Teamcenter
Network Performance Tuning

SIEMENS

White Paper

A Technical White paper describing Teamcenter's use of TCP, typical network performance issue analysis and solutions.

This paper outlines the TCP (Transmission and Control Protocol) and describes how to tune TCP to gain better performance. WAN connections with high-BDP (Bandwidth Delay Product) are discussed along with tuning TCP to improve performance on these connections. In practice the use of a network acceleration device such those from Riverbed, Cisco and Blue Coat are the simplest and most effective method of utilizing high-BDP connections. The latest versions of Windows and Linux provide some of the new TCP mechanisms to deal with high-BDP networks but of course do not provide the compression, caching etc., of the WAN accelerators.

Revision History

Document Version #	Approval Date	Modified By	Revisions
6.0	01 May 2013	David Howe	Reformat and additional debugging
6.1	08 Dec 2013	David Howe	FSC Send and Receive buffer tuning. VPN IPsec performance
6.2	18 Dec 2013	David Howe	Updated WAN acceleration configuration information.
6.3	07 Jan 2014	David Howe	Clarified Sysinternals tools descriptions and Teaming
6.4	21 Jan 2014	David Howe	Updated Windows details and add new performance analysis methods
6.5	25 Feb 2014	David Howe	Update Oracle SDU details
6.6	06 March 2014	David Howe	Added Windows Server 2012 R2 and Windows 8.1 details. Plus iPerf

Table of Contents

1 Teamcenter and the Network

As a network distributed application Teamcenter performance is dependent on the performance of the underlying network infrastructure. Not only is file delivery a performance issue but in a four-tier implementation so is the delivery of the metadata.

Latency and bandwidth are the two factors that determine your network connection's ability to transfer data efficiently.

The management of sockets within a system is also a source of problems. If these parameters are not tuned heavily loaded Web application servers, HTTP servers and File Management Servers (FMS) cache servers experience problems at peak load. Typically performance slows down and some users experience connection refusals. Therefore it is very important that TIME_WAIT and other such parameters are tuned. The mechanism to do this for each of our supported operating system is described in section 5 onward.

This paper outlines the TCP (Transmission and Control Protocol) and describes how to tune TCP to gain better performance. WAN connections with high-BDP (Bandwidth Delay Product) are discussed along with tuning TCP to improve performance on these connections. In practice the use of a network acceleration device such those from Riverbed, Cisco and Blue Coat are the simplest and most effective method of utilizing high-BDP connections. The latest versions of Windows and Linux provide some of the new TCP mechanisms to deal with high-BDP networks but of course do not provide the compression, caching etc., of the WAN accelerators.

1.1 Network Performance Checklist

1) Physical Connections and NICs:
   - Make sure connections are configured correctly preferably in autoconfiguration mode. Misconfiguration can cause lost data and degraded throughput. If any data loss is experienced this should be checked throughout the network between the server and the problematic client. See section 4.1 for background.
   - Check NICs and switch ports etc., for error logging. This can be a sign of performance issues.
   - Use NICs with TCP off-loading capabilities
   - Monitor switches and OS for packet loss
   - If you are using GigE networks use Jumbo Frames if all connections support this. 10 /100 Mb/s devices do not support Jumbo frames and not all Ethernet switches support them.

2) Disable Nagle (see section 2.2.3 and specific operating system configuration 5 onwards) on:
   - Web Application Servers
   - Teamcenter Enterprise Servers
   - Teamcenter FSC Servers

3) WAN/LAN: configure your server operating systems to support the highest BDP in your network. See section 3 and section 5 onwards for specific operating system details. On higher BDP, congested and poor networks consider using a WAN accelerator, see section 4.6. Configure:
   - Clients
   - Web Application Server
   - Teamcenter Enterprise Servers
   - FSC Servers (don't forget to check ulimit requirements) and tune buffers, see section 5.3.4 for FSC buffer tuning.
   - Oracle Servers

4) Tune Oracle Net. See section 4.5 for details
   - Make sure Nagel is disabled (default is off)
   - Set the SDU size
   - Set the receive buffers
   - Set the send buffer

Losing Packets? The check:
   - Check port physical connections (see above)
   - Check MTU/ICMP section 2.2

2 Transmission Control Protocol

TCP is often referred to TCP/IP. IP (Internet Protocol) is concerned with the way a message makes its way across the Internet or your own Intranet (internal network). The address for a computer, for example "134.244.154.35" is an IP address, to be exact it is an IPv4 address. Because of limits in the address range a new addressing system was developed called IPv6 which so far has had limited take up and is not yet part of this discussion. The definition of TCP is managed by the Internet Engineering Task Force (IETF) and is defined by a series of RFCs (Request For Comments) see http://www.ietf.org/rfc.html .

TCP is used to control the communication between two end systems for example a browser and the server. It is not concern about how their packets are routed or addressed that is part of IPs function. TCP was developed to provide reliable, ordered delivery of data from a program on one computer to another program on another computer. A key point is TCP is concerned with accurate deliver and not necessarily performance.

The intention of this section is not to cover all the protocol and responses you will find in a network analysis but to provide enough information to be able to follow a TCP dialogue. Throughout this section example network traces from Wireshark are provided. Wireshark is a network protocol analysis tool available under a GNU General Public License from http://www.wireshark.org .

2.1 TCP State Changes

When establishing or closing a connection, the end points exchange various packets and passes through different states to indicate the state of the connection. You can see the current connection states using the "**netstat**" command. There are 11 connections state:

1. LISTEN: waiting for a connection request from any remote client.
2. SYN-SENT: waiting for a matching connection request after sending a connection request.
3. SYN_RECV received a connection request, sent ACK, waiting for final ACK in three-way handshake (see below).
4. ESTABLISHED connection established
5. FIN_WAIT_1 this side of the connection has shutdown, now waiting to complete transmission of remaining buffered data.
6. FIN_WAIT_2 all buffered data sent and acknowledged, waiting for remote to shutdown
7. CLOSING both sides have shutdown but we still have data we have to finish sending
8. TIME_WAIT timeout to catch resent data before entering closed.
9. CLOSE_WAIT remote side has shutdown and is waiting for us to finish writing our data and to shutdown. Socket must be closed to exit CLOSE_WAIT.

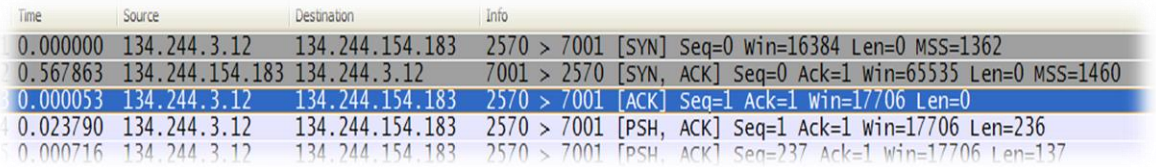
10. LAST_ACK connection has shut down after remote has shutdown. There may still be data in the buffer that the process has to finish sending
11. CLOSE represents no connection state at all. This is a logical state and one you will not see in "`netstat`" etc.

2.1.1 Establishing a connection

The term client and server are used here to describe the initiator and receiver rather than the role of the processes concern. A TCP connection always starts with the "three-way handshake". The dialogue is used to open the connection in both directions. The initial SYN request opening a conversation to the server and the replying with a SYN, ACK packet header, this acknowledges the requests and requests connection to the client, which is acknowledged with an ACK.

Client State		Server State
CLOSED		LISTENING
SYN SENT	SYN →	
	← SYN ACK	SYN RECIEVED
ESTABLISHED	ACK →	ESTABLISHED

Connection state changes during the initial connection sequence



Above is an extract of Wireshark trace showing a connection being established?

2.1.2 Closing a connection

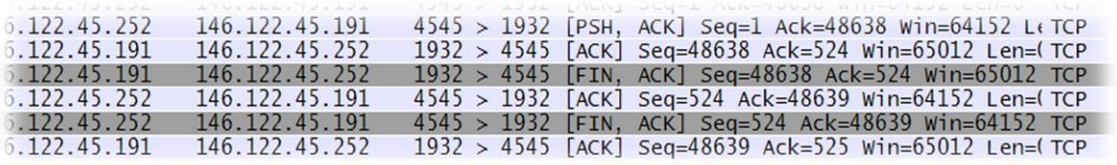
Again the role of the client and the server are not rigid as a server may also close a connection, but this is not the norm. The close sequence is a closure of both directions. The client requests a close with a FIN, the server acknowledges the FIN with an ACK. Once the application releases the socket, the server sends a second FIN, to which the client responds with a final ACK. The FIN is often combined with the last ACK of a request creating a FIN, ACK packet.

Client State		Server State
ESTABLISHED		ESTABLISHED
FIN_WAIT_1	FIN →	
	← ACK	CLOSE_WAIT
FIN_WAIT_2	← FIN	LAST_ACK
TIME_WAIT	ACK →	TIME_WAIT

Connection state changes during the connection closure sequence

The server will not send a second FIN until the application has released the socket. It is not uncommon to see a FIN_WAIT_2 and CLOSE_WAIT pair waiting for

an application to release the socket. Closing the owning process will release the socket from CLOSE_WAIT. The timeout for CLOSE_WAIT varies but is generally many hours or sometime not until the owning process closes.



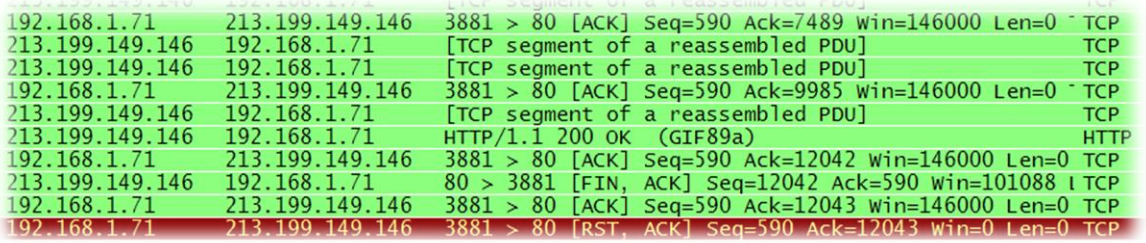
Above is an extract of Wireshark trace showing a connection closure sequence. Note the initial FIN is combined with an ACK for the previous packet.

2.1.3 Connection RESET (RST)

A process can abort a connection at anytime by sending an RST packet. The receiving and sending socket will go straight to a TIME_WAIT state.

What causes resets? A number of issues can cause the system to reset. They have many causes, not all are problems. Some examples are:

- Connection rejection – when a server has a resource issues, for example no sockets are available, the server will send a RST. You will see the initial SYN from the client responded to by RST. Typically the client will retry and subsequently succeed. In a trace of a busy network this is a fairly common sequence. It can show that the TCP resources of the server require tuning. Requests for services that are not available on that server will receive an ICMP (Internet Control Message Protocol) packet as a response describing the issue. A table of ICMP packets type is included in section 2.2.4.
- Data loss – when packets are lost, retries are made for about 9 minutes and then connection is reset. The total time out is not usually configurable. The time results from the "exponential backup" algorithm used for retrying, the first retry is after about a one second, after this the timeout value is doubled for each retransmission, with an upper limit of 64 seconds. The total time also depends on the packets Time to Live (TTL). So on a connection with many hops the total time will be longer. So typically on a WAN the timeout is about 10 mins.
- Unexpected packets received mid transfer can cause the receiving system to reset.
- An application crashing can also cause connection resets.
- Some Firewalls and application use resets to close connections to improve performance and to avoid CLOSE_WAIT states.

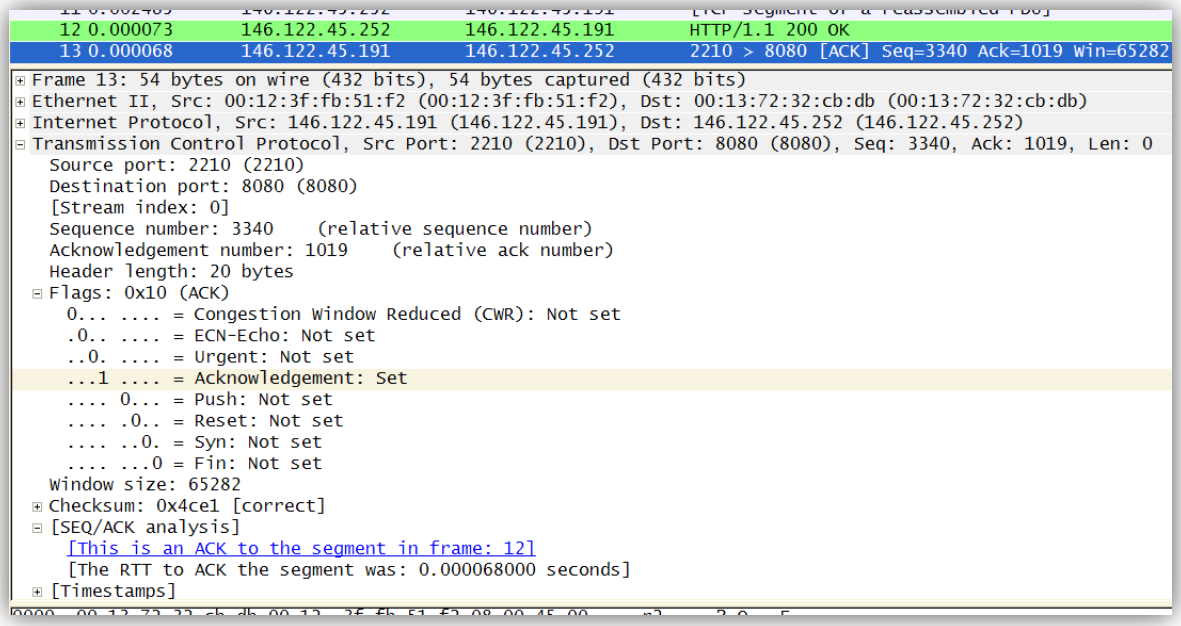


Above is an extract of Wireshark trace showing of IE8 using a reset to close a connection after loading a GIF file. This sequence is probably used to speed closure as it removes trips and eliminates possible CLOSE_WAIT conditions due to lost packets.

2.1.4 Data Packet Acknowledgement (ACK)

All network packets received are acknowledged by an ACK. Each packet is not immediately acknowledged, that is you will not always see a packet sent and the receiver acknowledging it. Acknowledgment is controlled by the TCP window size and other settings discussed in this paper.

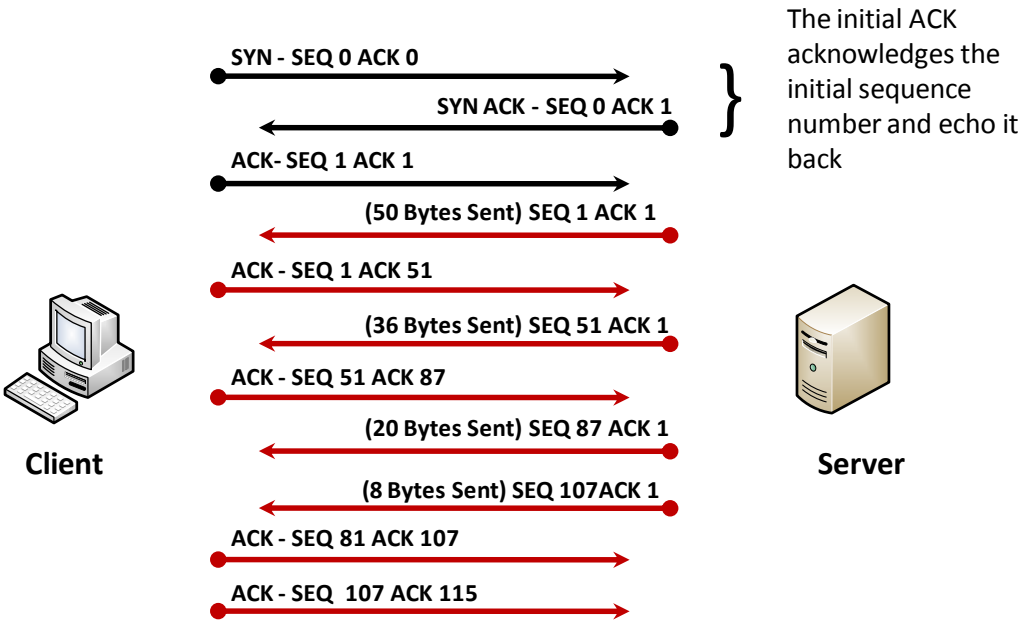
In Wireshark traces ACKs can be easily identified as shown below. A TCP packet can have many flags set, below only the ACK flags is set many this just an ACK packet. However many be set for example both SYN and ACK many a SYN/ACK packet.



A simple connection and acknowledgement sequence is show below. The initial SYN sets the initial sequence number; the subsequent packet's sequence number is the total number of bytes sent this session plus one to give the number of the first byte of this packet. The receiver acknowledges with an acknowledgement packet (ACK) number set to the anticipated value of the sequence number of the

next packet the server will send. The sequence number of the next packet is the sequence number of the received packet plus the data length. In the example below the last two packets are set before acknowledgement and consequently acknowledged after the second send, this is normal behaviour.

HTTP is built on TCP and uses this method but in some request it can be confusing to follow. Typically a request is made using an HTTP POST; this is a TCP PUSH/ACK that is both the ACK and PUSH flags are set. The PUSH flag tell the system to pass the data in the packet to the receiving application. The server will respond with HTTP 200, this is again is a PUSH/ACK which acknowledges the initial request and ask for the message (HTTP 200) to be passed to the application. The client acknowledges this packet is an ACK and then the server responds with the requested data. The initial data packet is a PUSH/ACK to request the data is passed to the application but subsequent packets just have the ACK lags set. Each of these data packets are acknowledged in the normal way.



Simple packet acknowledgement

If the receiver detects a gap in the sequence numbers, it will generate a duplicate ACK for each subsequent packet it receives on that connection, until the missing packet is successfully received (retransmitted).. If the sender receives more than three ACKs for the same packet (Wireshark reports these as duplicate ACKs) the packet is resent. See section 2.1.3 for details of packet loss handling. Duplicate ACKs in a trace indicate l missing or dropped packets. If there are large numbers of duplicate ACKs, it can be an indicator of a problem with network connection configuration (see section 5.1 for NIC configuration issues)

2.1.5 Using the Packet ACK to determine system performance

Performance issues are often blamed on Teamcenter when the issue real lies in the network. But how can you tell whether it is the network, latency or Teamcenter that is the issue? Measuring network latency vs. application latency will help identify where the problem lies. Packet-level analysis using Wireshark allows visual inspection of packet-level conversation between a client and the Teamcenter server. It is possible to see whether the network is the source of delay, or if the problem is in the Teamcenter server. This can be done by comparing the responsiveness of the ACK or initial HTTP 200 response to a client request, compared to the following response that includes data. The network should acknowledge the client request quickly (about the same time as one way latency) while the Teamcenter may take tenths of seconds or even seconds to respond with payload data. A slow HTTP 200 response to a request may indicate ether a network issue or a Web Application server problem. The data return response depends on the request so some judgment of the request must be used.

2.2 MTU (Maximum Transfer Unit) and Fragmentation

Intranets or the Internet are in fact a collection of small networks linked together, the packets are passing through routers, which connect these small networks together to form a path through the network.

Each system/router connected to the network is configured to accept a certain maximum packet size or Maximum Transfer Unit (MTU). If a system or network device receives a packet that is larger than its MTU, then it tries to fragment the packet up in smaller packets, or if that is not possible returns an ICMP "cannot fragment" error, failing that it just drops the packet.

The MTU (Maximum Transport Unit) is the amount of data a packet carries at the transport layer and varies for type of transport, for example for Ethernet the MTU is 1500. On Ethernet the TCP MSS (Maximum Segment Size) is typically set to 1460, which is the MTU minus 40 bytes for IP and TCP headers. Some UNIX systems include a TCP timestamp which reduces the MSS to 1440 bytes.

Network Transport Type	MTU (bytes) Details
Ethernet	1500
IEEE 802.3/802.2	1492
PPPoE	1492
Optimal MTU for PPPoE over ATM	1454
X.25 Dialup Connection	576
Ethernet (Jumbo) Gigabit networking	>=9000

Note: VPN creates an encryption overhead (encryption, padding, checksum, etc.), which adds about 100 bytes to the TCPIP-payload. This overhead must not exceed the MTU of the connection as the packets will become fragmented. When the other side receives an encrypted packet, it inspects the checksum and it will fail if

the packet has been tampered with (fragmented). See Section 2.2.2 for potential performance issues.

The MTU size for the connection path is dependent on the systems and network device e.g. routers that the packet has to travel through to reach its destination. During the initial connection set-up, as part of the three way handshake, the MSS (Maximum Segment Size) of the systems will be negotiated and set as the limit for the MSS for that connection. But the path MTU, that is the smallest MTU of any link on the path between the two hosts may vary over time as the network route changes and may be smaller than that supported by the hosts and set during the connection. Potentially this would lead to fragmentation and loss of performance.

The "do not fragment" flag is set in TCP packets to tell intermediate network devices such as routers, not to repackage the data. This flag is set when packets are sent outside of the local IP network area. If the MTU size on a network device is smaller than the MTU of the sent packet the receiving device should respond with an ICMP "cannot fragment" error (packet type 3 code 4). This tells the sender to use a smaller MTU size. A consequence of this behaviour is the MTU size and other settings are handled transparently.

