```cpp
#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 serverSocket, clientSocket;
    struct sockaddr_in serverAddr, clientAddr;
    int addrSize = sizeof(clientAddr);

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

    // Create socket
    if ((serverSocket = 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 = INADDR_ANY;
    serverAddr.sin_port = htons(PORT);

    // Bind socket
    if (bind(serverSocket, (struct sockaddr *)&serverAddr, sizeof(serverAddr))
== SOCKET_ERROR) {
        std::cerr << "Socket bind failed" << std::endl;
        closesocket(serverSocket);
        WSACleanup();
        return 1;
    }

    // Listen for connections
    if (listen(serverSocket, 1) == SOCKET_ERROR) {
        std::cerr << "Socket listen failed" << std::endl;
        closesocket(serverSocket);
        WSACleanup();
        return 1;
    }

    std::cout << "Server listening on port " << PORT << std::endl;

    // Accept connection
    if ((clientSocket = accept(serverSocket, (struct sockaddr *)&clientAddr,
&addrSize)) == INVALID_SOCKET) {
        std::cerr << "Connection accept failed" << std::endl;
        closesocket(serverSocket);
        WSACleanup();
        return 1;
    }

    std::cout << "Client connected" << std::endl;

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

    // Transmit video file over socket
    char buffer[BUFFER_SIZE];
    while (!videoFile.eof()) {
        videoFile.read(buffer, BUFFER_SIZE);
        int bytesRead = videoFile.gcount();
        if (send(clientSocket, buffer, bytesRead, 0) == SOCKET_ERROR) {
            std::cerr << "Socket send failed" << std::endl;
            videoFile.close();
            closesocket(clientSocket);
            closesocket(serverSocket);
            WSACleanup();
            return 1;
        }
    }

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

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

    return 0;
}
```