MC833 - Projeto 2
Cliente e Servidor UDP

Caio Henrique Pardal - RA: 195216

Introdução

O projeto tem por objetivo estabelecer uma comunicação UDP entre um cliente e um servidor iterativo. Além disso, será feita uma comparação entre o protocolo UDP e o TCP, por meio de análises de tamanho de código, confiabilidade e nível de abstração.

A aplicação desenvolvida consiste em um sistema de catálogo de filmes de um cinema. O servidor dessa aplicação armazena as seguintes informações dos filmes em cartaz: Título, Sinopse, Gênero, Salas de exibição do filme e um identificador único para cada filme. Este servidor deve ser capaz de receber as requisições do cliente, localizado em uma máquina diferente da que o servidor estará rodando, e transmitir todas as informações disponíveis para cada uma das possíveis requisições que o cliente pode realizar.

Essas possíveis requisições são: Cadastro de novos filmes, recebendo como resposta positiva o respectivo identificador, Remoção de filmes a partir de seus identificadores, Listar o título e salas de exibição de todos os filmes, Listar todos os títulos de filmes de um determinado gênero, Retornar o título do filme a partir de seu identificador, Retornar todas as informações de um filme a partir de seu identificador e Listar todas as informações de todos os filmes.

Sistema

1. Descrição Geral

Explicando um pouco sobre a arquitetura dos arquivos e do sistema, ambas as partes (cliente e servidor) estão separadas em diferentes pastas e arquivos diferentes. O código principal (contendo as implementações das funções principais e regras de negócio) são arquivos “.c” e existem arquivos “.h” para ambos, os quais definem as assinaturas de funções, incluem as bibliotecas necessárias e definem as constantes essenciais para o cliente e o servidor.

Em relação às constantes, deve-se ressaltar a importância de duas delas: “BUFFLEN” e “UDP_PORT”. A primeira é responsável por padronizar o tamanho da mensagem, fazendo com que haja um mapeamento 1:1 entre um envio do servidor com sua recepção no cliente, e vice-versa. A segunda define a porta em que deve-se estabelecer a conexão UDP entre os dois serviços. Ambas devem assumir os mesmos valores no servidor e no cliente. Outros componentes importantes nos headers são os wrappers para as funções

sendto e recvfrom. Os wrappers, além de capturar erros das syscalls, garantem que o tamanho da mensagem enviada/recebida seja sempre igual ao valor de “BUFFLEN”, garantindo o mapeamento 1:1 citado anteriormente.

Os arquivos “.c” do servidor e do cliente são responsáveis por coordenar a troca de mensagens. O cliente executa essencialmente três operações: enviar mensagem, receber mensagem e receber arquivo. O servidor, analogamente, executa três operações: enviar mensagem, receber mensagem e enviar arquivo. Troca de mensagens refere-se a enviar/receber uma única string contendo um comando ou dado, já troca de arquivos trata-se de uma série de trocas de mensagens contadas.

O fluxo de execução se assemelha ao do primeiro projeto, em que todo o fluxo de execução no servidor e no cliente são controlados por duas estruturas switch-case. A grande diferença é que enquanto o TCP do primeiro projeto é concorrente, o UDP é iterativo, o que implica que o mesmo trabalha com uma fila de *requests* de clientes. Outra diferença é no tratamento de erros do sistema. Enquanto no TCP uma falha de comunicação implicaria na parada do programa, no UDP os erros de comunicação são tolerados.

Dessa maneira, a consistência das informações recebidas não é uma garantia, porém o servidor UDP é mais flexível, visto que não é necessária uma conexão propriamente dita e a perda de mensagens ser tolerada. Com isso, há a implementação de um *timeout* no cliente, utilizando a função *setsockopt* que evita o bloqueio do cliente pela syscall.

A estrutura do armazenamento de dados segue o mesmo padrão descrito no primeiro projeto: os dados são estruturados como arquivos. No servidor, a pasta “data/” armazena arquivos textos (.txt) com os dados dos filmes. Esses arquivos são nomeados de acordo com o identificador dos filmes (que geralmente buscam seguir o título do filme, ao menos que este possua um espaço no seu nome), e este identificador é utilizado como chave na busca de arquivos. Existe ainda dois outros arquivos dentro dessa pasta, um deles é o arquivo “index.txt”, utilizado para armazenar todos os identificadores dos filmes, presentes naquele momento no servidor, e acessar os arquivos quando a opção selecionada não especifica um identificador. E o outro, é um arquivo chamado “help.txt” que contém todas as opções de requisições disponíveis que um cliente pode fazer e este arquivo é enviado para o cliente quando o mesmo digita “h”, no terminal. Como o cliente não armazena nenhum dado, mas apenas os exibe no terminal, este não possui uma pasta “data/” para armazenamento de informações.

