Impact of TCP Window Size on a File Transfer

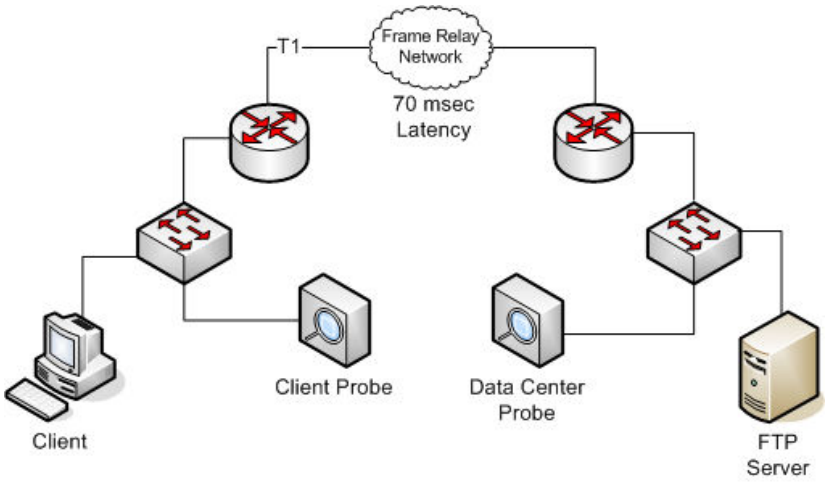
Key Concept—This example shows how AppTransaction Xpert diagnoses and visualizes application and network problems; it is not a step-by-step tutorial. If you have experience with AppTransaction Xpert, you can recreate this study by following the instructions in Recreate the Example. The screen images in this example were captured while running AppTransaction Xpert with the AppTransaction Xpert Decode Module (ADM) installed. If you do not have ADM installed, some screens might look different.

In this network, the client connects to the FTP server, located in the data center, through a WAN that consists of a T1 circuit (1.544 Mbps). When the client performs a 1 MB file transfer, the response time is about 25 seconds. Given the file size and the data rate, the ideal response time is about 5 seconds (1 MB x 8 bits/byte / 1.544 Mbps).

Possible causes for the slow response time fall into two general categories: network bottlenecks and server bottlenecks. This study identifies the cause of the performance problems and recommends possible solutions.

We begin our investigation by identifying and capturing a transaction of the file transfer. A probe was placed on both the client site and the data center, as shown in the following figure.

Figure 10-1 Network Diagram

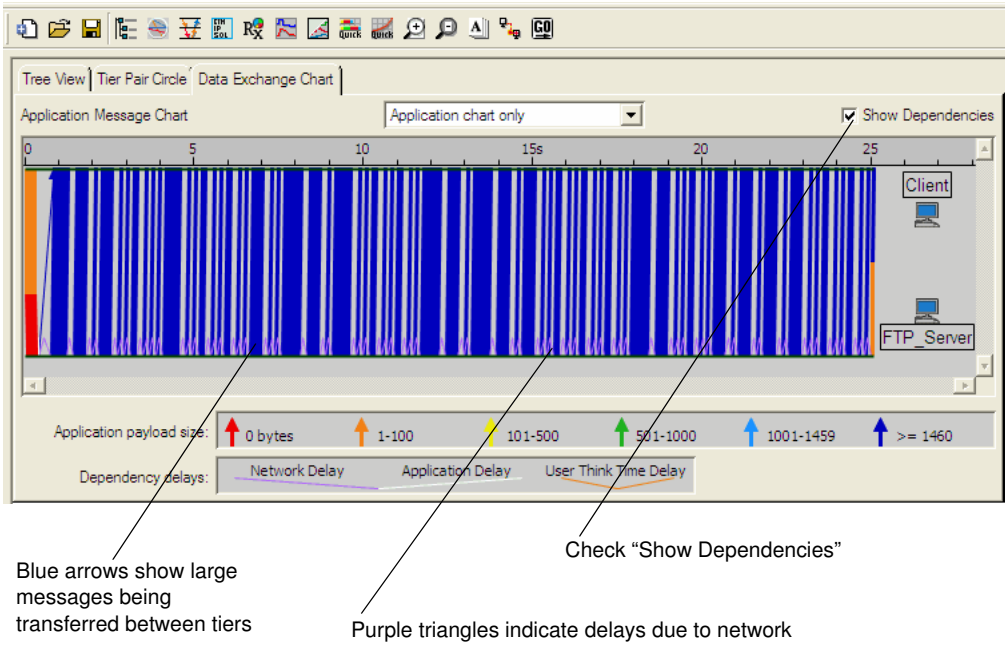


Packet trace captures were taken simultaneously at both the client and FTP server, then merged and synchronized to obtain the best possible analysis of delays at each tier and the network. A single probe would have sufficed, but it was not difficult to obtain data from both probes.