Fragmentation can be seen using `netat –e –s l`ook in the IPv4 section look for `"Reassembly Required"` and `"Fragments Created"`

2.2.1 Packet loss

Problems can occur when the network filters ICMP. Many network administrators have decided to filter ICMP at a router or firewall. There are valid (and many invalid) reasons for doing this, however it can cause problems. ICMP is an integral part of the Internet protocol and cannot be filtered without due consideration for the effects. In this case, if the ICMP "cannot fragment" errors cannot get back to the source host due to a filter, the host will never know that the packets it is sending are too large. This means it will keep trying to send the same large packets and the will keep being dropped silently. The sending system will keep retrying to no avail until it times out and the receiving process will just hang until it timeouts or is killed.

If you suspect problems with the MTU size on the network path, or if you are losing packets, then the **ping** utility can be used to help determine the cause of the problem. The **ping** utility uses ICMP requests to measure the latency. The latency measured is that of ICMP which may be different from that of TCP as some network devices may be configured to block, give low priority to or spoof (send a dummy return) ICMP packets. I do not want to devalue **ping** because it is a useful tool but spoofing etc., is always possibility that you should to be aware of when you look at the results. Look at the Sysinternals **psping** utility's TCP mode (see Appendix 3)

The **ping** utility can be told to send packets with a certain size to a certain destination address. **Ping** also has an option of disallowing the systems/routers in between the origin and destination to chop/fragment the packets. So if the packets are too large to handle, it will cause the system/router to fail handling the packets. This allows us to use **ping** to find the MTU for a certain connection.

```
ping <IP-Address> -f -l <Packet-size-28>
```

The usual minimum Internet standard packet size is 576 bytes and the maximum is 1500 bytes. When testing with the **ping** you have to subtract 28 from the packet size being tested. This is because **ping** itself adds an IP Header (20 Bytes) and an ICMP-Header (8 Bytes), which together is 28 bytes.

To test the MTU size of a connection to a system or router you are interested in, **ping** the IP-Address with different Packet-Sizes. Keep doing this until you find the packet size which is one byte too large for the network path to handle. This will give the maximum size of the MTU that can be used on this network path.

The MTU can be changed from the default on most systems. On Windows platforms since Windows 2000 (still valid for Windows 8.1 and Windows Server 2012 R2) you can use the **netsh** command to give a fixed MTU rather than use the default path MTU discovery:

Display MTU using:

```
netsh interface ip show interface
```

Change the MTU using:

```
netsh interface ipv4 set subinterface "Local Area
Connection" mtu=nnnn store=persistent
```

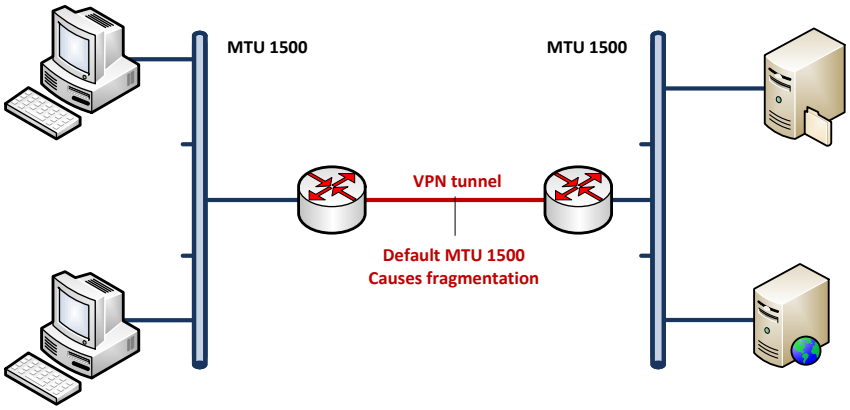
On UNIX system the MTU can usually be set using **ifconfig**:

```
ifconfig le0 mtu 1492
```

Note: When using ping to determine MTU size, the actual MTU require will be the ping size plus 28 Bytes for IP and ICMP header.

2.2.2 VPN IPsec Gateway performance - Fragmentation and MTU

As noted earlier, when a packet is nearly the size of the maximum transmission unit (MTU) of the physical connection of the encrypting device and it is encapsulated with IPsec headers, it will probably exceed the MTU of the connection. This situation causes the packet to be fragmented after encryption (post-fragmentation), which requires the IPsec peer to perform reassembly before decryption, degrading its performance.



This issue is most likely to occur when a WAN link between two sites (networks or sub-nets) is over a VPN gateway/tunnel client. This is less likely to be an issue with a software VPN client to a VPN gateway.

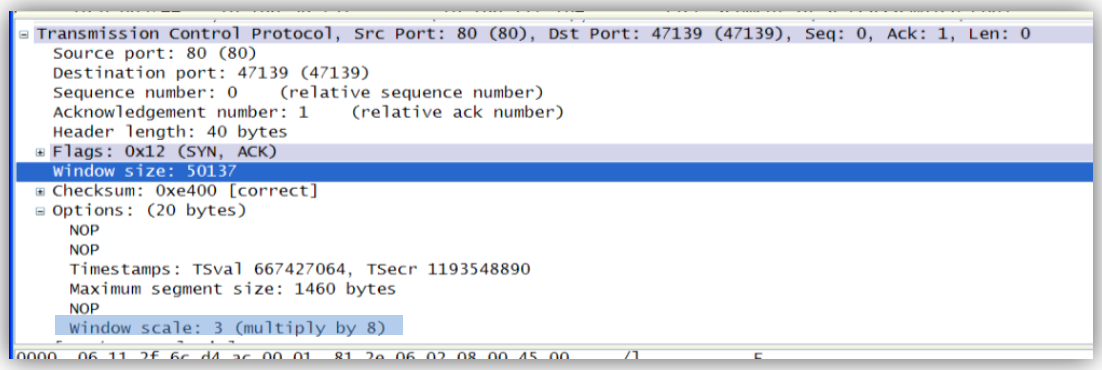
To minimize post-fragmentation, you have two options. The first is to increase the MTU size used on the VPN link to allow for the Ethernet MTU plus IPsec headers. The second is to set the MTU in the upstream data path to ensure that most fragmentation occurs before encryption (prefragmentation). In other words decrease the client MTU size. Reducing the client MTU will cause prefragmentation for the IPsec VPN(s) avoiding performance degradation by effectively shifting the reassembly task from the receiving IPsec peer to the receiving end hosts.

2.3 TCP windows

The TCP window is the maximum amount of data allowed to be sent out on the network before an acknowledgement is received. The size of window used is controlled by the receiver.

If the number of outstanding packets exceeds the window size then a CWR (congestion window reduce) packet is sent by the receiver and the data is resent using a new window that is 50% of the previous. Usually the system will try resetting the window size 5 times before resetting the connection.

The window size is advertised in the ACK (and SYNC, ACK) packet header. Below is an example from Wireshark showing the window size and scaling multiplier. The window size is often displayed scaled in tools like Wireshark to make life easier. Large windows are being advertised via this mechanism.



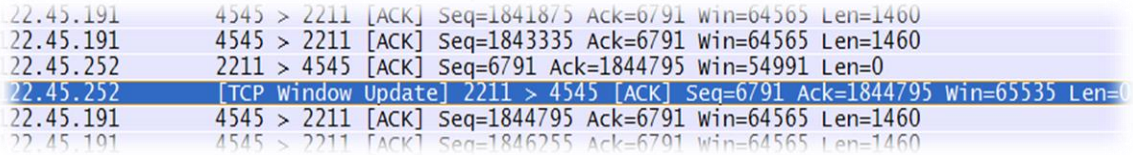
A Wireshark trace showing both window size and the scaling factor.

2.3.1 Window scaling

The amount of space in the TCP header reserved for the window size is restricted to 2 bytes so the maximum values that can be held is 65535 which is much smaller than the buffer space available on modern system. To address this, RFC 1323 introduced scaling which is simply a multiplier that use used with the window size to arrive at the buffer.

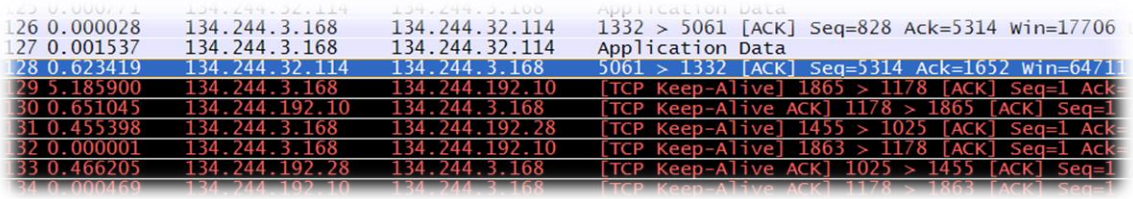
During transmission of data, the ACK packet tells the sending system the maximum buffer available which in turn is used to determine the maximum window size.

When analysing a trace, you may see what appears to be a duplicate ACK to a received packet. Closer inspection will show the window size has been increased. This occurs when the available buffer on the receiving system has increased, so it sends a “window update packet” to inform the sending system more space is available. This is an indicator the receiving system is under load.



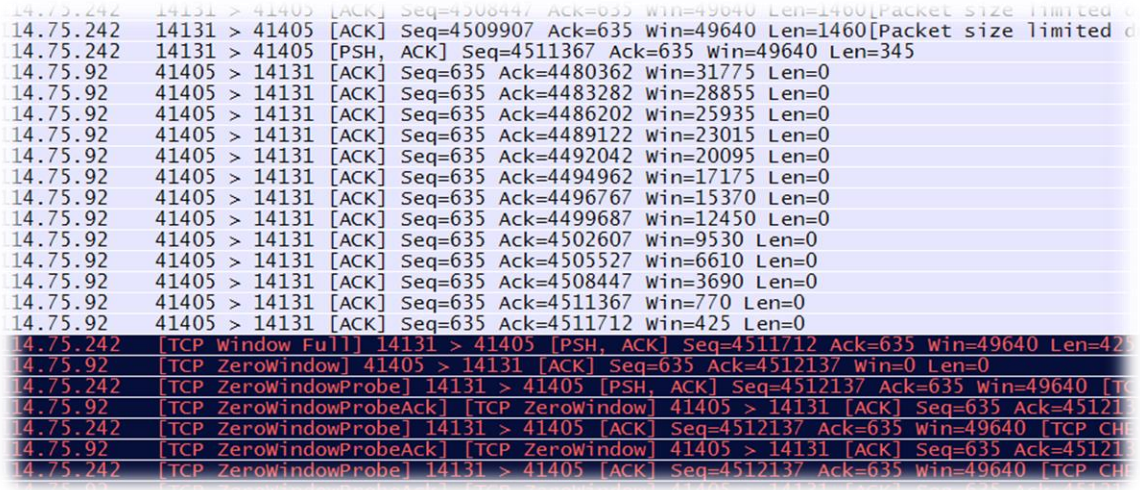
Above is an extract of a Wireshark trace showing a TCP window update. Note the sequence and acknowledgement numbers are as the previous packet but the window has been increased.

In extreme cases, the responding ACK may specify a window size of zero; this is known as a "TCP keep alive" packet and tells the other end of the connection to wait. The sending system will acknowledge with an ACK. The sequence may be repeated many times until the receiving system has free space. This occurs mainly when the receiving system is under load and has insufficient receive buffer space defined. Increasing the buffer space available will improve performance by eliminating these waits. The process for increasing the send and receive buffer are detail for each supported operating system later in this paper.



Above is an extract of Wireshark trace showing a sequence of TCP Keep alive packets being used to pause the download while the receiving buffer is full.

Another example of a window scaling issue is "TCP ZeroWindowProbe" which is another form of keep alive. The example below is typical; the window size is decreasing with each repeated ACK (window reset) until the window size, 425 bytes. The minimum useable window size is one segment, 1460 bytes; at 425 bytes, the window is too small to send any data so the sending system responds with "TCP Window Full". To which the receiver responds "TCP Zerowindow". From now on the sending system will probe the receiver to ask what window space is available by sending a "TCP ZeroWindowProbe" to which receiver responds with a "TCP ZeroWindowProbeAck" that contains the window status. The "exponential backup" algorithm used for retrying the probes, the first retry is after about a 0.5 second, after this the timeout value is doubled for each new probe. This continues until either the window issue is resolved or one side or the other resets the connection. Normally the condition passes and just slows down traffic. In some traces, the problem is caused by the process on the receiving system terminating through some sort of error. Normally increasing the buffer space available will improve performance by eliminating the waits cause by these probes. The process for increasing the send and receive buffer are detail for each supported operating system later in this paper.



Above is an extract of Wireshark trace showing a sequence of TCP ZeroWindowProbe packets.

2.3.2 TCP Slow Start

The TCP slow start algorithm (RFC 2001 and RFC 3742) is used to find the largest window size a network can use for each transmission. This algorithm starts with a small window size and increases it after each successful window is sent until either a packet is lost and a CWR packet is returned or the maximum window size is reached.

2.3.3 The Nagel algorithm

The Nagel algorithm (named after John Nagel and defined by RFC 896 http://www.faqs.org/rfcs/rfc896.html) was introduced to help congested networks and to supersede the older method of simply applying a delay of 299-500 ms on all small packets. The algorithm tries to combine a number of small outgoing messages destined for the same address into a single packet to reduce the number of packets on the network. As long as there is a sent packet for which the sender has received no acknowledgment, the sender should keep the new packet in the buffer until it has a full packet's worth of output to send or positive acknowledgement is received. The effect is to delay some small packets but it can have a disproportionately adversely effect on large data transfers.

A pseudo code example of the Nagle algorithm is as follows;

```
if there is new data to send
  if the window size >= MSS and available data is >= MSS
    send complete MSS segment now
  else
    if there is data sent but not yet acknowledged
      enqueue data in the buffer until an acknowledge is
received
    else
```

```
      send data immediately
    end if
  end if
end if
```

Introduced at around the same time by RFC 1122, was a concept of delayed acknowledgement (delayed ACK). As with Nagel it was introduced to help with congested networks. The effect of RFC 112 is received packets are acknowledged in two's. When a single packet is received, the system will wait 200ms for a second packet before sending an ACK. When combined with Nagel the result can be a 200 ms delay on receiving data.

The problem occurs when the data sent is divided into data segments (MSS) for transmission results and in a small final packet. The effect of Nagel is a small packet will not be sent until outstanding data has been acknowledged, that is an ACK has been received. So if the last packet sent was the first in the sequence, the delayed ACK algorithm will wait 200ms before sending the ACK to release the small packet. The effect can be serious and consequently most operating systems allow Nagel to be disabled globally and also allow it to be disabled programmatically. The parameter to do this varies but usually follows the pattern "Nagel_lim_def", details where this is available are given in section 5 onwards.

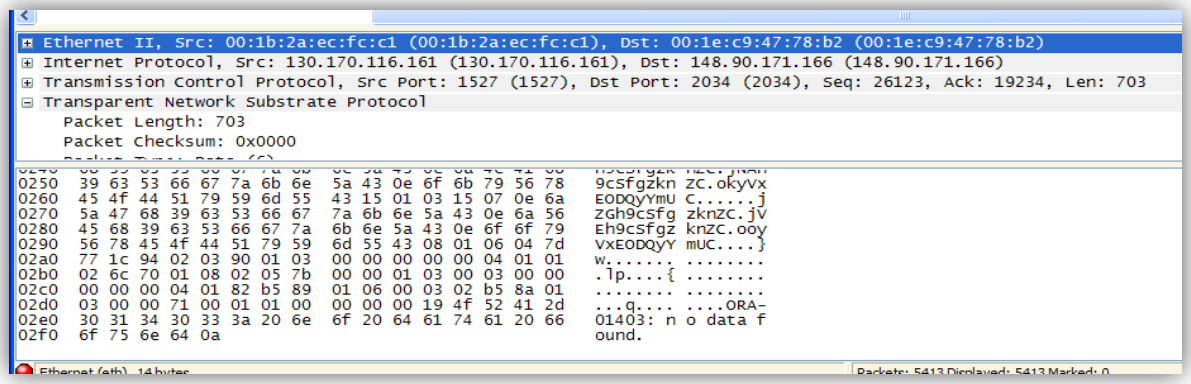
Another option is to disable the delayed ACK algorithm; for example on AIX set TCP_NODELAYACK. This will counter the effect but can have adverse effects on high-latency networks for which it was designed.

With any network application, testing the effect of disabling Nagel is simple and can give large returns. The effects of Nagel can apply to the communications between all Teamcenter at all tiers and should be seriously considered as a source of delay.

2.4 Analyzing Oracle Network Traffic

As with most other protocols, Wireshark provides a dissector for the Oracle network protocol TNS. It can be found under Analyze > Decode As... > Transport > TNS. Highlight an Oracle and select the TNS dissector to get the records the stream decoded.

One curious feature of Oracle traffic is the returned data from a "Select" is terminated with an "ORA=1403: no data found", regardless as to whether data rows were returned or not. It appears to be an end of data message. The example below is the result of a Select on PPEXRESSIONS. A single record is returned and terminated by ORA-1403.



2.5 ICMP (Internet Control Message Protocol) packets

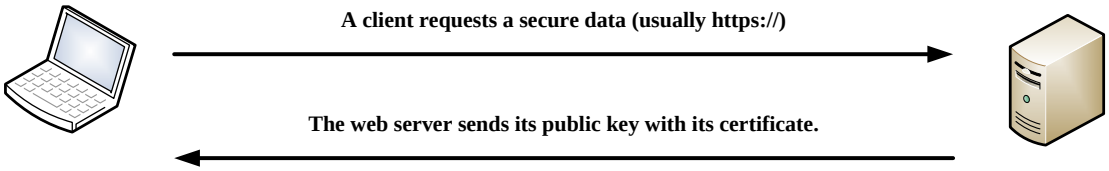
TYPE	CODE	Description	Query	Error
0	0	Echo Reply	x	
3	0	Network Unreachable		x
3	1	Host Unreachable		x
3	2	Protocol Unreachable		x
3	3	Port Unreachable		x
3	4	Fragmentation needed but no frag. bit set		x
3	5	Source routing failed		x
3	6	Destination network unknown		x
3	7	Destination host unknown		x
3	8	Source host isolated (obsolete)		x
3	9	Destination network administratively prohibited		x
3	10	Destination host administratively prohibited		x
3	11	Network unreachable for TOS (type of service)		x
3	12	Host unreachable for TOS		x
3	13	Communication administratively prohibited by filtering		x
3	14	Host precedence violation		x
3	15	Precedence cutoff in effect		x
4	0	Source quench		
5	0	Redirect for network		
5	1	Redirect for host		
5	2	Redirect for TOS and network		
5	3	Redirect for TOS and host		
8	0	Echo request	x	
9	0	Router advertisement		
10	0	Route solicitation		
11	0	TTL equals 0 during transit		x
11	1	TTL equals 0 during reassembly		x
12	0	IP header bad (catchall error)		x
12	1	Required options missing		x
13	0	Timestamp request (obsolete)	x	
14		Timestamp reply (obsolete)	x	
15	0	Information request (obsolete)	x	
16	0	Information reply (obsolete)	x	
17	0	Address mask request	x	
18	0	Address mask reply	x	

Note: Many network administrators have decided to filter ICMP at a router or firewall. There are valid (and many invalid) reasons for doing this, however it can cause problems such as packet loss. ICMP is an integral part of the Internet protocol and cannot be filtered without due consideration for the effects.

2.6 SSL/HTTPS

The Secure Socket Layer (SSL) protocol was created by Netscape to ensure secure transactions between web servers and browsers. The protocol uses a third party Certificate Authority (CA), to identify one end or both ends of the transaction. SSL allows web applications and web servers to communicate over a secure connection. In this secure connection, the data that is being sent is encrypted before being sending and is decrypted upon receipt and before processing. Both the client and the server encrypt all traffic before sending any data. SSL addresses the following important security considerations.

- **Authentication:** During your initial attempt to communicate with a web server over a secure connection, that server will present your web browser with a set of credentials in the form of a server certificate. The purpose of the certificate is to verify that the site is who and what it claims to be. In some cases, the server may request a certificate from the client to prove the client is who and what it claims to be (which is known as client or two way authentication).
- **Confidentiality:** When data is being passed between the client and the server on a network, third parties can view and intercept this data. SSL responses are encrypted so that the data cannot be deciphered by the third party and the data remains confidential.
- **Integrity:** When data is being passed between the client and the server on a network, it is possible for third parties to intercept this data. SSL ensure that the data is not viewed or modified in transit by that third party.



A client requests a secure data (usually https://)

The web server sends its public key with its certificate.

The client checks that the certificate was issued by a trusted party (usually a trusted root CA), that the certificate is still valid and that the certificate is related to the site contacted.

