

Application Note

Windows 2000/XP TCP Tuning for High Bandwidth Networks



mGuard smart



mGuard PCI



mGuard blade



mGuard industrial



mGuard delta

TABLE OF CONTENTS

1	Introduction	3
2	Test environment	4
3	Maximum Transmission Unit (MTU) and Fragmentation	5
3.1	Maximum Transmission Unit (MTU)	5
3.2	Maximum Segment Size (MSS)	5
3.3	Path Maximum Transmission Unit Discovery (PMTUD)	6
3.4	Summary	6
4	Bandwidth Delay Product (BDP) and Round Trip Time (RTT)	7
5	Send Socket Buffer Size and TCP Window Size	8
5.1	Send Socket Buffer Size	8
5.2	TCP Window Size	8
6	Performance Tuning for Low Delay Networks (BDP <= 64kB)	9
7	Performance Tuning for High Delay Networks (BDP > 64kB)	10
7.1	Tuning example for RTT = 12ms	10
7.2	Tuning example for RTT = 20ms	11
7.3	Tuning example for RTT = 40ms	12
8	Performance Tuning: Summary	13
9	Appendix A: Windows Registry Parameters	14
9.1	DefaultSendWindow	14
9.2	EnablePMTUDiscovery	14
9.3	MTU	14
9.4	Tcp1323Opts	15
9.5	TcpWindowSize	15
10	Appendix B: Dependencies	16
10.1	Influence of the send socket buffer size on the throughput	16
10.2	Influence of the send socket buffer size and the TcpWindowSize on the throughput	17
11	Appendix C: References	17

1 Introduction

Our appliances use hardware encryption for providing a high VPN throughput. Nevertheless customers couldn't reach good results in their real environment. We have started investigating on this performance issue and it turned out that a low VPN performance is rather caused by the Windows TCP settings, which need to be adjusted to the given network, than by the mGuard.

Recommendations on how to measure performance in packet-based networks are provided by RFC 1944 (by S. Bradner and J. McQuaid). This document defines a specific set of tests that vendors can use to measure and report the performance characteristics of network devices. The results of these tests will provide the user comparable data from different vendors with which to evaluate these devices. RFC1944 recommends using UDP packets for measuring TCP/IP performance because TCP relies on the sliding window mechanism for flow control, the latency between the sender and the receiver and depends on acknowledgements from the receiving end of the connection.


TCP tuning is complicated because there are many algorithms running and controlling TCP data transmissions concurrently, each with slightly different purposes. Actual throughput for a link depends on a number of variables like for example:

- Link speed (bits-per-second that can be transmitted)
- Propagation delay
- Window size
- Link reliability
- Network and intermediate device congestion
- Path MTU

Microsoft has really done a remarkable job. The TCP implementation includes virtually all of the recent extensions to improve performance but the default values of some parameters are too conservative and need to be adjusted for getting the optimum of performance. Usually you won't notice the reduced performance during your normal work but it gets visible when making performance measurement.

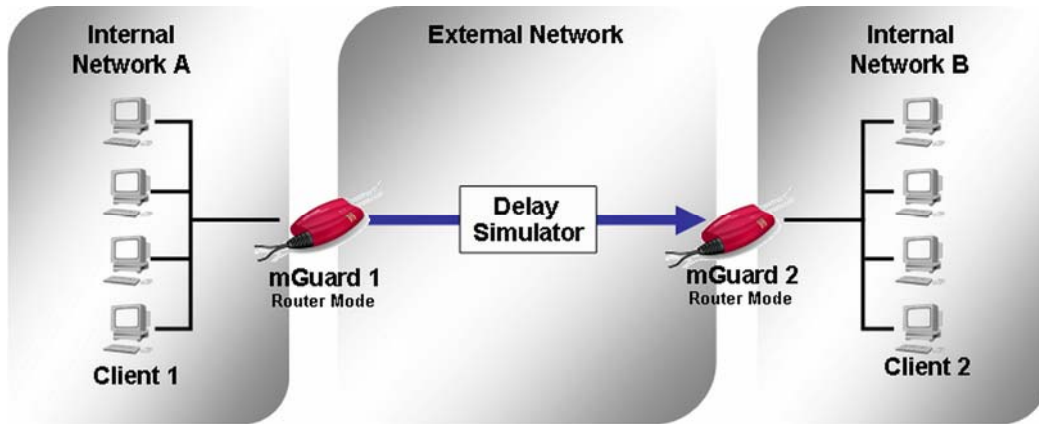
Tuning the Windows TCP settings according to this document will not only increase the VPN throughput. It will also increase the overall performance of your network. For low delay networks we could increase the overall performance by a factor of **1,8** and the VPN throughput by a factor of **1,5**. For high delay networks with an RTT of 40ms we could increase the overall performance by a factor of **11,7** and the VPN throughput by a factor of **3,4**. This is a remarkable result, which makes it worth to tune the Windows TCP settings.

The TCP/IP protocol suite implementation for Windows 2000/XP obtains all of its configuration data from the registry.

 **Note:** The modification of the Windows 2000/XP TCP parameters involves editing the registry. This should only be attempted by experienced system administrators because mistakes can render the system unbootable. After these registry changes are done, reboot to apply the changes.

