Relatório Protocolos TCP e UDP

Programação de Redes de Computadores
Instituto de Computação, Unicamp

Prof. Dr. Edmundo R. M. Madeira
Giovanne Lucas Dias Pereira Mariano 173317

15 de junho de 2022

Resumo

O projeto tem por objetivo exercitar os conceitos de programação de sockets, no contexto de redes de computadores, através da implementação de um sistema cliente-servidor que emula uma plataforma de streaming de vídeos.

Foi explorada a interface entre kernel e usuário, usando para protocolo de transporte o TCP e o UDP. O TCP se mostrou efetivo para a transmissão segura de mensagens, enquanto o UDP, ao rodar cliente e servidor na mesma máquina, demonstrou comportamento seguro também. Estes resultados observados vão de encontro com o previsto teoricamente e reforçam a relevância de se avaliar os protocolos frente às suas propostas. Ou seja, é necessário, para uma avaliação mais significativa, analisar a taxa de entrega de pacotes em rede distribuída e a velocidade desta entrega.

Além de explorar tais conceitos, o projeto permitiu a inspeção de diversos desafios da computação distribuída, como a consistência em dados armazenados como recurso compartilhado, a escalabilidade e a heterogeneidade. Para efeitos deste trabalho, estes desafios foram relevados, de maneira que o código gerado permite inconsistência em casos extremos, a escalabilidade foi limitada e supôs-se que tanto cliente quanto servidor rodarão em máquinas modernas com devida capacidade de armazenamento e processamento.

O repositório com o código desenvolvido para o projeto pode ser encontrado no github em lucsgiovanne/sockets.

1 Introdução

O projeto consiste em desenvolver, utilizando programação de sockets de redes de computadores, um sistema cliente-servidor simulando uma plataforma de streaming de vídeos.

O servidor armazena informações simples sobre os filmes e permite com que os usuários cadastrem novos filmes; alterem informações sobre filmes já cadastrados; removam um filme já cadastrado; e listem filmes e suas propriedades, podendo filtrar pelo identificador ou por um de seus gêneros.

Dessa forma, o projeto exercita a programação de sockets para comunicação cliente-servidor utilizando os protocolos TCP e UDP para transporte, através da construção de um sistema no qual os protocolos de transporte tem alto potencial para interferir na qualidade do serviço.

Neste contexto, é esperado que um modelo comparativo entre os dois protocolos, avaliando taxa e velocidade de entrega de pacotes, possa reiterar as projeções teóricas de que o TCP é mais adequado para transmissões que necessitam de garantia de entrega, enquanto o UDP é mais adequado para transmissões que priorizam uma maior velocidade de entrega, mesmo que não exista garantia de entrega.

2 Sistema

No caso do sistema implementado com protocolo TCP, a arquitetura cliente-servidor efetua transporte através de comunicação orientada a conexão. Ou seja, servidor escuta clientes que, uma vez conectados, podem efetuar as operações fornecidas pelo servidor.

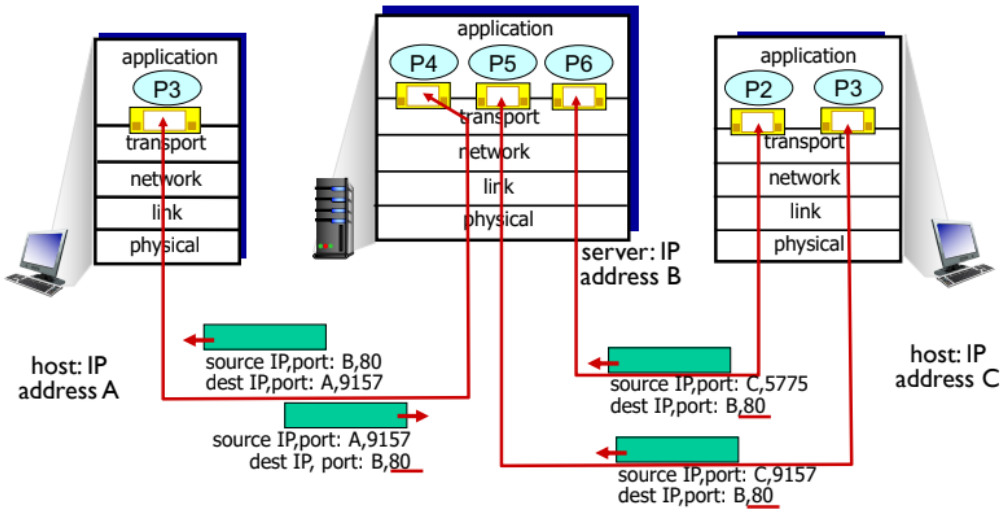

Figura 1: Ilustração da comunicação orientada a conexão em arquitetura cliente-servidor. Retirada do livro-texto [1].