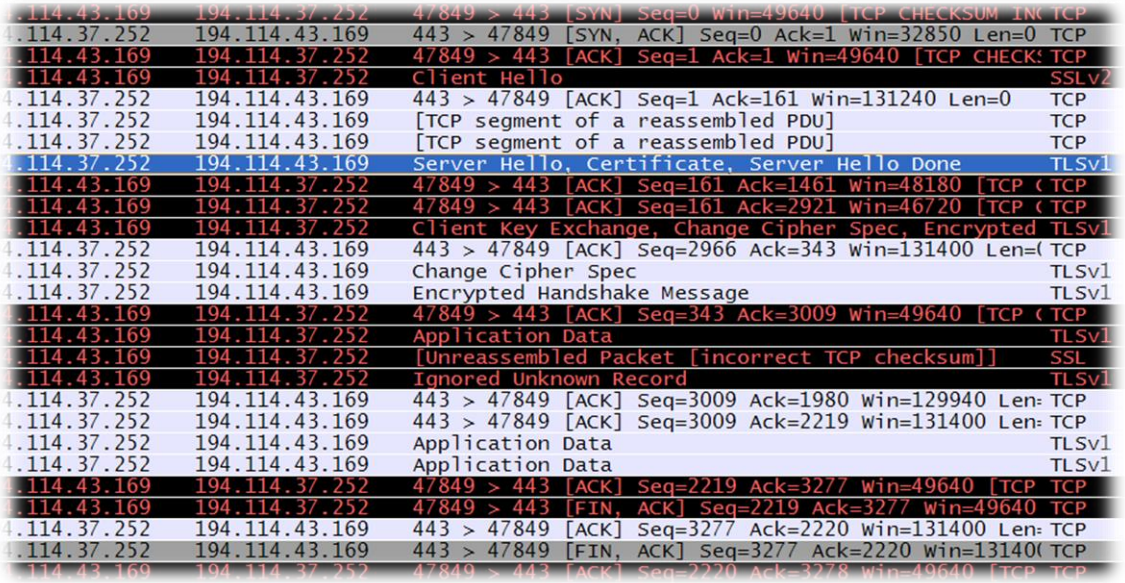
The client then uses the public key, to encrypt a random symmetric encryption key and sends it to the server with the encrypted URL required as well as other encrypted http data.

The web server decrypts the symmetric encryption key using its private key and uses the symmetric key to decrypt the URL and http data.

The web server sends back the requested html document and http data encrypted with the symmetric key.

The application decrypts the http data and html data using the symmetric key and displays the information.

SSL/TLS connection sequence.



Above is an extract of Wireshark trace showing a short HTTPS session: application data is of course encrypted and shows in Wireshark as just application data. The system is using TLSv1. Note: packet reassembly is a feature of Wireshark and not a sign of fragmentation. Wireshark assembles large messages of many packet to make it easier to interpret, this is an option that can be switched off.

2.6.1 Debugging content on an HTTPS/SSL connection

As the data stream is encrypted, debugging an application can be problematic. By its very nature SSL is not easy to crack and practically impossible to decrypt.

Debugging has to be based on a classic man in the middle attack.

Using a Teamcenter Thin (browser) or Teamcenter 9.1 (and above) RAC with TCCS this is easy to do using tools like fillder2 (http://www.fiddler2.com/fiddler2/) and Firebug (http://getfirebug.com/) . HTTPS decryption with fiddler2 is very simple, but it requires the user to accept an unsigned certificate which is common to the approach. (More details in the appendix)

Decrypting traffic from Teamcenter RAC prior to 9.1 and TCCS is much more complex and requires a lot of configuration. The principal is exactly the same as user with browsers; a forward proxy is placed between the client and the server. The client has to be configured to send it requests to the forward proxy which will respond with the usual https certificate exchange except of course its certificate will be self-signed. Unlike a normal forward proxy which simply forwards HTTPS traffic once the connection has been made, these debugging proxies will terminate the client connection at the proxy and create a new HTTPS connection from the forward proxy to the server. Now all data passing between client and

server is decrypted on the forward proxy made available to debug, then re-encrypted before forwarding on to the client. Configuring the RAC to accept self-signed certificates and handle a forward proxy is documented in Teamcenter Technical documentation. It is not easy for good reasons. A forward proxy built for this type of analysis is Burp from Port Swigger (http://portswigger.net/).

This section shows how a "man in the middle" attack is carried out against HTTPS and what it requires to be successful. In real life it's not easy to gain access to a data stream but it also this attack shows why you should take notice when your browser is querying the certificate. For example when it tells you the certificate is from the wrong node this could mean you are subject to a "man in the middle attack". Someone may have obtained the real certificate have put it in on to their proxy. So you should always think before selecting OK.

3 The Importance of Network Latency

Latency is the amount of time it takes a packet of data to move across a network connection. When a packet is being sent, there is "latent" time, when the computer that sent the packet waits for confirmation that the packet has been received. This "Round Trip Delay" or RTD is typically measured with the **ping** utility (see section 2.2.1 for a detailed description of **ping**).

The latency of a network is critical for the performance of a networked system. Teamcenter is developed to mitigate the effects of latency but although it is less sensitive than most applications it is still affected.

Latency is made up of three components:

- **Transmission:** Latency is limited by physics; it cannot be improved with money or technology. Packets simply need time to travel from one computer system to the other. They travel at the speed of light. (Note that the speed of light depends on the medium, such as copper, fiber, or air. The medium slows the speed down – however this is typically around 50 %.). Today's computers are so fast, that is, their cycle time is so short that the speed of light becomes a more and more relevant performance bottleneck.

- **Processing:** Part of the latency is also the speed with which one may address the destination, that is, the routing of the packets through the network. Just like talking on the phone: Even if we can transmit a tune very fast from A to B, dialling costs time. Because latency is not concerned with the amount of data, but with the speed, it is not influenced by bandwidth. i.e., the latency of a 10 Mbit/s and a 100 Mbit/s LAN with the same topology is the same, much to the surprise of many people who think a 100 Mbit/s LAN is ten times as fast as a 10 Mbit/s one. This misunderstanding comes from referring to high bandwidth networks as "fast", when in fact the speed at which data packets travel does not change, it is the network capacity to carry data that is measured. However, a bandwidth stressed link may well exhibit variations in measured latency times.

  A well-known latency effect occurs with the phone system: An overseas call that is transmitted over transatlantic lines arrives faster that one that gets transmitted over satellite. You also see this effect on television when video and audio stream are transmitted over different media and do not synchronize any more.

  While latency is strictly a physical capability, it may be influenced by technique, within the physical limits. Connections that dial (or establish a new dial-up connection) for each data transfer have a much higher latency; therefore non-permanent dial-up lines will be much slower than permanent leased lines. Each active network component within the connection (switches,

routers, firewalls) will slow down the transfer as well – the component needs time to receive the packet, analyse what must be done with it, and address the next destination.

- **Serialization delay:** In particular, latency of active network components is determined by serialization delay: components work only on packets that they have received completely and have to wait until the last bit has arrived before processing the packet. Of course, when several such "store and forward" nodes are in a network connection or path, these times add up. Therefore, latency is affected by the network infrastructure components (see below). One component that is often ignored that works using the "store and forward" technique is the computer system's network interface that buffers incoming and out going data, which is discussed later.

  In the context of performance, firewalls can have a dramatic influence. They do extensive processing of packets or even of complete requests and responses. That can slow down packets drastically and therefore make latency much worse. On the other hand, in the WAN, you cannot remove firewalls, because they are an important perimeter protection.

  Many companies are now opting to use MPLS (Multiprotocol Label Switching) provided by the large network service providers for example British Telecom and AT&T. MPLS supports QoS (Quality of Service) and allows packet prioritization, which helps to overcome some of the serialisation issues by jumping high priority packets to the front of the queue. Typically interactive applications such as Teamcenter will benefit from this technology.

The following table provides physical latencies for common distances or for passing through common network equipment:

Category	Latency
Transmission – Distance	
1000 km light speed in vacuum (max. possible speed)	3.3 ms
1000 km in wire / fibre	5 ms
Processing	
1 LAN Router	250µs (SW-Router) – 5µs (Layer-3 Switch)
1 LAN Switch	6µs (64B/Gigabit) – 42µs (1500B/Fast Ethernet)
1 Firewall (w/address filtering)	< 1ms (est.)
1 Firewall (w/stateful inspection)	Approx. 1ms depends on the level of inspection
Serialization	
100 Mbit (Fast) Ethernet	100 bytes: 8 µs / 1500 bytes: 120 µs
1 Analogue Modem [at 9.6 kbit/sec]	100 bytes: 83 ms / 1500 bytes: 1250 ms
ISDN Modem [at 64 Kbit/sec]	100 bytes: 12.5 ms / 1500 bytes: 187.5 ms

If the network path latency is not consistent with the table when considering distance and network devices, it may be possibly to improve latency by reviewing the configuration of the firewalls, routers etc. Of course, the network can also be overloaded, or configuration is simply not correct. Switching protocols may also influence the latency negatively, for example Ethernet → ATM → Ethernet where mismatches network packets size leads to inefficiency packet use and adds delays to data transfers.

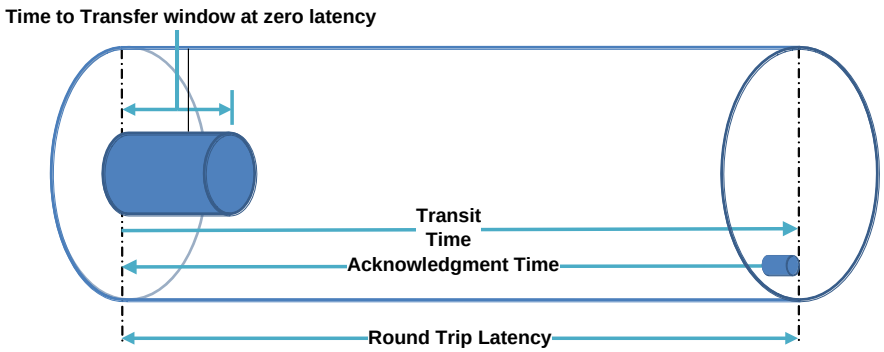
3.1 Latency and its effect on bandwidth utilization (BDP)

As network bandwidth increases, some of the default configuration settings for TCP/IP no longer result in maximum performance. This is particularly true for networks that have high bandwidth and high latency. In TCP, data is sent in "windows" which is basically a number of packets sent before the remote system is required to acknowledge their receipt. The number of packets in a TCP window varies according to the line etc., (see section 2.2). The problem is the system has to wait for acknowledgement before sending the next window of data. On high bandwidth and high latency networks this can seriously restrict the amount of throughput possible. A measure of this effect is the Bandwidth Delay Product (BDP) which is simply the bandwidth (in bytes/s) of the slowest link multiplied by the round trip delay (in seconds). The BDP is the amount data in transit on the link or another way to look at it is the amount of data held in the network, rather like the quantity of water held in a pipe of flowing water. For maximum efficiency, the sending system needs to be able to "fill" the networking connection before requiring an acknowledgement from the other end, the quantity of data required to do this is the BDP. If the receive or send buffers are less than the BDP then the available bandwidth cannot be used efficiently. By default, receive and send buffer are limited, consequently a high BDP network can have its total throughput seriously reduced.

For example, the BDP for a 1 Mbit/s bandwidth at a latency of 500ms is given by:

1,000,000 bit/sec * 1/8 byte/bit * 500 ms = 62.5 KB

The throughput of a network at a given bandwidth and latency can be estimated as the time to transfer the window data at zero latency plus the transit time and first acknowledgment time, divided into one to give effective Kbytes/s bandwidth



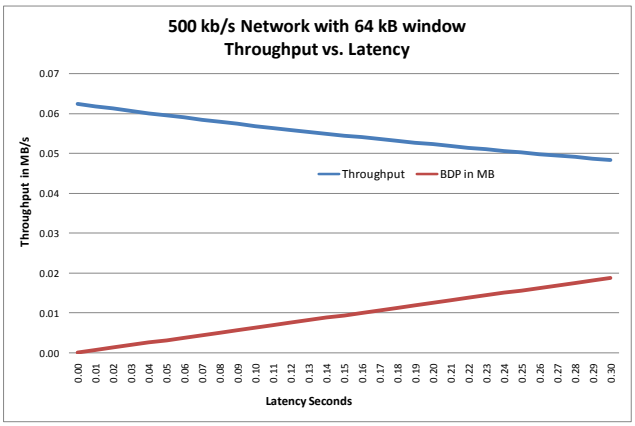
This can be expressed as:

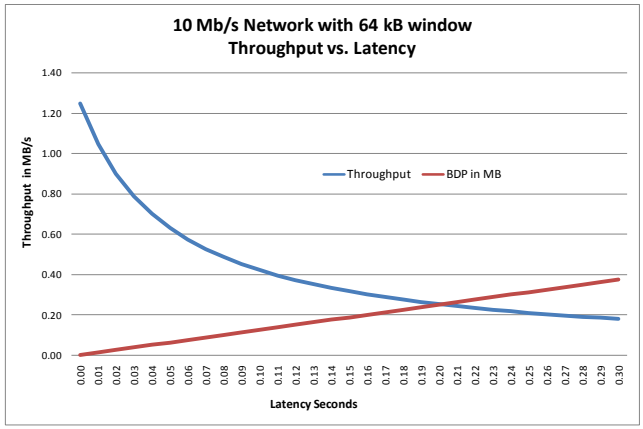
$$\left(\frac{1}{\left(\frac{window}{bandwidth}\right)+latency}\right)*window$$

Where:

- Bandwidth is in Kbytes
- Window is in Kbytes
- Latency is the round-trip latency in seconds

Example plots of throughput versus latency and BDP using a 64KB window. (See Appendix for more examples)





The graphs of throughput versus latency show how on a high-BDP network the throughput of a TCP connection is impacted by latency. The apparent throughput for a single connection is below the expected throughput, although, of course, the total throughput of the connection will be as defined by the connection's bandwidth.

Several extensions to TCP have been created to address the BDP issue and most modern TCP/IP stacks implement these features (see Section 7.1.2 Windows CTCP). However, these features are not always enabled by default and need to be explicitly turned on by system administrators. Many of these features require tuning to achieve any benefits. Details of the various operating and network

driver parameters to change are given in the following sections for each supported operating system.

3.1.1 Network Data Loss

Another source of bandwidth reduction is data loss with the network infrastructure. The table below shows typical losses.

Connection Type	Typical Loss
LAN	0%
Private Line	0%
MPLS	0.1% to 0.5% depends on service SLA
IP VPN	0.5% to 1%

The impact of losses on bandwidth can be severe for example:

- A 10Mbit/s connection with 50ms latency without loss would have a throughput of about 10 Mbit/s, with 0.1% loss that falls to approx.: 6.7 Mbit/s and with 1% 2.3 Mbit/s.
- A 20 Mbit/s connection with 125ms and zero loss would have throughput of about 4 Mbit/s, with 0.1% loss that falls to about 3 Mb/s and 0.5% only 1.3 Mbit/s.

3.1.2 WAN Accelerators

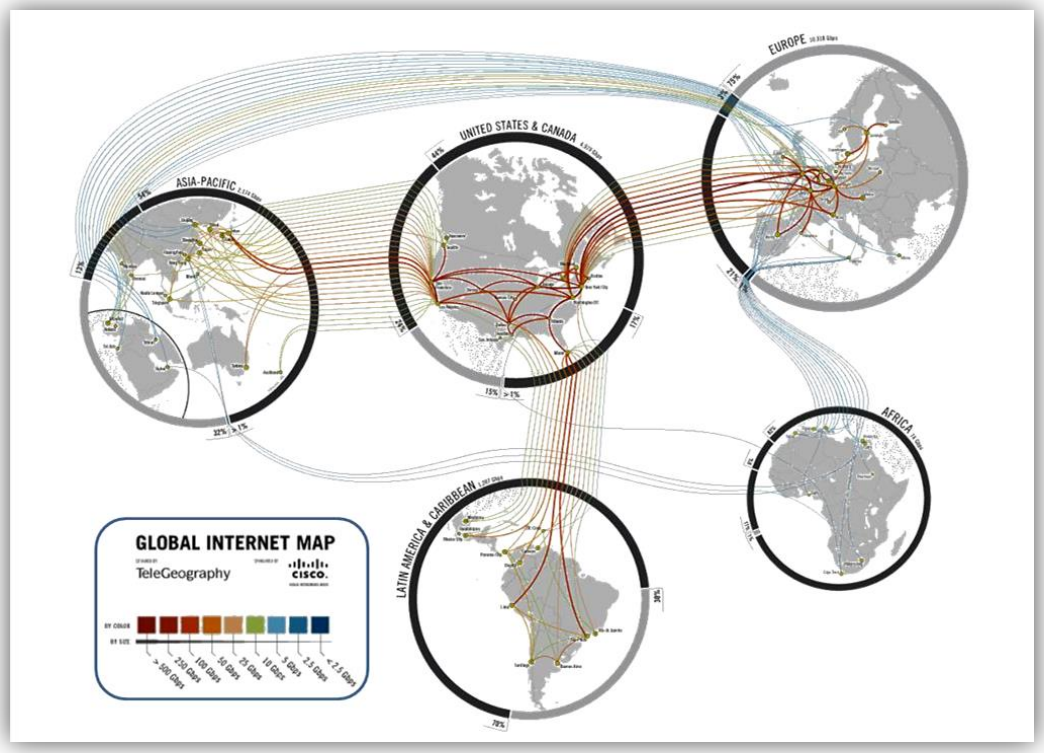
It is also worth mentioning at this point that network "accelerators" such as those from Riverbed, Cisco and Bluecoat make use of the TCP extensions along with other various proprietary optimization techniques to improve WAN performance. Accelerators implement QoS, data compression, caching, flow optimization etc., which assists interactive performance on heavily loaded as well as high BDP connections. As a result of these functions, Accelerators can potentially reduce the Teamcenter requirement for both bandwidth and network latency. Accelerators also improve the reliability of poor high lost network and as well as heavily congested networks. This means a potential cost benefit even on a low BDP networks.

Some customers have reported significant improvements to interactive WAN traffic using these devices, particularly where the BDP is high. For example, there are four-tier users in India working from servers in Denmark and receiving very acceptable interactive performance.

One advantage of this technology is that it requires less operating system and networking expertise to gain significant benefits than the alternative of tuning the various operating system parameters but this is at additional a financial cost.

See section 5.6 for a more detailed discussion of WAN accelerators and Teamcenter.

4 Global Networking: Global and Continental Deployment



Global Internet 2009

4.1.1 Teamcenter and Central Deployment

Teamcenter's unified architecture has been developed specifically to address the issues of deployment across high latency WANs making it possible to consider continental, global and cloud based deployments.

Teamcenter makes use of distributed and local caches to minimise network trips. The HTTP based Rich Application Client (RAC) caches user metadata locally allowing a more interactive user experience than can be achieved by a solely Web browser based system. The RAC cache also permits larger transfers of data reducing the number of network trips. The current version of Teamcenter (Teamcenter 8) is designed to operate to a maximum of between 250 and 300 ms of latency. The actual value depends on the application and use case. Any deployment requires testing across the proposed WAN connection in order to ensure the performance will be acceptable to the end user also to discover and fix issues discussed in this paper before the full deployment is rolled out.

4.1.2 Datacentre location and other considerations

Latency is one of the major issues when considering a continental or global deployment, Appendix 2 has an example for a global SLA based on latency. The

example is from 2008 and shows large parts of the world are now with in the latency tolerance of Teamcenter. Every year new cables are laid which improve the latency figures especially in Asia.

The second major issue is the availability of bandwidth and the ability to access bandwidth in the peak periods. Managed infrastructure (now considered a "Cloud Service") such as MPLS, provide an agreed SLA available to you at your point of presence and the cost will reflect the difficulties of providing bandwidth in your area. The scale of the issue increases with the amount of bandwidth required. Looking at the Global Network map above it is clear to see that some cities have better connection than others. It is not by accident that Cloud datacenters from Amazon, Yahoo, Microsoft, Google etc., are based at main connection points. In North America, Cloud providers usually have East coast and West coast datacenters because of in land latency and bandwidth issues. They also try to avoid intercontinental traffic to, again to provide a most consistent service. Of course theirs is a higher remit than that of a company PLM system but it is worth noting.

Clearly basing a Teamcenter datacenter in one of the cities served by good network connections could have significant benefits. A geographic location that provides the best connection (latency and bandwidth) for all sites should also be considered. However the actual location of a datacenter is usually down to politics and history but a datacenter in a networking backwater will make no senses. The legal issues surrounding the location of data outside of national or political boundaries can have considerable influence.

4.1.3 Cost consideration

Central deployments can offer significant cost reductions in administration, data distribution and promotes global processes. But the network connection is now essential and it should come as no surprise that networking costs will initially increase to provide sufficient bandwidth with an acceptable SLA (service Level Agreement). Disaster management with a centralised deployment is also a major consideration. The central datacentre is now critical; failure causes the shutdown of the whole company, not just part of it. Consequently, high availability systems and off site backup/stand by systems should be considered. Some applications, such as those feeding production lines may be unsuitable for central deployment if a long period of down time cannot be tolerated or if centrally deployed contingencies must be made for such an occasion.

4.1.4 The Future

Within continental boundaries, we can see the growth of low cost bandwidth continuing for the foreseeable future. However, when considering intercontinental connections, consideration of the available bandwidth and its

projected consumption should be made; for example the connection between the US and Europe is rated at about 39 terabits per second, currently only 25% is used but demand is expect to exceed 39Tb/s by 2014. In all probability the bandwidth issue will be addressed in time however, if you plan to require high bandwidth and reliability over a transatlantic link, contingency for future higher costs should be made to maintain a suitable SLA. Projected and current traffic analysis data is available from TeleGeography (see references section) and others.

5 Improving Client and Server Network Performance

This paper concentrates on client and server operating system network connection performance.

Like most modern applications, Teamcenter makes use of TCP for its network traffic (see section 2 for protocol details).

TCP connections have several important characteristics.

- First, they are a logical point-to-point circuit between two application layer protocols. TCP does not supply a one-to-many delivery service, it provides only one-to-one delivery.
- TCP connections are connection-oriented. Before data can be transferred, two application layer processes must formally negotiate a TCP connection using the TCP connection establishment process. Similarly, TCP connections are formally closed after negotiation using the TCP connection termination process.
- Reliable data sent on a TCP connection is sequenced and a positive acknowledgment is expected from the receiver. If a positive acknowledgment is not received, the segment is retransmitted. At the receiver, duplicate segments are discarded and segments arriving out of sequence are placed back in the proper order.
- TCP connections are full-duplex. For each TCP peer, the TCP connection consists of two logical pipes: an outgoing pipe and an incoming pipe. The TCP header contains both the sequence number of the outgoing data and an acknowledgment (ACK) of the incoming data.

