



## Application Note #5449

---

### The Suitability of Ethernet for Motion Control

---

This application note explains why TCP/IP over Ethernet is often the best choice for multi-axis motion control. Appendix A details the differences between three popular serial protocols; TCP/IP, a typical Fieldbus network, and simple RS-232.

#### ***The Assurance of Good Data***

Systems engineers often need to select the appropriate network for optimum performance of their motion control system. Quite often, the two competing protocols are TCP/IP over Ethernet or a Fieldbus architecture. Examples of Fieldbus systems include CanOPEN, SERCOS, or DeviceNet.

Ethernet protocol provides the engineer with a flexible, scalable network, where large packet size and collision recovery allow intelligent devices to communicate seamlessly with one another. If a system includes multi-axis controllers, the network traffic is minimized by relying on the time-critical servo loop to occur inside the controller. Since host data is sent less frequently than 1 millisecond (as is usually the case with intelligent, multi-axis controllers), the non-determinism associated with Ethernet is not an issue. Also, relatively large packets of data, like a text message to an operator interface, can easily be contained within a single packet, cutting back on the overhead necessary to route this information.

Fieldbus networks rely on small, deterministic packets of data to be sent between a single master and multiple slaves. This method of data transfer works very well when the information is repetitive, short, and required at regular intervals. An example of a typical Fieldbus network is a master reading the value of a switch, or streaming of contour data to a single axis motion controller. All data is limited to the protocol's format. To transmit large text strings, the information must be disseminated into multiple packets to avoid interfering with the real-time operation of the motion controllers. This makes it difficult to use Fieldbus for coordinated multi-axis control.

#### ***Detailed distinctions between network protocols***

Fieldbus systems and TCP/IP over Ethernet have many similarities, yet the methods of data verification used by these protocols differ. A fundamental difference is the client/server relationships of the protocols. Fieldbus architecture relies on multiple servers all communicating directly to one client. Servers are not capable of speaking to one another. With this relationship, only the client can request data, and the servers respond in a hierarchal manner. In contrast, Ethernet networks can consist of multiple clients communicating to one or more

servers. In addition, a server can also simultaneously be a client. This decentralized method of networking can lead to special cases where two or more devices attempt to use the network wiring at exactly the same time. This phenomenon is called a *collision*.

When a TCP/IP network recognizes a collision, it alerts the senders of the offending packets, which use a random number generator to specify a delay before resending. If, by wild coincidence, the servers collide again, or one packet collides with a third packet, then a new random number is generated, numerous times before the packet is considered 'failed' Depending on network traffic, this might occur on the order of once every 1000 years. The speed at which this collision recovery is performed is dependant on the network. For 10Base-T networks, the resend occurs approximately every ½ millisecond. For 100Base-T networks, this recovery occurs faster than 80 microseconds. Based on the extreme improbability that a packet would collide more than once, it can be safely stated that 10Base-T TCP/IP networks are non-deterministic by 0.5 milliseconds. It should be noted that Fieldbus networks can't have collisions because of the single-client architecture.

Another network phenomenon is the destruction or corruption of packetized data. In this case, both Fieldbus and TCP rely on checksums to assure the integrity of the packet. If the packet is 'bad', the client alerts the server for a resend. The result on the network's throughput capability is identical.

As the reliability of the control system is directly linked to the capability of the network, several studies have been undertaken that show the performance of TCP/IP for the life of a machine. In "Making Ethernet Work in Real Time", Sensors Magazine, November 2000, Dr. Stan Schneider concluded that a 10Base-T network transferring 1000 128-byte packets every second (10% loading) has a probability that all packets sent over five years will arrive with a delay shorter than 8 milliseconds is 96%<sup>1</sup>.

### ***Where Determinism Does Matter***

Deterministic networks require that parcels of information arrive at specific, repeatable intervals. These packets, usually 6 to 8 bytes, typically contain regular, repetitive information. An example would be the state of a switch, or single-axis contour data. Such a system might potentially consist of a host CPU communicating to multiple intelligent motors or 'smart' drives. These axis nodes, while they do possess limited processing power, require that the CPU respond to position error by modifying the motion commands. These motor commands are sent to the nodes via the network. This arrangement needs a serial protocol that is absolutely deterministic. Any delay or missed packet will result in a glitch in the motion. A simple 1-axis system is shown in Figure 1.

