Alternating Bit Protocol (ABP) pri slanju okvira podataka postavlja vreme pauze (TIMEOUT) koje ističe ukoliko u zadatom vremenu ne pristigne paket potvrde za poslati okvir. Predajnik je sinhronizovan sa prijemnikom korišćenjem bita na strani predaje i prijema. Ukoliko primljeni okvir nosi isti bit (bit predajnika) kao i bit na strani prijema okvir ce biti prihvacen, a paket potvrde će biti poslat ka predajniku. Pri tome se menja i bit na strani prijema (iz 0 u 1 i iz 1 u 0). Predajnik menja bit ukoliko primi paket potvrde. Ukoliko istekne vreme pauze bit predajnika se ne menja.

U ovom primeru se 1024 bita podataka šalje linijom brzine 4096 bita po sekundi. Iz tog razloga vreme slanja podataka traje 0.25 s. Vreme isteka pauze traje 0.52 sekunde. Paket potvrde je iste veličine kao i paket podataka. Potrebno je odrediti broj uspešno poslatih paketa, za zadatu verovatnoću bitske greške.

```
// Datoteka zad2.cpp
// Simulacija ABP (Alternating Bit Protocol) protokola

#include "aes.h"
#include <iostream>

// Ukljucujemo prostore imena aes i std
using namespace aes;
using namespace std;

// Konstante
const uint32_t velicina_paketa         = 1024; // bita
const uint32_t velicina_paketa_potvrde = 1024; // bita
const uint32_t protok_linije           = 4096; // bit/s

// Bit Error Rate
const double   BER                     = 1E-4;
// Verovatnoca da ce poslati paket biti primljen (da stize bez greske)
const double   prob                    = 1-(BER*velicina_paketa);

// Vreme slanja paketa
const double   vreme_paket             = 0.25;
// Vreme pristiznje potvrde
const double   vreme_potvrda           = 0.25;
// Vreme isteka pauze (TIMEOUT)
const double   vreme_isteka_pauze      = 0.52;


// Funkcija vraca slucajan broj na intervalu od 0 do 1
double random() { return rand()/(1.0*RAND_MAX); }

// Deklaracija unapred klasa predajnik i prijemnik
class predajnik;
class prijemnik;

using byte = unsigned char;

// Klasa dogadjaja slanja paketa
class slanje: public dogadjaj {
  // Pokazivac na objekat simulacije
  simulacija* _sim;
public:
  // Konstruktor
  slanje(simulacija* s,predajnik* p,vreme vn);
public:
  // Akcija dogadjaja
  void akcija();
};

// Klasa dogadjaja prijema paketa
class primi: public dogadjaj {
```

```
  // Pokazivac na objekat simulacije
  simulacija* _sim;
public:
  // Konstruktor
  primi(simulacija* s,prijemnik* p,const vreme vn);
public:
  // Akcija dogadjaja
  void akcija();
};

// Klasa dogadjaja isteka pauze
class istek_pauze: public dogadjaj {
  // Pokazivac na objekat simulacije
  simulacija* _sim;
  // Istek pauze se ne odigrava zato sto je paket potvrde stigao
  bool _pasivan;
public:
  // Konstruktor
  istek_pauze(simulacija* s,predajnik* p,const vreme vn);
public:
  // Pasivizacija isteka pauze
  void pasivizuj() {
    _pasivan = true;
  }
  // Akcija dogadjaja
  void akcija();
};

// Klasa dogadjaja prijema potvrde
class prijem_potvrde: public dogadjaj {
  // Pokazivac na objekat simulacije
  simulacija* _sim;
public:
  // Konstruktor
  prijem_potvrde(simulacija* s,predajnik* p,const vreme vn);
public:
  // Akcija dogadjaja
  void akcija();
};

// Klasa agenta predajnika
class predajnik: public agent {
  // Bit protokola na predaji
  byte _bit;
  // Pokazivac na objekat simulacije
  simulacija* _sim;
  // Pokazivac na objekat prijemnika
  prijemnik*  _prijemnik;
  // Pokazivac na istek pauze (timeout)
  istek_pauze *_istek_pauze;
public:
  // Konstruktor
  predajnik(simulacija* s):_bit(0),_sim(s) {}
public:
  // Vrati i postavi bit
  byte vrati_bit() const { return _bit; }
  void postavi_bit(const byte b) { _bit = b; }
  // Vrati i postavi istek pauze
  istek_pauze* vrati_istek_pauze() const { return _istek_pauze; }
  void postavi_istek_pauze(const istek_pauze *ip) {
    _istek_pauze = const_cast<istek_pauze*>(ip);
  }
  // Postavi prijemnik
