sistema de Backup de arquivos usando RAWSockets.

Heric Camargo

GRR20203959

Relatório Do Projeto: sistema de Backup de arquivos usando RAWSockets.

Estrutura

Pacote

- `start_marker`: 8 bits que indicam o início de uma mensagem.
- `size_seq_type`: 6+5+5 bits que indicam o tamanho, o número de sequência e o tipo.
- `data`: 0-63 bytes que contém os dados da mensagem.
- `crc`: 8 bits que contém o CRC da mensagem.
- `padding`: até a mensagem ter 64 bytes.

Status da Transferência

essa estrutura é usada nas funções de debug, mas, acabei usando alguns campos para o controle da lógica do programa. - `total_expected` - `total_received` - `expected_seq`: dores de cabeça aqui :(- `wrap_count` : e aqui - `...`

Fluxo

Backup

1. Início da transferência: O cliente envia um pacote BACKUP para o servidor.
   - **Client -> Server**: BACKUP packet (seq=0, len=18)
   - **Server -> Client**: ACK packet (seq=0)
   - **Server -> Client**: OK_SIZE packet (seq=0)
2. Transferência dos dados: O cliente envia pacotes DATA para o servidor.
   - **Client -> Server**: DATA packet (seq=N, len=63/40)
   - **Server -> Client**: OK packet (seq=N)
3. Fim da transferência: O cliente envia um pacote END para o servidor.
   - **Client -> Server**: END_TX packet
   - **Server -> Client**: OK_CHSUM packet

Restaura

1. Início da transferência: O cliente envia um pacote RESTORE para o servidor.
    - **Client -> Server**: RESTORE packet (seq=0, len=18)
    - **Server -> Client**: ACK packet (seq=0)
    - **Server -> Client**: OK_SIZE packet (seq=0)
2. Transferência dos dados: O cliente envia pacotes DATA para o servidor.
    - **Client -> Server**: DATA packet (seq=N, len=63/40)
    - **Server -> Client**: OK packet (seq=N)
3. Fim da transferência: O cliente envia um pacote END para o servidor.
    - **Client -> Server**: END_TX packet
    - **Server -> Client**: OK_CHSUM packet #### Verifica
4. Início da transferência: O cliente envia um pacote VERIFY para o servidor.
    - **Client -> Server**: VERIFY packet (seq=0, len=18)
    - **Server -> Client**: ACK packet (seq=0)
    - **Server -> Client**: OK_SIZE packet (seq=0) 1.1. Se o arquivo existe, o servidor envia um pacote PKT_ACK para o cliente.
    - **Server -> Client**: PKT_ACK packet (seq=0, len=18) 1.2. Se o arquivo não existe, o servidor envia um pacote PKT_ERROR para o cliente.
    - **Server -> Client**: PKT_ERROR packet (seq=0, len=18)