The tcp protocol is a full-duplex connection protocol, allowing traffic to flow in both directions at the same time. When a connection is established between a pc and a reader, two actual connections, or circuits, are made. One connection controls the traffic outbound to the reader, and the other controls traffic inbound from the reader. Should either of these connections be severed, data flow between the reader and the pc will be interrupted and it may appear that the reader is offline when in fact it is not.

Many things can cause the connection to be severed, such as a switch disconnect, power fail, or abnormal program termination on the pc end. It is therefore important to implement code in your software which can detect a severed connection so that your code can gracefully reconnect to the reader in the event of a lost connection.

The IBC readers, when acting online in the server mode, normally listen for a connection on port 57 by default. You establish a connection with the reader by opening up a socket in your software, and attempting a connection to port 57 on the reader. Once a connection is established, communications to and from the reader can occur. This will continue indefinately until one of the sides (the reader or pc) closes the connection, or the connection is somehow severed. It is important to note that the reader cannot transmit scanned data to the pc if there is no current connection. Therefore, if a connection is not established, or a connection has been severed, data scanned at the reader is lost, since there is nowhere to send it. If your program is running on the pc but you are not receiving reader data, then either there is no connection, or the connection has been lost or severed.

Note that with tcp, it is possible for one side (the reader or the pc) to lose the connection to the other side while the other side believes the connection is active. It is also possible for the connection to be severed in only one direction, leaving the other direction active. This results in a broken connection where it appears that the reader is not functional but in fact it is the tcp connection that is not functional at this point. Identifying these conditions is not easy but with some knowledge of how tcp works and proper software design most of these conditions can be detectable and dealt with.

All IBC readers utillize a tcp feature called a tcp keep-alive. The keep-alive mechanism tests the connection once every minute by sending a keep-alive message to the tcp stack on the pc. If the connection is alive, the pc responds back. If the reader does not receive a response, the reader disconnects the tcp connection and waits for the pc software to re-connect to the reader. This allows the reader to detect severed connections in most cases and recover from it.

It is suggested that you implement the same feature in your software code, to be able to detect connections which are severed. Windows sockets allow you to implement a keep-alive for the sockets you create in your software. The keep alive feature can be enabled with the setsockopt system call, which can easily be called from VB or C. You can find a lot of information on the net for coding the keep-alive functionality. Microsoft also has considerable information on this subject in their developers network area.

Another way to detect the state of the reader and socket is to simply send a version command to the reader periodically. If a response is not received from the Version command, then the connection may be severed and you will have to reconnect to the reader. Gracefully shut down the socket at the pc end, and then reconnect.

Note that should you detect a severed connection in your software, and attempt to reconnect to the reader, that the reader may not allow the connection because if has not detected the broken connection yet, since the reader keep-alive has a granularity of one minute. It could therefore take up to 1 minute before you are able to connect to the reader again.

160 Oak Street, Glastonbury, CT 06033 USA Tel: (860) 659-9660 Fax: (860) 657-3860
http://interbar.com sales@interbar.com

If the reader has not detected the broken connection, and you attempt to close the socket from your end, you may experience the same result, since the reader may not ever see the close request since the connection is severed.

This situation is easy to detect and correct. If you are able to ping the reader, or connect to the control port, then this tells you that the reader is functional, but the data port connection has been severed and is currently in the "unknown" state. The ping command does not use the tcp protocol, so a ping command should always be able to see the reader, and if you do not already have the control port open, you should always be able to connect to the control port.

The way to handle this through software, is to connect to the control port and issue the reboot command. This causes all connections on the reader side to close immediately, and will reset the data port making a connection to the reader possible again. Note that although this will effectively reset the reader side, if the pc side believes the connection is still alive, this technique may not work by itself. The advisable thing to do if this is the situation is to simply force a closure of the current socket, and then connect to the control port and issue the reboot command. You should take a look at the options available with the setsockopt api, especially the "linger" option, to gain an understanding of how windows treats sockets that are "lingering" around, and how to gracefully close these sockets if you need to.

There are cases where sockets can get into funky states that are sometimes hard to detect. There are times where the pc may tell you that a socket is closed, but it may not be fully closed. It may only be in the process of closing. This is because there are actually 2 close states for tcp connections because both sides of the connection need to cooperate for a graceful close. These are known as fin_wait and fin_wait2 states. You should also understand these states as well so you know how to deal with them. Also, under certain circumstances, It is possible for a keep-alive itself to force the other side of the connection to "reset" and close down after a number of retransmission attempts.

We also reccommend utillizing "non-blocking" calls in your software as opposed to "blocking" calls. Non-blocking calls allow you to continue on with your program execution while you are waiting for tcp data to be received. While you are waiting you should be checking for tcp incoming data and also checking the status of the connection so you can detect if the socket is gone. If you utillize blocking calls it is possible you may not be able to detect a severed connection on the other circuit.

To ensure that you have an uninterrupted flow of data between the reader and the pc, we suggest that you implement as many of the following steps which are possible for your coding environment:

- Implement the keep-alive feature if possible
- Use only non-blocking calls
- Periodically test the connection by sending a version command to the reader and wait for a response

Taking these steps will help you catch most disconnects and severed circuits quickly and allow you to recover from them gracefully.

We also reccommend that you simulate broken connections when testing your software to ensure that you can recover from these types of conditions should they occur.