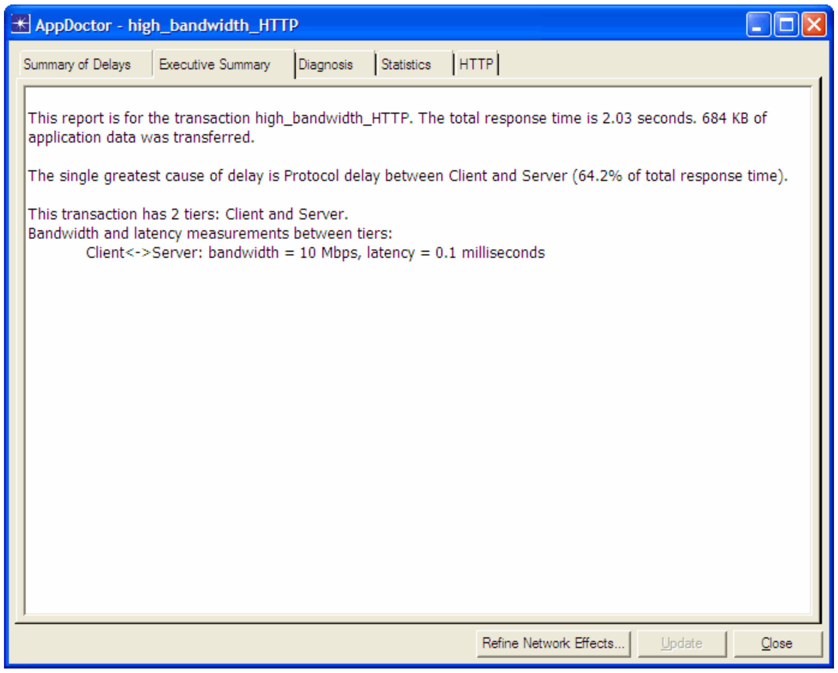
20 Diagnosing Applications with AppDoctor

Use AppDoctor to identify and diagnose bottlenecks, divide the total response time into separate components, and view detailed statistics of your network and application.

Figure 20-1 Executive Summary Tabbed Page in the AppDoctor Window



This report is for the transaction high_bandwidth_HTTP. The total response time is 2.03 seconds. 684 KB of application data was transferred.

The single greatest cause of delay is Protocol delay between Client and Server (64.2% of total response time).

This transaction has 2 tiers: Client and Server.
Bandwidth and latency measurements between tiers:
Client<->Server: bandwidth = 10 Mbps, latency = 0.1 milliseconds

To open AppDoctor, choose AppDoctor > <*appdoctor_page*>. The AppDoctor window has the following tabbed pages:

- Executive Summary page (Figure 20-1) summarizes the application task and identifies the primary cause of delay.
- AppDoctor Summary of Delays page divides the total application response time into separate components of network and application delay.
- AppDoctor Statistics page shows detailed statistics on network and application performance.
- AppDoctor Diagnosis page pinpoints and diagnoses bottlenecks and potential bottlenecks in your application.

Note—Depending on the type of application, the AppDoctor window might contain additional tabbed pages.

AppDoctor Summary of Delays

The AppDoctor Summary of Delays page divides the total response time into separate categories:

