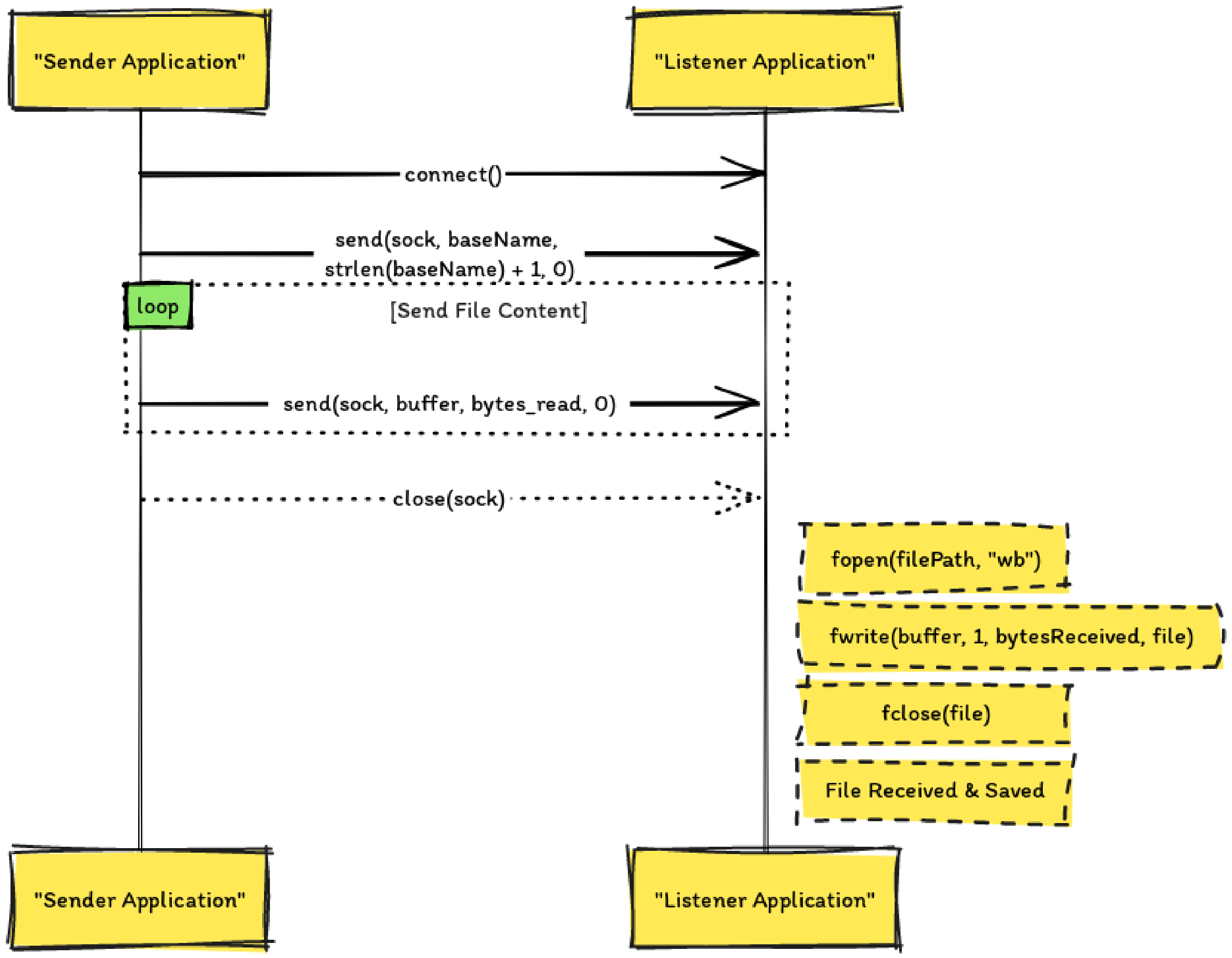
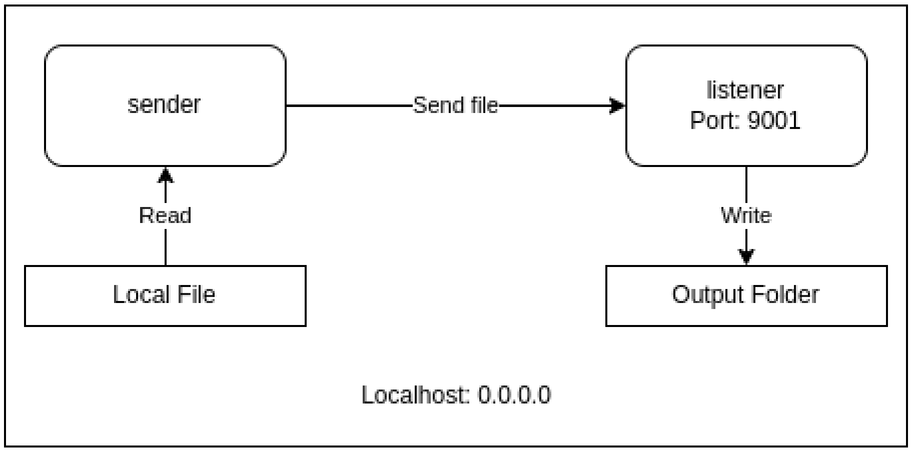
Practical Work 1 - TCP File Transfer