2 Test environment

We have used the following environment when testing the influence of the Windows TCP settings on the performance:



Client 1: Windows 2000, SP 4, CPU 1.20 GHz, 512 MB RAM

Client 2: Windows XP, SP2, CPU 2.80 GHz, 512 MB RAM

mGuard 1/2: mGuard smart enterprise XL, Firmware v3.1.1, 533 MHz processor, 64MB RAM

All interfaces were operated in 100Mbit/s Full-Duplex mode. As measurement tool we have used a slightly modified version of *nttcp* which was compiled with the Windows Winsock library instead of using the *cygwin.dll*. As *Delay Simulator* we have used *ipfw* which is based on FreeBSD.

3 Maximum Transmission Unit (MTU) and Fragmentation

This chapter explains the meaning of the *Maximum Transmission Unit* (MTU) and the corresponding registry parameters. Changing the default Windows settings is not required for tuning the network but you should ensure that they are not misconfigured.


3.1 Maximum Transmission Unit (MTU)

The *Maximum Transmission Unit* (MTU) is the maximum packet size in bytes that the transport transmits over the underlying network. The size includes the transport header. An IP datagram can span multiple packets. For example, an Ethernet MTU value is 1500 bytes.

To prevent fragmentation, the MTU should be large enough to hold any IP datagram in a single frame. IP datagrams larger than the MTU are divided into fragments whose size is a multiple of eight octets. The fragments travel separately to the destination computer, where they are reassembled before the datagram is processed.

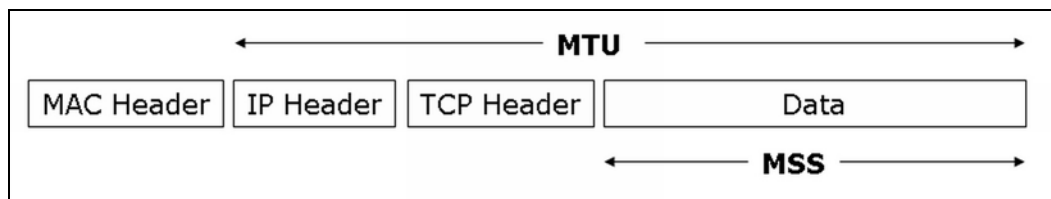
MTU detection is determined for all interfaces by the registry value of the *EnablePMTUDiscovery* (see below) entry. By default, the network adapter for each interface detects the largest MTU that the interface can transmit, and it uses that MTU for its transmissions. However, if MTU detection is disabled (*EnablePMTUDiscovery* = 0), the system uses a fixed MTU of 576 bytes which causes a loss of performance. If you change the default value of the **MTU** registry entry, you override either setting as it pertains to the interface represented by this subkey.

MTU detection is enabled by default in Windows 2000/XP, but the MTU value can be controlled with the addition of the parameter **MTU** to the registry. These parameters for TCP/IP are specific to individual network adapter cards. Refer to *Appendix A: Windows Registry Parameters* for getting further information about this parameter.

 **Note:** If you enter a value greater than the dynamically-determined MTU, the system uses the value of the dynamically-determined MTU instead. You can use this entry to reduce, but not to increase the size of the MTU.

3.2 Maximum Segment Size (MSS)

The TCP *Maximum Segment Size* (MSS) value specifies the maximum amount of TCP data in a single IP datagram that the local system can accept (reassemble). The IP datagram can be fragmented into multiple packets when sent. Historically, the MSS for a host has been the MTU at the link layer minus 40 bytes for the IP and TCP headers (e.g. MTU = 1500, MSS = MTU-40 = 1460). However, support for additional TCP options, such as time stamps, has increased the typical TCP+IP header to 52 or more bytes.




MTU versus MSS

3.3 Path Maximum Transmission Unit Discovery (PMTUD)

Path *Maximum Transmission Unit Discovery* (PMTUD) is an algorithm described in RFC1191. This algorithm attempts to discover the largest IP datagram that can be sent without fragmentation through an IP path and maximizes data transfer throughput.

PMTUD is enabled by default in Windows 2000/XP, but can be controlled with the addition of the parameter *EnablePMTUDiscovery* to the registry. Refer to *Appendix A: Windows Registry Parameters* for getting further information about this parameter.

By default, this entry applies to all interfaces. However, the MTU can be reduced for any particular interface by changing the default value of the MTU entry in the subkey for that interface.

 **Note:** Disabling PMTUD causes the system to use a fixed MTU of 576 bytes which causes a reduced performance.

3.4 Summary

- Ensure that the systems use PMTU discovery. The parameter *EnablePMTUDiscovery* shouldn't be defined in the registry or it should have the value 1. Otherwise an MTU of 576 bytes will be used which causes a reduced performance.
- Verify that the parameter *MTU* is not defined for the corresponding interfaces.

