Training per esperti

SERVIRE DATI
AD UN CLIENT UTILIZZANDO TCP/IP

Essendo disponibile sulla maggior parte dei computer, il TCP/IP permette di trasferire informazioni fra sistemi diversi

A cura di Alessandro Plantamura

A. INTRODUZIONE AL TCP/IP

Il TCP assicura la trasmissione affidabile sulle reti, consegnando i dati in sequenza senza errori, perdite o duplicazione. Il TCP ritrasmette il datagram finché non riceve una conferma (acknowledgment). Il TCP è un protocollo basato sulla connessione: ciò significa che i siti devono stabilire una connessione prima di trasferire dati. La trasmissione dei dati avviene fra un client ed un server. Il TCP permette più connessioni simultanee. Potete iniziare una connessione aspettando una connessione in ingresso o cercando attivamente una connessione con un indirizzo specificato. Quando stabilite connessioni TCP, dovete specificare l'indirizzo e una porta a quell'indirizzo. Porte differenti ad un dato indirizzo identificano servizi differenti a quell'indirizzo.

GENERALITÀ SUL MODELLO CLIENT/SERVER

Il modello client/server è un modello comune per le applicazioni in rete. In tale modello, un set di processi (client) richiede servizi da altri set di processi (server), come si vede nella fig. 1.

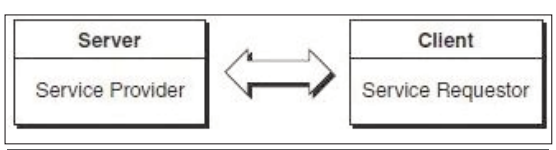

Figura 1. Modello client/server

GENERALITÀ SUL MODELLO CLIENT

La fig. 2 descrive il modello flowchart per la comunicazione TCP/IP in LabVIEW.

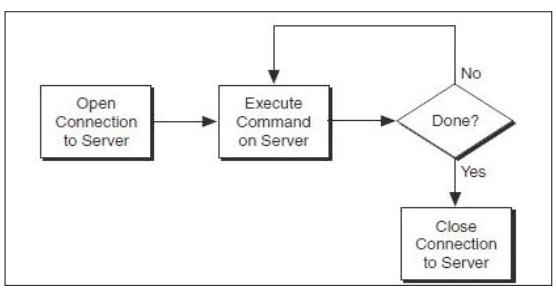

Figura 2. Modello client TCP/IP in LabVIEW

Per maggiori prestazioni, potete elaborare più comandi dopo avere aperto la connessione. Dopo l'esecuzione dei comandi, potete chiudere la connessione. Questo flowchart serve da modello per dimostrare l'implementazione di un dato protocollo in LabVIEW.

STEP PER LA COMUNICAZIONE CON IL MODELLO CLIENT

Il modello client permette ad un client di comunicare con un server aprendo in primo luogo la connessione verso il server, inviando un comando al server, ricevendo un comando dal server e chiudendo quindi la connessione verso il server.

APERTURA DI UNA CONNESSIONE VERSO IL SERVER

Il primo passo consiste nell'aprire una connessione verso il server. Il client deve aprire una porta della rete sulla macchina locale e quindi specificare l'indirizzo IP del server a cui connettersi.

INVIO DI COMANDI AL SERVER

Dopo avere stabilito una connessione verso il server, il client invia un messaggio riconosciuto dal server. Tale messaggio fa eseguire lavoro al server.

RICEZIONE DI RISPOSTE DAL SERVER

Dopo l'invio di un messaggio al server, il client deve sapere che il messaggio è stato accettato e che il server ha eseguito il lavoro richiesto. Ciò avviene attraverso una risposta inviata dal server al client.

CHIUSURA DELLA CONNESSIONE VERSO IL SERVER E REPORT DEGLI ERRORI

Dopo l'invio di tutti i comandi al server, il client chiude la connessione ed esegue il report degli errori.

GENERALITÀ SUL MODELLO SERVER

La fig. 3 illustra il modello semplificato di un server in LabVIEW. LabVIEW ripete questo intero processo finché non viene interrotto remotamente inviando un comando per terminare il VI. Il VI non riporta errori. Potrebbe restituire una risposta indicante che un comando non è valido, ma non

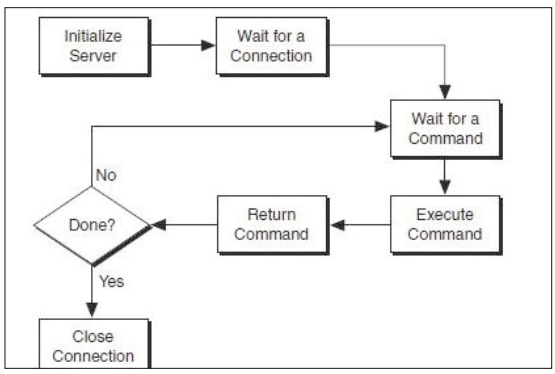

Figura 3. Modello server TCP/IP in LabVIEW

visualizza un'area di dialogo al verificarsi di un errore. Poiché un server potrebbe essere non presidiato, considerate attentamente come il server deve gestire eventuali errori. Probabilmente non volete visualizzare un'area di dialogo in caso di errore perché richiede un'interazione con l'utente sul server. Qualcuno dovrebbe cliccare il pulsante OK. Tuttavia, potreste desiderare che LabVIEW scriva un log delle transazioni e degli errori in un file o una stringa. Potete incrementare le prestazioni permettendo alla connessione di rimanere aperta. In questo modo, potete ricevere più comandi, ma bloccherete la connessione di altri client finché il client corrente non si disconnette. Potete ristrutturare il diagramma a blocchi in modo da gestire più client simultaneamente, come illustrato nella fig. 4.

