
Progetto Gestione Client-Server

Documentazione Completa

Progetto: Client-Server

Autori: ING. Maltraversi Marco

Versione:

1.12

Stato:

Finale

In collaborazione con



www.futuresoftware.it

info@futuresoftware.it

Indice

| | |
|---|---|
| Indice..... | 2 |
| 1. Introduzione..... | 2 |
| 2. Analisi e Specifica dei Requisiti..... | 3 |
| 2.1 Robot-Email | 3 |
| 2.2 Sincornizzazione ora..... | 4 |
| 2.3 Chat..... | 5 |
| 2.5 Trasferimento file | 6 |
| 2.4 Winsock..... | 7 |
| 2.6 Socket | 8 |
| 2.7 Immagini..... | 9 |

1. Introduzione

Il progetto consiste in un'insieme di programmi atti alla comunicazione client-server.

E' cosi strutturato:

Chat parlante costituita da un applicativo client e server,nella comunicazione sia il client che il server leggono i messaggi sintetizzati da una voce.

Sincronizzazione ora,sincronizza l'orologio del sistema collegandosi ad internet.

Trasferimento file fra client e server.

Robot per l'invio delle email ,che collegandosi ad un database stabilisce a chi inviare l'email .

2. Analisi e Specifica dei Requisiti

2.1 Robot-Email

RobotEmail è un programma che ha la funzione di Server AutoMailer verso Outlook Express, consente di gestire un Database di indirizzi Mail ed invia la posta mediante flag .

Il programma utilizza un Timer per monitorare costantemente il database delle Mail questo per verificare se vi sono nuovi messaggi da inviare, se così fosse crea un file “.log” contenente le eventuali nuove mail inviate.

Con questo programma non bisogna condividere manualmente il database andando a modificare l’ODBC,ma viene caricato automaticamente dal programma stesso.

(un ulteriore modifica sarebbe quella di creare un’area riservata all’Amministratore del programma facendo selezionare direttamente il database mediante un’interfaccia grafica invece di andare a scrivere direttamente nella porzione di codice il database caricato...

Riferimento nel codice...

Riga 99 nella funzione “Form_Load()”

Caricamento del nome del database...

```
sDBLocation = App.Path & "\data.mdb"
```

La funzione Timer:

All’interno di questa funzione viene gestito: il campo flag,

Il campo flag:

Viene effettuata una scansione dell’intera tabella del database, se il campo Flag nel Database è settato a 1 significa che l’Email risulta non inviata e quindi verrà spedita all’utente, una volta spedita, il flag verrà posto a zero ed il servizio continuerà a monitorare eventuali nuovi messaggi da inviare, questo significa che possiamo collegarci al Database ed inserire nuovi utenti, oltre che modificare i flag da 0 a 1 per ulteriori invii, tutto questo viene ripetuto tante volte quanti sono i record della tabella.

2.2 Sincronizzazione ora

SincronizzazioneOra è un programma per la sincronizzazione “dell’orologio” di un computer.

Il programma da come “uscita” 32 bit.

Questi bit vengono interpretati come il numero di secondi mancanti alle ore 00:00 (mezzanotte) del 1 gennaio 1900

Un piccolo esempio di funzionamento!

il tempo "1" sta a significare le ore 12:00:01 del 1 gennaio 1900

Questa “base” può funzionare fino all’anno 2036

Problemi riscontrati (gestiti e risolti)

Quando il nostro programma si connette al server remoto per sincronizzarsi, non ho una risposta istantanea, per nostra sfortuna.

Ma vi è un ritardo, dovuto a molteplici fattori, quali ritardo nella propagazione del segnale, ritardo nella elaborazione della richiesta, ed eventuali fattori aleatori quali il traffico della rete.

Quindi nei 32 bit del “tempo” che “ritorna” il nostro NTP servers

Contiene il tempo fra acknowledgement della connessione

e la ricezione dei dati.

Il problema insito nella risposta è che esso non tiene conto del tempo di latenza nella trasmissione

Ovvero tutti i tempi di cui sopra abbiamo già accennato.

Possiamo compensare l’errore che commettiamo aggiungendo metà del tempo di latenza.

In modo che il nostro programma posso sincronizzare il nostro calcolatore con l’effettivo orario!

