

Experiment-7

AIM: Write a program to implement flow control at data link layer using Sliding window protocol. Simulate the flow of frames from one node to another.

Program:

Program should achieve at least below given requirements. You can make it a bidirectional program wherein receiver is sending its data frames with acknowledgement (Piggybacking).



Main.java



Share

Run

```

1 import java.util.*;
2 class HammingCodeExample {
3     public static void main(String args[])
4     {
5         int size, hammingCodeSize, errorPosition;
6         int arr[];
7         int hammingCode[];
8         Scanner sc = new Scanner(System.in);
9         System.out.println("Enter the bits size for the data.");
10        size = sc.nextInt();
11        arr = new int[size];
12        for(int j = 0 ; j < size ; j++)
13        {
14            System.out.println("Enter " + (size - j) + "-bit of the data:");
15            arr[size - j - 1] = sc.nextInt();
16        }
17        System.out.println("The data which you enter is:");
18        for(int k = 0 ; k < size ; k++) {
19            System.out.print(arr[size - k - 1]);
20        }
21        System.out.println();
22        hammingCode = getHammingCode(arr);
23        hammingCodeSize = hammingCode.length;
24        System.out.println("The hamming code generated for your data is:");
25        for(int i = 0 ; i < hammingCodeSize; i++)
26        {
27            System.out.print(hammingCode[(hammingCodeSize - i - 1)]);
28        }
29        System.out.println();
30        System.out.println("For detecting error at the reciever end, enter position of a
        bit to alter original data "
31        + "(0 for no error):");
32        errorPosition = sc.nextInt();
33        sc.close();
34        if(errorPosition != 0) {
35            hammingCode[errorPosition - 1] = (hammingCode[errorPosition - 1] + 1) % 2;

```

Main.java



Share

Run

```
36     }
37     System.out.println("Sent Data is:");
38     for(int k = 0; k < hammingCodeSize; k++) {
39         System.out.print(hammingCode[hammingCodeSize - k - 1]);
40     }
41     System.out.println();
42     receiveData(hammingCode, hammingCodeSize - arr.length);
43 }
44 static int[] getHammingCode(int data[]) {
45     int returnData[];
46     int size;
47     int i = 0, parityBits = 0, j = 0, k = 0;
48     size = data.length;
49     while(i < size) {
50         if(Math.pow(2, parityBits) == (i + parityBits + 1)) {
51             parityBits++;
52         }
53         else {
54             i++;
55         }
56     }
57     returnData = new int[size + parityBits];
58     for(i = 1; i <= returnData.length; i++) {
59         if(Math.pow(2, j) == i) {
60
61             returnData[(i - 1)] = 2;
62             j++;
63         }
64         else {
65             returnData[(k + j)] = data[k++];
66         }
67     }
68     for(i = 0; i < parityBits; i++) {
69
70         returnData[((int) Math.pow(2, i)) - 1] = getParityBit(returnData, i);
71     }
```

Main.java

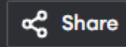


Share

Run

```
73     return returnData;
74 }
75 static int getParityBit(int returnData[], int pow) {
76     int parityBit = 0;
77     int size = returnData.length;
78
79     for(int i = 0; i < size; i++) {
80         if(returnData[i] != 2) {
81             int k = (i + 1);
82             String str = Integer.toBinaryString(k);
83             int temp = ((Integer.parseInt(str)) / ((int) Math.pow(10, pow))) % 10;
84             if(temp == 1) {
85                 if(returnData[i] == 1) {
86                     parityBit = (parityBit + 1) % 2;
87                 }
88             }
89         }
90     }
91     return parityBit;
92 }
93
94 static void receiveData(int data[], int parityBits) {
95     int pow;
96     int size = data.length;
97     int parityArray[] = new int[parityBits];
98     String errorLoc = new String();
99     for(pow = 0; pow < parityBits; pow++) {
100         for(int i = 0; i < size; i++) {
101             int j = i + 1;
102
103             String str = Integer.toBinaryString(j);
104
105             int bit = ((Integer.parseInt(str)) / ((int) Math.pow(10, pow))) % 10;
106             if(bit == 1) {
107                 if(data[i] == 1) {
108                     parityArray[pow] = (parityArray[pow] + 1) % 2;
```

Main.java



Run

```
108         parityArray[pow] = (parityArray[pow] + 1) % 2;
109     }
110 }
111 }
112     errorLoc = parityArray[pow] + errorLoc;
113 }
114     int finalLoc = Integer.parseInt(errorLoc, 2);
115     if(finalLoc != 0) {
116         System.out.println("Error is found at location " + finalLoc + ".");
117         data[finalLoc - 1] = (data[finalLoc - 1] + 1) % 2;
118         System.out.println("After correcting the error, the code is:");
119         for(int i = 0; i < size; i++) {
120             System.out.print(data[size - i - 1]);
121         }
122         System.out.println();
123     }
124     else {
125         System.out.println("There is no error in the received data.");
126     }
127     System.out.println("The data sent from the sender:");
128     pow = parityBits - 1;
129     for(int k = size; k > 0; k--) {
130         if(Math.pow(2, pow) != k) {
131             System.out.print(data[k - 1]);
132         }
133         else {
134             pow--;
135         }
136     }
137     System.out.println();
138 }
139 }
140
```

Output:

```
Output Clear
java -cp /tmp/VlgXxu7Hfd/HammingCodeExample
Enter the bits size for the data.
5
Enter 5-bit of the data:
1
Enter 4-bit of the data:
2
Enter 3-bit of the data:
3
Enter 2-bit of the data:
4
Enter 1-bit of the data:
5
The data which you enter is:
12345
The hamming code generated for your data is:
112340501
For detecting error at the reciever end, enter position of a bit to alter original data (0 for no
error):

=== Session Ended. Please Run the code again ===
```

Result:

The program to implement flow control at data link layer using Sliding window protocol. Simulate the flow of frames from one node to another has been executed successfully.