White Paper

An overview of network serial port servers, the potential impact of communications latency, and the results of a Comtrol latency benchmark measuring the elapsed time required to transmit and receive serial data between a networked Personal Computer and an Ethernet-attached serial port server.

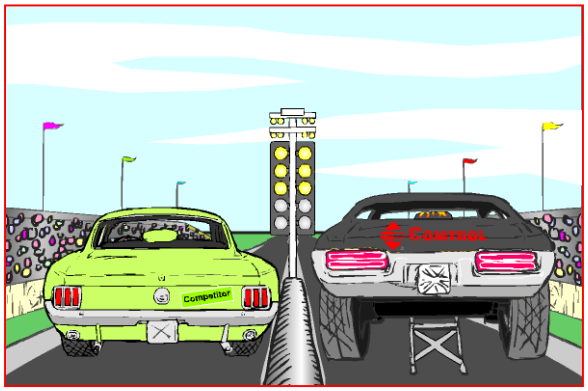
Benchmark Prepared and Executed By:
Grant Edwards, Sr. Principal Engineer, Comtrol Corporation
David Boldt , Sr. Technical Consultant, Comtrol Corporation

WELCOME …

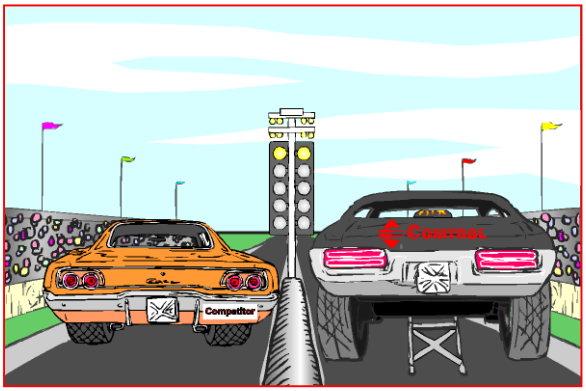
Thank you for your interest in learning more about state-of-the-art serial device connectivity.

This White Paper provides a snapshot of network serial port servers (a.k.a. – device servers, edge computers, network appliances, serial hubs) and how the leading products in this segment perform when connecting serial devices to application software using the Ethernet as the communication medium. This report focuses on LATENCY, the cause for it and the impact that it has on the speed of processing data across a local-area network. We believe the test is fair and equitable. And we believe that the results accurately portray the performance of the various products that are currently the market leaders in this technology segment. We want you to be totally convinced that the products perform as described in this report – so we have provided you with everything you need to recreate the benchmark yourself (except of course the hardware). If you run your own benchmark and have any questions, we would like to hear from you. Please contact David Boldt (benchmark co-author) at david.boldt@comtrol.com or 763-494-4100.

JUST FOR FUN … Once you have seen the benchmark results in tabular form, you might want to have some fun by watching the competitors *fight it out on the race-track*. Since the benchmark measures elapsed time, we thought a drag race would be in order – where the first to the finish line is the winner. To see all the action, please visit http://www.comtrol.com/devicemasterdrags



COMTROL vs DIGI SHOOT-OUT



COMTROL vs LANTRONIX SHOOT-OUT

EXECUTIVE SUMMARY

More than 20 years ago, engineers at Comtrol responded to a request for an expansion card to be used with the "new" *personal computer* so that it could connect multiple peripherals using serial communications. The result was the first "multiport serial concentrator card" for personal computers, which has since been widely adopted as the standard for connecting electro-mechanical devices using serial communications ("serial devices"). This traditional approach of connecting serial devices to a PC using a multiport card has a well-known set of limitations:

- *Need for specialized cabling (including cost and time required to accomplish the installation)*
- *Physical distance limitation of 50 feet between a PC and an RS-232 serial device*
- *Serial device operational dependency on continued availability of the host PC (application server)*
- *Personal computer cost of ownership (e.g. acquisition, maintenance, support)*

In the mid-1990s, Comtrol learned of the pending obsolescence of the PC ISA expansion bus that would force user migration to the (then) new PCI expansion bus. Anticipating more forced migration in the future (which has happened with the Universal Serial Bus and then "universal" PCI expansion bus), Comtrol searched for an alternative high-speed communication bus that might be more stable over the long term.

At the time, Ethernet was already installed within most major organizations across virtually all market sectors. Since Ethernet was already in place, using it as a data communications medium would eliminate the need for costly serial cabling. Plus, Ethernet also solved the 50-foot distance limitation of RS-232 serial communications, enabling deployment of a serial device anywhere an Ethernet connection was available. Finally, host-PC dependency could be more effectively managed by installing a backup version of the application software on a different networked PC. You could then switch from one to the other if the primary PC failed – keeping the serial device running!