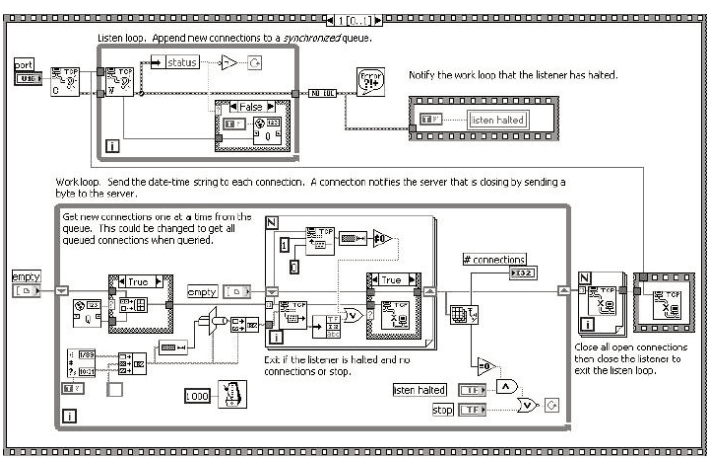

Figura 4. Gestione di più client simultaneamente

Il diagramma a blocchi usa le capacità multitasking di LabVIEW per eseguire due loop simultaneamente. Il loop superiore attende continuamente una connessione, quindi aggiunge la connessione ad una coda sincronizzata. Il loop inferiore controlla ciascuna delle connessioni aperte ed esegue tutti i comandi che sono stati ricevuti. Se si verifica un errore su una delle connessioni, la connessione viene interrotta. Quando l'utente abortisce il server, tutte le connessioni aperte si chiudono.

STEP PER LA COMUNICAZIONE CON IL MODELLO SERVER

Con il modello server è più complesso permettere a più client l'accesso al server. Il server deve innanzitutto inizializzarsi, quindi aspettare una connessione da un client. Quando la connessione è stabilita, il server attende un comando. Il server elabora ed esegue tutti i comandi e restituisce i risultati al client. Quando il client ha finito di inviare comandi al server, la connessione fra il server ed il client può essere chiusa.

INIZIALIZZAZIONE DEL SERVER

Il server viene inizializzato rimanendo in ascolto per una connessione su una particolare porta di rete.

ATTESA DI UNA CONNESSIONE

Il modello server permette a più client di connettersi al server. Ciò si ottiene aspettando una connessione da un client e mettendo quindi l'informazione della connessione in una coda utilizzando un modello di progetto produttore/consumatore. Il consumatore aspetta di ricevere un comando dal client.

ATTESA DI UN COMANDO

L'attesa di un comando permette di ricevere più comandi da più client. Ciò si ottiene utilizzando un consumatore per iterare attraverso ciascuno dei client al fine di determinare se un client ha inviato un messaggio.

ESECUZIONE DEL COMANDO E RESTITUZIONE DEI RISULTATI

Quando il server ha ricevuto un comando da un client, il server esegue il comando effettuando del lavoro. Quando il lavoro è terminato, il server risponde con i risultati del lavoro o un errore se non è stato possibile eseguire il lavoro.

CHIUSURA DELLA CONNESSIONE

Quando tutti i client hanno finito di inviare messaggi al server, la connessione alla porta di rete deve essere chiusa.

CAPACITÀ E PROBLEMI DEL MODELLO CLIENT/SERVER

Il modello client/server offre un meccanismo affidabile per inviare dati sulla rete in modo affidabile. Tuttavia, oltre ad una maggiore affidabilità, il modello client/server aggiunge complessità al modello di broadcast. Tale complessità è dovuta alla quantità extra di comunicazione fra il client ed il server per verificare che i messaggi siano ricevuti. Questa quantità extra di comunicazione fa sì che il modello client/server sia più lento.

B. IMPLEMENTAZIONE DEL MODELLO CLIENT/SERVER

Utilizzate il VI TCP/IP e le funzioni situate sulla palette TCP per implementare il modello client/server usando LabVIEW come applicazione client o server.

VI E FUNZIONI TCP/IP

I VI TCP/IP sono organizzati analogamente ad altre funzioni LabVIEW, dove è necessario in primo luogo aprire, quindi leggere/scrivere e chiudere. I VI TCP/IP LabVIEW hanno altra funzionalità che è specifica alla manutenzione di una connessione di rete TCP/IP come il VI TCP e le funzioni IP to String e String to IP.