4 Bandwidth Delay Product (BDP) and Round Trip Time (RTT)

The amount of data that can be in transit in the network, termed *Bandwidth Delay Product* (BDP), is simply the product of the bottleneck link bandwidth and the *Round Trip Time* (RTT). BDP is a simple but important concept in a window based protocol such as TCP. Some of the issues discussed below arise because of the fact that the BDP of today's networks has increased way beyond what it was when the TCP/IP protocols were initially designed. In order to accommodate the large increases in BDP, some high performance extensions have been proposed and implemented in the TCP protocol. But these high performance options are sometimes not enabled by default and will have to be explicitly turned on by the system administrators.

The largest buffer the original TCP (without the high performance options) supports is limited to 64 kBytes. If the BDP is small either because the link is slow or because the RTT is small (in a LAN, for example), the default configuration is usually adequate. But for a paths that have a large BDP, and hence require large buffers, it is necessary to have the high performance options discussed in the next sections be enabled.

To compute the BDP, we need to know the speed of the slowest link in the path and the *Round Trip Time* (RTT). The peak bandwidth of a link is typically expressed in Mbit/s. The RTT is a measure of the time it takes for a packet to travel from a computer, across a network to another computer, and back. The *ping* program can be used to retrieve the RTT.

The first step for tuning the network consists of retrieving the RTT. Execute the ping program on an IP address of the remote network. The RTT is displayed in milliseconds.

```
C:\>ping 192.168.28.100

Pinging 192.168.28.100 with 32 bytes of data:

Reply from 192.168.28.100: bytes=32 time=21ms TTL=125
Reply from 192.168.28.100: bytes=32 time=20ms TTL=125
Reply from 192.168.28.100: bytes=32 time=20ms TTL=125
Reply from 192.168.28.100: bytes=32 time=20ms TTL=125

Ping statistics for 192.168.28.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 20ms, Maximum = 21ms, Average = 20ms
```

Knowing the bottleneck link speed and the RTT, the BDP can be calculated as follows:

$$\text{BDP [kBytes]} = (\text{bandwidth [kBits/s]} / 8 [\text{bits}]) * \text{RTT [s]}$$

Example:

$$\begin{aligned} \text{Bandwidth} &= 100 \text{ Mbit/s, RTT} = 20\text{ms} \\ \text{BDP} &= (100000 [\text{kBits/s}] / 8 [\text{bits}]) * 0,02 [\text{s}] = 250 \text{ kBytes} \end{aligned}$$

Based on these calculations, it is easy to see why the typical default buffer size of 64 kBytes would be way inadequate for this connection.

5 Send Socket Buffer Size and TCP Window Size

There are two parameters which have a significant impact on the network performance: The send socket buffer size which is provided by the system and the TCP window size.

5.1 Send Socket Buffer Size

TCP sessions are established by sockets. Most operating systems also support separate per connection send and receive buffer limits that can be adjusted by the user, application or other mechanism as long as they stay within the maximum memory limits. These buffer sizes correspond to the `SO_SNDBUF` and `SO_RCVBUF` options of the `setsockopt()` call. The socket buffers must be large enough to hold a full BDP of TCP data plus some operating system specific overhead. If the send and/or receive window size is exceeded then the application will wait until again memory will be available which of course reduces the performance.

Windows uses as default 8192 bytes for the send socket buffer size for a system with more than 32MB RAM. This parameter needs to be adjusted for optimizing the throughput. If an application provides the possibility to specify this value then you should define it at the application level. Otherwise you need to set it through the parameter *DefaultSendWindow* of the *AFD Registry Parameters* in the Windows registry. Refer to chapter *Appendix A: Windows Registry Parameters* for getting further information about this parameter. If you modify this parameter in the registry, it is in effect for sockets used by all Winsock programs. The change causes the default send buffer allocated for each socket created to be equal to the new size.

This parameter needs to be tuned at the sender's site and should be equal to the calculated BDP. Refer to chapter *Influence of the send socket buffer size on the throughput* for getting an overview about how the throughput depends on the size of the send socket buffer.

5.2 TCP Window Size

The TCP window is the amount of unacknowledged data in flight between the sender and the receiver. Data is sent by TCP in segments that are typically 1460 bytes in length. If the sender is permitted a window size of only 1 segment, the sender transmits a single segment, and waits for an acknowledgement from the receiver. If the transmission delay between sender and receiver is long, this means very low throughput (very few segments transferred per unit time). Both sender and receiver spend most of their time waiting for messages to be transmitted from one end of the connection to the other.

Windows uses as default a TCP window size of 64kBytes which is inadequate for a high bandwidth high delay network. Large windows support is controlled by the registry key value *Tcp1323Opts*. You need to activate this option on both systems for enabling large windows support. If timestamps are not absolutely required then you should set *Tcp1323Opts* to 1. Otherwise, if you set *Tcp1323Opts* to 3, also timestamps will be transmitted, which reduces the performance.

Then you need to specify the appropriate TCP window size on the receiver's site with the parameter *TcpWindowSize*. Refer to chapter *Appendix A: Windows Registry Parameters* for getting further information about those parameters.

You should specify the value of the calculated BDP as *TcpWindowSize*. For greatest efficiency, the *TcpWindowSize* should be an even multiple of the *TCP Maximum Segment Size (MSS)*. You can record the packets for example with Ethereal for obtaining the value of the used MSS.

