Analyzing and Diagnosing Applications

CHAPTER 18 Analyzing Network and Tier Delays

A Transaction Analyzer model specifies a sequence of messages that are sent and received at different tiers. To analyze and troubleshoot an application effectively, it is important to understand the size and nature of the delays that occur between messages.

This section includes the following topics:

- Delay Categories
- Delays per Object (Message, Tier, Tier Pair, and Application)
- Application Message Dependencies
- User Think Time
- Parallel Effects

Delay Categories

AppTransaction Xpert can model the following categories of delay:

- **Network Delay—**

  Delay due to congestion, retransmissions, and other factors specific to the network in which the application was captured.

  AppTransaction Xpert calculates network delay based on the bandwidth and one-way latency values that were specified when the capture data was imported. If an application was captured at both the source and destination tier, AppTransaction Xpert can calculate exact network delays based on the measured departure and arrival times at each tier.

- **Tier Processing—**

  Delay due to application processing between message arrivals and transmissions at a specific tier. In most cases, AppTransaction Xpert assigns non-network delays to tier processing.

- **User Think Time—**

  Delay incurred whenever a client-side application waits for user input before it can proceed. If an application contains user think time, you can configure AppTransaction Xpert to model this type of delay.

  For more information, see User Think Time.

- **Parallel Effects—**

  In some cases, AppDoctor shows an additional category called Parallel Effects. This category appears when multiple types of delay occur in parallel, as often happens with HTTP and other protocols.

  For more information, see Parallel Effects.

Delays per Object (Message, Tier, Tier Pair, and Application)

When a packet trace is opened to create a Transaction Analyzer model, AppTransaction Xpert calculates delays for specific objects of interest. The following tables list the delays associated with each specific object.

Table 51 Application Message Delays

Delay	Description	References
Network Delay	Amount of transmission delay for an individual packet due to network delay (bandwidth, latency, protocol, and congestion effects) between the source and the destination tier	Application Message Dependencies User Think Time
Application Delay	Amount of delay due to application processing at a specific tier between message ***n*** and the previous message on which ***n*** depends	
User Think Time	Amount of delay due to user think time at a specific tier between the transmission of message ***n*** and the previous message on which ***n*** depends	

Table 52 Tier Delays

Delay	Description	References
Tier Processing	Total amount of delay due to application processing at a specific tier	"AppDoctor Statistics"
User Think Time	Total amount of delay due to user think time at the client tier	

Table 53 Tier Pair Delays

Delay	Description	References
Effect of Latency	Total amount of delay due to latency effects between two tiers	"AppDoctor Statistics"
Effect of Bandwidth	Total amount of delay due to bandwidth effects between two tiers	
Effect of Protocol	Total amount of delay due to protocol effects between two tiers	
Effect of Congestion	Total amount of delay due to congestion effects between two tiers	

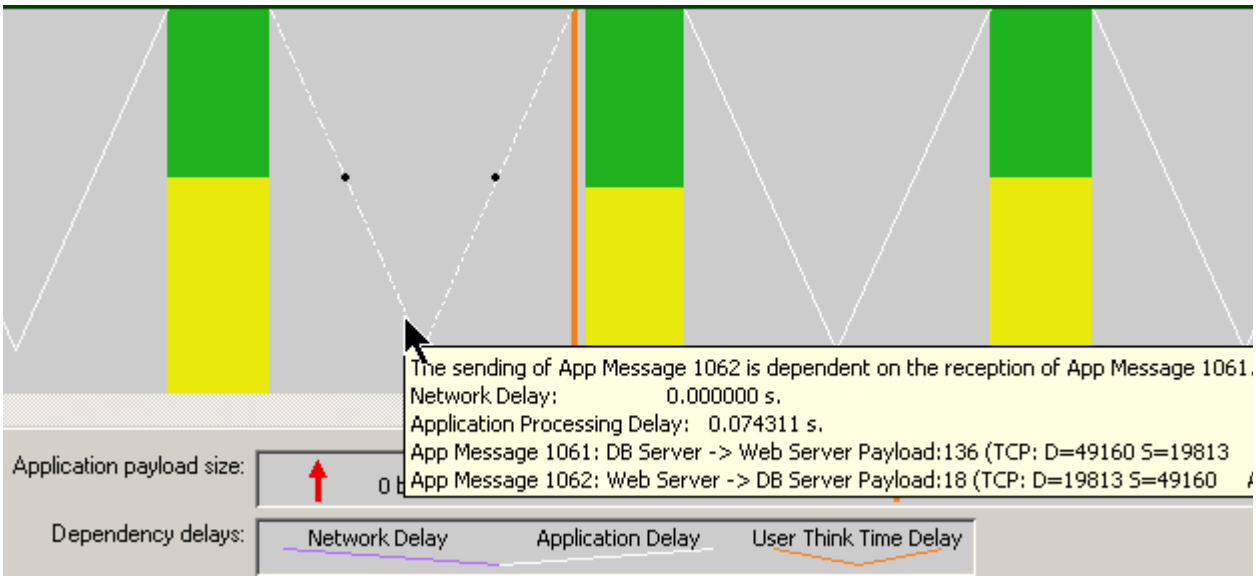
Table 54 Application Task Delays (Total per Transaction Analyzer Model File)

Delay	Description	References
Tier Processing	Total amount of delay due to application processing (all tiers)	"AppDoctor Summary of Delays" "AppDoctor Diagnosis" "AppDoctor Diagnosis Categories" Parallel Effects User Think Time
Latency	Total amount of delay due to latency effects (all tier pairs)	
Bandwidth	Total amount of delay due to bandwidth effects (all tier pairs)	
Protocol	Total amount of delay due to protocol effects (all tier pairs)	
Congestion	Total amount of delay due to congestion effects (all tier pairs)	
Parallel Effects	Total amount of delay due to parallel effects (all tiers/tier pairs)	
User Think Time	Total amount of delay due to user think time (client tier only)	

Application Message Dependencies

A dependency describes the causal relationship and delay time between two messages (message ***n*** and the previous message on which message ***n*** depends) at a specific tier. Dependencies can help you pinpoint situations in a data exchange where network and/or tier delays are impairing your application's performance.

Figure 109 Message Dependencies in the Application Message Chart



Dependencies have two important characteristics:

- Each dependency is associated with two sequential messages at a specific tier.
- Each dependency has an associated wait time, which is composed of two separate delays:
  - Tier (application/think time) delay—Delay at the tier between message ***n*** and the previous message on which ***n*** depends.

    In most cases, this delay is due to application processing. In some cases, a client-side application might require input from the user (for example, clicking Submit to confirm a request to a web server) before it can proceed. This type of delay is described in User Think Time.
  - Network delay—Transmission, propagation, and protocol delays incurred when transmitting the message. This is because the delay begins after the previous message has been completely received and ends when message ***n*** begins transmission.

    Most dependencies have no network delay. If a message arrives at a tier, and that tier sends a message in response, the delay between the two messages is due to application processing or think time.

    When the same tier sends two messages in sequence (message ***n*** and the previous message it depends on), the network effects of the previous message are discounted from the delay. If TCP is completely responsible for the delay between messages, as is the case with Nagle's algorithm, the entire delay is due to the network.

    The best place to visualize and analyze network delays is in AppDoctor (see "Diagnosing Applications with AppDoctor").

