```
/** ------------------------------------------------------------------
 * This version is used to simulate the transmission scenario in our
   paper "Link Selection for Secure Cooperative Networks with Buffer
   -Aided Relaying".
 * System Model: Two transmit schemes are used: adaptive-rate transm
   -ission and fixed transmission;
 * Wireless channel is Rayleigh fading model;
 * Output: the transmission probability, throughput, secrecy outage
   probability and secrecy throughput.
 * @author Ji He
 * @date 2018/11/08
 *//------------------------------------------------------------------


#include "randGenerator.h"

long seed = -1;

/**
Long period (> 2 * 1018) random number generator of L'Ecuyer with Bays
-Durham shuffleand added safeguards. Returns a uniform random deviate
between 0.0 and 1.0 (exclusive of the endpoint values). Call with 'idum'
a negative integer to initialise; thereafter, do not alter
'idum' between successive deviates in a sequence. 'RNMX' should approx
-imate the largest floating value that is less than 1.
**/
float ran2 (long  *idum )
{
    int j;
    long k;
    static  long idum2 = 123456789 ;
    static  long iy = 0;
    static  long iv[NTAB ];
    float temp ;
    if (*idum <= 0)
    {
    //Initialise.
        if (-(* idum ) < 1) *idum = 1;     //Be sure to prevent 'idum' = 0.
        else  *idum = -(* idum );
        idum2 =(* idum );
        for  (j = NTAB +7;j >= 0;j--)
        {
            //Load the shuffle table (after 8 warmups).
            k = (* idum ) / IQ1;
            *idum =IA1 * (*idum - k * IQ1) - k * IR1;
            if (*idum < 0) *idum += IM1;
            if (j < NTAB ) iv[j] = *idum ;
        }
        iy = iv[0];
    }
    k = (*idum ) / IQ1; //Start here when not initialising.
    *idum =IA1 * (*idum - k * IQ1) - k * IR1;  //Compute 'idum=(IA1*idum)' % IM1
    if (*idum < 0) *idum += IM1;//without overflows by Schrage's method.
    k = idum2 / IQ2;
    idum2 = IA2 * (idum2 - k * IQ2) - k * IR2; //Compute 'idum2=(IA2*idum)' % IM2
    if (idum2 < 0) idum2 += IM2;
    j = iy / NDIV; //Will be in the range 0_NTAB-1.
    iy = iv[j] - idum2 ; //Here 'idum' is shuffles, 'idum' and
    //'idum2' are combined to generate output.
    iv[j] = *idum ;
    if (iy < 1)iy += IMM1;
    if ((temp = AM * iy) > RNMX) return RNMX;
         //Because users don't expect endpoint values.
    else  return temp ;
}

```

```
67
68  /**
69  this function generates the exponentially distributed random variable
70  **/
71
72  float expGenerator(long  *idum , float lamb)
73  {
74      float dum;
75      do
76      {
77          dum = ran2 (idum );
78      }
79      while(dum == 0.0);
80      return -log(dum)/lamb;
81  }
82
83
84
85  #include "network.h"
86  #include <iostream>
87  #include "randGenerator.h"
88  using namespace std;
89  /*
90  int Alice::getaFlag()
91  {
92      return aFlag;
93  }
94  */
95  void Alice::init(double p)
96   {
97      power1 = p;
98  }
99  /*
100 int Bob::getbFlag()
101 {
102     return bFlag;
103 }*/
104 void Bob::init(double n)
105 {
106     noiVar1 = n;
107 }
108 /*
109 int Relay::getrFlag()
110 {
111     return rFlag;
112 }*/
113 void Relay::init(double p,double n)
114 {
115     power2 = p;
116     noiVar2 = n;
117 }
118 /*
119 int Eavesdropper::geteFlag(int hop)
120 {
121     eFlag = hop ;
122     return eFlags;
123 }*/
124 void Eavesdropper::init(double n)
125 {
126     noiVar3 = n;
127 }
128
129 void Network::setPowerNoivar(double p1, double p2,double n1,double n2,double n3)
130 {
131     alice.init(p1);
132     bob.init(n1);
```

```
133      relay.init(p2,n2);
134      eave.init(n3);
135  }
136
137  void Network::initARLinks()
138  {
139          aTorLinkCo = expGenerator(&seed,lambda1);
140          //cout<<"aTorLinkCo : "<< aTorLinkCo<< endl;
141  }
142  void Network::initRBLinks()
143  {
144          rTobLinkCo = expGenerator(&seed,lambda2);
145         // cout<<"rTobLinkCo: "<< rTobLinkCo<< endl;
146  }
147  void Network::initAELinks()
148  {
149          aToeLinkCo = expGenerator(&seed,lambda3);
150          //cout<<"aToeLinkCo : "<< aToeLinkCo<< endl;
151  }
152  void Network::initRELinks()
153  {
154          rToeLinkCo = expGenerator(&seed,lambda4);
155          //cout<<"rToeLinkCo : "<< rToeLinkCo<< endl;
156  }
157  /**optimal link selection**/
158  int Network::linkselection1(double p1,double p2,double n1,double n2,double n3)
159  {
160     // double Rrb = 0.0;                ///the channel instantaneous SNR
161      //double Rar = 0.0;                ///the channel instantaneous SNR
162      //double Rae = 0.0;                ///the channel instantaneous SNR
163      //double Rre = 0.0;                ///the channel instantaneous SNR
164      int temp;
165      double power2 =p2;     double power1 =p1;     double noiVar1 =n1;
166      double noiVar2 =n2;    double noiVar3 =n3;
167           Rrb = rTobLinkCo*power2/noiVar1;    Rar = aTorLinkCo*power1/noiVar2;
168           Rae = aToeLinkCo*power1/noiVar3;    Rre = rToeLinkCo*power2/noiVar3;
169
170          if(Rrb>=B&&Rrb/B>=Rar/A)
171        {
172           temp = 1;   /** the second hop is selected to transmit message**/
173        }
174        else if(Rar>=A&&Rrb/B<Rar/A)
175       {
176           temp = 0;   /** the first hop is selected to transmit message**/
177        }
178        else{
179           temp = -1;  /** Relay keeps idle **/