Refer to chapter *Influence of the send socket buffer size and the TcpWindowSize on the throughput* for getting an overview about how the throughput depends on the size of the send socket buffer and on the *TcpWindowSize*.

6 Performance Tuning for Low Delay Networks (BDP <= 64kB)

If the calculated BDP is less than 64kB, which is the case for low delay networks, then only the size of the send socket buffer provided by the system is the limiting parameter. This buffer size corresponds to the SO_SNDBUF option of the setsockopt() call. If the send socket buffer size is exceeded then the application will wait until again memory will be available. By default, the size of the send socket buffer is 8192 bytes for systems with more than 19MB RAM which is truly too small and cause a reduced performance.

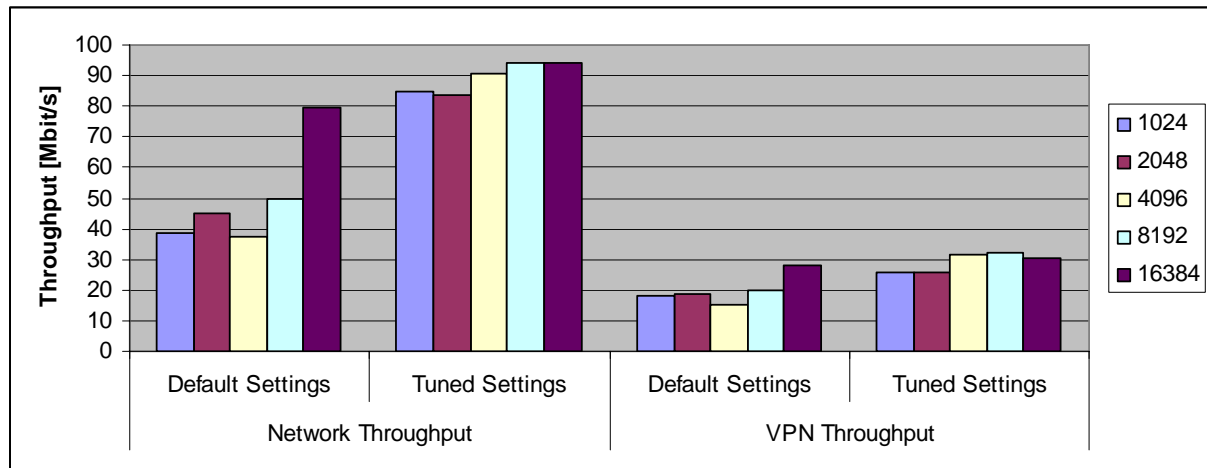
In our test environment we have measured an RTT of 2-3ms.

$$\text{BDP} = (100000 \text{ [kBits/s]} / 8 \text{ [bits]}) * 0,0025 \text{ [s]} = 31,25 \text{ kBytes}$$

The following tables illustrate the network and VPN throughput measured with the default Windows settings and with the tuned parameters (send socket buffer size = 32kB).

Throughput measurement

Packet Size [bytes]	Network Throughput [Mbit/s]		VPN Throughput [Mbit/s]	
	Default Settings	Tuned Settings	Default Settings	Tuned Settings
1024	38,53	85,02	18,08	25,99
2048	45,28	83,81	18,53	25,75
4096	37,40	90,79	15,04	31,73
8192	49,45	94,39	19,75	32,26
16384	79,45	94,39	27,85	30,60
Average	50,02	89,68	19,85	29,27



Optimizing the size of the send socket buffer (32kB) increases the overall performance by a factor of **1,8** (from 50,02Mbit/s to 89,68Mbit/s).

The VPN performance increases by a factor of **1,5** (from 19,85Mbit/s to 29,27Mbit/s). A throughput of 30Mbit/s for larger packet sizes is a good result, considering that the acknowledge packets sent by the remote site also need to be encrypted.

7 Performance Tuning for High Delay Networks (BDP > 64kB)

For supporting large windows the parameter *Tcp1323Opts* needs to be added to the Windows registry on both sites. Then you need to calculate the TCP window size based on the RTT, adjust the Windows registry entry *TcpWindowSize* on the receiver's site and set the send socket buffer size on the sender's site.

7.1 Tuning example for RTT = 12ms

At first we need to calculate the BDP based on the RTT and the peak bandwidth of the link:

$$\text{BDP} = (100000 \text{ [kBits/s]} / 8 \text{ [bits]}) * 0,012 \text{ [s]} = 150 \text{ kBytes}$$

For greatest efficiency, the *TcpWindowSize* should be an even multiple of the TCP *Maximum Segment Size* (MSS). Recording the packets with Ethereal has returned an MSS of 1460 bytes.