To view message dependencies, go to the Application Message Chart and select the Show Dependencies checkbox in the upper-right corner. Each dependency is composed of two lines that represent the proportion of

- Network Delay (purple line)
- Application Delay (white line)
- Think Time (orange line)

User Think Time

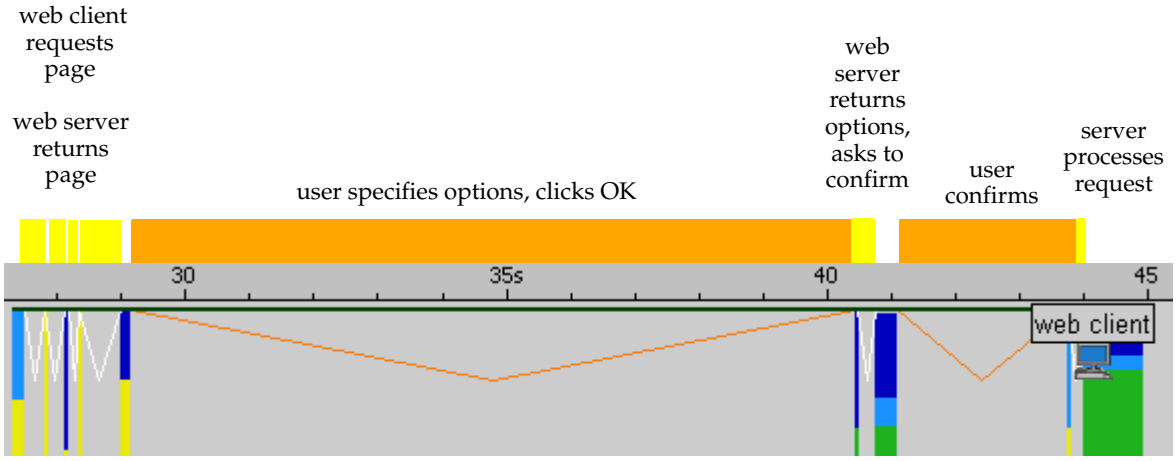
In the context of AppTransaction Xpert, *user think time* refers to the delay incurred when a client-side application requires user input before it can proceed. One complete user-level transaction might consist of several user-level actions that are separated by think time. In Figure 110, the server sends three separate pages; the second and third pages require user input before they can be sent. As a result, delays at the client tier might include user think time as well as application processing.

This section describes how to identify user think time in an application.

Note: AppTransaction Xpert assigns all non-network delays to application processing by default. If a Transaction Analyzer model includes any delays due to user think time, you must identify them as described in Identifying User Think Time Manually. Otherwise these think-time delays will distort your analysis results (especially in AppDoctor).

Note: AppTransaction Xpert permits user think time on the client tier only. By definition, the client is the tier that sends the first message.

Figure 110 Example: Application Delay and User Think Time at Client Tier



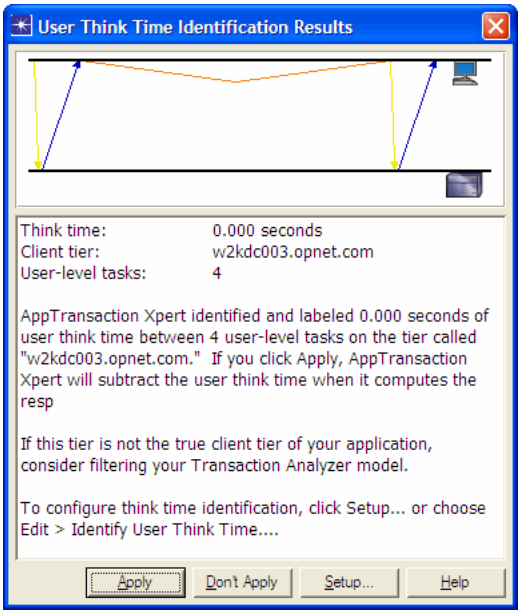
The following features enable you to view user think time:

- Application Message Dependencies—Shows the User Think Time between two messages
- "AppDoctor Summary of Delays"—Shows the total User Think Time as a percentage of the total response time
- "AppDoctor Statistics"—Shows the total User Think Time at the client tier, in seconds
- "QuickPredict"
- "QuickPredict Bar Charts"—Shows the total User Think Time as a percentage of the response time for a deployed application.

Identifying User Think Time in AppDoctor and QuickPredict

To obtain accurate results in AppDoctor and QuickPredict, you must identify all User Think Time— delays accurately. You might find, when you open one of these windows, that the following dialog box appears.

Figure 111 "User Think Time Identification Results" Dialog Box



This window appears when AppTransaction Xpert identifies client-tier delays that might be think time, but are currently assigned to application delay. Read this window carefully and determine whether the client tier (first talker) and the think-time delays are identified correctly. Then do one of the following:

- If the client tier and all user think time delays are identified correctly, click **Apply** and proceed.
- If the client tier is identified correctly and the Transaction Analyzer model contains no think time, click **Don't Apply** and proceed.
- If the tier identified as the client is not the first talker:
  - Click **Don't Apply**.
  - Filter the Transaction Analyzer model so that the first message in the model is the initial request sent by the client. For more information, see "Filtering Traffic".
- If you think that user think time is identified incorrectly, click **Setup** to change the settings for how user think time is calculated. For more information, see Identifying User Think Time Manually.

Identifying User Think Time Manually

AppTransaction Xpert identifies user think time when AppDoctor is run of the first time. When a new Transaction Analyzer model file is created, the initial user think time is 0 and all non-network delays at the client tier are assigned to application processing delay.

The following sections describe how to assign user think time:

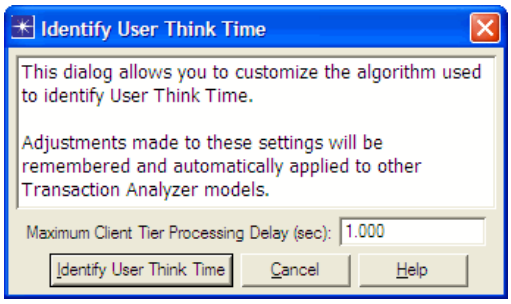
- Identifying Possible User Think Time Automatically
- Identify User Think Time (One Client-Side Dependency)

Note: AppTransaction Xpert assigns all non-network delay between two messages to either Application Delay or User Think Time. There is no way to divide the non-network delay in an individual dependency between these two categories.

Identifying Possible User Think Time Automatically

To identify user think time automatically for all dependencies on the client tier, choose Edit > Identify User Think Time… Then specify the Maximum Client Tier Processing Delay (sec) threshold and click Identify User Think Time.

Figure 112 "Identify User Think Time" Dialog Box



Maximum Client Tier Processing Delay (sec)

This preference specifies the maximum delay for application processing in one dependency. When you apply this operation, AppTransaction Xpert iterates through all dependencies. If the non-network delay in a dependency exceeds this threshold, AppTransaction Xpert assigns this delay to User Think Time.

Identify User Think Time (One Client-Side Dependency)

You can specify the category of non-network delay for individual dependencies in the Application Message Chart. Right-click on a dependency and choose "Treat dependency as user think time" or "Treat dependency as tier processing." This operation has no effect on the amount of network delay for that dependency.

Figure 113 Assigning a Dependency to User Think Time

Showing/Hiding User Think Time when Analyzing Applications

The "AppDoctor Summary of Delays" and "QuickPredict Bar Charts" windows have a "Show Think Time" checkbox that you can use to show/hide user think time in the window and corresponding analysis.

Clearing All User Think Time

Perform the following procedure to clear all user think time.

Procedure 59 Clearing All User Think Time

1. Clear all think-time delays that were identified automatically:

   1.1. Choose Edit > Identify User Think Time…

   - The "Identify User Think Time" Dialog Box appears.

   1.2. Enter a high value (such as 100) in the Maximum Client Tier Processing Delay (sec) field, then click the "Identify User Think Time" button.

   - All automatically-identified think time delays are cleared.
   - The message "No user think time was identified" appears in the status bar at the bottom of the window.

2. Reset the Maximum Tier Processing Delay to a reasonable default value:

   Note: If you do not change this value, the value specified in step 1.2. will be retained and think time will not be identified (unless the delay is very long).

   2.1. Choose Edit > Identify User Think Time… and enter a default value in the Maximum Client Tier Processing Delay (sec) field.

   2.2. Click the "Identify User Think Time" button.

   - The "User Think Time Identification Results" Dialog Box appears.

