NATIONAL INSTRUMENTS™

Application Note 160

Using LabVIEW™ with TCP/IP and UDP

Introduction

Internet Protocol (IP), User Datagram Protocol (UDP), and Transmission Control Protocol (TCP) are basic tools for network communication. The name TCP/IP comes from two of the best-known protocols of the Internet protocol suite, the Transmission Control Protocol and the Internet Protocol.

You can use TCP/IP to communicate over single networks or interconnected networks. The individual networks can be separated by large geographical distances. TCP/IP routes data from one network or Internet-connected computer to another. Because TCP/IP is available on most computers, it can transfer information among diverse systems.

LabVIEW and TCP/IP

You can use the TCP/IP protocols with LabVIEW on all platforms. LabVIEW includes TCP and UDP VIs and functions you can use to create client or server VIs.

IP

IP performs the low-level service of moving data between computers. IP packages data into components called datagrams. A datagram contains the data and a header that indicates the source and destination addresses. IP determines the correct path for the datagram to take across the network or Internet and sends the data to the specified destination.

IP cannot guarantee delivery. In fact, IP might deliver a single datagram more than once if the datagram is duplicated in transmission. Programs rarely use IP but use TCP or UDP instead.

UDP

UDP provides simple, low-level communication among processes on computers. Processes communicate by sending datagrams to a destination computer or port. A port is the location where you send data. IP handles the computer-to-computer delivery. After the datagram reaches the destination computer, UDP moves the datagram to its destination port. If the destination port is not open, UDP discards the datagram. UDP shares the same delivery problems of IP.

Use UDP in applications in which reliability is not critical. For example, an application might transmit informative data to a destination frequently enough that a few lost segments of data are not problematic.

342028C-01

March 2004

Using UDP in LabVIEW

Because UDP is not a connection-based protocol such as TCP, you do not need to establish a connection with a destination before you send or receive data. Instead, you specify the destination for the data when you send each datagram. Operating systems do not report transmission errors.

Use the UDP Open function to open a UDP socket on a port. The number of simultaneously open UDP ports depends on the operating system. The UDP Open function returns a network connection refnum that uniquely identifies the UDP socket. Use this connection refnum to refer to this socket in subsequent VI calls.

Use the UDP Write function to send data to a destination, and use the UDP Read function to read that data. Each write operation requires a destination address and port. Each read operation contains the source address and port. UDP preserves the datagram bytes that you specified for each command you send.

In theory, you can send datagrams of any size. However, you typically would not use UDP to send large datagrams because it is not as reliable as TCP.

When you finish all communications on a port, use the UDP Close function to free system resources.

UDP Multicast

You can use the UDP functions to communicate to a single client (single-cast) or to all computers on the subnet through a broadcast. If you want to communicate to multiple specific computers, you must configure the UDP functions to iterate through a list of clients. Using this technique creates duplicate network traffic because LabVIEW sends a separate copy of the data to each client and maintains a list of clients interested in receiving the data.

Use multicasting to communicate between a single sender and multiple clients on a network without requiring the sender to maintain a list of clients or send multiple copies of the data to each client. To receive data broadcast by a multicast sender, all clients must join a multicast group. The sender does not have to join a group to send data. The sender specifies a multicast IP address, which defines a multicast group. Multicast IP addresses are in the `224.0.0.0` to `239.255.255.255` range. When a client wants to join a multicast group, it subscribes to the multicast IP address of the group. After subscribing to a multicast group, the client receives data sent to the multicast IP address.

To multicast in LabVIEW, use the UDP Multicast Open VI to open connections capable of reading, writing, or reading and writing UDP data. Specify the time-to-live (TTL) for writing data, the multicast address for reading data, and the multicast port number for reading and writing data. The default TTL is 1, which means LabVIEW sends the datagram only to the local subnet. When a router receives a multicast datagram, it decrements the datagram TTL. If the TTL is greater than 1, the router forwards the datagram to other routers. The following table lists what action occurs to a multicast datagram when you specify a value for the **time-to-live** parameter.

`0`	Datagram remains on the host computer.
`1`	Datagram sent to every client on the same local subnet that subscribes to that IP address. Hubs/repeaters and bridges/switches forward the datagram. Routers do not forward the datagram if the TTL is `1`.

If you specify a value greater than 1, the datagram is sent and routers forward it through TTL-1 layers.

Refer to the UDP Multicast Receiver VI and the UDP Multicast Sender VI in the labview\examples\comm\UDP.llb for examples of using UDP multicasting.

