Trabalho Prático 3

Data de entrega: **17/06/2019**

Problema

Implementar um par de programas que operem no modelo cliente-servidor e exercitem tanto a transmissão unidirecional quanto a comunicação do tipo requisição-resposta sobre o protocolo UDP, utilizando um protocolo de janela deslizante. A implementação deve utilizar a biblioteca de sockets do Unix (Linux).

Operação

O processo de comunicação entre cliente e servidor deverá seguir um padrão simples, conforme ilustrado a seguir:

Cliente:
processa argumentos da linha de comando:
`host_do_servidor porto_servidor nome_arquivo tam_buffer tam_janela`
chama `gettimeofday` para tempo inicial
envia string com nome do arquivo (terminada em zero)
abre arquivo que vai ser gravado - pode ser `fopen(nome,"w+")`
loop `recv` buffer até que perceba que o arquivo acabou
escreve bytes do buffer no arquivo (`fwrite`)
atualiza contagem de bytes recebidos
fim_loop
fecha arquivo
chama gettimeofday para tempo final e calcula tempo gasto
imprime resultado: `"Buffer = %5u byte(s), %10.2f kbps (%u bytes em %3u.%06u s)"`
fim_cliente.

Servidor:
processa argumentos da linha de comando:
`porto_servidor tam_buffer tam_janela`
faz abertura passiva e aguarda conexão
recebe o string com nome do arquivo
abre arquivo que vai ser lido – pode ser `fopen(nome,"r")`
se deu erro, fecha conexão e termina
loop lê o arquivo, um buffer por vez até `fread` retornar zero
envia o buffer lido
se quiser, contabiliza bytes enviados
fim_loop
fecha arquivo
chama `gettimeofday` para tempo final e calcula tempo gasto
se quiser, imprime nome arquivo e número de bytes enviados
fim_servidor.

De forma resumida, o cliente deve enviar uma string com o nome do arquivo desejado, receber o arquivo em um buffer de cada vez e salvar os dados no disco à medida que eles chegam. Quando não houver mais bytes para ler, o cliente fecha o arquivo e gera uma linha com os dados da execução. O servidor por sua vez deve operar de forma complementar.
Como neste trabalho será usado UDP, pacotes podem ser perdidos ou corrompidos. Sua implementação deve lidar com perdas e erros de bits fazendo a retransmissão dos pacotes. Para simplificar, você pode considerar uma temporização de um segundo – pode ficar bem lento, mas isso não será um problema. Finalmente, pode assumir que quando o cliente envia o nome do arquivo para o servidor, essa mensagem nunca é perdida ou corrompida.

Definição do Protocolo

Neste trabalho, como não teremos TCP para garantir a entrega confiável, vocês serão responsáveis por criar um protocolo confiável sobre UDP usando janela deslizante. Para isso, vocês devem definir quais serão os campos do cabeçalho do seu pacote (que será enviado dentro do pacote UDP), que encapsulará os dados do arquivo. Como o protocolo será do tipo janela deslizante, ele deve ser baseado no Go-Back-N, Repetição Seletiva ou alguma variação do TCP. Na minha opinião, implementações baseadas no Go-Back-N são as mais fáceis de desenvolver. No entanto, alguns alunos aproveitam este trabalho para desenvolver algo mais próximo do TCP.
O protocolo nesse caso será unidirecional, isto é, ele deve ser construído para enviar dados apenas na direção do servidor para o cliente. A primeira mensagem, do cliente para o servidor, não precisa ser mandada com uma janela deslizante. Lembre-se, essa primeira mensagem nunca é perdida ou corrompida. Mensagens de confirmação podem ser necessárias em diferentes momentos e uma técnica de detecção de erros também deverá ser utilizada, pois erros poderão ocorrer durante os testes.
Um grande desafio deste trabalho é o tratamento de temporizações. Isso pode ser feito de diferentes maneiras, usando uma temporização associada ao `recv` (`https://linux.die.net/man/7/socket`) ou usando alarmes e tratadores de sinais com sockets, conforme código de exemplo e apostila disponibilizados no Moodle. A primeira é mais simples, mas não é garantida em ambientes que não o Linux e o FreeBSD (Mac). Se você optar por uma implementação baseada no Go-Back-N ou uma variação do TCP, você precisará utilizar apenas um único temporizador, assim como no TP 2.

Dica

Se você tiver um tratador de sinais associado a um timer e o timer for acionado enquanto o programa está parado no recv, a chamada retorna com um sinal de erro e errno igual a EINTR.

Código de Auxílio

No Moodle encontra-se um conjunto de arquivos auxiliares para este trabalho. Esse conjunto contém:

- um programa que utiliza sinais e temporização no seu funcionamento (impaciente.c) e que serve de exemplo no uso dessas funções;
- um módulo de encapsulamento das funções da interface de sockets (`tp_socket.h` e `tp_socket.c`) **que deve ser usado no trabalho**.

