```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;
using System.Collections;
using System.Windows.Forms;
using System.Security.Cryptography;
using System.Drawing;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;


namespace Server
{

    #region DELEGATES

    // oggetto che incapsula gli argomenti dell'evento StatusChanged
    public class StatusChangedEventArgs : EventArgs
    {
        // messaggio che descrive l'evento
        private string EventMsg;
        // property...
        public string EventMessage
        {
            get
            {
                return EventMsg;
            }
            set
            {
                EventMsg = value;
            }
        }
        // costruttore...
        public StatusChangedEventArgs(string eventMsg)
        {
            EventMsg = eventMsg;
        }
    }

    // delegati per specificare i parametri da passare all'evento
    public delegate void StatusChangedEventHandler(object sender, StatusChangedEventArgs e);
    public delegate void UpdateClipboardCallback(System.Collections.Specialized.StringCollection paths);
    public delegate void UpdateUsersListBoxCallback(ICollection users);

    #endregion

    class MainServer
    {
        #region INIZIALIZZAZIONE

        public static event StatusChangedEventHandler StatusChanged;

        // Hashtable varie...
        public const int MAX_CLIENTS = 4; // max utenti/connessioni...
        // - utenti     (-> connessioni)
        public static Hashtable tableUsers = new Hashtable(MAX_CLIENTS);
        // - connessioni (-> utenti)
        public static Hashtable tableConnections = new Hashtable(MAX_CLIENTS);
        // - utenti      (-> connessioni per condivisione schermo)
        public static Hashtable tableClientsCapture = new Hashtable(MAX_CLIENTS);
        // - utenti      (->connessioni clipboard)
        public static Hashtable tableConnectionsClipboard = new Hashtable(MAX_CLIENTS);

        private IPAddress ipAddress;
        private string password;
        private int port, port_capture;
        private static int num_clients = 0;

        private TcpListener tcpCaptureListener;
```

```
        private TcpClient tcpCaptureClient;
        private MainForm mainForm;

        // il thread che resta in ascolto di connessioni
        private Thread threadListener;
        private TcpListener tcpConnectionListener;

        private TcpListener tcpClipboardListener;

        // condizione per il loop di ascolto connessioni
        private bool running = false;

        private const string sharedFolderLocation = ".\\Shared Files";
            public string getSharedFolderLocation() { return sharedFolderLocation; }
        private const string clientDisconnectMessage = "BYE";

        #endregion

        public MainServer(MainForm mainForm)
        {
            this.mainForm = mainForm;

            if (!Directory.Exists(@sharedFolderLocation))
                Directory.CreateDirectory(@sharedFolderLocation);
        }

        // scatta ad ogni cambiamento di stato, lo uso principalmente per aggiornare il Log della chat
        public static void OnStatusChanged(StatusChangedEventArgs e)
        {
            StatusChangedEventHandler statusHandler = StatusChanged;
            if (statusHandler != null)
            {
                // invoco il delegato
                statusHandler(null, e);
            }
        }

        #region GESTIONE UTENTI

        public void addUser(TcpClient tcpUser, string username)
        {
            lock (this)
            {
                tableUsers.Add(username, tcpUser);
                tableConnections.Add(tcpUser, username);
            }

            sendAdminMessage(username + " joined.");
            updateUsersListBox();
        }
        public void removeUser(TcpClient tcpUser)
        {
            if (tableConnections[tcpUser] == null)
                return;

            sendAdminMessage(tableConnections[tcpUser] + " left.");
            lock (this)
            {
                tableUsers.Remove(tableConnections[tcpUser]);

                tableClientsCapture.Remove(tableConnections[tcpUser]);
                tableConnectionsClipboard.Remove(tableConnections[tcpUser]);

                tableConnections.Remove(tcpUser);
            }
            num_clients--;

            if (tcpUser.Connected)
                tcpUser.GetStream().Close();

            // se e' in corso una cattura schermo...
            if (MainForm.isCapturing())
            {
                // forzo la terminazione...
```

```
                MainForm.workerObject.forceStop();
                MainForm.workerThread.Join(1000);
                MainForm.workerThread.Abort();
                // reinizializzo...
