Application of LDPC Codes to Networks

R. Bijjargi, M.A. Ambroze and B.V. Ghita

Centre for Security, Communications and Network Research
Plymouth University, United Kingdom
e-mail: info@cscan.org

Abstract

Internet is growing very fast and it's rapid. At present internet is used by everyone for one or the other reason. Data is sent from source to destination in the form of packets. Packets are the small pieces of original data which contains original data along with header information. Each and every packet has header, which contains the information like source and destination address, mac address, etc.
Original data is sent from source to destination over network. But it is no guarantee that data will reach the destination without any errors that is it will not be corrupted. It may be corrupted and may contain some noise in it. Therefore error correcting codes are implemented in case of data transfer to avoid the data corruption.
Error correction codes are applied on data bits. But in the proposed project error correction codes are tried to implement on packet loss. If there i a packet loss then receiver will send a request to source for retransmission. This will lead to a network traffic and congestion over network. Using error correcting codes lost data is recovered by destination without asking for retransmission.
To avoid the retransmission original data is sent with overhead. Size of the overhead is less, definitely less than the lost data size. Data encryption and decryption is performed using Hamming Code to recover the lost data. As huge the Parity Matrix is less overhead will be transferred. Maximum 25% of the lost data can be recovered. To recover lost data buffer should contain all the remaining data along with parity data.

Keywords

Encryption, Decryption, Congestion, Network Traffic, Delay, Packet Loss, Buffer Size, Overhead.

1 Introduction

Invention of the Computer changed the whole world and networking is complemented it in a great way. Networking is started with ARPANET and grown rapidly. J C R Licklider of MIT wrote series of memo discussing about "Galactic Network" concept in the year 1962. In his concept he clearly mentioned that computers are interconnected and anybody from anywhere in the world can access the data which is present in it. It replicated very much about the present Internet (Internet Society,2011). In the beginning data was transferred in network through circuit switching. Later packet switching is introduced and it captured whole market within no time. Compare to the older technique and methods of data transfer, present data transfer rate is much higher. Still research is going on to make it more

accurate, better and faster. Error correcting codes have played an important role in the case of data transfer and is contributed by Claude E. Shannon.

Information theory took birth in the year 1948. In this year Claude E. Shannon published his thesis. He is the first person who gave the limits of reliable data transmission over the unreliable channels and provided the solution to achieve the limits. Like Turbo Codes, Low Density Parity Check (LDPC) codes form another class of Shannon Limit. LDPC codes were first discovered by Robert Gallager in the early 1960's in his MIT Ph. D. Dissertation. Low-density parity-check codes are a class of linear codes with sparse parity-check matrix (Lin and Costello,2004).

In case of data transmission over the network entire data is divided into number of packets. Packets contain original data along with source and destination address, that is IP address and MAC address of source and destination. Router chooses the path for each and every packet to travel and path is decided depending on the routing table. As the packets are transmitted through different path, they reach the destination in different time and in different order. In few cases packets may not reach the destination due to congestion, delay, jitter, etc. So in this case destination request the source to resend the packets which are not received. Therefore source will retransmit the lost packets.

In case of packet loss, lost packets are retransmitted by source to destination. Retransmission will increase the network traffic and leads to congestion. My approach is to avoid the retrans- mission in case of packet loss. First of all will develop a code to encrypt and decrypt the data based on Hamming Code and using C programming language. Hamming Codes are used for error correction by the help of parity matrix. C is a procedure oriented high level programming language.

2 Testbed setup for data transfer

As the whole data can not be sent over network at once. First it will be broken into number pieces called packets. Each packet is maximum size of 1500 bytes. This includes header length which contains source and destination IP address, Mac address, flags, etc.

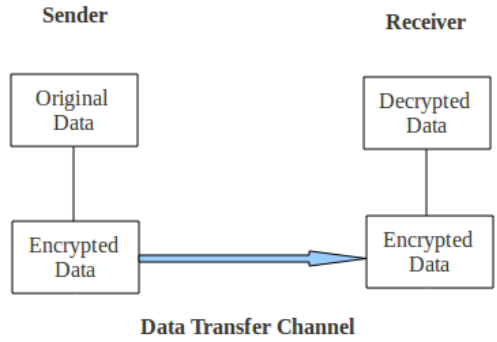

Figure 1: Testbed Setup for Data transfer over network

In the proposed project 1000 bytes of data size is considered excluding header data. Later header part will be added as per the requirement. Figure 1 shows the testbed setup for the data transfer over network.

It is clear from the figure 4.3 that data is encrypted before sending through channel. And it will be decrypted to get the original at the receiver end. Procedure of the data transfer is explained in the following section.

