

## Assignment 3: Implement p-persistent CSMA and CSMA/CD.

(Carrier-Sense Multiple Access with Collision Detection)

### 1) What is CSMA/CD?

CSMA/CD is a media access control method. It uses carrier-sensing to defer transmissions until no other stations are transmitting. This is used in combination with collision detection in which a transmitting station detects collisions by sensing transmissions from other stations while it is transmitting a frame. When this collision condition is detected, the station stops transmitting that frame, transmits a jam signal, and then waits for a random time interval which is a number between 0 to  $2^k-1$ , before trying to resend the frame.

```
4 #include <math.h>
5 #include <ncurses.h>
6
7 int channel[50]={0},n,tf=5,ttemp[10],a=0,k=2,w=0;
8 int ds[10]={1,1,1,1,1,1,1,1,1,1},is[10]={0},js[10]={0};
9 int status[10]={1,1,1,1,1,1,1,1,1,1};
10
11 typedef struct
12 {int data[5];
13  int dest;
14  int time;}station;
15
16 void send(int sno);
17 int stat_ini();
18 int send_data();
19 void collision();
20
21 int backoff()
22 {
23     int kmax=15,r,s,tb;
24     if(k>kmax)
25     {
26         printf("station exceeded its limit\n%d:",k);
27         exit(0);
28     }
29     else
30     {
31         s=(pow(2,k)-1);
32         r=rand()%s;
33         tb=r*tf;
34         printf(" 2^k-1: %d\n",s);
35         printf("The random no generated %d ",r);
36         k=k+1;
37     }
38     return (tb);
```

A channel array is taken and is initialised to 50.

A structure station is declared which has an array containing the data, and integer denoting the destination and the time.

The function backoff returns the time for which the station will back off once a collision has happened. It has a variable k and a random number is generated between 0 and  $2^k-1$  for which the station will back off.

```

42 station stat[10];
43 int main()
44 {
45     int i,j,ch;
46     //printf("*****\n\tCARRIER SENSE MULTIPLE ACCESS WITH COLLISION DETECTIO
47     //printf("\t\t\t\t\tCSMA|CD\t\t\t\t\t\n");
48     //printf("*****\n\t\t\t\t\tCSMA|CD\t\t\t\t\t\n");
49     printf("Enter the number of stations :");
50     scanf("%d",&n);
51     for(i=1;i<=n;i++)
52     {printf("\nEnter 1, If station%d want to transmit signal :",i);
53       scanf("%d",&ch);
54       stat[i].time=0;
55       if(ch==1)
56       {
57         printf("\nEnter time of sending signal of station%d :",i);
58         scanf("%d",&stat[i].time);
59         a++;
60         status[i]=0;
61         l:printf("\nEnter destination of station%d :",i);
62         scanf("%d",&stat[i].dest);
63         if(stat[i].dest==i||stat[i].dest<1||stat[i].dest>n)
64         {printf("Wrong destination, Try again\n");
65           goto l;}
66         printf("\nEnter the data(3-Bit data) :");
67         for(j=1;j<=3;j++)
68         {scanf("%d",&stat[i].data[j]);}
69       }
70     }
71     for(i=1;i<=n;i++)
72     {ttemp[i]=stat[i].time;}
73     send_data();
74     return 0;
75 }
76 }

```

A channel is taken which can have a maximum of 10 stations.

