

CS359 Computer Networks Lab 7

Name: M Maheeth Reddy

Roll No.: 1801CS31

Date: 18-Mar-2021

Cyclic Redundancy Check (CRC) : Implementation and Demonstration

Introduction: I have written two programs, client.c and server.c

A string is taken as input from user by the client program. The string is sent to the server in the form of ASCII and CRC remainders. The ASCII code of each character is taken, split into parts containing 4-bits each, then the corresponding CRC remainders are calculated using the CRC key = 1011, and those 7-bits are sent as a single packet to the server. The server uses the data and remainder bits to check if the data is corrupted. If the data got corrupted, it requests the client to send those bits once again. Using this, the corresponding characters are continuously concatenated to reconstruct the string being sent by the client.

Finally, the server acknowledges the client with md5sum of received string. If the md5sums at client and server sides match, the transmission was successful and Cyclic Redundancy Check's purpose is served.

Compilation:

`gcc client.c -o client`

`gcc server.c -o server`

Execution:

`./client`

`./server`

Sample Session:

Sending the message "H"

Client Side

```
[maheeth@maheeth-PC:~/D/netlab7]-[03:02:02 IST]
->$ gcc client.c -o client
[maheeth@maheeth-PC:~/D/netlab7]-[03:02:14 IST]
->$ ./client
Enter a message: H
[+]Server socket created successfully.
[+]Connected to Server.
[+]Sending Packets.

Sending Character: "H" (ASCII 72)
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0100
Data with remainder = 0100 111
Data to be sent = 1000
Data with remainder = 1000 101
-----
Sending Character: "\0" (ASCII 0)
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
-----
... (few more lines here) ...
[+]User message sent successfully.
MD5 client:    c1d9f50f86825a1a2302ec2449c17196
MD5 server:    c1d9f50f86825a1a2302ec2449c17196
[+]MD5 Matched
[+]Closing the connection.
```

Server Side

```
[maheeth@maheeth-PC:~/D/netlab7]-[02:37:32 IST]
->$ gcc server.c -o server
[maheeth@maheeth-PC:~/D/netlab7]-[02:38:04 IST]
->$ ./server
[+]Server socket created successfully.
[+]Binding successful.
[+]Listening....
Data and remainder received: 0000 000
Data Received: 0000    Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 0000 000
Data Received: 0000    Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 0000 000
Data Received: 0000    Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 0000 000
Data Received: 0000    Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 0000 000
Data Received: 0000    Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 0100 111
Data Received: 0100    Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 1000 101
Data Received: 1000    Remainder after CRC checking: 000
Data was received properly

Character "H" (ASCII 72) was received from client
... (few more lines here) ...
[+]Data received successfully.
Message sent by client:
"H"
MD5:    c1d9f50f86825a1a2302ec2449c17196
[+]MD5 sent
[+]Closing the connection.
```

Wireshark Packet Capture for the following input:

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Screenshot from Wireshark: Total Packets captured = **1394**

