Datagram socket:

client.java

```
import java.io.*;
import java.net.*;
public class Client {
public static void main(String args[])throws Exception{
        DatagramSocket ds=new DatagramSocket(2345);
        String msg;
        byte[] buf=new byte[100];
        while(true)
        {
                DatagramPacket rdp=new DatagramPacket(buf,buf.length);
                ds.receive(rdp);
                msg=new String(rdp.getData(),rdp.getOffset(),rdp.getLength());
                System.out.println(msg);
        }
}
}
```

server.java

```
import java.io.*;
import java.net.*;
public class Server {
public static void main(String args[]) throws Exception{
        DatagramSocket ds=new DatagramSocket();
        InetAddress ip=InetAddress.getByName("localhost");
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        int port=2345;
        String msg;
        while(true)
        {
                msg=br.readLine();
                DatagramPacket dp=new DatagramPacket(msg.getBytes(),msg.length(),ip,port);
                if(!msg.equals("quit"))
                        ds.send(dp);
                else
                        break;
        }
}
}
```

RSA algo:

```
import java.io.*;
import java.math.*;
import java.net.*;
import java.security.*;

public class RSAdemo {
private int bitlen,r;
private BigInteger n,p,q,phi,d,e;
```

```
public RSAdemo(int bits)
{
	bitlen=bits;
	SecureRandom r=new SecureRandom();
	p=new BigInteger(bitlen/2,100,r);
	System.out.println("p value is"+p);
	q=new BigInteger(bitlen/2,100,r);
	System.out.println("q value is"+q);
	n=p.multiply(q);
	phi=(p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));
	e=new BigInteger(bitlen/2,100,r);
	while(e.compareTo(BigInteger.ONE)>0 && e.compareTo(phi)<0)
	{
		if(phi.gcd(e).equals(BigInteger.ONE))
		{
			break;
		}
	}
	System.out.println("encrytion key: "+e);
	d=e.modInverse(phi);
	System.out.println("decryption key: "+d);

}
public synchronized BigInteger encrypt(BigInteger Message){
	return Message.modPow(e,n);
}
public synchronized BigInteger decrypt(BigInteger cipher){
	return cipher.modPow(d,n);
}
public static void main(String args[]) throws IOException{
	RSAdemo rsa=new RSAdemo(256);
	BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
	String text1,text2;
	BigInteger plaintext,ciphertext;
	System.out.println("enter plain text");
	text1=br.readLine();
	plaintext=new BigInteger(text1.getBytes());
	System.out.println("Ciphertext");
	ciphertext=rsa.encrypt(plaintext);
	System.out.println(ciphertext);
	plaintext=rsa.decrypt(ciphertext);
	text2=new String(plaintext.toByteArray());
	System.out.println("plaintext is: "+text2);


}
}
```

bellmanford algo:

```
import java.util.*;
public class BellmanFord {
private int distance[];
private int numberofvertices;
public static final int MAX_VALUE=999;
public BellmanFord(int numberofvertices)
{
      this.numberofvertices=numberofvertices;
      distance=new int[numberofvertices+1];

}
public void BellmanFordEvaluation(int source,int adjacencymatrix[][])
{
      for(int node=1;node<=numberofvertices;node++)
      {
            distance[node]=MAX_VALUE;

      }
      distance[source]=0;
      for(int node=1;node<=numberofvertices-1;node++)
      {
            for(int sourcenode=1;sourcenode<=numberofvertices;sourcenode++)
            {
                  for(int
destinationnode=1;destinationnode<=numberofvertices;destinationnode++)
                  {
                        if(adjacencymatrix[sourcenode][destinationnode]!
=MAX_VALUE)
                        {
                              if(distance[destinationnode]>distance[sourcenode]



                                    +adjacencymatrix[sourcenode]
[destinationnode])
                              distance[destinationnode]=distance[sourcenode]
                                          +adjacencymatrix[sourcenode]
[destinationnode];


                        }
                  }
            }
      }
      for(int sourcenode=1;sourcenode<=numberofvertices;sourcenode++)
      {
            for(int
destinationnode=1;destinationnode<=numberofvertices;destinationnode++)
            {
                  if(adjacencymatrix[sourcenode][destinationnode]!=MAX_VALUE)
                  {
                        if(distance[destinationnode]>distance[sourcenode]
                              +adjacencymatrix[sourcenode]
[destinationnode])
                              System.out.println("the graph contains negative
cycle");
                  }
            }
      }
      for(int vertex=1;vertex<=numberofvertices;vertex++)
      {
```

```
            System.out.println("distance of source "+source+" to "+vertex+" is "
+distance[vertex]);
      }

}
public static void main(String args[])
{
int numberofvertices=0;
int source;
Scanner sc=new Scanner(System.in);
System.out.println("enter the no of vertices");
numberofvertices=sc.nextInt();
int adjacencymatrix[][]=new int[numberofvertices+1][numberofvertices+1];
System.out.println("enter the adjacency matrix");
for(int sourcenode=1;sourcenode<=numberofvertices;sourcenode++)
{
      for(int
destinationnode=1;destinationnode<=numberofvertices;destinationnode++)
      {
            adjacencymatrix[sourcenode][destinationnode]=sc.nextInt();
            if(sourcenode==destinationnode)
            {
                  adjacencymatrix[sourcenode][destinationnode]=0;
                  continue;
            }
            if(adjacencymatrix[sourcenode][destinationnode]==0)
            {
                  adjacencymatrix[sourcenode][destinationnode]=MAX_VALUE;

            }

      }
}
System.out.println("enter the sourcevertex");
source=sc.nextInt();
BellmanFord bf=new BellmanFord(numberofvertices);
bf.BellmanFordEvaluation(source, adjacencymatrix);
sc.close();

}
}
```