Having identified Ethernet as the optimum communications bus, all that was missing was a cost-effective, reliable platform to connect the serial devices and handle network communication. In 1997 Comtrol developed the industry's first "serial port server" (InterChangeVS™). This groundbreaking innovation led the way for more powerful successor serial port server products, culminating with today's U.S. market leader* – DeviceMaster®!

**Source: NPD Intellect 2002 PC Products Unit Sales Summary – Comtrol adjusted*

Serial port servers have now been available for seven years. While the adoption rate is growing, the traditional approach of using PCs with multiport serial expansion cards remains the predominant device connectivity technology.

Comtrol believes that a major barrier of serial port server adoption is the presumed latency associated with network communications. Many design engineers assume that the overhead introduced by the Ethernet and its network communications protocol processing would decrease communication speed across the network, making it an unrealistic substitute technology for PC/Device direct connections.

In an effort to provide quantifiable evidence of network-induced latency, Comtrol established a benchmark study called the ***Serial Shoot-Out***. The test sent data from a PC, over an Ethernet network to a serial port server, and back again. The elapsed time of each test was recorded. The test was repeated twice, running 10,000 iterations each, and then the average was reported as the product's observed latency. To provide comparison to traditional device connectivity methods, the test was also run on a native serial port (located on a PC motherboard). Finally, to remove any doubt about the validity of the test, Comtrol engaged a third-party testing service to run the benchmark, which independently verified the results. As shown below, the Comtrol and test service results were very similar, indicating that the benchmark is valid and repeatable.

Product/Vendor	Latency (ms) Comtrol Test	Latency (ms) VeriTest
Workstation COM Port (Reference)	5.35	5.67
Comtrol DeviceMaster 16RM (RTS mode)	5.91	4.08
Comtrol DeviceMaster 16RM	8.55	7.13
Comtrol DeviceMaster 1 (RTS Mode)	9.47	4.07
Comtrol DeviceMaster 1	17.39	15.85
Comtrol PRIMO	13.10	10.65
Comtrol RocketPort Serial Hub IA (RTS)	9.20	9.16
Comtrol RocketPort Serial Hub IA	19.13	19.10
Digi One IA RealPort	107.21	110.06
Digi Portserver 16 Rack	124.27	120.01
Lantronix UDS-10	521.16	529.36
Lantronix UDS-100	523.17	529.36
Lantronix MSS4	755.51	562.96

All products were tested using industry-standard TCP/IP network communication protocol so the results are directly comparable. DeviceMaster products, however, were also tested using Rapid Transport Service™ (RTS), Comtrol's proprietary, patent-pending protocol that is designed to maximize serial data communications over Ethernet. Interestingly, when the DeviceMaster was run with RTS, it produced nearly the same latency as the directly connected serial port – and in the independent test lab report, actually BEAT the native serial port latency!

The results speak for themselves … all of the Comtrol products tested *significantly out-performed* the competition in both TCP/IP and RTS communication modes!

Using the latency and computed throughput generated by this test, design engineers now have a way to determine if deploying a serial port server as a replacement for a PC/Serial Card will meet performance requirements for a given application. It must be emphasized, however, that actual results may deviate from the test results primarily due to factors beyond the scope of this study (such as network traffic). The only way to know for certain if this alternate technology will work is to try it – and Comtrol has no-risk evaluation units for qualified customers!

There is one final observation about the test results.

On October 16, 2002, Digi International issued a news release declaring its Digi One IA as the fastest performer in a similar network-latency benchmark test. In comparing the results of the two tests, we find that the tested products produced similar performance with the exception of the Digi One IA. To date, Digi has not shared its test programs with any third parties, so Comtrol is unable to precisely determine why the results are so significantly different. Note that both the Comtrol and Digi test were performed by the same independent testing service, and they too are at a loss to explain the performance differential. Unlike Digi, Comtrol has provided all of the test parameters and the actual test software so that anyone (including Digi) can replicate this benchmark. We are confident with the results and confident that DeviceMaster is the unequalled serial port server performance leader!

Thank you for your interest in Comtrol and DeviceMaster.

SERIAL DEVICE CONNECTIVITY OVERVIEW

Personal computers equipped with serial multiport expansion cards remain the most prevalent method of connecting and communicating with electro-mechanical devices. Figure 1 depicts this configuration with a multiport card (RocketPort®) installed in the PC, communicating with an application through the multiport driver software and operating system.

