Impact of Limited Bandwidth on a Web Application

*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 Examples. The screen images in this example were captured while running AppTransaction Xpert in Windows with AppTransaction Xpert Decode Module (ADM) installed. If you are working on Linux, or do not have ADM installed, some screens might look different.

In this network, the majority of remote users are connected to the web server, located in the data center, through a Frame Relay cloud using circuits with bandwidths of 128 to 512 Kbps. In test measurements of sample pages, download times averaged above 20 seconds.

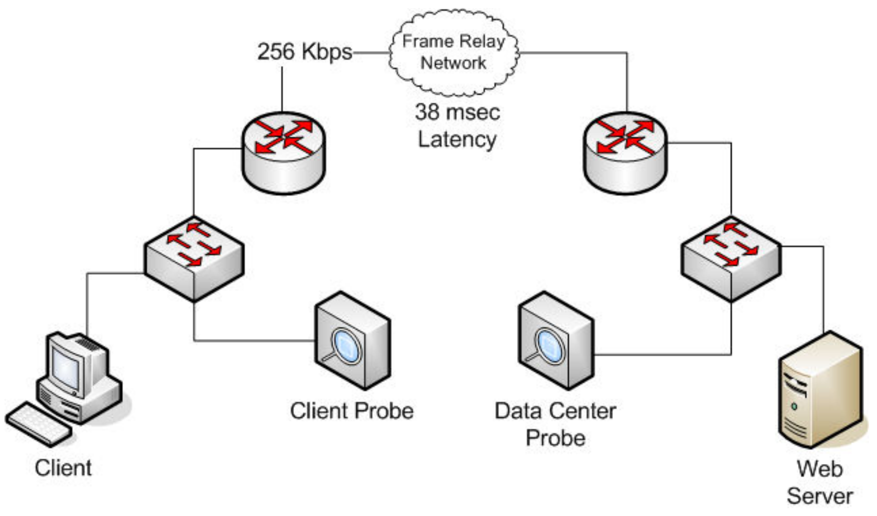
Possible causes of the poor download performance fall into two general categories:

- network bottlenecks
- server bottlenecks

The intent of this study is to identify the exact cause of the performance problems and to recommend fixes.

We began the investigation by identifying and capturing a typical transaction. Most remote offices connect through a 256 Kbps circuit, so we placed one probe on the client and another probe on the web server, as shown in the following network diagram.

Figure 6-1 Network Diagram



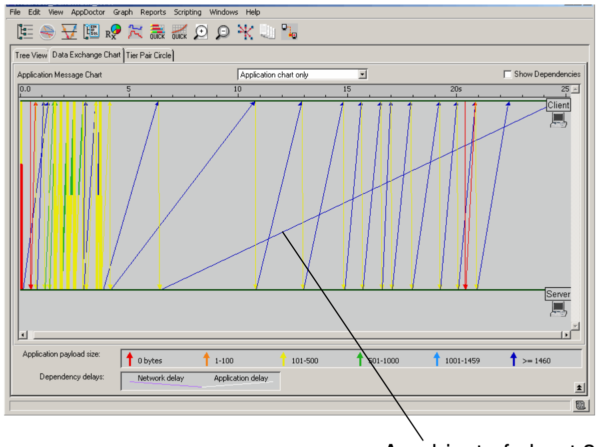
Packet trace captures were taken simultaneously at both the client and server, then merged and synchronized to obtain the best possible analysis of delays at each tier and the network.

Several sets of packet traces were captured and reviewed; most had response times of about 25 seconds, as measured with a stopwatch. For this study, we investigated a typical transaction—a web page download consisting of text and a mix of small and large images.

Diagnosis

After opening the packet traces in AppTransaction Xpert, we looked at the transaction in the Data Exchange Chart, which shows the flow of application messages between tiers over time.