$$150000 \text{ [Bytes]} / 1460 \text{ [Bytes]} = 102,74$$

$$\text{TcpWindowSize} = 104 * 1460 \text{ [Bytes]} = 151840 \text{ Bytes}$$

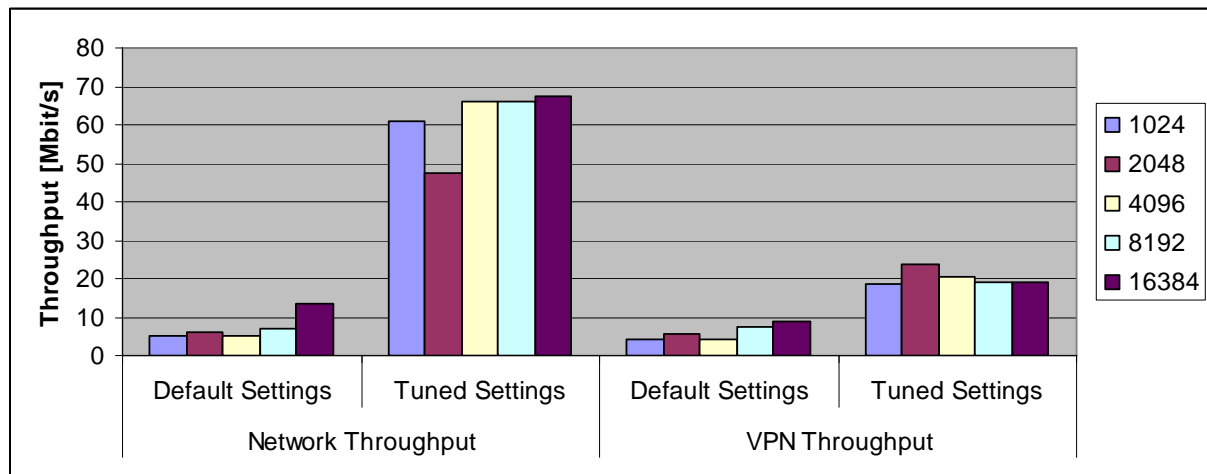
For the send socket buffer size we have used a value of 150kB.

Tuned Windows registry parameters:

Registry entry	Sender	Receiver
DefaultSendWindow	150kB	-
Tcp1323Opts	1	1
TcpWindowSize	-	151840

Throughput measurement:

Packet Size [bytes]	Network Throughput [Mbit/s]		VPN Throughput [Mbit/s]	
	Default Settings	Tuned Settings	Default Settings	Tuned Settings
1024	5,30	61,16	4,41	18,46
2048	6,14	47,29	5,62	23,66
4096	5,30	65,90	4,18	20,60
8192	6,81	65,91	7,40	19,28
16384	13,38	67,54	8,77	19,24
Average	7,39	61,56	6,08	20,25



Tuning the TCP settings increased the overall performance by a factor of **8,3** (from 7,39Mbit/s to 61,56Mbit/s).

The VPN performance has increased by a factor of **3,3** (from 6,08Mbit/s to 20,25Mbit/s).

7.2 Tuning example for RTT = 20ms

At first we need to calculate the BDP based on the RTT and the peak bandwidth of the link:

$$BDP = (100000 \text{ [kBits/s]} / 8 \text{ [bits]}) * 0,02 \text{ [s]} = 250 \text{ kBytes}$$

For greatest efficiency, the *TcpWindowSize* should be an even multiple of the TCP *Maximum Segment Size* (MSS). Recording the packets with Ethereal has returned an MSS of 1460 bytes.

$$250000 \text{ [Bytes]} / 1460 \text{ [Bytes]} = 171,23$$

$$TcpWindowSize = 172 * 1460 \text{ [Bytes]} = 251120 \text{ Bytes}$$

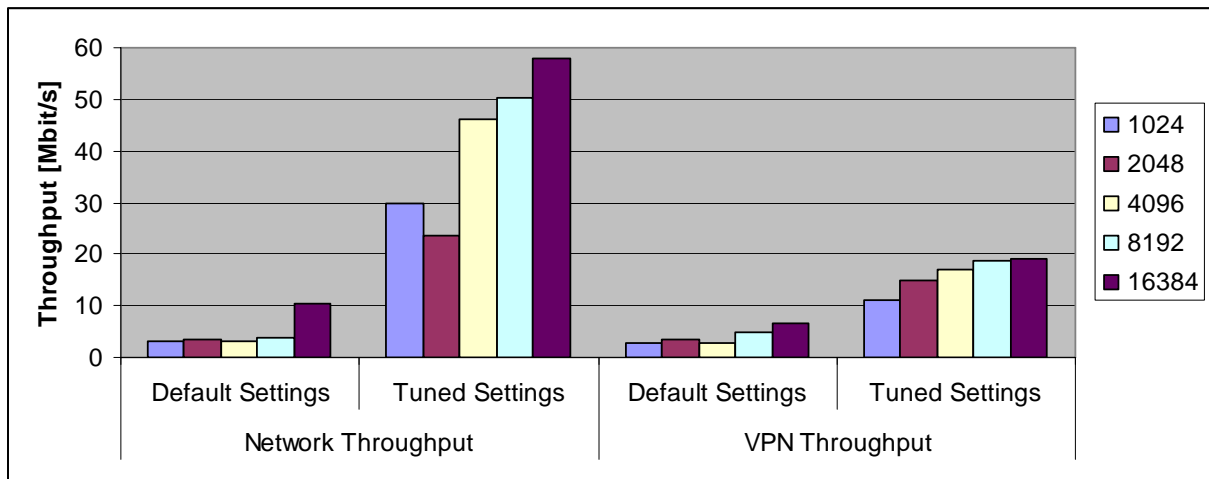
For the send socket buffer size we have used a value of 246kB.