180        }
181
182        return temp;
183  }
184
185  /**optimal link selection**/
186  int Network::linkselection2(double p1,double p2,double n1,double n2,double n3)
187  {
188     // double Rrb = 0.0;                ///the channel instantaneous SNR
189      //double Rar = 0.0;                ///the channel instantaneous SNR
190      //double Rae = 0.0;                ///the channel instantaneous SNR
191      //double Rre = 0.0;                ///the channel instantaneous SNR
192     int temp;
193      double power2 =p2;double power1 =p1;double noiVar1 =n1;double noiVar2 =n2;double
noiVar3 =n3;
194           Rrb = rTobLinkCo*power2/noiVar1;    Rar = aTorLinkCo*power1/noiVar2;
195           Rae = aToeLinkCo*power1/noiVar3;    Rre = rToeLinkCo*power2/noiVar3;
196
197          if(Rrb>=max(B,B*Rar/A)||(A<=Rar&&Rar<=pow(2,Ra)-1&&B<=Rrb&&Rrb<=B*Rar/A))
```

```
198         {
199            temp = 1;
200         }
201          else if(Rar>pow(2,Ra)-1&&Rar>A*Rrb/B)
202        {
203            temp = 0;   /** the first hop is selected to transmit message**/
204         }
205         else{
206            temp = -1;  /** Relay keeps idle **/
207         }
208
209         return temp;
210  }
211  int Network::sopgettag1(double p1,double p2,double n1,double n2,double n3)
212  {
213     int tag;double Car;double Cae;double Crb;double Cre;
214      double power2 =p2;    double power1 =p1;    double noiVar1 =n1;
215      double noiVar2 =n2;    double noiVar3 =n3;    double n=0;
216           Rar = aTorLinkCo*power1/noiVar2;Rrb = rTobLinkCo*power2/noiVar1;
217           Rae = aToeLinkCo*power1/noiVar3;Rre = rToeLinkCo*power2/noiVar3;
218
219         Car = log2(1+Rar); Cae = log2(1+Rae);
220         Crb = log2(1+Rrb); Cre = log2(1+Rre);
221           if(Rrb>=B&&Rrb/B>=Rar/A&&(Crb-Cre)<Rs)
222         {
223            tag = -1;
224         }
225           if(Rar>=A&&Rar/A>Rrb/B&&(Car-Cae)<Rs)
226        {
227            tag = 1;
228         }
229
230      return tag;
231  }
232
233  int Network::dopgettag(double p1,double p2,double n1,double n2,double n3)
234  {
235     int tag1;double Car;double Cae;double Crb;double Cre;
236      double power2 =p2;   double power1 =p1;    double noiVar1 =n1;
237      double noiVar2 =n2;   double noiVar3 =n3;
238           Rar = aTorLinkCo*power1/noiVar2;Rrb = rTobLinkCo*power2/noiVar1;
239           Rae = aToeLinkCo*power1/noiVar3;Rre = rToeLinkCo*power2/noiVar3;
240
241         Car = log2(1+Rar); Cae = log2(1+Rae);
242         Crb = log2(1+Rrb); Cre = log2(1+Rre);
243           if( Rar>=A&&Rar/A>=Rrb/B&&Rs>Car)
244         {
245            tag1 = 1;
246         }
247         if( Rar>=A&&Rar/A>=Rrb/B&&Rs<=Car)
248         {
249            tag1 = 0;
250         }
251          if(Rrb>=B&&Rrb/B>=Rar/A&&Rs>Crb)
252         {
253            tag1 = -1;
254         }
255         if(Rrb>=B&&Rrb/B>=Rar/A&&Rs<=Crb)
256         {
257            tag1 = 2;
258         }
259
260      return tag1;
261  }
262  int Network::tartag(double p1,double p2,double n1,double n2,double n3)
263  {
```

```
264     int tag2;double Car;double Cae;double Crb;double Cre;
265      double power2 =p2;    double power1 =p1;   double noiVar1 =n1;
266      double noiVar2 =n2;    double noiVar3 =n3;
267           Rar = aTorLinkCo*power1/noiVar2;Rrb = rTobLinkCo*power2/noiVar1;
268           Rae = aToeLinkCo*power1/noiVar3;Rre = rToeLinkCo*power2/noiVar3;
269        Car = log2(1+Rar); Cae = log2(1+Rae);
270        Crb = log2(1+Rrb); Cre = log2(1+Rre);
271      if(Rs<=Car&&Cae<=(Car-Rs))
272        {
273           tag2 = 1;
274        }
275      if(Rs<=Crb&&Cre<=(Crb-Rs))
276        {
277           tag2 = -1;
278        }
279        else
280          tag2=0;
281      return tag2;
282  }
283  /*
284  void transmitATR(double p1,double p2,double n1,double n2,double n3)
285  {
286      int tag;double Car;double Cae;double Crb;double Cre;double speed;
287     Car = log10(1+Rar)/log10(2); Cae = log10(1+Rae)/log10(2);
288     Crb = log10(1+Rrb)/log10(2); Cre = log10(1+Rre)/log10(2);
289      double power2 =p2;double power1 =p1;double noiVar1 =n1;
290      double noiVar2 =n2;double noiVar3 =n3;
291           Rrb = rTobLinkCo*power2/noiVar1;
292           Rar = aTorLinkCo*power1/noiVar2;
293           Rae = aToeLinkCo*power1/noiVar3;Rre = rToeLinkCo*power2/noiVar3;
294
295        speed=Car;
296
297  }
298  void transmitRTB(double p1,double p2,double n1,double n2,double n3)
299  {
300      int tag;double Car;double Cae;double Crb;double Cre;double speed;
301     Car = log10(1+Rar)/log10(2); Cae = log10(1+Rae)/log10(2);
302     Crb = log10(1+Rrb)/log10(2); Cre = log10(1+Rre)/log10(2);
303      double power2 =p2;double power1 =p1;double noiVar1 =n1;
304      double noiVar2 =n2;double noiVar3 =n3;
305           Rrb = rTobLinkCo*power2/noiVar1;Rar = aTorLinkCo*power1/noiVar2;
306           Rae = aToeLinkCo*power1/noiVar3;Rre = rToeLinkCo*power2/noiVar3;
307
308        speed=min(Rs,car);
309
310  }
311  */
312  int Network::sopgettag_21(double p1,double p2,double n1,double n2,double n3)
313  {
314      /**fixed-rate transmission a>2^Ra-1**/
315      int tag;double Car;double Cae;double Crb;double Cre;
316
317      double power2 =p2;    double power1 =p1;   double noiVar1 =n1;
318      double noiVar2 =n2;   double noiVar3 =n3;
319           Rrb = rTobLinkCo*power2/noiVar1;Rar = aTorLinkCo*power1/noiVar2;
