Vehicular communication: protocol design, testbed implementation and performance analysis

Conference Paper · January 2009
DOI: 10.1145/1582379.1582469 · Source: DBLP

Sumaiya Iqbal
Bangladesh University of Engineering and Technology

Shihabur Rahman Chowdhury
University of Waterloo

Athanasios Vasilakos

Cheng-Xiang Wang
Heriot-Watt University

Vehicular Communication: Protocol Design, Testbed Implementation and Performance Analysis

Sumaiya Iqbal[1], Shihabur Rahman Chowdhury[1], Chowdhury Sayeed Hyder[1], Athanasios V. Vasilakos[2] and Cheng-Xiang Wang[3]
[1] Dept. of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Bangladesh
[2] Dept. of Computer and Telecommunication Engineering
University of Western Macedonia, Greece
[3] School of Engineering and Physical Science
Heriot-Watt University, UK
sumaiya_iqbal@yahoo.com, shihab.buet@gmail.com, sayeed@csebuet.org
vasilako@ath.forthnet.gr, cheng-xiang.wang@hw.ac.uk

ABSTRACT

Vehicular Communication Networks and Systems (VCNS) and Intelligent Transportation Systems (ITS) are one of the most attractive and challenging topics in recent days since a well efficient protocol for vehicular communication can facilitate the reduction of traffic congestion and can provide us with many more promising applications. In this paper, we propose a protocol for vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communication. As one of the challenging parts of this paper, we present an experimental testbed in which two major applications of V2I & V2V communication (i.e. traffic congestion detection and emergency warning) is implemented. Based on careful analysis, we also calculate some key system parameters which reflect the efficiency of the protocol in different applications.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Wireless communication*; C.2.2 [**Computer Communication Networks**]: Network Protocols

General Terms

Design, Experimentation, Performance.

Keywords

Vehicular communication, message collision, data recovery, testbed, congestion, emergency warning.

1. INTRODUCTION

VCNS is an emerging type of wireless network in which vehicles and roadside units are the communicating nodes; providing each other with information, such as safety warnings and traffic information. *VCNS* is different from other networks in the sense that the nodes are moving here. Mobility of the nodes creates various problems along with the problems of wireless network. Despite all these limitations, *VCNS* provides various services, for example, eliminating traffic collisions, ensuring smooth flow of vehicles in the highways, reducing traffic congestion, enabling a plethora of new applications such as mobile infotainment etc. This makes *VCNS* one of the most challenging topics for research and development in recent days. We are moved by these facts and try to put some contributions in this field.

The paper is organized as follows. In section 2, we discuss some previous works in this field. Section 3 gives an architectural overview of our system. Overview of our proposed protocol is discussed in section 4 and the implementation details are described in section 5. In Section 6, we present the experimental results. We also shed some lights on future works in section 7.

2. PREVIOUS WORKS

Since V2V and V2I communication has numerous applications in our real life, so it has been already studied extensively. Some researchers have proposed different protocols for communication. A channel access protocol for inter-vehicle communication was introduced in [9]. In this protocol channel is allocated to the mobile/stationary nodes based on their instantaneous geographical location. Authors in [10] introduced a new information propagation scheme for vehicular networks which makes use of attribute based data from MANET methodologies. Agarwal *et al.* contributed in upstream message propagation in vehicular Ad Hoc networks [1].

Various kinds of applications have also been shown in previous days. A multi-hop wireless parking meter network (PMNET) to quickly find available parking space was proposed by researchers in [6]. Efficient protocol for secured vehicular communication and its impact on transport safety has been studied in recent days [7, 8]. Vehicle manufacturers are also showing their interest in *VCNS*. In order to improve driving safety, traffic organization and easy hotspot connec-

IWCMC'09, June 21-24, 2009, Leipzig, Germany

tions, six European car manufacturers founded the CAR 2 CAR Communication Consortium. Its goal is to create a European industrial standard for future communicating cars spanning all brands.