2.3. Click Don't Apply.

- The Maximum Client Tier Processing Delay (sec) field is reset, but the new setting is not applied.

3. For every dependency that was identified as Think Time manually, do the following:

3.1. Right-click on the dependency.

3.2. Choose "Treat dependency as tier processing".

End of Procedure 59

Parallel Effects

In some cases, AppDoctor shows an additional category called *Parallel Effects*. This category appears when multiple types of delay occur in parallel, as often happens with HTTP and other protocols.

Key Concept—If AppDoctor assigns a *significant* amount of delay to Parallel Effects, then you might need to address multiple issues to reduce the total response time.

A simple, high-level description of Parallel Effects is: *The amount of delay incurred when multiple types of delay (such as tier processing, bandwidth, and latency) occur simultaneously.* This section describes the Parallel Effects category in detail; for a summary of the most important points, see Parallel Effects Category: Summary.

This topic has the following subsections:

Parallel Effects and Irrelevant Traffic

WARNING: If the Parallel Effects category appears in AppDoctor, it might indicate that the Transaction Analyzer model contains extraneous traffic that is irrelevant to the application of interest. If a Transaction Analyzer model contains irrelevant traffic, AppDoctor results might not be accurate.

Each Transaction Analyzer model should contain traffic for one and only one user-level transaction, with all other traffic excluded. Many transactions are serial in nature: client requests file, server transmits file. Other transactions are parallel in nature: web client requests page, server sends multiple files (HTML code, Java applets, graphics files) in parallel. The Parallel Effects category should appear for the latter type of transaction only.

If the application is serial in nature—an FTP download, for example—and you still see a Parallel Effects category, this probably indicates that the Transaction Analyzer model contains extraneous traffic. You must do one of the following:

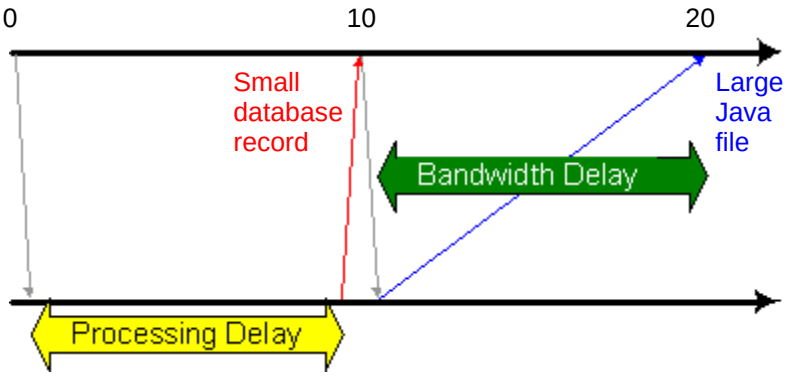
- Exclude the irrelevant traffic in AppTransaction Xpert
- Re-import the packet traces and filter the traffic more carefully during the import process.

For more information, see "Filtering Traffic".

Application Example 1: Serial Request/Responses

The following figure shows a highly simplified application, in which the client sends two requests: a database record that requires processing, and a very large Java file. Because there is only one transfer occurring at any one time, AppDoctor can divide the entire response time into individual categories.

Figure 114 Serial Application: Processing Delay, then Bandwidth Delay



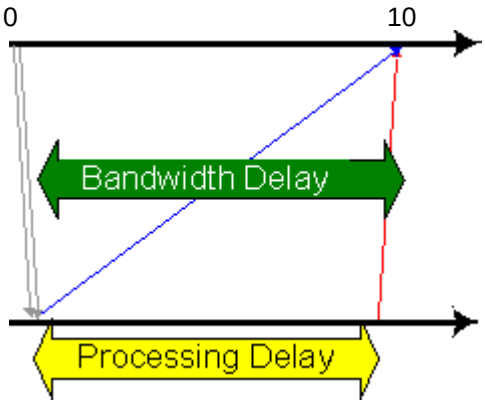
Response Time:

- Serial (v1).....20 seconds

Application Example 2: Parallel Request/Responses

Suppose the application developer rewrites the application so that the client sends both requests at once. Now there are two different types of delay occurring at once. Clearly it would be incorrect to assign the application delay to one effect or another; in this example, it is a mixture of two simultaneous effects. Therefore, AppDoctor assigns the delay incurred during this time window to the Parallel Effects delay category.

Figure 115 Parallel Application: Processing and Bandwidth Delay



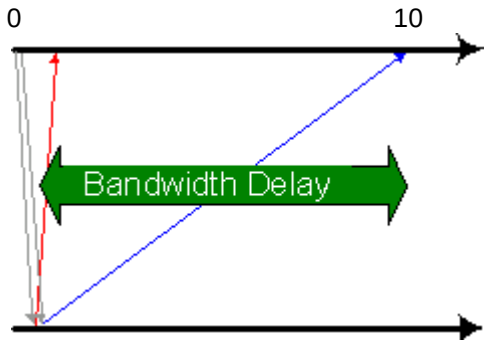
Response Times:

- Serial (v1).....20 seconds
- Parallel (v2).....10 seconds

Reduction in response time
by converting application
from serial to parallel.....**10 seconds**

Suppose the developer optimizes the server code so that it transmits the small database record almost immediately. Processing delay is nearly zero, but the response time is unchanged. Why? Because the previous version had two delays occurring in parallel, but only one delay was reduced. The large file transfer—and the associated bandwidth delay—remains.

Figure 116 Parallel Application: Processing Delay Reduced



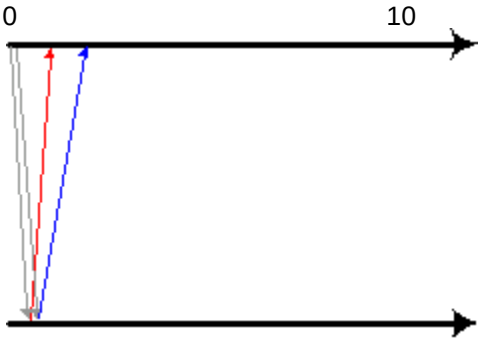
Response Times:

- Serial (v1).....20 seconds
- Parallel (v2).....10 seconds
- Reduce Processing (v3).....10 seconds

Reduction in response time by reducing processing alone.....**0 seconds**

Now, suppose the developer drastically reduces the size of the Java file. The bandwidth delay is now significantly reduced. By addressing both sources of delay, the developer reduced the total response time from about ten seconds to two seconds.

Figure 117 Parallel Application: Bandwidth Delay Reduced



Response Times:

- Serial (v1).....20 seconds
- Parallel (v2).....10 seconds
- Reduce Processing (v3).....10 seconds
- Reduce Bandwidth (v4).....2 seconds

Reduction in response time by reducing processing *and* bandwidth.....**8 seconds**

Example 2: Conclusion

This example illustrates a key aspect of parallel effects: To reduce the total response time of this application significantly, you might need to address multiple sources of delay. Reducing one individual delay will not necessarily reduce the amount of delay that appears as Parallel Effects in AppDoctor.

This is not to say that mixed delays are necessarily bad or undesirable. In fact, some applications perform multiple tasks in parallel as a way to speed up transactions and to use network resources more efficiently. (Converting our example application from serial to parallel reduced the response time from 20 seconds to about ten seconds.)

