Trasmissione/ricezione via TCP con i Socket di messaggi di lunghezza variabile

Negli esempi visti in precedenza, la lunghezza di un messaggio trasmesso/ricevuto era limitata ad un massimo di 1024 bytes. Infatti nei codici è stato impostato un array di byte (*buffer*), questo

```
byte[] messageReceived = new byte[1024];
```

il quale consente di memorizzare un messaggio non più grande di 1024 bytes. Se avessimo voluto trasmettere un messaggio più lungo, sarebbe stato '*mozzato*' o, addirittura, il software avrebbe sollevato una eccezione.

Ma se volessi trasmettere un messaggio composto da un numero di bytes *arbitrari* che potrebbero essere superiori a quelli consentiti dall'array, potrei, ad esempio, utilizzare due soluzioni:

1. dividere il messaggio originale in più parti e trasmetterli separatamente fermo restando il buffer a 1024 bytes
2. informare il destinatario che ho l'intenzione di trasmettere un messaggio con un determinato numero di byte in modo che possa predisporre un adeguato buffer di ricezione impostando l'array con l'esatta dimensione del messaggio e poi effettuare l'effettiva trasmissione del messaggio stesso

La prima soluzione, anche se fattibile, richiederebbe un notevole sforzo del programmatore nel suddividere il messaggio originale, molto grande, in tanti '*pezzi*' ognuno dei quali non più grande di 1024 bytes e spedirli in maniera sequenziale.

La seconda soluzione è decisamente più semplice da implementare e richiede la trasmissione di due semplici informazioni:

1. il numero di byte del nostro messaggio
2. il messaggio stesso

Il destinatario, una volta ricevuto l'esatto numero di byte, predispone il suo buffer in maniera da poter contenere il messaggio e restare in ascolto della ricezione del messaggio effettivo. Quindi la trasmissione avviene in due passaggi molto semplici e consente di trasmettere messaggi di qualsiasi dimensione.

E' ovvio che il codice del server e del client devono essere necessariamente modificati per adeguarsi alla nuova realtà.

Ora analizziamo il codice sia del server che del client.

Di seguito il codice del server:

```
class Program
    {
        static void Main(string[] args)
        {
            EseguiServer();
            Console.ReadKey();
        }

        public static void EseguiServer()
        {
            Socket listener = new Socket(AddressFamily.InterNetwork,SocketType.Stream, ProtocolType.Tcp);
            try
            {
                listener.Bind(new IPEndPoint(IPAddress.Any, 3306));
                listener.Listen(10);

                Console.WriteLine("Attesa di connessioni ... \n");

                string data = null;

                Socket clientSocket;
                clientSocket = listener.Accept();

                while (true)
                {
                    byte[] bytes = new byte[4];
                    int numByte = clientSocket.Receive(bytes,0,4,0);
                    int dim = BitConverter.ToInt32(bytes,0);
                    Console.WriteLine("Lunghezza del messaggio: {0}\n",dim);
                    byte[] dati = new byte[dim];
                    int ricevuti = clientSocket.Receive(dati, 0, dim, 0);
                    data = Encoding.ASCII.GetString(dati,0, dim);
                    if (data.IndexOf("@") > -1)
                     {
                        clientSocket.Shutdown(SocketShutdown.Both);
                        clientSocket.Close();
                        listener.Close();
                        break;
                     }
                    else
                     {
                       Console.WriteLine("Testo ricevuto da {1}\n\n{0}\n****** FINE MESSAGGIO ******",
                    data, IPAddress.Parse(((IPEndPoint)clientSocket.RemoteEndPoint).Address.ToString()));
                       byte[] message = Encoding.ASCII.GetBytes("OK messaggio ricevuto!!! \n");
                       clientSocket.Send(message);
                       numByte = 0;
                     }
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
        }
    }
```

Le modifiche che ho apportato sono poche ma significative e sono relative alle seguenti istruzioni contenute nel ciclo while:

`byte[] bytes = new byte[4];`	Array di 4 byte che conterrà la dimensione del messaggio che sarà ricevuto
`int numByte = clientSocket.Receive(bytes,0,4,0);`	Richiamo il metodo Receive a cui passo 4 parametri: 1. array (*buffer*) che conterrà la dimensione del messaggio 2. l'offset all'interno del buffer (*posizione*), in questo caso rappresenta il primo byte (*cioè con indice 0*) 3. numero di byte da ricevere cioè 4 byte 4. Specifica i comportamenti di invio e ricezione di socket (in questo caso 0 cioè nessun comportamento particolare)
`int dim = BitConverter.ToInt32(bytes,0);`	Converto il buffer con la dimensione in byte in un numero a 32 bit a partire dal primo elemento (0) dell'array
`Console.WriteLine("Lunghezza del messaggio: {0}\n",dim);`	Stampo la dimensione
`byte[] dati = new byte[dim];`	Creo un buffer che conterrà l'intero messaggio di quelle dimensioni (*dim*)
`int ricevuti = clientSocket.Receive(dati, 0, dim, 0);`	Ricevo il messaggio memorizzandolo nel buffer 'dati'
`data = Encoding.ASCII.GetString(dati,0, dim);`	Converto il buffer contenente tutti i byte del messaggio a partire dal primo byte (0) e di quella dimensione (*dim*), in una stringa che poi visualizzo

E questo è il codice del client:

```
class Program
    {
        static void Main(string[] args)
        {
            EseguiClient();
            Console.WriteLine("Premi un tasto per finire");
            Console.ReadKey();
        }
        // metodo
        static void EseguiClient()
        {
            try
            {
                IPAddress ipAddr = IPAddress.Parse("127.0.0.1");
                IPEndPoint remoteEndPoint = new IPEndPoint(ipAddr, 3306);
                // creo l'oggetto socket
                Socket sender = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
                try
                {
                    sender.Connect(remoteEndPoint);

                    while (true)
                    {
                        // contiene la codifica della quantità di bytes da spedire a 32 bit
                        byte[] lunghezza = new byte[4];
                        //Inserisco il messaggio
                        Console.WriteLine("Inserisci un messaggio");
                        Console.ReadKey();
                        string messaggio = File.ReadAllText(@"F:\Documents\credobis.txt");
                        if (messaggio.IndexOf("@") > -1)
                          break;
                        // memorizzo la lunghezza del messaggio
                        lunghezza = BitConverter.GetBytes(messaggio.Length);
                        Console.WriteLine(messaggio.Length);
                        // trasmetto la lunghezza del mio messaggio trasmettendo l'array di byte 'lunghezza'
                        sender.Send(lunghezza);
                        byte[] messageSent = Encoding.ASCII.GetBytes(messaggio);
                        sender.Send(messageSent);
                        byte[] messageReceived = new byte[1024];
                        int byteRecv = sender.Receive(messageReceived);
                        Console.WriteLine("Messaggio dal Server -> {0} da {1}\n",
                                          Encoding.ASCII.GetString(messageReceived, 0, byteRecv),
                                    IPAddress.Parse(((IPEndPoint)sender.RemoteEndPoint).Address.ToString()));
                    }
                }
```