All these previous works regarding V2I and V2V communication have mainly focused on either theoretical aspects of communication protocol or on a particular application (mainly car parking). We do not limit our work just in developing a protocol; rather we design a testbed where we make real implementation of two important application scenarios (congestion detection and emergency warning). We also evaluate the performance of our implementation.

3. SYSTEM ARCHITECTURE

Our developed system consists of two modules:
1. Road Side Station Module(RSSM)
2. Vehicle Module(VM)

Road Side Station Module: *RSSM* is a static infrastructure and acts as a local server for a specific area. It communicates with the vehicles within its radio range. It has two parts: i)Broadcaster(BRD) ii)Responder(RSP)

Broadcaster is responsible for broadcasting necessary information to all the vehicles in the range of that *RSSM*.
Responder is responsible for interacting with the vehicles (sends and receives necessary information) and also with the *BRD* to request it to broadcast any necessary information which is provided to it by *RSP*. It also stores information received from the vehicles for future decision making.

Vehicle Module: *VM* is embedded within the vehicles and so acts as moving nodes. It is responsible for communicating with other vehicles and *RSSM* and displays various information received from them.

Each part of every module in our system has its own ***Transceiver Unit(TU)*** and ***Control Unit(CU)*** . *TU* is responsible for wireless communication and *CU* is responsible for controlling the overall operation. So each module of the system is independent of each other and performs their specific job. The block diagram of our system architecture is shown in Figure 1.

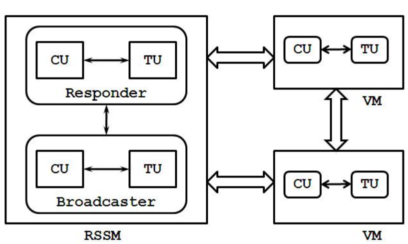

Figure 1: Block Diagram of System

4. COMMUNICATION PROTOCOL

In this section, we propose *Dual Channel Vehicular Communication(**DCVC**)* Protocol for V2V and V2I communication. All communication is performed in two separate channels(*ch1 & ch2*). One channel is assigned only for broadcasting and another channel for other message transmission. This reduces message collision and increases reliability of communication.

In our protocol, we devise an effective method for detecting data loss as well as for data recovery, which uses the *Binary Exponential Back off(**BEB**)* Algorithm [2]. Redundant data transmission is also avoided whenever it is possible.

4.1 Framing

Framing of data is necessary for synchronization, error control, flow control etc. There are various methods of framing. Of them, *Starting and ending flags, with bit stuffing* is used here. The frame structure is shown in Figure 2.

Starting Flag	Control	Error Checking Code	Type	Data	Ending Flag

Figure 2: Frame Structure

The ***Starting Flag*** field contains a special bit pattern which is used to indicate the start of a new frame.
The ***Control*** field is used for containing any pattern/number needed for synchronization.
The ***Error Checking Code*** is used for error detection.
The ***Type*** field identifies different types of frame.
The ***Data*** field contains any information that is needed to be communicated.
The ***Ending Flag*** field contains the same special bit pattern to indicate the end of a frame.

4.2 Communication Rules

Each *RSSM* and *VM* is assigned a unique number as their ID(***sID*** and ***vID***) respectively. The *BRD* broadcasts its *RSSM*'s *sID* at regular intervals in *ch 1* and the *RSP* always listens at *ch 2*. When a vehicle enters into the range of a *RSSM*, the *VM* receives the *sID* of that station. In response, *VM* sends the vehicle's *vID* in *ch 2* which is received by that *RSSM*'s *RSP*. Then for ensuring the proper communication between the *RSSM* and *VM*, *RSP* sends back an acknowledgement(ACK) to that vehicle in *ch 2*, which contains that vehicle's *vID* .

Now, there are some special aspects of our protocol for improving its performance.