320           Rae = aToeLinkCo*power1/noiVar3;Rre = rToeLinkCo*power2/noiVar3;
321         Car = log2(1+Rar); Cae = log2(1+Rae);
322         Crb = log2(1+Rrb); Cre = log2(1+Rre);
323        if(Ra-Cae<Rs)
324           {
325             tag = 1;
326           }
327         if(Rrb>=B&&Rrb/B>=Rar/A&&(Crb-Cre)<Rs)
328         {
329          tag = -1;
```

```
        }
    return tag;
}
int Network::sopgettag_22(double p1,double p2,double n1,double n2,double n3)
{
    /**fixed-rate transmission a<=2^Ra-1**/
  int tag;double Car;double Cae;double Crb;double Cre;

    double power2 =p2;   double power1 =p1;   double noiVar1 =n1;
    double noiVar2 =n2;  double noiVar3 =n3;
         Rrb = rTobLinkCo*power2/noiVar1;Rar = aTorLinkCo*power1/noiVar2;
         Rae = aToeLinkCo*power1/noiVar3;Rre = rToeLinkCo*power2/noiVar3;
        Car = log2(1+Rar); Cae = log2(1+Rae);
        Crb = log2(1+Rrb); Cre = log2(1+Rre);
       if(Ra-Cae<Rs)
          {
            tag = 1;
          }

        if((Crb-Cre<Rs&&Rrb>=max(B,B*Rar/A))||(Crb-Cre<Rs&&(A<=Rar&&Rar<=pow(2,Ra)-1
&&B<=Rrb&&Rrb<=B*Rar/A)))
         {
          tag = -1;
         }
    return tag;
}
int Network::top2gettag(double p1,double p2,double n1,double n2,double n3)
{
    ////cout<<"Rar: "<< Rar<< endl; cout<<"Rrb "<< Rrb<< endl;
   int tag;double Cae;double Car; Car = log10(1+Rar)/log10(2);
  Cae = log10(1+Rae)/log10(2);
    double power2 =p2;   double power1 =p1;   double noiVar1 =n1;
    double noiVar2 =n2;   double noiVar3 =n3;
         Rrb = rTobLinkCo*power2/noiVar1;  Rar = aTorLinkCo*power1/noiVar2;
         Rae = aToeLinkCo*power1/noiVar3;  Rre = rToeLinkCo*power2/noiVar3;
         if( Car<Ra)
     {
         tag = 1;
      }
    return tag;
}



#include <iostream>
#include <fstream>
#include <string>
#include "network.h"
#include "randGenerator.h"
//#define LOOP 100000000
#define max(a, b)  (((a) >= (b)) ? (a) : (b))
using namespace std;

/**This function calculates the simulated transmission outage probability for a
specific setting of p1,p2,n1,n2 and n3**/

void TPpro_sim(Network &network,double p1,double p2,double n1,double n2,double n3)
{
    int tCount1 = 0;  int tCount2 = 0; int tCount3 = 0; int tCount4=0;
    int tCount6=0;   double pr;  double pa;  double po;
    int temp1 = 0;  int LOOP = 100000000; int temp=0; int temp2=0;
   network.setPowerNoivar(p1,p2,n1,n2,n3);
    for(int i = 0;i < LOOP;i++)
    {
         ran2(&seed);
        network.initARLinks();
```

```
395          network.initRBLinks();
396          network.initAELinks();
397          network.initRELinks(); /** initialize all links **/
398         temp1 = network.linkselection1(p1,p2,n1,n2,n3);
399                                  /** conduct the link selection policy in case1 **/
400        if(i<=20000000){
401         if(temp1==1){        /** before 20% time slots,the relay-bob link is chose**/
402              if(tCount2==0)  /** the buffer of relay is empty**/
403                  {tCount2=tCount2;   /** the number of data in the buffer remain
unchanged **/
404                   tCount1=tCount1;  /** the data transmitted by relay are dummy
bits**/
405                  }
406              else                    /** the buffer of relay is not empty**/
407                 {tCount1=tCount1+1;
408                  tCount2=tCount2-1;}
409                }
410                     /** compute the probability that relay-bob link is selected**/
411          if(temp1==0)    /** the alice-relay link is chose**/
412          {tCount2=tCount2+1;
413           tCount1=tCount1;
414           temp2=temp2+1;
415          }
416          if(temp1==-1)    /** no link is chose**/
417          {tCount3=tCount3+1;
418          tCount1=tCount1;tCount2=tCount2;}
419         temp=tCount1; /** count the packet number of Bob**/
420        /** temp2=tCount2;**/
421      }
422      else{
423          if(temp1==1){    /** the relay-bob link is chose**/
424                  tCount4= tCount4+1;
425              if(tCount2==0)  /** the buffer of relay is empty**/
426                  {tCount2=tCount2;  /** the number of data in the buffer remain
unchanged **/
427                   tCount1=tCount1;  /** the data transmitted by relay are dummy
bits**/
428                  }
429              else                    /** the buffer of relay is not empty**/
430                 {tCount1=tCount1+1;
431                  tCount2=tCount2-1;}
432                }
433                     /** compute the probability that relay-bob link is selected**/
434          if(temp1==0)    /** the alice-relay link is chose**/
435          {tCount2=tCount2+1;
436           tCount1=tCount1; temp2=temp2+1;
437           tCount6=tCount6+1; /** after 20% time slots,the relay-bob link is chose**/
438          }
439          if(temp1==-1)    /** no link is chose**/
440          {tCount3=tCount3+1;
441          tCount1=tCount1;tCount2=tCount2;}
442      }
443      }
444      /** pr=(double)tCount1/(double)LOOP; total **/
445       pr=(double)tCount4/(double)80000000;
446       pa=(double)tCount6/(double)80000000;
447       po=(double)temp2/(double)LOOP;
448    /**  pa=(double)tCount6/(double)80000000;
449       po=(double)tCount3/(double)80000000;**/
450      cout<<"pr is : "<< pr<< endl;
451      cout<<"pa is : "<< pa<< endl;
452       cout<<"po is : "<< po<< endl;
453      cout<<"Throughput is : "<< tCount1<< endl;
