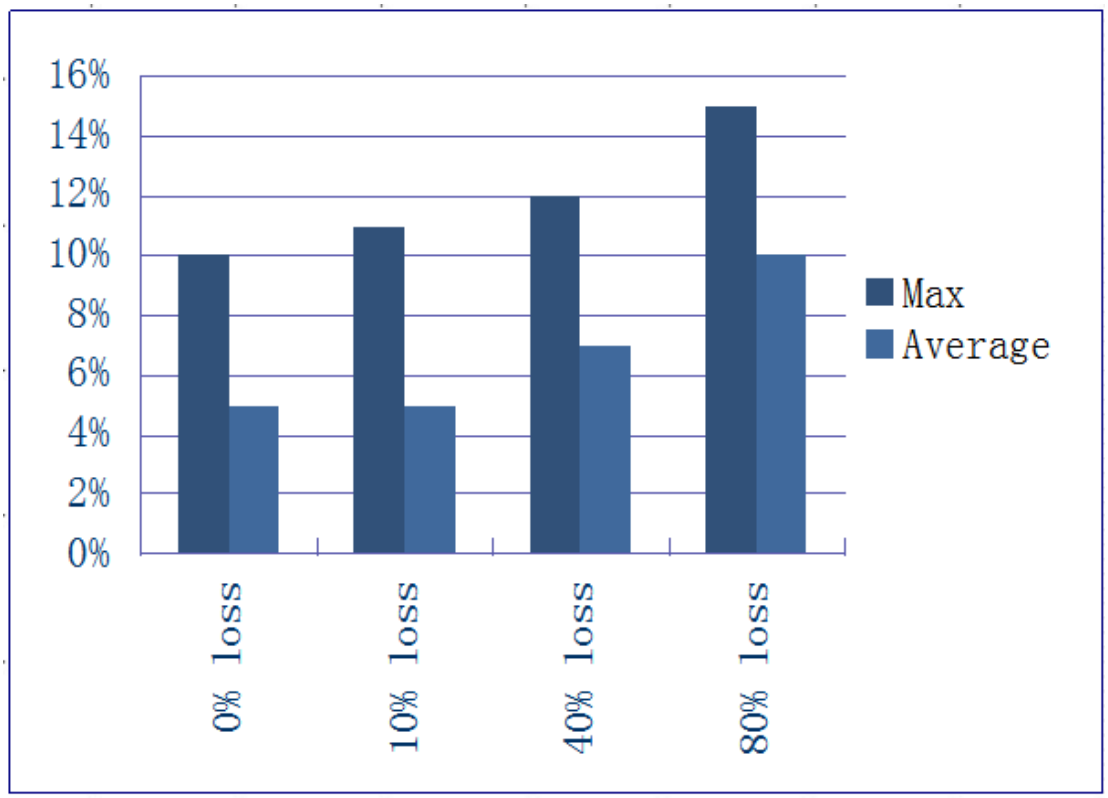

Figure 6.7: Result of testing CPU consumption in different packet loss rate

PSNR is most easily defined via the mean squared error (MSE). Given a noise-free $m*n$ monochrome image I and its noisy approximation K, MSE is defined as 6.1.

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2; \tag{6.1}$$

The PSNR can be defined as 6.2,6.3,6.4.

$$PSNR = 10\log_{10}(\frac{MAX_I^2}{MSE}) \tag{6.2}$$

$$PSNR = 10\log_{10}(\frac{MAX_I}{\sqrt{MSE}}) \tag{6.3}$$

$$PSNR = 20\log_{10}(MAX_I) - 10\log_{10}(MSE) \tag{6.4}$$

According to [45][46], acceptable values for wireless transmission quality loss are considered to be about 20 dB to 25 dB. In the absence of noise, the two images I and K are identical, and thus the MSE is zero. In this case the PSNR is undefined.

Figure 6.8 shows the result that the video quality is increased. We set our system about 10% packet loss rate.

