FERNANDO HAFNER
CÉSAR KALLAS

# REDES DE COMPUTADORES II

Relatório da Tarefa de Programação I

Pontifícia Universidade Católica de Campinas
Faculdade de Engenharia da Computação
Fevereiro /2006

Fernando Hafner R.A. 01002609 email: fernando.hafner@gmail.com
César Kallas R.A. 02099224 email: cesarkallas@gmx.net

Divisão das Notas :

Fernando Hafner - 50%
César Kallas - 50%

# PROTOCOLO DE TRANSMISSÃO CONFIÁVEL/KUROSSE

*Trabalho requisitado pelo professor de Redes de Computadores II, ministrada pelo professor Juan, da Pontifícia Universidade Católica, com o intuito de avaliação acadêmica dos alunos, para efeito de cálculo da média final.*

Pontifícia Universidade Católica de Campinas
Faculdade de Engenharia da Computação
Fevereiro /2006

# 1 ) Nível de Atendimento aos Requisitos da Atividade

Para a atividade de Protocolo de Transmissão Confiável, foi requisitado aos aluno da disciplina Redes de Computadores II, que fossem desenvolvidas as atividades relacionadas ao protocolo Bit Alternante e ao protocolo Go-Back-N. Ambas as atividades foram desenvolvidas pelo grupo. A atividade pede que na implementação dos protocolos seja utilizado tanto mensagens ACK como NACK, no entanto, no protocolo apresentado no livro (Kurosse e Ross, 2003 (1ª edição) ) , o mesmo trata somente uma versão que utiliza somente mensagens ACK . O grupo adotou a implementação do livro como referência.

# 2 ) Descrição das Rotinas Implementadas

Segue abaixo a descrição do que foi implementado em cada uma das rotinas :

### 2.1) Rotinas do Protocolo GBN

*A_output()* – a função A_output() apenas recebe a informação necessária da camada acima e monta um PDU contendo essa informação, além dos campos adicionais que serão utilizados para garantir a confiabilidade do protocolo. Vale lembrar que nem todos os campos da estrutura que nos foi apresentada para ser usada como o PDU da camada 4 é efetivamente utilizado ( p.e.: No envio do pacote de A para B, o campo *acknum* não é utilizado ). A função A_output() não realize efetivamente a saída dos pacotes, apenas os deixa marcados como prontos para serem enviados.

*A_input()* – a função A_input() recebe os pacotes provenientes de B. Como para a atividade desenvolvida foi considerado apenas o cenário onde há transferência de A para B, B só irá enviar pacotes de ACK para A. Nesse caso, vale salientar o que já foi dito na descrição da função A_output() que nem todos os campos da estrutura do PDU é utilizado. Para pacotes ACK, somente os campos de acknum e checksum são utilizados. Além disso, a medida que vai recebendo as confirmações de entrega dos pacotes, a função A_input() vai avançando a janela para que novos pacotes sejam enviados.

*A_timerinterrupt()* – a função A_timerinterrupt é chamada sempre que um temporizador é expirado. Depois que o temporizador expira, todos os pacotes que foram marcados como prontos na função A_output() são enviados e o temporizador é novamente disparado.

*A_init()* - a função A_init() inicializa algumas variáveis como número de seqüência inicial e dispara o temporizador.

*B_init()* - a função B_init() apenas inicializa a variável de número de seqüência inicial esperado para zero.

*B_input()* – a função B_input() recebe os pacotes provenientes de A. Depois de recebido, a função B_input() faz uma verificação do pacote : se o mesmo não estiver corrompido ( valores iguais entre o valor recebido e valor calculado) e o número de seqüência recebido é o mesmo que o número de seqüência esperado; então a carga útil do pacote é repassada a camada de cima e um pacote contendo uma resposta é montado. Se o pacote proveniente de A foi recebido com sucesso então uma resposta ACK é criada ( com número igual ao número de seqüência ).

*chksum()* – a função chksum() recebe como parâmetro todos os campos do pacote para calcular a soma de verificação, utilizada na detecção de erros. Para isso, uma simples soma de inteiros é feita de todos os campos do pacote.

**2.2)Rotinas do Protocolo Bit Alternante**

*float TEMPO_LIMITE = 25* - Tempo limite para entrega de um pacote, usado na função starttimer(a ou b, TEMPO_LIMITE)

*short int SEMAFARO* - Para controle de mensagens em transito, se tiver alguma mensagem sendo enviada por A essa variavel vai valer 1

*int sequencia_pacote, pacote_esperado* - Indice de envio e recebimento de pacotes

*struct pkt __pacote* - Estrutura para o pacote em transito

*getCheckSum(char *buffer, int len_buffer)* - a função getCheckSum() recebe como parâmetro todos os campos do pacote para calcular a soma de verificação, utilizada na detecção de erros. Para isso, uma simples soma de inteiros é feita de todos os campos do pacote.