2.1 Procedure

First of all a huge Parity Matrix is generated that is Parity Matrix (12,4095). This Hamming Code is used in both the side that is by sender as well as receiver. The data is encrypted as well as decrypted using the same Parity Matrix (12,4095).

```
srand(time(NULL));
for(i=0;i<4095;i++)
{
  for (j=0;j<8000;j++)
  {
    d=rand()%2;
    printf(" %hd ",d);
    odata[i][j]=d;
  }
  printf("\n");
}
```

Figure 2: Generating Random Data (Kioskea.net, 2011)

Parity Matrix is generated and saved in a file, this file is kept open and read the data at the time of encryption. The needed data for encryption is generated randomly. To generate random data a code is written. Figure 2 shows the code written for generate random data.

Once the random data is generated, data is encrypted using Parity Matrix. The procedure mentioned in the section 4.3.2 is used to encrypt the data.

Parity Matrix	Data packets	Parity packets	Total No of packets	Recoverable Packets
(3,7)	4	3	7	2
(4,15)	11	4	15	4
(5,31)	26	5	31	8
(6,63)	57	6	63	16
(7,127)	120	7	127	32
(8,255)	247	8	255	64
(9,511)	502	9	511	128
(10,1023)	1013	10	1023	256
(11,2047)	2036	11	2047	512
(12,4095)	4083	12	4095	1024

Table 1: Parity Matrix and Number of packets can be recovered

After encryption of data a random number is picked for a packet loss using the function **rand()**. Each parity matrix can recover a specified number of packets. Table 1 shows the recoverable packets for a particular parity matrix.

```
PL: PL=rand()%11;
if(PL==0)
  goto PL;
//PL=4;
printf("\nPackets Lost are : %d",PL);
for(i=0;i<PL;i++)
{
  LP[i]=rand()%4095;
  printf(" %d",LP[i]);
}
if(PL>=2)
{
  for(i=0;i<PL;i++)
  {
    for (j=0;j<(PL-1);j++)
    {
      if(LP[j]>LP[j+1])
      {
        temp=LP[j+1];
        LP[j+1]=LP[j];
        LP[j]=temp;
      }
    }
  }
}
```

Figure 3: Random Packet Loss and Sorting (Kioskea.net, 2011; Allian, 2011)

A random number is generated using C code for a packet loss. Later those many a random number is chosen between 0 to 4095. After generating all the random number they are sorted using bubble sort. C program for the random number generation for packet loss and sorting them in order is shown in Figure 3

```
for(i=0,m=0;i<4095|m<PL;i++)
{
  for (j=0;j<8000;j++)
  {
    if(i==LP[m])
    {
      erdata[i][j]=2;
      printf(" %hd ",erdata[i][j]);
      if(fwrite(&erdata[i][j],sizeof(int),1,fp)!=1)
      {
        printf("Write error occured.\n");
        fclose(fp);
        exit(1);
      }
      if(j==7999)
        m++;
    }
    else
    {
      erdata[i][j]=endata[i][j];
      printf(" %hd ",erdata[i][j]);
      if(fwrite(&erdata[i][j],sizeof(int),1,fp)!=1)
      {
        printf("Write error occured.\n");
        fclose(fp);
        exit(1);
      }
    }
  }
  printf("\n");
}
```

Figure 4: Implementation of Packet Loss through Code

Once all the numbers are set then the packets are loss is performed manually. This is done by using C program. As mentioned earlier packet loss or corrupted data is indicated by '2'. By using for loop lost packets are identified and they are replaced by '2'. The C program for manual packet loss using C code is shown in Figure 4

After performing the packet loss manually lost data is recovered using C code. Here a point is to be considered that a buffer size will be same as total number of packets sent including parity packets. So data is recovered only when all the packets are received. That is minimum required packets to recover the data is 75%. Because maximum data recovered is 25% in case of packet loss. The lost data is not requested for retransmission.

2.2 Results

Original data is huge and it is tedious task to verify the original with recovered data. Therefore a software tool called Ultra Compare Profession is used to compare the data. It will display clearly where the difference, if there is any difference. The result of the comparison is shown in the output window of the software which is present bottom of the window. The comparison result of the recovered data and original data is shown in figure 5

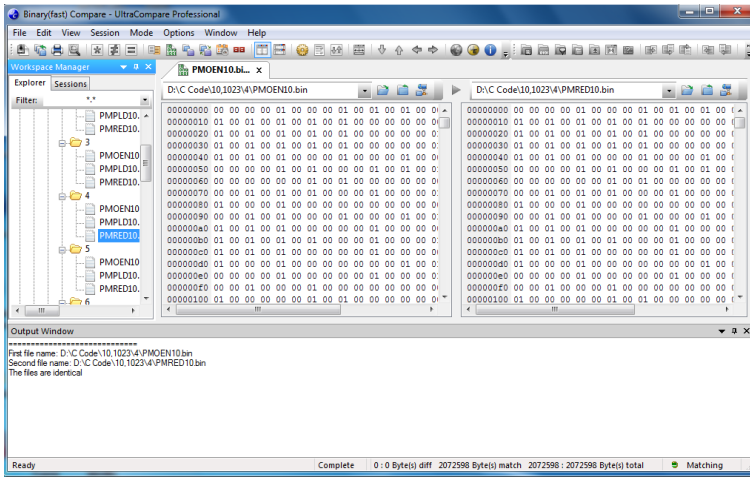

Figure 5: Data Comparison using Ultra Compare Software

Table 2 shows the total number of packets sent over network. It also includes the overhead and the recoverable data using the over head.