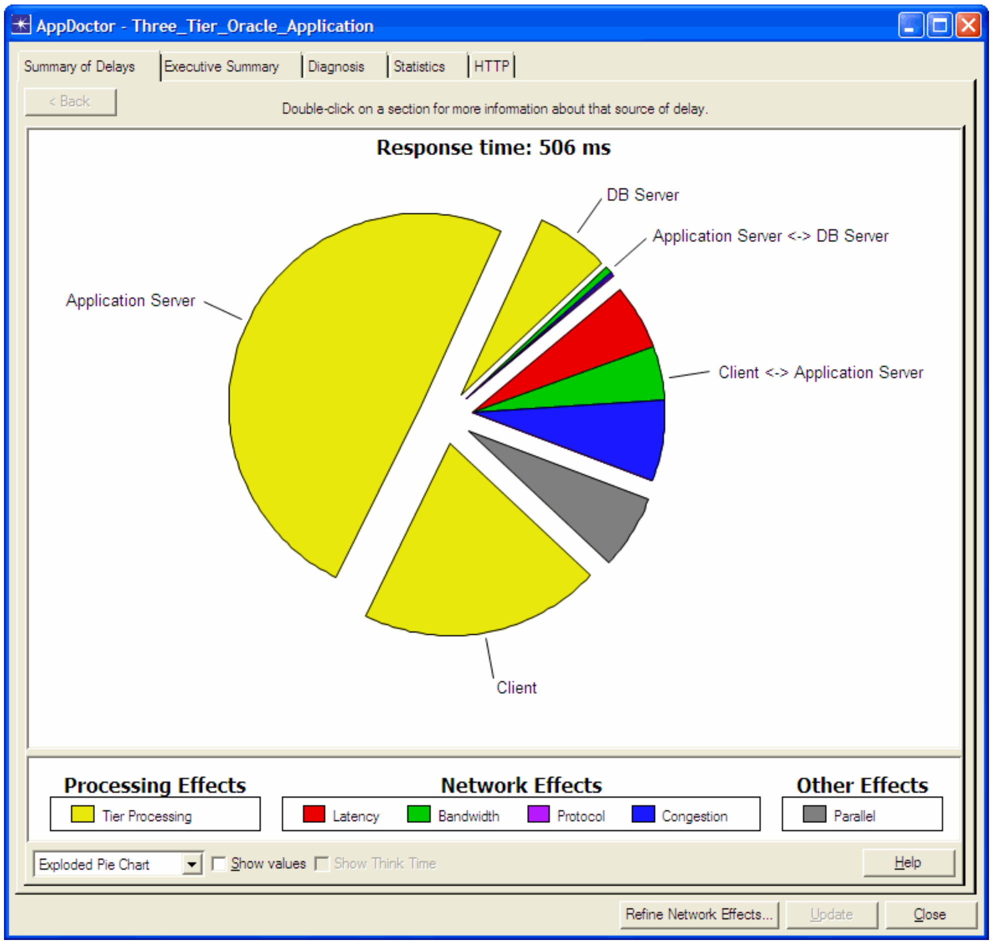
- Processing Effects—Application processing at each tier
- User Think Time——The total delay incurred whenever a client-side application waits for user input before it can proceed. If the application includes think time, this window shows the following:
  - — If Show Think Time is selected, the window shows the total duration (total time window of the application, from first message to last message, including think time)
  - — If Show Think Time is not selected, the window shows the response time (sum of all delays due to network, processing, and parallel effects; think time is not included)
- Network Effects—Latency, bandwidth, protocol, and congestion effects between each tier pair.

  Without bandwidth information, the bandwidth, protocol, and congestion delays are shown as a combined “Network Transfer” delay. To specify the bandwidth, click the “Refine Network Effects…” button.

  For more information, see Defining the Network Effects.
- Parallel Effects— If this category appears, it indicates one or both of the following:
  - — The application is performing multiple tasks simultaneously
  - — The Transaction Analyzer model contains extraneous traffic

This page highlights the relative amount of delay associated with each of these categories. In the following figure, processing delay at the Web Server tier accounts for 81.5% of the total response time.

Figure 20-2 Summary of Delays Tabbed Page in AppDoctor Window



You can set the View pull-down menu to see different types of summary charts. Check Show Values to see percentage values for each major component.

AppDoctor Drill-Downs: Examining the Cause of Delay

The Summary of Delays page enables you to drill-down into a delay category and view a graphical analysis of that specific type of delay. To drill-down, double-click on a category in the pie/bar chart.

You can view the following types of drill-down windows:

- Tier Processing drill-down—Lists all application dependencies, ranked by the amount of application processing delay in each dependency. To zoom in on a dependency in the Data Exchange Chart, double-click on the dependency in this drilldown window.
- User Think Time drill-down—Shows the total User Think Time delay for the application, and lists all individual dependencies that include think time (from the most think time to the least).
- Latency drill-down—Shows a graph of the Application Turns (sec) statistic. An application with frequent application turns can result in large latency delays. To identify “chatty” intervals, click the “Show in DEC” button in this drilldown window.
- Bandwidth drill-down—Shows a graph of the Network Throughput (Kbits/sec) statistic and the link bandwidth. If the network throughput exceeds the available bandwidth, the result is bandwidth delay. To identify intervals when this occurs, click the “Show in DEC” button in the drilldown window.
- Protocol drill-down—Shows graphs of factors that seem to cause protocol delay in the application. The available statistics are:
  - — TCP In-Flight Data (bytes)
  - — Retransmissions (packets)
  - — Out of Sequence Packets
  - — TCP Advertised Receive Window: <tier_A> to <tier_B>
  - — TCP Nagle’s Delay (sec)
  - — Packet Congestion
  - — Network Throughput (Kbits/sec)

  Because many factors can affect this type of delay, the most relevant factors are shown, by default, based on the current file.