*A_output(message) struct msg message* - Funcao que recebe o buffer(message) para ser transmitido da camada 5 (aplicacao). Essa funcao recebe o dado e envia para a camada 3 (rede) remetida pelo [A].

*A_input(packet) struct pkt packet* - Essa funcao e' chamada quando um pacote vem da camada 3 pra ser enviado para a camada 4 no [A], ou seja, quando [A] recebe um ACK (confirmação de recebimento de mensagem).

*A_timerinterrupt()* - Essa funcao eh chamada quando o tempo limite de [A] esgosta. Depois que o temporizador expira, todos os pacotes que foram marcados como prontos na função A_output() são enviados e o temporizador é novamente disparado.

*A_init()* - Essa funcao eh chamada no inicio da execucao, antes de qualquer outra funcao. Funcao de inicializacao do [A].

*B_output(message)struct msg message* - Funcao que recebe o buffer(message) para ser transmitido da camada 5 (aplicacao). Essa funcao recebe o dado e envia para a camada 3 (rede) remetida pelo [B].

*B_input(packet) struct pkt packet* - Essa funcao é chamada quando um pacote vem da camada 3 pra ser enviado para a camada 4 no [B]

*B_timerinterrupt()* - Essa funcao eh chamada quando o tempo limite de [B] esgosta

## 3 ) Instruções para Geração e Execução do Programa

Para geração do programa utilizando o ambiente Linux e o compilador GCC, basta compilar o arquivo fonte normalmente sem nenhum parâmetro adicional. O mesmo vale para a execução do programa.

**Exemplo**:

```
[chk@cesarkallas 1-bit-alternate-gbn]$ make
gcc bit-alternante.c -o bit-alternante
gcc gbn.c -o gbn

[chk@cesarkallas 1-bit-alternate-gbn]$ ./bit-alternante
[chk@cesarkallas 1-bit-alternate-gbn]$ ./gbn
```

## 4 ) Opiniões sobre a Atividade

A atividade permitiu relembrar conceitos apresentados na disciplina Redes de Computadores I, como a transferência de dados confiável e os protocolos apresentados para garantir essa confiabilidade. Além disso, também podemos observar o conceito das camadas de rede demonstrado no programa no qual tivemos como base o desenvolvimento das nossas atividades.

### 4.1 ) Tempo Gasto

O tempo total gasto para o desenvolvimento da atividade ( leitura/estudo da documentação e implementação) foi de 4 dias; dedicando aproximadamente 2 horas por dia.

### 4.2 ) Considerações Positivas

O programa que nos foi apresentado como base nos permitiu observar como funciona o protocolo Bit Alternante na prática. A flexibilidade do programa base, ajudou consideravelmente na execução de testes, podendo simular cenários específicos onde todos os pacotes são corrompidos, todos os pacotes são perdidos ou ainda uma combinação probabilística dos dois cenários.

### 4.3 ) Considerações Negativas

A atividade pede que seja utilizado as mensagens ACK e NACK, no entanto no protocolo apresentado por Kurosse e Ross, 2003 (1ª edição) , se utiliza apenas mensagens do tipo ACK. Na própria definição da estrutura que nos foi dada para ser utilizada como o pacote, não há distinção de ACK e NACK.

### 4.4 ) Sugestões para Melhoria da Atividade

O documento apresentado possui algumas inconsistências quanto a utilização da versão inicial do programa. Para a compilação do mesmo tanto em ambientes Linux e Windows, são necessárias algumas alterações no programa. No documento é dito que é possível apenas “copiar e colar” o código e compilá-lo, o que não é verdade.

Outra modificação que poderia ser feita, é a alteração da estrutura utilizada como pacote, adicionando-se um campo do tipo de mensagem, caso queira se utilizar ACK e NACK.

