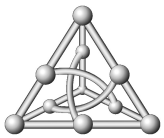

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Trabalho 06: Protocolo do *Bit* Alternante

Redes de Computadores

1 Descrição

Crie um arquivo `.zip` com todo o seu código e envie no Moodle. Você pode fazer este trabalho em um grupo de até três acadêmicos. Seu grupo não pode incluir ninguém que tenha feito parte do seu grupo no Trabalho 05.

Neste trabalho você deve transferir um arquivo de um computador para outro de 100 em 100 *bytes* (o último pacote pode ter menos de 100 *bytes*). Os dados devem ser enviados usando UDP com 1-*byte* de cabeçalho na frente dos dados. O cabeçalho é descrito a seguir.

Você deve escrever dois programas (`sender.c` e `receiver.c`). Cada programa recebe um arquivo como seu único parâmetro por linha de comando. O parâmetro para o *sender* (emissor) é o arquivo a ser enviado e o parâmetro para o *receiver* (receptor) é o nome do arquivo em que será salvo os dados recebidos.

Você deve escolhar uma porta UDP para que o *receiver* use para ficar escutando e para que o *sender* use para enviar para essa porta. O número da porta deve ser definida no cabeçalho do seu código como uma definição que deve ser usada tanto pelo *sender* quanto pelo *receiver*:

```
#define PORT_NUMBER 0
```

(Eu usei um número de porta inválido de propósito para que você escolha o seu próprio número de porta).

O *receiver* deve escutar neste número de porta. O *sender* deve enviar para este número de porta no endereço *localhost* (127.0.0.1, mas veja as opções abaixo), usando um *socket* UDP e a função `sendto`.

O *sender* deve:

- determinar o tamanho do arquivo (Parte 2)
- `malloc` um vetor com o mesmo tamanho do arquivo.
- ler o arquivo para dentro do vetor (depois de abrir o arquivo).
- enviar o conteúdo do vetor, 100 *bytes* por vez.
- cada pacote deve ter 1-*byte* de cabeçalho com o valor 0 ou 1, indicando o número de sequência. O primeiro pacote tem número de sequência 0. O cabeçalho é seguido pelos dados e terá sempre 100 *bytes*, exceto o último pacote.

- depois de enviar cada pacote, seu programa deve esperar um ACK, que deve ser um pacote de 1 *byte* com o valor 2 (para reconhecimento do 0) ou 3 (para reconhecimento do 1). Seu código deve ignorar qualquer pacote recebido que não é um ACK, depois de imprimir uma mensagem apropriada de erro.
- se você não receber um ACK dentro de 2 segundos (as measured by calling alarm – you will need a handler for SIGALRM), retransmita o pacote mais recentemente enviado (repetindo até que você receba o ACK respectivo).
- se o último pacote tiver exatamente 100 *bytes* de dados, envie um pacote adicional com apenas o cabeçalho (número de sequência) e nenhum dado. Se o último pacote tiver menos do que 100 *bytes* de dados, não envie o pacote com zero *bytes*.
- uma vez que o último pacote recebeu seu ACK correspondente, imprima uma mensagem informando que o arquivo foi transmitido corretamente.

De forma correspondente, o *receiver* deve:

- verificar se o arquivo já existe. Se existir, imprima uma mensagem de erro e encerre o programa.
- criar o arquivo.
- ler os pacotes repeditamente, distinguindo novos pacotes de outros retransmitidos (e descartando qualquer pacote inválido).
- salvar no arquivo os dados de cada pacote.
- enviar um ACK para cada pacote recebido. ACK's devem ser enviados tanto para novos pacotes quanto para pacotes que foram retransmitidos. Os ACK's devem ser enviados para o endereço do qual o pacote foi recebido.
- uma vez que o pacote recebido seja inferior a 100 *bytes* de dados, salve o último pacote (se ele tiver mais do que 0 *bytes* de dados), devolva um ACK, feche o arquivo e encerre o programa.

Coloque o código dos seus dois programas no pacote para ser enviado. O pacote de códigos deve incluir os arquivos `sender.c`, `receiver.c`, e o arquivo de cabeçalho compartilhado pelos dois programas e alguns arquivos fonte para teste de envio e recepção (não envie arquivos binários).

2 Medindo o Tamanho do Arquivo

Você pode usar o código a seguir para calcular o tamanho de um arquivo:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

static long file_size (char * file_name)
{
  struct stat st;
  if (stat (file_name, &st) == 0)
    return st.st_size;
  return 0;
}
```

3 Testando Retransmissão

Normalmente, o *receiver* será iniciado antes do *sender*. A forma mais simples de verificar se a retransmissão está funcionando é não iniciar o *receiver* ou então iniciá-lo só após algum tempo que o *sender* já está tentando enviar o arquivo. Dessa maneira, o *sender* deverá continuar tentando enviar o primeiro pacote indefinidamente até conseguir.

Para fazer um teste mais elaborado (mas não é necessário neste trabalho), você pode descartar o pacote (ao invés de enviá-lo) se, por exemplo:

```
time(NULL) % 10 == 0
```

4 Opções

Se você quiser, pode permitir que o *sender* tenha um segundo parâmetro que é o domínio ou número IP da máquina que está executando o *receiver*, e enviar para este endereço ao invés do *localhost*.

Se você quiser, ao invés de ler todo o arquivo e armazená-lo no vetor alocado, o *sender* pode ler o arquivo a cada 100 *bytes* em um *buffer* de tamanho fixo de 100 *bytes* (ou 101 *bytes*), enviar o que foi lido e repetir o processo.