Computer Network Lab

Name: Tonmoy Biswas

Roll No: 002110501133

Class: BCSE 3rd Year 1st Sem

Section: A3

Assignment No: 1

Deadline: 13/08/2023

Submission: 14/08/2023

## **Problem Statement: Design and implement an error detection module which has two schemes namely Checksum and Cyclic Redundancy Check(CRC).**

### **Design:**

This program is designed to implement Checksum and Cyclic Redundancy Check(CRC) to detect error in data transmission. At server side we extract the data word from a given text file containing a sequence of 0 and 1 and create codeword using the Checksum and CRC scheme and send the codeword to the server. At server side, the server using Checksum and CRC scheme check whether there is an error or not.

To simulate a noise channel we create a error injection module which is called at the sender side to inject error in the codeword.

Checksum and CRC module

Error Injection module

Checksum and CRC module



Error Delection

**Server**

Received Codeword

Input.txt

Erroneous Codeword

codeword

dataword

**Client**

**Schematic Diagram**

# **Implementation:**

CheckSum Scheme:

//return the checksum string of word\_len length

module.exports.getChecksum= function getChecksum(wordList,word\_len){

    let sum=0;

    let limit=(1<<word\_len);

    for(i of wordList){

        sum+=parseInt(i,2);//convert a binary string into decimal

        if(sum>=limit){//convert to wrap sum

            sum=sum%limit+1;//convert a decimal into binary string

        }

    }

    sum=sum.toString(2);

    //make the length equal to word\_len

    while(sum.length!=word\_len){

        sum='0'+sum;

    }

    let checksum="";

    //1's complement to get the checksum

    for(let i=0;i<word\_len;i++){

        if(sum.charAt(i)=="1"){

            checksum=checksum+"0";

        }

        else{

            checksum=checksum+"1";

        }

    }

    return checksum;

};

//return the codeword string

module.exports.getChecksumCodeWord= function getChecksumCodeWord(dataword\_raw,word\_len){

    //make the length multiple of word\_len(in our case 16)

    let dataword=dataword\_raw;

    // console.log("data word length ",dataword.length);

    while(dataword.length%word\_len!=0){

        dataword='0'+dataword;

    }

    let wordList=[];//will contain the 16 bit words of the dataword

    for(let i=0;i<(dataword.length/word\_len);i++){

        wordList.push(dataword.substring(i\*word\_len,(i+1)\*word\_len));

    }

    let checksum=this.getChecksum(wordList,word\_len);

    // console.log("binary check sum",checksum,"decimal checksum",parseInt(checksum,2));

    let checksum\_codeword=dataword\_raw+checksum;

    return checksum\_codeword;

};

//return boolean

module.exports.isValidChecksum=function isValidChecksum(codeword\_raw,word\_len){

    let codeword=codeword\_raw;

    // console.log("data word length ",codeword.length);

    while(codeword.length%word\_len!=0){

        codeword='0'+codeword;

    }

    let wordList=[];//will contain the 16 bit words of the codeword

    for(let i=0;i<(codeword.length/word\_len);i++){

        wordList.push(codeword.substring(i\*word\_len,(i+1)\*word\_len));

    }

    let checksum=this.getChecksum(wordList,word\_len);

    // console.log("binary check sum",checksum,"decimal checksum",parseInt(checksum,2));

    return parseInt(checksum,2)==0;

};

CRC Scheme:

// return the codeword string

module.exports.getCRCCodeWord= function(dataword\_raw,generator){

    let dataword=dataword\_raw;

    //creating augmented data word

    for(let i=0;i<generator.length-1;i++){

        dataword=dataword+"0";

    }

    //created augmented polynomial

    let polyarr=new Array(dataword.length);

    polyarr.fill(0);

    let l=dataword.length-1;

    for(let i=0;i<=l;i++){

        if(dataword.charAt(l-i)=="1"){

            polyarr[i]=1;

        }

    }

    let g=generator.length-1;// g is the degree of the polynomial as well as the length of the remainder

    //creating generator polynomial

    let garr=new Array();

    for(let i=0;i<=g;i++){

        if(generator.charAt(g-i)=="1"){

            garr.push(i);

        }

    }

    //calculating the remainder

    for(let i=l;i>=g;i--){

        if(polyarr[i]==1){

            let x=i-g;

            for(let ele of garr){

                polyarr[ele+x]=polyarr[ele+x]^1;