                MainForm.workerObject.forceStart();
                MainForm.workerThread = new Thread(MainForm.workerObject.DoWork);
                MainForm.workerThread.Start();
            }

            updateUsersListBox();
        }
        public void removeAllUsers()
        {
            lock (this)
            {
                tableUsers.Clear();
                tableClientsCapture.Clear();
                tableConnections.Clear();
                tableConnectionsClipboard.Clear();
            }

            updateUsersListBox();
        }
        private void updateUsersListBox()
        {
            if (mainForm == null)
                return;
            if (tableUsers.Count == 0)
            {
                try
                {
                    mainForm.Invoke(new UpdateUsersListBoxCallback(mainForm.updateUsersList), new object[] { ↙
 null });
                }
                catch (InvalidOperationException)
                {
                    //durante debug, da problemi quando si chiude la finestra
                    return;
                }
                return;
            }

            mainForm.Invoke(new UpdateUsersListBoxCallback(mainForm.updateUsersList), new object[] {        ↙
tableUsers.Keys });
        }

        #endregion

        #region GESTIONE CONNESSIONE

        public void configureServer(IPAddress address, string psw, string port, int port_scr)
        {
            ipAddress = address;
            port_capture = port_scr;
            this.password = psw;
            this.port = int.Parse(port);

            OnStatusChanged(new StatusChangedEventArgs(
                            "SERVER RUNNING\r\nip:port = " + ipAddress + ":" + port +
                            "\r\nscreen sharing port = " + port_capture + "\r\npassword: ***"));
        }
        public void setRunning(bool r)
        {
            if (running == r)
                return;
            running = r;

            if (r)
                MessageBox.Show("Server started running", "Info", MessageBoxButtons.OK, MessageBoxIcon.      ↙
Information);
            else
                MessageBox.Show("Server stopped running", "Info", MessageBoxButtons.OK, MessageBoxIcon.      ↙
Information);
        }
```

```
        public TcpListener startListening()
        {
            tcpConnectionListener = new TcpListener(ipAddress, port);
            tcpConnectionListener.Start();

            running = true; // permette di far partire il loop di ascolto
            openCaptureConnection();
            openClipboardConnection();

            // thread adibito a restare in ascolto
            threadListener = new Thread(listeningLoop);
            threadListener.Start();

            return tcpConnectionListener;
        }
        private void listeningLoop()
        {
            TcpClient tcpClient;

            //while(true)
            while (running)
            {
                // accetta connessione...
                try
                {
                    tcpClient = tcpConnectionListener.AcceptTcpClient();
                    // devo creare una nuova connessione ogni volta
                    // creo un thread che verifica ed accetta il cliente connesso
                    ParameterizedThreadStart parameterizedThreadVerifier = new ParameterizedThreadStart
    (acceptClient);
                    Thread threadVerifier = new Thread(parameterizedThreadVerifier);
                    threadVerifier.Start(tcpClient);
                }
                catch // arrivo qui in caso di errori gravi -> fermo il ciclo
                { break; }
            }

            // se arrivo qui, vuol dire che il Server si e' fermato (ad es. dal menu Disconnect)!
            removeAllUsers();
        }
        private void acceptClient(object tcpConnection)
        {
            /* MESSAGGI ACCETTAZIONE/RIFIUTO CLIENT:
             * ogni volta che un utente si connette, lo posso accettare mandandogli come risposta "1".
             * per rifiutarlo mando "0" poi il separatore "|" e la motivazione...
             */

            string msgClient;
            TcpClient tcpClient = (TcpClient)tcpConnection;
            StreamReader streamReceiver = new System.IO.StreamReader(tcpClient.GetStream());
            StreamWriter streamSender = new System.IO.StreamWriter(tcpClient.GetStream());

            // verifico i DIGEST delle password!!!