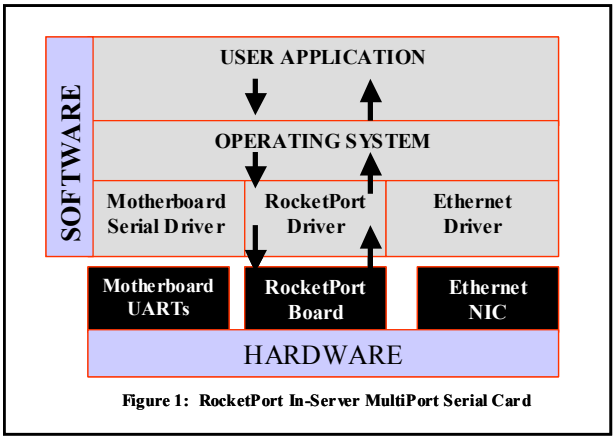


Figure 1: RocketPort In-Server MultiPort Serial Card

The "quality of service" provided by a serial multiport expansion card is directly related to the completeness of its driver software. The user application software is written to use specific serial communications features and functions supported by the operating system running on the PC. The mutliport card device driver implements the features and functions provided by the operating system, making them available on the additional serial ports added by the multiport expansion card. Accordingly, each driver must be customized to support the functions provided by each unique operating system, resulting in the quality of service relationship between software driver and card.

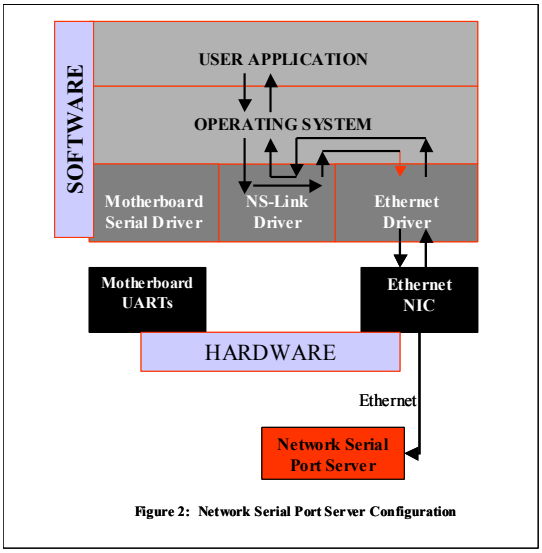
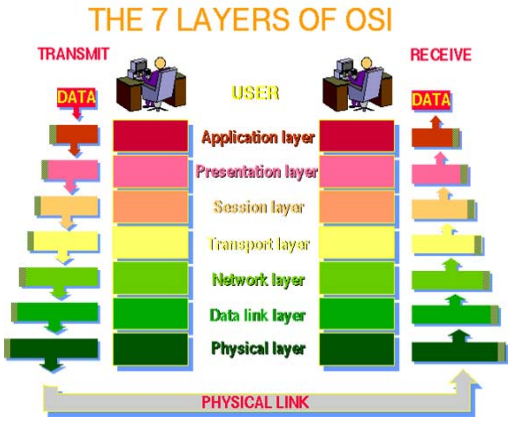


Figure 2: Network Serial Port Server Configuration

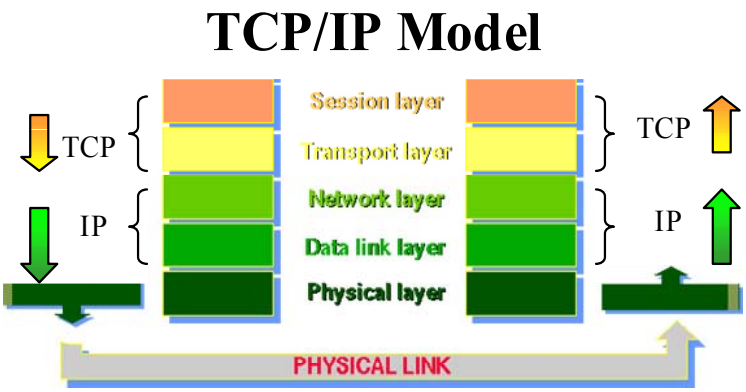
Figure 2 shows the alternate configuration using the Ethernet as the data communications bus. In this mode, the driver has a dual role. Not only does it have to interface with the operating system to provide the serial port features and functions, but it also must communicate over the network to the physical serial ports. For this reason, the driver for a serial port server is divided into two modules – one module runs on the host PC and another runs on the serial port server. The modules then communicate with each other using a network data transfer protocol. Therefore, in addition to supporting the features/functions of the operating system, these drivers must also make optimum use of network communications – which adds yet another element into the overall quality of service equation.

NETWORK COMMUNICATION PROTOCOL

Three characteristics define a network: topology (physical layout of connected computers), architecture (peer-to-peer or client/server), and protocol. Protocol defines a common set of rules and signals that are used by attached computers for communication. TCP/IP is undoubtedly the most popular protocol and the one that is prevalent in this market.