FUNZIONE TCP OPEN CONNECTION (CLIENT)

Utilizzate la funzione TCP Open Connection per stabilire attivamente una connessione con uno specifico indirizzo e porta. Se la connessione riesce, la funzione restituisce un refnum della connessione di rete che identifica in modo unico quella connessione. Usate questo refnum di connessione per fare riferimento alla connessione nelle successive chiamate del VI.
Se la connessione non è stabilita nel periodo di timeout specificato, la funzione termina e restituisce un errore. Connection ID è un refnum della connessione di rete che identifica in modo univoco la connessione TCP. I cluster error in ed error out descrivono le condizioni di errore.
L'indirizzo identifica un computer sulla rete e può essere espresso in notazione IP decimale puntata o come hostname. La porta è un numero addizionale che identifica un canale di comunicazione sul computer che il server usa per restare in ascolto di richieste di comunicazione. Quando create un server TCP, dovete specificare la porta che volete fare utilizzare dal server per la comunicazione. Se la connessione ha successo, la funzione TCP Open Connection restituisce un connection ID che identifica in modo univoco la connessione stessa. Utilizzate questo connection ID per fare riferimento alla connessione nelle successive chiamate dei VI.

FUNZIONE TCP READ (CLIENT E SERVER)

La funzione TCP Read legge un certo numero di byte da una connessione di rete TCP, restituendo i risultati in data out. mode indica il comportamento dell'operazione di lettura in base alle quattro opzioni seguenti.
Standard (default) – Attende finché tutti i byte richiesti sono arrivati o finché il tempo specificato in timeout ms si esaurisce. Restituisce il numero di byte che sono stati ricevuti fino a quel momento. Se arrivano meno byte del numero richiesto, restituisce il numero parziale di byte e riporta un errore di timeout.
Buffered - Attende finché tutti i byte richiesti sono arrivati o finché il tempo si esaurisce. Se arrivano meno byte del numero richiesto, non restituisce alcun byte e riporta un errore di timeout.
CRLF – Attende fino alla ricezione di un CR (carriage return) seguito da un LF (line feed) entro il numero di byte richiesti o finché il tempo si esaurisce. Restituisce i byte ricevuti fino a CR e LF compresi. Se non vengono trovati un CR e LF, non restituisce alcun byte e riporta un errore di timeout..
Immediate - Attende finché viene ricevuto un byte qualsiasi. Attende il timeout completo solo se non è stato ricevuto alcun byte. Restituisce il numero di byte che sono stati ricevuti fino a quel momento. Se non viene ricevuto alcun byte, riporta un errore di timeout.

FUNZIONE TCP WRITE (CLIENT E SERVER)

La funzione TCP Write scrive dati su una connessione di rete TCP. Tutti i dati scritti o letti sono nel tipo dati stringa. Il protocollo TCP/IP non valuta il tipo o formato dei dati trasferiti, quindi il tipo stringa è il metodo più flessibile. Potete utilizzare le funzioni Type Cast e Flatten to String per inviare tipi dati binari o complessi.
Se inviate tipi dati binari o complessi, dovete informare il ricevente circa gli esatti tipo e rappresentazione dei dati inviati per poter ricostruire l'informazione originale. Inoltre, quando usate la funzione TCP Read, dovete specificare il numero di byte da leggere. Un metodo comune è quello di inviare prima un intero a 32 bit per specificare la lunghezza della stringa di dati che segue. I VI TCP di esempio forniti con LabVIEW offrono maggiori informazioni su questi argomenti e su come i dati sono tipicamente formattati per comunicazioni TCP/IP.

FUNZIONE TCP CLOSE CONNECTION (CLIENT E SERVER)

Usate la funzione TCP Close Connection per chiudere la connessione verso l'applicazione remota. Se rimangono dei dati non letti e la connessione si chiude, potreste perdere dei dati. Usate un protocollo di livello più elevato per fare in modo che il vostro computer stabilisca quando chiudere la connessione.

VI TCP LISTEN, FUNZIONE TCP CREATE LISTENER E FUNZIONE TCP WAIT ON LISTENER (SERVER)

Un programma di comunicazione deve avere la possibilità di aspettare una connessione in arrivo. La procedura è quella di creare un listener (ascoltatore) ed aspettare una connessione TCP accettata su una porta specificata. Se la connessione ha successo, la funzione restituisce un connection ID e l'indirizzo e la porta del TCP remoto. Fate riferimento al VI Data Server situato in examples\comm\TCP.llb per un esempio di questa architettura.

STRING TO IP

La funzione String to IP converte una stringa in un indirizzo di rete IP o in un array di indirizzi di rete IP. Se String to IP è nel modo single output, il net address è il primo risultato restituito dal resolver del sistema operativo. Se String to IP è nel modo multiple output, il risultato è un array di tutti gli indirizzi di rete IP restituiti dal resolver del sistema operativo.
Se il nodo non riesce a convertire la stringa, il risultato è un valore zero nel modo single o un array vuoto nel modo multiple output.

IP TO STRING

La funzione IP to String converte un indirizzo di rete IP in una stringa.