## Anexo 1 – Fonte dos programas

### *GBN.C*

```
#include <stdio.h>
#include <stdlib.h>


/* ******************************************************************
 ALTERNATING BIT AND GO-BACK-N NETWORK EMULATOR: VERSION 1.1  J.F.Kurose

   This code should be used for PA2, unidirectional or bidirectional
   data transfer protocols (from A to B. Bidirectional transfer of data
   is for extra credit and is not required).  Network properties:
   - one way network delay averages five time units (longer if there
     are other messages in the channel for GBN), but can be larger
   - packets can be corrupted (either the header or the data portion)
     or lost, according to user-defined probabilities
   - packets will be delivered in the order in which they were sent
     (although some can be lost).
*******************************************************************/

#define BIDIRECTIONAL 0 /* change to 1 if you're doing extra credit */
                  /* and write a routine called B_output */
#define TAM_JANELA 8
#define TAM_BUFFER 50


/* a "msg" is the data unit passed from layer 5 (teachers code) to layer
*/
/* 4 (students' code).  It contains the data (characters) to be delivered
*/
/* to layer 5 via the students transport level protocol entities.
*/

struct msg {
   char data[20];
    };

/* a packet is the data unit passed from layer 4 (students code) to layer
*/
/* 3 (teachers code).  Note the pre-defined packet structure, which all
*/
/* students must follow. */

struct pkt {
   int seqnum;
   int acknum;
   int checksum;
   char payload[20];
    };



struct pkt pacote[TAM_BUFFER];      // vetor com pacotes que serao enviados.
int falta_enviar[TAM_BUFFER];       // vetor com pacotes faltam ser
enviados.
int numseq_esperado;          // numero de sequencia esperado por B.
int prox_pacote;
```

```
int limite;             // limite da janela
int ack_esperado;



/********* STUDENTS WRITE THE NEXT SEVEN ROUTINES *********/

int chksum(int seqnum, int acknum, char payload[20]) // Soma de Verificacao
{
      int soma = seqnum + acknum;
      int i=0;

      for(i; i<20 ; i++)
      {
            soma += payload[i];
      }

      return(soma);

}

/* called from layer 5, passed the data to be sent to other side */

A_output(message) struct msg message;
{
      pacote[prox_pacote].seqnum = prox_pacote % TAM_JANELA;
      pacote[prox_pacote].checksum = chksum(pacote[prox_pacote].seqnum,
pacote[prox_pacote].acknum, message.data);

      int i=0;

      for ( i ; i<20 ; i++)   // Copia mensagem para campo de carga util
      {
            pacote[prox_pacote].payload[i] = message.data[i];
      }

      falta_enviar[prox_pacote] = 1;      // Marca pacote para ser enviado

      prox_pacote = ++prox_pacote % TAM_BUFFER; // Proximo pacote
}

/* called from layer 3, when a packet arrives for layer 4 */

A_input(packet) struct pkt packet;
{

   printf("Recebi ACK de pacote...\n");

   if(chksum(packet.seqnum, packet.acknum, packet.payload) ==
packet.checksum)
      {

            printf("Pacote do ACK OK \n");
            printf("Recebi ACK do pacote %d\n", packet.acknum);

            if(packet.acknum == (ack_esperado % TAM_JANELA))
            {

               falta_enviar[ack_esperado] = 0;  // marco pacote como enviado
```

```
                stoptimer(0);
                starttimer(0, (float) 25);

                ack_esperado = ++ack_esperado % TAM_BUFFER;

                limite = ++limite % TAM_BUFFER;  // "deslizo" janela

            }
      }

   else

      printf("Pacote de ACK foi corrompido!\n");

}

/* called when A's timer goes off */

A_timerinterrupt()

{
      printf("Temporizador de A expirou...reenviando pacotes da janela\n");

      int i;
      for(i = 0; i < TAM_JANELA; i++)
      {
            // Se existe pacotes da janela a serem enviados, os envio
            if(falta_enviar[(ack_esperado + i) % TAM_BUFFER] == 1)
            {
                  printf("Reenviando pacote %d ... \n",
pacote[(ack_esperado + i) % TAM_BUFFER]);
                  tolayer3(0, pacote[(ack_esperado + i) % TAM_BUFFER]);
            }
            else break; // nao tenho mais pacotes da janela a serem
enviados
      }

      starttimer(0, (float) 25);  // Comeco denovo o temporizador

}

/* the following routine will be called once (only) before any other */

/* entity A routines are called. You can use it to do any initialization */

A_init()

{
      int i;
      for(i = 0; i < TAM_BUFFER; i++)
            falta_enviar[i] = 0;

      ack_esperado = 0;
      limite = TAM_JANELA-1;
      prox_pacote = 0;
      starttimer(0, (float) 25);

}


B_output(message)  /* need be completed only for extra credit */
```

```
  struct msg message;
{

}


/* called from layer 3, when a packet arrives for layer 4 at B*/

B_input(packet) struct pkt packet;
{

      printf("Recebi pacote de A...\n", packet.seqnum);

      if(chksum(packet.seqnum, packet.acknum, packet.payload) ==
packet.checksum)
      {
            printf("Pacote esta' OK...\n");

            if(packet.seqnum == numseq_esperado)
            {
                  tolayer5(packet.payload);                        // envia
para camada de cima
                  numseq_esperado = ++numseq_esperado % TAM_JANELA;    //
Proximo numero de sequencia
            }


            // Monto resposta ACK
            struct pkt resposta;
            resposta.acknum = packet.seqnum;
                  resposta.checksum = chksum(resposta.seqnum,
resposta.acknum, resposta.payload);
            printf("Enviando ACK de resposta para pacote %d\n",
resposta.acknum);
                  tolayer3(1, resposta);

      }
      else
                  printf("Pacote foi Corrompido!\n");

}

/* called when B's timer goes off */

B_timerinterrupt()
{

}

/* the following routine will be called once (only) before any other */

/* entity B routines are called. You can use it to do any initialization */

B_init()
{
   numseq_esperado = 0;
}
```