In addition, TCP views the data sent over the incoming and outgoing logical pipes as a continuous stream of bytes. The sequence number and acknowledgment number in each TCP header are defined along byte boundaries. TCP is not aware of record or message boundaries within the byte stream. The application layer protocol for example HTTP (Hyper Text Transfer Protocol) must provide the proper parsing of the incoming byte stream.

To limit the amount of data that can be sent at any one time and to provide receiver-side flow control, TCP peers use a window. The window is the span of data on the byte stream that the receiver permits the sender to send. The sender can send only the bytes of the byte stream that lie within the window. The window slides along the sender's outbound byte stream and the receiver's inbound byte stream. Tuning the window size is very important on high BDP connections as was discussed in the previous section.

TCP performance falls off rapidly on poor network connections where packet loss is high. The following RFC's have been introduced to help address poor network connection issues:

- RFC 2582: Fast Recovery Algorithm,
- RFC 2883: An Extension to the Selective Acknowledgment (SACK) Option for TCP,
- RFC 3517: A Conservative Selective Acknowledgment-based Loss Recovery Algorithm for TCP
- RFC 4138: Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP and the Stream Control Transmission Protocol (SCTP)

The latest versions of Teamcenter-supported operating systems have implemented these, however you have to move to Windows Vista or Windows Server 2008 to get full support. So anyone with Windows clients on poor network connections is advised to move to the newer versions of Windows.

5.1 Network Interface Card (NIC) Configuration

One of the critical and often ignored performance issues is the configuration of the NIC (Network Interface Card). Misconfiguration causes unreliable connections, high data loss and generally poor performance. It shows up clearly in file transfer performance, for example in an FMS file transfer.

Two direction (duplex) TCP streaming rates:

Network type	Raw bit Rate (Mbits)	Payload Rate (Mbits)	Payload Rate (MB)
10 Mbit Ethernet, Half Duplex	10	5.8	0.7
10 Mbit Ethernet, Full Duplex	10 (20 Mbit full duplex)	18	2.2
100 Mbit Ethernet, Half Duplex	100	58	7.0
100 Mbit Ethernet, Full Duplex	100 (200 Mbit full duplex)	177	21.1
1000 Mbit Ethernet, Full Duplex, MTU 1500	1000 (2000 Mbit full duplex)	1470 (1660 peak)	175 (198 peak)
1000 Mbit Ethernet, Full Duplex, MTU 9000	1000 (2000 Mbit full duplex)	1680 (1938 peak)	200 (231 peak)
FDDI, MTU 4352 (default)	100	97	11.6
ATM 155, MTU 1500	155 (310 Mbit full duplex)	180	21.5

Network type	Raw bit Rate (Mbits)	Payload Rate (Mbits)	Payload Rate (MB)
ATM 155, MTU 9180 (default)	155 (310 Mbit full duplex)	236	28.2
ATM 622, MTU 1500	622 (1244 Mbit full duplex)	476	56.7
ATM 622, MTU 9180 (default)	622 (1244 Mbit full duplex)	884	105

It is easy to see that Full Duplex is the best configuration of an Ethernet connection but it is very often left to the default.

You can usually configure the Ethernet adapters for the following modes:

- 10_Half_Duplex
- 10_Full_Duplex
- 100_Half_Duplex
- 100_Full_Duplex
- Auto_Negotiation

It is important that you configure both the adapter and the other endpoint of the cable (normally an Ethernet switch or another adapter if running in a point-to-point configuration without an Ethernet switch) correctly.

Incorrect configuration can lead to a duplex mismatch that is one end is using half duplex and the other end full duplex. This can results in:

- Packets dropped due to input errors result in retransmissions and delays.
- FMS file failures and slow file transfer
- Unexpected Cluster node failovers
- Unexpected IP Multipathing (IPMP) network interface failovers
- Unexpected Trunked Ethernet link aggregation link failovers
- Switch ports with “port monitoring” or “port security” enabled may shut down
- Ports with excessive errors
- Boot net install problems may fail or perform slowly.

The default setting is usually autonegotiation, which negotiates the speed and duplex settings for the highest possible data rates. For the autonegotiation mode to function properly, you *must* also configure the other endpoint (switch) for autonegotiation mode.

If one endpoint is manually set to a specific speed and duplex mode, then the other endpoint must also be manually set to the same speed and duplex mode. Having one end manually set and the other in autonegotiation mode can result in

duplex mismatch. The table below show the outcome for various settings of 10 Mb/s and 100 Mb/s network ports as defined by IEEE 802.3

Switch port / NIC	Half-Duplex	Full-Duplex	Autoconfigure
Half-Duplex	OK	☹	OK
Full-Duplex	☹	OK	☹
Autoconfigure	OK	☹	OK

Do not disable autonegotiation between switches or NICs unless absolutely required. Disabling autonegotiation should only be used as a troubleshooting aid or temporary workaround until the autonegotiation problem is resolved.

5.2 Web Application Server and File Server Cache (FSC) TCP Operating System Resource Tuning

By default, most operating systems do not set their TCP parameters at suitable values for a Web Application Server. Under heavy load an FSC behave very much like a Web Application Server, so the tuning recommendations for an FSC is the same as that for a Web Application Server. When the Web Application Server and a heavily loaded FSC are both on the same system, tuning the network connections becomes essential to avoid issues with socket resources.

A Web Application Server and an FSC both handle many individual transactions per second. This results in the rapid opening and closing of sockets. The default mechanism for dealing with the closure of a previously used socket in TCP was not designed for this environment. Consequently the default values for TIME_WAIT for example is too high and may cause socket exhaustion under peak loads by failing to release sockets in a timely manner. The parameters suggested in this paper are those for a typical Web Application Server.

5.3 FMS

5.3.1 Flow control

Flow control can consume a significant amount of CPU time and result in additional network latency as a result of data transfer interruptions. It is recommended that you increase buffer sizes to avoid flow control under normal operating conditions. A larger buffer size reduces the potential for flow control to occur, and results in improved CPU utilization. However, a large buffer size can have a negative effect on performance in some cases. If the TCP/IP buffers are too large and applications do not process data fast enough, in extreme cases paging can increase. The goal is to specify a value large enough to avoid flow control, but not so large that the buffer accumulates more data than the system can process.

The default buffer size is 8 KB. The maximum size is 8 MB (8096 KB). The optimal buffer size depends on several network environment factors including types of switches and systems, acknowledgment timing, error rates and network topology, memory size, and data transfer size. When data transfer size is extremely large, you might want to set the buffer sizes up to the maximum value to improve throughput, reduce the occurrence of flow control, and reduce CPU cost.

5.3.2 TCP streaming workload tuning

FMS streaming workloads move large amounts of data from one FSC to another. Clearly the main network measurement of interest to FMS is bandwidth, but on high-latency networks you should look at end-to-end latency. FMS WAN mode helps maximize bandwidth usage by splitting up the file and transferring it using many pipes to transfer the segments in parallel. But even when using this mode tuning the connection is worth considering, especially where BDP is high.

The Teamcenter Web Application Server streams a considerable amount of data to a client particularly during a Bill of Material expansion or when streaming Vis data from the JTcache. Consequently the Web Application Server application server should also be considered for tuning.

Similarly Oracle requests especially while expanding the product structure, and other such tasks, transfers considerable amounts of data and again the Oracle server should be considered for tuning. To benefit fully the Oracle SDU sizing (Session Data Unit or the size of the packets to send) also needs tuning.

The primary tunable parameters that affect TCP performance for streaming applications are the following:

- The receive space
- The send space
- Large windows (RFC 1323)
- Nagel (see section 2.2.3)
- Socket buffer maximum
- Special network hardware for example checksum off load.

The following table shows suggested sizes for the tunable values to obtain optimal performance on a LAN (low-BDP network); based on the type of adapter and the MTU size (this is a network technology limitation see section 2.1.4):

Device	Speed	MTU size	Send Space	Receive Space	Socket Buffer	Large window?
Ethernet	10 Mbit	1500	16384	16384	32768	N
Ethernet	100 Mbit	1500	16384	16384	65536	N
Ethernet	Gigabit	1500	131072	65536	131072	N
Ethernet	Gigabit	9000	131072	65535	262144	N
Ethernet	Gigabit	9000	262144	1310722	524288	Y
ATM	155 Mbit	1500	16384	16384	131072	N
ATM	155 Mbit	9180	65535	655353	131072	N
ATM	155 Mbit	65527	655360	6553604	1310720	Y
FDDI	100 Mbit	4352	45056	45056	90012	N
Fibre Channel	2 Gigabit	65280	655360	655360	1310720	Y

Ref: http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/prftungd/netperf3.htm

Notes:

1. Check the default value for sb_max (socket buffer limit) on your OS. The values shown in the table are the minimum values.
2. Performance is slightly better when using these options, with large widows (RFC1323) enabled and with jumbo frames on Gigabit Ethernet. To use this all devices connected must support Jumbo frames, 10/100 Mb/s device do not support Jumbo frames
3. Some combinations of TCP send and receive space will result in very low throughput, (1 Mbit/s or less). To avoid this problem, set the send space tunable to a minimum of three times the MTU size or greater or equal to the receiver's receive space.

The following are general guidelines for tuning TCP on Teamcenter servers (Web Application Server, Enterprise Server and FMS server):

- Set the TCP send and receive space to at least ten times the MTU size.
- For high speed adapters, the send space should be two times the value of the receive space.

5.3.3 Large TCP windows (RFC 1323) and High BDP Networks

The send and receive space consideration of the previous section are for a low-BDP networks but for a high-BDP network over which you are trying to move large amounts of data another consideration must be made.

The initial designs of TCP limited send and receive buffers to 64K. If the network BDP is small then these buffer sizes are adequate. On networks with a large BDP e.g. high-bandwidth, high-latency WANs larger buffers are needed to allow for a larger window (see section 2.2 for window details) to reduce the acknowledgement delay. RFC1323 describes a mechanism to allow for buffers larger than 64K and most modern TCP/IP stacks have this enabled by default.

To make use of large windows, the send and receive buffers need to be increased beyond the 64K default. In previous table, the send and receive buffers are constrained to 64K in most cases. On a high-BDP WAN you should consider doubling them to 128K and possibly larger; this needs to be adjusted by trial and error. Both client and server need to be configured otherwise the size will fall back to 64K

To be successful, all network equipment between the two FSC or FSC and clients must be able to accommodate large windows; otherwise the window size will be constrained to 64K.

Ideally, the window should be set to the product of end-to-end network bandwidth (in bytes/s) and the round-trip delay (in seconds), also referred to as the bandwidth-delay product. This value should be set according to the amount of TCP data expected to be received by the computer.

5.3.4 Tuning FSC buffers

In order for FSCs to take maximum benefit from large windows the buffer need to be tuned to at match the operating system send/receive buffer. 128K is a good starting point and should give some improvement on all networks.

At Teamcenter 8.3 the default internal FSC buffer size was changed from 16K to 64K, this can be overridden by setting the

`com.teamcenter.fms.servercache.FSCConstants.buffSize` property in the `fsc.properties` or `fsc.${FSC_ID}.properties` file.

The default socket buffer sizes is changed from the system default to (FSC buffer size * 2) + 1024. Again this value can be overridden this time by setting the `com.teamcenter.fms.servercache.FSCConstants.sockBuffSize` property in the `fsc.properties` or `fsc.${FSC_ID}.properties` files.

Example properties: (taken from the `fsc.properties.template` file):

```
# FSC internal buffer size.
# Default value is 64K.
# Value should be in 16K increments.
# Minimum is 16K.
#
#com.teamcenter.fms.servercache.FSCConstants.buffSize=64K
#
# Socket buffer size override.
# Default value is
(com.teamcenter.fms.servercache.FSCConstants.buffSize * 2) +
1024.
# The value 0 disables setting the socket buffer sizes (uses
system default).
# Minimum value is 8K (excluding the 0 case).
#
#com.teamcenter.fms.servercache.FSCConstants.sockBuffSize=129K
```

How to determine the system default socket buffer sizes?
Set the property `com.teamcenter.fms.servercache.FSCConstants.sockBuffSize=0` in the `fsc.properties` or `fsc.${FSC_ID}.properties` file.

Stop and restart the FSC the use `fscadmin` to set the performance logger to debug.

```
fscadmin -s
http://cii3w037:7168 ./loglevel/com.teamcenter.fms.servercache.Pe
rfLog/debug
```

Make a connection to the FSC. Repeating the last command would be sufficient

In the log kook for the log entry:

```
DEBUG - 2006/09/25-16:39:00,654 UTC {cii3w122} PERF: Socket
buffer sizes, recv: 8192, snd: 8192
[com.teamcenter.fms.servercache.FMSSocketListener.customizeReques
t(FMSSocketListener.java:76):SharedThreadPool-4]
```

This example shows the windows default of 8k send and receive buffer sizes.

5.3.5 FMS on high-BDP networks

As already discussed, FMS can benefit from tuning the windows size of high-BDP networks. FMS also offers a WAN mode that helps to maximize bandwidth usage.

WAN mode uses multiple pipes to address the throughput restrictions of high BDP networks. Because a single pipe through the network is restricted, one solution is

to split up any large files and transfer the parts in parallel through the network in multiple pipes. That way more of the available bandwidth can be used. This is exactly what the FMS WAN mode does. Only a maximum number if pipes or files splits are specified in the FMS configuration, the code adjusts the number of pipes used according to the BDP and other network conditions it detects. This mechanism has proved effective on Teamcenter multi-site connections between Europe and India.

However on very high-BDP networks, improvements can be gained by tuning the window size, buffers and using WAN mode.

5.3.6 FMS Threads and 'ulimit' (Unix/Linux)

File resources in an operating system are accessed via system resource called "file handles" or "file descriptors". With an operation system, many resources are considered to be files, not just files of disk. For example network connections (sockets and ports), are access as files. The resources are controlled at two levels, the operating system kernel and per process. On busy FMS servers, the system may not have enough "file handles" configured which will prevent additional files or sockets being opened. The use of handles is not one for one i.e. a "C" program "fopen" used to open a file will take three handles. You may encounter the same issue with the Web Application Server, the settings for "file handles" will vary by server and you must refer to the appropriate Web Application Server documentation.

You need to increase both the number of handles and the number of threads available to the FSC process.

The following is a Solaris example

FSC bootstrap/startup script

1. Create a backup of the following files (Note: Substitute fsc directory with "fms" for Teamcenter):

   - `<TC_ROOT_SUN>/fsc/rc.ugs.FSC_<sitename>_infodba`
   - `<TC_ROOT_SUN>/fsc/rc.ugs.FSC_Pool`
   - `<TC_ROOT_SUN>/fsc/startfsc.sh`
   - `<TC_ROOT_SUN>/pool_manager/rc.ugs.poolmgr_PoolX (where X may be "A", "B", etc.)`

2. Edit each of the mentioned above scripts and add the following line anywhere before the line that runs the Java command:

```
ulimit –n 12000
```

Fmsmaster.xml

1. Create a back of the following files (Note: Substitute fsc directory with "fms" for Teamcenter):

   - `<TC_ROOT_SUN>/fsc/fmsmaster_FSC_<sitename>_infodba.xml`
   - `<TC_ROOT_SUN>/fsc/fmsmaster_FSC_PoolX.xml (where X may be "A", "B", etc.)`

2. Edit the each of the file specified above and change the following line from the default of 255 to 1023.

```
< property name="FSC_MaximumThreads" value="1023"
overridable="false"/>
```

5.3.7 FMS performance logger

The FSC performance a logger can be used to output FMS throughput information to the log files. This can be useful for analysis of problems but will produce too much overhead and output for systems with a lot of concurrent usage.

There is a hard coded 500ms minimum threshold to prevent performance stats on small operations from being logged and will prevent inaccurate measurements due to buffering.

The performance statistics are logged at the debug level, and setting the entire FSC to debug is generally not a good idea. You can set the performance logger to debug using the following `fscadmin` command:

```
fscadmin -s <serveraddr>
./loglevel/com.teamcenter.fms.servercache.PerfLog/debug
```

For Eexample:

```
fscadmin -s http://cii3w037:7168
./loglevel/com.teamcenter.fms.servercache.PerfLog/debug
```

The log entries look like this:

```
DEBUG - 2006/09/25-16:44:40,694 UTC {cii3w122} PERF: Local
whole file binary download
(ee00000000000054a6dcc6a33cb8ae2e/TestFile8388608.bin) (8.0
/ 340.04)[1] * 0.023 MB/s, requestedRoute:
[fms.teamcenter.com^fsc_91_3, fms.teamcenter.com^fsc_90_3]
[com.teamcenter.fms.servercache.FMSWebHandlerRoot.handleLoca
lReadBinary(FMSWebHandlerRoot.java:5989):SharedThreadPool-4]
DEBUG - 2006/09/25-17:02:32,329 UTC {cii3w122} PERF: Local
whole file binary upload
```

```
(ee00000000000054a6dcc6a36350c744/TestFile8388608.bin.wholef
ile.upload-udtest-45.test) (8.0 / 342.081)[1] * 0.023 MB/s,
requestedRoute: [fms.teamcenter.com^fsc_91_3,
fms.teamcenter.com^fsc_90_3]
com.teamcenter.fms.servercache.FMSWebHandlerRoot.handleWrite
UploadFilePart(FMSWebHandlerRoot.java:4147):SharedThreadPool
-4]
DEBUG - 2006/09/25-16:44:59,413 UTC {cii3w037} PERF: Remote
whole file possible cache download
(ee00000000000054a6dcc6a33cb8ae2e/TestFile8388608.bin) (8.0
/ 359.556)[1] * 0.022 MB/s, requestedRoute: null
[com.teamcenter.fms.servercache.FMSWebHandlerRoot.handleRemo
teRead(FMSWebHandlerRoot.java:6666):SharedThreadPool-4]
DEBUG - 2006/09/25-17:08:32,442 UTC {cii3w037} PERF: Remote
whole file upload
(ee00000000000054a6dcc6a36350c744/TestFile8388608.bin.wholef
ile.upload-udtest-45.test) (8.0 / 356.731)[1] * 0.022 MB/s,
requestedRoute: null
[com.teamcenter.fms.servercache.FMSWebHandlerRoot.handleWrit
eUploadFilePartRemote(FMSWebHandlerRoot.java:4483):SharedThr
eadPool-4]
```

Keeping the performance logger enabled for long periods of time is not recommended.

5.4 Windows Servers latency issues.

5.4.1 Windows Server 2008 and XP 64 bit high latency issue.

You may experience slow TCP/IP performance and long data transfer delay times on a Microsoft Windows Server 2003-based computer or on a Microsoft Windows XP Professional x64 Edition-based computer. It can affect both FMS and BOM expansions over a WAN. There is a hot fix to help address this issue
http://support.microsoft.com/kb/925511/en-us

5.4.2 Windows 7 and Windows Server 2008 network performance issue

Receive Window Auto-Tuning was introduced to make HTTP traffic and data transfers over the network may be more efficient. Receive Window Auto-Tuning improves network performance by letting the operating system continually monitor routing conditions such as bandwidth, network delay, and application delay. Therefore, the operating system can configure connections by scaling the TCP receive window to maximize the network performance. To determine the optimal receive window size, the Receive Window Auto-Tuning feature measures the products that delay bandwidth and the application retrieve rates. Then, the Receive Window Auto-Tuning adapts the receive window size of the on-going transmission to take advantage of any unused bandwidth.

Auto-Tuning can have a considerable benefit address BDP issues. For example, the total achievable throughput is only 51 Mbps on a 1 GB connection with 10 ms latency (a reasonable value for a large corporate network infrastructure). With auto-tuning, however, the receive-side window is adjustable, and it can grow to meet the demands of the sender. It is entirely possible for a connection to achieve a full line rate of a 1 GB connection. Network usage scenarios that might have been limited in the past by the total achievable throughput of TCP connections can now fully use the network

However, in some cases you might experience slower data transfers or loss of connectivity if your network uses an older router and firewall that does not support this feature.

For example the performance of a new Windows 7 or 8 clients is slower than the old XP clients it will probably be due to Auto-Tuning.

Auto-Tuning can be disabled using the **netsh** commands line

```
netsh interface tcp set global autotuning=disable
```

5.5 Oracle

Oracle provides various parameters to assist in tuning the performance of the network layer. The following are describes some of the parameters that may impact performance of Teamcenter as the system size and load grow.

5.5.1 TCP.NODELAY

Oracle Net can use the Nagel algorithm (see section 2.2.7) which as discussed can cause delays. Many Teamcenter queries can potentially be delayed particularly when only a few users are active. Adding tcp.nodelay will stop buffer flushing delays and can improve performance. Depending on the oracle version, tcp.nodelay may be defaulted to YES. But to be sure it is off, add or change the following line in protocol.ora on both client and server if you are using Oracle 10 R1 or earlier. If you are using Oracle 10 R2 or 11 add tcp.nodelay to the sqlnet.ora on the server:

```
TCP.NODELAY = YES
```

Since Teamcenter 2007.1 product structure expansion has made use of batched-SQL, which produces streams of data sent to the client. The rate at which this data is transferred to the client is determined by the send buffer size and the SDU (Send Data Unit). Increasing these parameters on both client and server should improve throughput from the Oracle server to the business logic servers. The primary gain in performance from setting the SDU is the reduced number of calls to packet the data. It is recommended that this parameter be set to the maximum 32767 bytes, the defaults is 2048 bytes.