Since the *RSSM* always broadcasts its *sID* at regular intervals, it is possible that a vehicle within the range of a *RSSM* can get the same *sID* for multiple times. For this, the *VM* always saves the received *sID*. When it receives a *sID*, it compares it with the saved *sID*. If no match is found, the *VM* sends its *vID* and replaces the saved *sID* with the new one. Otherwise it does not transmit anything which reduces redundant data transmission.

Now, if multiple vehicles send their *vID* at the same time, at most one *vID* will be received by the *RSP*. Then, it sends an ACK containing the *vID* of the vehicle whose *vID* is received by it. So, only one vehicle gets proper ACK and other vehicles receive ACK which does not contain their *vID*. This is how other vehicles detect that a message collision has occurred and their data has been lost. Then for data recovery, they retransmit their *vID* after waiting a random time(in between 0 to $2^i - 1$ ms for ith attempt) according to the *BEB Algorithm* [2]. The vehicles keep retransmitting until a correct ACK is received or the upper limit of the *BEB Algorithm* is reached.

To keep track of the vehicles within its range, the *RSP* always maintains a list of vehicles as well as the total count of vehicles(Vehicle Count) that are currently within its range. If the *RSSM* is down for a while then the information is needed to be updated after it is powered up. And this information is also needed to be refreshed regularly(necessary to track the leaving of a vehicle from the range of a *RSSM*). For these purposes, the *BRD* broadcasts a special request when it is turned on and at regular intervals. When a *VM* receives this request, it must send its *vID* to the *RSSM* in any condition.

When a vehicle faces any emergency(e.g. accident, road blockage), it sends an emergency notification(*EN*) to *RSSM* in *ch 2*. Then, the *RSP* sends back an ACK to that vehicle using the same channel. The vehicle also broadcasts *EN* in *ch 1* for a fixed *ENC(Emergency Notification Counter)* number of times. *ENC* is set to a value assumed to be enough to maximize the number of successfully notified vehicles and also to prevent the delay from being very high. The *EN* contains the *vID* of the vehicle in emergency and its current location (*sID* of the *RSSM*). The vehicles receiving the *EN* then forward it using *ch 1* to their neighboring vehicles. To prevent *EN* being forwarded infinitely, it contains a value as *TTL (Time To Live)* which is decremented in each hop and when it becomes zero that *EN* is discarded . To prevent duplicate *EN* to be forwarded during this process, each *VM* maintains a list of *vID*s received by an *EN*. This list is reset in regular short intervals. When a *VM* receives an *EN* it looks up the *vID* within it in the list. If a match is found then the message is considered to be a duplicate one which is then discarded.

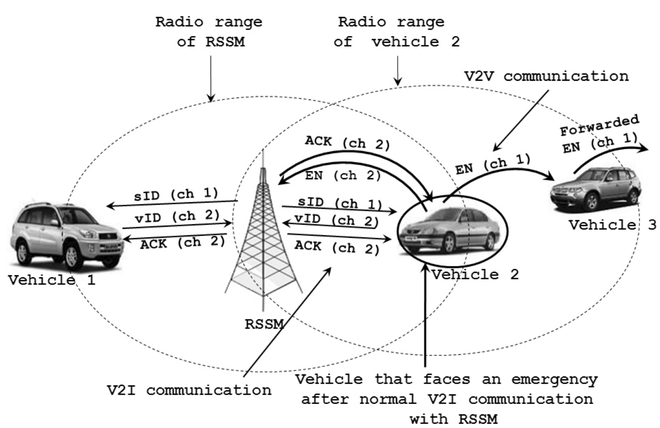

Figure 3: Overview of Communication

5. EXPERIMENTAL TESTBED

Our testbed has one *RSSM* and three *VM*s. Without loss of generality, we assume that this model of one *RSSM* and three *VM*s simulates a real life scenario.

5.1 Hardware Implementation

The hardware components used in our system are: a) AVR ATmega8 microcontroller b) Flash writer c) RFM12 FSK Transceiver d) 7805 Linear voltage regulator e) Antenna etc. Details of these components can be found in [3, 5, 4].

