```
#include <iostream>
#include <fstream>
#include <opencv2/opencv.hpp>
#include <winsock2.h>

#pragma comment(lib, "ws2_32.lib")

#define PORT 9999
#define BUFFER_SIZE 4096

int main() {
    WSADATA wsaData;
    SOCKET clientSocket;
    struct sockaddr_in serverAddr;
    int addrSize = sizeof(serverAddr);

    // Initialize Winsock
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        std::cerr << "WSAStartup failed" << std::endl;
        return 1;
    }

    // Create socket
    if ((clientSocket = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET) {
        std::cerr << "Socket creation failed" << std::endl;
        WSACleanup();
        return 1;
    }

    // Initialize server address
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); // Change to server's
IP
    serverAddr.sin_port = htons(PORT);

    // Connect to server
    if (connect(clientSocket, (struct sockaddr *)&serverAddr,
sizeof(serverAddr)) < 0) {
        std::cerr << "Connection failed" << std::endl;
        closesocket(clientSocket);
        WSACleanup();
        return 1;
    }

    std::cout << "Connected to server" << std::endl;

    // Open video file for writing
    std::ofstream videoFile("received_video.mp4", std::ios::binary);
    if (!videoFile) {
        std::cerr << "Video file open failed" << std::endl;
        closesocket(clientSocket);
        WSACleanup();
        return 1;
    }
```

```cpp
    // Receive video data and write to file
    char buffer[BUFFER_SIZE];
    int bytesRead;
    while ((bytesRead = recv(clientSocket, buffer, BUFFER_SIZE, 0)) > 0) {
        videoFile.write(buffer, bytesRead);
    }

    std::cout << "Video received successfully" << std::endl;

    // Close sockets and file
    videoFile.close();
    closesocket(clientSocket);
    WSACleanup();

    // Open and display received video
    cv::VideoCapture cap("received_video.mp4");
    if (!cap.isOpened()) {
        std::cerr << "Video file open failed" << std::endl;
        return 1;
    }

    cv::Mat frame;
    cv::namedWindow("Received Video", cv::WINDOW_NORMAL);
    while (true) {
        cap >> frame;
        if (frame.empty()) {
            break;
        }
        cv::imshow("Received Video", frame);
        if (cv::waitKey(30) == 27) {
            break;
        }
    }

    cap.release();
    cv::destroyAllWindows();

    return 0;
}
```