AppTransaction Xpert Analysis

After opening the packet trace into AppTransaction Xpert, we looked at the transaction in the Data Exchange Chart, which shows the flow of application messages between tiers over time. You might have to adjust the window size to match the scale shown.

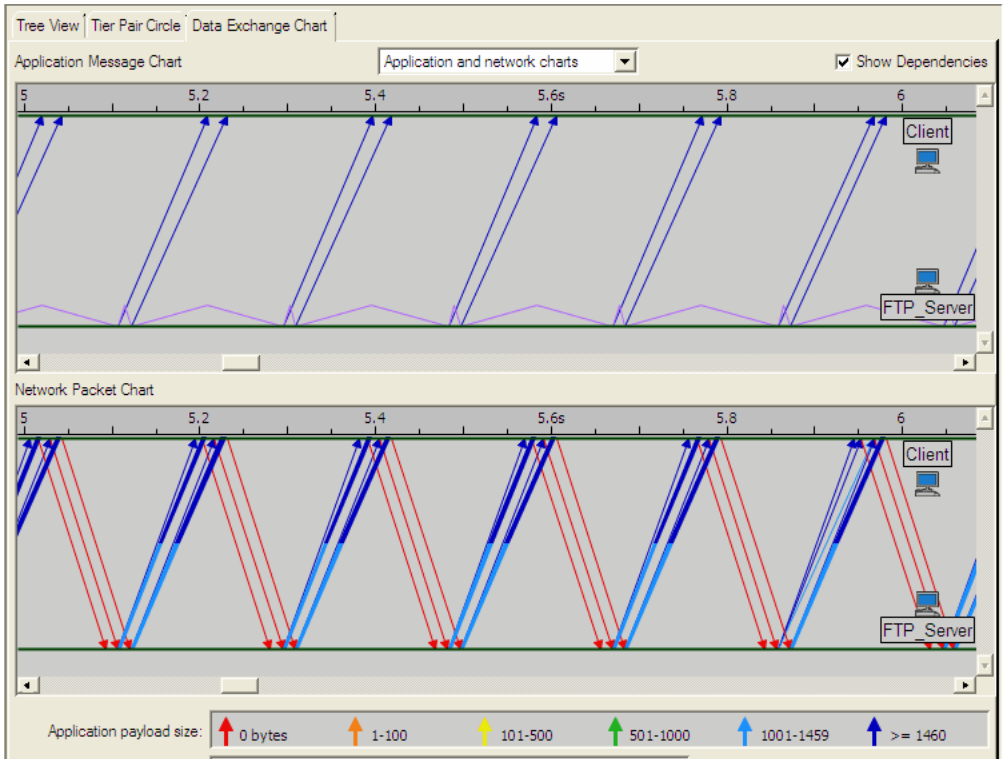
Figure 10-2 Data Exchange Chart (Application Chart with Dependencies)



The Data Exchange Chart shows the client initiating the file transfer and the server responding with several large (1,460 bytes or greater) application messages, as represented by the blue arrows between the Client and FTP_Server tiers. When we check the Show Dependencies box, AppTransaction Xpert shows that the delays are network-related, as indicated by the purple triangles on the FTP_Server tier.

Zooming in to a group of application messages and displaying both the Application Message Chart and the Network Packet Chart allows us to examine the relationship between the network packets and the application messages.

Figure 10-3 Zoomed View of Application and Network Charts

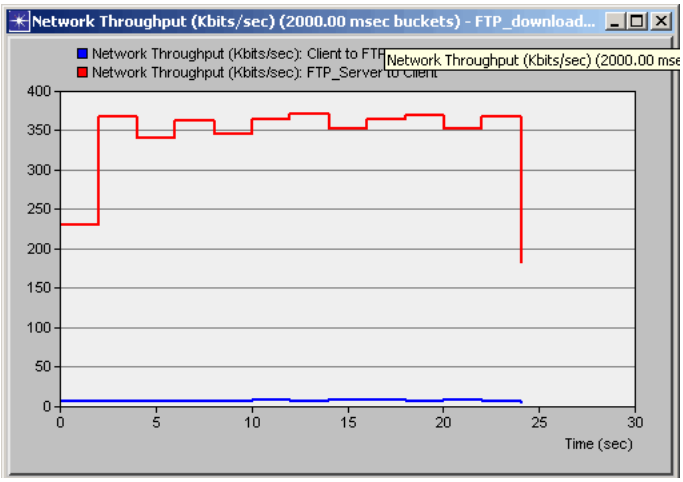


The Network Packet Chart displays the task as an exchange of packets as seen by the network. The Application Message Chart displays the task as an application-layer exchange of messages, excluding details from below the application layer.