Figure 6-2 Data Exchange Chart, WAN (256 Kbps)



An object of about 344 kilobytes took approximately 18 seconds to download

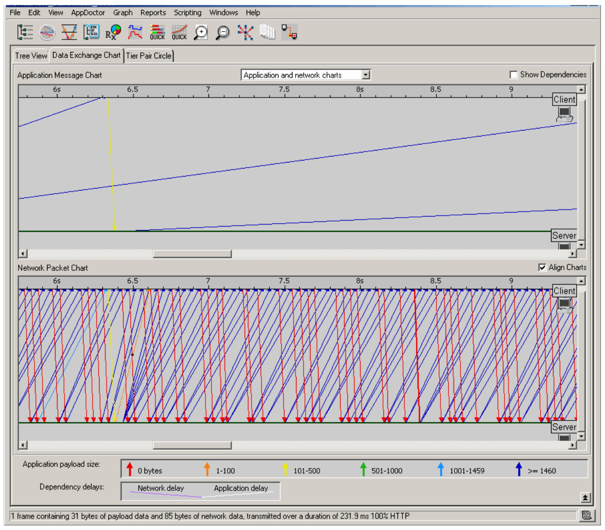
The Data Exchange Chart shows objects of various sizes being downloaded from the server to the client. One object appears to take much longer than the others.

From this object's tooltip, we can see the following information about the message:

- consists of over 200 packets
- has a throughput of 160 Kbps
- is about 344 KB
- takes more than 18 seconds to transmit

We zoomed into the first part of this message to see the object request and response exchange, and displayed both the Application Message Chart and the Network Packet Chart so we could examine the relationship between packets and application messages.

Figure 6-3 Zoomed View of the Request/Response Over the WAN (256 Kbps)

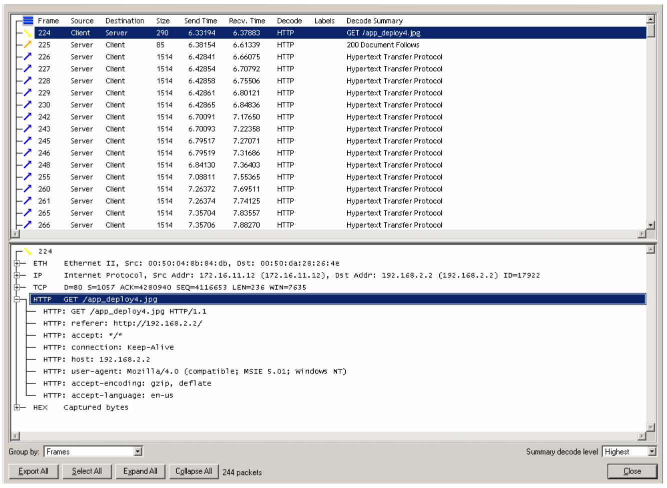


1 frame containing 31 bytes of payload data and 85 bytes of network data, transmitted over a duration of 231.9 ms 100% HTTP

The Data Exchange Chart shows that there is a repeating cycle of waiting for a dataless Client-to-Server ACK—zero-byte messages are red—before sending out one or two data packets from the client to the server. Due to this behavior, it appears that fewer than expected data packets were transferred in the ~3 second period shown above.

We used the Protocol Decode View to examine the contents of these messages to potentially gain more insight into the application.

Figure 6-4 Decodes of the Object



Frame	Source	Destination	Size	Send Time	Recv. Time	Decode	Labels	Decode Summary
224	Client	Server	290	6.33194	6.37883	HTTP		GET /app_deploy4.jpg
225	Server	Client	85	6.38154	6.61339	HTTP		200 Document Follows
226	Server	Client	1514	6.42841	6.66075	HTTP		Hypertext Transfer Protocol
227	Server	Client	1514	6.42854	6.70792	HTTP		Hypertext Transfer Protocol
228	Server	Client	1514	6.42858	6.75506	HTTP		Hypertext Transfer Protocol
229	Server	Client	1514	6.42861	6.80121	HTTP		Hypertext Transfer Protocol
230	Server	Client	1514	6.42865	6.84836	HTTP		Hypertext Transfer Protocol
242	Server	Client	1514	6.70091	7.17650	HTTP		Hypertext Transfer Protocol
243	Server	Client	1514	6.70093	7.22358	HTTP		Hypertext Transfer Protocol
245	Server	Client	1514	6.79517	7.27071	HTTP		Hypertext Transfer Protocol
246	Server	Client	1514	6.79519	7.31686	HTTP		Hypertext Transfer Protocol
248	Server	Client	1514	6.84130	7.36403	HTTP		Hypertext Transfer Protocol
255	Server	Client	1514	7.08811	7.55365	HTTP		Hypertext Transfer Protocol
260	Server	Client	1514	7.26372	7.69511	HTTP		Hypertext Transfer Protocol
261	Server	Client	1514	7.26374	7.74125	HTTP		Hypertext Transfer Protocol
265	Server	Client	1514	7.35704	7.83557	HTTP		Hypertext Transfer Protocol
266	Server	Client	1514	7.35706	7.88270	HTTP		Hypertext Transfer Protocol