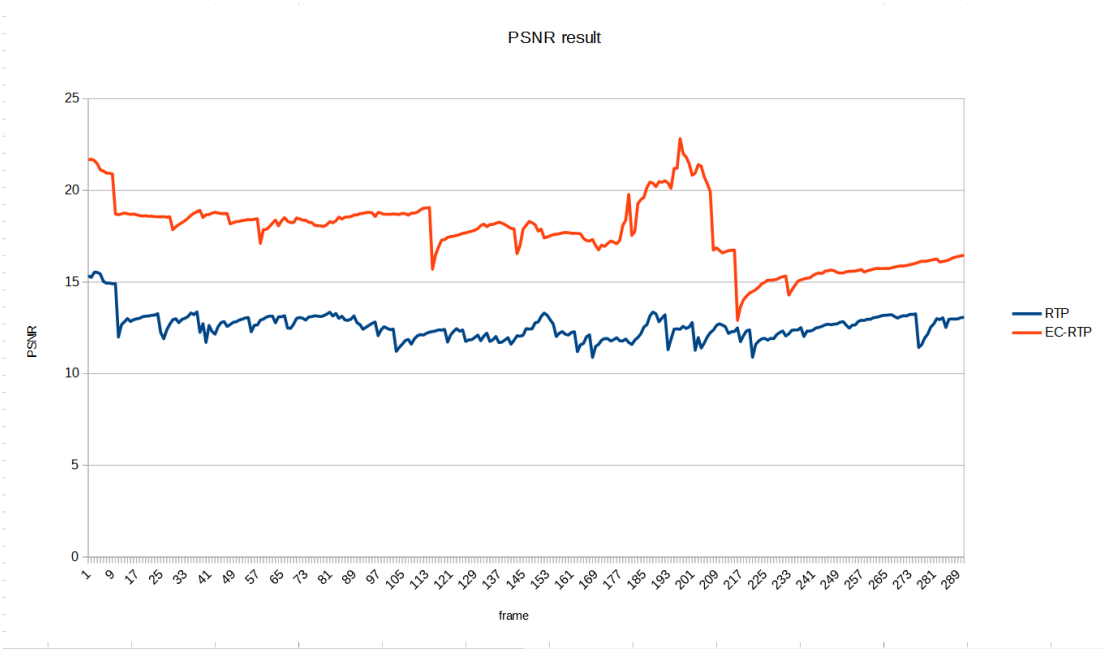

Figure 6.8: Result of testing video quality in 10% packet loss rate

Chapter 7

Conclusion

In our work, we have implemented a RTSP server and a packet loss simulator on the Android system, Which could be a good test platform for the VANETs video streaming. We have also give out 2 models to solve the compatibility problem, make the software design for Internet can perform good result. We use EC technology to solve the high packet loss rate problem, and use the redundancy rate adjustment method to avoid the shortcoming of redundancy technology. At last we do some experiment with our system. The results shows that VANETs could be an extension of Internet, EC technology can provide our better video. Besides, we have design a UDP sorting algorithm, which have high efficient way to sort UDP packets, and consume little resources.

References

[1] Bo Yu and Chengzhong Xu. Vehicular ad-hoc networks: An information-centric perspective. *ZTE Communications*, 8(3), 2010.

[2] Cristiano Rezende, Mohammed Almulla, Azzedine Boukerche, and Antonio A.F. Loureiro. The selective use of redundancy for video streaming over vehicular ad hoc networks. *ACM Transactions on Multimedia Computing, Communications and Applications*, July 2013.

[3] Adam Li. Rfc 5109. *RTP payload format for generic forward error correction*, 2007.

[4] Marco Pasin, Matteo Petracca, Paolo Bucciol, Antonio Servetti, and Juan Carlos De Martin. Error resilient real-time multimedia streaming over vehicular networks. *Vehicle Systems and Safety, Dallas, TX, USA*, 2009.

[5] James Lee Hafner. Weaver codes: Highly fault tolerant erasure codes for storage systems. In *FAST*, volume 5, pages 211–224, 2005.

[6] James S Plank, Jianqiang Luo, Catherine D Schuman, Lihao Xu, Zooko Wilcox-O'Hearn, et al. A performance evaluation and examination of open-source erasure coding libraries for storage. In *FAST*, volume 9, pages 253–265, 2009.

[7] Mingqiang Li, Jiwu Shu, and Weimin Zheng. Grid codes: Strip-based erasure codes with high fault tolerance for storage systems. *ACM Transactions on Storage (TOS)*, 4(4):15, 2009.

[8] Kevin M Greenan, Xiaozhou Li, and Jay J Wylie. Flat xor-based erasure codes in storage systems: Constructions, efficient recovery, and tradeoffs. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–14. IEEE, 2010.

[9] James Lee Hafner. Hover erasure codes for disk arrays. In *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*, pages 217–226. IEEE, 2006.

[10] Cristiano Rezende, Mohammed Almulla, and Azzedine Boukerche. The use of erasure coding for video streaming unicast over vehicular ad hoc networks. In *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, pages 715–718. IEEE, 2013.

[11] Jay J Wylie and Ram Swaminathan. Determining fault tolerance of xor-based erasure codes efficiently. In *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*, pages 206–215. IEEE, 2007.

[12] Johannes Bloemer, Malik Kalfane, Richard Karp, Marek Karpinski, Michael Luby, and David Zuckerman. An xor-based erasure-resilient coding scheme. 1995.