We can see the FTP Server sends packets to the client and waits for the arrival of acknowledgements before sending additional application data. This is not optimal behavior because it lengthens the file transfer time.

From the Data Exchange Chart, we can graph specific statistics relating to the transaction that we are diagnosing. We know the delays are network-related, but we want to see how bandwidth is being used during this file transfer.

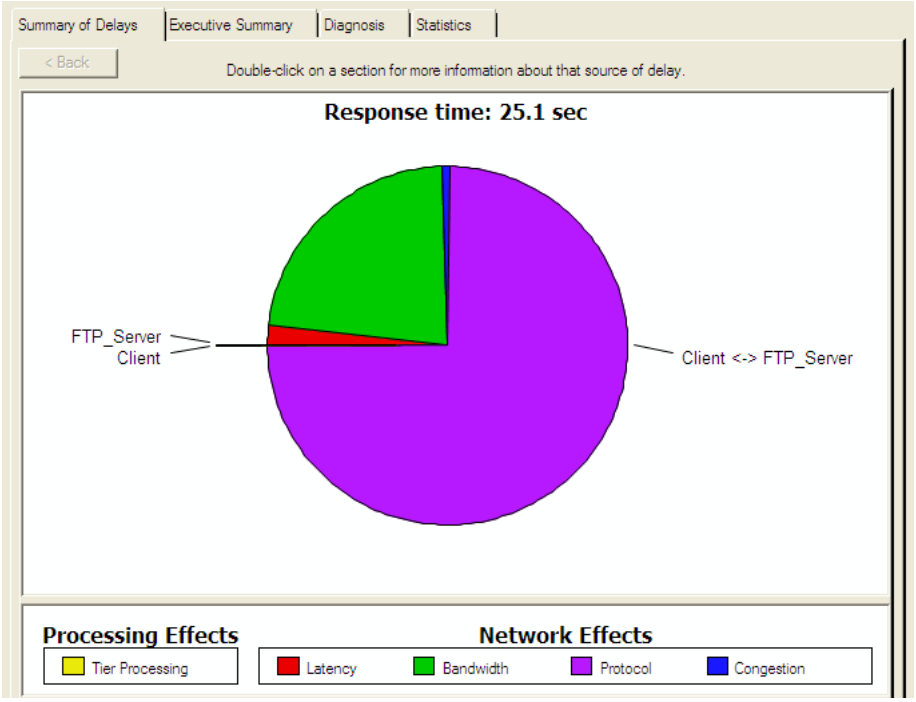
Figure 10-4 Time-Based Statistic of Network Throughput



The Network Throughput statistic above shows that the maximum download speed is about 350 Kbps, far less than the 1,544 Mbps the circuit provides.

AppDoctor's Summary of Delays is used to identify the factors contributing to the delay.

Figure 10-5 Summary of Delays for the File Download

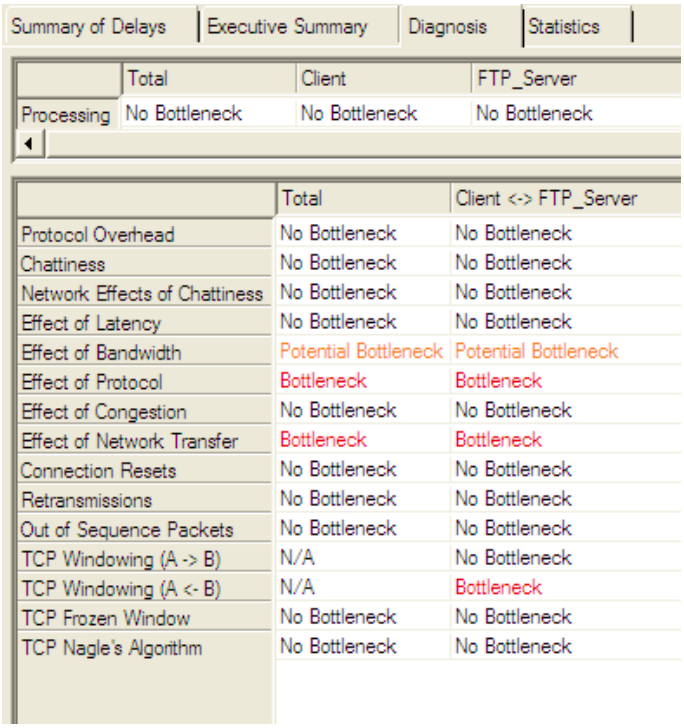


The chart identifies protocol delay as the primary source of delay in this file transfer. From this chart, we can also “drill down” into the protocol delay and view a graphical analysis of the different factors that might cause this particular delay.