454     /** cout<<"thoughput is : "<< tCount1<< endl;
455      cout<<"thoughput is : "<< tCount2<< endl;
456      cout<<"thoughput is : "<< tCount3<< endl;
```

```
457      cout<<"temp is : "<< temp<<  endl;
458      cout<<"temp is : "<< temp2<<  endl;**/
459
460  }
461  /**This function calculates the simulated secrecy outage probability for a
462  specific setting of p1,p2,n1,n2 and n3 in adaptive rate transimisson**/
463  void  SOPpr_sim_1(Network &network,double p1,double p2,double n1,double n2,double n3
)
464  {
465       int tCount1 = 0;int tCount2 = 0;double Psopa;double Psopr;double Psop;int temp1
= 0;  int flag = 0;
466       int LOOP = 1000000; int Count1 = 0;int Count2 = 0;int Count3 = 0;
467       double pr;double pa;double po;
468      network.setPowerNoivar(p1,p2,n1,n2,n3);
469
470      for(int i = 0;i < LOOP;i++)
471      {
472           ran2(&seed);
473          network.initARLinks();
474          network.initRBLinks();
475          network.initAELinks();
476          network.initRELinks();
477          temp1 = network.linkselection1(p1,p2,n1,n2,n3);
478           if(temp1==0)  /**the first hop is chosen.**/
479          {
480              tCount1=tCount1+1;
481          }
482          if(temp1==1)        /**the second hop is chosen.**/
483          {
484              tCount2=tCount2+1;
485          }
486      flag = network.sopgettag1(p1,p2,n1,n2,n3);
487
488       if(flag==1)      /**SOP in the first hop **/
489          {
490              Count1=Count1+1;
491          }
492      if(flag==-1)   /**SOP in the second hop **/
493          {
494              Count2=Count2+1;
495          }
496      }
497
498       pa=(double)tCount1/(double)LOOP; //old scheme
499       pr=(double)tCount2/(double)LOOP;
500
501       cout<<"Count2 : "<< Count2<< endl;
502
503      Psopa=(double)Count1/(double)tCount1;
504      Psopr=(double)Count2/(double)tCount2;
505      Psop=1-(1-Psopa)*(1-Psopr);
506         cout<<"Psopa is : "<< Psopa << endl; //
507        cout<<"Psopr is : "<< Psopr << endl; //old scheme
508        cout<<"Psop is : "<< Psop << endl;
509        cout<<"Pa is : "<< pa << endl;
510        cout<<"Pr is : "<< pr << endl;
511  }
512  /**This function calculates the simulated secrecy outage probability for a
513  specific setting of p1,p2,n1,n2 and n3 in fixed rate transimisson 2.1**/
514  void  SOPpr_sim_2(Network &network,double p1,double p2,double n1,double n2,double n3
)
515  {
516         int tCount1 = 0;   int tCount2 = 0;   double Psopa;
517         double Psopr;   double Psop;   int temp1 ;  int flag ;
518        int LOOP = 1000000; int Count1 = 0;  int Count2 = 0;  int Count3 = 0;
519        double pr;double pa;double po;
```

```
520
521       network.setPowerNoivar(p1,p2,n1,n2,n3);
522
523       for(int i = 0;i < LOOP;i++)
524       {
525            ran2(&seed);
526           network.initARLinks();
527          network.initRBLinks();
528           network.initAELinks();
529           network.initRELinks();
530           if(A>=pow(2,Ra)-1)
531            {
532                temp1 = network.linkselection1(p1,p2,n1,n2,n3);
533                 if(temp1==0)  /**the first hop is chosen.**/
534               {
535                  tCount1=tCount1+1;
536               }
537                  if(temp1==1)              /**the second hop is chosen.**/
538               {
539                  tCount2=tCount2+1;
540                }
541                 flag = network.sopgettag_21(p1,p2,n1,n2,n3);
542                  if(flag==1)                /**SOP in the first hop **/
543               {
544                Count1=Count1+1;
545               }
546                  if(flag==-1)               /**SOP in the second hop **/
547               {
548                Count2=Count2+1;
549               }
550          }
551           if(A<pow(2,Ra)-1)
552            {
553
554                flag = network.sopgettag_22(p1,p2,n1,n2,n3);
555                  if(flag==1)                /**SOP in the first hop **/
556               {
557                Count1=Count1+1;
558               }
559                  if(flag==-1)               /**SOP in the second hop **/
560               {
561                Count2=Count2+1;
562               }
563                  temp1 = network.linkselection2(p1,p2,n1,n2,n3);
564                  if(temp1==0)              /**the first hop is chosen.**/
565               {
566                  tCount1=tCount1+1;
567               }
568                  if(temp1==1)              /**the second hop is chosen.**/
569               {
570                  tCount2=tCount2+1;
571                }
572          }
573       }
574        pa=(double)tCount1/(double)LOOP;
575        pr=(double)tCount2/(double)LOOP;
576        cout<<"tCount1 : "<< tCount1<< endl;
577        cout<<"tCount2 : "<< tCount2<< endl;
578        cout<<"Count1 : "<< Count1<< endl;
579       cout<<"Count2 : "<< Count2<< endl;
580       Psopa=(double)Count1/(double)LOOP;
581       Psopr=(double)Count2/(double)tCount2;
582       Psop=1-(1-Psopa)*(1-Psopr);
583          cout<<"Psopa is : "<< Psopa << endl; //
584         cout<<"Psopr is : "<< Psopr << endl; //old scheme
585         cout<<"Psop is : "<< Psop << endl;
```

```
586         cout<<"Pa is : "<< pa << endl;
587         cout<<"Pr is : "<< pr << endl;
588
589  }
590  /**This function calculates the simulated transmission outage probability for a