We have used radio frequency as physical medium for wireless communication.

The key component of the *CU* is an ATmega8 microcontroller. Two interfaces of the microcontroller are used: *SPI*(Serial Peripheral Interface) and *USART*(Universal Synchronous Asynchronous Receiver Transmitter) [3]. The flash writer is used to write programs to ATmega8's ROM.

The key component of *TU* is RFM12 FSK (Frequency Shift Keying) radio transceiver. For transmitting data the internal *TX register* and for receiving data built in *FIFO register* of the RFM12 module is used. The end of transmission is checked by the *RGIT* bit of the status register and the completion of data reception is checked by the *FFIT* pin of RFM12 module. A 17.6 cm jumper wire is used as RFM12 module's antenna. A range of up to 100 m(radius) is achieved by this antenna [5]. A 7805 linear voltage regulator is used to prevent any voltage glitch that can harm the performance of RFM12 module.

We have used 434MHz and 439.75MHz frequencies of the 433MHz free ISM band as *ch 1 & ch 2* respectively. All the data transmission has been performed in 4800bps.

5.1.1 Road Side Station Module

BRD is equipped with a radio transceiver(RFM12 module) and a microcontroller(ATmega8) and they are connected through *SPI* of ATmega8.

RSP is equipped with similar components as the *BRD*. In addition, a 7-segment display is also connected with the microcontroller for showing various information.

The internal communication between *BRD* and *RSP* is carried out using *strobe i/o* and *USART*.

The block diagram of *RSSM* is shown in Figures 4.

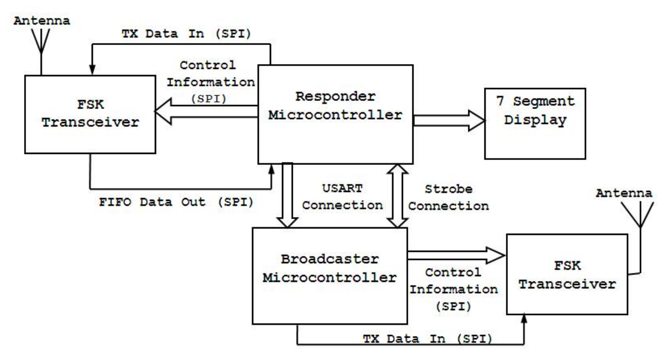

Figure 4: Block Diagram of RSSM

5.1.2 Vehicle Module

VM is equipped with a radio transceiver(RFM12 module), a microcontroller(ATmega8) which are connected through *SPI*. Here, the microcontroller is also connected with two 7 segment displays for showing various information and a switch to simulate different emergency events(e.g. accident, road blockage).

The block diagram of the *VM* can be found in Figure 5. The complete snapshot of *RSSM* and *VM* in our testbed is shown in Figure 6.

5.2 Implementation Details

Here, we describe some implementation specific issues which are used in our experimental testbed.

The format of the frame in our implementation follows the basic frame structure proposed in the ***DCVC*** protocol described in section 4. The frame format is given in Figure 7.