            MD5 md5 = new MD5CryptoServiceProvider();
            string passmd5 = BitConverter.ToString(md5.ComputeHash(ASCIIEncoding.Default.GetBytes
    (password)));

            // 1) leggo username
            string currUser = streamReceiver.ReadLine();

            if (currUser == "")
            {
                closeConnection(tcpClient, streamReceiver, streamSender);
                return;
            }

            if (num_clients == MAX_CLIENTS)
            {
                // scrivo risposta negativa
                streamSender.WriteLine("0|Too many connections. Try again later...");
                streamSender.Flush();
                closeConnection(tcpClient, streamReceiver, streamSender);
                return;
            }
```

```
            if (tableUsers.Contains(currUser))
            {
                // scrivo risposta negativa
                streamSender.WriteLine("0|Username already exists.");
                streamSender.Flush();
                closeConnection(tcpClient, streamReceiver, streamSender);
                return;
            }

            if (currUser == "Administrator")
            {
                // scrivo risposta negativa
                streamSender.WriteLine("0|The username is reserved.");
                streamSender.Flush();
                closeConnection(tcpClient, streamReceiver, streamSender);
                return;
            }

            // 2) controllo password
            if (!passmd5.Equals(streamReceiver.ReadLine()))
            {
                // scrivo risposta negativa
                streamSender.WriteLine("0|The password is not correct.");
                streamSender.Flush();
                closeConnection(tcpClient, streamReceiver, streamSender);
                return;
            }
            else
            {
                // TUTTO OK!

                // 3) scrivo risposta positiva
                streamSender.WriteLine("1");
                streamSender.Flush();

                // 4) mando la porta per condivisione schermo
                streamSender.WriteLine(port_capture);
                streamSender.Flush();

                // aggiungo utente
                num_clients++;
                addUser(tcpClient, currUser);
                acceptCaptureClient(currUser);
                acceptClipboardClient(currUser);
            }

            try
            {
                // !!da qui, resto in attessa di messaggi dal cliente!!
                while (running)
                {
                    if (tcpClient.GetStream().DataAvailable)
                    {
                        try
                        {
                            msgClient = streamReceiver.ReadLine();
                        }
                        catch // errori in trasmissione...
                        {
                            removeUser(tcpClient);
                            break;
                        }

                        // se arriva un messaggio di disconnessione o errato...
                        if (msgClient == null || msgClient.Trim().Equals(clientDisconnectMessage))
                        {
                            removeUser(tcpClient);
                            break;
                        }
                        else // se no viene mandato a tutti gli utenti connessi
                            sendUserMessage(currUser, msgClient);
                    }
                }
```

```
            }
            catch
            {
                // se si sono verificati errori nella comunicazione...
                removeUser(tcpClient);
            }

            // end loop
            closeConnection(tcpClient, streamReceiver, streamSender);
        }
        public void closeConnection(TcpClient tcpClient, StreamReader srReceiver, StreamWriter swSender)
        {
            tcpClient.Close();
            srReceiver.Close();
            swSender.Close();
        }

        #endregion

        #region GESTIONE CLIPBOARD

        private void openClipboardConnection()
        {
            // per la clipboard uso la porta per la cattura +1!
            // (viene comunicata al client ogni volta)
            tcpClipboardListener = new TcpListener(ipAddress, port_capture + 1);
            tcpClipboardListener.Start();
        }
        private void acceptClipboardClient(string currUser)
        {
            TcpClient clipboardClient = tcpClipboardListener.AcceptTcpClient();

            tableConnectionsClipboard.Add(currUser, clipboardClient);

            // creo thread per la ricezione della clipboard
            ParameterizedThreadStart parameterizedThreadReceiver = new ParameterizedThreadStart
    (receiveClipboard);
            Thread threadReceiver = new Thread(parameterizedThreadReceiver);
            threadReceiver.SetApartmentState(ApartmentState.STA);
            threadReceiver.Start(clipboardClient);
        }

        public void sendClipboardFile(object data)
        {
            /* MESSAGGI TRASMISSIONE FILE:
            * 1) "FileQ|U", dove Q e' la quantita' di file da ricevere, "|" e' un separatore dato che non si
            *      conosce il numero di cifre di Q, e U e' lo username.
