International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119 Vol. 3 No. 1 June 2016, pp. 44-56

http://ijpsat.ijsht-journals.org

Message Transmission Using VANET

Anan Srivastava, Kuhu Bisen and Mitesh Kothari

VIT University
India

ABSTRACT: Vehicular ad-hoc networks (VANETs) technology has emerged as an important research area over the last few years. Being ad-hoc in nature, VANET is a type of network that is created from the concept of establishing a network of cars for a specific need or situation. VANETs have now been established as reliable networks that vehicles use for communicating on highways or urban environments. While there are great benefits, there are a large number of challenges in VANET such as provisioning of QoS, high connectivity, bandwidth, security to vehicle and individual privacy. Finding the best route from source to destination in a complex and complicated road network is a major challenge because of the traffic conditions in modern cities. This paper presents an application of graph theory in VANET that offers a solution to this problem. We have made use of the Bellman-Ford algorithm and Dijktra algorithm. This application aims at reducing the total travel time of the message/packet by providing the best route. This paper also includes 2 error correcting codes; the Hamming (7,4) code and the Cyclic Redundancy Check (CRC) that will help detect and correct bit errors while transmitting information in VANET. We have also made use of the Lempel-Ziv-Welch (LZW) Encoding technique to encode the message in its respective bit format before transmission.

KEYWORDS: VANET, Modeling, Hamming (7, 4), Routing, VANET Security.

1. INTRODUCTION

Although, Vehicular Ad-hoc Network (VANET) is not a new topic, it continues to provide new research challenges and problems. The main objective of VANET is to help a group of vehicles to set up and maintain a communication network among them without using any central base station or any controller. VANETs best application is during a medical emergency on road However, along with these useful applications of VANET, emerge new challenges and problems. Lack of infrastructure in VANET puts additional responsibilities on vehicles. Every vehicle becomes part of the network and also manages and controls the communication on this network along with its own communication requirements. You may also notice how the set of candidate applications, to be developed for this technology, with available resources and limitations, make the VANET a distinct area of research in wireless communications. The main setback of VANET is the absence of infrastructure, such as access point or base stations, in the Wi-Fi, GSM or UMTS.

Vehicular ad-hoc networks are responsible for the communication between moving vehicles in a certain environment. A vehicle can communicate with another vehicle directly, which is called Vehicle to Vehicle (V2V) communication, or a vehicle can communicate with an infrastructure such as a Road Side Unit (RSU), known as Vehicle to Infrastructure (V2I). The nodes that are beyond the range of the transmission and have several multiple nodes between them act as multi-hop/signal booster nodes. There are many important applications of VANET in today's world. These applications range from critical medical services to comfort and leisure activities. A VANET must be able to fulfill all requirements of the ever changing user needs and should also comply with the standards and architectures of the available technology.

Corresponding Author: Kuhu Bisen

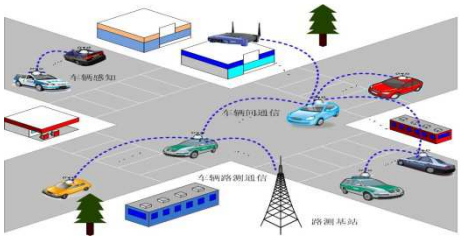

Fig. a. VANET representation

2. APPLICATION OF VANET

The use of radio communications for the interaction between Vehicle to Vehicle or Vehicle to Infrastructure is strongly aimed at establishing an ITS. The use of these technologies can help get real time news on current traffic conditions and other relevant details. Applications for VANET include safety, traffic management, driver assistance, web surfing, voice, gaming and infotainment applications. A vehicle may contain multiple application units which are either built-in or integrated on a portable communication device, running a variety of these applications. These applications have different delay, reliability, security, and bandwidth requirements. Due to the mere size, a VANET may grow into; researchers have identified the need for a new mobility mechanism, defined as network mobility. Some major application of VANET are reduction of traffic congestion, reduction of the emission of exhaust gas due to traffic, prevention of road accidents and improved social acceptance of the automobile. The Vehicle to Vehicle type of communication plays a rather dominant role based on much of the research till today.