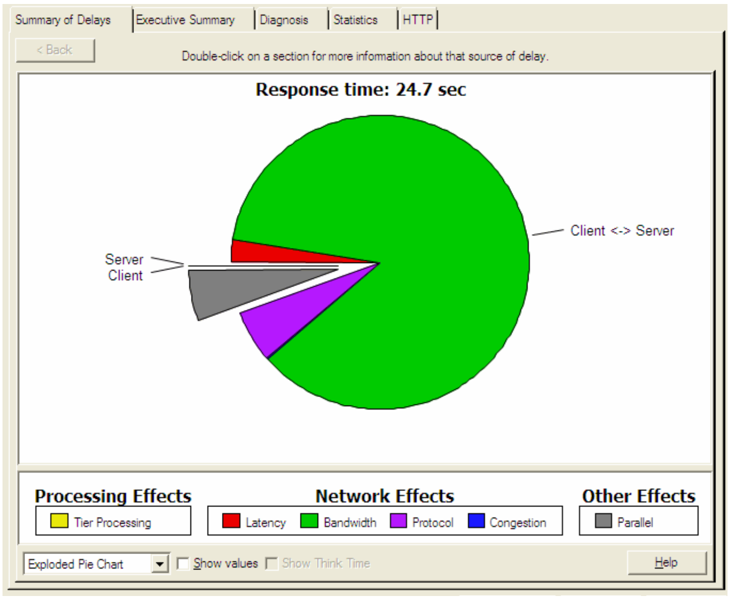
```
224
ETH    Ethernet II, Src: 00:50:04:8b:84:db, Dst: 00:50:da:28:26:4e
IP     Internet Protocol, Src Addr: 172.16.11.12 (172.16.11.12), Dst Addr: 192.168.2.2 (192.168.2.2) ID=17922
TCP    D=80 S=1057 ACK=4280940 SEQ=4116653 LEN=236 WIN=7635
HTTP   GET /app_deploy4.jpg
  HTTP: GET /app_deploy4.jpg HTTP/1.1
  HTTP: referer: http://192.168.2.2/
  HTTP: accept: */*
  HTTP: connection: Keep-Alive
  HTTP: host: 192.168.2.2
  HTTP: user-agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT)
  HTTP: accept-encoding: gzip, deflate
  HTTP: accept-language: en-us
HEX    Captured bytes
```

Group by: Frames
Summary decode level: Highest

The decode for Frame 224 shows the HTTP GET command for an object called “app_deploy4.jpg”. The server-to-client frames that follow represent the TCP segments containing the JPEG file (about 344 KB), that consists of 243 packets. (This information is known from the tooltip in the Data Exchange Chart.)

Examining the Data Exchange Chart and protocol decodes has helped us understand the flow of data among tiers and the content of the messages. Next, AppDoctor can provide a diagnosis of possible performance problems. Summary of Delays identifies the most important factors contributing to application delay.

Figure 6-5 AppDoctor Summary of Delays, WAN (256 Kbps)

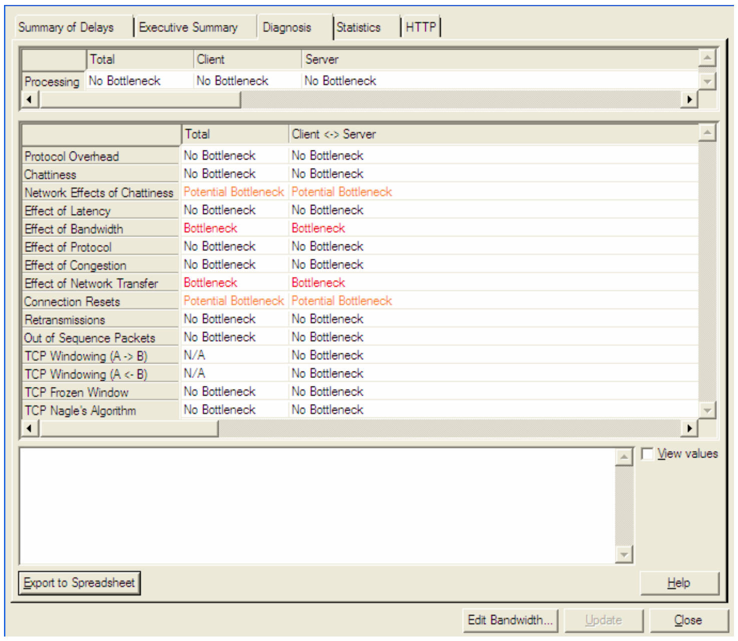


Summary of Delays identifies one main component of application response time: Bandwidth Delay.