Fallimento della sincronizzazione

In caso di fallimento nella sincronizzazione, il programma segnala all’utente che la sincronizzazione non è potuta giungere al termine.

2.3 Chat

1) IL LATO SERVER

A) Il server deve contenere i dati inseriti dall'utente quali: nome, nickname, età, e-mail, per poter utilizzare questo servizio.

B) Layout :Abbiamo scelto di utilizzare, come colori delle finestre, le stesse che utilizza windows per mantenere un aspetto più professionale del programma, ma si potevano selezionare diversi tipi di colori, anche colori speciali. Premendo il pulsante default o tramite la barra delle applicazioni della finestra principale, si ripristinano i colori delle finestre che abbiamo utilizzato noi.

C) START. Quando si preme questo bottone la chat invia le informazioni dell'utente al server, il quale riceve i dati e si prepara alla connessione.

d) Si può iniziare a chattare solamente quando il client è connesso

f) Invio automatico delle informazioni.

Questa è un'opzione per mandare alcuni dati mentre il client **“chiede” con un comando “whois”**

A nostra volta possiamo richiedere informazioni al client del tipo “whois”

Questo può avvenire solo se il client ha attivato (ON) questa opzione altrimenti non otterremo alcuna informazione.

G) Per ulteriori informazioni si può andare nel menu della barra delle applicazioni e premere il pulsante “cancella cronologia”

2) IL LATO CLIENT:

La principale differenza è che il client deve conoscere l'ip server.

Se l'ip sarà sbagliato o il client inizierà la trasmissione in maniera non corretta, il programma verrà terminato. 3) Si può facilmente testare l'applicazione su un computer qualsiasi, però devono essere installate le MsWinSock, le quali permettono di interagire tra più computer mediante i programmi creati con VisualBasic ecc., ma in più offrono l'opzione, come nel nostro caso, di testare un programma che normalmente dovrebbe interagire tra più computer, tramite un computer solo :

- A. Eseguire prima l'applicazione lato server
- B. Immagazzinare tutti i dati e premere start
- C. Avviare l'applicazione lato client
- D. Immagazzinare tutti i dati e l'ip del server
- E. Molto importante. Quando non sei connesso nel “net” si possiede comunque un ip

Si può testare il programma e il punto E senza essere connesso ad internet o ad una rete (grazie al winsock).I messaggi vengono inoltre sintetizzati ed è possibile sentire ciò che viene ricevuto sia nel lato client che server

2.5 Trasferimento file

Consiste nel trasferimento di file tra un client e un server.

Il programma si basa sul protocollo tcp.

Il TCP/IP e' un insieme di protocolli sviluppato per interconnettere reti di calcolatori, e quindi è quello che ho utilizzato per immettere questo sito sulla "grande rete" (Internet). Questi protocolli si presentano con un'architettura stratificata. Per i non addetti ai lavori vedrò di spiegarmi con un esempio.

Supponiamo di dover gestire del traffico di posta elettronica. Il primo livello di stratificazione e' quello identificato dal protocollo per l'invio del messaggio che identifica il mittente, il destinatario, nonché il testo dello stesso messaggio.

A questo livello si sovrappone il TCP, il protocollo che si incarica di controllare che i dati del mittente raggiungano effettivamente il destinatario. Il messaggio viene articolato in datagrammi e il TCP si preoccupa che ogni datagramma giunga correttamente al destinatario.

Ai due si sovrappone ancora un altro applicativo, l'IP, che e' deputato all'instradamento dei dati su Internet.

E per ultimo abbiamo un quarto livello, quello di interfaccia (Ethernet, seriale, ecc).

Il server si collega e ascolta sulla porta 1256

Il client effettua il collegamento al server specificando l'indirizzo ip del server.

Se il collegamento avviene correttamente si può iniziare il trasferimento dei file che avviene attraverso una particolare procedura di cui ne proponiamo di seguito una parte del codice:

```
Sub Pause(HowLong As Long)
Dim u% , tick As Long
tick = GetTickCount()

Do
u% = DoEvents
Loop Until tick + HowLong < GetTickCount
End Sub

' --- SendFile() Function
'
' Specifica un file da un computer ad un'altro via WinSock

Sub SendFile(Fname As String)
Dim DataChunk As String
Dim passes As Long
.....
```

I file inviati vengono salvati nella directory in cui si trova il client o server.