In the main function the input of the number of stations is taken and for each corresponding station the time, destination and data is also taken. The send\_data function is called for each stations that wants to send.

```

78 void collision(int sno)
79 {
80     int i,tb;
81     for(i=1;i<=n;i++)
82     {
83         if(status[i]!=1)
84         {
85             stat[i].time=ttemp[i];
86             is[i]=0,js[i]=0,ds[i]=1;
87         }
88     }
89     for(i=1;i<=(n*5);i++)
90     channel[i]=0;
91     printf("signal jam\n");
92     tb=backoff();
93     stat[sno].time=stat[sno].time+tb;
94     printf("backoff by: %d",tb);
95     send_data();
96 }
97 }

```

Function to determine if collision has occurred for a station number. If the collision has occurred signal jam will be printed and it will back off and will try to send the data again.

```

99 int stat_ini() //to select the least time
100 {
101     int x,i,min;
102     for(i=1;i<=n;i++)
103     {
104         if(stat[i].time!=0)
105         {
106             min=i;
107             break;
108         }
109     }
110     for(int i=0;i<n;i++)
111     {
112         printf("%d stat[i].time",stat[i].time);
113     }
114     printf("\n");
115     x=stat[min].time;
116     for(i=1;i<=n;i++)
117     {
118         if(x>stat[i].time&&stat[i].time!=0)
119             x=stat[i].time;}
120     printf("Time: %d\n",x);
121     return x;
122 }

```

This function selects the station with the least time which wants to send data. The minimum time is returned.

```

125 int send_data() //to the state which sends data
126 {
127     int tmin=stat_ini(),t,i,tb;
128     for(t=tmin;;t++)
129     {
130         for(i=1;i<=n;i++)
131         {
132             if(t==stat[i].time)
133                 //printf("\nstation:%d",i);
134                 if(ds[i]==1&&channel[i*5-4]!=0)
135                 {
136                     tb=backoff();
137                     printf("channel not free backing off by:%dsec",tb);
138                     stat[i].time=stat[i].time+tb;
139                 }
140             else
141             {
142                 send(i);
143                 stat[i].time++;}
144         }
145     }
146     printf("\n");
147     for(i=1;i<=(n*5);i++)
148         printf("%d",channel[i]);
149     getch();
150 }
151 }
152 }

```

This function uses the minimum time to find out which station has to send the data. If the channel is not free it waits by the back off time and the send function is called again but with increasing value of the time.

```

154 void send(int sno)
155 {
156     int databit=0;
157     if(ds[sno]<=3)
158         databit=stat[sno].data[ds[sno]];
159     printf("\nDatabit=%d\nsno:%d\n",databit,sno);
160     if(is[sno]==0 && js[sno]==0) //sending for first time
161         {is[sno]=js[sno]=sno*5-4;
162         channel[sno*5-4]=databit;}
163     else
164     {
165         if(channel[is[sno]+1]!=0 || channel[js[sno]-1]!=0)
166         {
167             printf("\nCollision!!\n");
168             getch();
169             collision(sno);
170         }
171     }
172     else
173     {if(ds[sno]<=3)
174     {
175         int x;
176         if(sno!=n)
177         { if(is[sno]<=(n*5)-1 || stat[sno].dest!=n)
178         {is[sno]++;}
179         for(x=is[sno];x>sno*5-4;x--)
180             channel[x]=channel[x-1];
181         }
182         if(sno!=1)
183         {if(js[sno]>=2 || stat[sno].dest==1)
184         {js[sno]--;}
185         for(x=js[sno];x<sno*5-4;x++)
186             channel[x]=channel[x+1];
187         }
188         channel[sno*5-4]=databit;
189     }
190     else
191     {
192         int x;
193         if(sno!=n)
194         {
195             if(is[sno]<=(n*5)-1 || stat[sno].dest!=n)
196                 is[sno]++;
197             x=is[sno];
198             int j;
199             for(j=1;j<=3;x--)
200                 {channel[x]=channel[x-1];j++;}
201             if(x!=(sno*5-4) || x==1)
202                 {channel[x]=0;}
203         }
204         if(sno!=1)
205         {
206             if(js[sno]>=2 || stat[sno].dest==1)
207                 js[sno]--;
208             x=js[sno];
209             int j;
210             for(j=1;j<=3;x++)
211                 {channel[x]=channel[x+1];
212                 j++;}
213             channel[x]=0;
214         }
215         if(channel[stat[sno].dest*5-3]==stat[sno].data[3])
216             {{printf("\n Data sending succesfull.\n");
217             status[sno]=1;
218             w++;
219             getch();}
220             if(a==w)
221                 {printf("All signals sent successfully\n");
222                 exit(0);}}
223             channel[sno*5-4]=databit;}}
224     }
225     ds[sno]++;
226 }