5.5.2 SDU

The default value of the SDU is 2048 which is generally too small for Teamcenter requests. SDU size for the database server can be configured by making the following change to either the sqlnet.ora or listener.ora file:

The maximum size of SDU that can be used depends on the version of Oracle in use. Prior to Oracle 11.2.0.2 the maximum size was 32K, for 11.2.0.2 it was increased to 64K and at Oracle 12.1 it has been increased to 2MB. Both the Oracle server and client must support the same size SDU, specifying a larger SDU in the server than is supported by the client will result in the default being used i.e. 8K.

Teamcenter used the following client versions:

- Teamcenter 8.3: Oracle Client 10.1.0.5.0
- Teamcenter 9.1: Oracle Client 10.1.0.5.0 (except Wntx64 10.2.0.2)
- Teamcenter 10.1: Oracle 11.2.0.2 (except HPP and HPI 10.1.0.5)

In both server side and client side sqlnet.ora, add the appropriate SDU size. For example:

```
DEFAULT_SDU_SIZE=32767
```

Or you can override the current setting in the client side sqlnet.ora and set the local connection value by adding the SDU clause to the listener.ora, for example:

```
SID_LIST_listener_name=
  (SID_LIST=
    (SID_DESC=
     (SDU=32767)

     (SID_NAME=TC)))
```

5.5.3 Send and Receive Buffers

In addition to setting the SDU parameter, network throughput can often be substantially improved by using the SQLNET.SEND_BUF_SIZE and SQLNET.RECV_BUF_SIZE Oracle Net parameters to increase the size of the network TCP send and receive I/O buffers. Setting the SEND_BUF_SIZE and RECV_BUF_SIZE to at least the 2 x BDP, will insure that when large amounts of data are being sent that the network bandwidth will be optimally utilized. To be of value the operating system buffers must be large enough to allow this buffer size. The maximum operation system buffer size will override the Oracle setting. Operating system buffer tuning is covered in later sections.

For example the BDP for a 100 Mbit/s bandwidth at a latency of 5ms is given by:

100,000,000 bit/sec * 1/8 byte/bit * 0.005 sec = 62.5 KB

The buffer should be 2x BDP so that is 125KB

Example LAN BDP values:

Bandwidth	Latency	Buffer size (Bytes)
100 Mb/s	1 ms	25,000
100 Mb/s	5 ms	125,000
1 Gb/s	1ms	250,000
1 Gb/s	5 ms	1,250,000

Client setup:

- You need to add the following to the copy of sqlnet.ora stored on the Teamcenter business logic server and located in the TC_DATA directory for that installation. The example below is for a 100Mb/s network with 1 ms latency. This is a good minimum setting:

```
RECV_BUF_SIZE=25000
SEND_BUF_SIZE=25000
```

Oracle Server setup:

- You need to add the following to sqlnet.ora. By default, sqlnet.ora is located in the $ORACLE_HOME/network/admin directory on UNIX operating systems and the ORACLE_HOME\network\admin directory on Windows operating systems. sqlnet.ora can also be stored in the directory specified by the TNS_ADMIN environment variable. The example below is for a 100 Mb/s network with 1 ms latency. This is a good minimum setting:

```
RECV_BUF_SIZE=25000
SEND_BUF_SIZE=25000
```

Or you can override the current setting in the client side sqlnet.ora and set the local connection value by adding the SEND_BUF and RECV_BUF_SIZE clauses to the listener.ora, for example for a 100 Mb/s network with a 1 ms latency:

```
tc.siemens.com=
      (DESCRIPTION=
                  (ADDRESS_LIST=
                   (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-
server)(PORT=1521)
                 (SEND_BUF_SIZE=25000)
                 (RECV_BUF_SIZE=25000))
                  (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-
server)(PORT=1521)
                (SEND_BUF_SIZE=25000)
                (RECV_BUF_SIZE=25000))
```

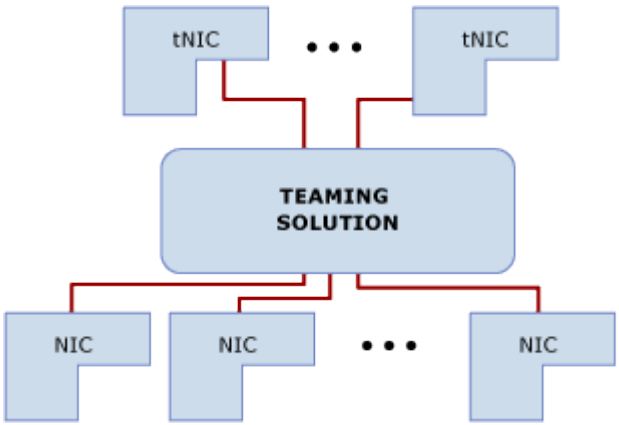
5.5.4 Fetch Array Size

The fetch array size is automatically set by the Teamcenter software to match the expected query result. It is no possible for users or administrators to tune this.

5.5.5 NIC Teaming (aka Link Aggregation, NIC Bonding, LBFO)

Teaming is combining of two or more network adapters so that the software above the team perceives them as a single adapter that incorporates failure protection and bandwidth aggregation. Teaming is defined by IEEE 802.3ad.

Windows Server 2012 includes Teaming as part of the system providing standard GUI management and configuration.



The use of Teaming between the Oracle server and the server room Ethernet switch is recommended on high load systems and system with many Enterprise servers. Teaming will provide bandwidth, high packets rates (essential for databases) and connection resilience.

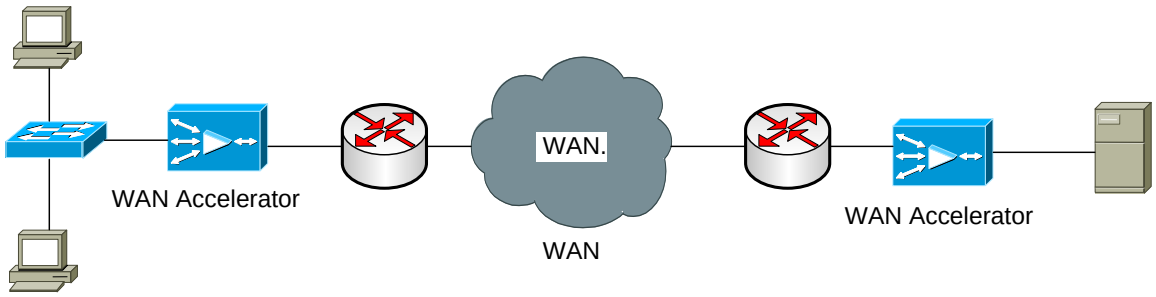
For FMS servers connect Teaming provides better throughput and resilience. For primary Volume serving FSC Teaming will support better throughput with a high number of parallel connections.

Teaming is also recommended for virtual servers to provide the additional bandwidth often required by multiple hosts.

5.6 WAN Accelerators

Network “accelerators” such as those from Riverbed, Cisco and Bluecoat can offer significant benefits of WAN connections and should be considered in any large scale deployment. They can potentially offer benefits not only to Teamcenter but practically all networked applications and reduce administration costs as well as improve application performance.

WAN accelerators offer various form of connection acceleration, the discussion here is restricted to those that offer potential benefit to Teamcenter that is Caching, Compression, TCP enhancement and Application Acceleration.



Basic WAN accelerator Configuration

5.6.1 Object Caching

WAN accelerators support two types of caching Object and Byte

Object caching has been available for many years and has traditionally been used to accelerate access to HTTP content. Typically objects cached include HTTP content, some vendors have extended their object caching support to include HTTPS content, streaming media objects, FTP, and CIFS files.

The mechanism of object caching is fairly straightforward and usually requires no configuration. The client sends a request for an object (file, document, image etc.) to a server and the request is intercepted by the proxy that is in between the client and origin server. Upon receiving the request, the proxy checks to see if it has a copy of the requested object in its cache. If so, the proxy responds by sending the cached object to the client. Otherwise it sends the request to the server. If the proxy requests the object from the original server, that object data is cached by the proxy so that it can fulfill future requests for the same object without retrieving it again from the origin server. The result of serving a cached object is that there is zero server-side bandwidth used for the content and an improvement in response time for the end user. Some WAN accelerators offer pre-population or pre-fetch of objects in to the cache by batch jobs, this allows frequently accessed data to be loaded into the cache without the waiting for the first user access.

Typically object caching is limited in that even if only one byte of an object changes, the entire object must be retrieved again as the object has changed. Object caching is also limited to files requested by the client, and is not used when a client is saving, or posting, a file to a server.

Beyond caching Teamcenter Thin client page objects, object caching does not offer much benefit to Teamcenter RAC or to the FMS system.

5.6.2 Byte Caching

Byte caching is a protocol, port, and IP address independent bidirectional caching technology that functions at the TCP layer by looking for common sequences of data. Byte caching is designed to work in a mesh, hierarchical, or any arbitrary network design and imposes no limitations on the design of a network.

Typically byte caching requires a symmetric deployment between two endpoints. This means WAN accelerator appliances are deployed at the remote sites as well as at the data centre. As appliances are on both sides of the WAN connection, they can maintain a cache of all TCP traffic being sent and received over WAN between the two sites. Each time data needs to be sent, it is scanned for duplicate segments in the cache. If any duplicates are found, the duplicate data is removed from the byte sequence, and tokens are inserted in its place. When the remote appliance receives the data, the tokens are removed from the byte sequence and the original data is reinserted. Byte caching allows the amount of data traversing the WAN to be as small as possible with no loss of the original content.

It's important to note that unlike object caching, byte caching requires the requests to be executed on the remote sites as normal. The caching reduces the WAN traffic and improves the performance the client sees but does not reduce on the servers.

Byte caching offers significant performance improvements to Teamcenter clients and to FMS traffic.

5.6.3 Connection Compression

WAN accelerators offer connection compression typically Persistent Lempel-Ziv (LZ) compression is used that can provide up to 5:1 extra compression. Compression of small objects e.g. small request can have a negative effect of performance.

For Teamcenter traffic compression of FMS has clear benefits however the RAC http connection contain many small request and when a large response is sent from the server the Web Application Server will compress the data. Consequently disabling compression on the Teamcenter RAC connection typically offers best WAN performance but will lead to a reduction in local performance. The connection will still make use of the Byte cache and you will see many request will benefit from it resulting in the report of high levels of compression.

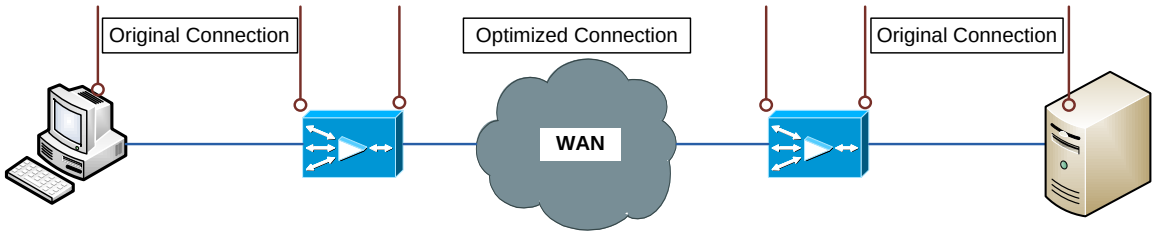
Riverbed Steelhead accelerators offer a tuning option called "Strip Compression". "Strip Compression" will remove "accepts: gzip" from the remote site requests

which will stop the Web Application Server compressing responses. This should allow local users to benefit from compression and remote users to benefit from data reduction.

5.6.4 Protocol Acceleration

As discussed in earlier sections of this paper the TCP protocol can be tuned to improve the transfer of data. Also newer RFC's provide for congestion control etc. WAN accelerators implement these RFC and use other techniques to improved the WAN connections efficiency and to handling conditions, including packet loss, congestion, and recovery. Experience has shown these appliances do provide a robust connection of extremely poor network circuits.

With an optimized connections, that is not pass through, the connection from the client to the server is split into three connections, client to WAN accelerator, the WAN connection and finally WAN accelerator to server. This TCP proxy strategy allows each connection to be managed independently and shields the communication systems from problematic WAN conditions. This functionality provides the connection robustness requires for poor or error prone connections.



The WAN connection is optimized but local LAN connection behaves as normal.

When configuring the WAN accelerator consider using large TCP buffers typically they are defaulted to 32 KB, Teamcenter can benefit from large buffers.

5.6.5 Application or Layer 7 Acceleration

WAN Accelerators usually include various application specific acceleration techniques to assist in reducing the negative effects of latency and bandwidth to provide improvements in general performance. Typically CIFS, Windows print services, HTTP, HTTPS, NFS, MAPI and video are supported.

5.6.6 Encrypted Traffic

In order to optimize the network traffic the WAN accelerator must be able to examine it. Encrypted traffic restricts the amount of performance gain these appliances can give. Most WAN accelerators can be configured with certificates to allow decryption and re-encryption of https and allow full acceleration.

5.6.7 FMS caching vs. WAN Accelerator caching

Byte caching improves the file access performance of Teamcenter clients and the effect of the caching improves as more data is collected in the cache. FMS also provides file caching specifically for Teamcenter applications.

To understand the differences between the two strategies lets consider a remote site with and without a FSC but both configured for WAN accelerator byte caching.

In the first case without a local FSC the file is requested from the remote FSC and is loaded over the WAN into the client FCC. A second access for the same file from that client will be served by the local FCC. However if a second client on the same site request the same file, request is past to the central FSC, this time the file will be read from the FSC but the WAN accelerator will send tokens to tell the local appliance to satisfy the request from cache. The result is a fast load into the FCC of the new client.

In the second case with a local FSC, the file is requested from the remote FSC and is loaded over the WAN via the local FSC into the client FCC. A second access for the same file from that client will be served by the local FCC. However if a second client on the same site request the same file the local FSC will satisfy the request, that is no traffic will pass over the WAN and no remote FSC access will take place.

Where there are a low number of remote clients or where the clients file access level is low the WAN accelerator caching functionality will be acceptable and give good results. However if there is a large number of users access file, for example many CAD users a local FSC will reduce the load on both the WAN and the central FSC.

5.6.8 WAN Acceleration Summary

WAN accelerator offer clear benefits and should be considered as part of any centralised implementation especially where there connection suffer from a high BDP, congestion or is error prone.

Typically configure the WAN accelerator Teamcenter FMS connection for byte caching, compression and protocol acceleration.

The Teamcenter RAC and Thin client should be configured for byte caching and protocol acceleration and no compression. The Web Application server already compresses data; also the many small requests are likely to have a negative impact.

Links to the various vendor sites are included in the reference section.

5.7 Quality of Service (QoS)

In general terms QoS is a collection of various technologies and techniques often known as Traffic or Packet shaping, which allows applications/users to request and receive predictable service levels in terms of data throughput capacity (bandwidth), latency variations (jitter) and delay.

Support for QoS in WAN connections is common and it is often provided as part of a MPLS connection. It is also provided within WAN acceleration solutions.

You can configure QoS to provide Teamcenter interactive traffic, the RAC and Thin client connections, to have higher quality of service; this is especially effective on heavily loaded network connections.

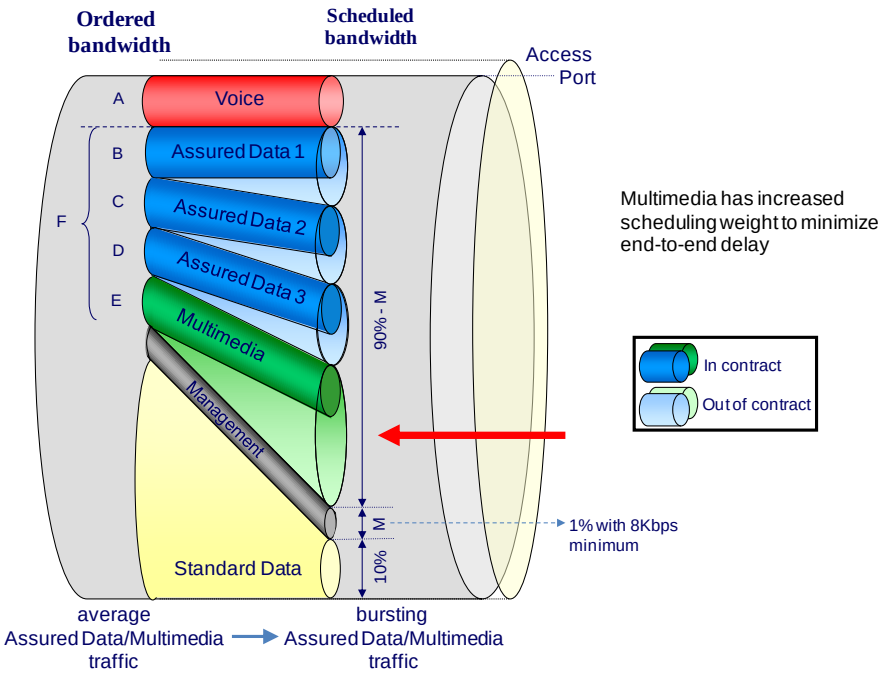
Even if you do not want to increase Teamcenter's QoS on a network you should be aware that most QoS system regard http and https traffic as low priority since it is usually browsing traffic. This can lead to performance issues for Teamcenter connections. This problem can be simply avoided by configuring Teamcenter to use ports other than 80, since browsing traffic is considered to be http on port 80 by QoS systems.

5.7.1 QoS on MPLS

QoS on MPLS is a good example of how QoS can work. Basically traffic is split into classes. The classes can be defined by protocol or via stateful inspection (basically the system looks into the network packets). HTTP can be generally controlled via Protocol i.e. HTTP on Port 80 has a low priority. For Teamcenter traffic stateful inspection would need to be used to define the Teamcenter URL. Typically there are only four classes since too many classes defeat the point of prioritisation

Once the classes of traffic are decided the policy for each class is defined. The policies define percentage of bandwidth, queue limits etc.
Services such as MPLS from BT come with six DSCP (Differentiated Services Code Point), basically predefined policies to which you define classes of traffic.

DSCP 6 Classes of Service.



DSCP Policy definitions.

The diagram above show how the policy is used to allocate the bandwidth depending on load. Voice is always given highest priority. Low priority data will be discarded in favour of the priority classes. So typically UDP will be dropped first followed by HTTP. Browsing traffic is considered low priority. This explains why occasionally pages fail to display.

5.8 Virus scanners

Anti-virus software is an essential, but it can be the source of performance issues that you should be aware of. Some issues, for example scanning of the FCC cache directories can impact performance, should be fairly obvious. Unfortunately there can be some more complex issues that are far from obvious that you need to watch for.

Some Virus scanners, such as F-Secure make use of the Layered Service Provider (LSP) functionality available in Windows Windsock 2. An LSP is a DLL that uses Winsock APIs to insert itself into the TCP/IP stack. Once in the stack, a Layered Service Provider can intercept and modify inbound and outbound Internet traffic. It allows processing all the TCP/IP traffic taking place between the Internet and the applications that are accessing the Internet (such as a web browser, the email client, etc.). The use of LSP functionality can usually be disabled on clients. LSPs have two potentially negative effects. First the software caches the data received, for example HTTP and processes it. This inherently adds latency to any data transfers. When applied to Browser pages the impact is likely to be small, but when applied to large file transfer such those by FMS or even the metadata

transfer for structure expansion it is potentially significant. Second any problems cause by the LSP data inspection and repackaging process are extremely hard to determine. For example until recently one virus scanner would cause NX assembly open failures when the client system was busy. Fortunately this has been fixed in the latest versions.

Any client performance tests should include disabling the virus checker to enable an assessment its impact on performance. This will allow the value of the additional scanning to be evaluated against the impact of system productivity.

6 Tuning Solaris Systems

Solaris 10 has large windows scaling set on by default. See the '*Solaris Tuneable Parameters Reference Manual*' for more details and more parameters.

Configure the following settings or variables according to your tuning needs:

- **file descriptors (ulimit)**
    - **Description:** Specifies the maximum number of open files supported. See section 5.4 for more detail. Some programs on Solaris have problems with a ulimit greater than 1024, so avoid setting values higher than 1024 as the system default.
    - **How to view or set ulimit:** Check the UNIX reference pages on the **ulimit** command for the syntax of different shells. For the Korn Shell (ksh) shell, use the **ulimit -n 1024** command. Use the **ulimit -a** command to display the current settings.
      Use the **ulimit -n 2000** command to set the values.
    - **Default value:** None
    - **Recommended value:** 8192
- **file descriptors (sq_max_size)**
    - **Description:** Specifies the depth of syncq (number of messages) before a destination STREAMS queue generates a QFULL message. When NCA (Network Cache and Accelerator) is running on a system with a lot of memory, increase this parameter to allow drivers to queue more packets of data. If a server is under heavy load, increase this parameter so that modules and drivers can process more data without dropping packets or getting backlogged.
    - **How to set sq_max_size:** In /etc/system, add:
      ```
      set sq_max_size=0
      ```
    - **Default value:** 10,000 messages
    - **Recommended value:** 0 (unlimited)
- **kernel semsys:seminfo_semopm**
    - **Description:** An entry in the /etc/system file can exist for this tuning parameter. This number is the maximum value of System V semaphore operations per semop call. The default value for this option is too low for highly concurrent systems.
    - **How to set seminfo_semopm:** Set this parameter through the /etc/system entry:
      ```
      semsys:seminfo_semopm = 200
      ```
    - **Default value:** None
    - **Recommended value:** 200 (100 is appropriate for most systems, but 200 might be needed in some cases.)