[13] Maxwell N Krohn, Michael J Freedman, and David Mazieres. On-the-fly verification of rateless erasure codes for efficient content distribution. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 226–240. IEEE, 2004.

[14] Michael Luby. Lt codes. 2002.

[15] Richard Karp, Michael Luby, and Amin Shokrollahi. Finite length analysis of lt codes. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, page 39. IEEE, 2004.

[16] S-YR Li, Raymond W Yeung, and Ning Cai. Linear network coding. *Information Theory, IEEE Transactions on*, 49(2):371–381, 2003.

[17] Mohsen Sardari, Faramarz Hendessi, and Faramarz Fekri. Ddrc: Data dissemination in vehicular networks using rateless codes. *Journal of Information Science & Engineering*, 26(3), 2010.

[18] Mohsen Sardari, Faramarz Hendessi, and Faramarz Fekri. Dmrc: dissemination of multimedia in vehicular networks using rateless codes. In *INFOCOM Workshops 2009, IEEE*, pages 1–6. IEEE, 2009.

[19] Rudolf Ahlswede, Ning Cai, S-YR Li, and Raymond W Yeung. Network information flow. *Information Theory, IEEE Transactions on*, 46(4):1204–1216, 2000.

[20] Seung-Hoon Lee, Uichin Lee, Kang-Won Lee, and Mario Gerla. Content distribution in vanets using network coding: the effect of disk i/o and processing o/h. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON'08. 5th Annual IEEE Communications Society Conference on*, pages 117–125. IEEE, 2008.

[21] Uichin Lee, Joon-Sang Park, Joseph Yeh, Giovanni Pau, and Mario Gerla. Code torrent: content distribution using network coding in vanet. In *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, pages 1–5. ACM, 2006.

[22] Shabbir Ahmed and Salil S Kanhere. Vanetcode: network coding to enhance cooperative downloading in vehicular ad-hoc networks. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 527–532. ACM, 2006.

[23] Xingjun Zhang, Cuiping Jing, Feilong Tang, Scott Fowler, Huali Cui, and Xiaoshe Dong. Joint redundant and random network coding for robust video transmission over lossy networks. *Mobile Information Systems*, 8(3):213–230, 2012.

[24] Atsushi Fujimura, Soon Y Oh, and Mario Gerla. Network coding vs. erasure coding: Reliable multicast in ad hoc networks. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7. IEEE, 2008.

[25] Peter Lambert, Wesley De Neve, Yves Dhondt, and Rik Van de Walle. Flexible macroblock ordering in h. 264/avc. *Journal of Visual Communication and Image Representation*, 17(2):358–375, 2006.

[26] N Qadri, Muhammad Altaf, Martin Fleury, Mohammed Ghanbari, and Hanadi Sammak. Robust video streaming over an urban vanet. In *Wireless and Mobile Computing, Networking and Communications, 2009. WIMOB 2009. IEEE International Conference on*, pages 429–434. IEEE, 2009.

[27] N Qadri, Muhammad Altaf, Martin Fleury, and Mohammed Ghanbari. Robust video communication over an urban vanet. *Mobile Information Systems*, 6(3):259–280, 2010.

[28] Abdul Razzaq and Ahmed Mehaoua. Video transport over vanets: Multi-stream coding with multi-path and network coding. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 32–39. IEEE, 2010.

[29] Gorry Fairhurst and Lloyd Wood. Advice to link designers on link automatic repeat request (arq). 2002.

[30] Min Xing and Lin Cai. Adaptive video streaming with inter-vehicle relay for highway vanet scenario. In *IEEE ICC 2012 Wireless Networks Symposium*, Dept of ECE, University of Victoria, BC, Canada, 2012.

[31] Charles Krasic, Jonathan Walpole, and Wu-chi Feng. Quality-adaptive media streaming by priority drop. In *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 112–121. ACM, 2003.

[32] Hsuan-Fu Ho, Kuochen Wang, and Yi-Ling Hsieh. Resilient video streaming for urban vanets. In *Proc. 7th Workshop on Wireless Ad Hoc and Sensor Networks*, 2011.

[33] Daniel Jiang and Luca Delgrossi. Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 2036–2040. IEEE, 2008.

[34] Stephan Eichler. Performance evaluation of the ieee 802.11 p wave communication standard. In *Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th*, pages 2199–2203. IEEE, 2007.

[35] Mahdi Asefi, Jon W Mark, and Xuemin Shen. A mobility-aware and quality-driven retransmission limit adaptation scheme for video streaming over vanets. *Wireless Communications, IEEE Transactions on*, 11(5):1817–1827, 2012.

[36] Pablo Piol, A. Torres, O. Lopez, M. Martinez, and Manuel P. Malumbres. Evaluating hevc video delivery in vanet scenarios. In *Wireless Days*, pages 1–6. IEEE, 2013. URL `http://dblp.uni-trier.de/db/conf/wd/wd2013.html#PinolTLMM13`.