In caso di fallimento viene visualizzato un messaggio di errore.

2.4 Winsock

WinSock è un controllo OCX che permette di stabilire una comunicazione tra due computer connessi a una rete (locale o mondiale) sfruttando l'architettura Client/Server e il protocollo TCP o UDP.

Per ottenere una connessione tra Client e Server occorre conoscere l'indirizzo IP (indirizzo univoco) del Server, formato da 4 numeri compresi tra 0 e 255 separati da un punto. Quindi ad un indirizzo IP corrisponde uno e un solo computer connesso alla rete.

L'indirizzo IP, nella maggior parte dei computer, cambia ogni volta che ci si connette alla rete, quindi quando il Client si connette al Server deve conoscere il suo IP in quel preciso momento. Quando un computer è OffLine, (non collegato alla rete), l'indirizzo standard che identifica il proprio PC, indipendentemente dall'effettivo indirizzo di rete, è 127.0.0.1.

Il protocollo **TCP** (Transfer Control Protocol) consente di creare e mantenere una connessione con un computer remoto. Utilizzando la connessione, entrambi i computer possono inviare e ricevere dati. Il protocollo TCP è un protocollo basato sulla connessione, in quanto l'utente deve stabilire una connessione prima di poter procedere.

Il protocollo **UDP** (User Datagram Protocol) è un protocollo senza connessione. A differenza del TCP, non viene stabilita alcuna connessione tra i computer e la transazione tra due computer è analoga al passaggio di un messaggio che viene inviato da un computer all'altro senza che venga stabilita una connessione esplicita tra i due computer. La dimensione massima dei dati che è possibile inviare in ciascuna trasmissione dipende della rete.

In particolare, prendendo TCP come protocollo di riferimento, nel momento in cui i due computer sono in comunicazione, essi ricoprono dei ruoli ben definiti.

Il computer Server invia i dati a chi ne fa richiesta utilizzando la porta specificata mentre il computer Client esegue tale richiesta. L'operazione che avviene tra Client e Server è riassumibile in cinque punti fondamentali:

- Il computer Server è in attesa di richieste di comunicazione sulla porta indicata.
- Il computer Client richiede al Server l'autorizzazione ad iniziare una comunicazione.
- Una volta accettata la richiesta del Client, il computer Server invia i dati.

2.6 Socket

Un socket è un oggetto attraverso il quale un'applicazione che ne fa uso in modo specifico, invia o riceve dati attraverso la rete con altri socket che fanno uso dell'IPS (Internet Protocol Suite). I socket sono bidirezionali, ossia consentono allo stesso momento un flusso di dati sia in entrata che in uscita. Esistono due tipi di socket:

- Stream sockets, che consentono un flusso di dati continuo ed illimitato con la garanzia di evitare duplicazione di dati inviati.
- Datagram sockets, i quali non garantiscono che l'ordine di ricezione dei dati sia lo stesso di quello di invio, col rischio di non poter evitare duplicazione di dati (cioè stessi pacchetti di dati possono arrivare a destinazione più volte).

Questo è l'elenco degli eventi:

Evento Close : viene generato quando il computer remoto interrompe la connessione.

Evento Connect : viene generato nel momento in cui la connessione è stata stabilita con successo.

Evento ConnectionRequest : viene generato nel momento in cui un computer client invia una richiesta di connessione. Il server può quindi decidere se accettare o meno la comunicazione. In caso di rifiuto, il client riceverà l'evento Close.

Evento DataArrival : viene generato nel momento in cui un computer client riceve dei dati. L'evento viene generato nel caso in cui tutti i nuovi dati inviati col metodo GetData siano recuperati con successo. Per evitare duplicazione di pacchetti di dati, solamente i nuovi dati saranno considerati. La sintassi è la seguente: Winsock.DataArrival (BytesTotali As Long) dove BytesTotali rappresenta il numero di dati che possono essere recuperati.

Evento SendComplete : viene generato quando tutti i dati sono stati inviati con successo.

Evento SendProgress : viene generato quando il processo di invio dei dati è ancora in corso.

2.7 Immagini

Inoltre è stato sviluppato un programma gestione che gestisce tutti gli altri programmi sopra elencati