leaky bucket:

```
import java.util.*;


public class LeakyBucket {
public static void main(String args[]){
      int n,bsize,op_rate,bcnt,nbc=0,i;
      int pkt[]=new int[15];
      Scanner sc=new Scanner(System.in);
      System.out.println("enter no. of arrivals ");
      n=sc.nextInt();
      System.out.println("enter bucketsize ");
      bsize=sc.nextInt();
      System.out.println("enter output rate ");
      op_rate=sc.nextInt();
      System.out.println("enter the no. of pkt at each sec");
      for(i=0;i<n;i++)
```

```
            pkt[i]=sc.nextInt();
      System.out.println("\n sec\t no-pkts \t bcount \t status \t pkt_sent \t
nbc");

System.out.println("-----------------------------------------------------");
      for(i=0;i<n;i++)
      {
            System.out.print(i+1);
            System.out.print("\t"+pkt[i]);
            bcnt=nbc+pkt[i];
            if(bcnt<=bsize)
            {
                  System.out.print("\t\t"+bcnt);
                  System.out.print("\t\t Accept");
                  System.out.print("\t\t"+min(bcnt,op_rate));
                  nbc=sub(bcnt,op_rate);
                  System.out.print("\t\t"+nbc);
                  System.out.println();
            }
            else{
                  bcnt=nbc;
                  System.out.print("\t\t"+bcnt);
                  System.out.print("\t\t Reject");
                  System.out.print("\t\t"+min(bcnt,op_rate));
                  nbc=sub(bcnt,op_rate);
                  System.out.print("\t\t"+nbc);
                  System.out.println();
            }
      }
      while(nbc!=0)
      {
            System.out.print(++i);
            bcnt=nbc;
            System.out.print("\t");
            System.out.print("\t\t"+bcnt);
            System.out.print("\t\t\t Accept");
            System.out.print("\t\t\t"+min(op_rate,nbc));
            nbc=sub(nbc,op_rate);
            System.out.print("\t\t\t"+nbc);
      }

}
private static int sub(int bcnt,int op_rate)
{
      if(bcnt>op_rate)
            return(bcnt-op_rate);
      else
            return 0;
}
static int min(int bcnt,int bsize)
{
      if(bcnt<bsize)
            return bcnt;
      else
            return bsize;
}
}
```

gsm.tcl

```
set ns [new Simulator]

set topo [new Topography]
$topo load_flatgrid 1052 600
create-god 6

set tf [open gsm.tr w]
$ns trace-all $tf

set nf [open gsm.nam w]
$ns namtrace-all $nf
$ns namtrace-all-wireless $nf 1052 600
set chan [new Channel/WirelessChannel];

$ns node-config -adhocRouting AODV \
          -llType LL \
          -macType Mac/802_11 \
          -ifqType Queue/DropTail/PriQueue \
          -ifqLen 1000 \
          -antType Antenna/OmniAntenna \
          -propType Propagation/TwoRayGround \
          -phyType Phy/WirelessPhy \
          -channelType Channel/WirelessChannel\
          -energyModel EnergyModel \
  -initialEnergy 100 \
  -rxPower 0.3 \
  -txPower 0.6 \
  -topoInstance $topo \
          -agentTrace ON \
          -routerTrace ON \
          -macTrace OFF

set n0 [$ns node]
$n0 set X_ 303
$n0 set Y_ 302
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20

set n1 [$ns node]
$n1 set X_ 527
$n1 set Y_ 301
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20

set n2 [$ns node]
$n2 set X_ 748
$n2 set Y_ 300
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
```

```
set n3 [$ns node]
$n3 set X_ 952
$n3 set Y_ 299
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20

set n4 [$ns node]
$n4 set X_ 228
$n4 set Y_ 500
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20

set n5 [$ns node]
$n5 set X_ 305
$n5 set Y_ 72
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20

$ns at 2 "$n5 setdest 900 72 75"

set tcp0 [new Agent/TCP]
$ns attach-agent $n4 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n5 $sink1

$ns connect $tcp0 $sink1

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 10.0 "$ftp0 stop"

proc finish { } {
global ns tf nf
$ns flush-trace
close $tf
close $nf
exec nam gsm.nam &
exit 0
}

for {set i 0} {$i<6} {incr i} {
$ns at 10 "\$n$i reset"
}

#$ns at 10 "$ns nam-end-wireless $val(stop)"
$ns at 10 "finish"
#$ns at $val(stop) "puts \"done\";$ns halt"
$ns run
```