When AppDoctor shows a delay due to parallel effects, it is simply saying: *If you want to reduce this amount of delay, you need to examine multiple types of delay (tier processing, bandwidth, and/or latency) and not just one.*

Analyzing Delays in the “Parallel Effects” Category

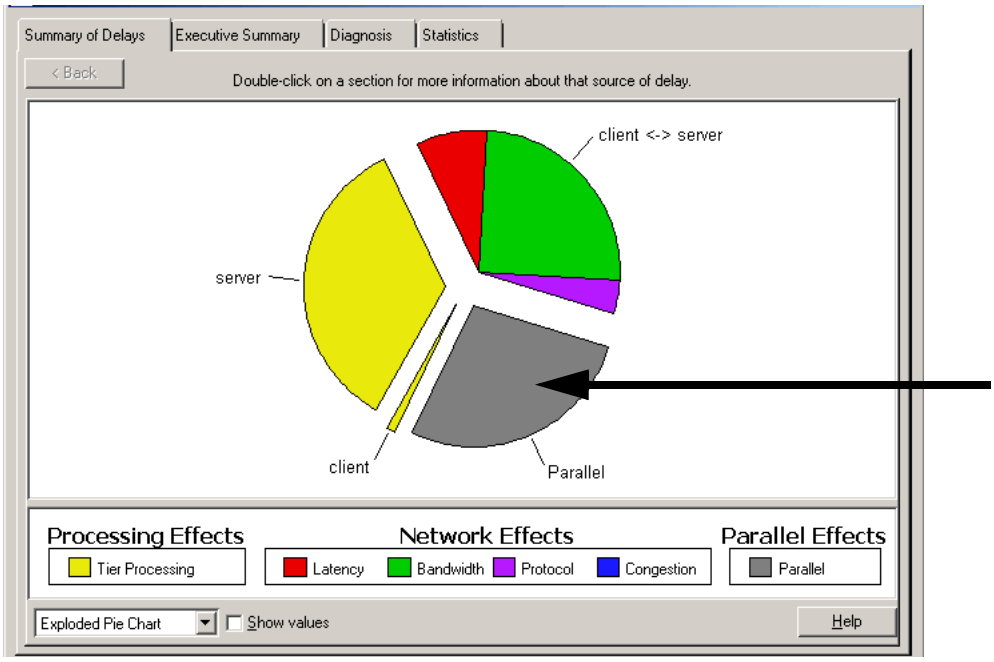
When AppDoctor finds multiple, simultaneous delays in a Transaction Analyzer model, it assigns the time incurred to the Parallel Effects delay category. Two features are especially useful for analyzing these simultaneous delays:

- To analyze these delays in the current application, open AppDoctor Summary of Delays and double-click on the Parallel Effects of Delay segment. For more information, see Viewing the Individual Components of Parallel Effects.
- To predict how changes in individual effects will affect the amount of delay assigned to Parallel Effects (and the overall response time), open the “QuickPredict Bar Charts“ window. (Choose Simulation > QuickPredict Bar Chart.)

Viewing the Individual Components of Parallel Effects

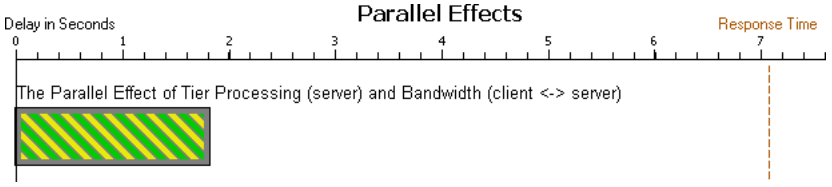
If the AppDoctor Summary of Delays shows a significant amount of delay in the Parallel Effects category, double-click on the gray section of the pie chart to display the Parallel Effects drill-down page (Figure 120).

Figure 118 AppDoctor Summary of Delays with Parallel Effects



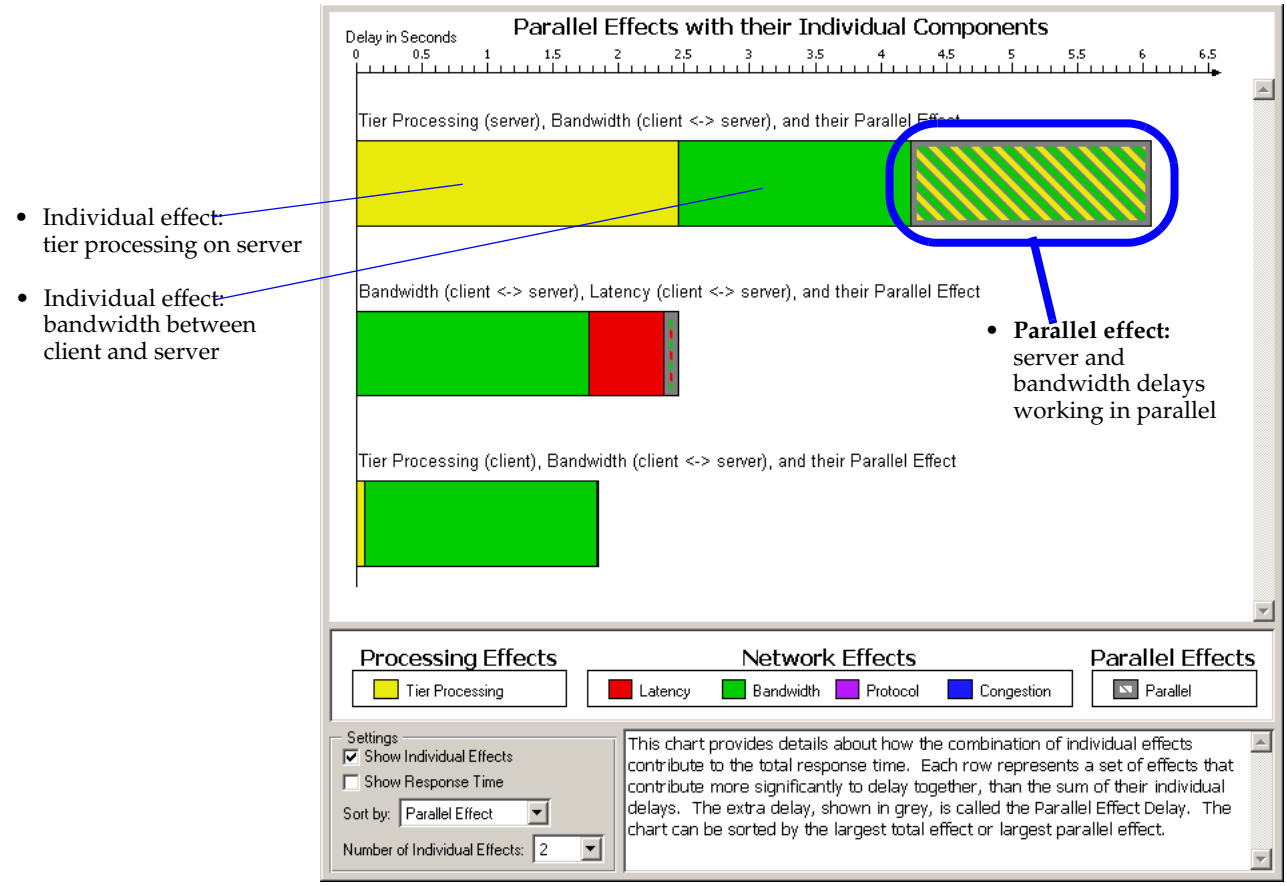
In the following figure, the graph shows that processing delay and bandwidth are working in parallel, and that this “parallel effect” is responsible for ~1.8 seconds of delay (out of a total response time of ~7.1 seconds).

Figure 119 Parallel Effects Drill-Down: Parallel Effect Bar Graph



The Parallel Effects drill-down page has a “Show Individual Effects” checkbox. In the previous graph, this option was turned off so that we saw *only* the delay assigned to Parallel Effects. The following figure shows the same graph with the individual effects (processing and bandwidth), as well as the “parallel effect” incurred when both individual effects are occurring at the same time.

Figure 120 Delay in “Parallel Effects” Category, with Individual Delays



To reduce the total amount of Parallel Effects delay significantly, we might need to change *both* the tier processing at the server *and* bandwidth between the client and the server. A reduction in one individual effect—even a significant reduction—does not guarantee that the amount of delay in the Parallel Effects category will be reduced.