            * 2a) "File: " per indicare l'imminente trasferimento
            * 2b) in alternativa, "Abort"
            * 3) se non si ha abortito, viene spedito il file serializzato
            */

            int quantity = 0;
            IDataObject file = (IDataObject)data;
            object fromClipboard = file.GetData(DataFormats.FileDrop, true);
            foreach (string sourceFileName in (Array)fromClipboard)
                quantity++;

            /* per essere thread-safe posso copiare localmente la lista utenti
             *      TcpClient[] tcpClients = new TcpClient[MainServer.tableConnectionsClipboard.Count];
                    MainServer.tableConnectionsClipboard.Values.CopyTo(tcpClients, 0);
             *
             * ...oppure imporre un lock/Monitor (meglio)
             * ...il lock è in esclusione con gli altri su questo oggetto (messi sui metodi add/remove degli
     utenti)
             */

            string why = "Sharing Clipboard";
            mainForm.Invoke(new DisableClipboardCallback(mainForm.disableClipboard), new object[] { why });

            // controlli PRIMA di spedire l'header "FileQ|Server"
            foreach (string sourceFileName in (Array)fromClipboard)
            {
```

```
            if (Directory.Exists(sourceFileName) || Path.GetFileName(sourceFileName) == "")
            {
                System.Windows.Forms.MessageBox.Show("Send failed: can't share a directory!", "Error",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);

                why = "Share Clipboard";
                mainForm.Invoke(new DisableClipboardCallback(mainForm.enableClipboard), new object[] { ↙
why });
                return;
            }

            try
            {
                FileInfo fleMembers = new FileInfo(sourceFileName);
                float size = (float)(fleMembers.Length / 1024 / 1024); //MB!
                if (size > 50)
                {
                    MessageBox.Show("Send failed: file '" + sourceFileName + "' is too big (50 MB max)!" ↙
,
                                    "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

                    why = "Share Clipboard";
                    mainForm.Invoke(new DisableClipboardCallback(mainForm.enableClipboard), new object[] ↙
 { why });
                    return;
                }
            }
            catch
            {
                why = "Share Clipboard";
                mainForm.Invoke(new DisableClipboardCallback(mainForm.enableClipboard), new object[] { ↙
why });
                return;
            }
        }

        lock (this)
        {
            TcpClient[] tcpClients = new TcpClient[tableClientsCapture.Count];
            tableClientsCapture.Values.CopyTo(tcpClients, 0);
            foreach (TcpClient tcpClient in tcpClients)
            {
                try
                {
                    if (tcpClient == null)
                    {
                        continue;
                    }

                    StreamWriter sw = new StreamWriter(tcpClient.GetStream());
                    sw.WriteLine("File" + quantity.ToString() + "|Server");
                    sw.Flush();

                    foreach (string sourceFileName in (Array)fromClipboard)
                    {
                        try
                        {
                            // serializzo file
                            byte[] fileNameByte = Encoding.ASCII.GetBytes(Path.GetFileName ↙
(sourceFileName));
                            byte[] fileData = File.ReadAllBytes(sourceFileName);
                            byte[] clientData = new byte[4 + fileNameByte.Length + fileData.Length];
                            BitConverter.GetBytes(fileNameByte.Length).CopyTo(clientData, 0);
                            fileNameByte.CopyTo(clientData, 4);
                            fileData.CopyTo(clientData, 4 + fileNameByte.Length);

                            sw.WriteLine("File: " + Path.GetFileName(sourceFileName));
                            sw.Flush();
                            sw.WriteLine(Convert.ToBase64String(clientData));
                            sw.Flush();

                            sendAdminMessage("Server shared his clipboard " +
                           ((quantity == 1) ? "(File)" : " (Multiple files)"));
                        }
```

```
                        catch (ArgumentException)
                        {
                            // si e' cercato di copiare una cartella
                            MessageBox.Show("Sharing failed: you can't copy a directory", "Error",
                                                MessageBoxButtons.OK, MessageBoxIcon.Error);

                            sendAdminMessage("Server ABORTED clipboard sharing.");

                            sw.WriteLine("Abort");
                            sw.Flush();

                            why = "Share Clipboard";
                            mainForm.Invoke(new DisableClipboardCallback(mainForm.enableClipboard), new ↙
    object[] { why });
                            return;
                        }
                        catch (Exception)
                        {
                            // puo' darsi che il Server si disconnetta/crashi mentre spedisce...

                            sendAdminMessage("Server ABORTED clipboard sharing.");

                            sw.WriteLine("Abort");
                            sw.Flush();

                            why = "Share Clipboard";
                            mainForm.Invoke(new DisableClipboardCallback(mainForm.enableClipboard), new ↙
    object[] { why });
                            return;
                        }
                    }
                    //sw.Close();
                }
                catch
                {
                    // errori di rete (il Server probabilmente e' uscito)
                    MessageBox.Show("Sharing failed: network error. (Server crashed?)", "Error",
                                                MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
            }
        }


        why = "Share Clipboard";
        mainForm.Invoke(new DisableClipboardCallback(mainForm.enableClipboard), new object[] { why });
    }
    public void sendClipboardText(string text)
    {
        //TcpClient[] tcpClients = new TcpClient[MainServer.tableConnectionsClipboard.Count];
        //MainServer.tableConnectionsClipboard.Values.CopyTo(tcpClients, 0);

        lock (this)
        {
            TcpClient[] tcpClients = new TcpClient[tableClientsCapture.Count];
            tableClientsCapture.Values.CopyTo(tcpClients, 0);
            foreach (TcpClient tcpClient in tcpClients)
            {
                try
                {
                    if (tcpClient == null)
                        continue;

                    StreamWriter sw = new StreamWriter(tcpClient.GetStream());
                    // avverto che sto mandando del testo!