591  specific setting of p1,p2,n1,n2 and n3 in fixed rate transimisson 2.1**/
592  void  Toppr_sim(Network &network,double p1,double p2,double n1,double n2,double n3)
593  {
594       int tCount1 = 0; double Ptop;  int flag21 = 0; int temp1;
595        int LOOP = 1000000; int Count1 = 0;
596      network.setPowerNoivar(p1,p2,n1,n2,n3);
597
598      for(int i = 0;i < LOOP;i++)
599      {
600           ran2(&seed);
601          network.initARLinks();
602         //network.initRBLinks();
603          network.initAELinks();
604        //  network.initRELinks();
605          temp1 = network.linkselection1(p1,p2,n1,n2,n3);
606          flag21 = network.top2gettag(p1,p2,n1,n2,n3);
607
608          if(flag21==1)
609          {
610              Count1=Count1+1;
611          }
612
613      }
614       //cout<<"Count1 : "<< Count1<< endl;
615       Ptop=(double)Count1/(double)LOOP;
616      cout<<"Ptop is : "<< Ptop << endl;
617  }
618  /**This function calculates the simulated transmission outage probability for a
619  specific setting of p1,p2,n1,n2 and n3**/
620
621  void poprpa22_sim(Network &network,double p1,double p2,double n1,double n2,double n3
)
622  {
623
624      int tCount1 = 0;    int tCount2 = 0; int tCount3 = 0;    int tCount4=0;
625      int tCount6=0;     double pr;    double pa;double po;
626      int temp1 = 0;    int LOOP = 100000000;  int temp=0;  int temp2=0;
627     network.setPowerNoivar(p1,p2,n1,n2,n3);
628      for(int i = 0;i < LOOP;i++)
629      {
630           ran2(&seed);
631          network.initARLinks();
632          network.initRBLinks();
633          network.initAELinks();
634          network.initRELinks();     /** initialize all links **/
635         temp1 = network.linkselection2(p1,p2,n1,n2,n3);
636                                    /** conduct the link selection policy in case1 **/
637        if(i<=20000000){
638         if(temp1==1){              /** before 20% time slots,the relay-bob link is
chose**/
639              if(tCount2==0)         /** the buffer of relay is empty**/
640                  {tCount2=tCount2;  /** the number of data in the buffer remain
unchanged **/
641                   tCount1=tCount1;  /** the data transmitted by relay are dummy
bits**/
642                  }
643             else                    /** the buffer of relay is not empty**/
644                {tCount1=tCount1+1;
645                 tCount2=tCount2-1;}
646               }
647                                   /** compute the probability that relay-bob link is
```

```
selected**/
          if(temp1==0)            /** the alice-relay link is chose**/
          {tCount2=tCount2+1;
           tCount1=tCount1;
           temp2=temp2+1;
          }
          if(temp1==-1)           /** no link is chose**/
          {tCount3=tCount3+1;
          tCount1=tCount1;tCount2=tCount2;}
         temp=tCount1;           /** count the packet number of Bob**/
                                 /** temp2=tCount2;**/
      }
      else{
          if(temp1==1){    /** the relay-bob link is chose**/
                 tCount4= tCount4+1;
              if(tCount2==0)  /** the buffer of relay is empty**/
                  {tCount2=tCount2;  /** the number of data in the buffer remain
unchanged **/
                   tCount1=tCount1;  /** the data transmitted by relay are dummy
bits**/
                  }
             else                    /** the buffer of relay is not empty**/
                {tCount1=tCount1+1;
                 tCount2=tCount2-1;}
               }
                     /** compute the probability that relay-bob link is selected**/
          if(temp1==0)    /** the alice-relay link is chose**/
          {tCount2=tCount2+1;
           tCount1=tCount1; temp2=temp2+1;
           tCount6=tCount6+1; /** after 20% time slots,the relay-bob link is chose**/
          }
          if(temp1==-1)    /** no link is chose**/
          {tCount3=tCount3+1;
          tCount1=tCount1;tCount2=tCount2;}
      }
      }
      /** pr=(double)tCount1/(double)LOOP; total **/
       pr=(double)tCount4/(double)80000000;
       pa=(double)tCount6/(double)80000000;
       po=(double)temp2/(double)LOOP;

      cout<<"pr is : "<< pr<< endl;
      cout<<"pa is : "<< pa<< endl;
       cout<<"po is : "<< po<< endl;
      cout<<"Throughput is : "<< tCount1<< endl;
  }
  /**This function calculates the simulated secrecy outage probability for a
  specific setting of p1,p2,n1,n2 and n3 in adaptive rate transimisson**/
  void  SOPa22_sim(Network &network,double p1,double p2,double n1,double n2,double n3)
  {
     int tCount1 = 0; double Psopa;  int flag21 = 0; int temp1;
        int LOOP = 1000000; int Count1 = 0;
      network.setPowerNoivar(p1,p2,n1,n2,n3);

      for(int i = 0;i < LOOP;i++)
      {
           ran2(&seed);
          network.initARLinks();
         //network.initRBLinks();
          network.initAELinks();
        //  network.initRELinks();
          temp1 = network.linkselection1(p1,p2,n1,n2,n3);


          if(flag21==-1)
          {
```

```
            Count1=Count1+1;
        }

    }
     //cout<<"Count1 : "<< Count1<< endl;
     Psopa=(double)Count1/(double)LOOP;
    cout<<"Psopa is : "<< Psopa << endl;
}

double Max_secrecy_outage_capacity(Network &network,double p1,double p2,double n1,
double n2,double n3)
{
    int LOOP = 100000000;   double reSpeed = 0;   double alSpeed = 0;
