Trabalho com UDP.

Implementação de Transferência de Arquivos Confiável sobre UDP com Sockets

O Protocolo UDP

Este projeto foca na implementação de uma aplicação que opera sobre o protocolo UDP, utilizando programação com sockets. O principal objetivo é o desenvolvimento de um Servidor UDP simplificado, para consolidar o conhecimento sobre o funcionamento básico e a programação com UDP, contrastando-o com os serviços que o TCP disponibiliza para a camada de aplicação.

Objetivo do Projeto:

Desenvolver uma aplicação cliente-servidor para transferência de arquivos utilizando o protocolo UDP. O foco principal é a implementação de mecanismos básicos de controle e confiabilidade diretamente sobre UDP, simulando funcionalidades que o TCP oferece nativamente.

Requisitos Gerais:

1. Linguagem de Programação: A escolha da linguagem é livre (ex: Python, Java, C/C++, etc.).
2. Restrição de Bibliotecas: Não é permitido o uso de bibliotecas de alto nível que abstraiam a manipulação direta do protocolo UDP. A implementação deve utilizar a API de sockets do sistema operacional (ou da linguagem escolhida) para criar, enviar e receber datagramas UDP.
3. (Opcional/Sugestão) Recomenda-se iniciar com um exemplo simples "Hello World" cliente/servidor UDP para familiarização com a API de sockets antes de abordar a transferência de arquivos.

Requisitos do Servidor UDP:

1. **Inicialização**: O servidor deve ser executado antes do cliente.
2. **Porta**: Deve operar em uma porta UDP especificada, com número maior que 1024 (portas abaixo de 1024 geralmente exigem privilégios de administrador).
3. **Recepção e Protocolo**:
   a. Aguardar conexões/mensagens de clientes.
   b. Interpretar as requisições recebidas. É necessário definir e implementar um protocolo de aplicação simples sobre UDP para que o cliente requisite arquivos (Exemplo de formato de requisição: `GET /nome_do_arquivo.ext`).
4. **Processamento da Requisição:**
   a. Verificar se o arquivo solicitado existe.
   b. Se o arquivo não existir: Enviar uma mensagem de erro claramente definida pelo seu protocolo para o cliente.
5. **Transmissão do Arquivo (se existir)**:
   a. O arquivo a ser transmitido deve ser relativamente grande (ex: > 1 MB) para justificar a segmentação.
   b. Segmentação: Dividir o arquivo em múltiplos segmentos/pedaços para envio em datagramas UDP.
   c. Cabeçalho Customizado: Cada segmento enviado deve conter informações de controle definidas pelo seu protocolo (ver "Considerações de Protocolo" abaixo).
   d. Retransmissão: Implementar lógica para reenviar segmentos específicos caso o cliente solicite (devido a perdas ou erros).

Requisitos do Cliente UDP:

1. **Inicialização**: O cliente deve ser executado após o servidor estar ativo.
2. **Conexão**: Permitir que o usuário especifique o endereço IP e a porta do servidor UDP ao qual deseja se conectar.
3. **Requisição**:
   a. Enviar uma requisição ao servidor, utilizando o protocolo de aplicação definido, para solicitar um arquivo específico (Exemplo de entrada do usuário: @IP_Servidor:Porta_Servidor/nome_do_arquivo.ext).
4. **Simulação de Perda**:
   a. **Implementar uma opção** (ex: via entrada do usuário ou configuração) que permita ao cliente descartar intencionalmente alguns segmentos recebidos do servidor. Isso é crucial para testar o mecanismo de recuperação de dados. A interface deve informar quais segmentos (ex: por número de sequência) estão sendo descartados.
5. Recepção e Montagem:
   a. Receber os segmentos do arquivo enviados pelo servidor.
   b. Armazenar e ordenar os segmentos recebidos corretamente.
   c. Verificar a integridade de cada segmento (ex: usando checksum ou resumos criptográficos como o MD5 e SHA.).
6. Verificação e Finalização:
   a. Após receber todos os segmentos esperados (ou um sinal de fim de transmissão do servidor), verificar a integridade e completude do arquivo.
   b. **Se o arquivo estiver OK**: Salvar o arquivo reconstruído localmente e informar o sucesso ao usuário. Opcionalmente, apresentar/abrir o arquivo.
   c. Se o arquivo estiver com erro ou incompleto:
      I. Identificar quais segmentos estão faltando ou corrompidos.
      II. Solicitar a retransmissão desses segmentos específicos ao servidor, utilizando o protocolo definido.
      III. Repetir o processo de recepção e verificação até que o arquivo esteja completo e correto.
      IV. Interpretação de Erros: Interpretar e exibir mensagens de erro recebidas do servidor (ex: "Arquivo não encontrado").