*Loopback: lo

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter... <Ctrl-F>

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000219588	127.0.0.1	127.0.0.1	TCP	94	56098 → 36895 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=28 TSval=766418605 TSecr=766418605
5	0.000235628	127.0.0.1	127.0.0.1	TCP	66	36895 → 56098 [ACK] Seq=1 Ack=29 Win=65536 Len=0 TSval=766418605 TSecr=766418605
6	0.001459934	127.0.0.1	127.0.0.1	TCP	74	36895 → 56098 [PSH, ACK] Seq=1 Ack=29 Win=65536 Len=8 TSval=766418606 TSecr=766418605
7	0.001472326	127.0.0.1	127.0.0.1	TCP	66	56098 → 36895 [ACK] Seq=29 Ack=9 Win=65536 Len=0 TSval=766418606 TSecr=766418606
8	0.001606523	127.0.0.1	127.0.0.1	TCP	94	56098 → 36895 [PSH, ACK] Seq=29 Ack=9 Win=65536 Len=28 TSval=766418606 TSecr=766418606
9	0.001617344	127.0.0.1	127.0.0.1	TCP	66	36895 → 56098 [ACK] Seq=9 Ack=57 Win=65536 Len=0 TSval=766418606 TSecr=766418606
10	0.001717149	127.0.0.1	127.0.0.1	TCP	74	36895 → 56098 [PSH, ACK] Seq=9 Ack=57 Win=65536 Len=8 TSval=766418606 TSecr=766418606
11	0.001728238	127.0.0.1	127.0.0.1	TCP	66	56098 → 36895 [ACK] Seq=57 Ack=17 Win=65536 Len=0 TSval=766418606 TSecr=766418606
12	0.001849789	127.0.0.1	127.0.0.1	TCP	94	56098 → 36895 [PSH, ACK] Seq=57 Ack=17 Win=65536 Len=28 TSval=766418607 TSecr=766418606
13	0.001849325	127.0.0.1	127.0.0.1	TCP	66	36895 → 56098 [ACK] Seq=17 Ack=85 Win=65536 Len=0 TSval=766418607 TSecr=766418607
14	0.001951702	127.0.0.1	127.0.0.1	TCP	74	36895 → 56098 [PSH, ACK] Seq=17 Ack=85 Win=65536 Len=8 TSval=766418607 TSecr=766418607
15	0.001962721	127.0.0.1	127.0.0.1	TCP	66	56098 → 36895 [ACK] Seq=85 Ack=25 Win=65536 Len=0 TSval=766418607 TSecr=766418607
16	0.002067491	127.0.0.1	127.0.0.1	TCP	94	56098 → 36895 [PSH, ACK] Seq=85 Ack=25 Win=65536 Len=28 TSval=766418607 TSecr=766418607
17	0.002076672	127.0.0.1	127.0.0.1	TCP	66	36895 → 56098 [ACK] Seq=25 Ack=113 Win=65536 Len=0 TSval=766418607 TSecr=766418607
18	0.002169388	127.0.0.1	127.0.0.1	TCP	74	36895 → 56098 [PSH, ACK] Seq=25 Ack=113 Win=65536 Len=8 TSval=766418607 TSecr=766418607
19	0.002177198	127.0.0.1	127.0.0.1	TCP	66	56098 → 36895 [ACK] Seq=113 Ack=33 Win=65536 Len=0 TSval=766418607 TSecr=766418607
20	0.002280609	127.0.0.1	127.0.0.1	TCP	94	56098 → 36895 [PSH, ACK] Seq=113 Ack=33 Win=65536 Len=28 TSval=766418607 TSecr=766418607
21	0.002286740	127.0.0.1	127.0.0.1	TCP	66	36895 → 56098 [ACK] Seq=33 Ack=141 Win=65536 Len=0 TSval=766418607 TSecr=766418607
22	0.002381646	127.0.0.1	127.0.0.1	TCP	74	36895 → 56098 [PSH, ACK] Seq=33 Ack=141 Win=65536 Len=8 TSval=766418607 TSecr=766418607
23	0.002392659	127.0.0.1	127.0.0.1	TCP	66	56098 → 36895 [ACK] Seq=141 Ack=41 Win=65536 Len=0 TSval=766418607 TSecr=766418607
24	0.002492508	127.0.0.1	127.0.0.1	TCP	94	56098 → 36895 [PSH, ACK] Seq=141 Ack=41 Win=65536 Len=28 TSval=766418607 TSecr=766418607
25	0.002501309	127.0.0.1	127.0.0.1	TCP	66	36895 → 56098 [ACK] Seq=41 Ack=169 Win=65536 Len=0 TSval=766418607 TSecr=766418607
26	0.002589032	127.0.0.1	127.0.0.1	TCP	74	36895 → 56098 [PSH, ACK] Seq=41 Ack=169 Win=65536 Len=8 TSval=766418607 TSecr=766418607
27	0.002596730	127.0.0.1	127.0.0.1	TCP	66	56098 → 36895 [ACK] Seq=169 Ack=49 Win=65536 Len=0 TSval=766418607 TSecr=766418607
28	0.002609835	127.0.0.1	127.0.0.1	TCP	94	56098 → 36895 [PSH, ACK] Seq=169 Ack=49 Win=65536 Len=28 TSval=766418607 TSecr=766418607
29	0.002766785	127.0.0.1	127.0.0.1	TCP	66	36895 → 56098 [ACK] Seq=49 Ack=197 Win=65536 Len=0 TSval=766418607 TSecr=766418607
30	0.002785918	127.0.0.1	127.0.0.1	TCP	74	36895 → 56098 [PSH, ACK] Seq=49 Ack=197 Win=65536 Len=8 TSval=766418608 TSecr=766418607
31	0.002802457	127.0.0.1	127.0.0.1	TCP	66	56098 → 36895 [ACK] Seq=197 Ack=57 Win=65536 Len=0 TSval=766418608 TSecr=766418608
32	0.002915712	127.0.0.1	127.0.0.1	TCP	94	56098 → 36895 [PSH, ACK] Seq=197 Ack=57 Win=65536 Len=28 TSval=766418608 TSecr=766418608
33	0.002927279	127.0.0.1	127.0.0.1	TCP	66	36895 → 56098 [ACK] Seq=57 Ack=225 Win=65536 Len=0 TSval=7664

Screenshot of Client side code

```
Sending Character: "\0" (ASCII 0)
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
Data to be sent = 0000
Data with remainder = 0000 000
[+]User message sent successfully.
MD5 client:      35899082e51edf667f14477ac000cbba
MD5 server:     35899082e51edf667f14477ac000cbba
[+]MD5 Matched
[+]Closing the connection.
```

MD5sums matched => Cyclic Redundancy Check is working efficiently.

Screenshot of Server side code

```
-----
Data and remainder received: 0000 000
Data Received: 0000      Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 0000 000
Data Received: 0000      Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 0000 000
Data Received: 0000      Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 0000 000
Data Received: 0000      Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 0000 000
Data Received: 0000      Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 0000 000
Data Received: 0000      Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 0000 000
Data Received: 0000      Remainder after CRC checking: 000
Data was received properly

Data and remainder received: 0000 000
Data Received: 0000      Remainder after CRC checking: 000
Data was received properly

Character "\0" (ASCII 0) was received from client
-----
[+]Data received successfully.
Message sent by client:
    "Lorem ipsum dolor sit amet, consectetur adipiscing elit."
MD5:    35899082e51edf667f14477ac000cbba
[+]MD5 sent
[+]Closing the connection.
```

-----X-----X-----