To promote compliance with the "rules", a model, known as the OSI (Open Systems Interconnection), was created to define a framework of network communication software. This OSI framework consists of seven "layers". Communication is passed from one layer to the next, starting at the application layer and proceeding down the "stack" to the bottom (physical) layer, over the channel and then back up the stack again to the receiving application.



TCP/IP protocol operates slightly differently. TCP/IP replaces four OSI layers with two new layers (TCP and IP) that generally follow the OSI processing schema. TCP/IP also adds routing and error checking capabilities that can increase processing time (and latency) on noisy or heavily loaded networks. Using TCP/IP, the PC-based driver module and the serial port server driver module communicate to transfer data across the physical layer. Using TCP/IP requires that each network device receive an IP (internet protocol) address so that it can be identified on the network. The complexities of the protocol and increased processing due to error checking can increase latency. As a result, network design engineers must consider the minimum tolerable latency of data communications and offset that against the overhead associated with the use of standard network protocols.

NETWORK COMMUNICATIONS LATENCY

In networking and serial communications, latency is defined as the DELAY (measured in time) associated with transferring a packet of data from a source to a destination. In applications using multiport serial cards, latency was rarely a major consideration because the driver and internal high-speed communication bus within the PC imposed only a few milliseconds of latency (delay). However, as we have seen, when a shared resource such as the Ethernet is used as the communication bus, external factors can impact performance and introduce latency. Factors such as protocol handling, network traffic, poorly written "COM port redirector" driver software, and network hubs/routers can introduce additional processing overhead that could potentially add hundreds of milliseconds to the time required for the data to reach its destination.

While a delay of fractional seconds seems irrelevant, in some applications, such as material handling, satellite communications and real-time device monitoring, this small delay can have a significant negative impact on the automation process. Here is a brief example:

Consider a package handling system. The package sits on a moving conveyor. It passes a scanner that reads a bar code on the package. Captured bar code data is transmitted from the scanner's serial port to a PC running a decoder and package routing application program. The system has only 30 milliseconds to analyze the bar code, determine the destination and then transmit a routing instruction back to the conveyor. If a serial port server and Ethernet connection is used instead of a direct PC connection, and this new approach introduces 50 ms of latency to the overall throughput, the package could pass the conveyor routing point before the routing instructions are received. To fix this problem, the conveyor speed must be reduced. But running the conveyor slower reduces its output and efficiency. Clearly this is not a desirable solution!

When the benefits of serial port servers are desirable, but the latency introduced by the TCP/IP network protocol is a problem, a different approach is needed. Comtrol has developed that alternative. Rapid Transport Service™ (RTS) is a proprietary, patent-pending network protocol developed by Comtrol that uses a modified protocol technique, augmented by special software. The result is a "low overhead" protocol that is simple to configure and that produces outstanding low-latency results. But due to modifications made to the protocol, RTS cannot support wide-area networking or routing functions. Therefore, RTS requires that the host PC server and the serial port server reside on the same Ethernet segment. While somewhat limited for WAN use, RTS can deliver the benefits of remote serial ports in the right situations.

SERIAL SHOOTOUT

Comtrol is committed to driver quality of service. Known for driver excellence with its Hostess® and RocketPort (multiport) cards, Comtrol engineers accepted the challenge to develop the most sophisticated, high-performance, serial port server driver in the industry. Part of the development process was the creation of a performance benchmark, which would measure the latency of Comtrol products as compared to native PC serial ports and competitive serial port server products.

The benchmark test sends a packet, consisting of one byte of data, over the Ethernet using either TCP/IP or the Comtrol RTS protocol, to a remote serial port server where the data is echoed back to the originating PC. The echoing of data in the tests is accomplished using a loop-back plug that connects the RS-232 Tx data pin to the Rx data pin. Use of a loop-back plug instead of a second computer to echo the data eliminates questionable results that might be artificially introduced by a second computer (since routing data through an intermediate computer adds an unwarranted level of complexity to the process). Since many manufacturers supply a loop-back plug, it is far simpler to set up such a test.

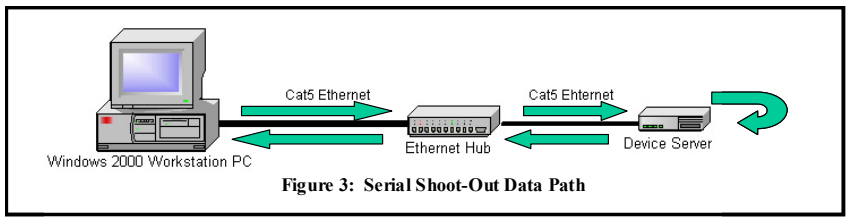


Figure 3: Serial Shoot-Out Data Path

This test provides a realistic approximation of performance under simulated real-world conditions and allows for easy replication with commonly available PC equipment. Plus, the configurable parameters and simplified setup of this test make it an invaluable tool for users and integrators looking to match the right device server product to timing-critical applications. The following hardware was used for the test.