Note—The Tier Processing is negligible for this transaction. In this case, the network is the cause of slow application response time.

AppDoctor’s Diagnosis provides a more granular view 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 6-6 AppDoctor Diagnosis, WAN (256 Kbps)



	Total	Client	Server
Processing	No Bottleneck	No Bottleneck	No Bottleneck

	Total	Client <-> Server
Protocol Overhead	No Bottleneck	No Bottleneck
Chattiness	No Bottleneck	No Bottleneck
Network Effects of Chattiness	Potential Bottleneck	Potential Bottleneck
Effect of Latency	No Bottleneck	No Bottleneck
Effect of Bandwidth	Bottleneck	Bottleneck
Effect of Protocol	No Bottleneck	No Bottleneck
Effect of Congestion	No Bottleneck	No Bottleneck
Effect of Network Transfer	Bottleneck	Bottleneck
Connection Resets	Potential Bottleneck	Potential 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	No Bottleneck
TCP Frozen Window	No Bottleneck	No Bottleneck
TCP Nagle's Algorithm	No Bottleneck	No Bottleneck

This diagnosis identifies one bottleneck (Effect of Bandwidth) and two potential bottlenecks (Connection Resets and Network Effects of Chattiness). Descriptions of each bottleneck follow.

Effect of Bandwidth

A packet's effect on bandwidth is a function of the size of the packet. Lower transmission speeds cause larger delays. Effect of Bandwidth is a percentage that is calculated by dividing the total delay incurred due to transmission speed by 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.

Network Effect of Chattiness

Network Effect of Chattiness is the total network delay incurred due to application turns, represented as a percentage of the total application response time. This application is incurring moderate network delays due to many application turns.

Connection Resets

Connection resets can occur for a variety of reasons, including:

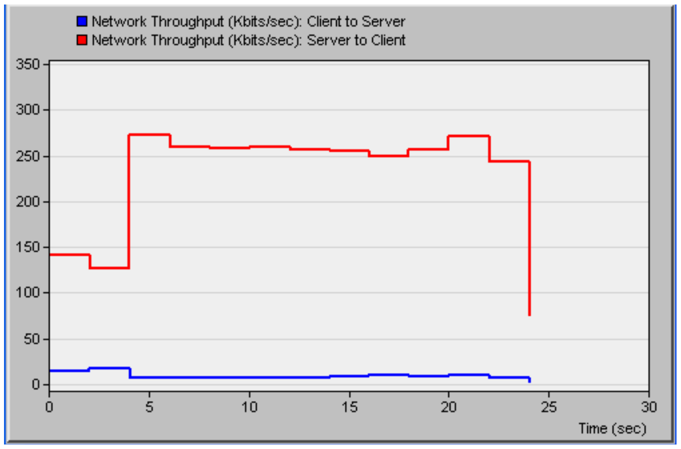
- An application is closing a connection.
- Delayed or duplicate connection control packets are received.
- An application is attempting to open a connection on a port where no application is listening. Note, however, that some protocols such as HTTP frequently have many connection resets, which is normal.

Diagnosis confirms the main bottlenecks for this web application are related to the network. A typical cause of poor application performance over WANs is limited bandwidth. The Summary of Delays confirms that bandwidth delay was responsible for a significant portion of the 25-second response time.

All indicators suggest throughput is limited by the amount of bandwidth between the two tiers.

One final confirmation of our hypothesis can be seen through the time-based statistic of network throughput.

Figure 6-7 Network Throughput of Web Transaction over WAN (256 Kbps)

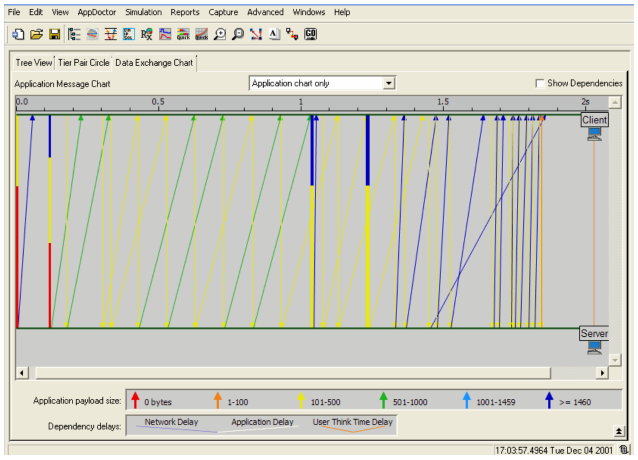


The application is sending data at an average of 256 Kbps, the maximum line rate of the Frame Relay circuit. A faster WAN link is required to accelerate this Web page download.