            }

        }

    }

    //getting the binary string remainder

    let remainder="";

    for(let i=0;i<g;i++){

        if(polyarr[i]==1){

            remainder="1"+remainder;

        }

        else{

            remainder="0"+remainder;

        }

    }

    let codeword=dataword\_raw+remainder;

    return codeword;

};

//return boolean

module.exports.isValidCRC= function(codeword\_raw,generator){

    let dataword=codeword\_raw;

    //creating polynomial of codeword

    let polyarr=new Array(dataword.length);

    polyarr.fill(0);

    let l=dataword.length-1;

    for(let i=0;i<=l;i++){

        if(dataword.charAt(l-i)=="1"){

            polyarr[i]=1;

        }

    }

    let g=generator.length-1;// g is the degree of the polynomial as well as the length of the remainder

    //creating generator polynomial

    let garr=new Array();

    for(let i=0;i<=g;i++){

        if(generator.charAt(g-i)=="1"){

            garr.push(i);

        }

    }

    //calculating the remainder

    for(let i=l;i>=g;i--){

        if(polyarr[i]==1){

            let x=i-g;

            for(let ele of garr){

                polyarr[ele+x]=polyarr[ele+x]^1;

            }

        }

    }

    //getting the binary string remainder

    let remainder="";

    for(let i=0;i<g;i++){

        if(polyarr[i]==1){

            remainder="1"+remainder;

        }

        else{

            remainder="0"+remainder;

        }

    }

    // let codeword=dataword\_raw+remainder;

    // return codeword;

    // console.log(remainder);

    return parseInt(remainder,2)==0;

};

Error Injection Module:

function randomIntFromInterval(min, max) { // min and max included

    return Math.floor(Math.random() \* (max - min + 1) + min)

}

module.exports.injectSingleError=function(codeword\_raw){

    let len=codeword\_raw.length;

    let randomIndex=randomIntFromInterval(0,len-1);

    if(codeword\_raw.charAt(randomIndex)=="1"){

        return codeword\_raw.replaceAt(randomIndex,"0");

    }

    else{

        return codeword\_raw.replaceAt(randomIndex,"1");

    }

};

module.exports.injectTwoIsolatedError=function(codeword\_raw){

    let len=codeword\_raw.length;

    let randomIndex=randomIntFromInterval(0,len-1);

    if(codeword\_raw.charAt(randomIndex)=="1"){

        codeword\_raw=codeword\_raw.replaceAt(randomIndex,"0");

    }

    else{

        codeword\_raw=codeword\_raw.replaceAt(randomIndex,"1");

    }

    let ranInd=randomIntFromInterval(2,len-2);

    randomIndex=(randomIndex+ranInd)%len;

    if(codeword\_raw.charAt(randomIndex)=="1"){

        codeword\_raw=codeword\_raw.replaceAt(randomIndex,"0");

    }

    else{

        codeword\_raw=codeword\_raw.replaceAt(randomIndex,"1");

    }

    return codeword\_raw;

}

module.exports.injectOddError=function(codeword\_raw){

    let len=codeword\_raw.length;

    let no=randomIntFromInterval(0,len);

    if(no%2==0){

        no=(no+1)%len;

    }

    let mark=new Array(len);

    mark.fill(0);

    let count=0;

    while(count<no){

        let randomIndex=randomIntFromInterval(0,len-1);

        if(mark[randomIndex]==0){

            mark[randomIndex]=1;

            count++;

            if(codeword\_raw.charAt(randomIndex)=="1"){

                codeword\_raw=codeword\_raw.replaceAt(randomIndex,"0");

            }

            else{

                codeword\_raw=codeword\_raw.replaceAt(randomIndex,"1");

            }

        }

    }

    return codeword\_raw;

};

module.exports.injectBurstError=function(codeword\_raw,error\_len){

    if(error\_len==0){

        return codeword\_raw;

    }

    let len=codeword\_raw.length;

    error\_len=Math.min(len,error\_len);

    let randomIndex=randomIntFromInterval(0,len-error\_len);

    let newError="";

    for(let i=0;i<error\_len;i++){

        if(codeword\_raw.charAt(i+randomIndex)=="1"){

            codeword\_raw=codeword\_raw.replaceAt(i+randomIndex,"0");