In this sense, Parallel Effects differs from the other “individual” delay categories such as Tier Processing, Bandwidth, and Latency. If you reduce the Tier Processing or Bandwidth delay by *X* seconds, this one change alone will reduce the total response time by *X* seconds. There is no equivalent cause-and-effect relationship for Parallel Effects.

Note: This example shows the combination of two individual effects. However, a Parallel Effects delay can be composed of three or four effects. This is especially true for multi-tiered applications. It is good practice to change the “Number of Individual Effects” menu item to determine if more than two effects are combining to produce significant amounts of Parallel Effects delay.

Parallel Effects Category: Summary

If an application has tier processing, bandwidth, and latency delays occurring in parallel, then the Parallel Effects delay category appears in the "AppDoctor Summary of Delays" and "AppDoctor Statistics" pages.

Note the following important aspects of Parallel Effects:

- If AppDoctor shows a delay due to Parallel Effects, then you might need to address multiple types of delay to reduce this delay. Reducing only one effect might not reduce the amount of Parallel Effects delay.
- If the transaction of interest is sequential rather than parallel in nature, a Parallel Effects delay indicates that the Transaction Analyzer model probably contains irrelevant traffic. In this case, try to exclude the irrelevant traffic or re-import the capture data and filter more carefully.
- When AppDoctor shows a significant amount of delay in the Parallel Effects category, it is good practice to examine how individual effects are contributing to the delay. To do this, double-click on the Parallel Effects section in the AppDoctor Summary of Delays chart. In the Parallel Effects with their Individual Components graph (Figure 119), you can see:
  - How individual delays contribute to differing amounts of Parallel Effects delay
  - How much these individual delays, plus the Parallel Effects delay, contribute to the total response time
  - Whether two or more delays are combining to produce significant amounts of Parallel Effects (set the "Number of Individual Effects" menu)
- To predict how changes in individual delays affect the total Parallel Effects delay, view the application in the "QuickPredict Bar Charts" window. (Choose Simulation > QuickPredict Bar Charts.)

CHAPTER 19 Protocol Decodes

Protocol decodes can “drill down” into the protocol and application data of individual packets. You can view protocol decodes in the “Tree View“, “Data Exchange Chart“, and “Tier Pair Circle“.

AppTransaction Xpert Decode Module includes additional support for detailed, protocol-specific decodes as well as transactional-analysis capabilities and decodes for additional protocols (such as HTTP and Citrix).

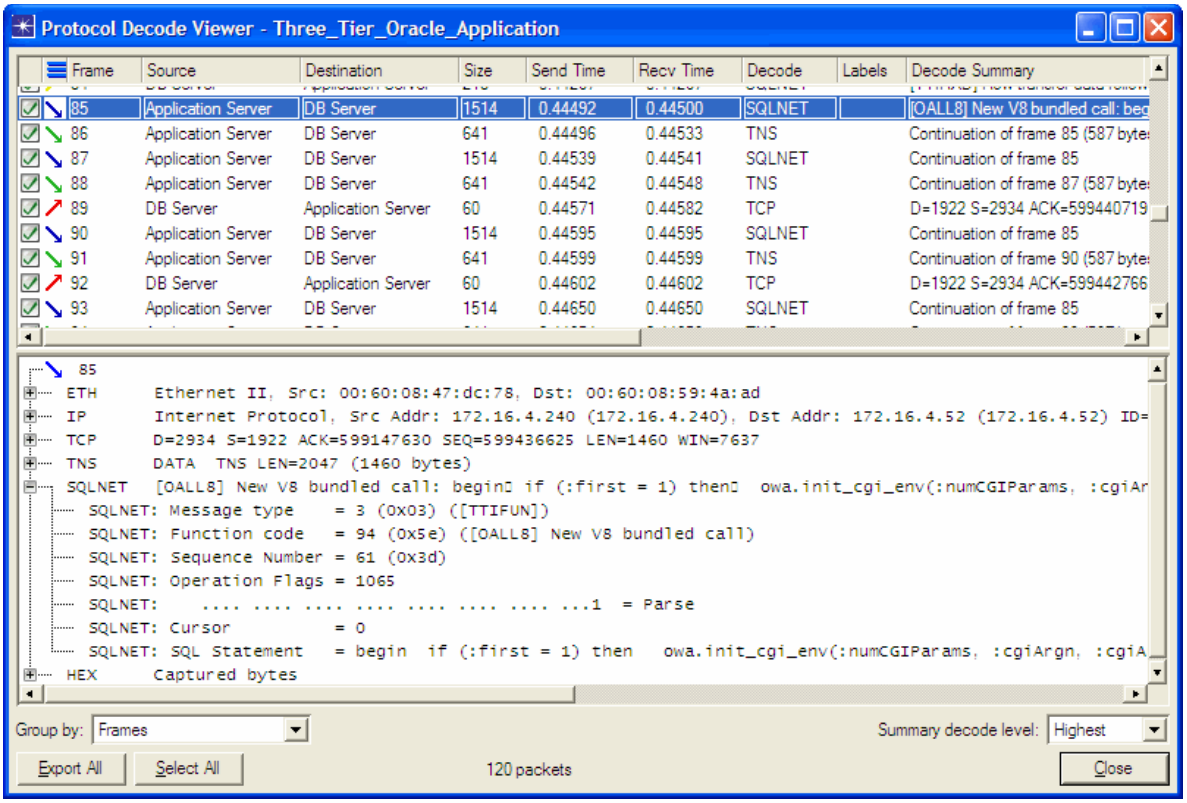
This section has the following topics:

- Protocol Decode Viewer
- Streamed Data Viewer
- Finding Packets
- AppTransaction Xpert Decode Module
- Redecoding Traffic
- Troubleshooting: Incorrectly Identified Protocols

Protocol Decode Viewer

The Protocol Decode Viewer, shown in the following figure, displays detailed protocol decodes for one or more packets in a Transaction Analyzer model. Use this feature to view detailed decodes of the protocol fields in individual packets. You can also view the captured bytes of individual packets in both HEX and ASCII format by expanding the HEX node in the bottom treeview.

Figure 121 Protocol Decode Viewer



To view protocol decodes:

1) Select one or more packets or messages in the Tree View, Tier Pair Circle, or Data Exchange Chart.

2) Right-click on the selection and choose “Show Protocol Decodes for Selected Items”.

For information about the operations in this window, see:

- Table 55 Protocol Decode Viewer: Main Window Options
- Table 56 Protocol Decode Viewer: Right-Click Menu Options

If you have a license for AppTransaction Xpert Decode Module, you can view additional data in the Protocol Decode Viewer. For more information, see AppTransaction Xpert Decode Module.

Table 55 Protocol Decode Viewer: Main Window Options

Option	Description
Group By	Determines how the frames are organized in the treeview: • Frames—All frames are organized in numerical order • Connections—Selected frames are organized by connection • Application Messages—Selected frames are organized by application message Note that if you group by Connections or Application Messages, the treeview shows only the frames you selected to decode—not necessarily all the frames in that connection or message.
Export All	Exports decode data for all frames in the viewer to a text (.txt) file. To export a subset, select the frames, right-click, and choose Export Selected. As part of the export, specify whether to include - summary lines - detailed decodes
Expand All	Expands all top-level branches (frames, connections or messages) in the treeview.
Select All	Selects all frames in the viewer.
Summary Decode Level	Determines the protocol layer to show in the Decode and Decode Summary fields.

The following table lists the menu operations that are available when you right-click on one or more selected frames in the Protocol Decode Viewer.