3. APPLICATION OF GRAPH IN VANET

Practical applications of graph theory include route planning in different type of networks. The research question for this work is how efficiently the concepts of graph theory can be used in route planning for VANET. Consider the figure c, the cars in the figure are represented as the nodes in the graph and the wireless networks, as edges. The protocol stack, called the network layer, is responsible for routing the transmitting data. This will make use of the Bellman-Ford algorithm and Dijktra algorithm. Due to the highly dynamic nature of the VANET, finding and maintaining paths for the routing packets is a difficult task. However, there are a few proposals that have been put on the table in order to solve this problem. VANET routing can be classified under topology based routing, position-based routing, cluster-based routing, broadcast routing and geo-cast routing. These groups include a whole range of different protocols, each covering specific requirements and needs. In this paper, we will describe how graph theory is used in VANET. We have used a technique that first requires to create a table, listing out the source, destination and the edge weight from one node to another. Once that has been done, we create a table 2, having all the neighboring tables that include the information of packet transmission from that node to other nodes via the source node. This ends when all the nodes have received all the tables. In addition, the table should be constantly updated periodically or when there is a change in network topology. The minimum distance to transfer packet information, from one node to another, is calculated using the Bellman-Ford algorithm. The routing table of every node has the IP addresses of the destination node (destination IP); sequence number of the destination node (destination sequence number); flag of validity of the route (whether the route is valid or invalid); distance in number of hops (hop count), list of precursors for that route, time of validity (lifetime). If a node wants to transfer a packet, we have to know 2 things; which all nodes are connected to the network and the topology of the network so that the shortest path can be calculated using the Bellman-Ford algorithm. The routing based on distance vector, is simple from a computational point of view and does not require large quantities of memory. However, establishing a secure connection between two nodes requires more time.

4. SCENARIO

In this paper we will proceed with a demonstration of a real scenario depicting the application of graph theory in VANET. The following figure represents an intersection of four roads, where the vehicles form a network and transfer data through the theoretical considerations set out above.

The yellow lines in the figure show the links that exist between the various wireless nodes (vehicle), the yellow car on the

left has 3 connections, one with the blue car and the other with the cars cut off from the below picture. Some of the cars are cut off from the figure, but will be considered in the graph that represents the network. Now going to consider only the necessary parts of this scenario we get the following picture:

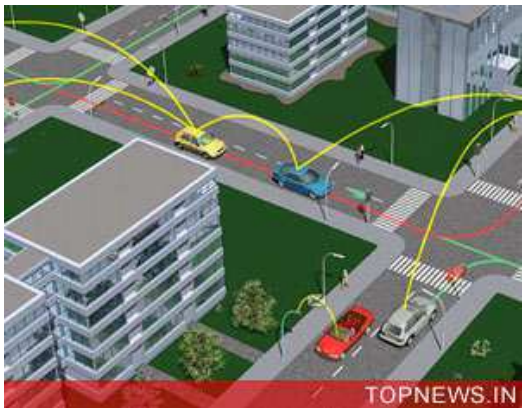

Fig. b. Example scenario

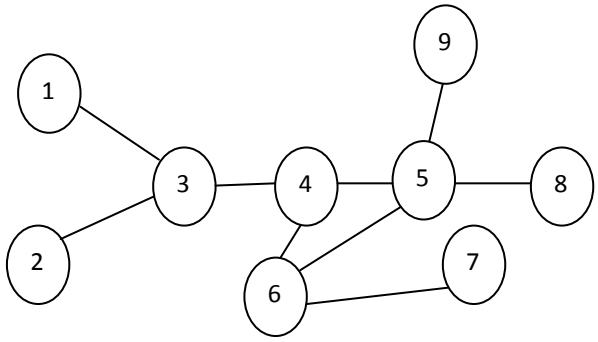

Fig. c. Graphical representation of the above scenario:

Please note that the weight of each edge here is an arbitrary value, randomly assigned, while in reality it is calculated from a series of factors such as vehicle speed, weather conditions etc. Each node (vehicle) has a protocol stack that consists of a physical layer formed by the hardware that allows wireless connection, a network layer that is responsible for implementing the protocol and an application layer formed by the number of hardware utilities. We use the distance vector and the Bellman-Ford algorithm for the resolution of paths. The above graph shows the shortest path between two nodes for data exchange.

Finally we can construct a routing table for each node with the shortest path length.

From the below table, it is clear that each node knows the topology of the graph and by considering the case in which one node requires the transmission of data to another node, it will first calculate the shortest path between those two nodes and then proceed with the transmission.