## bit-alternante.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* ******************************************************************
 ALTERNATING BIT AND GO-BACK-N NETWORK EMULATOR: VERSION 1.1  J.F.Kurose

   This code should be used for PA2, unidirectional or bidirectional
   data transfer protocols (from A to B. Bidirectional transfer of data
   is for extra credit and is not required).  Network properties:
   - one way network delay averages five time units (longer if there
     are other messages in the channel for GBN), but can be larger
   - packets can be corrupted (either the header or the data portion)
     or lost, according to user-defined probabilities
   - packets will be delivered in the order in which they were sent
     (although some can be lost).
*******************************************************************/

#define BIDIRECTIONAL 0 /* change to 1 if you're doing extra credit */
                                                                    /*
and write a routine called B_output */

#define LEN_PAYLOAD 20

/* a "msg" is the data unit passed from layer 5 (teachers code) to layer
*/
/* 4 (students' code).  It contains the data (characters) to be delivered
*/
/* to layer 5 via the students transport level protocol entities.
*/
struct msg {
  char data[LEN_PAYLOAD];
};

/* a packet is the data unit passed from layer 4 (students code) to layer
*/
/* 3 (teachers code).  Note the pre-defined packet structure, which all
*/
/* students must follow. */
struct pkt {
   int seqnum;
   int acknum;
   int checksum;
   char payload[LEN_PAYLOAD];
};

// Tempo limite para entrega de um pacote
float TEMPO_LIMITE = 25;

// Para controle de mensagens em transito, se tiver alguma mensagem sendo
enviada por A essa variavel vai valer 1
short int SEMAFARO;

// Indice de envio e recebimento de pacotes
int sequencia_pacote, pacote_esperado;

// Estrutura para pacote em transito
struct pkt __pacote;
```

```
/********* STUDENTS WRITE THE NEXT SEVEN ROUTINES ********/


/**
      Retornar o checksum de uma cadeia do tipo char (buffer)
**/
int getCheckSum(char *buffer, int len_buffer)
{
      int checksum = len_buffer;
      int i;

      for(i=0; i < len_buffer; i++) {
            checksum =+ (int)(buffer[i]);
      }
      return checksum;
}


/**
      Funcao que recebe o buffer(message) para ser transmitido da camada 5
(aplicacao)
      Essa funcao recebe o dado e envia para a camada 3 (rede) remetida
pelo [A]
**/
A_output(message) struct msg message;
{
      // Verifica se nao ha uma mensagem em transito, se estiver, ignora
essa mensagem nova
      if(SEMAFARO == 1) {
            printf("    [A] descartando a msg(%d) - nao pode ser
recebida\n", getCheckSum(message.data, LEN_PAYLOAD));
            return;
      }
      else
            SEMAFARO = 1;

      // Gera o pacote a ser enviado
      int checksum = getCheckSum(message.data, LEN_PAYLOAD);
      checksum += sequencia_pacote + __pacote.acknum;
      strcpy(__pacote.payload, message.data);
      __pacote.checksum = checksum;
      __pacote.seqnum = sequencia_pacote;

      // Envia o novo pacote e inicia o cronometro
      printf("    [A] enviando pacote: ack(%d) : seq(%d) : checksum(%d)\n",
__pacote.acknum, __pacote.seqnum, __pacote.checksum);
      tolayer3(0, __pacote);
      starttimer(0, TEMPO_LIMITE);
}

/**
      Essa funcao e' chamada quando um pacote vem da camada 3
      pra ser enviado para a camada 4 no [A]
**/
A_input(packet) struct pkt packet;
{
      // Verifica se o checksum do pacote recebido
      int checksum = getCheckSum(packet.payload, LEN_PAYLOAD);
      checksum += packet.seqnum + packet.acknum;
```