  void postavi_prijemnik(const prijemnik* p) {
    _prijemnik = const_cast<prijemnik*>(p);
  }
  // Postavi prijemnik
  prijemnik* vrati_prijemnik() const {  return _prijemnik;  }
public:
  // Posalji paket
  void posalji_paket() {
```

```
    // Raporedjujemo dogadjaj slanja paketa (za predajnik)
    _sim->rasporedi(new slanje(_sim,this,0.0));
  }
};

// Klasa agenta prijemnika
class prijemnik: public agent {
  // Bit protokola na prijemu
  byte   _bit;
  // Broj primljenih bita
  size_t _broj_primljenih_bita;
  // Pokazivac na objekat simulacije
  simulacija* _sim;
  // Pokazivac na objekat predajnika
  predajnik*  _predajnik;
public:
  // Konstruktor
  prijemnik(simulacija* s):_bit(0),_broj_primljenih_bita(0),_sim(s) {}
public:
  // Vrati i postavi bit
  byte vrati_bit() const { return _bit; }
  void postavi_bit(const byte b) { _bit = b; }
  // Postavi predajnika
  void postavi_predajnik(const predajnik* p) {
    _predajnik = const_cast<predajnik*>(p);
  }
  // Primi paket
  void primi_paket() {
    // Ukoliko se biti predjnika i prijemnika slazu
    if(_bit == _predajnik->vrati_bit()) {
      // Menjamo bit na prijemu
      _bit = !_bit;
      // Uvecavamo broj primljenih bita poruke
      _broj_primljenih_bita += velicina_paketa;
      // Rasporedi dogadjaj prijema potvrde (za predajnik)
      _sim->rasporedi(new prijem_potvrde(_sim,_predajnik,vreme_potvrda));
    }
  }
};

// Konstruktor dogadjaja slanje
slanje::slanje(simulacija* s,predajnik* p,vreme vn):dogadjaj(1),_sim(s) {
  postavi_agenta(0,p);
  postavi_vreme(_sim->vrati_vreme()+vn);
}

// Akcija dogadjaja slanja paketa
void slanje::akcija() {
  // Predajnik
  predajnik* pred = dynamic_cast<predajnik*>(vrati_agenta(0));

  // Po slanju paketa rasporedjuje se dogadjaj isteka pauze (za predajnik)
  _sim->rasporedi(new istek_pauze(_sim,pred,vreme_isteka_pauze));

  if(random()<prob) {
    // Po slanju paketa rasporedjuje se dogadjaj prijema paketa (za prijemnik)
    _sim->rasporedi(new primi(_sim,pred->vrati_prijemnik(),vreme_paket));
  }
}

// Konstruktor dogadjaja primi
primi::primi(simulacija* s,prijemnik* p,const vreme vn):dogadjaj(1),_sim(s) {
  postavi_agenta(0,p);
  postavi_vreme(_sim->vrati_vreme()+vn);
}

// Akcija dogadjaja prijema paketa
void primi::akcija() {
  // Prijemnik
  prijemnik *pri = dynamic_cast<prijemnik*>(vrati_agenta(0));
  // Prijemnik prima paket
```

```
  pri->primi_paket();
}

// Konstruktor dogadjaja istek pauze
istek_pauze::istek_pauze(simulacija* s,predajnik* p,const vreme
vn):dogadjaj(1),_sim(s),_pasivan(false)  {
  p->postavi_istek_pauze(this);
  postavi_agenta(0,p);
  postavi_vreme(_sim->vrati_vreme()+vn);
}

// Akcija dogadjaja istek pauze
void istek_pauze::akcija() {
  // Predajnik
  predajnik *pred = dynamic_cast<predajnik*>(vrati_agenta(0));
  // Ukoliko istek pauze nije pasivizovan
  if(!_pasivan)
    // saljemo novi paket.
    pred->posalji_paket();
}

// Konstruktor dogadjaja prijema potvrde
prijem_potvrde::prijem_potvrde(simulacija* s,predajnik* p,const vreme vn):dogadjaj(1),_sim(s) {
  postavi_agenta(0,p);
  postavi_vreme(_sim->vrati_vreme()+vn);
}

// Akcija dogadjaja prijem potvrde
void prijem_potvrde::akcija() {
  // Predajnik
  predajnik *pred = dynamic_cast<predajnik*>(vrati_agenta(0));
  // Pasivizujemo istek pauze
  pred->vrati_istek_pauze()->pasivizuj();
  // Menjamo bit na predaji
  pred->postavi_bit(!pred->vrati_bit());
  // Saljemo novi paket;
  pred->posalji_paket();
}

int main() {
  // Objekat simulacije
  simulacija sim;
  // Predajnik
  predajnik tx(&sim);
  // Prijemnika
  prijemnik rx(&sim);

  // Uspostavljamo vezu izmedju prijemnika i predajnika
  tx.postavi_prijemnik(&rx);
  rx.postavi_predajnik(&tx);

  // Predajnik salje paket prijemniku
  tx.posalji_paket();

  // Zapocinjemo simulaciju
  sim.izvrsi();

  return 0;
}
```