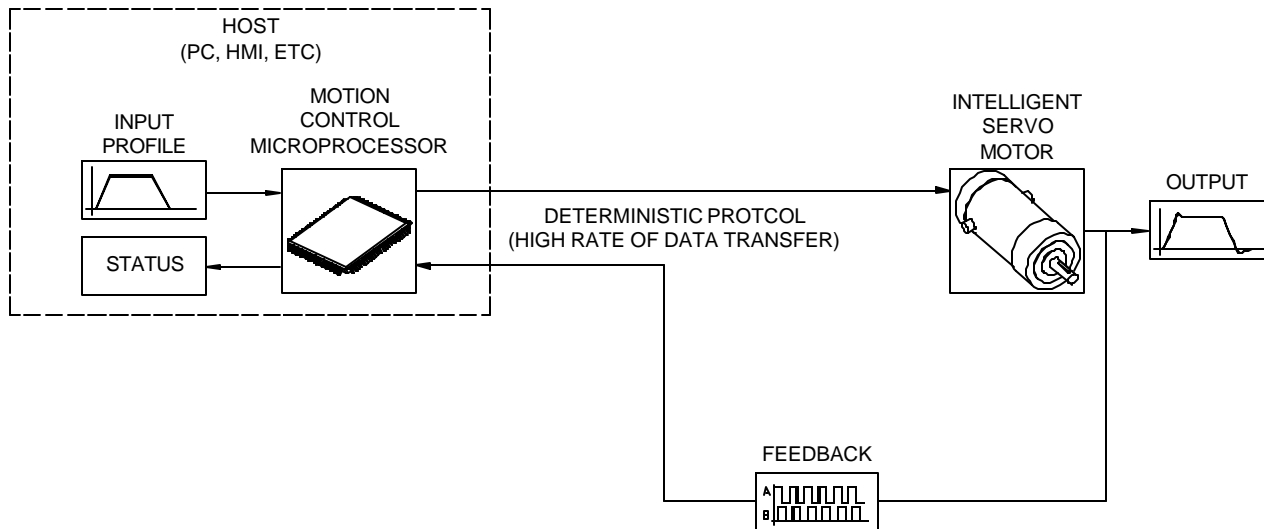


Figure 1- Typical Deterministic Architecture for Single Axis of Motion

***Why TCP/IP is the right choice for complex control networks***

Ethernet networks are the protocol of choice for the vast majority of office networks and internet traffic. A major reason for this is the variety of data being transmitted. Very rarely do two contiguous packets contain like data. Also, TCP/IP only sends packets when data is present; Fieldbus sends the data regardless of system activity. With packets easily larger than 300 bytes, TCP/IP is the right choice for message strings, intricate commands, or multi-byte status records. Fieldbus networks tend to have a difficult time with these higher-level communiqés.

As an example, consider a Galil single-axis controller connected to a host PC. Once a program or motion profile has been generated or executed, the controller does not require any more external communication in order to complete the task. All servo loop closure and I/O handling are done locally. Figure 2 shows one axis of motion configured in just such a manner.

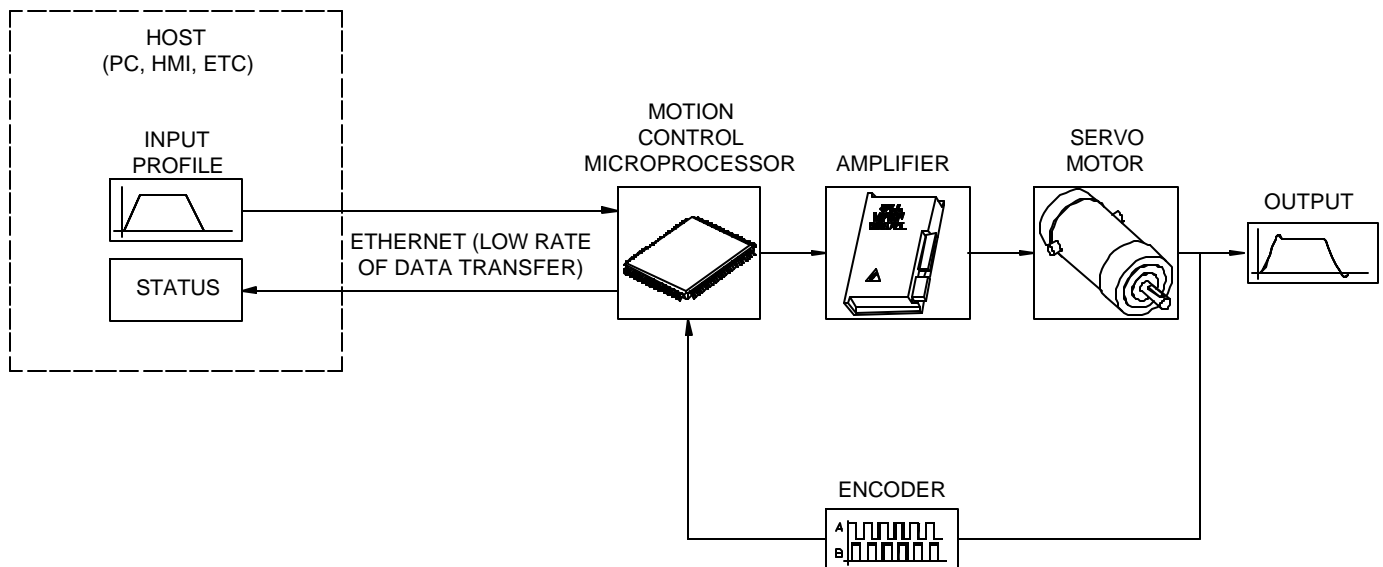


Figure 2- Typical TCP/IP Architecture for Single Axis of Motion

By relying on the local Galil intelligence to perform the local tasks, the user can realize significant benefits:

1. Non-determinism is a non-issue. Because all the time-critical actions like servo loop closure are done locally, there is no need for time-stamped data from a master in order to achieve quality motion. Any high-level command from the host would be on the order of 'Begin application program' or 'Update HMI with system parameters' which can arrive in a non-deterministic manner.
2. Network traffic. By limiting network traffic to 'Execute', 'Halt' or 'Update Status' commands, all those loop closure packets disappear. A programmer can easily see a tenfold decrease in network data transfer. This has the added benefit of nearly eliminating the possibility of collisions, making determinism all but inconsequential.
3. Multi-axis control. If a systems engineer has multiple axes near one another, then to communicate with a single device, namely a multi-axis Galil controller, has obvious benefits over communicating with discrete single-axis nodes. These discrete devices have no knowledge of the other nodes, unless explicitly told by the host. Again, a huge decrease in communication overhead can be realized by allowing multiple axes of control over one network connection
4. Tight coupling of axes. Even with the best of deterministic networks, and the fastest CPU with the leanest of code, systems Engineers will have a difficult time of coupling multiple axes closely enough to each other.

Coordinate vector moves, electronic camming, or even sharing of I/O all suffer from this multiple-node model. Galil's version has the coupled axes all run by the same local microprocessor.

Because of the vast popularity of the Internet, almost all PC users are familiar with TCP/IP in some form or another. By relying on the local loop closure inherent to all Galil motion controllers, a Systems Engineer can create a robust, non-deterministic control network with off-the-shelf commercial products and a friendly, popular, open-source serial data protocol.

### **References**

<sup>1</sup> The choice of 1000 packets of 128 bytes was chosen because this is a typical Ethernet control network load. A variety of performance specifications were tested in Dr. Schneider's article. For a brief synopsis of this test, see the article "Is Ethernet Suitable for Motion Control?" Jacob Tal, ServoTrends, July 2002

Due to the popularity of Galil's standalone Ethernet controllers, a significant body of literature already exists. For more information, see Application Note #5439, found at

<http://www.galilmc.com/support/appnotes/miscellaneous/note5439.pdf>

and the following Servotrends article:

<http://www.galilmc.com/literature/servotrends/july02/ethernetbenefits.pdf>

and some related links:

[http://ethernet.industrial-networking.com/articles/i03\\_motioncontrol.asp](http://ethernet.industrial-networking.com/articles/i03_motioncontrol.asp)

<http://www.galilmc.com/literature/articles/md032201.pdf>

## **Appendix A: Compare & Contrast Between Serial Protocols**

The following pages detail three types of serial communication- RS-232, TCP/IP, and a typical Fieldbus architecture. Each method has both benefits and drawbacks; it is up to the Engineer to determine the protocol that best suits the needs of the system.

### **Serial (RS-232, RS-422)**

This method of communication was initially developed in the 1950's. RS-232 relies on the transmission and receipt of serial (usually ASCII) data along a pair of copper wires. There is no data checking, and distances between devices are limited to the power of the circuit's line drivers. RS-422 provides some noise immunity by the use of differential signals.

#### **Pro:**

1. With short transmission lines, this method of communication is nearly bulletproof. If there is perceived data corruption, a technician can test one of four signal wires to diagnose the problem.
2. Popularity- Even today, most PC's are equipped with a serial (COM) port. In addition, Windows operating systems come equipped with *Hyperterminal*, a terminal emulator program.

#### **Con:**

1. Baud Rate- Even the best RS232/422 line has limited bandwidth. Most systems are limited to 115kbaud, which can severely limit data throughput.
2. Because RS-232 is so simple, there is no intrinsic data protection. If there is induced noise on the cables, the receiver might receive garbled characters, and the sender would be none the wiser.
3. Imminent Extinction- As PC's (especially laptops) get smaller and lighter, and with the popularity of USB and FireWire, at some point commercial PCs will no longer be equipped with the familiar 9- and 25- pin D-shell COM ports.