Tuned Windows registry parameters:

Registry entry	Sender	Receiver
DefaultSendWindow	246kB	-
Tcp1323Opts	1	1
TcpWindowSize	-	251120

Throughput measurement:

Packet Size [bytes]	Network Throughput [Mbit/s]		VPN Throughput [Mbit/s]	
	Default Settings	Tuned Settings	Default Settings	Tuned Settings
1024	3,21	29,90	2,81	11,25
2048	3,39	23,58	3,57	15,07
4096	3,22	46,01	2,79	17,14
8192	3,78	50,42	4,95	18,60
16384	10,28	57,79	6,48	18,94
Average	4,78	41,54	4,12	16,20



Tuning the TCP settings increased the overall performance by a factor of **8,7** (from 4,78Mbit/s to 41,54Mbit/s).

The VPN performance has increased by a factor of **3,9** (from 4,12Mbit/s to 16,20Mbit/s).

7.3 Tuning example for RTT = 40ms

At first we need to calculate the BDP based on the RTT and the peak bandwidth of the link:

$$BDP = (100000 \text{ [kBits/s]} / 8 \text{ [bits]}) * 0,04 \text{ [s]} = 500 \text{ kBytes}$$

For greatest efficiency, the *TcpWindowSize* should be an even multiple of the TCP *Maximum Segment Size* (MSS). Recording the packets with Ethereal has returned an MSS of 1460 bytes.

$$500000 \text{ [Bytes]} / 1460 \text{ [Bytes]} = 342,47$$

$$TcpWindowSize = 344 * 1460 \text{ [Bytes]} = 502240 \text{ Bytes}$$

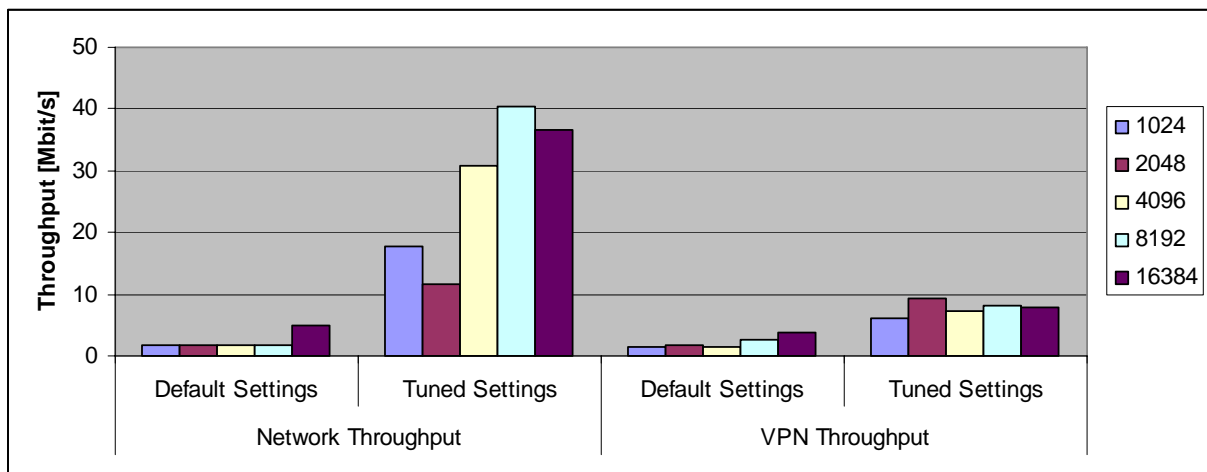
For the send socket buffer size we have used a value of 490kB.

Tuned Windows registry parameters:

Registry entry	Sender	Receiver
DefaultSendWindow	490kB	-
Tcp1323Opts	1	1
TcpWindowSize	-	502240

Throughput measurement:

Packet Size [bytes]	Network Throughput [Mbit/s]		VPN Throughput [Mbit/s]	
	Default Settings	Tuned Settings	Default Settings	Tuned Settings
1024	1,62	17,66	1,51	6,05
2048	1,87	11,55	1,84	9,43
4096	1,62	30,80	1,49	7,21
8192	1,77	40,32	2,76	8,27
16384	4,86	36,55	3,82	7,91
Average	2,35	27,38	2,28	7,77



Tuning the TCP settings increased the overall performance by a factor of **11,7** (from 2,35Mbit/s to 27,38Mbit/s).

The VPN performance has increased by a factor of **3,4** (from 2,28Mbit/s to 7,77Mbit/s).

8 Performance Tuning: Summary

As we have seen in the previous chapters, tuning the Windows TCP settings may increase the overall performance and the VPN performance dramatically. Here is a summary of the required steps:

1. Get the RTT by using the *ping* program.
2. Calculate the BDP according to the formula:
BDP [kBytes] = (bandwidth [kbit/s] / 8 [bits]) * RTT [s]
3. If BDP <= 64kB, specify as send socket buffer size the calculated BDP value. You can do this either through the application, if this is provided, or through the Windows registry entry *DefaultSendWindow*.
4. If BDP > 64kB:
 - Enable large windows on both sites by settings the Windows registry parameter *Tcp1323Opts=1*.
 - On the sending site, specify as send socket buffer size the calculated BDP value. You can do this either through the application, if this is provided, or through the Windows registry entry *DefaultSendWindow*.
 - Retrieve the used MSS for example with Ethereal.
 - On the receiving site, set the Windows registry entry *TcpWindowSize* to the calculated BDP value as an even multiple of the MSS.