    double speed = 0;    double sethroughput=0;    int temp1;
   double relaysize=0;   double bobsize=0;   double rebuffer=0;int flag;
   int flag1;    int tcount;         //double Rs =4.433;
                    //queue<double> Rebuffer;queue<double> Bobuffer;
                  // cout<<Rebuffer.size()<<endl;
                  //cout<<Bobuffer.size()<<endl;
    network.setPowerNoivar(p1,p2,n1,n2,n3);
     for(int i = 0;i < LOOP;i++)
    {
         ran2(&seed);
        network.initARLinks();
        network.initRBLinks();
        network.initAELinks();
        network.initRELinks();

         temp1 = network.linkselection1(p1,p2,n1,n2,n3);
         flag1 = network.tartag(p1,p2,n1,n2,n3);
       reSpeed=log2(1+network.Rrb);alSpeed=log2(1+network.Rar);
       if(temp1==1)//the second hop is chosen to transmit message
        {
            if(flag1==-1)
            {
                tcount=tcount+1;

                speed=min(rebuffer,Rs);
                //speed=min(Rebuffer,reSpeed);
                rebuffer=rebuffer-speed;
                bobsize=bobsize+speed;
            }
        }
      if(temp1==0)
        {
            if(flag1==1)
            {
              rebuffer=rebuffer+Rs;
            }
        }
       // cout<<"bobsize is :"<<bobsize<<endl;
      //   cout<<"tcount is :"<<tcount<<endl;
    }
      // sethroughput=bobsize/LOOP;
       //  cout<<"sethroughput is :"<<sethroughput<<endl;

      cout<<"tcount is :"<<(double)tcount/LOOP*Rs<<endl;

}

int main()
{
    Network network;
   //TPpro_sim(network,1,1, 1, 1, 1);
    SOPpr_sim_1(network,1,1, 1, 1, 1);
   //Max_secrecy_outage_capacity(network,1,1,1,1,1);
```

```
776     // SOPpr_sim_2(network,1,1, 1, 1, 1);
777    // Toppr_sim(network,1,1, 1, 1, 1);
778
779    // poprpa22_sim(network,1,1, 1, 1, 1);
780     //SOPa22_sim(network,1,1, 1, 1, 1);
781
782      return 0;
783  }
```

%% The algorithm is called in the main function when you solve the optimal problem.

```
function [X0,f_val]=LiPaOp(A,b,x0,Aeq,beq,label)
% diff_val(x0) is used to find the partial derivative of the given
function the initial point at x0
%The functionfval(x0)is used to find the value of the given function at
the initial point x0
format long;
eps=1.0e-6;
x0=transpose(x0); %The initial point x0 is the row vector
func
sz=length(x0);
if label==1
[m,n]=size(A);  %Decompose A into two parts, i.e., where A1 is the
effective constraint set
for k=1:1:100
    A1=A;
    A2=A;
    b1=b;
    b2=b;
    for i=m:-1:1
        if abs(A2(i,:)*x0-b2(i,:)) < 0.1
            A2(i,:)=[];
            b2(i,:)=[];
        end
    end
    for i=m:-1:1
        if abs(A1(i,:)*x0-b1(i,:))>=0.1
            A1(i,:)=[];
            b1(i,:)=[];
        end
    end
    A1;
    A2;
    b1;
    b2;

    i2=rank(A2);
    AE=[A1;Aeq];
    [i1,j1]=size(AE);
    r=rank(AE);
    if r<i1
        'Dissatisfied rank'
        return
    end
    if i2==0
        'invalid'
        return
    end
    %Solving the linear programming problem yields a feasible descent
direction d0

     s=diff_val(x0);
     c=double(s);
     lb=-1*ones(sz,1);
```

```
    ub=ones(sz,1);
    k1=length(b1);
    k2=length(beq);
    p=zeros(k1,1);
    q=zeros(k2,1);
    [d0,mn,m1,m2,m3]=linprog(c,A1,p,Aeq,q,lb,ub);
    d0;mn;
    df=abs(s*d0);
    if df<0.1
        'The optimal solution is:'
        x0
        f_val=fval(x0)
        k
        return
    else
        %Perform a one-dimensional search to find the minimum value of
the function f(x(k+1))
        b_=b2-A2*x0;
        d_=A2*d0;
        [dh,dl]=size(d_);
        ul=1;
        for i=1:1:dh
            if d_(i,:)>=0
                u=1;
            else
                u=0;
            end
            ul=ul*u;
        end
        ul;b_;d_;
        vmax=inf;
        if ul==0
            vmax=inf;
        else
            for i=1:1:dh
                if d_(i,:)>0
                    v=b_(i,:)/d_(i,:);
                    if v<vmax
                        vmax=v;
                    end
                end
            end

        end
    end
    vmax;
    h=fmin(x0,d0,vmax);
    a=x0+h*d0;
    f_val=fval(a);
    x0=x0+h*d0;
    '****************'
    X0=x0
    f_val=fval(x0)
    k
    '****************'
end
end
```

```
%%the trade-off between throughput and E2E SOP if Pa<=Pr or Pa>Pr  for
my paper
%%when the Pa <=Pr, the thgroughput is larger.
function main1
    clear all
    close all
    clc
    global a
    x0=[1.0,1.0,0.1];A=[-1,0,0;0,-1,0;0,0,-
1];b=[0;0;0];lb=[0.00001,0.00001,0.00001];ub=[1000,1000,1000];Aeq=[];be
q=[];
    max2=[]; Rar=sqrt(100); Rrb=sqrt(1000); Rae=1 ; Rre=3.9811; i=0;
   %% you can change to different values
   K=[0.001 0.0014 0.0021 0.003 0.0043 0.0062 0.0089 0.0127 0.0183
0.0264 0.0379 0.0546 0.0785 0.1129 0.1624 0.2336 0.336 0.4833 0.6952
1];   %% the E2E SOP constraint
       Pa=(exp(-x(1)/Rar)-x(1)*Rrb*exp(-
(x(1)/Rar+x(2)/Rrb))/(x(1)*Rrb+x(2)*Rar))
   %%The probability that Alice is selected to transmit the message
       Pr=(exp(-x(2)/Rrb)-x(2)*Rar*exp(-
(x(1)/Rar+x(2)/Rrb))/(x(1)*Rrb+x(2)*Rar))
  %%The probability that Relay is selected to transmit the message

    fun=@(x) -((exp(-x(2)/Rrb)-x(2)*Rar*exp(-
(x(1)/Rar+x(2)/Rrb))/(x(1)*Rrb+x(2)*Rar)))*x(3);
      %% the throughput of the system

     opts5 = optimset('Algorithm','sqp','MaxFunEvals',100000);
            problem5 = createOptimProblem('fmincon','objective',...
