

Project 3 TCP and Error Handling

By Srujan Vepuri



November 26, 2022

UAlbany

Contents

[System documentation 2](#_Toc120393586)

[A high-level data flow diagram for the system 2](#_Toc120393587)

[A list of routines and their brief descriptions 4](#_Toc120393588)

[Implementation details 5](#_Toc120393589)

[Test documentation 7](#_Toc120393590)

[How you tested your program 7](#_Toc120393591)

[Testing outputs 7](#_Toc120393592)

[User documentation 8](#_Toc120393593)

[How to run your program 8](#_Toc120393594)

[Describe parameter 9](#_Toc120393595)

# System documentation

## A high-level data flow diagram for the system

Diagram

Description automatically generated

This figure shows my data flow diagram.

## A list of routines and their brief descriptions

* My client side will provide their input in the intext.txt file
  + Frist My encode will handle all CRC, hamming error and fixing using Hamming error.
  + It will be reasonable for making trying into code and allowing the user to put where they error to be.
  + They make a binary file
  + My client TCP will read this encode part then pass all bits to my Server side.
  + After Server has done counting and truing A I E O U to capitals letters counts digits.
  + Then it calls Decoder which just removes CRC, Hamming, if any errors found than fix those errors.
  + Then it will make result.txt file and turn every bit to char.
* Server will launch (would running first before client).
  + It will listen and pick those bits quickly and write them down to temp file.
  + Then it will call Server Decoder
    - It will find errors using hamming code error detection than print at what index the error was
  + After server Decoder is done It will call ServerPthreads than will make 7 threads and run the rest of code.
  + This will provide us with resultsServer.txt
  + Then my server will call Encoder pass this resutslServer.txt file
  + It will be reasonable for making trying into code and allowing the user to put where the error to be.
  + Then it will produce a binary file.
  + After Encoding it will grab that file and call and send to client
  + Then waits for a new client.

## Implementation details

1. First, I made My TCP server and client. I made sure that I can send Hello and receive Hello and send hello.
2. Then I Focused on Encoding my text file.
   1. I Modified my encode.c from project 2. Now it takes in filename and it does add parity bit right away.
   2. Then I made new class called Parity bit where It should Handle CRC.
   3. Then I made Hamming code where GRADER can Add error bit into my code Where they have chosen what index they want the error to be then it will go head and attack.
3. Then Went to back to Client started adding How client will send data through TCP
   1. I first made new service calls which is described in #2.
   2. Then Best Thought will go read all bits and save t buffer. Then the buffer will pass through socket to client.
4. Then went fixed server to accept a huge load data by making buffer a bigger size.
5. Then I used the same from project 2 to decode. But Now I have deal with hamming error
   1. Frist, I counted know which index are bits and which are parity bits.
   2. Then I count and bits
   3. After couniting all bits (size of 12)
   4. Then I get reminder of those bits and cross check with parity bits
      1. If an error, then I just add up parity bits where It does not much index and count mod 2.
      2. Then go to that index and flip those bits.
      3. Then it will change those bits and new output file
6. After getting results and fixing errors from #5 we will go simple decode and start removing all other bits till we only have left with pure bits
   1. Once we char bits then we turn each 8 bits to one char value and write that value to serverinputfiel.txt
7. Then Now we our message now I worked with p-threads
   1. Frist, I made 7 methods that describe in pdf file for project 3.
   2. Then made 5 char arrays with size 7 the last index being to end str=’= ‘\0’
   3. Then I made 7 mutex
   4. I used the first mutex for CharA than my second CharE and so on till to writer.
   5. Then the only way each index can access is once the other array has copied the previous array. Once copied then it let sem\_post() index to start.
   6. Then Write will save and write to ServerEncoder.
   7. Once they read the last chart in txt file then the digit will give another message to pass. Or rather mutex to add total number of digits in the txt file.
8. Once All done the server will start encoder process same process as described #2
9. It will send it to the TCP socket.
10. Then the client will decode check for errors. If found, then fixes those errors. Then it will start to return to a new binary file.
11. Then we are left pure bits which turn into char and all write into result.txt

# Test documentation

How testing went.

## How you tested your program

* Frist, I made new text file where the file had “Hello 123”.
  + In here I mostly tested How hamming code and homing code error handles. Error detection.
  + I made sure that hamming code produce the right results where I also calculated those results by hand.
  + I also tested that errors bits are presented when binary is being accessed by decoders
  + I also tested that It has same error and fix those errors.
  + I made way to add errors after code has been code.
  + Then ran into problems with Pthread.
* So, now I copied all text pdf project 3
  + Then Now, I fixed the problem with 7 being array size.
  + Then I add to surer that total char/7 ==0 and when it does not equal to 0 that no errors and make sure That fork and execv do not cause or create a zombie child.
* I also missed with binary files by adding 2 errors back-to-back (Which will sometimes not pick up the errors)

## Testing outputs

* From Pdf file I used project 3.
* I also made my test files where I had numbers and chars lots of A E I O U.

# User documentation

Tells the user what to do.

## How to run your program

This time I have easier run my code.

Please run this code on Linux Manichee and also make sure you have 2 terminals to work since it TCP socket

* Cd to my folder
  1. Type in “make” and press enter
  2. In one of your terminals
     + You run “make runServer” or “./s”
     + Make sure server run firsts
  3. In your other terminal
     + You can run “make runClient” or “./c”
  4. An example of this would be:

Text

Description automatically generated

## Describe parameter

* Yes, we have few parameters First the suer must provide a “intext.txt” file in order to make the code run. The text file much more than 7 characters in that file (it does matter what it in the file)
* Text

  Description automatically generated
  + This the best and safe way for the user to enter error data into the binary files.
  + This latter will be picked up the error.