Relatório T1 Redes

Bruno Aquiles de Lima
Luan Marko Kujavski

1 Introdução

O trabalho consiste em implementar uma plataforma de "streaming" de vídeo, onde o cliente pode fazer requisições para listar os vídeos disponíveis e para assisti-los. O link para o repositório do projeto é `https://github.com/LoanMaikon/redes/tree/main/trabalho1`

2 Funcionamento dos programas

Após rodar `make`, são gerados os executáveis `server` e `client`. O uso dos programas é da seguinte forma: `sudo ./server <interface>`. Para o `client`, o uso é exatamente o mesmo.

2.1 Funcionamento do Client

Ao rodar o `client`, são apresentadas as seguintes opções:

1- `Mostrar vídeos` - *Lista os vídeos disponíveis no servidor e seus respectivos IDs*
2- `Baixar arquivo` - *Baixa um arquivo do servidor com base no ID do vídeo*
3- `Sair` - *Encerra a execução do programa*

2.2 Funcionamento do Server

- **Inicialização:** O servidor lista os vídeos no diretório `movies/` e armazena os nomes em uma estrutura `movies_t`, que contém um vetor de strings e um `unsigned long int` para o tamanho do vetor.
- **Loop Principal:** O servidor entra em um loop aguardando requisições de listagem e download. Requisições de outros tipos são ignoradas.
- **Requisição de Listagem:** Quando uma requisição de listagem é recebida, os nomes dos vídeos são enviados ao cliente na ordem definida pela estrutura `movies_t`.

3 Decisões de implementação

3.1 Timeout

No socket, foi definido o tempo de espera da função `recv` para 300 ms, assim o processo não fica travado esperando um pacote. Assim, foi possível a construção do timeout usando a função `time()`.

3.2 Implementação janela deslizante

Cliente:

- Mantém um vetor server_packets de 5 pacotes (buffers de 67 bytes). Recebe até 5 pacotes, valida a sequência e corrige pacotes fora de ordem, mantendo uma variável para controlar a sequência correta dos pacotes.

Servidor:

- Utiliza uma lista ligada window_packet_t para armazenar pacotes. Cada nodo contém um pacote e um ponteiro para o próximo nodo, e o cabeçalho window_packet_head_t contém ponteiros para o primeiro e o último nodo e um unsigned long int para o tamanho da lista.

- O servidor lê arquivos em blocos de 63 KB, quebra-os em pacotes, e os organiza na lista ligada. Se a quantidade de pacotes restantes for menor que o tamanho da janela, os pacotes antigos são desalocados, e novos pacotes são lidos e concatenados no final da lista. O primeiro nodo da nova leitura tem uma sequência incrementada em relação ao último pacote do bloco anterior.

- A lista ligada facilita a remoção de pacotes antigos e a adição de novos pacotes, mantendo a janela de pacotes sempre preenchida.

3.3 Tamanho dos pacotes

O tamanho de todos os pacotes ficou fixo em 67 bytes. Incialmente, foi implementado com tamanho dinâmico, porém, tornou o código mais complexo e não compensava a economia, pois apenas o último pacote de cada bloco de 63 KB teria tamanho diferente.

3.4 Escape para bytes problemáticos

Durante a construção dos pacotes de dados, para todo byte 0x81 e 0x88, é adicionado em seguida um byte 0xFF para evitar problemas de interpretação. Não é feito esse tratamento no momento do envio dos pacotes com os nomes dos vídeos, pois o intervalo dos bytes dos caracteres é 0x20 até 0x7E.