Table 56 Protocol Decode Viewer: Right-Click Menu Options

Option	Description
Copy to Clipboard	Copy the selected information to the clipboard. If you select a sub-branch within a frame, only the corresponding decode information is copied. If you select one or more frames, all decode information for those frames is copied.
Exclude Others	Exclude all items *except those currently selected* from the Transaction Analyzer model.
Exclude Selected Items	Exclude the selected items from the Transaction Analyzer model.
Export Selected	Export decode information for the selected frames. This menu item appears only when you right-click on one or more selected frames.
Graph Statistics for Selected Items	View statistic graphs for the selected items. For more information, see "Viewing Statistics".
Permanently Delete Others	Delete all items *except those currently selected* from the Transaction Analyzer model.
Permanently Delete Selected Items	Delete selected items from the Transaction Analyzer model.
Remove from Selection	Remove the selected frame(s) from the decode viewer tree. This does not remove the frames from the Transaction Analyzer model.

Table 56 Protocol Decode Viewer: Right-Click Menu Options (Continued)

Option	Description
Rename <*tier_name*>	Rename the tier manually or using DNS lookup.
Set as Time Zero	Changes the Send Time for the selected packet to 0.00000. This operation is useful when you know that this is the first packet sent in the transaction of interest, and that earlier packets are irrelevant.
Show Protocol Decodes for Selected Items	View the selected frames in a new Protocol Decode Viewer window.
Show Streamed Bytes for Selected Items	View the selected packets in the Streamed Data Viewer. For more information, see Streamed Data Viewer.
Zoom to DEC Zoom to Tree	Zoom the Data Exchange Chart or Tree View window to the selected frames.

Related Topics

- Protocol Decodes

Streamed Data Viewer

The Streamed Data Viewer shows all data transferred over a particular network layer as a single concatenated stream of data. This feature is particularly useful when you want to view all application data (typically all data above TCP) transferred over a single connection or within a single application message.

To open the Streamed Data Viewer:

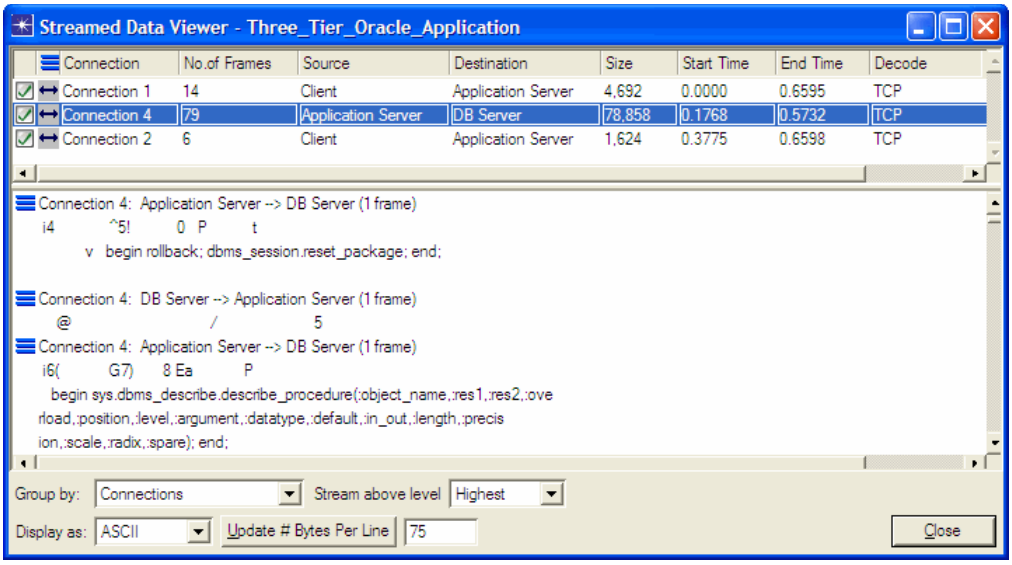
1) Select a group of packets or messages in the Protocol Viewer, Data Exchange Chart, or Tree View window.

2) Right-click and choose “Show Streamed Bytes for Selected Items”.

For information about the operations in this window, see:

- Table 57 Streamed Data Viewer: Main Window Options
- The right-click menu operations in the Data Stream Viewer are identical to those in the Protocol Decode Viewer; see Table 56 Protocol Decode Viewer: Right-Click Menu Options

Figure 122 Streamed Data Viewer



The following table lists the main window options for the Streamed Data Viewer.

Table 57 Streamed Data Viewer: Main Window Options

Option	Description
Display As	Displays the streamed data in ASCII, Hexadecimal, or EBCDIC format
Group By	Determines how the streamed data is shown: • Connections—Show a single data stream for each connection • Application Messages—Show a single data stream for each message **Note—**This window shows data for selected items only, and not necessarily all the data in that connection or message.
Stream Above Level	Determines the protocol layer above which all data is shown. If this menu is set to 0, the stream includes all data in the stream; if this menu is set to Highest, the stream includes data at the highest (application) layer only. The Highest setting is typically the most useful for viewing streams of application data.
Update # Bytes Per Line	Determines the number of bytes to show in each line of the Streamed Data Viewer treeview

Related Topics

- Protocol Decodes

Finding Packets

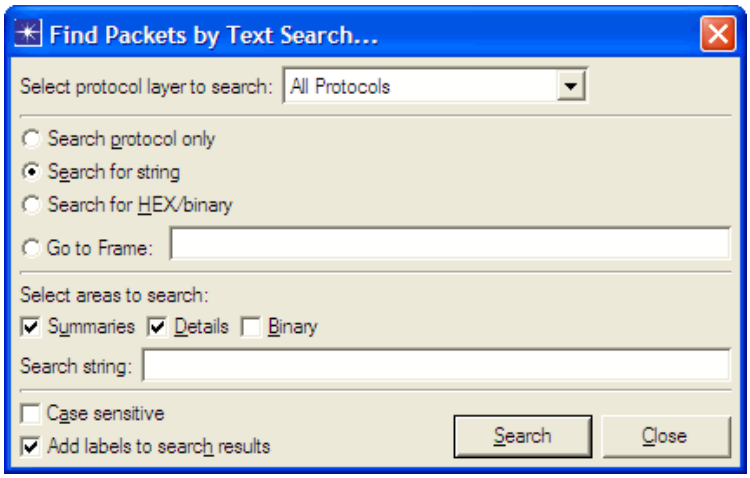
You can search for packets based on:

- Protocol data (for example, all packets that contain HTTP data)
- Packet contents (all packets that contain a particular string)
- Frame number (as specified in the original packet trace file)

Note: To search for packets, AppTransaction Xpert must be able to read the packet trace file(s) used to create the model file. If you see a message indicating that the trace files cannot be found, choose File > Trace File Info in the Treeview to specify the location of the packet trace files.

To search for specific packets, choose Edit > Find Packets, which opens the "Find Packets by Text Search…" dialog box.

Figure 123 Find Packets by Text Search… Dialog Box



You can find packets based on a search string, a specific protocol, or a set of frame numbers:

- To search for packets that contain a specific string, select "Search for string" or "Search for Hex/Binary" and enter the search string.

  Hint—When performing a search for a string, If the "Add labels to search results" option is selected, then packets containing the search string are labeled in the Data Exchange Chart. Use this option to quickly find and visualize a subset of traffic. This option is also useful for creating a screenshot for a report. Optionally, use the "Colorize Labels" visualization (View > Visualization > Colorize Labels) to color the packets based on search strings. (For example, after searching for "HTTP", use the visualization option to color all packets containing the string "HTTP" with the same color.) The display of labels is cumulative for subsequent string searches. To clear labels from the Data Exchange Chart, choose Edit > Remove All Labels.

- To search for packets that contain a specific protocol, set the "Select protocol layer to search" menu, then select "Search protocol only".
- To search for packets by frame number, select "Go to Frame" and enter the frame numbers. For example, enter `21, 25-35` to find packets 21 and 25 through 35.

After clicking Search, the results are shown in the Protocol Decode Viewer. To view specific packets in the Tree View or Data Exchange Chart, select one or more packets (by shift-clicking or dragging the cursor), right-click, and choose Zoom to DEC or Zoom to Tree.