HOST PC SPECIFICATIONS			
Test PC	**Processor**	**RAM**	**Operating System**
Gateway Select	AMD K-7	64 MB	Windows 2000 Pro

NETWORK SPECIFICATIONS		
Ethernet Hub Model	**Vendor**	**Connection**
Netelligent	Compaq	10Base-T

After defining the path that the data will take and the equipment specifications for the "test track", a specialized computer program is required to execute the benchmark. The program runs on the host PC, generates the test data, sends the packet out over the network, and measures the round-trip elapsed time. The Python programming language was used to implement the benchmark program. Python is an object-oriented programming language that compiles into byte-code and runs on a virtual machine [similar to Java, .NET, Perl, and others]. It was selected because:

1. Python is safe, readable, and easy to modify (for trial and error experimentation)
2. The Python implementation does not require the license of a compiler or development license, making it convenient for any third party to recreate the program for independent execution of the benchmark test
3. The *win32all* extensions provide direct access to win32 system calls used to exercise serial ports.

In order to run the benchmark program (shown in Appendix A), a Python implementation will have to be installed on the host system. Installation of Python is described below (for more information on Python, visit http://www.python.org/)

Installing Python. Python installs from http://ActiveState.com. The package contains the standard Python implementation plus some additional tools and libraries (including the win32all library that is used by the benchmark program). The download/installation process is as follows:

1. Download ActivePython from http://www.activestate.com/Products/Language_Distributions/
2. Double-click on the downloaded file (e.g. ActivePython-2.2.2-224-win32-ix86.msi) to install ActiveState Python. Select "typical" installation (requires approx 35MB of disk)

The "stock" version of Python can also be downloaded by visiting http://www.python.org/2.2.2/. The win32all extensions must then be separately downloaded from the following site: http://starship.python.net/crew/mhammond/win32/Downloads.html. The default location for Python download/installation differs from that used by ActiveState Python. As a result, the path of the Python executable in the DOS batch file used to run the benchmark program must be modified before the benchmark program can be run.

Once the benchmark file is acquired, the unzipped file then creates the following two files: bench.py (the Python program used to measure round-trip latency on a COM port) and bench.bat (a DOS batch file used to run bench.py). If the Python executable file is installed in a location other than C:/Python22/python.exe, the path will have to be modified accordingly. The benchmark program is run from a command prompt in the directory where bench.bat and bench.py are located: C:\> bench.bat COM1. This command will write single byte data blocks to COM1 and measure the time that elapses until the data is read back. The measurement will be repeated for 100 iterations with a delay of 100ms between iterations. Finally, the min, mean, max, and standard deviation of the measured delays (in milliseconds) will be displayed.

The following options may be specified on the command line (before the COM port parameter):

-q	Print only the final results -- don't print results from each iteration.
-c <count>	Run <count> iterations [default 100]
-d <delay>	Wait for <delay>ms between reading data and next write. [default 100]
-b <baud>	Set port to <baud> bits per second. [default 9600]
-i <time>	When reading, use an inter-character timeout of <time>ms [default 2]
-s <size>	Use a block size of <size>bytes. [default 1]
-t <time>	Use a total read timeout of <time>ms. [default 2000]

In this example, the baud rate is set to 115200 and the number of iterations to 10. The theoretical minimum time for writing and then reading a single byte at 115,200 baud, using a 16550 UART (with FIFO enabled), is approximately 0.3ms. It appears that the normal system overhead (including the benchmark program and system calls) is less than 0.3ms. This overhead will be constant from one test to another and is quite small compared to the latencies we are going to measure over the network.

```
C:\>bench.bat -c10 -b115200 com2
Connected to \\.\com2 at 115200 baud
  0  0.6
  1  0.6
  2  0.5
  3  0.5
  4  0.5
  5  0.5
  6  0.6
  7  0.5
  8  0.5
  9  0.5
min/mean/max SD = 0.54 0.55 0.59 0.01
```

The benchmark test was executed using the following parameters:

inter-character timeout: 2ms
total read timeout: 2,000ms
iteration delay: 100ms
baud rate: 9,600bps
block size: 1 byte
number of iterations: 10,0000
[All parameters are default values except iteration count.]