- Congestion drill-down—Shows graphs of factors that seem to cause congestion delay in the application. The Congestion drilldown has the same list of available statistics as the Protocol drilldown (see previous bullet point), but highlights congestion-related statistics by default.

  Because many factors can affect this type of delay, the most relevant factors are shown, by default, based on the current file.

- Network Transfer drill-down—Shows the most significant information for troubleshooting network transfer problems.

  Network Transfer delay is a combination of Bandwidth, Protocol, and Congestion delays. To determine the individual components of Network Transfer delay, click the “Refine Network Effects…” button. For more information, see Defining the Network Effects.

- Parallel Effects drill-down—If AppDoctor assigns a portion of delay time to the Parallel Effects category, this drilldown shows how the combination of individual effects contribute to the total response time. For more information, see Viewing the Individual Components of Parallel Effects.

Related Topics

- *Diagnosing Applications with AppDoctor*

AppDoctor Statistics

The Statistics tabbed page lists various application- and network-related statistics. To export the contents of this page to a tab-delineated text file, click the “Export to Spreadsheet” button.

The following tables list the tier statistics and the tier-pair statistics.

Table 20-1 AppDoctor Tier Statistics

Statistic	Description
User Think Time (sec)	Total delay due to an application at the client tier waiting for user input (for example, if the user needs to confirm a request and click Submit before the request is sent to the server). For more information, see User Think Time—.
Effect of Network (sec)	Total delay caused by network-related factors: transmission delays, propagation delays, protocol delays, etc.
Effect of Processing (sec)	Total delay caused application processing at a one tier or at all tiers. This is the sum of the “Application Delay” portion of all dependencies in the application task.
Parallel Effects (sec)	Total delay occurred when tier processing, bandwidth, and/or latency delays occur simultaneously. For more information, see Parallel Effects.

Table 20-2 AppDoctor Tier-Pair Statistics

Statistic	Description
Application Data (bytes)	Total number of application bytes transmitted
Application Messages	Total number of application messages
Application Turns	Total number of application turns; an application turn occurs when the flow of messages changes direction (for example, when a TierA —> Tier B message is followed by a TierB —> TierA message)
Application Turns (sec)	Frequency of application turns per second
Average Application Message (bytes)	Average size (in bytes) of each application message
Average Network Packet Payload (bytes)	Average number of total bytes in each network packet
Bandwidth (Kbps)	Bandwidth on the link connecting two tiers
Connection Resets	Total number of transport-connection resets
Effect of Bandwidth (sec)	Delay due to transmission speed
Effect of Latency (sec)	Total delay due to propagation

Table 20-2 AppDoctor Tier-Pair Statistics (Continued)

Statistic	Description
Effect of Network Transfer (sec)	Total delay caused by the following network-related factors: bandwidth, protocol, and congestion. To determine the individual components of delay, click the “Refine Network Effects…” button.
Effect of Processing (sec)	Total delay due to application processing at a given tier or at all tiers. This is the sum of the “Application Delay” part of all dependencies in the application task.
Effect of Protocol (sec)	The total amount of queueing and other protocol-related delays between the tiers.
Effect of Congestion (sec)	The total amount of delay due to slow links, network-induced queueing, and other congestion-related effects between tiers.
Latency (ms)	Propagation time for one packet transmitted between two tiers (per-tier-pair only)
Max Application Bytes Per Turn	Maximum number of bytes transmitted from source to destination tier in a single application turn
Max Unacknowledged Data (Bytes)	Maximum number of bytes transmitted (but unacknowledged) from source to destination tier in a single application turn
Network Data (bytes)	Total number of bytes sent at all network layers
Network Packets	Total number of network packets
Out of Sequence Packets	Total number of out-of-sequence packets
Response Time (sec)	Application response time, as seen from the initiating (client) tier. This is measured from the time it sends the first application-payload packet to the time it receives the last application-payload packet. This statistic parallels the “AppTransaction Xpert. Task Response Time (sec)” statistic found in the Project Editor’s Choose Results dialog box. This response time is the primary way to measure your application’s performance when simulating “what-if?” scenarios.
Retransmissions	Total number of packet retransmissions
TCP Frozen Window (sec)	The amount of time that the TCP receive window is “frozen” (i.e., less than the MSS). The sender cannot send any data until the window becomes unfrozen.
TCP Nagle’s Algorithm (sec)	The amount of delay incurred due to Nagle's algorithm, which is a sender-side algorithm that reduces the number of small packets sent out.
TCP Triple-Duplicate loss indications	Number of triple-duplicate ACK messages. A triple-duplicate message occurs when the source tier receives four ACK messages for the same packet: the original message plus three duplicates. This indicates that the destination tier did not receive a data packet. In response, the source tier re-sends this packet.