STEP 1: Creating tables from all nodes (S = Source, D = Destination, W = Weight).

Node 1			Node 2			Node 3			Node 4			Node 5			Node 6			Node 7			Node 8			Node 9		
S	D	W	S	D	W	S	D	W	S	D	W	S	D	W	S	D	W	S	D	W	S	D	W	S	D	W
1	3	5	2	3	3	3	1	5	4	3	4	5	4	6	6	5	6	7	6	2	8	5	4	9	5	3
--	--	--	--	--	--	3	2	3	4	5	6	5	6	6	6	4	5	--	--	--	--	--	--	--	--	--
--	--	--	--	--	--	3	4	4	4	6	5	5	9	3	6	7	2	--	--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--	--	--	--	--	5	8	4	--	--	--	--	--	--	--	--	--	--	--	--

STEP 2: The nodes communicate with their neighboring tables built by deleting duplicate records

Node 1			Node 2			Node 3			Node 4			Node 5			Node 6			Node 7			Node 8			Node 9		
S	D	W	S	D	W	S	D	W	S	D	W	S	D	W	S	D	W	S	D	W	S	D	W	S	D	W
1	3	5	2	3	3	3	1	5	4	3	4	5	4	6	6	5	6	7	6	2	8	5	4	9	5	3
3	2	3	3	1	5	3	2	3	4	5	6	5	6	6	6	4	5	6	4	5	5	4	6	5	4	6
3	4	4	3	4	4	3	4	4	4	6	5	5	9	3	6	7	2	6	5	6	5	6	6	5	6	6
--	--	--	--	--	--	4	5	6	5	9	3	5	8	4	5	9	3	--	--	--	5	9	3	5	9	3
--	--	--	--	--	--	4	6	5	5	8	4	6	7	2	5	8	4	--	--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--	--	5	6	6	4	3	4	4	3	4	--	--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--	--	6	7	2	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--	--	3	1	5	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--	--	3	2	3	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Assume that, node 1 wants to transmit a packet to node 7

Table 2. Minimum Weights Table

Iter	N	N 2	N 3	N 4	N 5	N 6	N 7	N 8	N 9
h=0	0	∞	∞	∞	∞	∞	∞	∞	∞
h=1	0	∞	5 (1)	∞	∞	∞	∞	∞	∞
h=2	0	8 (3)	5 (1)	9 (3)	∞	∞	∞	∞	∞
h=3	0	8 (3)	5 (1)	9 (3)	15 (4)	14 (4)	∞	∞	∞
h=4	0	8 (3)	5 (1)	9 (3)	15 (4)	14 (4)	∞	19 (5)	18 (5)
h=5	0	8 (3)	5 (1)	9 (3)	15 (4)	14 (4)	16 (6)	19 (5)	18 (5)

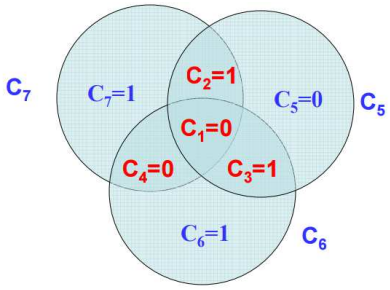

Fig. c. Coverage of Parity Bits

Table 1. Shortest Path Table

S	D	W
1	3	5
2	3	3
3	4	4
4	5	6
4	6	5
5	6	6
5	8	4
5	9	3
6	7	2

5. APPLICATION OF HAMMING (7, 4) CODE IN VANET

For the error correction and security application, the channel coding plays a vital role because the transmission channel may be easily affected by noise. So it is necessary to consider another aspect which concerns the application context, the apparatus available for this type of application does not have large computational capacity in terms of processor and memory, so encoding and decoding takes time. This also makes the communication between devices on the network very slow. Hence, to resolve this, we use the Hamming code for encoding and decoding. Hamming code is linear, perfect, and capable of detecting and correcting errors, which makes it perfect to directly implement it into hardware encoding and decoding.

There are two methods to encode and decode by hamming,

METHOD 1: Logical operations.