- **TCP_TIME_WAIT_INTERVAL**
    - **Description:** Determines the time that must elapse before TCP/IP can release a closed connection and reuse its resources. This interval between closure and release is known as the TIME_WAIT state or twice the maximum segment lifetime (2MSL) state. During this time, reopening the connection to the client and server costs less than establishing a new connection. By reducing the value of this entry, TCP can release closed connections faster and provide more resources for new connections. When high connection rates occur, a large backlog of the TCP/IP connections accumulates and can slow down both the Web Application Server's and FMS server's performance. These servers can stall during certain peak periods. If the server stalls, the **netstat** command shows that many of the sockets that are opened to the HTTP server are in the CLOSE_WAIT or FIN_WAIT_2 state. Visible delays can occur for up to four minutes, during which time the server does not send any responses, but CPU utilization may stay high, with all of the activities in system processes. Client may receive TCP reset packets (RST) from the servers and clients report no connection or connection refused.
    - **How to view or set tcp_time_wait:** Use the **get** command to determine the current interval and the **set** command to specify an interval of 30 seconds. For example:

```
ndd -get /dev/tcp tcp_time_wait_interval
ndd -set /dev/tcp tcp_time _wait_ interval 30000
```

    - **Default value:** 60,000 milliseconds
    - **Recommended value:** 30,000 milliseconds

- **TCP_FIN_WAIT_2_FLUSH_INTERVAL**
    - **Description:** Specifies the timer interval prohibiting a connection in the FIN_WAIT_2 state to remain in that state. When high connection rates occur, packets can get lost a socket get held in FIN_WAIT_2 (see section 2.1.2 for more detail)
    - **How to view and set tcp_fin_wait_2_flush_interval:** Use the **get** command to determine the current interval and the **set** command to specify an interval of 67.5 seconds. For example:

```
ndd -get /dev/tcp tcp_fin_wait_2_flush_interval
ndd  -set  /dev/tcp  tcp_fin_wait_2_flush_interval
67500
```

    - **Default value:** 675,000 milliseconds
    - **Recommended value:** 67,500 milliseconds

- **TCP_KEEPALIVE_INTERVAL**
    - **Description:** The keepAlive packet ensures that a connection stays in an active and established state.
    - **How to view or set tcp_keepalive_interval:** Use the **ndd** command to determine the current value or to set the value. For example:

```
ndd -set /dev/tcp tcp_keepalive_interval 300000
```

    - **Default value:** 7,200,000 milliseconds
    - **Recommended value:** 900,000 milliseconds

- **TCP_CONN_REQ_MAX_Q**
    - **Description:** Connection backlog. Change the following parameter when a high rate of incoming connection requests result in connection failures for example on a busy Web Application Server or a central FMS server
    - **How to view or set tcp_conn_req_max_q:** Use the **ndd** command to determine the current value or to set the value. For example:

```
ndd -get /dev/tcp tcp_conn_req_max_q
ndd -set /dev/tcp tcp_conn_req_max_q 1024
```

    - **Default value:** For Solaris 10, the default value is 128.
    - **Recommended value:** 1024

- **TCP_CONN_REQ_MAX_Q0**
    - **Description:** Specifies the default maximum number of incomplete (three-way handshake not yet finished) pending TCP connections for a TCP listenerConnection backlog. Change the following parameter when a high rate of incoming connection requests result in connection failures for example on a busy Web Application Server or a central FMS server
    - **How to view or set tcp_conn_req_max_q0:** Use the **ndd** command to determine the current value or to set the value. For example:

```
ndd -get /dev/tcp tcp_conn_req_max_q0
ndd -set /dev/tcp tcp_conn_req_max_q0 1024
```

    - **Default value:** For Solaris 10, the default value is 1024.
    - **Recommended value:** 4096

- **TCP_CWND_MAX**
    - **Description:** Specifies the maximum value of the TCP congestion window in bytes. The actual window used can never grow beyond tcp_cwnd_max. Thus, tcp_max_buf should be greater than tcp_cwnd_max. For large

Windows the value must be larger than the expected maximum window size. (See last Solaris section for suggested setup)

- **How to view or set tcp_cwnd_max:** Use the **ndd** command to determine the current value or to set the value. For example:

```
ndd -get /dev/tcp tcp_cwnd_max
ndd -set /dev/tcp tcp_cwnd_max 1468760
```

- **Default value:** For Solaris 10, the default value is 1048576.
- **Recommended value:** 1468760

- **TCP_MAX_BUF**

  - **Description:** Specifies the maximum buffer size in bytes. This parameter controls how large the send and receive buffers are set to by an application that uses the 'setsocket' function call

```
ndd -get /dev/tcp tcp_max_buf
ndd -set /dev/tcp tcp_max_buf 1468760
```

  - **Default value:** For Solaris 10, the default value is 1048576.
  - **Recommended value:** 1468760 (This has to be higher than the individual values of tcp_recv_hiwat and tcp_xmit_wat values, See the table in Section 5.3.2)

- **TCP_XMIT_HIWAT and TCP_RECV_HIWAT**

  - **Description:** These parameters controls how large the send and receive buffers are set to by an application that uses

```
ndd -get /dev/tcp tcp_xmit_hiwat
ndd -get /dev/tcp tcp_recv_hiwat
ndd -set /dev/tcp tcp_xmit_hiwat 49152
ndd -set /dev/tcp tcp_recv_hiwat 49152
```

  - **Default value:** For Solaris 10, the default value is 49,152.
  - **Recommended value:** Set in accordance with the table in section 5.3.2 but do not reduce below the default.

- **TCP_NAGLIM_DEF**
  - **Description:** Specifies whether to use the Nagel Algorithm or not. See section 2.2.3.

```
ndd -get /dev/tcp tcp_naglim_def
ndd -set /dev/tcp tcp_naglim_def 1
```

- **Default value:** For Solaris 10, the default value is 0.
- **Recommended value:** 1, that is disable Nagel Algorithm.

6.1.1 Setting up large TCP windows on Solaris

If you want to use large TCP windows with FMS over a high-BDP WAN then set the following to 2 x BDP or use these suggested values.

```
ndd -set /dev/tcp tcp_xmit_hiwat 4000000
ndd -set /dev/tcp tcp_recv_hiwat 4000000
ndd -set /dev/tcp tcp_max_buf to 4000000
ndd -set /dev/tcp tcp_cwnd_max 4000000
```

Every TCP connection now reserves 4M buffer (max buf) so ideally this should be used on an FSC that is used to communicate to remote clients or other FSCs. Remote clients or FSCs must be configured for large TCP windows too. Use operating system tool “netsat” to estimate how many connections are open and from that the memory used. If it is an issue try using 2MB or 2 x BDP. See Section 5.3.4 for FSC buffer tuning.

7 Tuning Windows Systems.

Configure the following settings or variables according to your specific tuning needs. Note using netsh with Windows Server 2008 can cause incorrect entries in the registry:

- **TcpTimedWaitDelay**
    - **Description:** Determines the time that must elapse before TCP/IP can release a closed connection and reuse its resources. This interval between closure and release is known as the TIME_WAIT state or twice the maximum segment lifetime (2MSL) state. During this time, reopening the connection to the client and server costs less than establishing a new connection. By reducing the value of this entry, TCP can release closed connections faster and provide more resources for new connections. When high connection rates occur, a large backlog of the TCP/IP connections accumulates and can slow down both Web Application Server and FMS server's performance. These servers can stall during certain peak periods. If the server stalls, the" netstat" command shows that many of the sockets that are opened to the HTTP server are in the TME_WAIT or CLOSE_WAIT state. Visible delays can occur for up to 4 minutes, during which time the server does not send any responses, but CPU utilization may stay high, with all of the activities in system processes. Client may receive RST (reset) and clients report no connection or connection refused.
    - **How to view or set TcpTimedWaitDelay:**
        1. Use the "regedit" command, access the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters registry subkey`, and create a new REG_DWORD value named `TcpTimedWaitDelay`.
        2. Set the value to decimal 30, which is Hex 0x0000001e. This value sets the wait time to 30 seconds.
        3. Stop and restart the system.
    - **Default value:** 0xF0, which sets the wait time to 240 seconds (4 minutes).
    - **Recommended value:** A minimum value of 0x1E, which sets the wait time to 30 seconds.
- **MaxUserPort**
    - **Description:** Determines the highest port number that TCP/IP can assign when an application requests an available user port from the system. On busy server setting this will reduce contention. Note that Microsoft strongly recommends for SQL server that you always use pooling with the SQL Server drivers rather than use this setting. But for IIS, Web Application Servers and FMS this is applicable.
    - **How to view or set MaxUserPort:**
        1. Use the **regedit** command, access the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\`

`Services\TCPIP\Parameters registry subkey`, and create a new REG_DWORD value named `MaxUserPort`.

2. Set this value to at least decimal 32768.
3. Stop and restart the system.

- **Default value:** 5000
- **Recommended value:** At least decimal 32768.

- **MaxConnect Backlog**

  - **Description:** If many connection attempts are received simultaneously, increase the default number of pending connections that are supported by the operating system. So again for busy Web application servers and FMS servers this is worth considering.

  - **How to view or set MaxConnect Backlog:**
    1. Use the "regedit" command and access the

       `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\AFD\Parameters` registry sub key

    2. Create and set (and create if necessary) the following values:
       - `"EnableDynamicBacklog"=dword:00000001`
       - `"MinimumDynamicBacklog"=dword:00000020`
       - `"MaximumDynamicBacklog"=dword:00001000`
       - `"DynamicBacklogGrowthDelta"=dword:00000010`

    3. These values request a minimum of 20 and a maximum of 1000 available connections. The number of available connections is increased by 10 each time that there is fewer than the minimum number of available connections.

    4. Stop and restart the system.

- **TPC acknowledgements**

  - **Description**: TCP can be the source of some significant remote method delays. You can increase TCP performance by immediately acknowledging incoming TCP segments, in all situations.

    Similarly, to immediately acknowledge incoming TCP segments on a server that runs a Microsoft Windows XP or Windows Server 2003 operating system (not support in Windows Server 2012):

  - **How to view or set TcpAckFrequency:**
    1. Use the "regedit" command and access the

       `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\`

2. On the Edit menu, click New > `DWORD` Value.
3. Name the new value, TcpAckFrequency, and assign it a value of 1.
4. Close the Registry Editor.
5. Stop and restart the system.

7.1.1 Setting up large TCP windows on Windows XP and Windows Server 2003:

For Windows XP (and Windows Server 2003), the maximum receive window size has a number of significant attributes. First, the default value is based on the link speed of the sending interface. The actual value automatically adjusts to even increments of the maximum segment size (MSS) negotiated during TCP connection establishment. To use large windows you have to tune the registry. The maximum window size can be manually configured using:

```
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Glob
alMaxTcpWindowSize
```

The value can be set to a maximum of 65,535 bytes (without window scaling) or 1,073,741,823 (with window scaling). Set to 2 x BDP or 4048,000 as a good starting point.

You can enable window scaling by setting

```
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Tcp1
323Opts
```

Enable scaling by setting the value to 1 or 3. 1 enables windows scaling, 3 enables windows scaling and timestamps. Setting 3 helps where retransmissions are an issue.

7.1.2 Compound TCP – Windows 7, 8, Server 2008 and 2012

To better utilize the bandwidth of TCP connections with high BDP, the Microsoft's Next Generation TCP/IP stack uses Compound TCP (CTCP). CTCP more aggressively increases the send window for connections with large receive window sizes and BDPs. CTCP attempts to maximize throughput on these types of connections by monitoring delay variations and losses. In addition, CTCP ensures that its behaviour does not negatively impact other TCP connections. More details can be found at http://research.microsoft.com/apps/pubs/default.aspx?id=70189

CTCP is enabled by default in computers running Windows Server 2008 and disabled by default in computers running Windows 7 and 8. You can enable CTCP with

```
netsh interface tcp set global congestionprovider=ctcp
```

You can disable CTCP with

```
netsh interface tcp set global congestionprovider=none
```

CTCP has been successful in improving file transfer performance over poor connections and those with a high BDP.

Note: TIME_WAIT and other parameters must still be set for Web applications servers and FMS. Also to take advantage of the windows See Section 5.3.4 for FSC buffer tuning.

7.1.3 Datacentre CTCP – Windows Server 2012 R2 and Windows 8.1

Windows Server 2012 R2 and Windows 8.1 also support DCTCP (Datacentre CTCP). DCTCP leverages Explicit Congestion Notification (ECN) and a simple multibit feedback mechanism at the host. In the data centre, operating with commodity, shallow buffered switches, DCTCP delivers the same or better throughput than TCP, while using 90% less buffer space. Unlike TCP, it also provides high burst tolerance and low latency for short flows. While TCP's limitations restrict the traffic sent, DCTCP enables an application to handle 10X the current background traffic, without impacting foreground traffic. Further, a 10X increase in foreground traffic does not cause any timeouts, thus largely eliminating incast (many to one communication) problems. For more details see Microsoft White Paper "DCTCP: Efficient Packet Transport for the Commoditized Data Center"

7.1.4 Tuning Windows Server 2012 R2 and Windows 8.1

Microsoft has made considerable improvement in network performance with Windows Server 2012 R2 and Windows 8.1. "netsh" is being deprecated and replaced by PowerShell commands. You will find "netsh" is unreliable and should be avoided if possible.

The following commands get information on hardware settings

- `Get-NetAdapter W*n`
- `Get-NetAdapter L*n* -IncludeHidden (e.g. SSTP, PPTP)`
- `NetAdapter.Get-NetAdapterIPsecOffload`
- `NetAdapter.Get-NetAdapterLso`
- `NetAdapter.Get-NetAdapterRss`
- `NetAdapter.Get-NetAdapterRsc`
- `NetAdapter.Get-NetAdapterVmq | Where-Object –FilterScript {$_.Enabled}`
- `NetAdapter.Get-NetAdapterSrioVf –Name "Ethernet 1"`

These commands set specific properties on the hardware

- `NetAdapter.Set-NetAdapterRss –Name "Ethernet 1" –Profile NUMAStatic`
- `NetAdapter.Set-NetAdapterSriov –Name "Ethernet 1" –NumVFs 31 –VPorts 64`

And these commands enable performance features on the hardware

- `NetAdapter.Enable-NetAdapterEncapsulatedPacketTaskOffload` *`MyAdapter`*
- `NetAdapter.Enable-NetAdapterIPsecOffload` *`MyAdapter`*
- `NetAdapter.Enable-NetAdapterLso` *`MyAdapter`* `-IPv4`
- `NetAdapter.Enable-NetAdapterRdma` *`MyAdapter`*
- `NetAdapter.Enable-NetAdapterRsc` *`MyAdapter`*

When should you use these new features? The table below from Microsoft gives suggestions. In general you can regard FMS servers as "Higher Throughput". Web and Enterprise tiers are mainly "Higher Scalability".

Performance Metric	Loopback Fast Path	Registered I/O (RIO)	Large Send Offload (LSO)	Receive Segmentation Offload (RSC)	Receive Side Scaling (RSS)	Virtual Machine Queues (VMQ)	Remote DMA (RDMA)	Single Root I/O Virtual (SR-IOV)
Lower End-to-End Latency	X	X					X	X
Higher Scalability		X			X	X		
Higher Throughput	X	X	X	X	X	X	X	X
Lower Path Length	X	X	X	X			X	X
Lower Variability		X						

Setting templates are provided that can be customized if required. There are five OOTB; Automatic, Compat, Custom, Datacenter and Internet. Datacenter and Internet are configured to give best performance with the two different environments, datacenter and general client. The template contents can be displayed using the PowerShell command "Get-NetTCPSetting". Below are the Datacenter and Internet template contents.

Datacenter

```
SettingName                     : Datacenter
MinRto(ms)                      : 20
InitialCongestionWindow(MSS)    : 4
CongestionProvider              : DCTCP
CwndRestart                     : True
DelayedAckTimeout(ms)           : 10
DelayedAckFrequency             : 2
MemoryPressureProtection        : Disabled
AutoTuningLevelLocal            : Normal
AutoTuningLevelGroupPolicy      : NotConfigured
AutoTuningLevelEffective        : Local
```

```
EcnCapability                 : Disabled
Timestamps                    : Disabled
InitialRto(ms)                : 3000
ScalingHeuristics             : Disabled
DynamicPortRangeStartPort     : 49152
DynamicPortRangeNumberOfPorts : 16384
AutomaticUseCustom            : Disabled
NonSackRttResiliency          : Disabled
ForceWS                       : Disabled
MaxSynRetransmissions         : 2
```

Internet

```
SettingName                   : Internet
MinRto(ms)                    : 300
InitialCongestionWindow(MSS)  : 4
CongestionProvider            : CTCP
CwndRestart                   : False
DelayedAckTimeout(ms)         : 50
DelayedAckFrequency           : 2
MemoryPressureProtection      : Disabled
gAutoTuningLevelLocal          : Normal
AutoTuningLevelGroupPolicy    : NotConfigured
AutoTuningLevelEffective      : Local
EcnCapability                 : Disabled
Timestamps                    : Disabled
InitialRto(ms)                : 3000
ScalingHeuristics             : Disabled
DynamicPortRangeStartPort     : 49152
DynamicPortRangeNumberOfPorts : 16384
AutomaticUseCustom            : Disabled
NonSackRttResiliency          : Disabled
ForceWS                       : Disabled
MaxSynRetransmissions         : 2
```

In general on Windows Server 2012 use the datacenter template for the resource tiers i.e. Web, Enterprise, Database and Resource. Also use this template for Cloud Servers. The default is “Internet” so “Datacenter” must be implanted.

On Windows Server 2012 R2 the templates can be implemented globally to all connections using PowerShell for example:

```
New-NetTransportFilter –SettingName Datacenter
```

For example

```
PS C:\Users|howe> New-NetTransportFilter -SettingName
Datacenter

SettingName       : Datacenter
Protocol          : TCP
LocalPortStart    : 0
LocalPortEnd      : 65535
```

```
RemotePortStart    : 0
RemotePortEnd      : 65535
DestinationPrefix  : *
```

For Windows 8.1 Client the Internet template is a good choice and it is the default. Individual settings are still configurable via 'PowerShell

You cannot modify the standard templates but you can modify custom templates "DatacenterCustom" and "InternetCustom". For example

```
Set-NetTcpSetting -SettingName DatacenterCustom
InitialCongestionWindowMss 10
```

The customer template is implemented using PowerShell

```
New-NetTransportFilter -SettingName DatacenterCustom
```

The current template is displayed under "AppliedSetting" when using

```
Get-NetTcpConnection
```

```
Get-NetTcpConnection

LocalAddress                LocalPort RemoteAddress               RemotePort State       AppliedSetting
------------                --------- -------------               ---------- -----       --------------
::                          55555     ::                          0          Listen


192.168.1.78                51433     161.134.128.154             443        CloseWait   Datacenter
192.168.1.78                51432     161.134.128.154             443        CloseWait   Datacenter
192.168.1.78                51431     161.134.128.154             443        CloseWait   Datacenter
192.168.1.78                51430     161.134.128.154             443        CloseWait   Datacenter
192.168.1.78                51429     161.134.128.154             443        CloseWait   Datacenter
```

7.1.5 Windows Performance Monitor - Server 2012 R2 and Windows 8.1

The performance monitor in Windows Server 2012 also has new additional network performance counters

- Microsoft Winsock BSP
  - Datagrams Drops or Rejected
- NUMA Node Memory
- Per-NUMA memory lists
  - Overall system memory info per Node
  - Network Adapter
- Processor Performance
  - Power related counters
- Physical Network Interface Activity
- RDMA Activity

This link gives good general tuning Window tuning advice for Windows Server 2012 http://msdn.microsoft.com/en-us/library/windows/hardware/dn529133 .

8 Tuning Linux systems

Configure the following settings and variables according to your tuning needs:

- **Linux file descriptors (ulimit)**
    - **Description:** Specifies the maximum number of open files supported. If the value of this parameter is too low, a "too many files open" error is displayed in the Web Application Servers logs. FMS under high load may also require an increase in ulimit. See section 5.3. Some programs have problems with a "ulimit" greater than 1024, so avoid setting values higher than 1024 as the system default.
    - **How to view or set:** Check the UNIX reference pages on the "ulimit" command for the syntax of different shells. To set the ulimit command to 8000 for the KornShell shell (ksh), issue the ulimit -n 8000 command. Use the "ulimit –a" command to display the current values for all limitations on system resources.
    - **Default value:** For SUSE Linux Enterprise Server 9 (SLES 9), the default is 1024.
    - **Recommended value:** 8000
- **timeout_timewait parameter**
    - **Description:** Determines the time that must elapse before TCP/IP can release a closed connection and reuse its resources. This interval between closure and release is known as the TIME_WAIT state or twice the maximum segment lifetime (2MSL) state. During this time, reopening the connection to the client and server costs less than establishing a new connection. By reducing the value of this entry, TCP can release closed connections faster and provide more resources for new connections. When high connection rates occur, a large backlog of the TCP/IP connections accumulates and can slow down both Web application server's FMS server's performance. These servers can stall during certain peak periods. If the server stalls, the "netstat" command shows that many of the sockets that are opened to the HTTP server are in the TME_WAIT or CLOSE_WAIT state. Visible delays can occur for up to four minutes, during which time the server does not send any responses, but CPU utilization may stay high, with all of the activities in system processes. Client may receive RST (reset) and clients report no connection or connection refused.
    - **How to view or set tcp_fin_timeout:**

      Issue the following command to set the timeout_timewait parameter to 30 seconds:

      ```
      echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
      ```

- **Connection backlog**
    - **Description:** Change the following parameters when a high rate of incoming connection requests result in connection failures:
    - ```
      echo 3000 > /proc/sys/net/core/netdev_max_backlog
      echo 3000 > /proc/sys/net/core/somaxconn
      ```

- **TCP_KEEPALIVE_INTERVAL**
    - **Description:** Determines the wait time between isAlive interval probes.
    - **How to view or set tcp_keepalive_intvl:**

      Issue the following command to set the value:
      ```
      echo 15 > /proc/sys/net/ipv4/tcp_keepalive_intvl
      ```
    - **Default value:** 75 seconds
    - **Recommended value:** 15 seconds

- **TCP_KEEPALIVE_PROBES**
    - **Description:** Determines the number of probes before timing out.
    - **How to view or set:** Issue the following command to set the value:
      ```
      echo 5  > /proc/sys/net/ipv4/tcp_keepalive_probes
      ```
    - **Default value:** 9 seconds
    - **Recommended value:** 5 seconds

- **NAGEL**
    - On Linux Nagel is by default off. See section 2.2.3.

