APPLICATION NOTE

Network Communication/Performance

Purpose

The purpose of this application note is to highlight the importance of tuning software to lower latency in networks used for control. Several tuning techniques are described.

Introduction

Network performance is the quality of service of a communications link as seen by the user. Several parameters including throughput and latency can be used to measure the network performance. For application utilizing a network for control it may be advantageous to lower communication latency, sacrificing overall throughput. Several techniques for tuning a system to lower latency will be presented.

Do I need to tune my system to lower latency?

It is important to note that in most cases MagneMotion's 'fire-and-forget' command strategy is sufficient. However, latency can become an issue if timing constraints are tight, or large amounts of data are sent simultaneously. Some examples include:

- Tight control of vehicle movement requires a command to reach the motor controlling a vehicle very quickly.
- Tight control of vehicle movement requires very little time between when a vehicle arrives at its destination and a command complete message is delivered to the host process.
- The response from a query for all status (High Level Controller, Node Controller, Motor, and Vehicle) on a large system can back up Windows receive buffers.

Introduction to Network Performance

Network performance is the quality of service of a communications link as seen by the user. This quality is measured by one or more of the following performance measurements:

Bandwidth	Bandwidth is the measure of the maximum data rate that can be transferred. It is generally conveyed in terms of bits/second from the physical point of view, and does not include the packet framing or the normal overhead associated with coordinating data across the channel.
Throughput	Throughput is the actual amount of data transferred over time.
Latency	Latency is the delay that occurs from the sending of data to the receiver processing it.
Jitter	Jitter is the variation in the time between packets arriving.
Error Rate	Error rate is the number of corrupted bits expressed as a percentage or fraction of the total sent.

Hardware devices which are connected to networks are generally tuned to meet the basic bandwidth needs of the network. Software programs using networks, however, must then be tuned for operation based on the requirements of communication with the far end device.

For example, downloading a streaming video and a rocket launch abort signal have very different requirements in terms of how they use the network.

A streaming video application is concerned with overall throughput. This application will try to package up as much data as possible, and send it in bulk across the network to its destination. To achieve this, the source computer may implement “buffering” where it waits to send data until it has collected enough to create a full packet. The advantages of this method are high throughput, which results in better overall usage of the bandwidth available. The disadvantages are that is has a higher latency, potentially higher jitter, and slower network response.

In contrast, a rocket launch abort signal must be delivered as fast as possible. In this case, the packet is sent immediately without buffering or waiting. The network throughput is very low, but is not a concern. The sole purpose is to deliver the packet across the network with the lowest latency possible.

Using a Network for Control

When writing applications that use a network for controlling devices, consideration should be centered on packet latency rather than throughput. The determining factor should be how much packet delay the application can tolerate before the packet is considered “late”.

For example, the Windows operating system TCP Socket call to recv(…) will “block” the calling thread and buffer data until it has received enough to satisfy the request or 200ms elapses. If the control application cannot tolerate a 200ms delay, then the socket must be turned to a lower timeout, lower buffer limits, or the application must be redesigned for “non-blocking” operation.

Tuning Techniques to Lower Latency

The following are some application and Windows socket tuning techniques that can be used to lower communication latency.

Application Thread vs Separate Communication Thread

In general terms, a separate lightweight communication thread can have better responsiveness than slaving the communications within the main application loop. This is especially true if the application has a graphical interface that is servicing paint events, and software timers. They delays incurred by repaint operations can take hundreds of milliseconds to complete, or seconds depending on the complexity. Those delays incurred by the application overhead increase the jitter, which can be unacceptable.

Changing the Application Timebase

The default time base granularity on Windows for all applications is 15.6ms. This includes all timeouts including mutex, semaphores and IO. For most applications, this granularity is sufficient, but for some low latency operations it is advantageous to increase this frequency.

Windows provides the ability to change the application time base by using the timeBeginPeriod(…) and timeEndPeriod(…) APIs. Using these functions changes the internal scheduler granularity in the operating system by increasing the rate at which it is called. The disadvantage of this technique is that it will increase the internal scheduler for the operating system itself. That results in the scheduler running more often, consuming more CPU and more power to process all system processes. The time period selected should be no lower than what is needed to meet the application timing.

```
timeBeginPeriod(5);
ProcessThread();
timeEndPeriod(5);
```

Typically these operations wrap the main thread loop, but they can be used to control the time base for the entire application.

Blocking vs NonBlocking

On the Windows operating system, an un-tuned TCP Socket will delay up to 200ms waiting for data to arrive before it returns to the caller. A socket set to non-blocking will return immediately with or without data available.

```
u_long iMode=1;
```

```
Result = ioctlsocket(Socket, FIONBIO, &iMode);
```

Reading data from a non-blocking socket in the Windows Operating system is shown below. Note that the return value will indicate "WSAEWOULDBLOCK" if no data is available

```
Result = recv(Socket, &data, 100, 0);
if(Result > 0)
{
   // data ready
}
else if(Result == 0)
{
   // Connection closed
}
else
{
   if(Result != WSAEWOULDBLOCK)
   {
      // some kind of real error
   }
}
```

Set Receive Timeout

If a blocking socket is needed for the application, then lowering the "receive blocking timeout" can help reduce additional latency.

```
DWORD Timeout = 5;
setsockopt(Socket, SOL_SOCKET, SO_RCVTIMEO,
    (char *)&Timeout, sizeof(Timeout));
```

Disabling the TCP "Nagle" Algorithm

The "Nagle" Algorithm is named after John Nagle, who noted that improved efficiency can be achieved by buffering data inside the TCP/IP stack to improve network usage. It was designed to prevent the sending of small packets on the network to increase throughput at the cost of more latency for each packet.

For many low latency applications, disabling this will lower the latency for messages crossing the network, but because more small packets will be sent, the overhead to process them will

increase. Disabling this will have a negative effect on the throughput of small network packets, but ensures they are delivered without delay.

```
BOOL NoDelay = 1;
setsockopt(Socket, IPPROTO_TCP, TCP_NODELAY,
(char *)&NoDelay, sizeof(NoDelay));
```

Summary

In most cases MagneMotion's 'fire-and-forget' command strategy is sufficient, though latency can become an issue due to tight timing constraints or large amounts of data being sent simultaneously. In cases where this becomes an issue, several tuning techniques have been presented to lower communication latency.

More Information

MagneMotion Website: www.magnemotion.com

Questions & Comments: www.magnemotion.com/about-magnemotion/contact.cfm

Revision History

Rev.	Change Description
A	Initial release