Considerações Obrigatórias para o Design do Protocolo:

O aluno **deve projetar e justificar** as escolhas para os seguintes aspectos do protocolo de aplicação sobre UDP:

1. Segmentação e Tamanho do Buffer:
   a. Como o arquivo será dividido? Qual o tamanho máximo de dados por datagrama UDP (payload)?
   b. Este tamanho deve ser fixo ou variável? Como ele se relaciona com o MTU (Maximum Transmission Unit) da rede? (Pesquisar sobre MTU e fragmentação IP).
   c. Os buffers de envio (servidor) e recepção (cliente) precisam ter tamanhos relacionados?
2. Detecção de Erros:
   a. Como a integridade dos dados em cada segmento será verificada?
   b. É necessário implementar um checksum?
   c. Qual algoritmo usar? (Ex: CRC32, soma simples).
3. Ordenação e Detecção de Perda:
   a. Como o cliente saberá a ordem correta dos segmentos? É necessário um número de sequência?
   b. Como o cliente detectará que um segmento foi perdido (e não apenas atrasado)?
4. Controle de Fluxo/Janela (Opcional Avançado):
   a. Considerar como evitar que o servidor envie dados mais rápido do que o cliente consegue processar (embora um controle de fluxo completo seja complexo e possa estar fora do escopo inicial).
5. Mensagens de Controle:
   a. Definir claramente os formatos das mensagens de: requisição de arquivo, envio de segmento de dados (incluindo cabeçalhos com número de sequência, checksum, etc.), confirmação de recebimento (se houver), solicitação de retransmissão e mensagens de erro.

Lembre-se: O vídeo deve ser uma demonstração prática, complementada por explicações claras sobre o funcionamento e as decisões de projeto.

O vídeo deve ser anexado como link nos comentários da entrega dentro da atividade do classroom. **NÃO** deve ser enviado o arquivo!

Roteiro para Vídeo de Apresentação: Transferência Confiável de Arquivos sobre UDP
O vídeo de apresentação deve demonstrar o funcionamento da aplicação cliente-servidor desenvolvida, destacando a implementação dos mecanismos de controle diretamente sobre UDP utilizando a API de Sockets.

Itens Obrigatórios a Demonstrar e Explicar:

1. Ambiente e Setup:
   a. Mostrar a execução do Servidor UDP.
   b. Mostrar a execução do(s) Cliente(s) UDP.
   c. Demonstrar como o cliente especifica/obtém o endereço IP e a porta do servidor.
   d. Mostrar que a implementação utiliza a API de Sockets diretamente, sem bibliotecas de abstração UDP.
2. Protocolo de Aplicação e Requisição:
   a. Demonstrar o cliente requisitando arquivos ao servidor usando o protocolo de aplicação proposto (Ex: GET /nome_arquivo).
   b. Mostrar o cliente requisitando com sucesso arquivos diferentes (pelo menos dois).
3. Transferência e Segmentação:
   a. Demonstrar a transferência de um arquivo grande (ex: > 10 MB).
   b. Explicar e demonstrar como o arquivo é dividido em segmentos (chunks) para transmissão via UDP.
      I. Informar qual o tamanho do buffer/segmento de dados utilizado.
      II. Explicar brevemente a influência do tamanho do segmento e sua relação com o MTU (Maximum Transmission Unit).
4. Mecanismos de Confiabilidade (Demonstrar e Explicar):
   a. Ordenação: Mostrar como os números de sequência são usados para ordenar os segmentos no cliente e detectar perdas.
   b. Integridade:
      I. Demonstrar o uso de checksum ou hash (MD5/SHA) para verificar a integridade dos segmentos recebidos. Indicar qual método foi usado.
      II. Mostrar o processo de montagem e conferência do arquivo completo no cliente após o recebimento.
   c. Recuperação de Erros/Perdas:
      I. Simular a perda de pacotes: Demonstrar um mecanismo (idealmente no cliente ou simulado) que cause a perda de segmentos durante a transmissão. (Nota: O requisito original mencionava o servidor descartando, o que é incomum. É mais didático simular a perda na rede ou no cliente para testar a recuperação).
      II. Mostrar o cliente detectando a falta de segmentos (por timeout, NACKs, ou análise de sequência).
      III. Demonstrar o cliente solicitando a retransmissão dos segmentos faltantes/corrompidos ao servidor.
      IV. Explicar o mecanismo usado para disparo da retransmissão (Ex: timer de timeout, ACKs/NACKs).
5. Tratamento de Erros do Servidor:
   a. Demonstrar o cenário em que o cliente requisita um arquivo inexistente.
   b. Mostrar o servidor enviando uma mensagem de erro definida no protocolo.
   c. Mostrar o cliente recebendo e apresentando o erro de forma clara (Ex: "Erro: Arquivo não encontrado").
   d. Mencionar/Demonstrar outros erros tratados, se houver.
6. Cenários de Teste Específicos:
   a. Demonstrar o servidor lidando com dois clientes simultâneos (ou em rápida sucessão) requisitando arquivos diferentes.
   b. Mostrar o que acontece se o cliente tentar conectar antes do servidor ser iniciado (demonstrar o erro resultante).
   c. Mostrar o que acontece se o servidor for interrompido durante uma transferência (demonstrar o comportamento do cliente - ex: timeout, erro).
7. Pontos Chave para Enfatizar na Explicação:
   a. A natureza não confiável do UDP e como os mecanismos implementados (sequência, checksum, retransmissão) buscam mitigar isso.
   b. As escolhas de design feitas no protocolo de aplicação.
   c. As limitações da solução implementada.

- O aluno deve usar Socket UDP e não pode usar bibliotecas que mascarem o trabalho.

- Para agilizar a verificação de integridade são utilizadas somas de verificação (checksums) ou resumos criptográficos como o MD5 e SHA.