

Performance Enhancements in the Next Generation of TCP-IP Stack

Joseph Davies

This article describes the performance enhancements of the Next Generation TCP/IP stack, which include Receive Window Auto-Tuning, Compound TCP, Explicit Congestion Notification (ECN), enhancements for wireless traffic, and improved routing path detection and recovery.

Introduction

The TCP/IP stack in Microsoft Windows 2000 introduced a set of performance enhancements that included TCP receive window scaling, selective acknowledgments, and better roundtrip time (RTT) estimation. In the years since Windows 2000 was released, there have been many changes to networking bandwidth, including more prevalent use of high-bandwidth and wireless links. The Next Generation TCP/IP stack in Windows Vista and Windows Server 2008 includes a new set of performance enhancements to increase throughput in high-bandwidth, high-latency, and high-loss networking environments.

This article describes the following performance enhancements:

- Receive window auto-tuning
- Compound TCP
- ECN support
- Enhancements for wireless traffic
- Improved routing path detection and recovery

Receive Window Auto-Tuning

The TCP receive window size is the amount of data that a TCP receiver allows a TCP sender to send before having to wait for an acknowledgement. After the connection is established, the receive window size is advertised in each TCP segment. Advertising the maximum amount of data that the sender can send is a receiver-side flow control mechanism that prevents the sender from sending data that the receiver cannot store. A sending host can only send at a maximum the amount of data advertised by the receiver before waiting for an acknowledgment and a receive window size update.

Receive Windows in Windows Server 2003 and Windows XP

For the TCP/IP stack in Windows Server 2003 and Windows XP, the maximum receive window size:

- Has a default value 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.
- Can be manually configured. The

```
HKEY_LOCAL_MACHINE\System  
\CurrentControlSet\Services\Tcpip\Parameters\TCPWindowSize
```

and

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters\Interface  
\TCPWindowSize
```

registry values can be set to a maximum of 65,535 bytes (without window scaling) or 1,073,741,823 (with window scaling). Without window scaling, you can only achieve a throughput of approximately 5 megabits per second (Mbps) on a path with a 100 millisecond RTT, regardless of the path bandwidth.

Performance Enhancements in the Next Generation of TCP-IP Stack

Joseph Davies

- Can be scaled up to 1 gigabyte (GB) with window scaling Window scaling, defined in RFC 1323, allows TCP to negotiate a scaling factor for the window size during connection establishment. You can enable window scaling by setting the

`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters\Tcp1323Opts`

registry value to 1 or 3. By default, window scaling is only used on a connection if the received synchronize (SYN) segment contains the Window Scale option.

- Can be specified by an application An application can specify the maximum receive window size for a connection by using the `SO_RCVBUF` Windows Sockets option when a connection is initiated. For window scaling, the application must specify a window size larger than 65,535 bytes.

The correct value for the receive window is often difficult to determine. In order to fill the capacity of the network between the sender and receiver, the window size should be set to the bandwidth-delay product for the connection, which is the bandwidth multiplied by the round-trip time. Even if you correctly determine the bandwidth-delay product, you do not know how quickly the receiving application will retrieve data from the incoming data buffer (the application retrieve rate).

Despite the support for scalable windows, the maximum receive window size in Windows Server 2003 and Windows XP can still limit throughput because it is a fixed maximum size for all TCP connections (unless specified by the application), which can enhance throughput for some connections and decrease throughput for others. Additionally, the fixed maximum receive window size for a TCP connection does not vary with changing network conditions.

Receive Windows in the Next Generation TCP/IP Stack

To solve the problem of correctly determining the value of the maximum receive window size for a connection based on the current conditions of the network, the Next Generation TCP/IP stack supports Receive Window Auto-Tuning. Receive Window Auto-Tuning continually determines the optimal receive window size by measuring the bandwidth-delay product and the application retrieve rate, and adjusts the maximum receive window size based on changing network conditions.