                fun,'x0',x0, 'Aineq', A, 'bineq', b, 'nonlcon', @mycon,
'options',opts5);
             for a=K
            i=i+1;
      [x2,fval2] = fmincon(fun,x0,[],[],[],[],lb,ub,@mycon,opts5)
[x2,fval2] =LiPaOp(fun,x0,[],[],[],[],lb,ub,@mycon,opts5)
              max(i)=-fval2;
          end
         max

end

function[c,ceq]=mycon(x) %%c is the constraint conditions

global a  %% the global variable of the constraint on E2E SOP

   Rar=sqrt(100); Rrb=sqrt(1000); Rae=1 ; Rre=3.9811;  %%10,10,0 ,2 dB.
1.50515 sqrt(20)
  Pa=exp(-x(1)/Rar)*(1-x(1)*Rrb*exp(-x(2)/Rrb)/(x(1)*Rrb+x(2)*Rar));
  Pr=exp(-x(2)/Rrb)*(1-x(2)*Rar*exp(-x(1)/Rar)/(x(1)*Rrb+x(2)*Rar));
   Pso1=(Rae*2^x(3)/(Rae*2^x(3)+Rar)*exp(-(x(1)/Rar+(x(1)+1-
2^x(3))/(Rae*2^x(3))))*(1-...

(x(1)*Rrb/(x(2)*Rar)+x(1)*Rrb/(x(2)*Rae*2^x(3)))/(1+x(1)*Rrb/(x(2)*Rar)
```

```
+x(1)*Rrb/(x(2)*Rae*2^x(3)))*exp(-(x(2)/Rrb)))/(exp(-x(1)/Rar)-
x(1)*Rrb*exp(-(x(1)/Rar+x(2)/Rrb))/(x(1)*Rrb+x(2)*Rar)));
   %%the secrecy outage probability in the first hop
   Pso2=(Rre*2^x(3)/(Rre*2^x(3)+Rrb)*exp(-(x(2)/Rrb+(x(2)+1-
2^x(3))/(Rre*2^x(3))))*(1-...

(x(2)*Rar/(x(1)*Rrb)+x(2)*Rar/(x(1)*Rre*2^x(3)))/(1+x(2)*Rar/(x(1)*Rrb)
+x(2)*Rar/(x(1)*Rre*2^x(3)))*exp(-(x(1)/Rar)))/(exp(-x(2)/Rrb)-
x(2)*Rar*exp(-(x(1)/Rar+x(2)/Rrb))/(x(1)*Rrb+x(2)*Rar)));

%%the secrecy outage probability in the second hop

  Pso=1-(1-Pso1)*(1-Pso2);   %%the E2E SOP

   c(1)=(2^x(3)-1)-x(1);
   c(2)=(2^x(3)-1)-x(2);
   c(3)=Pr-Pa;
%     c(8)=Pn-0.01;
   c(4)=Pso-a;
   ceq=[];
   c=[c(1),c(2),c(3),c(4)];
end



%%the trade-off between SOP and throughput  for
Pa<=Pr ,Pa>Pr ,\alpha<2^R_a-1, \alpha>=2^R_a-1  FR case
function main
    clear all
    close all
    clc
    global a
    x0=[1.0,1.0,0.1];A=[-1,0,0;0,-1,0;0,0,-
1];b=[0;0;0];lb=[0.00001,0.00001,0.00001];ub=[1000,1000,1000];Aeq=[];be
q=[];
    max2=[]; Rar=sqrt(100); Rrb=sqrt(1000); Rae=1; Rre=1.5849; i=0;
Ra=3;

    K=[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2];

    fun=@(x) 1-(1-(exp(-(2^(Ra-x(3))-1)/Rae)))*(1-
(((Rre*2^x(3)/(Rre*2^x(3)+Rrb)*exp(-(x(2)/Rrb+(x(2)+1-
2^x(3))/(Rre*2^x(3))))*(1-...

(x(2)*Rar/(x(1)*Rrb)+x(2)*Rar/(x(1)*Rre*2^x(3)))/(1+x(2)*Rar/(x(1)*Rrb)
+x(2)*Rar/(x(1)*Rre*2^x(3)))*exp(-(x(1)/Rar))))+...
       Rre*2^x(3)/(Rre*2^x(3)+Rrb)*((1-
1/(1+x(2)*Rar/(x(1)*Rrb)+x(2)*Rar/(x(1)*2^x(3)*Rre)))*exp(-
(x(1)/Rar+x(2)/Rrb+(x(2)+1-2^Ra)/(2^x(3)*Rre)))-...
       exp(-((2^Ra-1)/Rar+x(2)/Rrb+(x(2)+1-
2^Ra)/(2^x(3)*Rre)))+1/(1+x(2)*Rar/(x(1)*Rrb)+x(2)*Rar/(x(1)*2^x(3)*Rre
))*...
       exp(-((2^Ra-1)/Rar+x(2)*(2^Ra-1)/(x(1)*Rrb)+(x(2)*(2^Ra-1)+1-
2^Ra)/(x(1)*2^x(3)*Rre)))))/...