Nota-se também que para que seja estabelecida a conexão entre cliente e servidor, deve-se rodar o servidor em uma máquina e o cliente em outra (pode ser feito em terminais na mesma máquina também) por meio dos comandos (após terem sido compilados os programas): **Servidor: ./<nome do programa> e Cliente: ./<nome_do_programa> <ip_do_cliente> ou localhost, caso esteja na mesma máquina.**

2. Casos de Uso

Os casos de uso para o cliente e o servidor são:

Comando	Ação
“h” ou “help”	Mostra a lista de requisições e comandos disponíveis
1(espaço)<nome_do_filme>(ponto e vírgula ;)<sinopse>(ponto e vírgula ;)<gênero>(ponto e vírgula ;)<salas de exibição separadas por traços(-)>	Cadastra um novo filme usando as informações passadas e recebe como resposta positiva o identificador do filme adicionado
2(espaço)<identificador>	Remove um filme a partir do identificador passado
3	Lista o título e salas de exibição de todos os filmes
4(espaço)<gênero>	Lista todos os títulos de filmes de um determinado gênero
5(espaço)<identificador>	Dado o identificador de um filme, retorna o título do filme
6(espaço)<identificador>	Dado o identificador de um filme, retorna todas as informações deste filme
7	Lista todas as informações de todos os filmes

3. Estrutura de dados e armazenamento

Os dados do servidor estão todos armazenados dentro do diretório “server/data”. A estrutura dos dados que são armazenados consiste em um arquivo “index.txt” para registrar os identificadores dos filmes cadastrados no sistema, arquivos “[identificador].txt” para armazenar as informações relativas ao filme em cartaz e um arquivo “help.txt” que armazena todas as requisições possíveis para o cliente fazer e o servidor processar e realizar.

Falando mais especificamente dos arquivos de cada um dos filmes, estes seguem uma estrutura própria: Na primeira linha do arquivo encontra-se o título do filme, na segunda

a sua sinopse, na terceira o seu gênero e, por fim, na quarta, as salas de exibição daquele filme. Vide representação a seguir:

Linha 1	Título do filme
Linha 2	Sinopse
Linha 3	Gênero
Linha 4	Salas de exibição

4. Detalhes de implementação

Implementação do servidor UDP

O servidor UDP foi implementado focando-se no armazenamento de dados de forma persistente e na iteratividade das requisições entre clientes. Esse servidor, por ser iterativo, consiste em um único processo associado à porta “UDP_PORT” com protocolo IPv4 (AF_INET). Após feita a configuração inicial das informações próprias do servidor e da vinculação de uma porta em que este será utilizado, foi alocada a struct *cliaddr*. Esta é utilizada para armazenar as informações dos clientes, possibilitando a resposta do servidor UDP.

Na perspectiva do cliente, os wrappers para mensagens UDP do cliente são mais tolerantes a erros garantindo que, ao invés de interromper a execução do cliente, o cliente possa retomar o fluxo de execução permitindo novas requests mesmo que uma troca de mensagens tenha sido inutilizada. Além disso, devido a possibilidade de perda de mensagens, foi definido um timeout utilizando a função *setsockopt* a fim de evitar que o cliente seja bloqueado pela syscall. Nota-se que, no caso de timeout, a troca de mensagens é interrompida e, consequentemente, as informações recebidas tornam-se incompletas e inconsistentes, mas tanto o cliente quanto o servidor continuam a execução.

Para fins de praticidade, supôs-se que todos os comandos fornecidos pelo cliente eram válidos dentro do escopo de operações do servidor (sem distinções de permissões de usuários). Como o protocolo UDP não garante a ordenação das mensagens que são enviadas, a maior parte dos casos de erros foram tolerados, de forma que o servidor continue rodando mesmo após uma perda de mensagens ou embaralhamento das informações. O servidor pode aceitar requisições de múltiplos clientes iterativamente, atendendo as requisições por ordem de chegada e sem misturá-las, visto que ele atende um cliente por vez. A finalidade das suposições são para evitar eventuais tratamento de erros, os quais trariam uma maior complexidade para o servidor e fugiria do objetivo proposto para o projeto.

Comparação TCP e UDP

Ao comparar os códigos de cada um dos servidores, percebe-se que o servidor UDP requer bem menos procedimentos do que o TCP para se estabelecer uma comunicação. Isso se deve, pois o UDP além de não usar *fork* para paralelizar o atendimento dos cliente, ele também não requer a necessidade de aceitar conexões usando o *accept*. O que torna esse servidor uma opção bem mais simples e rápida de ser implementada.

Entretanto, essa praticidade possui um *tradeoff*, uma vez que ao “pular” vários passos para estabelecer uma comunicação, a sua confiabilidade cai. A principal falha que se pode destacar, é a perda de mensagens, com possibilidade de embaralhamento de dados. O que pode ser verificado ao trocar mensagens utilizando do servidor implementado neste projeto.

Se compararmos o tempo de entrega das mensagens, perceberemos que muito provavelmente o tempo de entrega de servidores UDP serão menores, pois o TCP garante a ordem da entrega das mensagens e própria entrega destas, fazendo uso de ACKs, por exemplo. O que não ocorre no caso do UDP, por permitir essas falhas durante sua comunicação com clientes.

Conclusão

O projeto criou um cliente que se comunica com servidores UDP e um servidor UDP (iterativo) que se comunica com clientes em diferentes portas e IPs que se comunicam com o mesmo. As mensagens trocadas entre ambos dizem respeito à um catálogo de filme e descrevem o exposto na seção “Casos de uso”.

A partir do estabelecimento da conexão entre o cliente e servidor, pode-se observar como o protocolo UDP funciona na prática, notando os seus casos de uso, erros e tempo de execução, o que permite que se faça uma análise comparativa entre este protocolo e o protocolo TCP.

Ao fazer isso, percebe-se que o protocolo UDP transmite mensagens com muito mais rapidez do que o TCP, porém na medida que o faz com mais rapidez, perde confiabilidade na transmissão das mensagens, resultando em erros e falhas.

Dessa maneira, pode-se afirmar que o projeto atingiu o objetivo esperado, ao estabelecer uma conexão entre um servidor UDP e um cliente e possibilitar a troca de mensagens entre ambos.

Referências

Beej's Guide to Network Programming (http://beej.us/guide/bgnet/html/single/bgnet.html)