Receive Window Auto-Tuning enables TCP window scaling by default, allowing up to a 16 MB window size. As the data flows over the connection, the Next Generation TCP/IP stack monitors the connection, measures the current bandwidth-delay product for the connection and the application receive rate, and adjusts the receive window size to optimize throughput. The Next Generation TCP/IP stack no longer uses the `TCPWindowSize` registry values.

With better throughput between TCP peers, the utilization of network bandwidth increases during data transfer. If all the applications are optimized to receive TCP data, then the overall utilization of the network can increase substantially, making the use of Quality of Service (QoS) more important on networks that are operating at or near capacity.

Note Some Internet gateway devices and firewalls block packet flows because they do not correctly interpret the scaling factor used in TCP connections. Because of this, Internet Explorer in Windows Vista uses an initial scaling factor of 2. Other applications use a default initial scaling factor of 8. Microsoft is investigating changing the initial scaling factor for Internet Explorer-based connections to 8 in a future update of Windows Vista. Microsoft is working with the manufacturers of these devices so that they can be updated for compliance with TCP window scaling.

Performance Enhancements in the Next Generation of TCP-IP Stack

Joseph Davies

Compound TCP

The existing algorithms that prevent a sending TCP peer from overwhelming the network are known as slow start and congestion avoidance. These algorithms increase the amount of segments that the sender can send, known as the send window, when initially sending data on the connection and when recovering from a lost segment. Slow start increases the send window by one full TCP segment for either each acknowledgement segment received (for TCP in Windows XP and Windows Server 2003) or for each segment acknowledged (for TCP in Windows Vista and Windows Server 2008). Congestion avoidance increases the send window by one full TCP segment for each full window of data that is acknowledged.

These algorithms work well for LAN media speeds and smaller TCP window sizes. However, when you have a TCP connection with a large receive window size and a large bandwidth-delay product (high bandwidth and high delay), such as replicating data between two servers located across a high-speed WAN link with a 100 ms round trip time, these algorithms do not increase the send window fast enough to fully utilize the bandwidth of the connection. For example, on a 1 Gigabit per second (Gbps) WAN link with a 100 ms round trip time (RTT), it can take up to an hour for the send window to initially increase to the large window size being advertised by the receiver and to recover when there are lost segments.

To better utilize the bandwidth of TCP connections in these situations, the Next Generation TCP/IP stack includes Compound TCP (CTCP). CTCP more aggressively increases the send window for connections with large receive window sizes and large bandwidth-delay products. CTCP attempts to maximize throughput on these types of connections by monitoring delay variations and losses. CTCP also ensures that its behavior does not negatively impact other TCP connections.

In testing performed internally at Microsoft, large file backup times were reduced by almost half for a 1 Gbps connection with a 50ms RTT. Connections with a larger bandwidth delay product can have even better performance. CTCP and Receive Window Auto-Tuning work together for increased link utilization and can result in substantial performance gains for large bandwidth-delay product connections.

CTCP is enabled by default in computers running Windows Server 2008 and disabled by default in computers running Windows Vista. You can enable CTCP with the netsh interface tcp set global congestionprovider=ctcp command. You can disable CTCP with the netsh interface tcp set global congestionprovider=none command.

ECN Support

When a TCP segment is lost, TCP assumes that the segment was lost due to congestion at a router and performs congestion control, which dramatically lowers the TCP sender's transmission rate. With Explicit Congestion Notification (ECN) support on both TCP peers and in the routing infrastructure, routers experiencing congestion mark the packets as they forward them. TCP peers receiving marked packets lower their transmission rate to ease congestion and prevent segment losses. Detecting congestion before packet losses are incurred increases the overall throughput between TCP peers. Windows Vista supports ECN but it is disabled by default. You can enable ECN support with the:

```
netsh interface tcp set global ecncapability=enabled
```

command.

Performance Enhancements in the Next Generation of TCP-IP Stack

Joseph Davies

ECN is described in RFC 3168.

Enhancements for Wireless Traffic

Wireless networks such as those based on IEEE 802.11, General Packet Radio Service (GPRS), or Universal Mobile Telecommunications System (UMTS) provide mobility and ubiquitous connectivity for portable computing devices. However, wireless networks can also have high packet losses depending on network conditions, signal attenuation, electromagnetic interference, and the changing location of the mobile computer. In high-loss networking environments, frequent TCP timeouts and retransmissions can dramatically decrease throughput.

The Next Generation TCP/IP stack supports the following RFCs to optimize throughput in high-loss environments:

- **RFC 2582:** The NewReno Modification to TCP's Fast Recovery Algorithm

The Fast Recovery algorithm, defined in RFC 2001, is based on the Reno algorithm, which increases the amount of data that a sender can send when a segment is retransmitted due to a fast retransmit event. Although the Reno algorithm works well for single lost segments, it does not perform as well when there are multiple lost segments. The NewReno algorithm provides faster throughput by changing the way that a sender can increase their sending rate during fast recovery when multiple segments in a window of data are lost and the sender receives a partial acknowledgement (an acknowledgement for only part of the data that has been successfully received).

- **RFC 2883:** An Extension to the Selective Acknowledgement (SACK) Option for TCP

SACK, defined in RFC 2018, allows a receiver to indicate up to four noncontiguous blocks of received data. RFC 2883 defines an additional use of the fields in the SACK TCP option to acknowledge duplicate packets. This allows the sender of the TCP segment containing the SACK option to determine when it has retransmitted a segment unnecessarily and adjust its behavior to prevent future retransmissions. The fewer retransmissions that are sent, the better the overall throughput.

- **RFC 3517:** A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP

The current implementation of TCP/IP in Windows Server 2003 and Windows XP uses SACK information only to determine which TCP segments have not arrived at the destination. RFC 3517 defines a method of using SACK information to perform loss recovery when duplicate acknowledgements have been received, replacing the fast recovery algorithm when SACK is enabled on a connection. The Next Generation TCP/IP stack keeps track of SACK information on a per-connection basis and monitors incoming acknowledgements and duplicate acknowledgements to more quickly recover when multiple segments are not received at the destination.

- **RFC 4138:** Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP and the Stream Control Transmission Protocol (SCTP)

Spurious retransmissions of TCP segments can occur when there is a sudden and temporary increase in the RTT. When the increase occurs, the retransmission timeouts (RTOs) of previously

Performance Enhancements in the Next Generation of TCP-IP Stack

Joseph Davies

sent segments begin to expire and TCP starts retransmitting them. If the increase occurs just before sending a full window of data, a sender can retransmit the entire window of data. The F-RTO algorithm prevents spurious retransmission of TCP segments through the following behavior:

When the RTO expires for multiple segments, TCP retransmits just the first segment. When the first acknowledgement is received, TCP begins sending new segments (if allowed by the advertised window size). If the next acknowledgement acknowledges the other segments that have timed out but have not been retransmitted, TCP determines that the timeout was spurious and does not retransmit the other segments that have timed out.

- The result of this behavior is that for environments that have sudden and temporary increases in the RTT, such as when a wireless client roams from one wireless AP to another, F-RTO prevents unnecessary retransmission of segments and more quickly returns to its normal sending rate. The use of SACK-based loss recovery and F-RTO are best suited for connections that use GPRS links.

Improved Routing Path Detection and Recovery

To automatically attempt a new routing path for remote traffic when TCP connections begin retransmitting segments, the TCP/IP stack in Windows Server 2003 and Windows XP supports dead gateway detection. Dead gateway detection is a fail-over mechanism that automatically changes the default gateway of a computer to the next configured default gateway (when a computer has been configured with multiple default gateways on an interface). However, dead gateway detection in Windows Server 2003 and Windows XP does not do the following:

- Distinguish whether the local default gateway has failed or a remote gateway (router) has failed.
- Provide a fail-back method to change the default gateway back to the primary default gateway when the primary routing path is restored.

The Next Generation TCP/IP stack addresses these issues through Neighbor Unreachability Detection for IPv4 and fail-back support for an automatically changed default gateway configuration.

Neighbor Unreachability Detection for IPv4

Neighbor Unreachability Detection is a feature of IPv6 that allows a node to track whether a neighboring node is reachable. A neighboring node is reachable if there has been a recent confirmation that IPv6 packets sent to the neighboring node were received and processed by the neighboring node. IPv6 tracks reachability through an exchange of unicast Neighbor Solicitation and Neighbor Advertisement messages or by relying on upper layer protocols such as TCP to indicate that a connection is sending and receiving data. With Neighbor Unreachability Detection, an IPv6 node can determine when a neighboring node is no longer reachable and report the error condition to other components and applications.

The Next Generation TCP/IP stack also supports Neighbor Unreachability Detection for IPv4 traffic by tracking the reachable state of IPv4 neighbors in the IPv4 route cache. IPv4 Neighbor Unreachability Detection determines reachability through an exchange of unicast Address Resolution Protocol (ARP) Request and ARP Reply messages or by relying on upper layer protocols such as TCP. With IPv4 Neighbor Unreachability Detection, IPv4-based communications benefit by determining when neighboring nodes, including routers, are no longer reachable and reporting the condition.

Performance Enhancements in the Next Generation of TCP-IP Stack

Joseph Davies

With the Next Generation TCP/IP stack, if TCP connections begin to retransmit segments, the stack tries to determine whether the current default gateway is still reachable by sending a unicast ARP Request message. If the default router does not reply, the Next Generation TCP/IP stack changes the default gateway to the next one in the list. If the default router replies, dead gateway detection eventually changes the default gateway to the next one in the list.

Fail-back Support for Default Gateway Changes

In TCP/IP for Windows Server 2003 and Windows XP, when dead gateway detection changes the default gateway, the new default gateway remains the primary gateway for default route traffic until dead gateway detection switches to the next one in the list (cycling through the list of default gateways) or until the computer is restarted. Therefore, dead gateway detection in TC/IP for Windows Server 2003 and Windows XP provides a fail-over function, but not a fail-back function.

The lack of fail-back for default gateways can cause throughput problems on a subnet containing two routers: a high-capacity primary router and a lower capacity backup router. If the high-capacity router has a temporary failure, dead gateway detection could cause hosts on the subnet to switch over to the backup router. When the high-capacity router becomes available again, none of the hosts on the network use it because they have switched to the backup router.

The Next Generation TCP/IP stack provides fail-back for dead gateways by periodically attempting to send TCP traffic through the dead gateway. If the TCP traffic sent through the dead gateway is successful, the Next Generation TCP/IP stack switches the default gateway to the previously determined dead gateway.

In our example with the high-capacity router and lower-capacity backup router, if the high-capacity router becomes unavailable, the hosts on the subnet switch their default gateways to the backup router and periodically attempt to send TCP traffic through the high-capacity router. When the high-capacity router becomes available and hosts determine that TCP traffic sent through the high-capacity router is successfully received, the hosts on the subnet switch their default gateway back to the high-capacity router.

Support for fail-back to primary default gateways can provide faster throughput by sending traffic through the primary default gateway on the subnet.