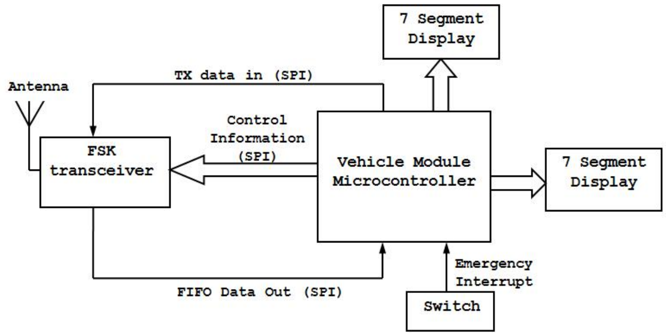

Figure 5: Block Diagram of VM

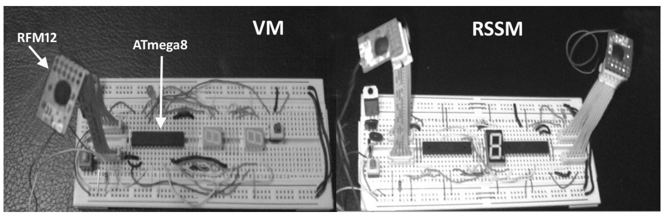

Figure 6: Complete RSSM and VM module

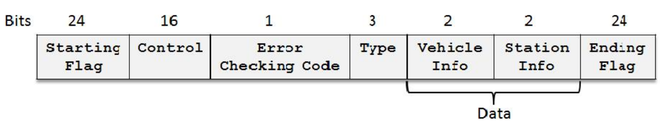

Figure 7: Frame Format for Implementation

Starting Flag and **Ending Flag:** A 24bit special bit pattern "`0xAAAAAA`" is used as the starting and ending flag. This is a requirement for the RFM12 module [5].
Control : According to the requirement of RFM12 module, a special bit pattern "`0x2DD4`" is contained in the control field which must precede the information within the frame for synchronization purposes [5].
Error Checking Code : This field contains an even parity bit which is calculated based on the bits of Type and Data field for error checking.
Type: There are 5 types of frames in our implementation which is identified by the Type field of the frame.

- **Type 1[000] - Station Broadcast(SB) frame:** This frame is used to broadcast *sID* of *RSSM*. *RSSM* broadcasts it once in every 0.1s.
- **Type 2[001] - Vehicle Response(VR) frame:** This frame is used by *VM* to send its *vID* to *RSSM*.
- **Type 3[010] - Acknowledgement(ACK) frame:** *RSSM* sends acknowledgement to *VM* using this frame.
- **Type 4[011] - Emergency Notification(EN) frame:** *VM* sends *EN* message to *RSSM* and other vehicles within its range using this frame.
- **Type 5[100] - Station On(SO) frame:** This frame is broadcasted by *RSSM* when it powers up and also after every 5s to refresh its stored information.

Data: Data field is divided into two subfields: i)Vehicle Info ii)Station Info. Each contains different information according to frame type which listed in Table 1.

Frame Type	Vehicle Info	Station Info
SB	Vehicle Count	sID
VR	vID	sID
ACK	vID	sID
EN	vID	sID
SO	Vehicle Count	sID

Table 1: Contents of Data field

6. EXPERIMENTAL RESULTS

In this section, we perform some tests to evaluate the efficiency of our protocol. Here, we consider two popular applications - congestion detection and emergency notification. During these tests we measure some time intervals as key parameters. For this, the modules are interfaced with a computer using parallel port and necessary software is written in *Java*.

6.1 Application 1 : Congestion Detection

We have used the number of vehicles to approximate the congestion status of an area. Whenever a vehicle enters into the range of a *RSSM*, it is immediately informed about the Vehicle Count of that area. So the driver can take decision whether to go through that area or avoid it.

6.1.1 Implementation Issues

When a vehicle enters into the range of a *RSSM*, the *RSSM* and *VM* communicates according to the ***DCVC*** protocol described in section 4.2. The vehicle list is maintained by the *RSP*. The *BRD* gets the Vehicle Count from the *RSP* by serial communication over *USART* interface of ATmega8. This Vehicle Count is broadcasted with each *SB* and *SO* frame in every 0.1s and 5s respectively. After broadcasting the *SO* frame, the *BRD* notifies *RSP* to reset the vehicle list through *strobe i/o*. So a vehicle entering the range of a *RSSM* gets the the current Vehicle Count in that area which is shown in the display attached with the vehicle.

6.1.2 Performance Analysis

To evaluate the protocol performance, we define some system parameters and perform some tests under different conditions to measure them for our experimental setup.

$\mathbf{RT}_{rssm}$(*Response time for updating the Vehicle Count in RSSM when a new vehicle enters its range*): It indicates the time required for the *RSSM* to track a new vehicle within its range.