Related Topics

- *Diagnosing Applications with AppDoctor*

AppDoctor Diagnosis

The Diagnosis tabbed page highlights bottlenecks and potential bottlenecks in the application and the network. This page diagnoses the entire application (Total column) and each tier pair. For detailed descriptions of each category, see AppDoctor Diagnosis Categories.

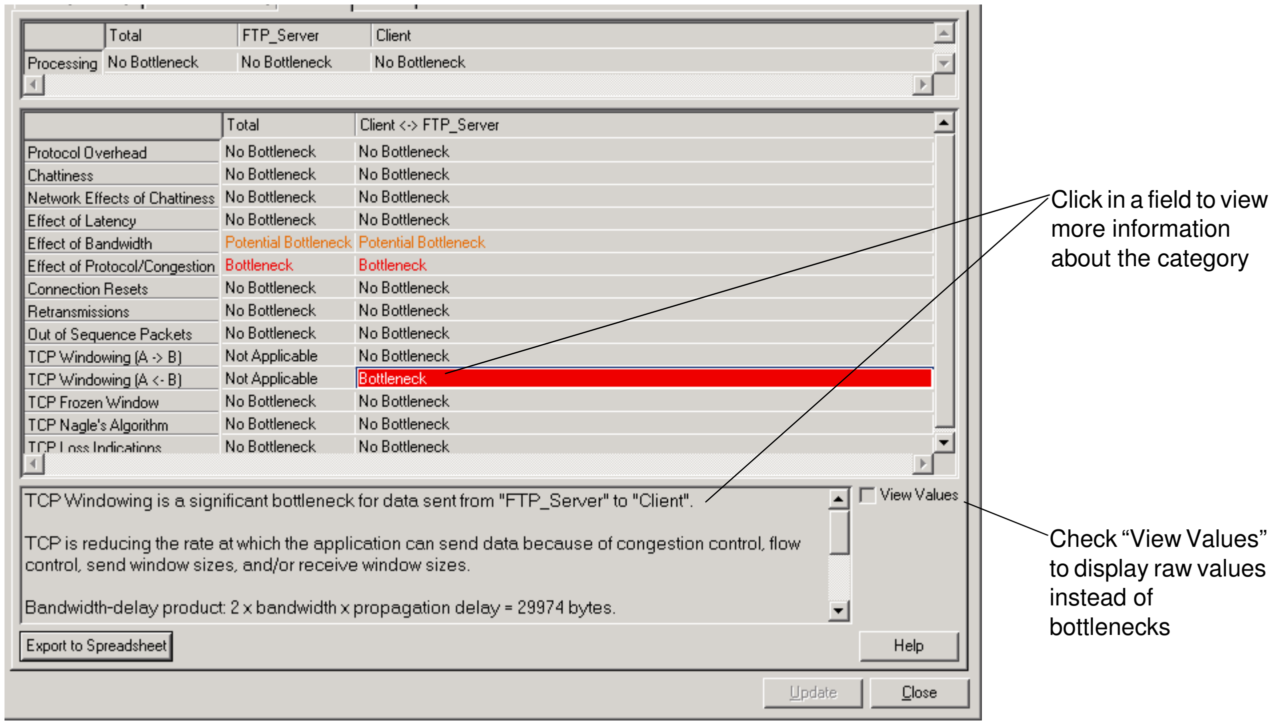
AppDoctor uses specific thresholds to determine bottlenecks and potential bottlenecks. To change the default threshold levels, choose Edit > Preferences to open the Edit Preferences dialog box; then search on the string “app”. AppTransaction Xpert has threshold preferences for each AppDoctor diagnosis category.

For most diagnosis categories, AppDoctor reports a bottleneck if the calculated value is above the threshold. The two exceptions are Chattiness Bottleneck and TCP Windowing Bottleneck; for these categories, AppDoctor reports a bottleneck if the calculated value is below the threshold.

AppDoctor reports a potential bottleneck if the calculated value is close to the threshold, but does not exceed it. AppDoctor never reports a full bottleneck for the Connection Resets Potential Bottleneck, because some applications (such as HTTP) use resets in their normal operations.

To export the contents of the AppDoctor Diagnosis page to a tab-delineated text file, click the “Export to Spreadsheet” button. Note that you can export bottleneck diagnoses or raw values, depending on the “View Values” setting.