TCP

TCP ensures reliable transmission across networks, delivering data in sequence without errors, loss, or duplication. TCP retransmits the datagram until it receives an acknowledgment.

System Requirements

Before using TCP/IP, confirm that you have the necessary requirements, which vary depending on the platform you use.

- **(Windows and UNIX)** TCP/IP is built in. You do not need to use a third-party product to communicate using TCP/IP. If your network is configured properly, LabVIEW requires no additional setup.
- **(Mac OS)** LabVIEW networking requires Open Transport, included in Mac OS 7.5 and later.

Using TCP in LabVIEW

TCP is a connection-based protocol, which means that sites must establish a connection before transferring data. TCP permits multiple, simultaneous connections.

You initiate a connection by waiting for an incoming connection or by actively seeking a connection with a specified address. In establishing TCP connections, you have to specify the address and a port at that address. A number between 0 and 65,535 represents a port. UNIX reserves port numbers less than 1,024 for privileged applications. Different ports at a given address identify different services at that address.

Use the TCP Open Connection function to actively establish a connection with a specific address and port. If the connection is successful, the function returns a network connection refnum that uniquely identifies that connection. Use this connection refnum to refer to the connection in subsequent VI calls.

You can use the following techniques to wait for an incoming connection:

- Use the TCP Listen VI to create a listener and wait for an accepted TCP connection at a specified port. If the connection is successful, the VI returns a connection refnum, the address, and the port of the remote TCP client.
- Use the TCP Create Listener function to create a listener and use the TCP Wait on Listener function to listen for and accept new connections. The TCP Wait on Listener function returns the same listener ID you wired to the function and the connection refnum for a connection. When you finish waiting for new connections, use the TCP Close Connection function to close a listener. You cannot read from or write to a listener.

The advantage of using the second technique is that you can use the TCP Close Connection function to cancel a listen operation, which is useful when you want to listen for a connection without using a timeout, but you want to cancel the listen when another condition becomes true. You can close the listen VI at any time.

When you establish a connection, use the TCP Read function and the TCP Write function to read and write data to the remote application.

Use the TCP Close Connection function to close the connection to the remote application. If unread data remains and the connection closes, you might lose data. Use a higher level protocol for your computer to determine when to close the connection. After a connection is closed, you cannot read from it or write to it again.

Deciding between TCP and UDP

TCP is the best protocol to use if you want reliable data transmission. UDP is a connectionless protocol with higher performance, but it does not ensure reliable data transmission.

Creating a TCP Client

Note Refer to the *LabVIEW Help* for the most recent version of these instructions and details for each of the functions.

Complete the following steps to create a TCP client using the TCP functions.

1. Use the TCP Open Connection function to open a connection to a server. You must specify the Internet address of the server and the port for the server.

   The **address** identifies a computer on the network. The **remote port** identifies a communication channel on the computer that the server uses to listen for communication requests. When you create a TCP server, you specify the port that you want the server to use for communication.
2. Use the TCP Write function to send a message to a server.
3. Use the TCP Read function to read a message from the server. You must specify the number of bytes you want to read.
4. Use the TCP Close Connection function to close the connection to the server.

Refer to the Simple Data Client VI in the `labview\examples\comm\TCP.llb` for an example of a TCP client.

Timeouts and Errors

When you design a network application, consider carefully what should happen if something fails. For example, if the server crashes, determine how each client VI handles it.

One solution is to make sure that each client VI has a timeout. If something fails to produce results after a certain amount of time, the client continues execution. In continuing, the client can try to reestablish the connection or report the error. If necessary, the client VI can shut down the application.

Creating a TCP Server

Note Refer to the *LabVIEW Help* for the most recent version of these instructions and details for each of the functions.

Complete the following steps to create a TCP server using the TCP functions.

1. Use the TCP Listen VI to wait for a connection. You must specify the **port**. This **port** must be the same port that the client attempts to access.
2. If a connection is established, use the TCP Read function to read from that port to retrieve a message.
3. Use the TCP Write function to return results. The data must be in a form that the client can accept.
4. Use the TCP Close Connection function to close the connection.

Refer to the Simple Data Server VI in the `labview\examples\comm\TCP.llb` for an example of a TCP server.

TCP and UDP Examples

Refer to the `labview\examples\comm\TCP.llb` and the `labview\examples\comm\UDP.llb` for examples of using the TCP and UDP VIs and functions.

342028C-01 Mar04