gsm.awk

```
BEGIN{
count1=0
pack1=0
time1=0
}
{
if($1=="r" && $3=="_5_" && $4=="AGT")
{
count1++
pack1=pack1+$8
time1=$2
}
}
END{
printf("the throughput from n4 to n5: %f Mbps \n",((count1*pack1*8)/(time1*1000000)));
}
```

cdma.tcl

```
Mac/802_11 set cdma_code_bw_start_ 0
Mac/802_11 set cdma_code_bw_stop_ 63
Mac/802_11 set cdma_code_bw_start_ 64
Mac/802_11 set cdma_code_bw_stop_ 127
Mac/802_11 set cdma_code_cqich_start_ 128
Mac/802_11 set cdma_code_cqich_stop_ 195
Mac/802_11 set cdma_code_handover_start_ 196
Mac/802_11 set cdma_code_handover_stop_ 255

set f0 [open out02.tr w]
set f1 [open lost02.tr w]
set f2 [open delay02.tr w]

set ns [new Simulator]
set topo [new Topography]

set tf [open cdma.tr w]
set nf [open cdma.nam w]

$ns trace-all $tf
$ns namtrace-all-wireless $nf 1500 1500

$topo load_flatgrid 1500 1500

set god [create-god 25]
$ns color 0 red
$ns node-config -adhocRouting AODV \
        -llType LL \
        -macType Mac/802_11 \
        -ifqType Queue/DropTail/PriQueue \
```

```
        -ifqLen 1000 \
        -antType Antenna/OmniAntenna \
        -propType Propagation/TwoRayGround \
        -phyType Phy/WirelessPhy \
        -channelType Channel/WirelessChannel \
        -energyModel EnergyModel \
  -initialEnergy 100 \
  -rxPower 0.3 \
  -txPower 0.6 \
  -topoInstance $topo \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace OFF

for {set i 0} {$i < 25 } {incr i} {
set node_($i) [$ns node]
$node_($i) set X_ [expr rand() * 1500]
$node_($i) set Y_ [expr rand() * 1000]
$node_($i) set Z_ 0.000000000000
}

for {set i 0} {$i < 25 } {incr i} {
set xx [expr rand() * 1500]
set yy [expr rand() * 1000]
$ns at 0.1 "$node_($i) setdest $xx $yy 5"
}

for {set i 0} {$i < 25 } {incr i} {
$ns initial_node_pos $node_($i) 55
}

for {set i 0} {$i < 25 } {incr i} {
$ns at 10.0 "$node_($i) reset";
}

set udp0 [new Agent/UDP]
$ns attach-agent $node_(4) $udp0

set sink [new Agent/LossMonitor]
$ns attach-agent $node_(20) $sink

 set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1000
$cbr0 set interval_ 0.1
$cbr0 set maxpkts_ 10000
$cbr0 attach-agent $udp0
$ns connect $udp0 $sink
$ns at 1.00 "$cbr0 start"

set holdtime 0
set holdseq 0
```

```
set holdrate1 0

proc record {} {
        global sink f0 f1 f2 holdtime holdseq holdrate1

        set ns [Simulator instance]
        set time 0.9

        set bw0 [$sink set bytes_]
        set bw1 [$sink set nlost_]

        set bw2 [$sink set lastPktTime_]
        set bw3 [$sink set npkts_]

        set now [$ns now]

                puts $f0 "$now [expr (($bw0+$holdrate1)*8)/(2*$time*1000000)]"
                puts $f1 "$now [expr $bw1/$time]"

        if { $bw3 > $holdseq } {
                puts $f2 "$now [expr ($bw2-$holdtime)/($bw3-$holdseq)]"
                } else {
                puts $f2 "$now [expr ($bw3-$holdseq)]"
                }

        $sink set bytes_ 0
        $sink set nlost_ 0

        set holdtime $bw2
        set holdseq $bw3

        set holdrate1 $bw0
        $ns at [expr $now+$time] "record"
}

$ns at 0.0 "record"

$ns at 1.0 "$node_(4) add-mark m blue square"
$ns at 1.0 "$node_(20) add-mark m magenta square"
$ns at 1.0 "$node_(4) label SENDER"
$ns at 1.0 "$node_(20) label RECEIVER"
$ns at 0.01 "ns trace-annotate \"Network Deployment\""

proc stop {} {
global ns tf nf f0 f1 f2
close $f0
close $f1
close $f2
exec nam cdma.nam

exec xgraph out02.tr -geometry -x TIME -y thr -t Throughput 800x400 &
exec xgraph lost02.tr -geometry -x TIME -y loss -t Packet_loss 800x400 &
```

```
exec xgraph delay02.tr -geometry -x TIME -y delay -t End-to-End-Delay 800x400 &

$ns flush-trace
}

$ns at 10 "stop"
$ns at 10.0002 "puts \"NS EXITING...\";$ns halt"
puts $tf "M 0.0 nn 25 x 1500 y 1500 rp"
puts $tf "M 0.0 prop Propagation/TwoRayGround ant Antenna/OmniAntenna"

puts "Starting Simulation..."
$ns run
```