System Workflow



Test Setup



Code Walkthough

Sender and Receiver Workflow

This walkthrough describes the code workflow of a simple file transfer system using TCP/IP sockets in C. The system consists of a sender and a receiver. The sender sends a file to the receiver, which saves it to a specified directory.

Sender Workflow

The sender's main tasks are parsing command-line arguments, creating a socket, connecting to the receiver, sending the file name, and finally sending the file content.

1. **Parse Command-Line Arguments**: The sender takes command-line arguments to specify the input file `-i`, the destination port `-d`, and the host address `-h`.

```
int opt;
char *filename = NULL;
int port = 0;
char *hostAddress = "127.0.0.1";

while((opt = getopt(argc, argv, "i:d:h:")) != -1) {
    switch(opt) {
        case 'i':
            filename = optarg;
            break;
        case 'd':
            port = atoi(optarg);
            break;
        case 'h':
            hostAddress = optarg;
            break;
        // Default case omitted for brevity
    }
}
```

2. **Create Socket**: The sender creates a TCP socket using `socket()`.

```
int sock = 0;
struct sockaddr_in serv_addr;
sock = socket(AF_INET, SOCK_STREAM, 0);
```

3. **Connect to the Receiver**: It uses `connect()` to establish a connection to the receiver's address and port.

```
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(port);
inet_pton(AF_INET, hostAddress, &serv_addr.sin_addr);
connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
```

4. **Send the File Name**: Before sending the file content, the sender sends the file name using `send()`.

```
 char *baseName = basename(filename);
send(sock, baseName, strlen(baseName) + 1, 0);
```

5. **Send the File Content**: The sender reads chunks of the file and sends them until the entire file is transmitted.

```
FILE *file = fopen(filename, "rb");
char buffer[BUFFER_SIZE];
while (!feof(file)) {
    int bytes_read = fread(buffer, 1, BUFFER_SIZE, file);
    send(sock, buffer, bytes_read, 0);
}
```

Receiver Workflow

The receiver's main tasks are parsing command-line arguments, creating a socket, binding to a port and address, listening for connections, accepting a connection, reading the file name, and saving the received file content.

1. **Parse Command-Line Arguments**: The receiver takes command-line arguments for the port `-p`, output folder `-o`, and bind address `-a`.

```
int port = 0;
char outputFolder[FILE_NAME_MAX] = {0};
char bindAddress[INET_ADDRSTRLEN] = "0.0.0.0";

while((opt = getopt(argc, argv, "p:o:a:")) != -1) {
    switch(opt) {
        case 'p':
            port = atoi(optarg);
            break;
        case 'o':
            strncpy(outputFolder, optarg, sizeof(outputFolder) - 1);
            break;
        case 'a':
            strncpy(bindAddress, optarg, sizeof(bindAddress) - 1);
            break;
        // Default case omitted for brevity
    }
}
```

2. **Create and Bind Socket**: The receiver creates a TCP socket and binds it to the specified address and port using `bind()`.

```
int server_fd = socket(AF_INET, SOCK_STREAM, 0);
struct sockaddr_in address;
address.sin_family = AF_INET;
address.sin_addr.s_addr = inet_addr(bindAddress);
address.sin_port = htons(port);
bind(server_fd, (struct sockaddr *)&address, sizeof(address));
```

3. **Listen and Accept Connection**: It listens for incoming connections with `listen()` and accepts a connection with `accept()`.

```
listen(server_fd, 3);
int new_socket = accept(server_fd, (struct sockaddr *)&address,
(socklen_t*)&addrlen);
```

4. **Read the File Name and Content**: The receiver first reads the file name sent by the sender, then reads the file content and saves it.

```
char fileName[FILE_NAME_MAX];
read(new_socket, fileName, sizeof(fileName));
FILE *file = fopen(filePath, "wb");
char buffer[BUFFER_SIZE];
while ((bytesReceived = read(new_socket, buffer, BUFFER_SIZE)) > 0) {
    fwrite(buffer, 1, bytesReceived, file);
}
```