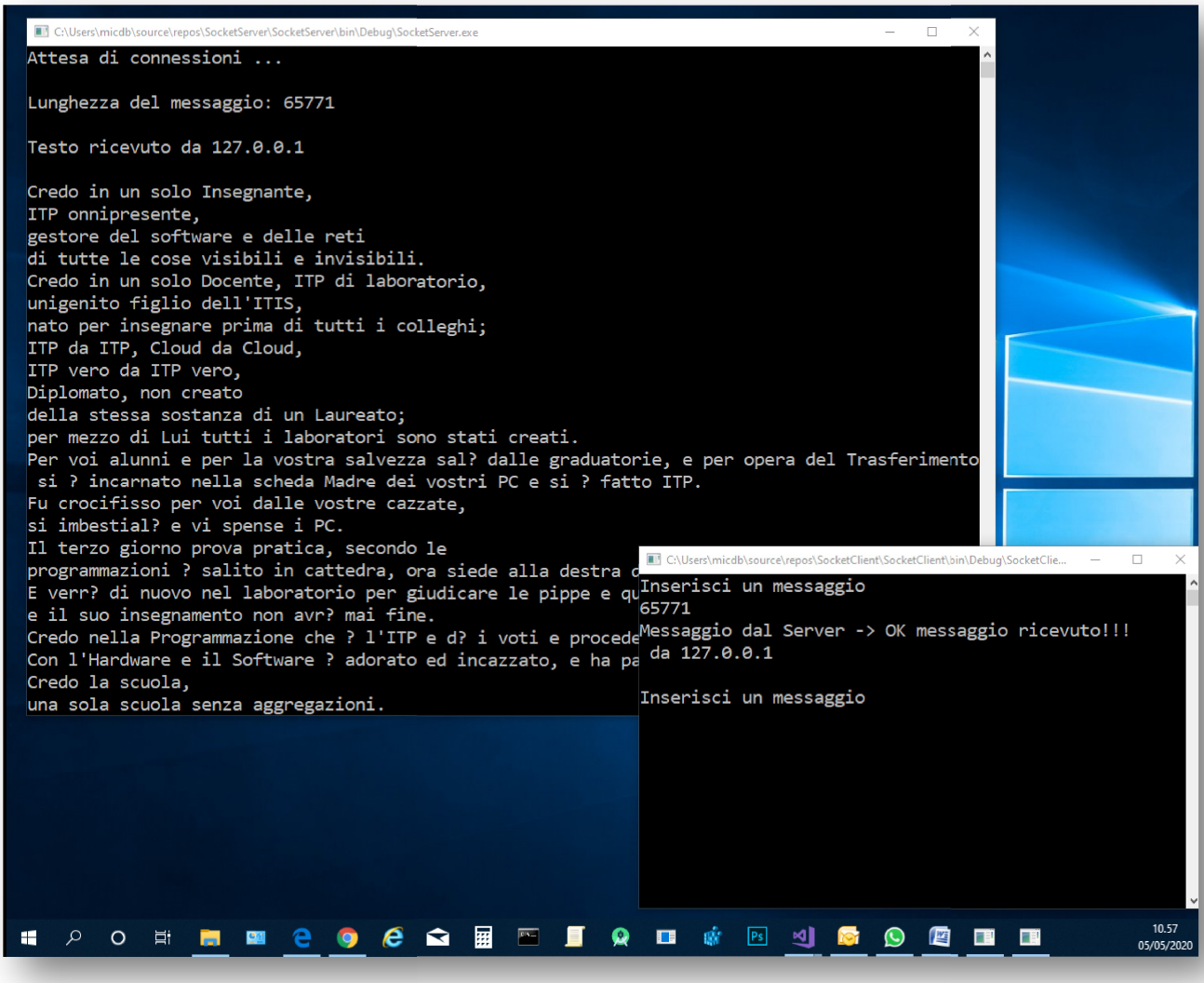
```
                catch (ArgumentNullException ane)
                {
                    Console.WriteLine("ArgumentNullException : {0}", ane.ToString());
                }
                catch (SocketException se)
                {
                    Console.WriteLine("SocketException : {0}", se.ToString());
                }
                catch (Exception e)
                {
                    Console.WriteLine("Unexpected exception : {0}", e.ToString());
                }
                sender.Shutdown(SocketShutdown.Both);
                sender.Close();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
        }
    }
```

Spiegazione:

`byte[] lunghezza = new byte[4];`	Creo un array (*buffer*) che conterrà la dimensione del messaggio
`Console.WriteLine("Inserisci un messaggio");` `Console.ReadKey();`	Visualizzo un messaggio e attendo la pressione di un tasto
`string messaggio = File.ReadAllText(@"F:\Documents\credobis.txt");`	Con questa istruzione, apro un file di testo puro (*e non uno qualsiasi*) per simulare l'inserimento di un messaggio molto lungo. La dimensione del messaggio 'estratto' dal file di testo è di circa 65KB che trasmetterò al server
`if (messaggio.IndexOf("@") > -1)` `  break;`	Se il messaggio contiene il carattere '@', uscirà dal ciclo
`lunghezza = BitConverter.GetBytes(messaggio.Length);`	Una volta estratta la stringa da file, converto la sua dimensione in byte e la memorizzo nel buffer lunghezza
`Console.WriteLine(messaggio.Length);`	Visualizzo la dimensione del messaggio
`sender.Send(lunghezza);`	Trasmetto la dimensione al server
`byte[] messageSent = Encoding.ASCII.GetBytes(messaggio);`	Converto tutto il messaggio in una sequenza di byte che memorizzo nel buffer '*messageSent*' per la trasmissione
`sender.Send(messageSent);`	Trasmetto l'intero buffer
`byte[] messageReceived = new byte[1024];`	Creo il buffer che conterrà il messaggio di ritorno dal server
`int byteRecv = sender.Receive(messageReceived);`	Ricevo il messaggio dal server