        }

        else{

            codeword\_raw=codeword\_raw.replaceAt(i+randomIndex,"1");

        }

    }

    return codeword\_raw;

};

module.exports.bypasscrc=function(codeword,generator){

    if(codeword.length>generator.length){

        return codeword;

    }

    let l=generator.length;

    for(let i=0;i<l;i++){

        let codeind=codeword.length-1-i;

        let generatorind=generator.length-1-i;

        if(generator.charAt(generatorind)=="1"){

            if(codeword.charAt(codeind)=="1"){

                codeword=codeword.replaceAt(codeind,"0");

            }

            else{

                codeword=codeword.replaceAt(codeind,"1");

            }

        }

    }

    return codeword;

}

module.exports.bypasschecksum=function(codeword\_raw,word\_len){

    let codeword=codeword\_raw;

    // console.log("data word length ",dataword.length);

    while(codeword.length%word\_len!=0){

        codeword='0'+codeword;

    }

    let wordList=[];//will contain the 16 bit words of the dataword

    for(let i=0;i<(codeword.length/word\_len);i++){

        wordList.push(codeword.substring(i\*word\_len,(i+1)\*word\_len));

    }

    let l=wordList.length;

    // let flag=false;

    for(let i=0;i<word\_len;i++){

        let one=0;

        let zero=0;

        for(let j=1;j<l;j++){

            if(wordList[j].charAt(i)=="1"){

                one++;

            }

            else{

                zero++;

            }

        }

        if(zero>0&&one>0){

            let oneind=0;

            for(let j=1;j<l;j++){

                if(wordList[j].charAt(i)=="1"){

                    oneind=j;

                    break;

                }

            }

            let zeroind=0;

            for(let j=1;j<l;j++){

                if(wordList[j].charAt(i)=="0"){

                    zeroind=j;

                    break;

                }

            }

            wordList[oneind]=wordList[oneind].replaceAt(i,"0");

            wordList[zeroind]=wordList[zeroind].replaceAt(i,"1");

            break;

        }

    }

    let newCodeword="";

    for(let str of wordList){

        newCodeword=newCodeword+str;

    }

    return newCodeword.substring(newCodeword.length-codeword\_raw.length);

}

# **Test Cases and Results:**

Below are some test cases for which the schemes are failed to detect error.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Codeword | Received  Codeword | Checksum | CRC8 | CRC10 | CRC16 | CRC32 |
| 10010010110011100000110010  10110101101001011000111111  011100100(For Checksum) | 10010010110010100000110010  10110101101001011001111111  011100100(For Checksum) | Not detected | N/A | N/A | N/A | N/A |
| 10010010110011100000110010  101101011010010110010011011  (For CRC-8) | 1001001011001110000011001  0101101011010010110010011011  (For CRC-8) | N/A | Not detected | N/A | N/A | N/A |
| 10010010110011100000110010  10110101101001011000001011011  (For CRC-10) | 1001001011001110000011001  010110101101001011000001011011  (For CRC-10) | N/A | N/A | Not detected | N/A | N/A |
| 1001001011001110000011001010  110101101001011001001101111101110  (For CRC-16) | 10010010110011100000110010  10110101101001011001001101111101110  (For CRC-16) | N/A | N/A | N/A | Not detected | N/A |
| 100100101100111000001100101011  01011010010110001000011011001100111111110010111  (For CRC-32) | 100100101100111000001100  10101101011010010110001000011011001100111111110010111  (For CRC-32) | N/A | N/A | N/A | N/A | Not detected |

# **Analysis:**

From the above test cases we can see that all the scheme has some limitation.

For the checksum scheme if equal number of 1s and 0s get flipped at the same bit position during transmission then the checksum remains same but it introduced error in the codeword. Since the chechsum remains same the error is not detected at the server side.

For the CRC scheme if the error is divisible by the dividend(generator polynomial) then the error is not detected at the server side.

# **Comments:**

Through this assignment I have learnt the Checksum and Cyclic Redundancy Check(CRC) scheme for error detection in data communication and there importance in data transmission and also how it is implemented in physical layer. Also I come to know that when this schemes get failed to detect error.

Implementing the error generating algorithm that will not be detected by the schemes were a tricky part and gives me a deep understanding of the schemes.