#### **Example System:**

Any interface with legacy components (old DOS PCs, etc) that is connected to a centralized motion controller or I/O module. Any real-time processing is done local to the controller. This avoids the need to transmit large amounts of data between the host device and the controller.

## **Fieldbus (CanOPEN, DeviceNET, etc.)**

This style of communication has seen incremental advancements over the past few years. As a category, fieldbus systems can run over RS-485 networks, or sometimes in a more token-ring type of configuration. Most fieldbus architectures describe themselves as *deterministic*, meaning data always arrives at a known time, and, due to internal checksums, can be treated as valid upon receipt.

The focus of Fieldbus is to keep the 'brains' all at the center, and dole out activity to all the nodes. These nodes in turn update the master's data record, causing the master to adjust accordingly.

### **Pro:**

1. Because of determinism, the system designer can count on data arriving when expected. This is crucial if the host CPU is in charge of developing the motion profile, and sends actual motion commands across the network to nodes, which process the command, perform some task, then update the master as to the results.

### **Con:**

1. Complexity- Because of the multi-tiered data verification, there are multiple layers and protocols that even a basic user needs to learn. If errors do occur, the retransmission of data is up to the programmer, rather than performed as a matter of course.
2. Exclusivity- Once a field bus is chosen, the system designer is limited to those devices that are compatible. While there are numerous devices out there for each major fieldbus, connection to an incompatible device requires expensive and sometimes unreliable translators.

### **Example System:**

A central, high-speed CPU running a full-featured control application that needs to communicate to a variety of intelligent nodes. A network like this would consist of devices such as fieldbus I/O and integrated motor/controller packages, which close the control loop locally, but rely on real-time motion profiles to be streamed from a host.

## ***Ethernet (TCP/IP, UDP)***

TCP/IP has become the international standard for almost all internet traffic. TCP/IP relies on the packetization of data into discrete blocks, then sending this data out at regular intervals with an address header and a checksum footer. UDP is essentially TCP/IP without the overhead of client acknowledgements, making it the user's responsibility to ensure accurate packet transmission. UDP is useful when a system is in a benign environment and transmission distances are relatively short. TCP/IP is a *non-deterministic* protocol, meaning the host won't send a packet until there's enough data, and the frequency of packet transmission is tied to network traffic.

TCP/IP in a controls network relies on the sharing of intelligence. If nodes have the processing ability, it is best to close any time-critical loops internally. This cuts back on actual control network traffic, freeing the network for non-critical tasks such as axis status and part count.

### **Pro:**

1. Almost every PC sold in the world today has a RJ-45 plug to connect to a network. If a PC is network-enabled, then TCP/IP and UDP are almost certainly installed on the PC and ready to use. There is no need for unusual connectors or costly development packages. A basic Telnet session (also available in any current Windows OS) is all a programmer needs to communicate to an Ethernet device.
2. Speed of data transmission- Up to about 1998, the de facto speed of Ethernet was 10 Mbaud, denoted 10Base-T. However, as Internet traffic has increased geometrically and file transfers have gotten much larger, the 100Base-T (100 Mbaud) has become much more common. In reality, a user would see about 75-80 Mbaud of data transmission after the packetizing overhead. Compared to RS-232 at 115 kBaud, a programmer can expect a 700-fold increase in data transmission.
3. Cable length- Ethernet, and especially TCP/IP, can travel relatively vast distances using commercially available Cat 5 cable. Of course, factory noise and cable quality play a role, but 50 meters (or more) is entirely acceptable for reliable data transmission.
4. Data Integrity- The programmer is in no way forced to check the data when it is received. The internal workings of a protocol such as TCP/IP handle this entirely. All these safety features were developed to ensure accurate data through routers, miles of copper wire, possible satellite uplink, etc, as a packet makes its way from a server in LA to a user's PC in New York. For factory automation, the network is considerably simpler.
5. Multiple connections- Connecting to several devices through one cable is essentially required for today's factory automation. In this manner, one client can talk to numerous other servers, which in turn could have their own servers. This differs drastically from RS-232.

**Cons:**

1. The lack of determinism using TCP/IP can make some control schemes untenable. But, if devices are specifically chosen to keep time-critical applications local to the node, then this becomes irrelevant.
2. Because of the ubiquity of Ethernet, the responsible engineer needs to keep in mind the unsavory prospect of system intrusion by outside forces. The last thing a factory needs is for a machine to be taken over by an unscrupulous hacker in a foreign country. The simple solution is basic common sense. NEVER allow for outside access to the automation network.

**Example System:**

A simple host, be it a windows-based visual interface, a standalone HMI, or even master motion controller needs to communicate to a variety of devices from different suppliers. All motion control nodes have the intelligence to close servo loops locally, and any I/O modules process and act upon events. These intelligent slaves update the master at regular intervals, but all time-critical applications are done without network traffic.