**4. Framing Protocol: WAP for Character Count**

**Steps to Implement the Character Count Protocol**

**1. Understand the Problem**

* **Sender**: Constructs frames with a leading character count byte.
* **Receiver**: Reads frames based on the count byte and processes the content.

**2. Key Components**

* **Sender**:
  + Calculate the length of the message.
  + Add the length as the first byte.
  + Send the frame.
* **Receiver**:
  + Read the first byte to determine the length of the frame.
  + Extract the rest of the frame based on the length.

**3. Plan the Implementation**

* Write a program that:
  1. Simulates the sender creating frames.
  2. Simulates the receiver reading frames.

**4. Code Implementation**

Here’s an example in C:

#include <stdio.h>

#include <string.h>

// Function to simulate the sender

void sender(const char \*messages[], int num\_messages, char frames[][100]) {

for (int i = 0; i < num\_messages; i++) {

int length = strlen(messages[i]); // Calculate the message length

frames[i][0] = length; // First byte is the length

strcpy(frames[i] + 1, messages[i]); // Copy the message after the length

}

}

// Function to simulate the receiver

void receiver(char frames[][100], int num\_frames) {

for (int i = 0; i < num\_frames; i++) {

int length = frames[i][0]; // Read the first byte as length

printf("Frame %d (Length: %d): %.\*s\n", i + 1, length, length, frames[i] + 1);

}

}

int main() {

const char \*messages[] = {"Hello", "World", "Character Count Protocol"};

int num\_messages = sizeof(messages) / sizeof(messages[0]);

char frames[10][100]; // Array to store frames

// Step 1: Sender creates frames

sender(messages, num\_messages, frames);

// Step 2: Receiver processes frames

printf("Receiver Output:\n");

receiver(frames, num\_messages);

return 0;

}

**5. Explanation of the Code**

1. **Sender Function**:
   * Calculates the length of each message.
   * Adds the length as the first byte of the frame.
   * Appends the message after the length byte.
2. **Receiver Function**:
   * Reads the first byte of each frame to get the message length.
   * Extracts and prints the message using the length.
3. **Main Function**:
   * Simulates messages as input.
   * Calls the sender and receiver functions to demonstrate the protocol.

**6. Compilation and Execution**

* Compile the code:

bash

Copy code

gcc -o char\_count\_protocol char\_count\_protocol.c

* Run the program:

bash

Copy code

./char\_count\_protocol

**7. Expected Output**

plaintext

Copy code

Receiver Output:

Frame 1 (Length: 5): Hello

Frame 2 (Length: 5): World

Frame 3 (Length: 26): Character Count Protocol

**8. Notes**

* The frames array is used to simulate communication. In a real scenario, frames could be sent over a network or written to a file.
* The protocol assumes that frames are correctly formatted with the first byte indicating the length.