8.1.1 Setting up large TCP windows on Linux

Linux kernel 2.6.17 now has sender and receiver side automatic tuning and a 4 MB DEFAULT maximum window size. The Teamcenter supported versions of SUSE and Red Hat currently support use kernel 2.6.13.15 or lower.

For versions earlier than 2.6.17

Use TCP auto-tuning in kernel

- Set /proc/sys/net/ipv4/tcp_moderate_rcvbuf = 1 (1=on)

Tune TCP Max Memory

- /proc/sys/net/ipv4/tcp_rmem
- /proc/sys/net/ipv4/tcp_wmem

The tcp_rmem and tcp_wmem contain arrays of three parameter values: the 3 numbers represent minimum, default and maximum memory values. Tune the maximum memory to 2xBDP
      ```

Tune the socket buffer sizes

- /proc/sys/net/core/rmem_max
- /proc/sys/net/core/wmem_max

Both should be set to 2 x BDP

Ensure that TCP Performance features are enabled

- /proc/sys/net/ipv4/tcp_sack
- /proc/sys/net/ipv4/tcp_window_scaling
- /proc/sys/net/ipv4/tcp_timestamps

8.1.2 Red Hat

Reduce the TIME_WAIT by setting the tcp_fin_timeout kernel value on /proc/sys/net/ipv4/tcp_fin_timeout, using the command echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout to set it to 30 seconds.

Increase the range of ephemeral ports by setting ip_local_port_range kernel value on /proc/sys/net/ipv4/ip_local_port_range, using the command echo "32768 65535" > /proc/sys/net/ipv4/ip_local_port_range, this will set the port range from 32768 to 65535.

The kernel value parameters aren't saved with these commands, and are reset to the default values on system reboot, thus make sure to place the commands on a system start-up script such as /etc/rc.local.

9 Tuning AIX systems

Change the following configuration settings or variables according to your needs. Most of these parameters can be set using system management tool "smit":

- **TCP_TIMEWAIT**
    - **Description:** Determines the time that must elapse before TCP/IP can release a closed connection and reuse its resources. This interval between closure and release is known as the TIME_WAIT state or twice the maximum segment lifetime (2MSL) state. During this time, reopening the connection to the client and server costs less than establishing a new connection. By reducing the value of this entry, TCP/IP can release closed connections faster, providing more resources for new connections. Adjust TCP_TIMEWAIT, if the running application requires rapid release or the creation of new connections, or if a low throughput occurs due to many connections sitting in the TIME_WAIT state.
    - **How to view or set TCP_TIMEWAIT:**
      Issue the following command to set TCP_TIMEWAIT state to 15 seconds:

      usr/sbin/no –o tcp_timewait =1

- **file descriptors (ulimit)**
    - **Description:** Specifies the various restrictions on resource usage on the user account. The "ulimit –a" command displays all the ulimit limits. The "ulimit –a" command specifies only the number of open files that are permitted. The default number of open files setting (2000) is typically sufficient for most applications. If the value set for this parameter is too low, errors might occur when opening files or establishing connections. Because this value limits the number of file descriptors that a server process might open, a value that is too low prevents optimum performance. See section 5.4 for more details.
    - **How to view or set:** Perform the following steps to change the open file limit to 10,000 files:
        - Open the command window.
        - Edit the /etc/security/limits file. Add the following lines to the user account that the WebSphere Application Server process runs on:
          nofiles = 10000
          nofiles_hard = 10000
        - Save the changes.
        - Restart your AIX system.
        - To verify the result, type the "ulimit" -a command on the command line. For example, type # ulimit -a
    - **Default value:** For the AIX operating system, the default setting is 2000.
    - **Recommended value:** The value is application dependent and applies exclusively to application program data and the application stack.

Increasing the ulimit file descriptor limits might improve performance of WebSphere. Increasing some of the other limits might be needed depending on your application. See section 4.3 for advice on FSC requirements and settings.

- **TCP_KEEPIDLE**
  - **Description:** The keepAlive packet ensures that a connection stays in an active/ESTABLISHED state.
  - **How to view or set TCP_KEEPIDLE:** Use the **no** command to determine the current value or to set the value. The change is effective until the next time you restart the machine. To permanently change the value, add the "no" command to the /etc/rc.net directory. For example:

    no -o tcp_keepidle=600

  - **Default value:** 14400 half seconds (2 hours).
  - **Recommended value:** 600 half seconds (5 minutes).
- **TCP_KEEPINTVL**
  - **Description:** Specifies the interval between packets that are sent to validate the connection.
  - **How to view or set:** Use the following command to set the value to 5 seconds:

    no -o tcp_keepintvl=10
  - **Default value:** 150 (1/2 seconds)
  - **Recommended value:** 10 (1/2 seconds)
- **TCP_KEEPINIT**
  - **Description:** Specifies the initial timeout value for TCP connection.
  - How to view or set: Use the following command to set the value to 20 seconds:

    no -o tcp_keepinit=40
  - **Default value:** 150(1/2 seconds)
  - **Recommended value:** 40 (1/2 seconds)
- **TCP_NODELAY**
  - **Description:** Enable and disables the Nagel algorithm. See section 2.3 for more details.
  - **How to view or set:** Use the following command to set disable Nagel :

    no -o tcp_nodelay=1
  - **Default value:** 0
  - **Recommended value:** 1
- **TCP_SENDSPACE**
  - **Description:** Specifies the initial timeout value for TCP connection.
  - **How to view or set:** Use the following command to set the value to 32K:

```
no -x tcp_sendspace
no -o tcp_sendspace=32768
```

- **Default value:** 16K
- **Recommended value:** See section 5.3.2

- **TCP_RECVSPACE**
  - **Description:** Specifies the initial timeout value for TCP connection.
  - **How to view or set tcp_recvspace:** Use the following command to set the value to 32k

```
no -x tcp_recvspace
no -o tcp_recvspace=32678
```

  - **Default value:** 16K
  - **Recommended value:** See section 5.3.2

- **SB_MAX**
  - **Description:** Specifies the initial timeout value for TCP connection.
  - **How to view or set sb_max:** Use the following command to set the value to 20 seconds:

```
no -x sb_max
no -o sb_max= 1048576
```

  - **Default value:** 1048576
  - **Recommended value:** See section 5.3.2

- **RFC1323**
  - **Description:** This option enables TCP to use a larger window size, at the expense of a larger TCP protocol header. This enables TCP to have a 4 GB window size. For adapters that support a 64K MTU (frame size), you must use RFC1323 to gain the best possible TCP performance.
  - **How to view or set rfs1323:** Use the following command to enable RFC 1323:

```
no –x rfc1323
no –o rfc1323=1
```

  - **Default value**: 0 (enable by default AIX 6.1)
  - **Recommended value:** See section 4.3.2

- **TCP_NAGEL_LIMIT**
  - **Description:** This parameter controls the use of the Nagle algorithm. See section 2.2.3.
  - **How to view or set sb_max:** Use the following command to set the value to 20 seconds:

```
no -x tcp_nagel_limit
no -o tcp_nagel_limit= 1
```

  - **Recommended value:** 1, Nagel disabled

9.1.1 Setting up large TCP windows on AIX:

- **Description**: If you want to use large TCP windows with FMS over a high-BDP WAN then set the following :

```
no -o  tcp_recvspace=4000000
no -o tcp_sendspace=4000000
no -o sb_max=8000000
no -o rfc1323=1
```

10 Tuning HP-UX systems

.Configure the following settings and variables according to your tuning needs:

TCP_CONN_REQUEST_MAX

- **Description:** Specifies the maximum number of connection requests that the operating system can queue when the server does not have available threads. When high connection rates occur, a large backlog of TCP/IP connection requests builds up and client connections are dropped. Adjust this setting when clients start to time out after waiting to connect. Verify this situation by issuing the "netstat –p" command. Look for the following value: *connect requests dropped due to full queue*
- **How to view or set:** Use the "ndd" command to determine the current value or to set the value. For example:
  ndd -get /dev/tcp tcp_conn_request_max
  ndd -set /dev/tcp tcp_conn_request_max 8192.

- **Default value:** 4096
- **Recommended value:** In most cases the default is sufficient. Consider adjusting this value to 8192, if the default proves inadequate.

TCP_KEEPALIVE_INTERVAL

- **Description:** Determines the interval between probes.
- **How to view or set:** Use the "ndd" command to determine the current value or to set the value. For example:
  ndd -get /dev/tcp tcp_keepalive_interval
  ndd -set /dev/tcp tcp_keepalive_interval 7200000

- **Default value:** 36,000
- **Recommended value:** 7,200,000 milliseconds

TCP_KEEPALIVES_KILL

- **Description:** Determines the maximum number of times to probe before dropping.
- **How to view tcp tcp_keepalives:** Use the "ndd" command to determine the current value the value. For example:
  ndd -get /dev/tcp tcp_keepalives_kill 5000

- **Default value:** 1
- **Recommended value:** 5000 milliseconds
  (Note: Cannot be modified in HP-UX 11i onwards)

TCP_RECV_HIWATER_DEF

- **Description:** This parameter controls the receive buffers size.
- **How to view or set:** Use the **"**ndd" command to determine the current value or to set the value. For example:

  ndd -get /dev/tcp tcp_recv_hiwater_def
  ndd -set /dev/tcp tcp_recv_hiwater_def 32768

- **Default value:** 32,678 bytes
- **Recommended value:** See Section 5.3.2

TCP_XMIT_HIWATER_DEF

- **Description:** This parameter control the send buffer size.
- **How to view or set:** Use the "ndd" command t**o d**etermine the current value or to set the value. For example:

  ndd -get /dev/tcp tcp_xmit_hiwater_def
  ndd -set /dev/tcp tcp_xmit_hiwater_def 32768

- **Default value:** 32,768 bytes
- **Recommended value:** See Section 5.3.2

TCP_NAGLIM_DEF

- **Description:** This parameter controls the use of the Nagel algorithm. (See section 2.2.3)
- **How to view or set:** Use the "ndd" command t**o d**etermine the current value or to set the value. For example:

  ndd -get /dev/tcp tcp_naglim_def
  ndd -set /dev/tcp tcp_naglim_def 1

- **Recommended value:** 1, that is disable Nagel Algorithm.

10.1.1 Setting up large TCP windows on HPUX:

- **Description**: If you want to use large TCP windows with FMS over a high-BDP WAN then set the following :

  ndd -set /dev/tcp tcp_xmit_hiwater_def 4000000
  ndd -set /dev/tcp tcp_recv_hiwater_def 4000000

11 Network Performance on Virtualized System

Poor network performance on virtualise server can be due to issues unique to that environment. Below are various points that should be considered when investigating network performance on virtualised servers.

- In a native environment, CPU utilization plays a significant role in network throughput. To process higher levels of throughput, more CPU resources are needed. The effect of CPU resource availability on the network throughput of virtualized applications is even more significant. Because insufficient CPU resources will reduce maximum throughput, it is important to monitor the CPU utilization of high-throughput workloads.

- Have more than one network card installed on the physical server and dedicate one network interface to server administration. This means no virtual networks will be configured to use this NIC. For high-workload virtual machines, you might want to dedicate a physical network adapter on the server to the virtual network the virtual machine is using. Ensure virtual machines that share a physical adapter do not oversubscribe to the physical network. Use the Reliability and Performance Monitors to establish a performance baseline for the load and then adjust NIC configurations and loads accordingly.

- When using RSS (receive Side Scaling) consider using more than one vCPU. This is important for server running FSCs.

- Make sure that TSO is enabled and is working. To verify that TSO is working, you may want to use Wireshark. If the packet size going out of the vNIC is larger than the standard MTU, TSO is likely working.

- To establish a network connection between two virtual machines that reside on the same system, connect both virtual machines to the same virtual switch. If the virtual machines are connected to different virtual switches, traffic will go through wire and incur unnecessary CPU and network overhead.

- The vNIC may not support the hardware options the guest OS is expecting. The guest OS may show for example “chimney off load” is enabled but in fact the virtual driver may not support it.

Although the following are VMWare Specific issues the issue raise should be considered for the virtualisation system you are using.

- On VMware the VMkernel network device driver defaults to speed and duplex settings of auto-negotiate. These settings will work correctly with network switches that are also set to auto-negotiate. If your switch is configured for a specific speed and duplex setting, however, you must force the network driver to use the same speed and

duplex setting (for Gigabit links, network settings should be set to auto-negotiate and not forced).

- In ESX 3.5, there are two versions of the vmxnet virtual network adapter, normal vmxnet and enhanced vmxnet.
    - The normal vmxnet device does not support jumbo Ethernet frames or TSO.
    - A virtual machine with an enhanced vmxnet device cannot VMotion to an ESX 3.0.x host.
    - The vmxnet driver from VMware Tools included with ESX 3.5 supports both normal vmxnet and enhanced vmxnet devices. (Note that the vmxnet driver from VMware Tools included with ESX 3.0.x does not support enhanced vmxnet devices.)
    - NIC morphing installs the normal vmxnet device.
    - To use the enhanced vmxnet device you must explicitly select Enhanced vmxnet within the VI Client hardware configuration page.

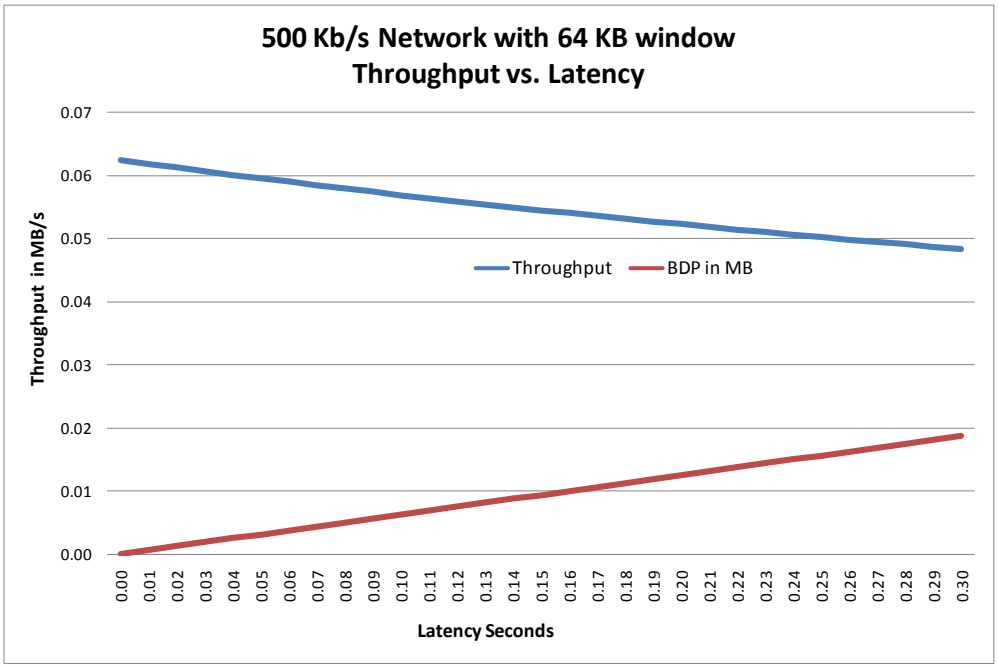
- In ESX 3.5, the enhanced vmxnet device supports jumbo frames for better performance. (Note that the e1000 and vlance devices do not support jumbo frames.) To enable jumbo frames, set the MTU size to 9000 both in the guest and in the virtual switch configuration. The physical NICs at both ends and all the intermediate hops/routers/switches must also support jumbo frames.

- In ESX 3.5, TSO is enabled by default in the VMkernel, but is supported in guests only when they are using the enhanced vmxnet device or the e1000 device. TSO can improve performance even if the underlying hardware does not support TSO.

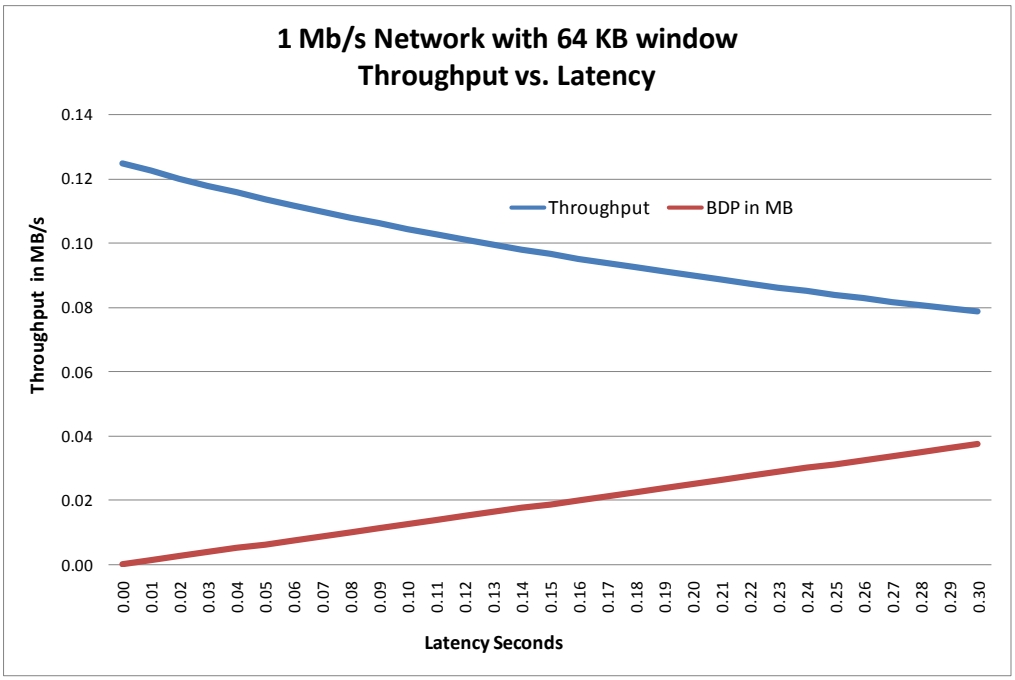
Finally and possibly most import of all check the performance advice and documentation on networking provided by the vendor of the virtualization system you are using. All the vendors provide information on known issue etc. and again this should be check.