SERIAL SHOOT-OUT: PRODUCTS TESTED

Vendor/Product		Part No.	Price*	Description
	**COMTROL** DeviceMaster RTS-1	99000-0	$309.90	**Ethernet**: One 10/100Base-T **Serial Ports**: One RS-232/422/485
	**COMTROL** DeviceMaster PRIMO	98810-6	$236.06	**Ethernet**: One 10/100Base-T **Serial Ports**: One RS-232/422/485
	**COMTROL** RocketPort Serial Hub ia	98651-5	$459.81	**Ethernet**: One 10/100Base-T **Serial Ports**: Two RS-232/422/485 **Other**: DIN Rail Mount
	**COMTROL** DeviceMaster RTS-16RM	98800-7	$1,217.81	**Ethernet**: Two 10/100Base-T Fast Ethernet Switch **Serial Ports**: Sixteen RS-232/422/485 **Other** 1-U Rack Mount
	**DIGI** One IA RealPort	700017777	$529.95	**Ethernet**: One 10/100Base-T **Serial Ports**: One RS-232/422/485 **Other** DIN Rail Mount
	**DIGI** PortServer TS-16	70001743	$1,099.17	**Ethernet**: One 10/100Base-T **Serial Ports**: Sixteen RS-232/422/485 **Other** DIN Rail Mount
	**LANTRONIX** U D S - 10	UDS10--01	$129.59	**Ethernet**: One 10Base-T **Serial Ports**: One RS-232/422/485
	**LANTRONIX** U D S - 100	DS100-01	$169.35	**Ethernet**: One 10/100Base-T **Serial Ports**: One RS-232/422/485
	**LANTRONIX** MSS - 4	MSS-4-D-01	$539.89	**Ethernet**: One 10/100Base-T **Serial Ports**: Four RS-232/422/485

* Information taken from the CDW (catalog) Web site (www.cdw.com) on 2/7/03 with the exception of the Digi One Realport IA, which was obtained from B&B Electronics (www.bb-elec.com on 2/07/03).

The Serial Shoot-Out measures the performance of leading serial port servers against each other and against native PC-based serial ports. It is not feasible to test every product, so a *representative sample* of comparable product was selected. Selection criteria included price, configuration, 2002 market share (NPD Intellect adjusted), and a serial COM port driver.

BENCHMARK RESULTS

An examination of the results obtained from the benchmark testing reveals a wide range in performance among the tested products. Latency is represented as the elapsed time (in milliseconds) required to execute one iteration of the network communications test. Therefore, the LOWER NUMBERS REPRESENT SUPERIOR PERFORMANCE.

To establish a common performance baseline, the test first was performed using the internal serial port of the Windows 2000 host PC. This port, COM 1, registered an average time of 5.35 ms to complete the loopback test at a port setting of 9600 bps. By establishing this number for a standard 16550 UART-based COM port, a reasonable conclusion can be reached that similar results achieved by a network-based COM port indicate performance rivaling an in-server multiport card or native PC COM port. The table below lists the model information and results obtained for each product included in the benchmark test.

Product	Driver	Driver Rev.	Network Protocol	Avg. Latency (ms)
Workstation COM Port (Ref)	Windows 2000	N/A	N/A	5.35
Comtrol DeviceMaster RTS 16	Windows 2000	5.22.0.0	Comtrol RTS	5.91
Comtrol DeviceMaster RTS 16	Windows 2000	5.22.0.0	TCP/IP	8.55
Comtrol DeviceMaster RTS 1	Windows 2000	5.22.0.0	Comtrol RTS	9.47
Comtrol DeviceMaster RTS 1	Windows 2000	5.22.0.0	TCP/IP	17.39
Comtrol DeviceMaster Primo	Windows 2000	1.0.2.0	TCP/IP	13.10
Comtrol RocketPort Serial Hub IA	Windows 2000	5.22.0.0	Comtrol RTS	9.20
Comtrol RocketPort Serial Hub IA	Windows 2000	5.22.0.0	TCP/IP	19.13
Digi Portserver 16 Rack	Windows 2000	2.5.67.0	TCP/IP	124.27
Digi One IA RealPort	Windows 2000	2.5.67.0	TCP/IP	107.21
Lantronix UDS-10	Windows 2000	1.0	TCP/IP	521.16
Lantronix UDS-100	Windows 2000	1.0	TCP/IP	523.17
Lantronix MSS4	Win32	1.0	TCP/IP	755.51

Please contact Comtrol to obtain the test result from the independent test service retained by Comtrol to separately execute this benchmark.

COMTROL vs. DIGI – BENCHMARK COMPARISON

In October 2002 Digi International issued a press release extolling their Digi One IA RealPort product's latency ranking in a comparison of competing serial port (device) servers. The latency benchmark was supplied by Digi, and performed by VeriTest on Digi's behalf. The VeriTest report compared the latency of several serial port (device) servers from various manufacturers in a test that sent one data byte from a "master" PC via serial communication to a device server and on to a "slave" PC. This test measured the elapsed time for that data to make the round trip from the "master" PC, through the device server to the "slave" PC and return to the "master" PC. The test results proclaimed that the Digi IA RealPort bested the field of competitors, which included Comtrol, Lantronix, and Moxa, by what VeriTest categorized as "significantly lower round-trip latencies". The following is a direct reprint of the Digi press release announcing their benchmark results as published on October 16th, 2002.