Figure 20-3 Diagnosis Tabbed Page in AppDoctor Window


View Values
Export to Spreadsheet
Help
Update
Close
Click in a field to view more information about the category
Check “View Values” to display raw values instead of bottlenecks

	Total	FTP_Server	Client
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/Congestion	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)	Not Applicable	No Bottleneck
TCP Windowing (A <- B)	Not Applicable	Bottleneck
TCP Frozen Window	No Bottleneck	No Bottleneck
TCP Nagle's Algorithm	No Bottleneck	No Bottleneck
TCP Loss Indications	No Bottleneck	No Bottleneck

TCP Windowing is a significant bottleneck for data sent from "FTP_Server" to "Client".

TCP is reducing the rate at which the application can send data because of congestion control, flow control, send window sizes, and/or receive window sizes.

Bandwidth-delay product: 2 x bandwidth x propagation delay = 29974 bytes.

Related Topics

- *Diagnosing Applications with AppDoctor*

AppDoctor Diagnosis Categories

The Diagnosis page identifies bottlenecks (red text) and potential bottlenecks (yellow text). The following table lists the available diagnosis categories.

Table 20-3 AppDoctor Diagnosis Categories

Bottleneck	Description
Tier Processing Bottleneck	Processing on this tier is contributing significantly to the application response time.
Protocol Overhead Bottleneck	Large amounts of protocol overhead are increasing the utilization of your network.
Chattiness Bottleneck	The application is sending many small requests and responses. Many small requests and responses make inefficient use of tier and network resources.
Network Effects of Chattiness Bottleneck	The application is incurring significant *networking* delays due to many application turns.
Effect of Latency Bottleneck	The application is experiencing a significant bottleneck due to the time it takes packets to propagate across the network.
Effect of Bandwidth Bottleneck	The application is experiencing a significant bottleneck due to the transmission speed.
Effect of Protocol Bottleneck	The application is experiencing a significant bottleneck due to protocol effects.
Effect of Congestion Bottleneck	The application is experiencing a significant bottleneck due to congestion effects.
Effect of Network Transfer	The application is experiencing a significant bottleneck due to network transfer effects. Specify the bandwidth to diagnose the delays of the individual components (bandwidth, protocol, and congestion).
Connection Resets Potential Bottleneck	The application is experiencing an excessive number of connection resets. AppDoctor never reports a full bottleneck for this category, because some applications (such as HTTP) use resets in their normal operations.
Out of Sequence Packets Bottleneck	The Out of Sequence Packets bottleneck is similar to the Retransmissions bottleneck in that both bottlenecks indicate the same underlying condition: that the transport protocol (such as TCP) is retransmitting packets at the source tier. These two bottlenecks differ in that AppTransaction Xpert usually detects retransmissions at the source tier, while it detects out-of-sequence packets at the destination tier. The statistic value represents the percentage of packets received at the destination tier that were out of sequence.

Table 20-3 AppDoctor Diagnosis Categories (Continued)

Bottleneck	Description
Retransmissions Bottleneck	Retransmissions are significantly affecting application response times.
TCP Windowing Bottleneck	TCP Windowing is a significant bottleneck for data sent between host A and host B. TCP is reducing the rate at which the application can send data because of congestion control, flow control, send window sizes, and/or receive window sizes.
TCP Frozen Window	The advertised TCP Receive Window has dropped to a value smaller than the Maximum Segment Size (MSS). This is affecting your application response time.
TCP Nagle’s Algorithm	Nagle's algorithm is present and is slowing application response times.

For general information about the Diagnosis page, see AppDoctor Diagnosis.

Tier Processing Bottleneck

Processing on this tier is contributing significantly to the application response time.

Explanation

Processing delay is due to file I/O, CPU processing, memory access, and other tier-specific processes. The numeric value represents tier processing delay as a percentage of the total application response time.

Suggestions

1) Increase the processing speed and capabilities of the tier:
   - a) Increase physical memory, if a tier frequently uses virtual memory (pages).
   - b) Increase CPU speed and/or number, if a tier frequently runs at 100% utilization of its CPUs.
   - c) Add faster disks, if a tier frequently accesses disks.
   - — Many small read/writes will benefit from faster disk seek times.
   - — Many large read/writes will benefit from faster disk throughput.

2) Improve the processing efficiency of the application programs.
   - a) For database applications, possible solutions include:
   - — Index fields that are frequently queried.
   - — Redesign database queries to reduce database load.
   - — Do not send records across the network one at a time, when a set is transferred.
   - b) For all applications:
   - — Reduce the number of allowed simultaneous connections to limit the load on the tier.
   - — Profile the program to determine execution bottlenecks.

3) Reduce the load on this tier by sharing its work with additional computers.