                    sw.WriteLine("TextServer");
                    sw.Flush();
                    sw = null;
                    // mando il testo
                    Byte[] sendBytes = Encoding.ASCII.GetBytes(text);
                    NetworkStream sendstr = tcpClient.GetStream();
                    sendstr.Write(sendBytes, 0, sendBytes.Length);
                }
                catch
```

```
                {
                    removeUser(tcpClient);
                    continue;
                }
            }
        }
        sendAdminMessage("Server shared clipboard (Text)");

        string why = "Share Clipboard";
        mainForm.Invoke(new DisableClipboardCallback(mainForm.enableClipboard), new object[] { why });
    }
    public void sendClipboardImage(Bitmap bmp)
    {
        //TcpClient[] tcpClients = new TcpClient[MainServer.tableConnectionsClipboard.Count];
        //String[] users = new String[MainServer.tableUsers.Count];
        //MainServer.tableConnectionsClipboard.Values.CopyTo(tcpClients, 0);
        //MainServer.tableUsers.Keys.CopyTo(users, 0);

        lock (this)
        {
            TcpClient[] tcpClients = new TcpClient[tableClientsCapture.Count];
            tableClientsCapture.Values.CopyTo(tcpClients, 0);
            foreach (TcpClient tcpClient in tcpClients)
            {
                try
                {
                    if (tcpClient == null)
                        continue;

                    NetworkStream networkStream = tcpClient.GetStream();

                    StreamWriter sw = new StreamWriter(tcpClient.GetStream());
                    sw.WriteLine("ImagServer");
                    sw.Flush();

                    IFormatter myformat = new BinaryFormatter();
                    myformat.Serialize(networkStream, bmp);

                }
                catch
                {
                    removeUser(tcpClient);
                    continue;
                }
            }
        }
        sendAdminMessage("Server shared his clipboard (Image)");

        string why = "Share Clipboard";
        mainForm.Invoke(new DisableClipboardCallback(mainForm.enableClipboard), new object[] { why });
    }
    private void receiveClipboard(object tcpConnection)
    {
        byte[] buff = new byte[1024];
        string text, username;
        // path dei file ricevuti
        System.Collections.Specialized.StringCollection paths = new System.Collections.Specialized.
StringCollection();

        TcpClient tcpClibpoardClient = (TcpClient)tcpConnection;
        NetworkStream networkStream = tcpClibpoardClient.GetStream();
        StreamReader streamReader = new StreamReader(tcpClibpoardClient.GetStream());
        bool abort = false, error = false;

        // I PRIMI 4 CARATTERI INDICANO IL TIPO DI DATO NELLA CLIPBOARD:
        // "File", "Text", "Imag"
        while (running)
        {
            try
            {
                string msgLine = streamReader.ReadLine();

                username = msgLine.Substring(4);
```

```
                    #region FILE

                    string fileFullPath = "";

                    if (msgLine.Substring(0, 4).Equals("File"))
                    {

                        string why = "Receiving";
                        // per immettere dei dati nella clipboard devo disattivarla (?)
                        mainForm.Invoke(new DisableClipboardCallback(mainForm.disableClipboard), new object
    [] { why });
                        paths.Clear();

                        /* MESSAGGI TRASMISSIONE FILE:
                         * 1) "FileQ|U", dove Q e' la quantita' di file da ricevere, "|" e' un separatore
    dato che non si
                         *      conosce il numero di cifre di Q, e U e' lo username.
                         * 2a) "File: " per indicare l'imminente trasferimento
                         * 2b) in alternativa, "Abort"
                         * 3) se non si ha abortito, viene spedito il file serializzato
                         */

                        // in questo caso, la linea dopo "File" contiene la quantita', un '|' per separare,
    poi lo username
                        int separator_index = msgLine.IndexOf('|');
                        username = msgLine.Substring(separator_index + 1);
                        string quant_str = msgLine.Substring(4, separator_index-4); //Substring(start,
    LENGTH!!!)

