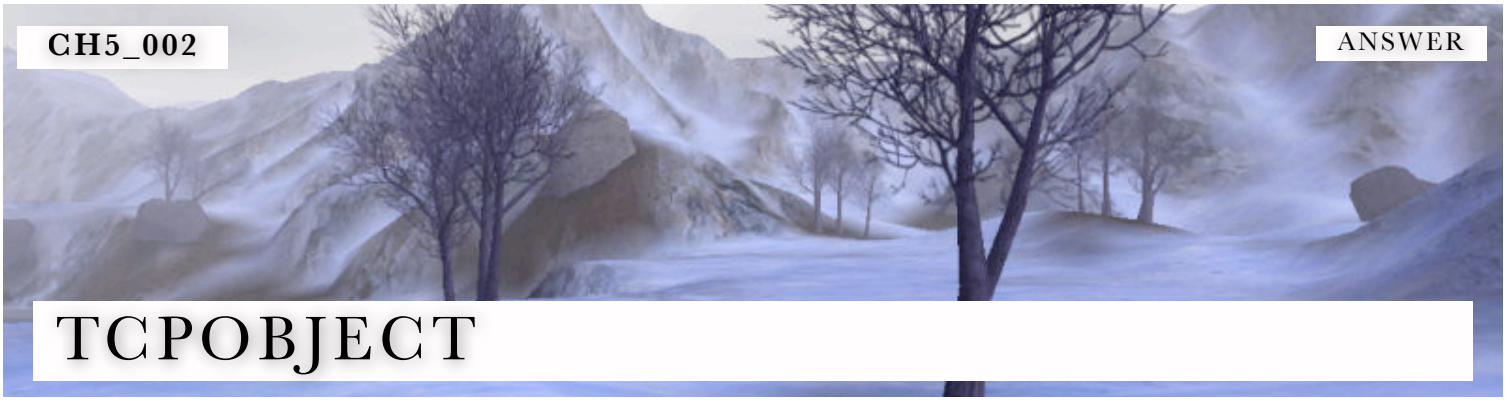

2 TCP0* - Connecting

Finished Code:

```
function TCPServer::onConnectRequest( %Obj , %addrBuf , %idBuf )
{
   echo("TCPServer::onConnectRequest( " @ %Obj @ " , " @ %addrBuf @ " , "
                                      @ %idBuf @ " ) " );

   // Accept the request and create a new proxy connection object
   $lastTCPObjectProxy = new TCPObject( TCPProxyConnection , %idBuf );

   TCPProxyGroup.add($lastTCPObjectProxy);
}

function TCP0()
{
   cleanUp(); // PLEASE KEEP

   %server = new TCPObject( TCPServer );

   TCPServerGroup.add( %server );

   TCPServer.listen( 5000 );

   %client = new TCPObject( TCPClient );

   TCPClientGroup.add( %client );

   TCPClient.connect("IP:127.0.0.1:5000");
}
```

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

3 TCP1* - Client to Proxy Messaging

Finished Code:

```
////
//              TCP1 - Simple Client to Proxy Message (immediate)
////
function TCP1()
{
   TCP0();

   TCPClient.send("Client to Proxy -> Message 0\n"); // NEVER SENT
}

////
//              TCP1a - Simple Client to Proxy Message (delayed)
////
function TCP1a()
{
   TCP0();

   TCPClient.send("Client to Proxy -> Message 0\n"); // GETS LOST
   TCPClient.schedule( 1000 , send ,
                       "Client to Proxy -> Message 1\n"); //OK
}

////
//              TCP1b - Simple Client to Proxy Message (delayed, multiple sends)
////
function TCP1b()
{
   TCP0();

   TCPClient.schedule( 1000 , send , "Client to Proxy -> Message 0\n");
   TCPClient.schedule( 1000 , send , "Client to Proxy -> Message 1\n");
   TCPClient.schedule( 1000 , send , "Client to Proxy -> Message 2\n");
   TCPClient.schedule( 1000 , send , "Client to Proxy -> Message 3\n");
}
```

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

4 TCP2* - Proxy to Client Messaging

Finished Code:

```
////
//              TCP2 - Simple Proxy to Client Message (works immediately)
////
function TCP2()
{
   TCP0();

   ($lastTCPObjectProxy).send("Proxy to Client -> Message 0\n");
}

////
//              TCP2 - Simple Proxy to Client Message (works immediately)
////
function TCP2a()
{
   TCP0();

   schedule( 1000 , 0 , delayedProxySend );
}

function delayedProxySend()
{
   $lastTCPObjectProxy.send("Proxy to Client -> Message 0\n");
}
```

Answers:

1. We got an error because the connection was not yet established when we attempted to send data. We needed to wait for the onConnected() callback to be called first.

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

4 TCP3* - Affect Of newline In Messaging

Finished Code:

```
////
//              TCP3 - Client to Proxy Messages (effect of newline)
////
function TCP3()
{
   TCP0();

   TCPClient.schedule( 1000 , send ,
                       "Client to Proxy -> Message 0 (no newline); ");
}

////
//              TCP3a - Client to Proxy Messages (effect of newline)
////
function TCP3a()
{
   TCP0();

   TCPClient.schedule( 1000 , send ,
                       "Client to Proxy -> Message 0 (no newline); ");

   TCPClient.schedule( 2000 , send ,
                       "Client to Proxy -> Message 1 (w/ newline)\n");
}
```

Answers:

1. It buffers the message, but doesn't send it because there is no newline termination on the message.
2. No, the data is just buffered.

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

5 TCP4* - Disconnecting and Buffered Messages

Finished Code:

```
////
//              TCP4 - Client to Proxy Messages (effect of disconnect)
////
function TCP4()
{
   TCP0();

   TCPClient.schedule( 1000 , send ,
                       "Client to Proxy -> Message 0 (no newline); ");

   TCPClient.schedule( 1000 , disconnect );
}

////
//              TCP4a - Client to Proxy Messages (effect of disconnect on other agent's
messages)
////
function TCP4a()
{
   TCP0();

   TCPClient.schedule( 1000 , send ,
                       "Client to Proxy -> Message 0 (no newline); ");

   schedule( 1500 , 0 , delayedProxyDisconnect );
}
function delayedProxyDisconnect()
{
   ($lastTCPObjectProxy).disconnect();
}
```

Answers:

1. We called disconnect, which automatically flushed the connection, sending the data.
2. Disconnecting on the opposite end of a connection doesn't cause the opposing end to flush. Only the end of the connection that disconnect() is called on will do the flush.

Copyright GarageGames.com 2000 - 2008 All Rights Reserved

6 TCP5* - Connecting (localhost Lookup)

Finished Code:

```
////
//              TCP5- Simple Client to Proxy Server (using localhost)
////
function TCP5()
{
   cleanUp(); // PLEASE KEEP

   %server = new TCPObject( TCPServer );

   TCPServerGroup.add( %server );

   TCPServer.listen( 5000 );

   %client = new TCPObject( TCPClient );

   TCPClientGroup.add( %client );

   TCPClient.connect("localhost:5000");

   TCPClient.schedule( 1000 , send ,
                       "Client to Proxy -> localhost works too!\n");
}
```

Copyright GarageGames.com 2000 - 2008 All Rights Reserved