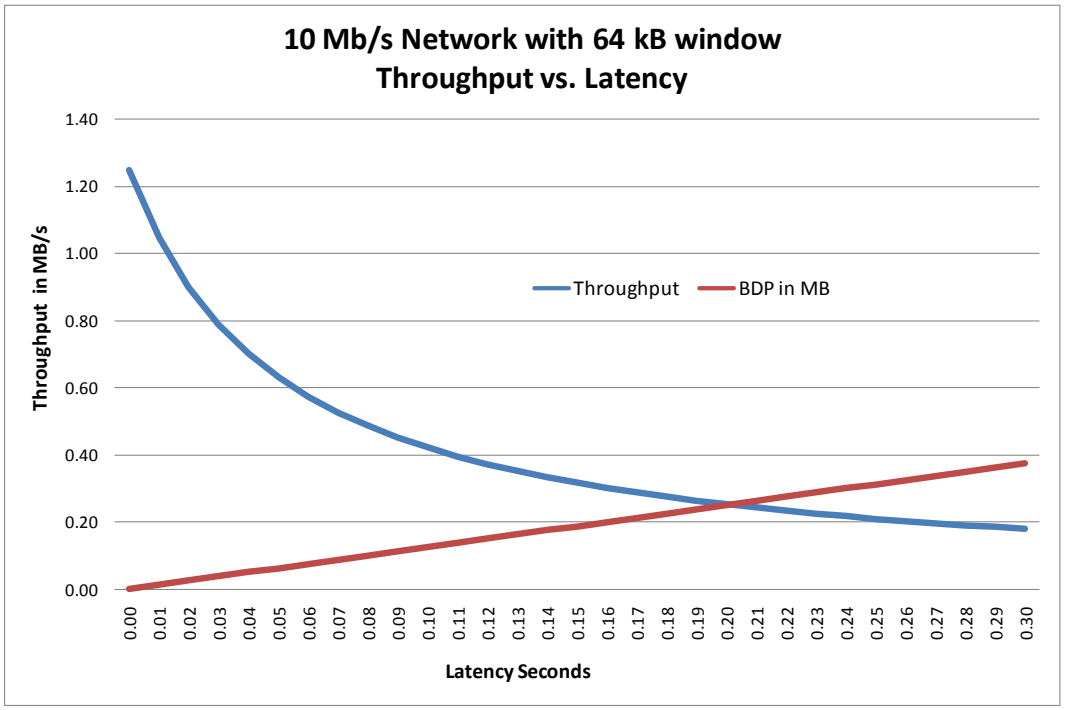
Appendix 1 - Throughput versus latency

Graphs show various throughputs versus latency combinations of bandwidth and window size on a zero data loss connection:

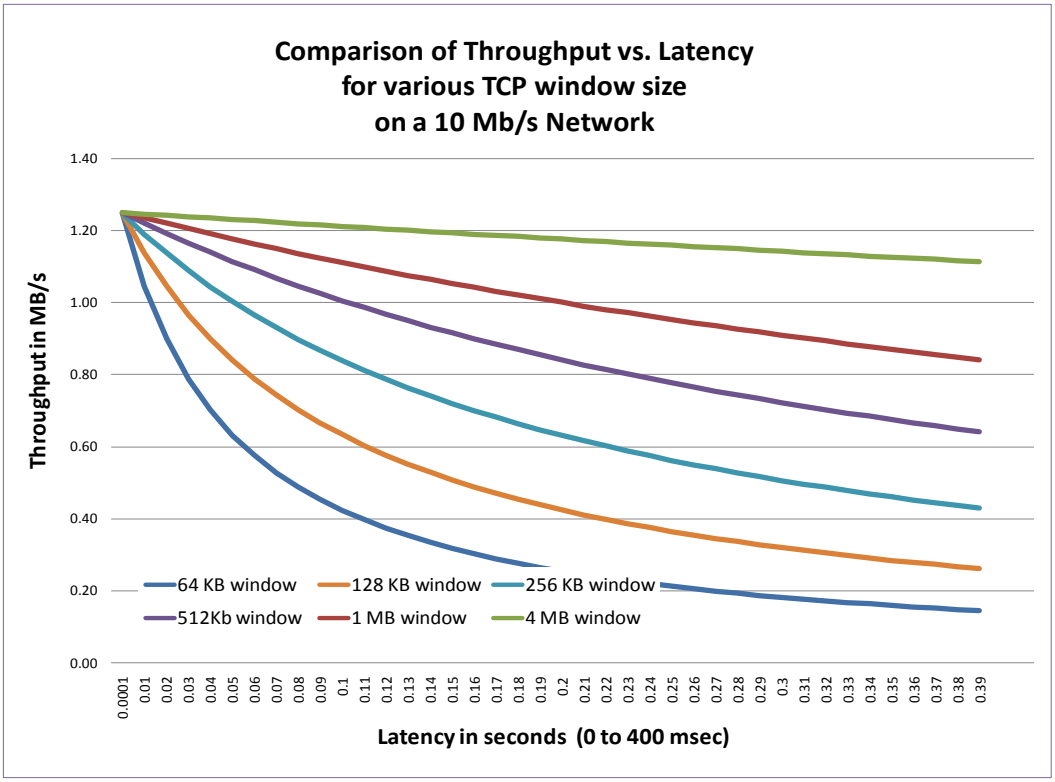


The effect of latency on low-BDP networks is less significant than high-BDP networks





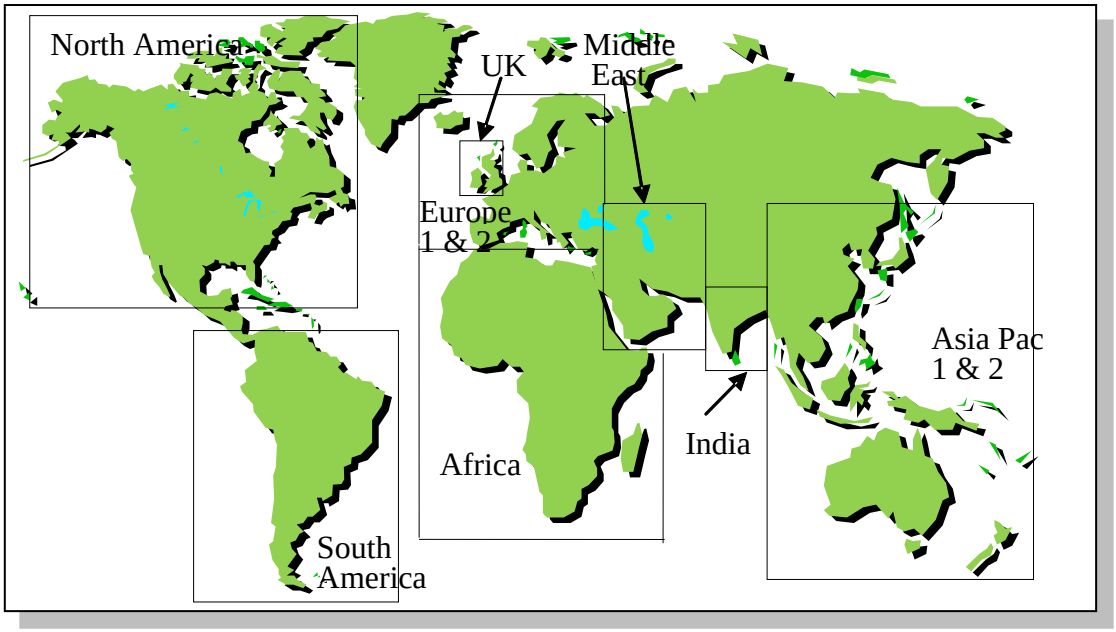
Throughput decreases significantly for high-BDP networks



This graph shows how increasing the window size improves overall throughput of the network.

12 Appendix 1 – Example Global Latency SLA

The following is from BT and gives an example of global SLA BT's MLPS network in 2008. Each provider's SLA (Service Level Agreement) varies and also as this data is from 2008 many would have improved. But these tables provide a good guide to what can be expected. Note the latency is between GpoPs (Global point of Presence), there will be additional latency connecting your sites to the local GpoP or PoP (Point of Presence).



The diagram above shows the BT SLA zones and the table below identifies the GpoP Countries in each SLA zone.

Europe 1	Europe 2	North America	Asia Pacific 1	Middle East
Austria	Estonia	Canada	Australia	Bahrain
Belgium	Bulgaria	Mexico	Hong Kong	Israel
Denmark	Croatia	USA	Japan	Kuwait
Finland	Cyprus		Singapore	Lebanon
France	Czech Rep.	**South America**		Saudi Arabia
Germany	Greece	Argentina	**Asia Pacific 2**	UA Emirates
Ireland	Hungary	Brazil	China	
Italy	Poland	Chile	Indonesia	**Africa**
Luxembourg	Romania	Columbia	Philippines	Egypt
Netherlands	Russia	Peru	South Korea	Morocco
Norway	Slovakia	Venezuela	Malaysia	South Africa
Portugal	Slovenia		New Zealand	
Spain	Turkey	**India**	Thailand	**None**
Sweden	Ukraine	India	Taiwan	Kazakhstan
Switzerland		Pakistan		
UK (including Enhanced Access)				

SLAs for network performance are then specified within a zone and between zones. The UK domestic services network is considered as a separate zone as it is so extensive, and for historic reasons 7 other European countries/regions also have separate SLAs, namely: Belgium, France, Germany, Ireland, Netherlands, the Nordics region and Spain.

However, for SLAs between other zones and the European countries/regions (i.e. UK and the 7 others), the European countries/regions are considered to be included within the Europe 1 zone.

International long lined sites are considered to be in the zone of their parent GpoP, so for instance a site in French Guiana long-lined from Paris is considered to be Europe 1 for zonal SLAs, but neither in South America nor France.

12.1 Round Trip Delay

These tables use Round Trip Delay (RTD) rather than the less specific term latency. RTD is the time taken for a packet to get to its destination and for its acknowledgement to return. RTD is measured by sending a short sequence of time stamped test packets and recording the time delay when the acknowledgements return. The sequence of test packets is ten test packets of 80 bytes for class 1 (EF), ten test packets of 100 bytes for Class 2 (AF) and two test packets of 100 bytes for the DE class. This is repeated nominally every minute, 24 hours a day and 365 days a year.

For the Core RTD SLA an average of the RTD values over a calendar month is reported for each class. This table is from 2008 so is only a guide.

Core SLA Zone to SLA Zone	EF RTD (in ms)	AF RTD (in ms)	DE RTD (in ms)
Within Belgium	8	9	20
Within France	20	23	31
Within Germany	30	35	40
Within Ireland	16	19	27
Within India	46	49	55
Within the Netherlands	9	10	16
Within the Nordics	25	27	33
Within Spain	21	23	30
Within the UK (including Enhanced Access)	20	23	30
Within Europe Region 1	40	45	60
Europe Region 1 to Europe Region 2	80	85	100
Within Europe Region 2	110	120	135
Amsterdam/London to New York	95	100	120
Europe Region 1 to North America	145	150	160
Europe Region 2 to North America	180	185	195
Within North America	65	70	80

Asia Pac Region 1 to Europe Region 1	320	325	335
Asia Pac Region 2 to Europe Region 1	320	325	340
Asia Pac Region 1 to Europe Region 2	340	350	365
Asia Pac Region 2 to Europe Region 2	340	350	365
Asia Pac Region 1 to North America	230	235	245
Asia Pac Region 2 to North America	260	265	280
Within Asia Pac (Regions 1 and 2)	110	120	150
Within India	46	49	55
India Region to Europe Region 1	225	230	240
India Region to Europe Region 2	250	260	280
India Region to North America	310	320	340
India Region to Asia Pac Region 1	135	140	150
India Region to Asia Pac Region 2	135	140	155
Within Africa	TBA	TBA	TBA
Africa to Europe Region 1	220	230	245
Africa to Europe Region 2	235	245	260
Africa to North America	350	360	375
Africa to Asia Pac Region 1	320	330	345
Africa to Asia Pac Region 2	320	330	345
Africa to India	260	270	285
Within South America	100	110	125
South America to Europe Region 1	290	295	300
South America to Europe Region 2	325	335	345
South America to North America	215	225	235
South America to Asia Pac Region 1	440	450	460
South America to Asia Pac Region 2	440	455	465
South America to India	420	430	450
South America to Africa	500	510	530
Within Middle East	165	180	200
Middle East to Europe Region 1	140	150	165
Middle East to Europe Region 2	160	170	185
Middle East to North America	245	255	270
Middle East to South America	395	405	420
Middle East to Asia Pac Region 1	290	300	315
Middle East to Asia Pac Region 2	290	300	315
Middle East to India	230	240	255
Middle East to Africa	300	310	325
Middle East to South America	395	405	420

13 Appendix – 2 Using Fiddler2 to analyze RAC traffic

Fiddler2 (http://fiddler2.com) is a debugging proxy that can be used to debug Web Browsers and other application such as Teamcenter RAC (9.1 onwards). It can decrypt HTTPS traffic and is simpler to install than other HTTPS decryption methods.

To use it configure Teamcenter to use TCC with a forward proxy, this is detailed in the documentation. This can be done using either TEM or editing tccs/fwdproxy_cfg. Configure the proxy on 127.0.0.1 port 8888

Decryption of HTTPS relies on the RAC using the Fiddler2 certificate. You will need to install this certificate into the browser certificate store (http://fiddler2.com/documentation/Configure-Fiddler/Tasks/TrustFiddlerRootCert) TCCS should access the certificate from there and accept it.

14 Appendix – 3 Sysinternals Networking Utilities

Sysinternals tools can be down loaded from (http://technet.microsoft.com/en-us/sysinternals/default)

The psing tool can use either ICMP or its own TCP based ping. The TCP pings avoid the network device spooking of ICMP and also give a more accurate value from the OS level. The TCP ping relies on a client and server so can only be used with Windows servers and client. The utility will also carry out a basic bandwidth test which again reflect the server performance through the WAN rather than just the expect bandwidth.

15 Appendix – 4 Testing Network Performance with NTttcp

Microsoft provides NTttcp to help measure network driver performance and throughput on different network topologies and hardware setups. (Download from http://msdn.microsoft.com/en-gb/windows/hardware/gg463264.aspx)

Like the Sysinternal utility (psping) in client server mode it can only be used between two Windows systems.

When setting up NTttcp, consider the following:

- A single thread should be sufficient for optimal throughput.
- Multiple threads are required only for single-to-many clients.
- Posting enough user receive buffers (by increasing the value passed to the **-a** option) reduces TCP copying.
- You should not excessively post user receive buffers because the first buffers that are posted would return before you need to use other buffers.
- It is best to bind each set of threads to a logical processor (the second delimited parameter in the **-m** option).
- Each thread creates a logical processor that connects to (listens) a different port.

Example Syntax for NTttcp Sender and Receiver

Syntax	Details
Example Syntax for a Sender NTttcps –m 1,0,10.1.2.3 –a 2	Single thread. Bound to CPU 0. Connects to a computer that uses IP 10.1.2.3. Posts two send-overlapped buffers. Default buffer size: 64 K. Default number of buffers to send: 20 K.
Example Syntax for a Receiver NTttcpr –m 1,0,10.1.2.3 –a 6 –fr	Single thread. Bound to CPU 0. Binds on local computer to IP 10.1.2.3. Posts six receive-overlapped buffers. Default buffer size: 64 KB. Default number of buffers to receive: 20 K. Posts full-length (64 K) receive buffers.

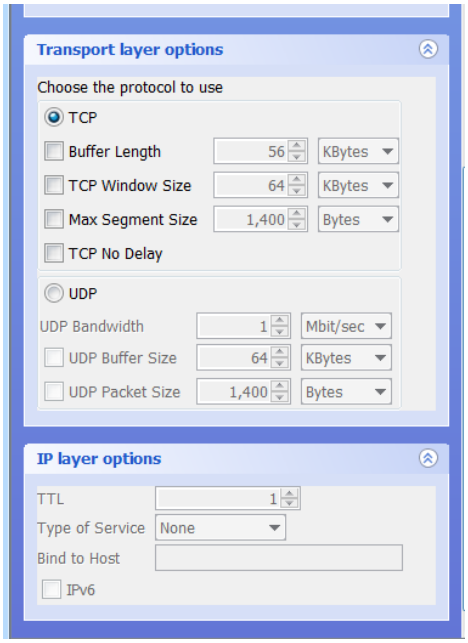
Note Make sure that you enable all offloading features on the network adapter.

TCP/IP Window Size

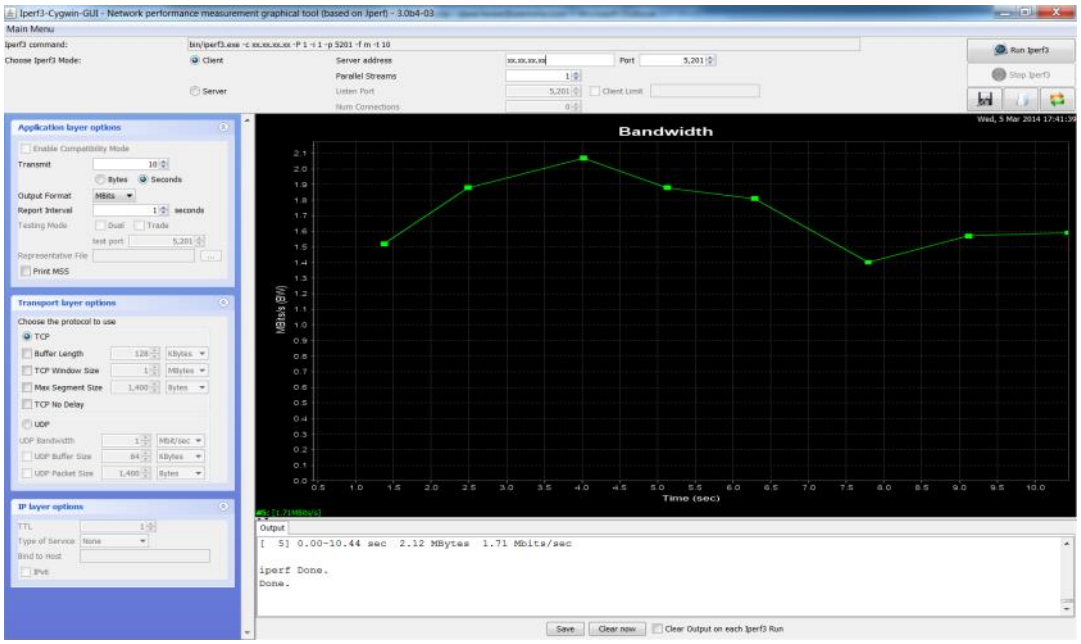
For 1 GB adapters, the settings shown in previous table should provide good throughput because NTttcp sets the default TCP window size to 64 K through a specific logical processor option (SO_RCVBUF) for the connection. This provides good performance on a low-latency network. In contrast, for high-latency networks or for 10 GB adapters, the default TCP window size value for NTttcp yields less than optimal performance. In both cases, you must adjust the TCP window size to allow for the larger bandwidth delay product. You can statically set the TCP window size to a large value by using the -rb option. This option disables TCP Window Auto-Tuning, and we recommend using it only if the user fully understands the resultant change in TCP/IP behaviour. By default, the TCP window size is set at a sufficient value and adjusts only under heavy load or over high-latency links.

16 Appendix – 5 Testing Network Performance with iPerf3

Like NTttcp and 'psping' iPef3 is a traffic generating tool that can be used to find network performance issue as well as measure bandwidth. As with the other tools it uses a client and server to measure the network. iPerf also supports various tuning parameter that can be adjusted to find the optimal settings for example the optimal FSC buffer size (see section 5.3.2 on tuning FSC buffers)



Combined with Wireshark it is possible to detect issues such as queuing (increasing and decreasing throughput). It is available for both Windows and UNIX. The Windows GUI version can be found at http://code.google.com/p/iperf3-cygwin-gui/ and the iPerf home is at GitHub https://github.com/esnet/iperf



17 Appendix 6 – Tools for managing trace files

There are various tools provided with Wireshark and by others to assist in handling large file or continuous tracing.

17.1 Sanitizing trace data

Customer are often concern network files with contain their proprietary data, IP addresses etc., Trace Wrangler (www.tracewrangler.com) can be used to sanitize pcapng files by removing or replacing sensitive data.

17.2 Wireshark Utilities

Often over looked Wireshark has several utilities provided with located in \Program Files\Wireshark along with documentation on them.

- Capinfo – return statistics e.g. current file size, number of packets etc. of the pcap file.
- Dumpcap – command line packet capture tool.
- Editcap - can be used to split large pcap files.
- Mergecap – used to combine multiplr capture files.
- Rawshark – read a stream of packets from a file or pipe and prints a line describing its output.
- Reordercap – allows a pcap file to time order, useful for use with merged pcap files.
- Text2pcap – convert an ASICC hex dump for network traffic to a pcap file.
- Tshark – network protocol analyzer. Produce pcap file that can be analyzed by Wireshark. Many useful options including a ring buffer that allows long term captures.

18 References

Blue Coat – WAN Acceleration Technologies
http://www.bluecoat.com/

Cisco - WAN Acceleration Technologies (WAAS)
http://www.cisco.com/go/waas

EDS - Performance Analysis of Wide Area Networks and Network Applications
Dr. Klaus Schmidt 05 Oct 2002

HP - Annotated Output of "ndd -h" (HPUX)
ftp://ftp.cup.hp.com/dist/networking/briefs/annotated_ndd.txt

IBM - Tuning for server environments
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tprf_tuneprf.html

IBM- AIX Performance and Tuning Guide
http://www.redbooks.ibm.com/redbooks/pdfs/sg246478.pdf

IBM - Performance Management Guide
http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/prftungd/netperf3.htm

Internet Engineering Task Force
http://www.ietf.org

A Compound TCP Approach for High-speed and Long Distance Networks - in Proc. IEEE Infocom, April 2006 K. Tan, Jingmin Song, Qian Zhang, Murari Sridharan,

Microsoft - The Cable Guy TCP Receive Window Auto-Tuning
http://technet.microsoft.com/en-us/magazine/2007.01.cableguy.aspx

Microsoft - Performance Tuning Guidelines for Windows Server 2012 R2
http://msdn.microsoft.com/en-us/library/windows/hardware/dn529133

Microsoft - Performance Tuning Guidelines for Windows Server 2008
http://download.microsoft.com/download/b/b/5/bb50037f-e4ae-40d1-a898-7cdfcf0ee9d8/All-Up/WS08PerformanceTuningGuideFinal_En.docx

Microsoft - Performance Tuning Guidelines for Windows Server 2003
http://download.microsoft.com/download/2/8/0/2800a518-7ac6-4aac-bd85-74d2c52e1ec6/tuning.doc

Oracle® Database Net Services Administrator's Guide 10g Release 2 (10.2)

Riverbed – WAN Acceleration Technologies
http://www.riverbed.com/products/appliances/

SUN Blueprint – Understanding Tuning TCP
Deepak Kakadia
http://www.sun.com/blueprints/0304/817-5773.pdf

SUN - Solaris Tuneable Parameters Reference Manual
http://docs.sun.com/app/docs/doc/806-7009

SUN Blueprint – Ethernet Autonegotiations Best Practices
Jim Eggers and Steve Hodnet
http://www.sun.com/blueprints

TeleGeography – Global Internet Maps
http://www.telegeography.com/product-info/map_internet/images/internet_map09_lg.gif

Troubleshooting with Wireshark
Laura Chappell
http://www.wiresharkbook.com

Windows Sysinternals – tools to manage and trouble shoot Windows
http://technet.microsoft.com/en-us/sysinternals/default