                        int quant = int.Parse(quant_str);

                        for (int j = 0; j < quant; j++)
                        {
                            // per ogni file in ricezione, ottengo una stringa formata da "File: X" dove X e
    ' il nome del file
                            // in caso di errore invece "Abort"
                            string line = streamReader.ReadLine();

                            if (line == null)
                            {
                                error = true;
                                break;
                            }
                            if (line.Equals("Abort"))
                            {
                                abort = true;
                                break;
                            }
                            if (!line.Substring(0, 6).Equals("File: "))
                            {
                                error = true;
                                break;
                            }

                            // trasferimento file!

                            string fileName = line.Substring(6);

                            sendAdminMessage(username + " is sharing a file: '" + fileName + "'");

                            byte[] clientData = Convert.FromBase64String(streamReader.ReadLine());

                            // apro file da scrivere
                            fileFullPath = Path.GetFullPath(@sharedFolderLocation + "\\") + username + "_" +
     fileName;
                            BinaryWriter bWrite = new BinaryWriter(File.Open(fileFullPath, FileMode.Create))
    ;

                            int offset = fileName.Length + 4;
                            // scrivo (con offset: "File: " (4?) + lunghezza nome file)
                            bWrite.Write(clientData, offset, clientData.Length - offset);
                            // chiudo!!
                            bWrite.Close();
                            // aggiungo alla lista dei file in locale
```

```
                        paths.Add(fileFullPath);
                        // aggiorno il log ogni volta che salvo un file...
                        OnStatusChanged(new StatusChangedEventArgs("File saved in: '" + fileFullPath +
                            "'\r\n(Use 'File -> Open folder' to see shared files)"));

                        // rimbalzo il fle agli altri utenti

                        lock (this)
                        {
                            TcpClient[] tcpClients = new TcpClient[tableClientsCapture.Count];
                            tableClientsCapture.Values.CopyTo(tcpClients, 0);
                            foreach (TcpClient tcpClient in tcpClients)
                            {
                                try
                                {
                                    if (tcpClient == null || username.Equals(tableConnections
[tcpClient]))
                                        continue;

                                    StreamWriter sw = new StreamWriter(tcpClient.GetStream());
                                    // stessa prassi:
                                    // primo messaggio col tipo ("File1|U", 1 perche' mandiamo 1 file
alla volta)
                                    sw.WriteLine("File1|" + username);
                                    sw.Flush();
                                    // poi "File: fileName"
                                    sw.WriteLine("File: " + Path.GetFileName(fileName));
                                    sw.Flush();
                                    // poi il file serializzato
                                    sw.WriteLine(Convert.ToBase64String(clientData));
                                    sw.Flush();
                                    sw = null;
                                }
                                catch
                                {
                                    sendAdminMessage("Error receiving clipboard from user '" + username
+ "'. User will be removed");
                                    removeUser(tcpClient);
                                    continue;
                                }
                            }
                        }
                    }
                    if (error)
                        sendAdminMessage(username + " encountered an error while sharing clipboard.");
                    if (abort)
                        sendAdminMessage(username + " aborted clipboard sharing.");
                    else
                    {
                        sendAdminMessage(username + " shared clipboard " +
                                        ((quant == 1) ? "(File)" : " (Multiple files)"));

                        // aggiorno la clipboard con tutti i path dei file salvati in locale!