Total No. of Packets	Overhead(%)	Recoverable Data (%)
7	42.86	28.57
15	26.67	26.67
31	16.13	25.81
63	9.52	25.40
127	5.51	25.20
255	3.14	25.10
511	1.76	25.05
1023	0.98	25.02
2047	0.54	25.01
4095	0.29	25.01

Table 2: Total no. of packets, overhead and recoverable data in %

Using the data present in Table 2 a graph is drawn and which is shown in figure 6. From the graph it is clear that the maximum data recovery is 25%. Even though how big / huge parity matrix is used to for encryption and decryption of data. But as the Parity Matrix size is increased the overhead is reduced. Therefore we can use huge parity matrix for data recovery so that it will carry less overhead.

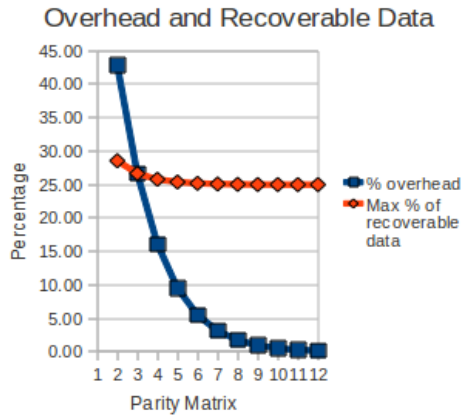

Figure 6: Graph for Overhead and Data Recovery

3 Conclusion

As the internet is growing so fast and almost all the work is done through internet. Message passing, media, entertainment, everything is done through network using computers. Present circumstances everyone want the work done quickly and faster, without any errors. To achieve the same concept error correcting codes should be implemented in the network while data transmission. Even though there are number of error correcting codes are present in market, proposed project tried to implement a error correcting code for packet loss. It is using a very simple and easy code to provide the most accurate answer. So that there should not be any retransmission of data in case of data loss. If this is achieved then surely it will speed up the data transfer rate. Also avoids the congestion and network traffic to some extent by stopping retransmission of lost data.

4 Limitations

As nothing is perfect and all matter have its own limitations. Proposed project also got some limitations. The main limitation is it has to carry some additional data along with original data. This additional data is used to recover the lost data.

The main and important limitation of the proposed project is, it is unable to recover more than

5 packets. Even though a huge Parity Matrix is used for encryption and decryption of data. The problem associated with this is, it is can not recover a lost data because one or the other data is also lost which is used to recover the lost data.

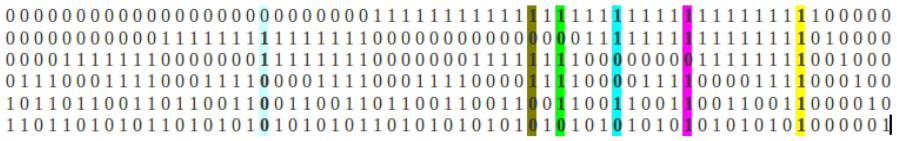

Figure 7: Limitation for a Hamming Code

Figure 7 shows that the 6 bits are lost and are highlighted in different colors. When the program try to recover any of the lost bit then one or the other is lost whose value is need to calculate the lost bit value.

5 Future Work

An important problem associated with Hamming Code is that it can not recover more than 5 packets. It is mainly due to random data in matrix that parity matrix. No matter how huge parity matrix is but the problem remains same. To over come this problem different matrix should be generated which will over come this problem.

In the future work a point to be considered to reduce the overhead size as much as possible.

6 References

Allian A., (2011), Sorting Algorithms - Bubble Sort [accessed on 26-09-2011] http://www.cprogramming.com/tutorial/computersciencetheory/sorting1.html

Internet Society (2011), " Brief History of the Internet" [accessed on 18-12-2011] http://www.internetsociety.org/internet/internet-51/history-internet/brief-history- internet

Kioskea.net (2011), Generating random numbers with rand() Share , [accessed on 25-09-2011] http://en.kioskea.net/faq/878-generating-random-numbers-with-rand

Lin S.; Costello, D. J. (2004) 'Low-Density Parity-Check Codes in Error Control Coding, Pearson Education, Inc., USA: 851-947.

Oualline S., (1997), Practical C Programming 3rd Edition, O Reilly & Associates, Inc., CA

Schildt H., (1997), Teach Yourself C ,Third Edition, McGraw-Hill, USA.