9 Appendix A: Windows Registry Parameters

9.1 DefaultSendWindow


Registry Key	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\AFD\Parameters
Value Type	REG_DWORD – Number
Default	4096 for smaller computers (< 19 MB), 8192 for medium computers (< 32MB on Windows 2000 Professional, < 64MB on Windows 2000 Server), 8192 for large computers (> 32 MB on Windows 2000 Professional, >64 MB on Windows 2000 Server).
Description	The number of send bytes that AFD buffers on a connection before imposing flow control. For some applications, a larger value here gives slightly better performance at the expense of increased resource utilization. Applications can modify this value on a per-socket basis with the SO_SNDBUF socket option.

9.2 EnablePMTUDiscovery

Registry Key	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
Value Type	REG_DWORD - Boolean
Valid Range	0, 1 (false, true)
Default	1 (true)
Description	When this parameter is set to 1 (true) TCP attempts to discover the Maximum Transmission Unit (MTU or largest packet size) over the path to a remote host. By discovering the Path MTU and limiting TCP segments to this size, TCP can eliminate fragmentation at routers along the path that connect networks with different MTUs. Fragmentation adversely affects TCP throughput and network congestion. Setting this parameter to 0 causes an MTU of 576 bytes to be used for all connections that are not to hosts on the local subnet.

9.3 MTU

Registry Key	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\[Adapter ID] where "Adapter ID" refers to the services subkey for the specific adapter card.
Value Type	REG_DWORD - Number
Valid Range	88–the MTU of the underlying network
Default	0xFFFFFFFF
Description	This parameter overrides the default Maximum Transmission Unit (MTU) for a network interface. The MTU is the maximum packet size, in bytes, that the transport can transmit over the underlying network. The size includes the transport header. An IP datagram can span multiple packets. Values larger than the default for the underlying network cause the transport to use the network default MTU. Values smaller than 88 cause the transport to use an MTU of 88.

 **Note:** Windows 2000 TCP/IP uses PMTU detection by default and queries the NIC driver to find out what local MTU is supported. Altering the MTU parameter is generally not necessary and may result in reduced performance.

9.4 Tcp1323Opts

Registry Key	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
Value Type	REG_DWORD – Number (flags)
Valid Range	0, 1, 2, 3 0 (disable RFC 1323 options) 1 (window scale enabled only) 2 (timestamps enabled only) 3 (both options enabled)
Default	No value; the default behavior is as follows: do not initiate options but if requested provide them.
Description	This parameter controls RFC 1323 time stamps and window-scaling options. Time stamps and window scaling are enabled by default, but can be manipulated with flag bits. Bit 0 controls window scaling, and bit 1 controls time stamps.

9.5 TcpWindowSize

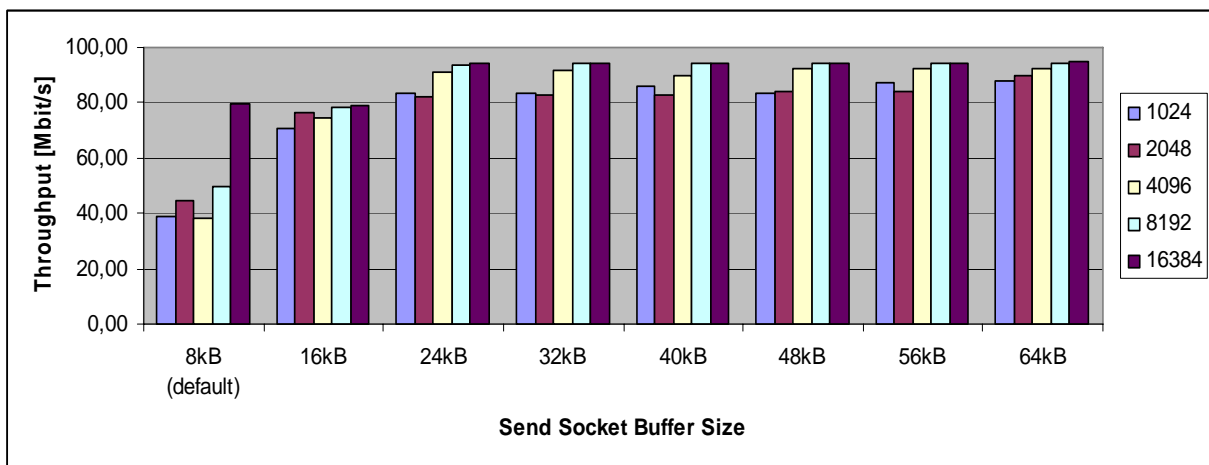
Registry Key	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters or HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\[Adapter ID] where "Adapter ID" refers to the services subkey for the specific adapter card.
Value Type	REG_DWORD – Number of bytes
Valid Range	0–0x3FFFFFFF (1073741823 decimal). In practice the TCP/IP stack will round the number set to the nearest multiple of maximum segment size (MSS). Values greater than 64 KB can be achieved only when connecting to other systems that support RFC 1323 Window Scaling.
Default	The smaller of the following values: - xFFFF - <i>GlobalMaxTcpWindowSize</i> (another registry parameter) - The larger of four times the maximum TCP data size on the network - 16384 rounded up to an even multiple of the network TCP data size The default can start at 17520 for Ethernet, but may shrink slightly when the connection is established to another computer that supports extended TCP head options, such as SACK and TIMESTAMPS, because these options increase the TCP header beyond the usual 20 bytes, leaving slightly less room for data.
Description	This parameter determines the maximum TCP receive window size offered. The receive window specifies the number of bytes that a sender can transmit without receiving an acknowledgment. In general, larger receive windows improve performance over high-delay, high-bandwidth networks. For greatest efficiency, the receive window should be an even multiple of the TCP Maximum Segment Size (MSS). This parameter is both a per-interface parameter and a global parameter, depending upon where the registry key is located. If there is a value for a specific interface, that value overrides the system-wide value.