Esse módulo de encapsulamento deve ser usado **obrigatoriamente** no trabalho, no lugar das chamadas diretas às funções da biblioteca de sockets. Na avaliação, seu programa (que deverá usar essas funções) será ligado a uma versão desse módulo que servirá para simular perdas e erros no canal de transmissão em certas situações.
OBSERVAÇÃO: Programas que sejam desenvolvidos usando as funções da biblioteca sockets diretamente ao invés do módulo de encapsulamento fornecido NÃO serão avaliados.

Medições de Desempenho

Uma vez que os programas estejam funcionando corretamente, deve-se desenvolver uma avaliação do desempenho do par de programas semelhante ao que foi feito no TP2. Deve-se medir a vazão da comunicação, isto é, a taxa de transferência obtida entre cliente e servidor (basicamente, o número total de bytes enviados dividido pelo tempo medido no cliente).
As medições devem verificar como o desempenho varia quando diferentes tamanhos de buffer e de janela são utilizados. Em particular, deve-se incluir nas medições os casos com mensagens de tamanho 100, 1.000 e 4.000 bytes. Outros valores podem ser escolhidos conforme suas observações indicarem a necessidade. Já o tamanho da janela deve ser variado em potências de 2, iniciando em 2 e até que o desempenho atinja o máximo para aquele tamanho de mensagem. (Cabe a vocês determinarem quando isso ocorre experimentalmente).
O tamanho do arquivo pode ser escolhido de forma a garantir um tempo de teste nem muito longo nem muito curto. Para testes em que o par de programas executa na mesma máquina, um arquivo de aproximadamente 3 MB pode ser um bom ponto de partida.

O Relatório

Junto com os programas desenvolvidos deve ser entregue um relatório (PDF) contendo as seguintes seções:

1. Introdução: descrição do objetivo do trabalho.
2. Implementação: cabeçalho do protocolo, detalhes da implementação do protocolo (Go-Back-N, Repetição Seletiva ou alguma variação do TCP), decisões de projeto envolvidas, como a técnica utilizada para fazer o enquadramento do arquivo (indicar ao receptor quando o arquivo acaba) e o tratamento de temporizações.

3. Metodologia: dados sobre os experimentos, como a configuração das máquinas utilizadas, a localização das mesmas na rede. Indique também como foram feitas as medições (`gettimeofday`, imagino), quantas vezes o teste foi executado, se foram execuções diferentes do programa ou apenas um loop ao redor do programa todo para fazer tudo um certo número de vezes.
4. Resultados: apresente a informação coletada, tanto na forma de tabelas quanto na forma de gráficos.

   Cada tabela deve conter, para o experimento em questão, uma linha para cada tamanho de mensagem, com o número de mensagens enviadas, o tempo total médio medido, o desvio padrão dos tempos medidos e a vazão média observada no experimento.

   Cada gráfico deve usar escalas adequadas (logarítmicas ou lineares, conforme o caso), os eixos devem ser identificados e possuir claramente a identificação das unidades em cada um. Os resultados devem ser apresentados por linhas retas interligando os pontos medidos, que devem ser destacados com marcas claras. Se um gráfico contiver mais que um conjunto de pontos, as linhas devem ser claramente identificadas. Se possível, cada gráfico deve incluir barras verticais indicando a variância de cada valor calculado.
5. Análise: para cada experimento, discuta os resultados observados. Os resultados foram de acordo com o esperado? Você é capaz de explicar por que as curvas se comportam como o fazem? Houve algum elemento claramente de destaque nos resultados que merece uma análise especial (por exemplo, um pico/vale inesperado nos gráficos, desvios muito significativos nas medições)?
6. Conclusão: como todo trabalho técnico, algumas palavras finais sobre o resultado do trabalho, tanto das observações quanto do seu aprendizado sobre o assunto.

O relatório não precisa incluir a listagem dos programas.

Submissão Eletrônica

A entrega eletrônica deve ser feita pelo Moodle e deve constar de um arquivo do tipo .zip contendo os seguintes documentos:

- todos os arquivos fonte utilizados para construir os programas (arquivos de terminação .c, .h, makefile);
- o programa cliente chamará: clienteFTP
- o programa servidor chamará: servidorFTP
- uma cópia eletrônica do relatório;
- se possível, arquivos contendo os dados coletados, seja como planilhas ou como arquivos texto contendo os dados na forma gerada pelos programas.

Não inclua nesse conjunto os arquivos objeto (.o) nem os executáveis utilizados!

Observações Gerais

- O trabalho pode ser feito em grupos de até 3 pessoas. No entanto, todos devem realizar a submissão de maneira individual pelo Moodle.
- Se você não fizer um makefile, inclua instruções de como compilar o seu projeto.
- **Programas cuja saída não sigam o formato descrito acima, ou que exijam dados de entrada em formato diferente do descrito, ou que sejam submetidos de forma incorreta (especialmente no que diz respeito à submissão eletrônica) serão penalizados no processo de avaliação.**
- **Dúvidas:** não hesitem em escrever para o professor e o monitor no fórum de discussão do Moodle. Tentaremos responder toda pergunta em menos de 48 horas.
- Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar.
- **Vão valer pontos clareza, indentação e comentários no programa.**