                        mainForm.Invoke(new UpdateClipboardCallback(updateClipboardWithFilePaths), new
object[] { paths });
                    }

                    why = "Share Clipboard";
                    mainForm.Invoke(new DisableClipboardCallback(mainForm.enableClipboard), new object[]
 { why });
                }

                #endregion

                #region TEXT
                else if (msgLine.Substring(0, 4).Equals("Text"))
                {
                    string Reason = "Receiving";
                    mainForm.Invoke(new DisableClipboardCallback(mainForm.disableClipboard), new object
[] { Reason });

                    byte[] bytes = new byte[tcpClibpoardClient.ReceiveBufferSize];

                    // bloccante finche' non viene letto un byte
```

```
                    networkStream.Read(bytes, 0, (int)tcpClibpoardClient.ReceiveBufferSize);
                    string text_bytes = Encoding.ASCII.GetString(bytes);
                    text = trimToEnd(text_bytes);

                    lock (this)
                    {
                        TcpClient[] tcpClients = new TcpClient[tableClientsCapture.Count];
                        tableClientsCapture.Values.CopyTo(tcpClients, 0);
                        foreach (TcpClient tcpClient in tcpClients)
                        {
                            try
                            {
                                if (tcpClient == null || username.Equals(tableConnections[tcpClient]))
                                    continue;

                                StreamWriter sw = new StreamWriter(tcpClient.GetStream());
                                sw.WriteLine("Text" + username);
                                sw.Flush();
                                sw = null;

                                NetworkStream sendstr = tcpClient.GetStream();
                                Byte[] sendBytes = Encoding.ASCII.GetBytes(text);
                                sendstr.Write(sendBytes, 0, sendBytes.Length);
                            }
                            catch
                            {
                                sendAdminMessage("Error receiving clipboard from user '" + username + "' ↙
. User will be removed");
                                removeUser(tcpClient);
                                continue;
                            }
                        }
                    }

                    // devo incapsularlo in un oggetto IDataObject per memorizzarlo nella clipboard!
                    IDataObject dataClipboard = new DataObject();
                    dataClipboard.SetData(text);
                    Clipboard.SetDataObject(dataClipboard, true);

                    Reason = "Share Clipboard";
                    mainForm.Invoke(new DisableClipboardCallback(mainForm.enableClipboard), new object[] ↙
 { Reason });
                    sendAdminMessage(username + " shared his clipboard (Text): '"+ text +"'");
                }
                #endregion

                #region IMAGE
                else if (msgLine.Substring(0, 4).Equals("Imag"))
                {
                    string why = "Receiving";
                    mainForm.Invoke(new DisableClipboardCallback(mainForm.disableClipboard), new object  ↙
[] { why });

                    Stream imageStream = tcpClibpoardClient.GetStream();
                    IFormatter formatter = new BinaryFormatter();
                    Bitmap bmp = (Bitmap)formatter.Deserialize(imageStream);

                    //TcpClient[] tcpClients = new TcpClient[MainServer.tableConnectionsClipboard.Count] ↙
;
                    //String[] users = new String[MainServer.tableUsers.Count];
                    //MainServer.tableConnectionsClipboard.Values.CopyTo(tcpClients, 0);
                    //MainServer.tableUsers.Keys.CopyTo(users, 0);

                    lock (this)
                    {
                        TcpClient[] tcpClients = new TcpClient[tableClientsCapture.Count];
                        tableClientsCapture.Values.CopyTo(tcpClients, 0);
                        foreach (TcpClient tcpClient in tcpClients)
                        {
                            try
                            {
                                if (tcpClient == null || username.Equals(tableConnections[tcpClient]))
                                    continue;
```

```
                                NetworkStream nS = tcpClient.GetStream();

                                StreamWriter sW = new StreamWriter(tcpClient.GetStream());
                                sW.WriteLine("Imag" + username);
                                sW.Flush();

                                IFormatter myformat = new BinaryFormatter();
                                myformat.Serialize(nS, bmp);
                            }
                            catch
                            {
                                sendAdminMessage("Error receiving clipboard from user '" + username + "' ↙
    . User will be removed");
                                removeUser(tcpClient);
                                continue;
                            }
                        }
                    }
                    Clipboard.SetImage(bmp);

                    sendAdminMessage(username + " shared his clipboard (Image)");

                    why = "Share Clipboard";
                    mainForm.Invoke(new DisableClipboardCallback(mainForm.enableClipboard), new object[] ↙
     { why });
                }
                #endregion
            }
            catch
            {
                // di solito si arriva qui quando il Client relativo al thread si e' disconnesso...