METHOD 2: Using Generator matrix (G) and Parity matrix (H). For the Hamming code (7, 4), taking advantage of the scheme in Table 2 is obtained the following code words.

d1	d2	d3	d4	p1	p2	d1	p3	d2
0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	1	0
0	0	1	0	0	1	0	1	0
0	0	1	1	1	0	0	0	0
0	1	0	0	1	0	0	1	1
0	1	0	1	0	1	0	0	1
0	1	1	0	1	1	0	0	1
0	1	1	1	0	0	0	1	1
1	0	0	0	1	1	1	0	0
1	0	0	1	0	0	1	1	0
1	0	1	0	1	0	1	1	0
1	0	1	1	0	1	1	0	0
1	1	0	0	0	1	1	1	1
1	1	0	1	1	0	1	0	1
1	1	1	0	0	0	1	0	1
1	1	1	1	1	1	1	1	1

From Table 1 shows that for Hamming (7, 4) is obtain as follows.

$$p_1 = d_1 + d_2 + d_3$$

$$p_2 = d_1 + d_3 + d_4$$

$$p_3 = d_2 + d_3 + d_4$$

The coverage of the parity bit can be defined by Figure c.

The parity bit P1 indicates that it is able to correct an error in either d1, d2 or/and d4.

The word p_1, p_2, p_3, d_1, d_2, d_3, d_4 that must be coded to generate other three bits k_1, k_2 and k_3 defined as follows:

$$k_1 = p_1 + d_1 + d_2 + d_4$$

$$k_2 = p_2 + d_1 + d_3 + d_4$$

$$k_3 = p_3 + d_2 + d_3 + d_4$$

This type of coding is extremely fast and simple to implement because of the fact that, it requires the use of only XOR gates.

Second method with matrices G and H

The three-digit parity of different combinations of the four digits of information in the following scheme

$$p_1 = i_2 + i_3 + i_4 \; p_2 = i_1 + i_3 + i_4 \; p_3 = i_2 + i_3 + i_4$$

Which are arranged to form the 16 words of the code:

In total 2^k =16 and 2^n = 128 configurations are use for 7 bits, and the code words differ in at least three positions. Then d = 3 and the code can detect all single and double errors and is able to correct all. For correction, simply search the word that differs in only one bit from the one received.

The form c = i • G where c is the vector of 7 elements

transfer (code word), i is the vector of 4 elements of the digits of information, G is the matrix generating size 4x7:

p1	p2	p3	d1	d2	d3	d4
0	1	1	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Table 4. Output Of The Encoder

K1	K2	K3	Meaning	Output Encoder
0	0	0	No Error	0000000
0	0	1	Error in receiving bit 1(bit p1)	1000000
0	1	0	Error in receiving bit 2(bit p2)	0100000
0	1	1	Error in receiving bit3(bit p3)	0010000
1	0	0	Error in receiving bit4(bit p4)	0001000
1	0	1	Error in receiving bit5(bit p5)	0000100
1	1	0	Error in receiving bit6(bit p6)	0000010
1	1	1	Error in receiving bit7(bit p7)	0000001

Table 5. Correction Table Using Matrix G And H

I_1	I_2	I_3	I_4	I_1	I_2	I_3	I_4	P_1	P_2	P_3
0	0	0	0	→ 0	0	0	0	0	0	0
0	0	0	1	→ 0	0	0	1	1	1	1
0	0	1	0	→ 0	0	1	0	1	1	0
0	0	1	1	→ 0	0	1	1	0	0	1
0	1	0	0	→ 0	1	0	0	1	0	1
0	1	0	1	→ 0	1	0	1	0	1	0
0	1	1	0	→ 0	1	1	0	0	1	1
0	1	1	1	→ 1	1	1	1	1	0	0
1	0	0	0	→ 1	0	0	0	0	1	1
1	0	0	1	→ 1	0	0	1	1	0	0
1	0	1	0	→ 1	0	1	0	1	0	1
1	0	1	1	→ 1	0	1	1	0	1	0
1	1	0	0	→ 1	1	0	0	1	1	0
1	1	0	1	→ 1	1	0	1	0	0	1
1	1	1	0	→ 1	1	1	0	0	0	0
1	1	1	1	→ 1	1	1	1	1	1	1

we can calculate the Syndrome y = c + e where e is the error vector,

$$s = y.H = \underline{c}.H + \underline{e}.H = \underline{e}.H$$

The Syndrome is a vector of n - k = 3 components, which says if the three rules are equal or less satisfied. The syndrome may occur in 23 = 8 ways, while the possible configurations of error (including the lack of errors) are 27 = 128. Easily occurs that the same syndrome corresponds to 128/8 = 16 different configurations of error. If the calculation of the Syndrome produces a null vector means that the received word is correct, , otherwise adding the coset leader, corresponding to the obtained syndrome, the word received will be possible to correct the error.