```
      if(checksum != packet.checksum) {
            printf("    [A] falha checksum: ack(%d) : seq(%d) :
checksum(%d) : status(""corrompido ~ checksum esperado(%d)"")\n",
packet.acknum, packet.seqnum, packet.checksum, checksum);
            return;
      }

      // Verifica se a resposta ACK foi de um pacote enviado
      if(packet.acknum != sequencia_pacote) {
            printf("    [A] falha ack: ack(%d) : seq(%d) : checksum(%d) :
status(""nao reconhecido"")\n", packet.acknum, packet.seqnum,
packet.checksum);
            return;
      }

      // para o tempo de espera
      printf("    [A] ACK reconhecido: ack(%d) : seq(%d) : checksum(%d) :
status(""ok"")\n", packet.acknum, packet.seqnum, packet.checksum);
      stoptimer(0);

      // aumenta para o novo frame a ser enviado
      sequencia_pacote += 1;

      // libera o [A] para novas mensagens
      SEMAFARO = 0;
}

/**
      Essa funcao eh chamada quando o tempo limite de [A] esgosta
**/
A_timerinterrupt()
{
      printf("    [A] tempo limite esgotado para envio do pacote:
checksum(%d)\n", __pacote.checksum);

      // Reenvia o pacote
      A_output(__pacote.payload);
}

/**
      Essa funcao eh chamada no inicio da execucao, antes de qualquer outra
funcao
      Funcao de inicializacao do [A]
**/
A_init()
{
      pacote_esperado = 0;
      //__pacote.seqnum = 0;
      //__pacote.acknum = 0;
      SEMAFARO = 0;
}

/**
      Funcao que recebe o buffer(message) para ser transmitido da camada 5
(aplicacao)
      Essa funcao recebe o dado e envia para a camada 3 (rede) remetida
pelo [B]
**/
B_output(message)struct msg message;
{
}
```

```
/**
      Essa funcao e' chamada quando um pacote vem da camada 3
      pra ser enviado para a camada 4 no [B]
**/
B_input(packet) struct pkt packet;
{
      // Verifica o checksum do pacote recebido
      int checksum = getCheckSum(packet.payload, LEN_PAYLOAD);
      checksum += packet.seqnum + packet.acknum;

      struct pkt response;

      // Compara o checksum do pacote recebido com o gerado no envio do
pacote, se nao for igual cancela o recebimento
      if(checksum != packet.checksum) {
            printf("    [B] falha checksum: ack(%d) : seq(%d) :
checksum(%d) : status(""corrompido"")\n", packet.acknum, packet.seqnum,
packet.checksum);
            return;
      }

      // Verifica se o pacote que chegou foi o enviado, se nao for igual
ignora
      if(packet.seqnum != pacote_esperado) {
            printf("    [B] falha de sequencia: ack(%d) : seq(%d) :
checksum(%d) : status(""nao reconhecido"")\n", packet.acknum,
packet.seqnum, packet.checksum);
            return;
      }

      // Confirma o recebimento do pacote correto e o repassa para a camada
5
      printf("    [B] novo pacote: ack(%d) : seq(%d) : checksum(%d) :
status(""ok"")\n", packet.acknum, packet.seqnum, packet.checksum);
      printf("    [B] repassando o pacote para a camada 5: ack(%d) :
seq(%d) : checksum(%d) : status(""ok"")\n", packet.acknum, packet.seqnum,
packet.checksum);
      tolayer5(packet.payload);

      // Aumenta para o proximo frame esperado
      pacote_esperado += 1;

      printf("    [B] confirmando o recebimento com o ACK: ack(%d) :
seq(%d) : checksum(%d) : status(""ok"")\n", packet.acknum, packet.seqnum,
packet.checksum);

      // Confirma o recebindo do pacote com um ACK
      response.acknum = packet.seqnum;
      response.checksum = getCheckSum(response.payload, LEN_PAYLOAD);
      response.checksum += response.seqnum + response.acknum;
      tolayer3(1, response);
}

/**
      Essa funcao eh chamada quando o tempo limite de [B] esgosta
**/
B_timerinterrupt()
{
      sequencia_pacote = 0;
}
```

```
/**
      Essa funcao eh chamada no inicio da execucao, antes de qualquer outra
funcao
      Funcao de inicializacao do [B]
**/
B_init()
{
}
```

## Anexo 2 – Saída dos programas

### *Protocolo GBN*

```
-----  Stop and Wait Network Simulator Version 1.1 --------

Enter the number of messages to simulate: 20
Enter  packet loss probability [enter 0.0 for no loss]:0.2
Enter packet corruption probability [0.0 for no corruption]:0.2
Enter average time between messages from sender's layer5 [ > 0.0]:10
Enter TRACE:2

EVENT time: 18.705740,  type: 1, fromlayer5  entity: 0

EVENT time: 25.000000,  type: 0, timerinterrupt   entity: 0
Temporizador de A expirou...reenviando pacotes da janela
Reenviando pacote 0 ...

