Redes de Computadores

Trabalho 1
Rede de Compartilhamento de Arquivos - RCA

Profa. Hana Karina S. Rubinsztejn

Descrição:

Este trabalho tem por objetivo a implementação da funcionalidade básica de um sistema de troca de arquivos entre pares, frequentemente denominado *peer-to-peer*, onde os programas de todos os usuários da rede podem agir simultaneamente como cliente e servidor.

Implemente um programa para representar uma rede de compartilhamento de arquivos (RCA), que considera as seguintes regras de negócios.

Todo programa quando executado deve funcionar como cliente e servidor ao mesmo tempo.

Um programa, ao ser iniciado, deverá identificar os arquivos presentes no seu diretório corrente e se preparar para compartilhá-los. Ao mesmo tempo, ele deve exibir opções ao usuário para listar os usuários e os arquivos existentes na RCA, para procurar um arquivo e para buscar um arquivo . Se o arquivo já existir no diretório local, o programa deve apenas escrever uma mensagem com essa informação. Se o programa não souber onde o arquivo está, deverá perguntar aos demais programas se eles têm o arquivo procurado. Essa pergunta deve ser repassada a todos os programas da rede. Caso o programa de um usuário tenha um arquivo com aquele nome, ele responde a pergunta com o tamanho do arquivo. Ao conhecer o local onde está o arquivo (resposta à procura, ou informado pelo usuário), o programa do usuário então deve buscar o arquivo através de uma conexão TCP (confiável).

As principais funcionalidades enquanto **servidor** são:

1. Armazenar o log das solicitações que foram feitas com: IP, tipo de solicitação, data e hora (não é necessário mostrar na tela, apenas armazene);
2. Responder consulta sobre seu IP (“usuário” de rede)
3. Fornecer a lista de arquivos contidos em seu diretório;
4. Responder positivamente caso possua determinado arquivo;
5. Enviar um determinado arquivo solicitado por outro usuário, cronometrar o tempo gasto para a transferência e o salvar no log.

Enquanto **cliente** as principais funcionalidades são:

1. Solicitar (via BROADCAST[1]) a lista de usuários da RCA. Devem ser retornados o IP de cada usuário da rede;
2. Solicitar (via BROADCAST) a lista de arquivos de todos os programas executando na rede local;
3. Procurar (via BROADCAST) um arquivo específico na RCA.
4. Baixar um arquivo de um determinado servidor.

Todos os programas devem utilizar a mesma porta (ex: 5555). Quando um programa solicitar a lista de arquivos da RCA, todos os outros programas devem enviar a sua lista. Ela deve ser apresentada na linha de comando de quem solicitou da seguinte maneira: <IP> <nome do arquivo>.

Por exemplo:

129.168.1.1 agenda.pdf
129.168.1.1 relatorio.doc
129.168.1.2 verificacao.txt
129.168.1.2 imagem.jpg
129.168.1.2 lista.txt
129.168.1.3 notas.pdf

Não é permitido implementar um arquivo como cliente, e outro arquivo como servidor. O cliente e o servidor devem ser o mesmo programa.

Dicas:

1 - Vocês terão que definir um protocolo da troca de mensagens de controle usando UDP (funcionalidades 1, 2 e 3 do "cliente"). Recomendo que os cabeçalhos tenham um tamanho fixo e representado em binário (por exemplo: uma `struct`), por ser mais simples e eficiente do que usar strings para isso.

2 - A descrição apresentada indica claramente a existência de funcionalidades distintas associadas ao programa: (*i*) a interface com o usuário, que lê os comandos e exibe resultados, (*ii*) um módulo que inicia a consulta, espera pelas respostas, (*iii*) um que processa as mensagens de consultas recebidas, e (*iv*) outro que recebe as conexões TCP e cuida da transferências dos arquivos.
Para implementar essas funcionalidades pode-se utilizar diferentes processos.

[1] Necessário indicar que o socket utiliza *broadcast*: use a função `setsockopt (sock, SOL_SOCKET, SO_BROADCAST, &setar,sizeof(int))`. No envio indicar na estrutura de endereço, o IP 255.255.255.255 ou `htonl(-1);`

3 - Por conta desse isolamento de funcionalidades, um programa deve ser capaz de continuar atendendo consultas mesmo enquanto recebe ou envia um arquivo. Além disso, ele deve ser capaz de atender diversos pedidos de transferência em paralelo (os tutoriais de sockets, por exemplo, o do Beej e do Frost - mostram servidores concorrentes usando fork/threads, e servidores monoprocessados usando *select*).

4- A troca de arquivos deve ser via conexão TCP. Diferentes tipos de arquivos podem ser encaminhados. Teste se imagens, arquivos texto, pdf, são transferidos corretamente.

Tratamento de temporizações

O tratamento de temporizações é uma questão importante de projeto e deve ser bem documentada no relatório.

No caso do RCA o problema de temporizações é bastante simplificado em relação a outros protocolos, pois há poucos casos onde pode haver temporizações, em especial para evitar o bloqueio de um programa devido à espera de recebimento de uma mensagem.

O tratamento de temporizações pode ser feito de diferentes maneiras: com o uso da função `select`; com uso de temporização associada às funções de recebimento; ou usando alarmes de tratadores de sinais do SO (`SIGALRM` e com a função `setitimer`).

A segunda é mais simples, mas não é garantida em ambientes que não o Linux e o FreeBSD. Ela é feita através do uso da opção `SO_RCVTIMEO` da função `setsockopt`[2] que permite que se defina uma temporização diretamente relacionada ao socket, de forma que uma operação `recvfrom` temporiza sozinha, retornando com erro (-1) e setando um código de erro (`errno`) com `EWOULDBLOCK` ou `EAGAIN`[3].

Na página do curso há um exemplo de um programa que utiliza sinais e temporização no seu funcionamento (`impaciente.c`), que serve de exemplo no uso das funções.

---

[2] Exemplo:

```
setsockopt(sockid, SOL_SOCKET, SO_RCVTIMEO,(struct timeval *)&tv, sizeof(struct timeval));
```

[3] Para acessar os códigos de erro, inclua `errno.h` que é um arquivo cabeçalho da biblioteca padrão de C que fornece macros para identificar e relatar erros de execução através de códigos de erro. Os erros podem ser obtidos através da macro `errno` que fornece um número inteiro positivo contendo o último código de erro fornecido por alguma função ou biblioteca. Exemplo: compare se `errno == EWOULDBLOCK`.

Entrega do Trabalho:

O trabalho deverá ser submetido eletronicamente utilizando o programa **moodle**.

Cada grupo (de até 3 participantes) deve entregar um arquivo tipo zip ou tar.gz, contendo os arquivos fontes do trabalho (e **makefile**) e um relatório. Esse relatório deve conter o nome dos autores e descrever como a rede de compartilhamento de arquivos funciona; o protocolo implementado (formato de mensagens); uma seção explicando como executar os programas; uma seção descrevendo se tudo que foi especificado/solicitado no o trabalho foi ou não implementado; e, por fim, outra seção com as funcionalidades **extras**, se implementadas.

Os grupos irão demonstrar a execução do trabalho em data definida no moodle, durante o horário da aula em laboratório.

Observações:

a) Seu programa deve compilar no Linux utilizando gcc;
b) Também serão aceitos programas em Java. Neste caso deve-se implementar uma interface grafica para a execução do programa e suas fucionalidades.
c) Comente todas as linhas do seu código;
d) Na entrega não inclua os arquivos objeto (.o) nem os executáveis utilizados!