For example, if the received word is w = 1110000 then w • H = 000 which is the syndrome 0000000.
From that v = W + u = 1,110,000 + 0,000,000 = 1,110,000.
If the received word is w = 1110001 then w • H = 001 which is the syndrome of 0,000,001.
From that: v = w + u =1110001 + 0000001 = 1110000. The word received was correct.

Table 6. SDA For The Hamming Code (7, 4)

Coset leaders u	Sindrome u H
0000000	000
1000000	011
0100000	101
0010000	110
0001000	111
0000100	100
0000010	010
0000001	001

HAMMING BINARY BLOCK CODE WITH k=4 AND n=7
Message digits: C 1 C 2 C 3 C 4
• Code word C 1 C 2 C 3 C4 C 5 C 6 C 7
Parity Check Equations: C1 ⊕ C2 ⊕ C3 ⊕ C5 = 0
C1 ⊕ C3 ⊕ C4 ⊕ C6 = 0
C1 ⊕ C2 ⊕ C4 ⊕ C7 = 0
Parity Check Matrix: 1 1 1 0 1 0 0
1 0 1 1 0 1 0
1 1 0 1 0 0 1

In computer science terms, parity comes in two varieties even and odd. A string of bits (1's and 0's) has even parity if it contains an even number of 1's (the modulo 2 sum of the bits is 0), otherwise it has odd parity (the modulo 2 sum of the bits is 1). Many error detection schemes utilize parity bits. A parity bit is an extra bit that forces a binary string to have a specific parity. The parity bit for an even parity block may be computed by summing all the bits modulo 2. The parity bit for an odd parity block may be computed by summing all the bits modulo 2 and adding 1. The table below lists all possible three bit values and value of a parity bit required to create a four bit sequence with even parity.

Even Parity

3 Bit String	Parity Bit	Verification
000	0	0 + 0 + 0 + 0 = 0
001	1	0 + 0 + 1 + 1 = 0
010	1	0 + 1 + 0 + 1 = 0
011	0	0 + 1 + 1 + 0 = 0
100	1	1 + 0 + 0 + 1 = 0
101	0	1 + 0 + 1 + 0 = 0
110	0	1 + 0 + 1 + 0 = 0
111	1	1 + 1 + 1 + 1 = 0

6. LEMPEL-ZIV-WELCH (LZW) ENCODING

Like its predecessor LZSS (LZ77), the Lempel-Ziv-Welch algorithm uses a dynamically generated dictionary and encodes strings by a reference to the dictionary. It is intended that the dictionary reference should be shorter than the string it replaces. As you will see, LZW achieves its goal for all strings larger than 1.

Encoding

The LZSS algorithm uses a sliding window dictionary. Each entry in the dictionary is a character. LZSS code words consist of an offset into the dictionary and the number of characters following the offset to include in an encoded string.
Unlike LZSS, entries in the LZW dictionary are strings, and every LZW code word is a reference to a string in the dictionary.

The Dictionary and Encoded Strings

The LZW dictionary is not an external dictionary that lists all known symbol strings. Instead, the dictionary is initialized with an entry for every possible byte. Other strings are added as they are built from the input stream. The code word for a string is simply the next available value at the time it is added to the dictionary.

An "encoded" string is used to add new strings to the dictionary. The encoded string is built from the input stream. The input stream is read 1 byte at a time. If the string formed by concatenating the new byte to the encoded string is in the dictionary, the new byte is added to the end of the encoded string. Otherwise a dictionary entry is made for the new string and the code word for the coded string is written to the output stream. Then the byte that was just read becomes the encoded string.

The Basic Encoding Algorithm

Based on the discussion above, encoding input consists of the following steps:
Step 1. Initialize dictionary to contain one entry for each byte. Initialize the encoded string with the first byte of the input stream.
Step 2. Read the next byte from the input stream.
Step 3. If the byte is an EOF go to step 6.
Step 4. If concatenating the byte to the encoded string, produces a string that is in the dictionary:

- Concatenate the byte to the encoded string.
- Go to step 2.