To test our hypothesis, we performed the same web page download in a lab where both client and server were placed on a 10 Mbps LAN. A protocol analyzer captured packets, and this trace was analyzed in AppTransaction Xpert. If this download is substantially faster, we have shown that the culprit is indeed the limited bandwidth WAN link.

In the Data Exchange Chart for the LAN example, shown below, we see that the application response time on the 10 Mbps LAN is about 2 seconds. Compare that to the 25 second application response time shown in the original 256 Kbps WAN trace.

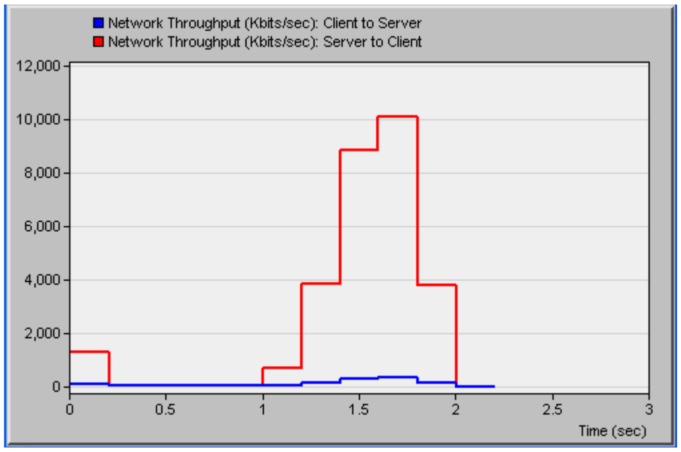
Figure 6-8 Data Exchange Chart of a Web Page Download Over a LAN (10 Mbps)



Conclusion

The network throughput graph for the LAN test shows the Web download is capable of filling a 10 Mbps link. This confirms the initial diagnosis that the application is bandwidth sensitive and increasing bandwidth on the network significantly improves application response time.

Figure 6-9 Network Throughput of Web Transaction Over LAN (10 Mbps)



Recreate the Examples

To recreate the WAN (256 Kbps), load the Transaction Analyzer model "low_bandwidth_HTTP" (in <*reldir*>\sys\examples\AppTransaction Xpert\examples) or perform the following procedure.

Procedure 6-1 Recreate the WAN (256 Kbps) Example

1 Open the following packet traces in AppTransaction Xpert:
(**File** > **Open Packet Trace(s)** >
In Transaction Analyzer (Simultaneous Captures)…)

- low_bandwidth_HTTP_server.enc
- low_bandwidth_HTTP_client.enc

2 Rename the tiers:

2.1 Rename 172.16.11.12 to Client

2.2 Rename 192.168.2.2 to Server

3 Set bandwidth and latency between locations:
(**AppDoctor** > **Refine Network Effects**…)

3.1 Set remote bandwidth to 256 Kbps

3.2 Latency should have been automatically detected as approximately 37.5 ms

4 When creating the Network Throughput chart, use a bucket width of 2000 ms.

End of Procedure 6-1

To recreate the LAN (10 Mbps) example, load the Transaction Analyzer model "high_bandwidth_HTTP" (in <*reldir*>\sys\examples\AppTransaction Xpert\examples) or perform the following procedure.

Procedure 6-2 Recreate the LAN (10 Mbps) Example

1 Open the following packet traces in AppTransaction Xpert:
(**File** > **Open Packet Trace(s)** >
In Transaction Analyzer (Simultaneous Captures)…)

- high_bandwidth_HTTP_server.enc
- high_bandwidth_HTTP_client.enc

2 Rename the tiers:

- Rename 172.16.4.52 to Client
- Rename 192.168.2.2 to Server

3 Set bandwidth and latency between locations:
(**AppDoctor** > **Refine Network Effects**…)

- Set remote bandwidth to 10,000 Kbps
- Latency should have been automatically detected as approximately 0.1 ms

4 When creating the Network Throughput graph, use a bucket width of 200 ms.

End of Procedure 6-2