Para o sistema implementado com protocolo UDP, por outro lado, a arquitetura cliente-servidor também foi utilizada, porém sem que a comunicação seja orientada a conexão. Desse modo, o servidor se comporta mais como um ouvinte e o cliente como um orador.

Uma visão de mais baixo nível do sistema pode ser representada expondo as chamadas de sistema que o servidor e o cliente utilizam durante uma comunicação típica.

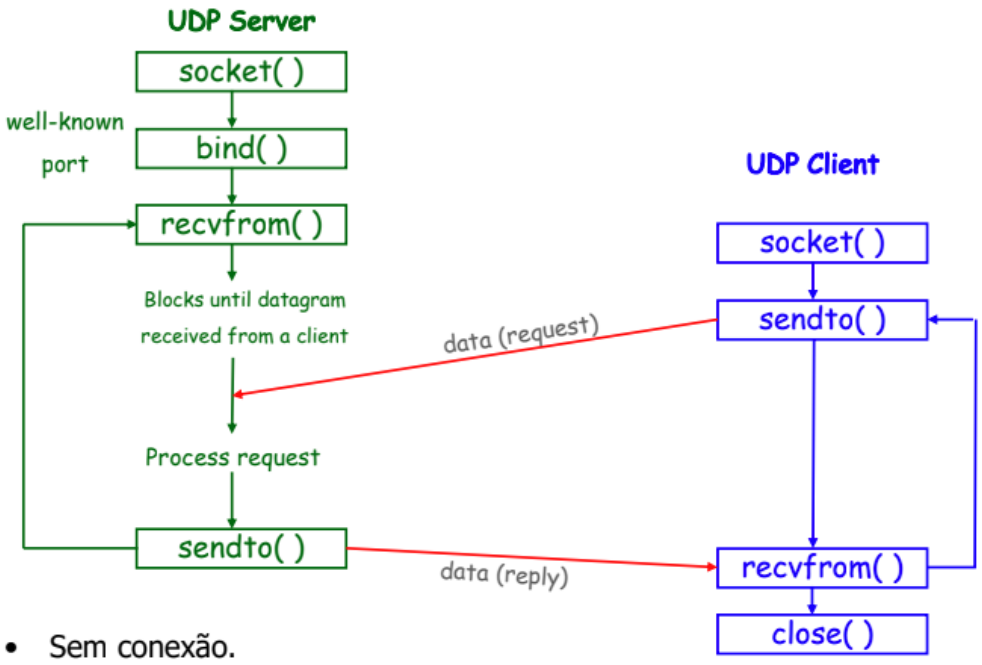

Figura 2: Diagrama de chamadas de sistema da comunicação UDP entre cliente-servidor sem conexão. Retirada do livro-texto [2].

Independentemente do protocolo utilizado na camada de transporte da rede, o servidor gerencia uma coleção de registros de filmes.

Cada registro contém:

- identificador numérico;
- título;
- gêneros;
- diretor(a);
- ano de lançamento.

As operações permitidas ao usuário são:

- cadastrar um novo filme, determinando um identificador numérico no cadastro;
- acrescentar um novo gênero em um filme;
- listar todos os títulos, junto a seus respectivos identificadores;
- listar informações (título, diretor(a) e ano) de todos os filmes de um determinado gênero;
- listar todas as informações de todos os filmes;
- listar todas as informações de um filme a partir de seu identificador;
- remover um filme a partir de seu identificador.

3 Armazenamento

Para gerenciar os registros dos filmes, foi implementado um banco de dados - apesar de não satisfazer os requisitos formais para ser classificado como um banco de dados, não possuindo um SGBD ou estrutura equivalente, neste documento o termo será utilizado como sinônimo por simplicidade - extremamente simples, com apenas as funcionalidades suficientes para suprir os casos de uso já detalhados.

O banco de dados armazena até 10 registros, de modo a suprir apenas o requisito de teste descrito no projeto. Porém, pode ser facilmente reconfigurado para outro tamanho máximo observando-se que a escalabilidade é bastante falha, uma vez que otimização não foi um requisito considerado.

Outro fator de atenção é que o banco de dados não é resiliente a concorrência, de modo que, apesar de o servidor aceitar requisições por múltiplos sockets simultaneamente, não foi implementado mecanismo de trava para operações nos dados, de maneira que requisições simultâneas podem gerar inconsistências.