Related Topics

- Protocol Decodes

AppTransaction Xpert Decode Module

This section describes the features included with AppTransaction Xpert Decode Module:

- Enhanced Protocol Decodes
- Transaction Analysis with AppTransaction Xpert Decode Module
- Packet-Specific Dependencies
- Enhanced HTTP Transaction Analysis with the AppTransaction Xpert Decode Module
- Citrix Decodes in AppTransaction Xpert Decode Module
- Web Services and Transaction Analysis
- MS SQL Decodes
- DB2 Decodes and Transaction Analysis
- CORBA Transaction Analysis
- VoIP Analysis

Enhanced Protocol Decodes

AppTransaction Xpert Decode Module enables you to pinpoint application-specific messages (for example, SQL statements) and problematic sub-transactions that might be causing delays in your application. This module can decode over 400 protocols and applications.

To display decodes, select a set of packets or messages (in any window), right-click, and choose "Show Protocol Decodes for Selected Items".

Transaction Analysis with AppTransaction Xpert Decode Module

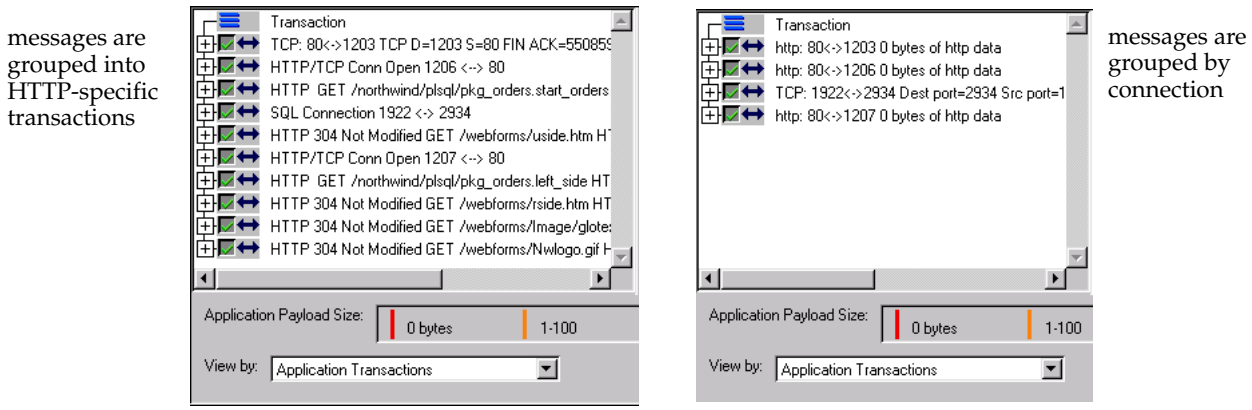
With AppTransaction Xpert Decode Module, AppTransaction Xpert can organize a task into separate transactions based on certain protocols. For example, AppTransaction Xpert can decode HTTP messages and organize a task into separate GET and connection open/close transactions. This feature enables you to view a task as a series of protocol-specific transactions (in addition to the standard AppTransaction Xpert organization based on connections, messages and packets).

Viewing Application Transactions

You can view the component transactions of a Transaction Analyzer model in the Tree View page. Set the View By pull-down menu to Application Transactions or Tier Pairs/Application Transactions. (For more information, see "Tree View".)

The following figure shows how AppTransaction Xpert reads the same task (a three-tier, web-based transaction) with and without AppTransaction Xpert Decode Module (ADM).

Figure 124 Per-Transaction View of a Web-Based Transaction With and Without AppTransaction Xpert Decode Module

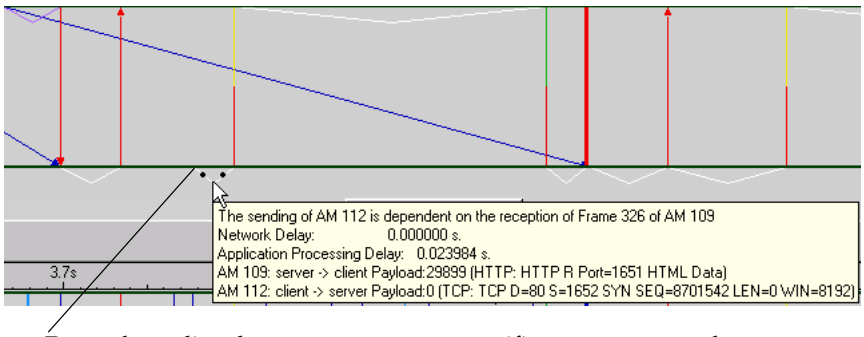


Packet-Specific Dependencies

Most dependencies record the cause-and-effect relationship between entire messages. Sometimes a message might depend on a particular packet within a message. These dependencies are known as *packet-specific dependencies*. You can determine that a dependency is packet-specific if it does not visibly connect to an arrow in the Application Message Chart. This is different from a standard dependency line, which visibly connects two message arrows. For more information about dependencies, see "Application Message Dependencies".

In the following figure, AppTransaction Xpert determined that Application Message 112 (an HTTP acknowledgement message) depends on the client tier receiving a specific packet in Application Message 109 (long blue arrow).

Figure 125 A Packet-Specific Dependency



Enhanced HTTP Transaction Analysis with the AppTransaction Xpert Decode Module

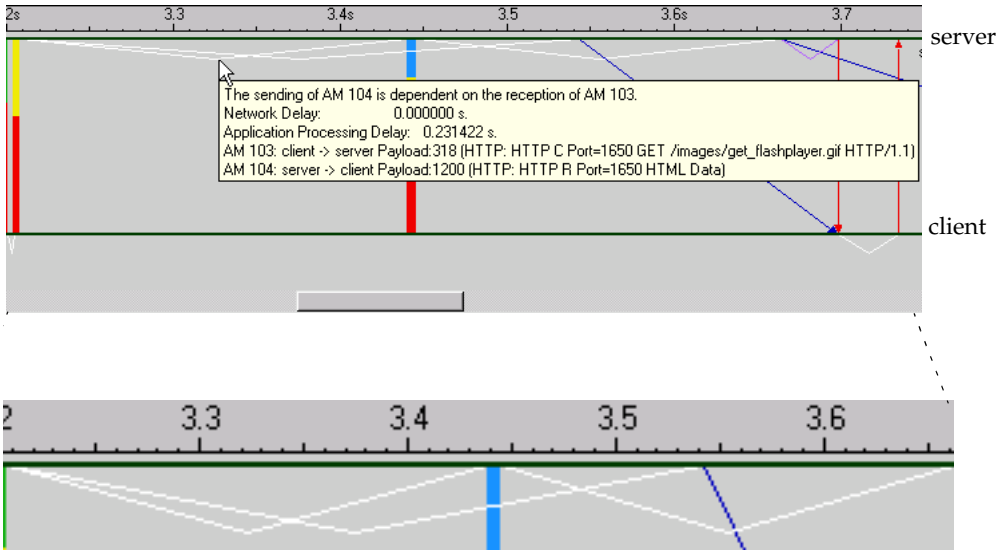
With AppTransaction Xpert Decode Module, AppTransaction Xpert can calculate messages and dependencies based on individual HTTP transactions. This results in more accurate analyses. Examples of HTTP-specific transactions include

- An HTTP/TCP Connection Open or Connection Close
- An HTTP GET request followed by a response

When an HTTP response depends on a request from another tier, AppTransaction Xpert sets up a dependency at the responding tier. This might result in multiple "dependency chains" that track multiple request/response transactions.

The following figure shows an HTTP application as seen with AppTransaction Xpert Decode Module. In this task, a web client (bottom tier) requests a web page that includes multiple GIF files. The Application Message Chart shows multiple dependencies as the server transfers a series of GIF files to the client. Note that AppTransaction Xpert can decode the specific HTTP message from the client to the server (a GET message that requests the file get_flashplayer.gif).

Figure 126 Dependencies for Multiple HTTP Transactions



You can view the graphic files transferred in an HTTP transaction, as described in the following procedure.