Protocol Overhead Bottleneck

Large amounts of protocol overhead are increasing the utilization of your network.

Explanation Protocol headers add overhead to each application message. Protocols also send packets that have no application data (such as TCP acknowledgement packets). This overhead can introduce delays by increasing congestion in the network; these delays can be especially significant if you are sending a large number of small application messages.

The numeric value represents the percentage of total data that is protocol overhead.

Suggestions Make the application send fewer, larger application messages. This will utilize network resources more efficiently.

Chattiness Bottleneck

The application is sending many small requests and responses. Many small requests and responses make inefficient use of tier and network resources.

Explanation The data sent per application turn is small. This may cause significant network delay. Additionally, a significant portion of application processing time can be spent processing requests and responses.

If you have a “Chattiness” bottleneck without a “Network Cost of Chattiness” bottleneck, this means that

- The application is not incurring significant network delays due to chattiness;
- The application might be incurring significant processing delays due to overhead associated with handling many small application requests/responses;
- The application's “Network Cost of Chattiness” could dramatically increase in a higher-latency network.

The numeric value is the number of application bytes per application turn.

Suggestions Make your application send fewer, larger application messages. This will utilize network and tier resources more efficiently. For example, a database application should not send groups of records across the network one record at a time.

Network Effects of Chattiness Bottleneck

The application is incurring significant *networking* delays due to many application turns.

Explanation Each time the conversation changes direction (an "application turn"), the packets incur a network delay while traversing the network. Mildly chatty applications suffer over high-latency networks (like WANs), while very chatty applications can suffer even over low-latency networks (like LANs).

Interactive applications (such as telnet) tend to be chattier than non-interactive applications.

The numeric value is the total network delay incurred as a result of application turns, represented as a percentage of the overall application response time.

Suggestions Consider the following:

1) Make your application send fewer, larger application messages, which utilizes the network and tier resources more efficiently. For example, a database application should not sent groups of records across the network one record at a time.

2) Consider decreasing the network latency between chatty tiers:

   a) If the application has a "Transmission Delay" bottleneck between the chatty tiers, increasing the transmission speed between those tiers might help.

   b) If the application has a "Propagation Delay" bottleneck between the chatty tiers, decreasing the propagation delay between those tiers might help.

Effect of Latency Bottleneck

The application is experiencing a significant bottleneck due to the time it takes packets to propagate across the network.

Explanation Each time the application conversation changes direction (an "application turn"), the application waits for packets to propagate across the network. Propagation delay is typically a function of the speed of light and the distance traveled. Devices latencies also add to propagation delays.

The numeric value is the total delay incurred due to propagation, represented as a percentage of the overall application response time.

Suggestions Consider the following solutions:

1) Move the affected tiers physically closer together.
2) If the application has a "Chattiness" bottleneck, addressing the application's chattiness might significantly improve application response time.

Effect of Bandwidth Bottleneck

The application is experiencing a significant bottleneck due to the transmission speed.

Explanation A packet's transmission delay is a function of the size of the packet. Lower transmission speeds cause larger delays.

The numeric value represents the total delay incurred due to transmission speed, as a percentage of the overall application response time.

Suggestions Consider the following solutions:

1) Use faster links.
2) Send less data.

Effect of Protocol Bottleneck

The application is experiencing a significant bottleneck due to protocol effects.

Explanation Network protocols (such as TCP) frequently perform flow control, congestion control, and other effects that can throttle the rate at which applications send data. Other protocol effects that can impact application performance include retransmissions and collisions.

The numeric value is the total delay incurred due to protocol effects, represented as a percentage of the overall application response time.

Suggestions Consider the following solutions:

1) Check for evidence of problems such as collisions at the MAC layer or retransmissions at the transport layer.
2) If the application is using TCP:
   a) Check to see if the application is using Nagle's Algorithm.
   — Applications can disable Nagle's Algorithm with a system call such as "setsockopt(`..., TCP_NODELAY,...)`"
   — Nagle's Algorithm can frequently be disabled for an entire tier.
   — Check to see if "TCP Windowing" is reported as a bottleneck.

Effect of Congestion Bottleneck

The application is experiencing a significant bottleneck due to congestion effects.

Explanation If high amounts of network traffic result in links that are heavily utilized (regardless of the available bandwidth on the link), the network will induce a variable queuing delay. This delay might throttle the rate at which applications send data. Other congestion effects that can affect application performance include retransmissions and collisions.

The numeric value is the total delay incurred due to congestion effects, represented as a percentage of the overall application response time.

