1. CRC

#include<bits/stdc++.h>

using namespace std;

bool acceptPacket(){

    return rand()%2;

}

int main(){

    int windowSize;

    cout<<"Enter size of the sliding window"<<endl;

    cin>>windowSize;

    int numberOfSN = windowSize + 1;

    int senderWindow[windowSize];

    vector<int>receiverWindow;

    bool nAck[numberOfSN] = {0};

    bool sendPackets = true;

    bool accepted[numberOfSN] = {0};

    int currentPacket =  0;

    do{

        for(int i = 0;i < windowSize;i++){

            senderWindow[i] = (currentPacket%numberOfSN);

            cout<<"Sender has sent "<<senderWindow[i]<<" packet"<<endl;

            currentPacket++;

        }

        cout<<"------------------------"<<endl;

        bool nAckFound = true;

        while(nAckFound){

            for(int i=0;i<windowSize;i++){

                int packetSequenceNumber = senderWindow[i];

                if((nAck[packetSequenceNumber] and acceptPacket()) or (!accepted[packetSequenceNumber] and acceptPacket())){

                    receiverWindow.push\_back(packetSequenceNumber);

                    cout<<packetSequenceNumber<<" has been accepted"<<endl;

                    nAck[packetSequenceNumber] = false;

                    accepted[packetSequenceNumber] = true;

                }

                else{

                    nAck[packetSequenceNumber] = true;

                    cout<<"Negative Acknowledgement Sent for "<<packetSequenceNumber<<endl;

                }

            }

            bool anyNAckExists = false;

            for(int i=0;i<windowSize;i++){

                int packetNumber = senderWindow[i];

                if(nAck[packetNumber])

                    anyNAckExists = true;

            }

            nAckFound = anyNAckExists;

        }

        memset(nAck,0,sizeof(nAck));

        memset(accepted,0,sizeof(accepted));

        receiverWindow.clear();

        cout<<"Want to send more packets ?"<<endl;

        cin>>sendPackets;

    }while(sendPackets);

}

2.Aloha

#include<bits/stdc++.h>

using namespace std;

bool nodeSelected(){

    return rand()%2;

}

void acceptNode(vector<int>&nodesTransmitting,bool ack[],int &completedTransmissions){

    for(int i: nodesTransmitting)

        cout<<"node "<<i<<" is trasmitting data"<<endl;

    if(nodesTransmitting.size() != 1){

        cout<<"Collision has occured"<<endl;

        return;

    }

    ack[nodesTransmitting[0]] = 1;

    completedTransmissions++;

}

int main(){

    int numberOfNodes;

    cout<<"Enter the number of nodes"<<endl;

    cin>>numberOfNodes;

    bool ack[numberOfNodes] = {0}; // Till now, no node has received the acknowledgement of the data being received.

    int completedTransmissions = 0; // No node has transmitted the data thus far. So no transmissions have taken place yet

    int timePeriod = 0;

    int numberOfTransmissionAttempts = 0;

    vector<int>nodesReadyToTrasmit; // Contains all the nodes that are currently sends the packets through the transmission channel

    while(completedTransmissions < numberOfNodes) { // Continue as long as all the nodes do not complete their transmission

        for(int i = 0;i < numberOfNodes; i++){

            if(!ack[i]) // If the packet has not yet been transmitted successfully, then

                if(nodeSelected()) // A node to be transmitted is selected with 50% probability, randomly

                    nodesReadyToTrasmit.push\_back(i);

        }

        acceptNode(nodesReadyToTrasmit,ack,completedTransmissions);

        numberOfTransmissionAttempts += nodesReadyToTrasmit.size();

        nodesReadyToTrasmit.clear();

        timePeriod++;

    }

    double g = (numberOfTransmissionAttempts\*1.0)/timePeriod;

    double coefficient = exp(-2\*g);

    double throughput = g \* coefficient;

    cout<<throughput<<endl;

}

2. CSMA

#include<iostream>

#include<cstdlib>

#include<vector>

#include<cmath>

#define NUM\_OF\_SLOTS 10000

using namespace std;

bool nodeSelected(){

    return rand()%2;

}

void acceptNode(vector<int>&nodesTransmitting,bool ack[],int &completedTransmissions){

    for(int i: nodesTransmitting)

        cout<<"node "<<i<<" is trasmitting data"<<endl;

    if(nodesTransmitting.size() != 1){

        cout<<"Collision has occured"<<endl;

        return;

    }

    ack[nodesTransmitting[0]] = 1;

    completedTransmissions++;

}

int main(){

    int numberOfNodes;

    cout<<"Enter the number of nodes"<<endl;

    cin>>numberOfNodes;

    bool ack[numberOfNodes] = {0}; // Till now, no node has received the acknowledgement of the data being received.

    int slot[NUM\_OF\_SLOTS];

    int completedTransmissions = 0; // No node has transmitted the data thus far. So no transmissions have taken place yet

    int timePeriod = 0;

    int numberOfTransmissionAttempts = 0;

    for(int i=0;i<NUM\_OF\_SLOTS;i++)

        slot[i] = (rand()%numberOfNodes);

    int currentSlot = 0;

    while((completedTransmissions < numberOfNodes) and (currentSlot < NUM\_OF\_SLOTS)) { // Continue as long as all the nodes do not complete their transmission

        timePeriod++;

        numberOfTransmissionAttempts++;

        int candidate = slot[currentSlot];

        cout<<candidate<<endl;

        if(!ack[candidate]){

            ack[candidate] = 1;

            completedTransmissions++;

        }

        currentSlot++;

    }

    double g = (numberOfTransmissionAttempts\*1.0)/timePeriod;

    double coefficient = exp(-2\*g);

    double throughput = g \* coefficient;

    cout<<throughput<<endl;

}