```

The send function is actually responsible for sending the data of the station which is given as an argument. The channel was initialised to 0. If for a particular bit the channel is not 0 that means it already in use by some other station then collision has occurred. This is the part where collision is detected

If the collision has not occurred the data bit in the channel will be set to the data. If the status of the stations are same then data sending is successful. When all the stations have sent their data all signals sent successfully will be printed.

Output:

```

(default) soumik@soumik-X555LAB:~/Documents/networks_lab/csma_cd$ ./a.out
Enter the number of stations :2

Enter 1, If station1 want to transmit signal :1
Enter time of sending signal of station1 :1
Enter destination of station1 :2
Enter the data(3-Bit data) :4
5
6

Enter 1, If station2 want to transmit signal :1
Enter time of sending signal of station2 :2
Enter destination of station2 :1
Enter the data(3-Bit data) :7
8
9
0 stat[i].time1 stat[i].time
Time: 1

Databit=4
sno:1

4000000000
Databit=5
sno:1

Databit=7
sno:2

5400070000
Databit=6
sno:1

Databit=8
sno:2

```

The main takes the input of number of stations, time, data and destination of each station. Since 2 stations are there, 10 is the size of the channel. At time  $t=1$  only the station 1 sends data. So its just 4. At time  $t=2$  both station 1 and 2 send data. So output of channel is 5400070000.

```

6540780000
Databit=0
sno:1

Databit=9
sno:2

Collision!!
signal jam
2^k-1: 3
The random no generated 1 backoff by: 50 stat[i].time1 stat[i].time
Time: 1

Databit=4
sno:1

4000000000
Databit=5
sno:1

5400000000
Databit=6
sno:1

6540000000
Databit=0
sno:1

0654000000
Databit=0
sno:1

0065400000
Databit=0
sno:1

0006540000
Databit=0
sno:1
2^k-1: 7
The random no generated 4 channel not free backing off by:20sec
0000654000
Databit=0

```

At time =4, both the stations want to change the 4<sup>th</sup> bit data from 0. So there is a collision. Therefore the backoff function will be called and a random number for which it will wait is generated. The data is sent again.

```

0000065400
Databit=0
sno:1
Data sending successful
000006540
Databit=0
sno:1
00000654
Databit=0
sno:1
0000065
Databit=0
sno:1
000006
Databit=0
sno:1
000000000
Databit=0
sno:1
000000000
Databit=0
sno:1
000000000
Databit=0
sno:1
000000000
Databit=0
sno:1
000000000
Databit=0
sno:1
000000000

```

The data from the station 1 is sent first. When all the bits are sent data sending successful is printed. Next the station 2 waits for the time as specified by the backoff function.

After the backoff time the data of station 2 will be sent again. This time there wont be any collision as the station 1 has already sent the data.

```

000000000
Databit=0
sno:1
000000000
Databit=0
sno:1
Databit=7
sno:2
0000070000
Databit=0
sno:1
Databit=8
sno:2
0000780000
Databit=0
sno:1
Databit=9
sno:2
0007890000
Databit=0
sno:1
Databit=0
sno:2
0078900000
Databit=0
sno:1
Databit=0
sno:2
0789000000
Databit=0
sno:1

```

```
Databit=0  
sno:2  
  
7890000000  
Databit=0  
sno:1  
  
Databit=0  
sno:2  
  
Data sending succesfull.  
All signals sent successfully  
(default) soumik@soumik-X555LAB
```

After all the data of the station 2 is sent All signals sent will be printed.