Application delay (client/server processing time) is negligible for this transaction. In this case, the network is the cause of slow application response time.

The AppDoctor Diagnosis provides a more granular analysis of possible bottlenecks. This feature tests the current transaction against factors that often cause performance problems in network-based applications. Values that cross a specified threshold are marked as bottlenecks or potential bottlenecks.

Figure 10-6 Diagnosis for the File Download



	Total	Client	FTP_Server
Processing	No Bottleneck	No Bottleneck	No Bottleneck

	Total	Client <-> FTP_Server
Protocol Overhead	No Bottleneck	No Bottleneck
Chattiness	No Bottleneck	No Bottleneck
Network Effects of Chattiness	No Bottleneck	No Bottleneck
Effect of Latency	No Bottleneck	No Bottleneck
Effect of Bandwidth	Potential Bottleneck	Potential Bottleneck
Effect of Protocol	Bottleneck	Bottleneck
Effect of Congestion	No Bottleneck	No Bottleneck
Effect of Network Transfer	Bottleneck	Bottleneck
Connection Resets	No Bottleneck	No Bottleneck
Retransmissions	No Bottleneck	No Bottleneck
Out of Sequence Packets	No Bottleneck	No Bottleneck
TCP Windowing (A -> B)	N/A	No Bottleneck
TCP Windowing (A <- B)	N/A	Bottleneck
TCP Frozen Window	No Bottleneck	No Bottleneck
TCP Nagle's Algorithm	No Bottleneck	No Bottleneck

Diagnosis identifies three bottlenecks and one potential bottleneck. Each should be investigated. Descriptions of each are below.

Effect of Bandwidth

Packet transmission delay is a function of the size of the packet. Lower transmission speeds cause larger delays. This value is the total delay incurred due to transmission speed, as a percentage of the overall application response time.

Effect of Protocol

Network protocols (such as TCP) frequently perform flow control, congestion control, etc., which may throttle the rate at which applications send data. Other protocol effects that can impact application performance include retransmissions and collisions. This value is the total delay incurred due to protocol effects, represented as a percentage of the overall application response time.

Effect of Network Transfer

Network Transfer delay is a combination of Bandwidth, Protocol, and Congestion delays. If the bandwidths are not know/specified, then you don't know which combination of the three is the root cause. If the bandwidths are known/specified, AppTransaction Xpert still reports the network transfer delay, but it also specifies the breakdown between the three categories. (See AppDoctor's "Summary of Delays" and "Statistics" tabs.)

TCP Windowing

When an application is sending bulk data over a high-bandwidth and high-latency network, TCP window sizes must be large enough to permit TCP to send many packets in a row without having to wait for TCP acknowledgements.

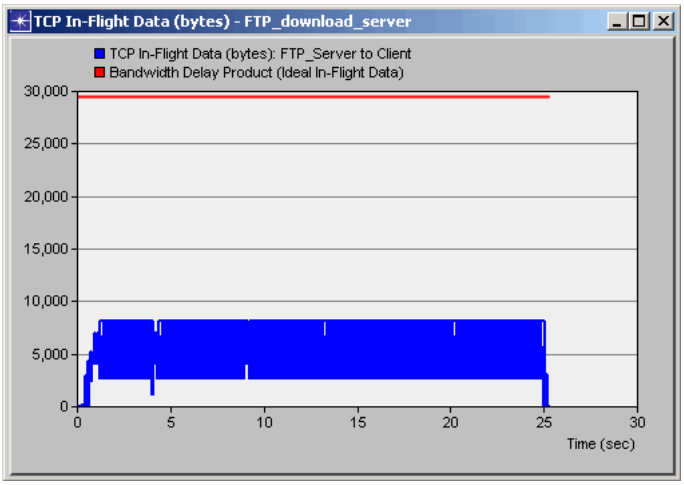
TCP will send data only if the amount of sent-but-not-yet-acknowledged data is less than the minimum of the congestion control window, sender window, and receiver window sizes.