Cada registro é um struct em C e a tabela é armazenada com persistência em um arquivo ".dat".

Note que as estruturas de armazenamento foram implementadas de maneira independente com relação ao protocolo de transporte, de modo que ambas as implementações do sistema cliente-servidor compartilham as estruturas de armazenamento.

4 Detalhes de Implementação

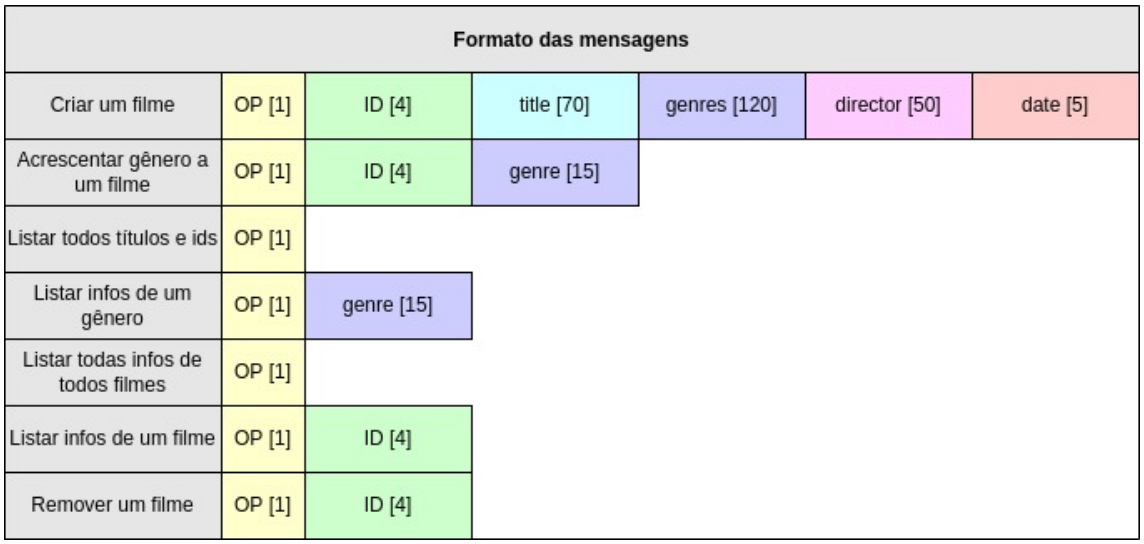

Figura 3: Ilustração do formato das mensagens para requisições ao servidor. O número de bytes relativos a cada argumento aparece entre "[ ]"no diagrama.

A comunicação via socket do servidor, tanto para o protocolo TCP quanto UDP, foi construído sobre a base fornecida no guia de programação de redes do Beej [3], de modo que foram necessárias apenas algumas modificações para melhorar a legibilidade do código e extendê-lo para que se comporte como um servidor que responde a requisições de cada uma das operações permitidas (operações inválidas recebem resposta indicando falha).

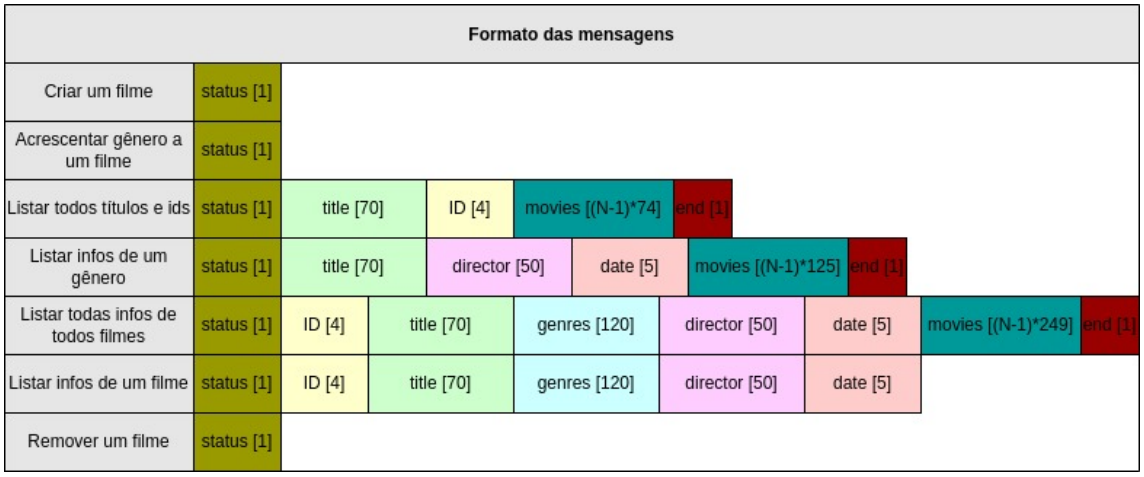

Figura 4: Ilustração do formato das mensagens para respostas aos clientes. O número de bytes relativos a cada argumento aparece entre "[ ]"no diagrama.