	Condition (No. of *VM*s)	No. of Test Cases	$\mathrm{RT}_{rssm(avg)}$ (ms)	Increase of $\mathrm{RT}_{rssm(avg)}$ (%)
a)	1	15	168.8667	–
b)	2	19	189.2941	12.1
c)	3	20	208.75	10.28

Table 2: Result Summary for $\mathbf{RT}_{rssm}$

$\mathbf{RT}_{vm}$(*Response Time for updating the Vehicle Count in VM when a new vehicle enters into the range of RSSM*): It indicates the time required for other vehicles to track a new vehicle into the range of the *RSSM*.

	Condition	No. of Test Cases	$RT_{vm(avg)}$ (ms)
a)	3 vehicle in system	20	219.6

Table 3: Result Summary for RT_{vm}

Table 2 and 3 summarize the result for RT_{rssm} and RT_{vm} respectively. Note that, with the increase of vehicles $RT_{rssm(avg)}$'s rate of increase is diminishing. And also $RT_{vm(avg)}$ for same condition(3 vehicles in system) is greater than that of $RT_{rssm(avg)}$. This is due to the rule of our protocol that information about a new vehicle is received first by *RSSM* and then it is broadcasted to the *VM*s. We also plot the $RT_{rssm(avg)}$ for different number of vehicles in our system in Figure 8. Since, we have three *VM*s in our testbed we use ***Logarithmic Interpolation*** to determine the trend of data for 100 *VM*s.

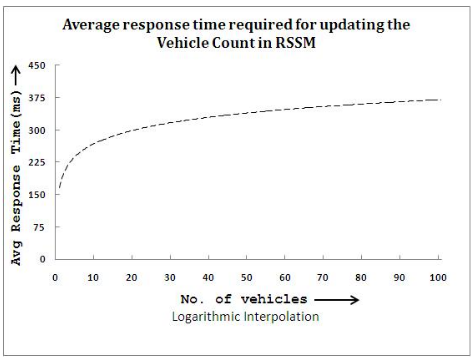

Figure 8: $RT_{rssm(avg)}$ vs. No. of vehicles

6.2 Application 2: Emergency Warning

In our system, whenever a vehicle encounters an accident it immediately sends this information to the *RSSM* and also broadcasts this message to all other vehicles in its range.

6.2.1 Implementation Issues

Without loss of generality, we assume that whenever an emergency event occurs the detection mechanism of the vehicle generates a high voltage level to inform the *VM* about it. Here, we use a switch to simulate an emergency event.

When a vehicle faces an emergency, it takes necessary steps described in section 4.2. In our testbed we have set the value of *ENC* to 20. Due to hardware limitation, the vehicle in emergency broadcasts *EN* but further forwarding by the vehicles that receive that *EN* is not implemented. In our testbed this EN is shown in the displays by the letter "C" along with the *vID* of the vehicle that faced the accident and *sID* of the station to indicate the area where the accident occurred.

6.2.2 Performance Analysis

For this application, we also perform some analysis to calculate some system parameters that show the performance of our protocol.

$RTEN_{rssm}$(*Response time for receiving EN at RSSM*): It indicates the time required for the *RSSM* to track an emergency event within its range(V2I communication).

	Condition	No. of Test Cases	$RTEN_{rssm(avg)}$ (ms)
a)	1 vehicle in system	21	110.1429
b)	2 vehicles in system	20	388.7
c)	3 vehicles in system	15	634.2667

Table 4: Result Summary for $RTEN_{rssm}$

$RTEN_{vm}$(*Response Time for receiving EN at vehicles*): It indicates the amount of time *VM*'s need to track an emergency event within its range (V2V communication).

	Condition	No. of Test Cases	$RTEN_{vm(avg)}$ (ms)
a)	2 vehicle in system	22	308.8182
b)	3 vehicles in system	22	425.1429

Table 5: Result Summary for $RTEN_{vm}$

Table 4 and 5 summarize the result for $RTEN_{rssm}$ and $RTEN_{vm}$ respectively. However, the $RTEN_{rssm(avg)}$ and $RTEN_{vm(avg)}$ could also be plotted in a similar way as $RT_{rssm(avg)}$ in Figure 8 for different number of vehicles.