- Sender and receiver windows have default sizes that can be overridden by an application.
- The congestion control window is dynamically-sized by TCP in response to retransmissions and other factors.
- TCP will be forced to wait for acknowledgments if any windows are less than the "bandwidth-delay product". The bandwidth-delay product is (2 x Bandwidth x Propagation Delay).

TCP is reducing the rate at which the application can send data because of a window that is too small. AppTransaction Xpert auto-calculates the optimum window size based on the "bandwidth-delay product" as 29,974 bytes.

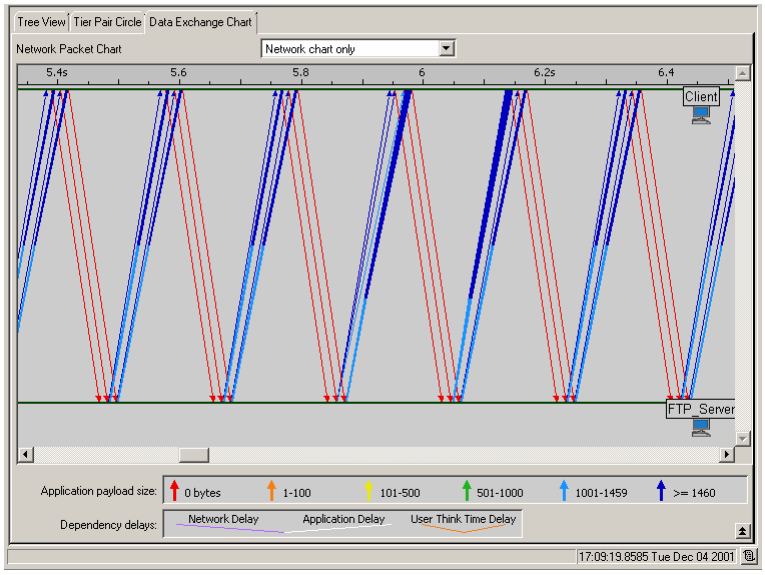
The "TCP In-Flight Data (bytes): FTP Server to Client" statistic illustrates this effect.

Figure 10-7 Time-based Statistic Graph of TCP In-Flight Data



The graph shows TCP is limiting the unacknowledged, in-flight data to about 8 KB although the ideal window size is about 30 KB.

Figure 10-8 Data Exchange Chart (Network View)



The data flow has reached the steady state equal to the TCP Window Size

AppTransaction Xpert diagnosed the bottleneck for this application transaction to be TCP windowing. The Data Exchange Chart above highlights this process. As an application transmits data, TCP increases the amount of allowable, unacknowledged data that can be sent, up to the TCP window size.

Because TCP normally increases throughput as the network proves it can handle the traffic, the beginning of the transmission is called “slow start.” The TCP window prevents application throughput from growing forever and saturating the network. The graph above shows that by the sixth exchange, the flow has reached steady state.

Conclusion

Analysis of this transaction revealed that the client had the send window size set to 8 KB. The window protects the network from being saturated by a single application, however in this example, TCP's flow control mechanism was the cause of the poor file download time. AppDoctor's Diagnosis suggested increasing the TCP window size to about 30 KB. It was not necessary to increase processing power at the client or server tiers, nor was it necessary to increase bandwidth. Any of these *solutions* would been both costly and ineffective.

AppTransaction Xpert allows you to troubleshoot poor application performance and pinpoint the potential causes.

Recreate the Example

Load the Transaction Analyzer model “FTP_download_over_WAN” (from *<reldir>*\sys\examples\AppTransaction Xpert\examples) or perform the following procedure.

Procedure 10-1 Recreating the Example

1. Open the following packet traces in AppTransaction Xpert:
   (**File** > **Open Packet Trace(s)** >
   In Transaction Analyzer (Simultaneous Captures)…)
   - FTP_download_server.enc
   - FTP_download_client.enc
2. Rename the tiers:
   - Rename 172.16.4.52 to Client
   - Rename 192.168.2.2 to FTP_Server
3. Set bandwidth and latency between tiers:
   (**AppDoctor** > **Refine Network Effects**…)
   - Set remote bandwidth to T1 (1.544 Mbps)
   - Latency should have been automatically detected as 76.2 ms
4. Open the Treeview window.

   Notice that there are two TCP connections in this trace:
   - Connection 1 carries the FTP control messages
   - Connection 2 carries the FTP data transfer

End of Procedure 10-1