       (exp(-x(2)/Rrb)+x(1)*Rrb*exp(-((2^Ra-1)/Rar+x(2)*(2^Ra-
1)/(x(1)*Rrb)))/(x(1)*Rrb+x(2)*Rar)-exp(-(x(2)/Rrb+(2^Ra-1)/Rar)))));
```

```
 %%the expression of E2E SOP, you can understand it by the following
detailed comments

     opts5 = optimset('Algorithm','sqp','MaxFunEvals',100000);
            problem5 = createOptimProblem('fmincon','objective',...
               fun,'x0',x0, 'Aineq', A, 'bineq', b, 'nonlcon', @mycon,
'options',opts5);
             for a=K
            i=i+1;
      [x2,fval2] = fmincon(fun,x0,[],[],[],[],lb,ub,@mycon,opts5)
[x2,fval2] =LiPaOp(fun,x0,[],[],[],[],lb,ub,@mycon,opts5)
               max(i)=-fval2;

          end
         max
      fid = fopen('1.txt','wt');
      fprintf(fid,'%g\n',max);
      fclose(fid);
end

function[c,ceq]=mycon(x)
global a
   Rar=sqrt(100);    Rrb=sqrt(1000);   Rae=1;    Rre=1.5849; i=0; Ra=3;
Pa=exp(-(2^Ra-1)/Rar)-x(1)*Rrb*exp(-((2^Ra-1)/Rar+x(2)*(2^Ra-
    1)/(x(1)*Rrb)))/(x(1)*Rrb+x(2)*Rar);          %%a<2^Ra-1
Pa=exp(-x(1)/Rar)-x(1)*Rrb*exp(-
(x(1)/Rar+x(2)/Rrb))/(x(1)*Rrb+x(2)*Rar);         %%a>=2^Ra-1
Pr=exp(-x(2)/Rrb)-x(2)*Rar*exp(-
(x(1)/Rar+x(2)/Rrb))/(x(1)*Rrb+x(2)*Rar);         %%a>=2^Ra-1
    Pr=exp(-x(2)/Rrb)+x(1)*Rrb*exp(-((2^Ra-1)/Rar+x(2)*(2^Ra-
1)/(x(1)*Rrb)))/(x(1)*Rrb+x(2)*Rar)-exp(-(x(2)/Rrb+(2^Ra-
1)/Rar));                                         %%a<2^Ra-1
    Pso1=exp(-(2^(Ra-x(3))-1)/Rae);
 %  Psop1=(Rae*2^x(3)/(Rae*2^x(3)+Rar)*exp(-(x(1)/Rar+(x(1)+1-
2^x(3))/(Rae*2^x(3))))*(1-...
 %
(x(1)*Rrb/(x(2)*Rar)+x(1)*Rrb/(x(2)*Rae*2^x(3)))/(1+x(1)*Rrb/(x(2)*Rar)
+x(1)*Rrb/(x(2)*Rae*2^x(3)))*exp(-(x(2)/Rrb)))/(exp(-x(1)/Rar)-
x(1)*Rrb*exp(-(x(1)/Rar+x(2)/Rrb))/(x(1)*Rrb+x(2)*Rar)));
%    Pso2=(Rre*2^x(3)/(Rre*2^x(3)+Rrb)*exp(-(x(2)/Rrb+(x(2)+1-
2^x(3))/(Rre*2^x(3))))*(1-...
%
(x(2)*Rar/(x(1)*Rrb)+x(2)*Rar/(x(1)*Rre*2^x(3)))/(1+x(2)*Rar/(x(1)*Rrb)
+x(2)*Rar/(x(1)*Rre*2^x(3)))*exp(-(x(1)/Rar)))/(exp(-x(2)/Rrb)-
x(2)*Rar*exp(-(x(1)/Rar+x(2)/Rrb))/(x(1)*Rrb+x(2)*Rar)));
%
%    PSOP2=(Rre*2^x(3)/(Rre*2^x(3)+Rrb)*exp(-(x(2)/Rrb+(x(2)+1-
2^x(3))/(Rre*2^x(3))))*(1-...
%
(x(2)*Rar/(x(1)*Rrb)+x(2)*Rar/(x(1)*Rre*2^x(3)))/(1+x(2)*Rar/(x(1)*Rrb)
+x(2)*Rar/(x(1)*Rre*2^x(3)))*exp(-(x(1)/Rar))));
%   Pso2=((Rre*2^x(3)/(Rre*2^x(3)+Rrb)*exp(-(x(2)/Rrb+(x(2)+1-
2^x(3))/(Rre*2^x(3))))*(1-...
%
(x(2)*Rar/(x(1)*Rrb)+x(2)*Rar/(x(1)*Rre*2^x(3)))/(1+x(2)*Rar/(x(1)*Rrb)
+x(2)*Rar/(x(1)*Rre*2^x(3)))*exp(-(x(1)/Rar))))+...
```

```
%        Rre*2^x(3)/(Rre*2^x(3)+Rrb)*((1-
1/(1+x(2)*Rar/(x(1)*Rrb)+x(2)*Rar/(x(1)*2^x(3)*Rre)))*exp(-
(x(1)/Rar+x(2)/Rrb+(x(2)+1-2^Ra)/(2^x(3)*Rre)))-...
%        exp(-((2^Ra-1)/Rar+x(2)/Rrb+(x(2)+1-
2^Ra)/(2^x(3)*Rre)))+1/(1+x(2)*Rar/(x(1)*Rrb)+x(2)*Rar/(x(1)*2^x(3)*Rre
))*...
%        exp(-((2^Ra-1)/Rar+x(2)*(2^Ra-1)/(x(1)*Rrb)+(x(2)*(2^Ra-1)+1-
2^Ra)/(x(1)*2^x(3)*Rre)))))/...
%        (exp(-x(2)/Rrb)+x(1)*Rrb*exp(-((2^Ra-1)/Rar+x(2)*(2^Ra-
1)/(x(1)*Rrb)))/(x(1)*Rrb+x(2)*Rar)-exp(-(x(2)/Rrb+(2^Ra-
1)/Rar)));%%a<2^Ra-1
 %  Pso=1-(1-Pso1)*(1-Pso2);
   c(1)=(2^x(3)-1)-x(1);
   c(2)=(2^x(3)-1)-x(2);
   c(3)=Pr-Pa;
   c(4)=a-Pr*x(3);
   c(5)=x(3)-Ra;
   c(6)=x(1)-2^Ra+1;
   ceq=[];
   c=[c(1),c(2),c(3),c(4),c(5),c(6)];
end
```