Step 5. If concatenating the byte to the encoded string produces a string that is not in the dictionary:

- Add the new sting to the dictionary.
- Write the code for the encoded string to the output stream.
- Set the encoded string equal to the new byte.
- Go to step 2.

Step 6. Write out code for encoded string and exit.

Example: Node 1 wants to send Node 7 the following message: "this_is_his_thing" is encoded as follows:

New Byte	Encoded String	New Code	Code Output
t	t	*None*	*None*
h	h	256 (th)	t
i	i	257 (hi)	h
s	s	258 (is)	i
_	_	259 (s_)	s
i	i	260 (_i)	_
s	is	*None*	*None*
_	_	261 (is_)	258 (is)
h	h	262 (_h)	_
i	hi	*None*	*None*
s	s	263 (his)	257 (hi)

_	s_	*None*	*None*
t	t	264 (s_t)	259 (s_)
h	th	*None*	*None*
i	i	265 (thi)	256 (th)
n	n	266 (in)	i
g	g	267 (ng)	N
None	*None*	*None*	G

In the example above, a 17 character string is represented by 13 code words. Any actual compression that would occur would be based on the size of the code words. In this example code words could be as short as 9 bits. Typically code words are 12 to 16 bits long. Of course the typical input stream is also longer than 17 characters.

Decoding

It shouldn't be a big surprise that LZW data is decoded pretty much the opposite of how it's encoded. The dictionary is initialized so that it contains an entry for each byte. Instead of maintaining an encoded string, the decoding algorithm maintains the last code word and the first character in the string it encodes. New code words are read from the input stream one at a time and string encoded by the new code is output.

During the encoding process, the code prior to the current code is written because concatenating the first character of the current code with the string encoded by the prior code generated a code that wasn't in the dictionary. When that happened the string formed by the concatenation was added to the dictionary. The same string needs to be added to the dictionary when decoding.

The Basic Decoding Algorithm

Based on the discussion above, decoding input consists of the following steps:

Step 1. Initialize dictionary to contain one entry for each byte.

Step 2. Read the first code word from the input stream and write out the byte it encodes.

Step 3. Read the next code word from the input stream.

Step 4. If the code word is an EOF exit.

Step 5. Write out the string encoded by the code word.

Step 6. Concatenate the first character in the new code word to the string produced by the previous code word and add the resulting string to the dictionary.

Step 7. Go to step 3.

Example 2: Decode the stream 't' 'h' 'i' 's' '_' 258 '_' 257 259 256 'i' 'n' 'g' produced by the previous example

Input Code	Encoded String	Added Code	String Output
t	t	*None*	t
h	h	256 (th)	h
i	i	257 (hi)	i
s	s	258 (is)	s
_	_	259 (s_)	_

258	is	260 (_i)	is
_	_	261 (is_)	_
257	hi	262 (_h)	hi
259	s_	263 (his)	s_
256	th	264 (s_t)	th
i	i	265 (thi)	i
n	n	266 (in)	n
g	g	267 (ng)	g

The decode string matches the original encoded string. There's no need to include extra information commonly required by statistical algorithms like Huffman Code and Arithmetic Code. So compression ratios are never reduced by extra data cost.

7. CYCLIC REDUNDANCY CHECK

Let us assume k message bits and n bits of redundancy. Associate bits with coefficients of a polynomial

1 0 1 1 0 1 1

1x6+0x5+1x4+1x3+0x2+1x+1 = x6+x4+x3+x+1

Let M(x) be the message polynomial
Let P(x) be the generator polynomial
P(x) is fixed for a given CRC scheme
P(x) is known both by sender and receiver
Create a block polynomial F(x) based on M(x) and P(x) such that F(x) is divisible by P(x).

$$\frac{F(x)}{P(x)} = Q(x) + \frac{0}{P(x)}$$

Prove that $x^n M(x) + C(x)$ is divisible by $P(x)$

$$P(x)\overline{)\,x^n M(x)}^{\,Q(x)}, \text{ remainder } C(x)$$

$$\therefore x^n M(x) = P(x)Q(x) + C(x)$$

$$\frac{x^n M(x) + C(x)}{P(x)} = \frac{P(x)Q(x)}{P(x)} + \frac{C(x) + C(x)}{P(x)}$$

EXAMPLE

1. Send
M(x) = 1110000 => x6+x5+x4 (7 bits)
P(x) = 11001 => x4+x3+1 (5 bits, n = 4)
=> 4 bits of redundancy
Form xnM(x) => 1110000 0000
=> x9+x8+x5+x4
Divide xnM(x) by P(x) to find C(x)

```
          1011001
11001 ) 11100000000
        11001
        -----
        0010100
          11001
          -----
          011010
            11001
            -----
            00011000
               11001
               -----
               00001 = C(x)
```

Send the block 1110000 00001

2. Receive

```
          1011001
11001 ) 11100000001
        11001
        -----
        0010100
          11001
          -----
          011010
            11001
            -----
            00011001
               11001
               -----
               00000
```

No remainder → Accept the word.