VeriTest Report Confirms That Digi One IA RealPort Performs Significantly Better Than Competitors

(Minnetonka, Minn., October 16, 2002) - Digi International®, Inc. (Nasdaq: DGII), the leader in Connectware, today released a report summarizing the results of latency performance tests which showed that when tested against competitors including Lantronix, Moxa and Comtrol, the Digi One IA RealPort™ generated the lowest latency of all device servers tested.

Key findings of the report stated that: "The Digi One IA RealPort device server generated significantly lower overall average latency compared to the other device servers we tested." The performance tests were conducted by VeriTest, a leading provider of independent outsourced testing solutions, and a division of Lionbridge Technologies, Inc. (Nasdaq: LIOX).

Many industrial and control applications require low latency in order to operate reliably, with maximum throughput. These include discrete control systems, CNC/DNC and critical event reporting. Often, these systems were based on the performance of direct wire line connections, without assuming the overhead of crossing network boundaries.

"The lower the latency generated, the better the speed and throughput for time critical applications," explains Joel Young, vice president of device server products for Digi International. "These test results clearly demonstrate that Digi products provide the fastest way to and through your network, and prove without a doubt that Digi is the leading provider of state-of-the-art device server products."

The test measured the average latency over 10,000 iterations, and consisted of a master application sending a single 8-bit character to the device server under test and then waiting for the character to be received by the slave application and echoed back to the master. Latency results were measured in milliseconds and the results were as follows:

Digi One IA RealPort:	5.81
Moxa NPort Express DE-311:	10.49
Comtrol DeviceMaster DMSH-1:	12.18
Comtrol RocketPort Serial Hub ia:	29.31
Lantronix UDS100:	586.29
Lantronix CoBox-DR1:	707.35
Lantronix MSS-VIA:	762.73
Lantronix UDS-10:	860.80

Digi Press Release Continued

The Digi One IA RealPort is part of the Digi One™ product suite, which is designed to network enable virtually any serial device in the industrial automation environment. The product allows connection of new and legacy serial devices and PCs from anywhere on the factory floor, using any operating system. Advanced Modbus, Ethernet/IP, Allen-Bradley Ethernet, DF1, Compoway/F, FINS and Hostlink features allow for improved sharing of information between various factory floor devices, and advanced SNMP features allow for management of the industrial device from a centralized location. These features make the Digi One IA RealPort more versatile and cost-effective than other products on the market.

Joe Dunsmore, president and CEO of Digi International, said, "These tests further validate the quality of our device server products, and once again verify our position as the leading provider of innovative device networking solutions. We remain committed to providing our customers with the products that will make their daily operations easier and help their businesses grow."

Comtrol Review of Digi's Benchmark White Paper

Both Comtrol and Digi "white papers" agree on the cause and effect that latency has on the feasibility of implementing serial port servers. But when the results of the testing are compared, Comtrol and Digi differ significantly in one area – Digi's performance!

Three points should be noted. First, based on Comtrol's benchmark, Digi's claim that its Digi One IA RealPort has "… significantly lower overall average latency compared to the other device servers we tested" would be incorrect. Second, in Digi marketing information, the results of this benchmark are generalized to sound like all Digi serial port server products achieved this level of performance. It is clear in reviewing the benchmark that in each case (except the Lantronix UDS10 and UDS 100) different products from the same vendor produced different latency. The Digi benchmark (report) showed test reports for only one Digi product. Finally, Digi has not made its test program available nor do they offer an explanation for the rationale of using an unusual network topology (placing a "slave PC" into the configuration) when in practice, the data path is clearly from the host PC to the serial port server and directly back to the host PC.

With the exception of the Digi One IA RealPort, most of the other products tested by Digi and Comtrol fall within a reasonable tolerance. The improvement in Comtrol product performance between the Digi test and the Comtrol test can be explained quite easily. Comtrol offers a standard configuration option in all serial port server drivers that maximizes network performance in situations where latency is an issue. In a benchmark such as this one, this well-documented feature should have been activated by Digi – and if it had been, Comtrol's superior performance would have been clearly demonstrated!

