

Relatório da Implementação (Grupo: Rafael Lacerda, Otávio Alves, Miguel Davi)

Descrição Detalhada

Implementar uma comunicação confiável usando o protocolo UDP entre dois programas:

1. Transmissor: envia mensagens via UDP para um receptor.
2. Receptor: recebe os pacotes, verifica a ordem, envia ACKs e descarta duplicatas.

O sistema simula o comportamento de stop-and-wait protocol com verificação de sequência e ACK explícito, garantindo entrega ordenada e sem duplicação.

1. Estrutura do Protocolo de Aplicação

Para adicionar confiabilidade, foi definido um protocolo de aplicação simples com duas estruturas de dados principais: Packet para os dados e AckPacket para as confirmações.

- **struct Packet:**
 - int seqNum: Um número de sequência. No nosso caso (pare-e-espere), ele alterna entre 0 e 1. Ele é essencial para ordenar as mensagens e identificar duplicatas.
 - char data[1024]: O conteúdo da mensagem (payload) que o usuário digita.
- **struct AckPacket:**
 - int ackNum: O número de sequência do pacote que está sendo confirmado. Isso permite que o transmissor saiba exatamente qual pacote foi recebido com sucesso.

2. Implementação do Transmissor (Processo 1)

O transmissor é responsável por ler a entrada do usuário, empacotá-la e garantir que ela chegue ao receptor.

- **Inicialização:** Cria um socket UDP e configura o endereço do servidor (127.0.0.1 na porta 8080).
- **Mecanismo de Timeout:** A função setsockopt com a opção SO_RCVTIMEO é crucial. Ela configura um timeout de 2 segundos na função de recebimento (recvfrom). Se o transmissor não receber um ACK dentro desse tempo, a chamada recvfrom falhará, permitindo-nos implementar a lógica de retransmissão.
- **Loop Principal:**
 1. Lê uma linha de texto do teclado.
 2. Monta um Packet com o número de sequência atual (seqNum) e a mensagem.
 3. Entra em um loop de retransmissão (while (!ackReceived)):

- Envia o pacote para o receptor usando `sendto`.
- Tenta receber um `AckPacket` usando `recvfrom`.
- **Se um ACK é recebido (`recv_len > 0`):** Ele verifica se o `ackNum` corresponde ao `seqNum` que enviou. Se corresponder, a entrega foi um sucesso. A flag `ackReceived` se torna `true`, o loop de retransmissão termina, e o `seqNum` é invertido (0 vira 1, 1 vira 0) para a próxima mensagem.
- **Se o timeout ocorre (`recv_len < 0`):** A mensagem "Timeout!" é exibida. O loop de retransmissão continua, fazendo com que o mesmo pacote seja enviado novamente. Isso garante a retransmissão automática.

3. Implementação do Receptor (Processo 2)

O receptor aguarda passivamente por pacotes, os valida e os entrega para a "camada de aplicação" (neste caso, imprimindo na tela).

- **Inicialização:** Cria um socket UDP e o "amarra" (bind) à porta 8080, permitindo que ele receba mensagens de qualquer endereço IP (`INADDR_ANY`).
- **Controle de Ordem:** Mantém uma variável `expectedSeqNum`, que guarda o número de sequência do próximo pacote que ele espera receber. Começa em 0.
- **Loop Principal:**
 1. Aguarda indefinidamente por um pacote usando `recvfrom`.
 2. Quando um pacote chega, ele verifica o `packet.seqNum`.
 3. Se `packet.seqNum == expectedSeqNum`: É o pacote correto e esperado.
 - A mensagem é impressa na tela (garantia de entrega e ordem).

Um `AckPacket` é montado com o `expectedSeqNum` e enviado de volta ao remetente.

- O `expectedSeqNum` é invertido para esperar pelo próximo pacote.
- 4. Se `packet.seqNum != expectedSeqNum`: É um pacote duplicado ou fora de ordem.
 - A mensagem é descartada (não é impressa), garantindo que não haja duplicatas.
 - **Crucial:** Ele reenvia um ACK do último pacote recebido com sucesso. Por exemplo, se ele espera o pacote 1 mas recebe o 0, ele reenvia o ACK 0. Isso resolve o problema de ACKs perdidos, pois informa ao transmissor que o pacote 0 já foi recebido e que ele pode prosseguir.

Exemplo de teste funcional

- **Requisitos:** Dois terminais, compilador `g++` e sistema Linux/macOS ou WSL no Windows.
- **Passos:**
 1. Compile os códigos:
 - `g++ transmissor.cpp -o transmissor`
 - `g++ receptor.cpp -o receptor`

2. Execute o receptor primeiro e depois execute o transmissor em outro terminal:
 - ./receptor
 - ./transmissor
3. Teste de comunicação: Apenas digite diferentes mensagens no transmissor, como por exemplo: “Olá mundo” ou “Trabalho de Redes”.
4. Teste de retransmissão: Interrompa temporariamente o receptor (Use Ctrl+Z), envie mensagem no transmissor. Observe o timeout e retransmissão automática. Retome o receptor (fg).
5. Teste de duplicação: Mande a mesma mensagem duas vezes. Observe que o receptor ignora o segundo envio se seqNUM estiver repetido.