                // per sicurezza, rimuoviamo l'utente
                removeUser(tcpClibpoardClient);
                break;
            }
        }
        //end loop
        //tcpClibpoardClient.GetStream().Close();
        tcpClibpoardClient.Close();
        streamReader.Close();
        networkStream.Close();

        closeClipboardConnection();
    }
    private static void updateClipboardWithFilePaths(System.Collections.Specialized.StringCollection     ↙
paths)
    {
        Clipboard.SetFileDropList(paths);
    }
    public void closeClipboardConnection()
    {
        // chiudiamo prima tutti gli stream in uscita della clipboard
        // nel caso il Server stia spedendo!
        foreach(TcpClient client in tableConnectionsClipboard.Values)
            if (client.Connected)
                client.GetStream().Close();

        tcpClipboardListener.Stop();
    }

    #endregion

    #region GESTIONE CATTURA

    public void openCaptureConnection()
    {
        try
        {
            tcpCaptureListener = new TcpListener(ipAddress, port_capture);
            tcpCaptureListener.Start();
        }
        catch (Exception) { return; }
    }
    public void acceptCaptureClient(string currUser)
```

```
        {
            try
            {
                tcpCaptureClient = tcpCaptureListener.AcceptTcpClient();
                tableClientsCapture.Add(currUser, tcpCaptureClient);
            }
            catch (Exception) { return; }
        }

        // funzione usata dal CaptureWorker per spedire le immagini della cattura schermo!
        public static void sendFrameCapture(Bitmap bmp)
        {
            IFormatter formatter = new BinaryFormatter();

            // Problema: non posso usare lock(this) poiche' sono in un metodo statico.
            // Devo considerare il caso di un utente che si disconnette mentre spedisco...

            TcpClient[] tcpClients = new TcpClient[tableClientsCapture.Count];
            tableClientsCapture.Values.CopyTo(tcpClients, 0);
            foreach (TcpClient tcpClient in tcpClients)
            {
                try
                {
                    if (tcpClient == null)
                        continue;

                    NetworkStream clientNetworkStream = tcpClient.GetStream();
                    formatter.Serialize(clientNetworkStream, bmp);
                }
                catch(Exception)
                {   // l'utente si e' disconnesso nel mentre...
                    // removeUser(tcpClient); -> non posso usarlo, poiche' sono in un metodo statico

                    continue;
                    // faccio continuare il ciclo, al prossimo turno l'utente non sara' nella lista
                }
            }

        }
        public void closeCaptureConnection()
        {
            tcpCaptureListener.Stop();
        }

        #endregion

        #region MESSAGGISTICA

        // messaggi mandati dall'admin (Server) a tutti i clienti connessi
        public void sendAdminMessage(string msg)
        {
            string msg_full = "Administrator: " + msg;
            StreamWriter streamWriterToClient;

            OnStatusChanged(new StatusChangedEventArgs(msg_full));

            lock (this)
            {
                TcpClient[] tcpClients = new TcpClient[tableClientsCapture.Count];
                tableClientsCapture.Values.CopyTo(tcpClients, 0);
                foreach (TcpClient tcpClient in tcpClients)
                {
                    try
                    {
                        if (msg.Trim() == "" || tcpClient == null)
                            continue;

                        // mando messaggio
                        streamWriterToClient = new StreamWriter(tcpClient.GetStream());
                        streamWriterToClient.WriteLine(msg_full);
                        streamWriterToClient.Flush();
                        streamWriterToClient = null;
                    }
                    catch // se si e' verificato un problema, l'utente non e' piu' raggiungibile -> lo
```

```
    rimuovo.
                    {
                        removeUser(tcpClient);
                    }
                }
            }
        }
        // messaggi mandati da un Client
        public void sendUserMessage(string from, string msg)
        {
            // come la funzione sendAdminMessage...

            string msg_full = from + " says: " + msg;
            StreamWriter streamWriterToClient;

            OnStatusChanged(new StatusChangedEventArgs(msg_full));

            lock (this)
            {
                TcpClient[] tcpClients = new TcpClient[tableClientsCapture.Count];
                tableClientsCapture.Values.CopyTo(tcpClients, 0);
                foreach (TcpClient tcpClient in tcpClients)
                {
                    try
                    {
                        if (msg.Trim() == "" || tcpClients == null)
                            continue;

                        streamWriterToClient = new StreamWriter(tcpClient.GetStream());
                        streamWriterToClient.WriteLine(from + " says: " + msg);
                        streamWriterToClient.Flush();
                        streamWriterToClient = null;
                    }
                    catch
                    {
                        removeUser(tcpClient);
                    }
                }
            }
        }

        #endregion

        private string trimToEnd(string input)
        {
            int index = input.IndexOf('\0');
            if (index < 0)
                return input;
            return input.Substring(0, index);
        }
    }
}
```