Foi escolhido o formato indicado na figura 3 para mensagens de requisição, que determina a operação e os os bytes relativos a cada argumento dependendo da operação selecionada.

Para as mensagens de resposta, foi escolhido o formato representado na figura 4.

Uma análise pós-implementação trás o sentimento de que uma melhor prática seria utilizar serialização dos campos, como bem argumentado ainda no guia de programação de redes do Beej.

As operações têm valores em ordem crescente, de cima para baixo na ilustração, indo de 1 a 7.

Nas respostas, status 0 indica que a operação foi bem sucedida e o conteúdo pode ser utilizado nos casos em que há dados retornados. O status 1 indica que houve algum erro e a operação foi abortada, de modo que o conteúdo da mensagem deve ser ignorado.

Além do banco de dados, foi implementado um backend que processa server-side as requisições, conectando com o banco de dados e gerando as respostas que serão consumidas pelo cliente.

O cliente, por sua vez, além de tratar de suas responsabilidades com relação à comunicação TCP, utiliza um frontend para permitir ao usuário escolher operações, preencher argumentos - foi utilizada uma função adaptada do guia [4] - e renderizar as respostas recebidas do servidor.

O backend e o frontend foram implementados de maneira modular, de forma que o sistema utiliza essas mesmas estruturas independente do protocolo (TCP ou UDP) escolhido.

Para compilar os programas gerados, basta navegar até o diretório relativo ao protocolo desejado utilizar o gcc para compilar. Por exemplo:

```
cd client-server/udp
gcc -o server.out server.c
gcc -o client.out client.c
```

Com os programas compilados, o servidor deve ser inicializado primeiro em um terminal exclusivo e, então, o cliente pode rodar em outro terminal, atentando-se para fornecer corretamente o endereço do servidor - caso rodando servidor e cliente

na mesma máquina, o endereço provavelmente será 127.0.0.1 (::ffff:127.0.0.1) - ao rodar o executável.

Ou seja, em um terminal deve-se iniciar o servidor:

```
./server.out
```

E, em outro, deve-se rodar o cliente:

```
./client.out ADDRESS
```

A porta através da qual cliente e servidor se comunicam é definida em código, de modo que fornecer apenas o endereço é necessário na hora da inicialização do programa. Outra importante observação é que os programas utilizando UDP aceitam apenas endereços de IP em formato IPv6.

5 Conclusão

Implementar um aplicativo cliente-servidor com programação de sockets como ensinado pelo Beej em seu guia é um ótimo exercício para escancarar quais são os principais desafios no contexto de redes de computadores.

Além dos desafios no contexto de redes de computadores, também se evidenciaram desafios da computação distribuída e de gerenciadores de banco de dados. As principais provocações foram a dificuldade em produzir sistemas tolerantes a falha, "imunes"a inconsistência e satisfatoriamente escaláveis no contexto de aplicações simples.

A concorrência em cima de um servidor foi o principal fator de dificuldades, uma vez que implica na necessidade de tratamentos robustos para manutenção de consistência, e soluções dinâmicas e otimizadas para armazenamento e consulta de dados.

Além das diferenças de implementação, não foram possíveis maiores conclusões comparativas com relação à utilização dos procolos TCP e UDP. Ou seja, foi confirmado o entendimento de que para analisar efetivamente as diferenças entre os protocolos, é necessário avaliar a velocidade e as taxas de entrega de pacotes.

Neste contexto, é possível inferir que o projeto foi bem sucedido não apenas como exercício dos conceitos e tecnologias de redes de computadores, mas também como inspiração para reflexões em várias áreas da computação que se intersectam com redes no contexto de aplicações cliente-servidor.

Referências

[1] KUROSE, J. F.; ROSS, K. W. Redes de computadores e a internet. *São Paulo: Person*, v. 28, 2006.

[2] FALL, K. R.; STEVENS, W. R. *TCP/IP illustrated, volume 1: The protocols*. [S.l.]: addison-Wesley, 2011.

[3] HALL, B. *Beej's Guide to Network Programming*. 2020. Disponível em: <https://beej.us/guide/bgnet/html/>.

[4] EGAN, D. *Integer Input in C*. 2019. Disponível em: <https://dev-notes.eu/2019/05/Integer-Input-in-C/>.