EVENT time: 30.465096,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 0
          TOLAYER3: packet being corrupted

EVENT time: 35.124840,  type: 1, fromlayer5  entity: 0

EVENT time: 35.696709,  type: 2, fromlayer3  entity: 0
Recebi ACK de pacote...
Pacote de ACK foi corrompido!

EVENT time: 37.680447,  type: 1, fromlayer5  entity: 0

EVENT time: 39.948215,  type: 1, fromlayer5  entity: 0

EVENT time: 48.662170,  type: 1, fromlayer5  entity: 0

EVENT time: 50.000000,  type: 0, timerinterrupt   entity: 0
Temporizador de A expirou...reenviando pacotes da janela
Reenviando pacote 0 ...
Reenviando pacote 1 ...
          TOLAYER3: packet being lost
Reenviando pacote 2 ...
Reenviando pacote 3 ...
Reenviando pacote 4 ...
          TOLAYER3: packet being corrupted

EVENT time: 51.299217,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 0
```

```
EVENT time: 53.665558,  type: 2, fromlayer3  entity: 0
Recebi ACK de pacote...
Pacote do ACK OK
Recebi ACK do pacote 0

EVENT time: 60.727642,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 2
          TOLAYER3: packet being corrupted

EVENT time: 62.511566,  type: 2, fromlayer3  entity: 0
Recebi ACK de pacote...
Pacote de ACK foi corrompido!

EVENT time: 66.983368,  type: 1, fromlayer5  entity: 0

EVENT time: 68.190559,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 3
          TOLAYER3: packet being corrupted

EVENT time: 75.392395,  type: 2, fromlayer3  entity: 0
Recebi ACK de pacote...
Pacote de ACK foi corrompido!

EVENT time: 77.209221,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote foi Corrompido!

EVENT time: 78.647858,  type: 1, fromlayer5  entity: 0

EVENT time: 78.665558,  type: 0, timerinterrupt   entity: 0
Temporizador de A expirou...reenviando pacotes da janela
Reenviando pacote 1 ...
          TOLAYER3: packet being lost
Reenviando pacote 2 ...
          TOLAYER3: packet being corrupted
Reenviando pacote 3 ...
          TOLAYER3: packet being lost
Reenviando pacote 4 ...
Reenviando pacote 5 ...
          TOLAYER3: packet being corrupted
Reenviando pacote 6 ...
          TOLAYER3: packet being corrupted

EVENT time: 88.424805,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote foi Corrompido!

EVENT time: 89.398903,  type: 1, fromlayer5  entity: 0

EVENT time: 89.731194,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 4

EVENT time: 92.609032,  type: 2, fromlayer3  entity: 0
Recebi ACK de pacote...
```

```
Pacote do ACK OK
Recebi ACK do pacote 4

EVENT time: 95.309814,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote foi Corrompido!

EVENT time: 95.853302,  type: 1, fromlayer5  entity: 0

EVENT time: 101.559746,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote foi Corrompido!

EVENT time: 103.665558,  type: 0, timerinterrupt   entity: 0
Temporizador de A expirou...reenviando pacotes da janela
Reenviando pacote 1 ...
Reenviando pacote 2 ...
Reenviando pacote 3 ...
          TOLAYER3: packet being corrupted
Reenviando pacote 4 ...
          TOLAYER3: packet being lost
Reenviando pacote 5 ...
          TOLAYER3: packet being lost
Reenviando pacote 6 ...
          TOLAYER3: packet being corrupted
Reenviando pacote 7 ...
          TOLAYER3: packet being lost
Reenviando pacote 0 ...

EVENT time: 110.784439,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 1

EVENT time: 112.061897,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 2
          TOLAYER3: packet being lost

EVENT time: 114.511261,  type: 1, fromlayer5  entity: 0

EVENT time: 115.538498,  type: 2, fromlayer3  entity: 0
Recebi ACK de pacote...
Pacote do ACK OK
Recebi ACK do pacote 1

EVENT time: 117.679565,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote foi Corrompido!

EVENT time: 119.485901,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote foi Corrompido!

