Solution: UDP Sensor Client-Server

Overview

This solution demonstrates how to create a simple UDP client and server in C++ to simulate sensor data transmission.

Server Code

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int sockfd;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_len = sizeof(client_addr);

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
        return 1;
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(5000);

    if (bind(sockfd, (struct sockaddr*)&server_addr,
sizeof(server_addr)) < 0) {
        perror("Bind failed");
        close(sockfd);
        return 1;
    }

    std::cout << "Listening on port 5000..." << std::endl;

    while (true) {
```

```
        char buffer[1024];
        memset(buffer, 0, sizeof(buffer));

        int n = recvfrom(sockfd, buffer, sizeof(buffer), 0,
                         (struct sockaddr*)&client_addr,
&addr_len);

        if (n > 0) {
            char client_ip[INET_ADDRSTRLEN];
            inet_ntop(AF_INET, &client_addr.sin_addr, client_ip,
INET_ADDRSTRLEN);
            std::cout << "Received from " << client_ip << ": " <<
buffer << std::endl;

            // Optional acknowledgment
            // sendto(sockfd, "ACK", 3, 0, (struct
sockaddr*)&client_addr, addr_len);
        }
    }

    close(sockfd);
    return 0;
}
```

Client Code

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int sockfd;
    struct sockaddr_in server_addr;

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
        return 1;
    }
```

```
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(5000);
    server_addr.s_addr = inet_addr("127.0.0.1");

    while (true) {
        const char* message = "Temperature: 23°C";

        sendto(sockfd, message, strlen(message), 0,
               (struct sockaddr*)&server_addr,
sizeof(server_addr));

        std::cout << "Sending: " << message << std::endl;

        sleep(1);
    }

    close(sockfd);
    return 0;
}
```

Explanation

Server

- **socket()**: Creates a UDP socket.
- **bind()**: Binds the socket to port 5000 and all available IP addresses.
- **recvfrom()**: Waits for incoming UDP packets and receives them.
- **inet_ntop()**: Converts the client's IP address to a readable string.
- **Optional**: You can send an acknowledgment back to the client.

Client

- **socket()**: Creates a UDP socket.
- **sendto()**: Sends a message to the server at `127.0.0.1:5000`.
- **sleep(1)**: Waits for 1 second between messages.

Summary

- This exercise demonstrates basic UDP communication.
- UDP is **stateless** and **unreliable**: packets may be lost.
- Useful in IoT scenarios where speed matters more than guaranteed delivery.

Optional Challenge

You can extend this solution by:

- Adding a checksum in the message.
- Making the server send an "ACK" message.
- Making the client wait for ACK and retry if not received.