Suggestions Consider the following solutions:

1) Use faster links.
2) Send less data.
3) Reschedule the application to occur off-hours, or when there is less traffic.

Effect of Network Transfer

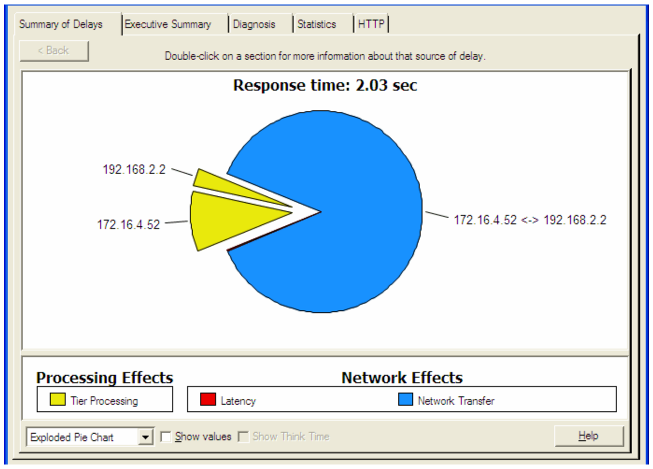
The application is experiencing a significant bottleneck due to network transfer delays.

Explanation Before bandwidth is specified, the bandwidth, protocol, and congestion components are combined and reported as Network Transfer effects.

For example, the following figure shows an AppDoctor Summary of Delays chart before and after the bandwidth is specified. Before the bandwidth is specified, the results indicate that Network Transfer is the major portion of delay. After specifying the bandwidth, the AppDoctor results update automatically with more detailed information.

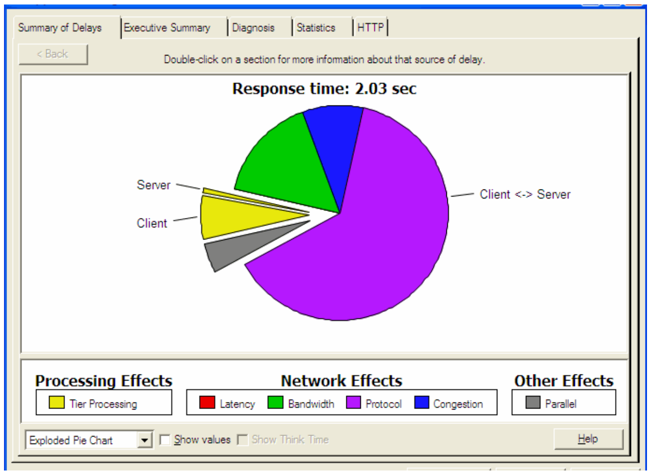
Figure 20-4 Example: Defining Bandwidth for Detailed Network Analysis Results

Before the bandwidth is specified…



Click the "Refine Network Effects… button to define bandwidth (local and remote). You can also define capture locations, tier locations, and latencies.

After the bandwidth is specified…



After defining the bandwidth, the AppDoctor results display detailed results of the analysis.

Suggestion Specify the bandwidth to diagnose the effect of the individual components (bandwidth, protocol, and congestion.)

To edit the bandwidth, click the "Refine Network Effects…" button.

For more information, see Defining the Network Effects.

Connection Resets Potential Bottleneck

The application is experiencing an excessive number of connection resets. AppDoctor never reports a full bottleneck for this category, because some applications (such as HTTP) use resets in their normal operations.

Explanation

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

1) An application is closing a connection.
2) Delayed or duplicate connection control packets are received.
3) An application is trying 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.

The numeric value is the total number of connection resets.

Suggestions

Repeat the experiment wherein you captured application traffic. Import the new packet trace into AppTransaction Xpert to determine if connection resets are typical.

Out of Sequence Packets Bottleneck

The Out of Sequence Packets bottleneck is similar to the Retransmissions bottleneck, in that both bottlenecks indicate the same underlying condition: that the transport protocol (such as TCP) is retransmitting packets at the source tier.

These two bottlenecks differ in that AppTransaction Xpert usually detects retransmissions at the source, while it detects out-of-sequence packets at the destination tier. Thus if you captured traffic at both the source and the destination tiers, the statistics for these two bottlenecks should be about equal. If you captured at the source or destination tier only, you might see a discrepancy between these bottlenecks and their resulting statistics.

The statistic value represents the percentage of packets received at the destination tier that were out of sequence.

For more information about retransmissions, see Retransmissions Bottleneck.

Retransmissions Bottleneck

Retransmissions are significantly affecting application response times.