EVENT time: 122.502937,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 0

EVENT time: 124.532890,  type: 1, fromlayer5  entity: 0
```

```
EVENT time: 127.889793,  type: 2, fromlayer3  entity: 0
Recebi ACK de pacote...
Pacote do ACK OK
Recebi ACK do pacote 0

EVENT time: 139.781647,  type: 1, fromlayer5  entity: 0

EVENT time: 140.538498,  type: 0, timerinterrupt   entity: 0
Temporizador de A expirou...reenviando pacotes da janela
Reenviando pacote 2 ...
Reenviando pacote 3 ...
Reenviando pacote 4 ...
Reenviando pacote 5 ...
Reenviando pacote 6 ...
Reenviando pacote 7 ...
Reenviando pacote 0 ...
Reenviando pacote 1 ...

EVENT time: 141.346619,  type: 1, fromlayer5  entity: 0

EVENT time: 142.062210,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 2
          TOLAYER3: packet being lost

EVENT time: 142.489059,  type: 1, fromlayer5  entity: 0

EVENT time: 148.630692,  type: 1, fromlayer5  entity: 0

EVENT time: 151.496704,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 3
          TOLAYER3: packet being lost

EVENT time: 159.521698,  type: 1, fromlayer5  entity: 0

EVENT time: 160.322601,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 4
          TOLAYER3: packet being corrupted

EVENT time: 160.912079,  type: 1, fromlayer5  entity: 0

EVENT time: 165.538498,  type: 0, timerinterrupt   entity: 0
Temporizador de A expirou...reenviando pacotes da janela
Reenviando pacote 2 ...
          TOLAYER3: packet being lost
Reenviando pacote 3 ...
          TOLAYER3: packet being lost
Reenviando pacote 4 ...
          TOLAYER3: packet being lost
Reenviando pacote 5 ...
Reenviando pacote 6 ...
          TOLAYER3: packet being corrupted
Reenviando pacote 7 ...
          TOLAYER3: packet being corrupted
Reenviando pacote 0 ...
```

```
Reenviando pacote 1 ...
          TOLAYER3: packet being lost

EVENT time: 169.145721,  type: 2, fromlayer3  entity: 0
Recebi ACK de pacote...
Pacote de ACK foi corrompido!

EVENT time: 170.170578,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 5
          TOLAYER3: packet being corrupted

EVENT time: 171.319611,  type: 2, fromlayer3  entity: 0
Recebi ACK de pacote...
Pacote de ACK foi corrompido!

EVENT time: 172.915817,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 6

EVENT time: 173.758713,  type: 1, fromlayer5  entity: 0

EVENT time: 180.484329,  type: 2, fromlayer3  entity: 0
Recebi ACK de pacote...
Pacote do ACK OK
Recebi ACK do pacote 6

EVENT time: 181.621902,  type: 1, fromlayer5  entity: 0

EVENT time: 182.403992,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 7

EVENT time: 184.909210,  type: 2, fromlayer3  entity: 1
Recebi pacote de A...
Pacote esta' OK...
Enviando ACK de resposta para pacote 0

EVENT time: 188.091812,  type: 2, fromlayer3  entity: 0
Recebi ACK de pacote...
Pacote do ACK OK
Recebi ACK do pacote 7

EVENT time: 188.674591,  type: 1, fromlayer5  entity: 0

EVENT time: 190.538498,  type: 0, timerinterrupt   entity: 0
 Simulator terminated at time 190.538498
 after sending 20 msgs from layer5
```

## Protocolo Bit Alternante

```
[chk@cesarkallas 1-bit-alternate-gbn]$ make
gcc bit-alternante.c -o bit-alternante
gcc gbn.c -o gbn
[chk@cesarkallas 1-bit-alternate-gbn]$ ./bit-alternante
-----  Stop and Wait Network Simulator Version 1.1 --------
```

```
Enter the number of messages to simulate: 3
Enter  packet loss probability [enter 0.0 for no loss]:0.0
Enter packet corruption probability [0.0 for no corruption]:0.0
Enter average time between messages from sender's layer5 [ > 0.0]:1000
Enter TRACE:2

EVENT time: 1870.573975,  type: 1, fromlayer5  entity: 0
    [A] enviando pacote: ack(0) : seq(0) : checksum(97)

EVENT time: 1876.039062,  type: 2, fromlayer3  entity: 1
    [B] novo pacote: ack(0) : seq(0) : checksum(97) : status(ok)
    [B] repassando o pacote para a camada 5: ack(0) : seq(0) : checksum(97)
: status(ok)
    [B] confirmando o recebimento com o ACK: ack(0) : seq(0) : checksum(97)
: status(ok)