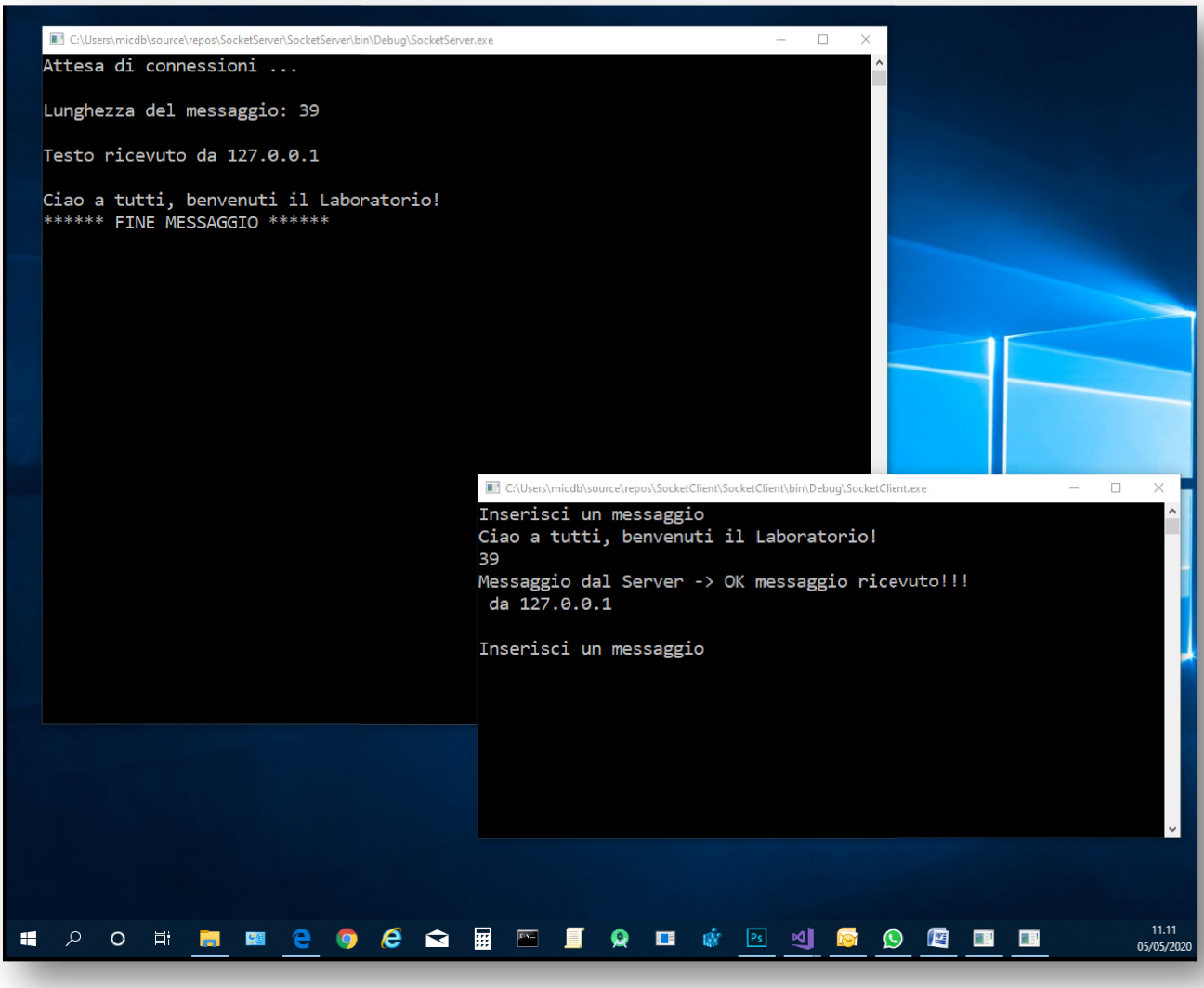
`Console.WriteLine("Messaggio dal Server -> {0} da {1}\n",`
`Encoding.ASCII.GetString(messageReceived, 0, byteRecv),`
`IPAddress.Parse(((IPEndPoint)sender.RemoteEndPoint).Address.ToString()));`
	Visualizzo il messaggio del server

Prova di esecuzione in locale:



Questa è la prova di esecuzione in locale; la console più grande è quella del server che ha ricevuto un messaggio di ben **65771** byte trasmesso dal client (console piccola). In entrambe le visualizzazioni, viene visualizzata la dimensione del messaggio.

Nella prossima pagina una prova di esecuzione con un messaggio digitato da tastiera:



Anziché utilizzare un file, ho digitato un breve messaggio di **39** byte che è stato trasmesso in tranquillità.