[37] H Schulzrinne, S Casner, R Frederick, and V Jacobson. Rfc 3550. *RTP: a transport protocol for real-time applications*, 7, 2003.

[38] Y.-K. Wang, R. Even, Huawei Technologies, T. Kristensen, and R. Jesup. Rfc 6184. *RTP Payload Format for H.264 Video*, May 2011.

[39] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Rfc 2784. *Generic Routing Encapsulation (GRE)*, April 2000.

[40] Tim Berners-Lee, Larry Masinter, and Mark McCahill. Rfc 1738: Uniform resource locator. *Internet Engineering Task Force*, 1994.

[41] Google. *URL class Manual*, . `http://developer.android.com/reference/java/net/URL.html`.

[42] H. Schulzrinne and R. Lanphier A. Rao. Rfc 2326. *Real Time Streaming Protocol (RTSP)*, April 1998.

[43] Google. *VideoView class Manual*, . `http://developer.android.com/reference/android/widget/VideoView.html`.

[44] Jon Bentley. *Programming Pearls*. Dorling Kindersley Pvt Ltd, 2006.

[45] Nikolaos Thomos, Nikolaos V Boulgouris, and Michael G Strintzis. Optimized transmission of jpeg2000 streams over wireless channels. *Image Processing, IEEE Transactions on*, 15(1):54–67, 2006.

[46] Xiangjun Li and Jianfei Cai. Robust transmission of jpeg2000 encoded images over packet loss channels. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 947–950, 2007.

[47] Dan Wing. Rfc 3605. *Symmetric RTP/RTP Control Protocol (RTCP)*, October 2007.

[48] G. Dommety. Rfc 2890. *Key and Sequence Number Extensions to GRE*, September 2000.

[49] Mark Handley, Colin Perkins, and Van Jacobson. Rfc 4566. *SDP: session description protocol*, July 2006.

[50] Hao Jiang, Siyue Chen, Yang Yang, Zhizhong Jie, Henry Leung, Jun Xu, and Lin Wang. Estimation of packet loss rate at wireless link of vanet–rple. In *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, pages 1–5. IEEE, 2010.

[51] Jari Korhonen and Ye Wang. Effect of packet size on loss rate and delay in wireless links. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 3, pages 1608–1613. IEEE, 2005.

[52] Yung-Cheng Chu and Nen-Fu Huang. Delivering of live video streaming for vehicular communication using peer-to-peer approach. In *2007 Mobile Networking for Vehicular Environments*, pages 1–6. IEEE, 2007.

[53] Zhe Wang and Mahbub Hassan. Blind xor: low-overhead loss recovery for vehicular safety communications. *Vehicular Technology, IEEE Transactions on*, 61(1):35–45, 2012.

[54] Fei Xie, Kien A Hua, Wenjing Wang, and Yao Hua Ho. Performance study of live video streaming over highway vehicular ad hoc networks. In *Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th*, pages 2121–2125. IEEE, 2007.

[55] Quan Huynh-Thu and Mohammed Ghanbari. The accuracy of psnr in predicting video quality for different video scenes and frame rates. *Telecommunication Systems*, 49(1): 35–48, 2012.

[56] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2366–2369. IEEE, 2010.

[57] Zhou Wang, Alan C Bovik, and Ligang Lu. Why is image quality assessment so difficult? In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 4, pages IV–3313. IEEE, 2002.

[58] Quan Huynh-Thu and Mohammed Ghanbari. Scope of validity of psnr in image/video quality assessment. *Electronics letters*, 44(13):800–801, 2008.

[59] Cisco. *Implementing Tunnels*. `http://www.cisco.com/c/en/us/td/docs/ios/12_4/interface/configuration/guide/inb_tun.html`.

[60] Netpbm. *pnmpsnr User Manual*, March 2001. `http://netpbm.sourceforge.net/doc/pnmpsnr.html`.

[61] Apple. *What is Quicktime 7?* `http://www.apple.com/ca/quicktime/what-is/`.

[62] Qiben Yan, Ming Li, Zhenyu Yang, Wenjing Lou, and Hongqiang Zhai. Throughput analysis of cooperative mobile content distribution in vehicular network using symbol level network coding. *Selected Areas in Communications, IEEE Journal on*, 30(2): 484–492, 2012.

[63] Kayhan Zrar Ghafoor, Kamalrulnizam Abu Bakar, Zaitul Marlizawati Zainuddin, Chih-Heng Ke, and Alberto J Gonzalez. Reliable video geocasting over vehicular ad hoc networks. *Adhoc & Sensor Wireless Networks*, 15, 2012.