We implemented both the applications on our testbed. Some snapshots of our work is shown in Figure 9.

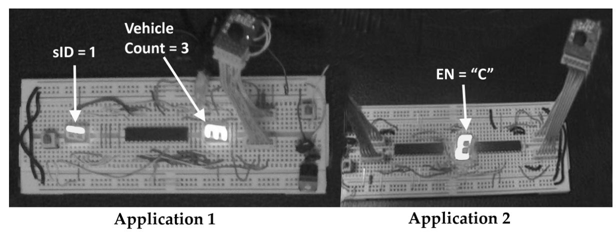

Figure 9: Results of Experiment

6.3 Frame Transmission Rate

Here, we find out *Successful Frame Transmission Rate **(SFTR)*** for V2I communication using the ***DCVC*** protocol. **$SFTR_{V2I}$**(*SFTR for V2I communication*): This is a measure of how efficiently our protocol can successfully transmit data and handle data loss by message collision. *SFTR* is calculated by using the following formula:

$$SFTR = \frac{nSF}{nTF} \times 100\% \tag{1}$$

$nSF = Number\ of\ successfully\ transmitted\ frames$
$nTF = Total\ number\ of\ transmitted\ frames$

The result for $SFTR_{V2I}$ is summarized in Table 6. We plot $SFTR_{V2I}$ for different number of vehicles in Figure 10. And it is seen from the plot that with increasing number of vehicles the $SFTR_{V2I}$ is reducing at a very slow rate which shows that our protocol is highly effective in successfully transmitting the frames.

6.4 Speed Limit for Vehicles

Here, we calculate the speed limit of the vehicles which allows them to stay within the range of *RSSM* so that the

	Condition	nTF_{avg}	nSF_{avg}	$SFTR_{V2I}$ (%)
a)	1 vehicle in system	99	99	100
b)	2 vehicles in system	99	99	100
c)	3 vehicles in system	99	98.6667	99.6667

Table 6: Result Summary for $SFTR_{V2I}$

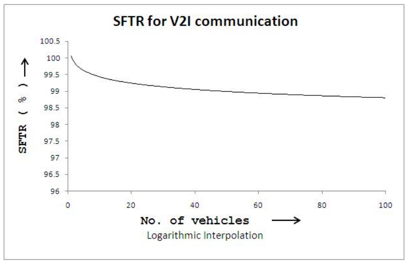

Figure 10: $SFTR_{V2I}$ vs. No. of Vehicles

RSSM can properly communicate with them.
Let,

B = Effective bitrate for frame transmission
L = Frame size
S = Distance covered by vehicle
t_{frame} = A single frame transmission time = $\frac{L}{B}$
D_r = Delay due to data retransmission
D_{RSSM} = Delay between successive broadcasting by *RSSM*
T = Minimum time a vehicle should stay in the range of a *RSSM* to be recognized
ϵ = Reduction in allowable velocity due to delay in data transmission which is caused by various problems that occur for the mobility of vehicles such as channel fading, communication obstacles, shadowing, Doppler shifts etc and hardware delay.
V = Velocity of a vehicle = $\frac{S}{T} - \epsilon$

In our testbed, B = 4800bps, F = 72bit(Figure 7), S = diameter of the radio range of *RSSM* = 200m, D_{RSSM} = 100ms and $t_{frame} = \frac{72}{4800}$s = 15ms.

Best Case(No message collision): In this case, three frames*(SB,VR,ACK)* are successfully exchanged once between *RSSM* and *VM* (*see* section 4.2).

So, $D_r = 0$. $T = D_r + D_{RSSM} + 3(t_{frame}) = 0 + 100 + (3 \cdot 15)$ms = 145ms. Therefore, $V = \frac{200m}{145ms} - \epsilon = 1379.31ms^{-1} - \epsilon$.