8. CONCLUSION

The application of graph theory in routing of different networks is a new area of research. We consider VANET as a dynamic graph where the vehicles act as the node of the graph and the wireless links as edges. The aim is to transmit and receive the data between the vehicles in real time by constructing a secure network. Transmission and reception is occurring at network layer and the techniques used in this paper are based on Bellman-Ford algorithm and Dijktra algorithm. Recent study has proved that further innovation and technical protocols have enabled easier transmission with fewer problems. We have used Lempel-Ziv-Welch (LZW) Encoding to encode words into their ASCII characters for easier transmission, and then check for their correct transmission. To check for any errors or redundancy, we have made use of Hamming (7,4) code and Cyclic Redundancy Check(CRC). An example to portray both logics is mentioned in the paper.

REFERENCES

[1] Dipperstein, M. (2016). *Hamming (7,4) Code Discussion and Implementation*. [online] Michael.dipperstein.com. Available at: http://michael.dipperstein.com/hamming/ [Accessed 30 Mar. 2016].

[2] Cs.jhu.edu. (2016). *The Cyclic Redundancy Check*. [online] Available at: http://www.cs.jhu.edu/~scheideler/courses/600.344_S02/CRC.html [Accessed 4 Apr. 2016].

[3] Kim, S. and Lee, I. (2014). *A Secure and Efficient Vehicle-to-Vehicle Communication Scheme using Bloom Filter in VANETs*. [online] http://www.sersc.org/. Available at: http://www.sersc.org/journals/IJSIA/vol8_no2_2014/2.pdf [Accessed 4 Apr. 2016].

[4] http://article.sapub.org/. (2013). *Vehicular Ad-Hoc Networks (VANETs) - An Overview and Challenges*. [online] Available at: http://article.sapub.org/pdf/10.5923.j.jwnc.20130303.02.pdf [Accessed 30 Mar. 2016].

[5] Zia, T., Khan, M., ur Rehman, S. and Zheng, L. (2013). *Vehicular Ad-Hoc Networks (VANETs) - An Overview and Challenges*. [online] http://article.sapub.org/. Available at: http://article.sapub.org/pdf/10.5923.j.jwnc.20130303.02.pdf [Accessed 30 Mar. 2016].

[6] Wolf, J. (2008). *AN INTRODUCTION TO ERROR CORRECTING CODES*. [online] ucsd.edu. Available at: http://circuit.ucsd.edu/~yhk/ece154c-spr15/ErrorCorrectionI.pdf [Accessed 30 Mar. 2016].

[7] uotechnology.edu.iq. (n.d.). *Error Detection and Correction*. [online] Available at: http://uotechnology.edu.iq/dep-eee/lectures/4th/Communication/Information%20theory/6.pdf [Accessed 4 Apr. 2016].
[8] W. Smith, S. (2011). *LZW Compression*. [online] Dspguide.com. Available at: http://www.dspguide.com/ch27/5.htm [Accessed 4 Apr. 2016].
[9] SEDGEWICK, R. and WAYNE, K. (n.d.). *BELLMAN-FORD DEMO*. [online] http://algs4.cs.princeton.edu/. Available at:
[10] http://algs4.cs.princeton.edu/lectures/44DemoBellmanFord.pdf [Accessed 30 Mar. 2016].
[11] Techie Me. (2015). *Shortest Path using Dijkstra's Algorithm - Techie Me*. [online] Available at: http://techieme.in/shortest-path-using-dijkstras-algorithm/ [Accessed 30 Mar. 2016].
[12] RexFord, J. (2006). *Shortest-Path Routing*. [online] www.cs.princeton.edu. Available at: http://www.cs.princeton.edu/courses/archive/spr06/cos461/.../15Routing.ppt [Accessed 30 Mar. 2016].