Explanation

Transport protocols such as TCP will retransmit packets if packets are lost or excessively delayed. This leads to longer application response times because:

1) Data must be transmitted more than once.

2) When TCP observes packet loss, it infers that the network is congested. This causes TCP to reduce the rate at which applications can send traffic. Retransmissions increase the likelihood of “TCP Windowing” bottlenecks because retransmissions cause TCP to shrink the Congestion Control Window.

   Lossy channels, such as Frame Relay or ATM, can allow applications to *burst* above sustainable data transmission rates. These bursts allow greater data rates, but packets within a burst are more likely to be dropped.

The numeric value represents the percentage of packets that were retransmitted.

Suggestions

Consider the following solutions:

1) To determine whether retransmissions are typical, capture and re-import another packet trace of the application traffic.

2) Reduce the likelihood of congestion in the network by increasing network capacity. If possible, you might want to determine where packet losses are occurring.

3) Enable TCP extensions (such as Selective Acknowledgements and Fast Retransmit/Fast Recovery) that increase the ability of TCP to cope with packet loss and retransmissions. The method of enabling these TCP extensions varies by operating system, and you might need to upgrade your operating system version.

4) If you are using Frame Relay and/or ATM, decrease the “lossiness” by enabling traffic shaping. Traffic shaping keeps data transmission rates within the sustainable level, which means packets are less likely to be dropped. Contact your service provider and/or vendor for more information.

TCP Windowing Bottleneck

TCP Windowing is a significant bottleneck for data sent between host A and host B. TCP is reducing the rate at which the application can send data because of congestion control, flow control, send window sizes, and/or receive window sizes.

Explanation

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 ACKs. TCP will only send data 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.

- The sender and receiver windows have default sizes, which 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`**.
- For more information, see *TCP/IP Illustrated, Vol. 1: The Protocols by W. Richard Stevens.*

The numeric value is the amount of bandwidth-delay product used by the TCP connection.

Suggestions

Make sure the application is using windows which are larger than the bandwidth-delay product. The maximum amount of sent-but-not-yet-acknowledged data from the source to the destination host appears as the AppDoctor statistic “Max Unacknowledged Data Sent”. It is likely that one of the windows is equal to the AppDoctor statistic “Max Unacknowledged Data Sent”.

1) Check the receive window size for the destination host.
   a) This is visible in the TCP protocol decode as WIN=XXXX.
   b) If this is approximately equal to “Max Unacknowledged Data Sent”, then the receive window is likely the bottleneck.
   — Applications can set the receive window size with a system call such as "`setsockopt (..., SO_RCVBUF,...)`"
   — The default receive window size can frequently be set for an entire tier.

2) If there are many TCP retransmissions, then the congestion control window might be the bottleneck. You can decrease the negative effects of retransmissions by enabling TCP extensions such as Fast Retransmit/Fast Recovery and Selective Acknowledgments. The method of enabling these TCP extensions varies according to your operating system, and you might need to upgrade your operating system version.

3) If receive window and retransmissions do not seem to be the bottleneck, try increasing the send window size for the source host.

   a) Applications can set the send window size with a system call such as "`setsockopt (..., SO_RCVBUF,...)`"

   b) The default receive window size can frequently be set for an entire tier.

TCP Frozen Window

The advertised TCP Receive Window has dropped to a value smaller than the Maximum Segment Size (MSS). This is affecting the application response time.

Explanation

The advertised TCP Receive Window has dropped to a value smaller than the MSS. When this occurs, the sender cannot send any data until the receive window is one MSS or larger.

To determine if the receive window has become larger, the sending side periodically sends one-byte probe packets. The contents of these probe packets depends on the particular implementation, but they are usually sent with an exponential backoff.

The usual case of a TCP frozen window is that the application on the receiving side is not taking data from the TCP receive buffer quickly enough.

Suggestions

Consider the following solutions:

1) Send less data.

2) Have the receiving application retrieve the data more quickly; if the application cannot process all the data at once, consider storing the data in another buffer.

3) Upgrade the receiving computer.

TCP Nagle’s Algorithm

Nagle's algorithm is present and is slowing application response times.

Explanation Nagle's algorithm is a sending-side algorithm that reduces the number of small packets on the network, thereby increasing router efficiency. Nagle's algorithm is causing excessive numbers of delayed ACKs and is slowing down the application.

Suggestions Consider the following solutions:

1) Disable Nagle's algorithm for this application.

2) Rewrite the application such that it sends fewer, larger packets, or does not encounter a TCP delayed ACK.

3) Configure TCP on the receiving host so that TCP acknowledges every packet it receives.

Related Topics

- *AppDoctor Diagnosis*