Worst Case(Maximum message collision): In this case, the *RSSM* successfully sends its *SB/SO* frame in a single try. But the response from the vehicle is lost and the vehicle waits for maximum amount of time between successive retransmissions according to *BEB* algorithm [2].

So, $D_r = \sum_{i=1}^{10}(2^i - 1) + (2^{10} - 1) \cdot 6 = (2036 + 6138)$ms = 8174ms.
$T = D_r + D_{RSSM} + t_{frame} + 16 \cdot 2 \cdot t_{frame}$ = (8174 + 15 + 480)ms = 8669ms. Therefore, $V = \frac{200m}{8669ms} - \epsilon = 23.07ms^{-1} - \epsilon$.

So, the *RSSM* can properly communicate with the vehicles moving at approximately $1379.31ms^{-1}$ and $23.07ms^{-1}$ in the best case and worst case respectively. Since this is an experimental testbed, we have used the free ISM band and low bitrate due to resource limitations. But 802.11 MAC sub layer implemented hardware could easily be used for high frequency and bitrate which would definitely improve the protocol's performance.

7. FUTURE ENHANCEMENTS

There are lots of scopes to improve our protocol and enhance the application of it on which we are still working. For example, according to our protocol, each *RSSM* stores all information received from the vehicles within its range. A *Central Server* can be introduced which can communicate with multiple *RSSM*s and collect the information from them along with the timestamps of the events. Then this centrally stored information can be used for different purposes. Such as, tracking the location of a vehicle at any time(can be helpful to police force, private car owner), providing quick help to the vehicle in emergency by knowing its location etc.

8. CONCLUSION

In summary, we introduce an effective protocol for V2I and V2V communication and test it on a testbed. We also implement two of many possible applications of the protocol and characterize the performance of it.

9. REFERENCES

[1] A. Agarwal, D. Starobinski and T.D.C. Little. Exploiting Downstream Mobility to Achieve Fast Upstream Message Propagation in Vehicular Ad Hoc Networks. *Proc. Mobile Networking for Vehicular Environments 2007, Infocom 2007*, May 2007.

[2] Andrew S. Tanenbaum. *Computer Networks*. Pearson Education Inc and Dorling Kindersley Publishing, Inc, 4th edition, 2003.

[3] Atmel Corporation, 2325 Orchard Parkway, San Jose, CA 95131, USA. *ATMega8 Datasheet*, 2007.

[4] FairChild Semiconductor Corporation . *7805 Datasheet*, 2001.

[5] Hope Microelectronics Co.,Ltd, Rm B.8/F LiJingGe Emperor Regency 6012 ShenNan Rd., Shenzhen,China. *RFM12 Datasheet*, 2006.

[6] P. Basu and T.D.C. Little. Wireless Ad Hoc Discovery of Parking Meters. *MobiSys 2004 Workshop on Applications of Mobile Embedded Systems*, June 2004.

[7] P. Papadimitratos, G. Calandrielloy , J. Hubaux and A. Lioyy. Impact of vehicular communications security on transportation safety. *Computer Communications Workshops, Infocom 2008*, April 2008.

[8] R. Lu, X. Lin,H. Zhu,PH. Ho and X. Shen. ECPP: Efficient Conditional Privacy Preservation Protocol for Secure Vehicular Communications. *Proc. The 27th Conference on Computer Communications 2008, Infocom 2008*, April 2008.

[9] S. Katragadda, G. Murthy, R. Rao, M. Kumar and Sachin R. A Decentralized Location-based Channel Access Protocol for Inter-Vehicle Communication. *Proc. of IEEE VTC 2003*, October 2003.

[10] T.D.C. Little and A. Agarwal. A New Information Propagation Scheme for Vehicular Networks. *Proc. 3rd Intl. Conf. on Mobile Systems, Applications and Services (Mobisys 2005)*, June 2005.