10 Appendix B: Dependencies

10.1 Influence of the send socket buffer size on the throughput

The following table and diagram illustrate the influence of the send socket buffer size on the throughput. The measured RTT was 2,5ms. The calculated BDP is 32kB.

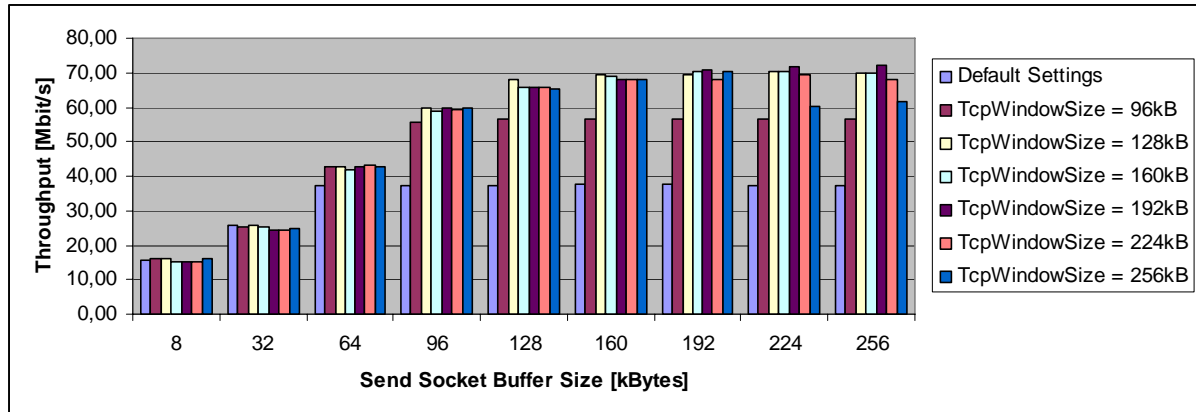
Packet Size [bytes]	Send socket buffer size [kB]							
	8 (default)	16	24	32	40	48	56	64
1024	39,16	70,98	83,52	83,72	85,89	83,18	87,18	87,66
2048	44,62	76,37	82,35	82,91	82,67	83,94	83,97	89,60
4096	38,16	74,36	91,07	91,46	89,65	92,20	92,45	92,29
8192	49,55	78,54	93,85	94,39	94,39	94,39	94,43	94,53
16384	79,93	79,12	94,34	94,39	94,39	94,39	94,57	94,61
Average	50,28	75,87	89,03	89,37	89,40	89,62	90,52	91,74



Setting the size of the send socket buffer to 32kB will increase the overall performance from 50,28Mbit/s to 89,37Mbit/s. Using a higher value than 32kB won't increase the performance much more but will be at the expense of increased resource utilization.

10.2 Influence of the send socket buffer size and the `TcpWindowSize` on the throughput

The following diagram displays the impact of the send socket buffer size and the `TcpWindowSize` on the throughput. We have used a fixed packet size of 16k and modified the size of the send socket buffer size on the sender's site and the value of `TcpWindowSize` on the receiver's site. The `ping` command has returned an RTT of 12ms. The calculated BDP is 150kB.



With the default settings we reach the maximum throughput for a send socket buffer size of 64kBytes. Using a higher value won't increase the performance because in this case the limiting factor is the `TcpWindowSize` of the receiver's site, which is limited to 64kBytes.

With large window support enabled (Windows registry parameter `Tcp1323Opts`) we reach an optimal throughput when using the calculated BDP value as send socket buffer size on the sender's site and as `TcpWindowSize` on the receiver's site. Using a higher value for `TcpWindowSize` does not increase the throughput.

11 Appendix C: References

- RFC 1191 - Path MTU Discovery
- RFC 1323 - TCP Extensions for High Performance
- RFC 1242 - Benchmarking Terminology for Network Interconnection Devices
- RFC 1944 - Benchmarking Methodology for Network Interconnect Devices
- Microsoft Windows 2000 TCP/IP Implementation Details (White Paper)