EVENT time: 1881.270630,  type: 2, fromlayer3  entity: 0
    [A] ACK reconhecido: ack(0) : seq(1) : checksum(65) : status(ok)

EVENT time: 3512.484131,  type: 1, fromlayer5  entity: 0
    [A] enviando pacote: ack(0) : seq(1) : checksum(99)

EVENT time: 3514.504639,  type: 2, fromlayer3  entity: 1
    [B] novo pacote: ack(0) : seq(1) : checksum(99) : status(ok)
    [B] repassando o pacote para a camada 5: ack(0) : seq(1) : checksum(99)
: status(ok)
    [B] confirmando o recebimento com o ACK: ack(0) : seq(1) : checksum(99)
: status(ok)

EVENT time: 3518.971924,  type: 2, fromlayer3  entity: 0
    [A] ACK reconhecido: ack(1) : seq(0) : checksum(65) : status(ok)

EVENT time: 5209.403320,  type: 1, fromlayer5  entity: 0
    [A] enviando pacote: ack(0) : seq(2) : checksum(101)

EVENT time: 5214.266602,  type: 2, fromlayer3  entity: 1
 Simulator terminated at time 5214.266602
 after sending 3 msgs from layer5
[chk@cesarkallas 1-bit-alternate-gbn]$ exit


[chk@cesarkallas 1-bit-alternate-gbn]$ ./bit-alternante
-----  Stop and Wait Network Simulator Version 1.1 --------

Enter the number of messages to simulate: 5
Enter  packet loss probability [enter 0.0 for no loss]:0.5
Enter packet corruption probability [0.0 for no corruption]:0.0
Enter average time between messages from sender's layer5 [ > 0.0]:1000
Enter TRACE:2

EVENT time: 1870.573975,  type: 1, fromlayer5  entity: 0
    [A] enviando pacote: ack(0) : seq(0) : checksum(97)
          TOLAYER3: packet being lost

EVENT time: 1895.573975,  type: 0, timerinterrupt   entity: 0
    [A] tempo limite esgotado para envio do pacote: checksum(97)
    [A] descartando a msg(8) - nao pode ser recebida

EVENT time: 3512.484131,  type: 1, fromlayer5  entity: 0
    [A] descartando a msg(98) - nao pode ser recebida
```

```
EVENT time: 4504.727539,  type: 1, fromlayer5  entity: 0
    [A] descartando a msg(99) - nao pode ser recebida

EVENT time: 5028.524414,  type: 1, fromlayer5  entity: 0
    [A] descartando a msg(100) - nao pode ser recebida

EVENT time: 6443.002930,  type: 1, fromlayer5  entity: 0
    [A] descartando a msg(101) - nao pode ser recebida

EVENT time: 7383.361328,  type: 1, fromlayer5  entity: 0
 Simulator terminated at time 7383.361328
 after sending 5 msgs from layer5
[chk@cesarkallas 1-bit-alternate-gbn]$


[chk@cesarkallas 1-bit-alternate-gbn]$ ./bit-alternante
-----  Stop and Wait Network Simulator Version 1.1 --------

Enter the number of messages to simulate: 2
Enter  packet loss probability [enter 0.0 for no loss]:0.0
Enter packet corruption probability [0.0 for no corruption]:0.5
Enter average time between messages from sender's layer5 [ > 0.0]:1000
Enter TRACE:2

EVENT time: 1870.573975,  type: 1, fromlayer5  entity: 0
    [A] enviando pacote: ack(0) : seq(0) : checksum(97)
          TOLAYER3: packet being corrupted

EVENT time: 1876.039062,  type: 2, fromlayer3  entity: 1
    [B] novo pacote: ack(0) : seq(0) : checksum(97) : status(ok)
    [B] repassando o pacote para a camada 5: ack(0) : seq(0) : checksum(97)
: status(ok)
    [B] confirmando o recebimento com o ACK: ack(0) : seq(0) : checksum(97)
: status(ok)

EVENT time: 1878.220703,  type: 2, fromlayer3  entity: 0
    [A] ACK reconhecido: ack(0) : seq(1) : checksum(65) : status(ok)

EVENT time: 3512.484131,  type: 1, fromlayer5  entity: 0
    [A] enviando pacote: ack(0) : seq(1) : checksum(99)

EVENT time: 3517.405518,  type: 2, fromlayer3  entity: 1
 Simulator terminated at time 3517.405518
 after sending 2 msgs from layer5
[chk@cesarkallas 1-bit-alternate-gbn]$
```