DIGI One IA RealPort Performance Variance

Latency ... Reported by Digi 5.81ms

Latency ... Observed by Comtrol 107.21ms

Comtrol observation: Actual Digi latency achieved in the Comtrol benchmark was *18 TIMES SLOWER* than the latency reported by Digi

Since the results achieved by Comtrol were so dramatically different, the test was re-run on the Digi IA RealPort multiple times – with nearly the same latency observed in each instance. Meanwhile, the latency of the other non-Digi products, including Comtrol products, approached the latency reported by VeriTest/Digi. Question: ***How can all the competing products achieve nearly the same latency and the Digi latency be 18 TIMES SLOWER in the Comtrol test?***

Appendix A
Comtrol Benchmark Software
Application Source Code

```
#!/usr/local/bin/python.exe
from win32file import *
from win32event import *
import win32con
import getopt, time, sys, operator, math, random, socket

def mean(v):
    return reduce(operator.add,v)/len(v)

def stddev(v,m):
    devsq = [(x-m)*(x-m) for x in v]
    return math.sqrt(reduce(operator.add,devsq)/(len(devsq)-1))

def toHex(data):
    return " ".join(["%02x" % ord(c) for c in data])

def openPort(s):
    if not s.startswith('\\'):
        s = "\\\\.\\" + s
    handle = CreateFile(s, win32con.GENERIC_READ | win32con.GENERIC_WRITE,
                        0, None,  # exclusive access, no security
                        win32con.OPEN_EXISTING,
                        win32con.FILE_ATTRIBUTE_NORMAL, None)
    SetCommMask(handle, EV_RXCHAR)
    SetupComm(handle, 4096, 4096)
    PurgeComm(handle, PURGE_TXABORT|PURGE_RXABORT|PURGE_TXCLEAR|PURGE_RXCLEAR)
    SetCommTimeouts(handle, (ictimeout, 0, timeout, 0, timeout))
    dcb = GetCommState(handle)
    dcb.BaudRate = eval("CBR_%d" % baud)
    dcb.ByteSize = 8
    dcb.fOutX = 0
    dcb.fInX = 0
    dcb.Parity = NOPARITY
    dcb.StopBits = ONESTOPBIT
    SetCommState(handle, dcb)
    print "Connected to %s at %s baud" % (s, dcb.BaudRate)
    return handle

testData = "".join([chr((i&0xff)|0x40) for i in range(2048)])

opts,ports = getopt.getopt(sys.argv[1:],"qc:d:b:s:i:t:",
                           ("count=","delay=","baud=","size=",
                            "interchar-timeout=","timeout=","quiet"))
baud = 9600
count = 100
delay = 100
blocksize = 1
ictimeout = 2
timeout = 2000
quiet = 0

for opt,val in opts:
    if opt in ('-c', '--count'):
        count = int(val)
    elif opt in ('-d', '--delay'):
        delay = int(val)
    elif opt in ('-s', '--size'):
        blocksize = int(val)
    elif opt in ('-b', '--baud'):
        baud = int(val)
    elif opt in ('-i','--interchar-timeout'):
        ictimeout = int(val)
```

```
    elif opt in ('-t','--timeout'):
        timeout = int(val)
    elif opt in ('-q','--quiet'):
        quiet = 1
    else:
        raise "Unknown option "+opt

if ictimeout <= 0:
    ictimeout = 0xFFFFFFFF

if len(ports) < 1 or len(ports) > 2:
    raise "need one or two port IDs"

txhandle = openPort(ports[0])
if len(ports) == 2 and ports[0] != ports[1]:
    rxhandle = openPort(ports[1])
else:
    rxhandle = txhandle

loops = 0
deltas = []

delay = delay / 1000.0  # convert delay from ms to seconds

while loops < count:
    txdata = testData[(loops & 0xff):][:blocksize]
    txTime = time.clock()
    rxdata = ''
    rc,txcount = WriteFile(txhandle,txdata)
    if rc or txcount != len(txdata):
        raise "WriteFile error %d,%d" % (rc,txcount)
    while 1:
        rc,data = ReadFile(rxhandle,blocksize)
        if rc:
            raise "ReadFile error %d" % rc
        rxdata += data
        if len(txdata) == len(rxdata):
            break
    rxTime = time.clock()
    if rxdata != txdata:
        sys.stdout.write("Tx: %s\n" % toHex(txdata))
        sys.stdout.write("Rx: %s\n" % toHex(rxdata))
        sys.stdout.flush()
        raise "data error"
    delta = (rxTime-txTime)*1000.0  # convert elapsed time to ms
    deltas.append(delta)
    sys.stdout.write("%d %5.1f\n" % (loops,delta))
    time.sleep(delay)
    loops += 1

deltaMean = mean(deltas)
deltaSD = stddev(deltas,deltaMean)

sys.stdout.write("min/mean/max SD = %0.2f %0.2f %0.2f %0.2f\n" % \
                 (min(deltas),deltaMean,max(deltas),deltaSD))
```