Procedure 60 Viewing a GIF or JPEG File in an HTTP Transaction

1. Open a Transaction Analyzer model of an HTTP transaction.

   For this procedure to work successfully, the following conditions must be true:

   - AppTransaction Xpert must be able to read the packet trace(s) used to create the Transaction Analyzer model.
   - The file must include at least one GET transaction of a GIF or JPEG file.

2. In the Tree View page, set the View By pull-down menu to Application Transactions.

3. In the treeview (left) pane, select an HTTP GET transaction in which a GIF/JPEG file is transferred. (You might need to resize the treeview pane so that you can see details about the GET transaction.) A GIF/JPEG transaction has a line that reads like this:

   `HTTP GET /<dir_name>/<file_name>.gif HTTP 1.1...`

4. Right-click on the GIF/JPEG transaction and choose HTTP > Save/View Object.

   - The Choose Path for Saving HTTP Object dialog box appears.

5. Make sure that the "View objects in web browser" checkbox is selected, then click Save.

End of Procedure 60

Citrix Decodes in AppTransaction Xpert Decode Module

AppTransaction Xpert Decode Module (ADM) can decode Citrix traffic. Use the following features to view Citrix results:

- Tree View tabbed page—When the View By menu is set to “Tier Pairs - Application Transactions”, the treeview groups similar messages together. For example, you might see an “ICA Keyboard” transaction that groups all keystrokes, or an “ICA Initialization” transaction that groups all packets in the initialization sequence.
- AppDoctor window—If a task has Citrix traffic, the AppDoctor window includes a “Citrix ICA” tabbed page that shows summary information and statistics about the Citrix traffic.
- Protocol decodes—ADM can decode CGP- and ICA-specific information in individual packets.

Capturing Citrix Traffic

Note the following:

- You must capture the Citrix traffic as described in Procedure 61.
- For a complete list of supported Citrix versions, see the System Requirements on the Support Center.
- If Citrix is using a non-default port, you must edit the following file to specify the port number:

  <*install_dir*>\sys\configs\protocols_v2.ace_dict

- ADM does not decode encrypted traffic. Before capturing Citrix traffic, set the encryption level to Basic or None. Typically, Basic encryption encrypts only the traffic in which the username and password are exchanged; all other traffic is uninterrupted. For more information, see the following procedure and the Citrix documentation.

Procedure 61 Capturing a Citrix Application

1. Verify that the encryption level on the Citrix client is set to Basic or None.

   For information about setting the encryption level on the Citrix client, see the Citrix documentation.

2. Start the capture operation on the capture agent or protocol analyzer before connecting to the Citrix server.

   - If using the Citrix Program Neighborhood, start the capture before double-clicking on the ICA connection.
   - If connecting through NFuse, start the capture before clicking on the application or server.

   For general information about capturing application traffic with AppTransaction Xpert and other programs, see “Capturing Application Traffic: Overview“.

3. Connect to the Citrix server and run the application transaction that you want to capture.

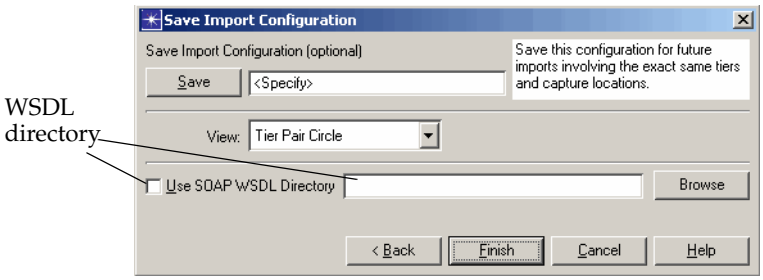
End of Procedure 61

Web Services and Transaction Analysis

You can view AppDoctor statistics and transaction information for .Net, SOAP, and UDDI traffic. You can see SOAP-specific information in the Tree View tabbed page, in AppDoctor, and in the Protocol Decode Viewer.

For specific instructions for importing packet trace files, see "Creating a Transaction Analyzer Model". If you want to specify a WSDL directory during the import capture operation, use the following Web services dialog box to specify the directory you want. You must specify a WSDL directory in order for AppDoctor analysis to take place.

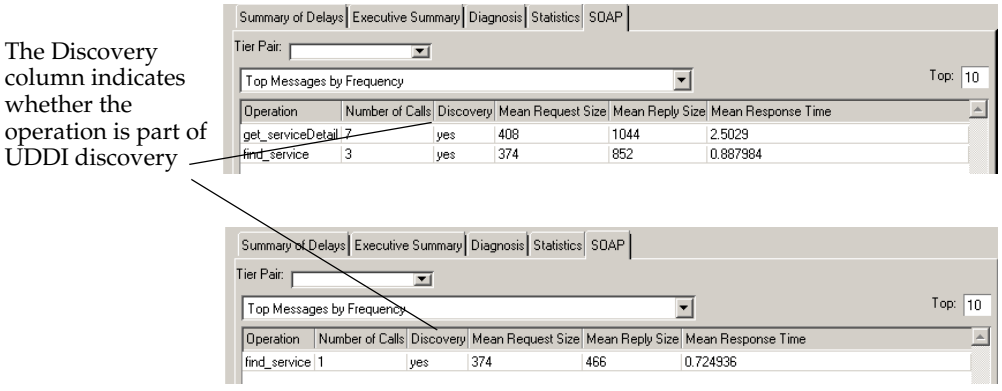
Figure 127 Web Service Import Configuration Dialog Box



Note: SOAP will not appear unless you select a WSDL directory during import and the WSDL files contained in this directory match the Web Services Message in the trace.

The Web Services Decode statistics page looks similar to the CORBA Decode page described previously. The first example in the following figure shows the statistics for two operations (i.e., function calls) and the second example shows a single SOAP operation.

Figure 128 SOAP Tabbed Page in AppDoctor Window



The drop-down menu above the statistical display provides four sorting options:

- Top messages by frequency
- Request size (in bytes)
- Reply size (in bytes)
- Speed (with the slowest messages at the top of the list)

In the Tier Pair: drop-down menu, you can view the total of tier pairs or each of the different tier pairs.

The main screen shows a list of the function calls. The default setting is 10 operations. You can edit the number in the Top: text box to show more or fewer operations.

MS SQL Decodes

AppTransaction Xpert Decode Module can analyze Microsoft SQL transactions. The SQL decode engine is based on the TDS protocol specifications as described in the FreeTDS website (http://www.freetds.org). You can view SQL-specific information in the following windows:

- The Protocol Decodes Viewer shows SQL decode information.
- The Tree View tabbed page organizes a Transaction Analyzer model into transactions based on SQL data.

There are two preferences that control how SQL data is displayed in the Protocol Decodes Viewer:

- ace_decoder_ms_sql_num_rows_per_response—The maximum number of rows that appear for each response.
- ace_decoder_ms_sql_string_display_length—The maximum number of characters shown for each individual SQL string.

DB2 Decodes and Transaction Analysis

AppTransaction Xpert supports decodes of DB2 versions 8.1, 8.2, 9.0, and 9.2. The Tree View page organizes a Transaction Analyzer model into separate DB2 transactions. The Protocol Decodes Viewer shows details of SQL operations and DRDA commands.

Note the following:

- It is good practice to capture DB2 transactions the same way you capture other types of transactions: start capture => start DB2 transaction => end DB2 transaction => stop capture. This ensures that you capture all the traffic in the transaction of interest.

  If your capture data does not include the initial DRDA Environment information, you can still decode the DB2 data. However, you might need to set the following preferences:

  – ace_db2_client_is_big_endian

  – ace_db2_server_is_big_endian

  To view these and other preferences, choose Edit > Preferences and search for “db2”.
- AppTransaction Xpert Decode Module does not decode packets that were encrypted using DB2 security features.
- AppTransaction Xpert Decode Module supports ASCII and EBCDIC encoding schemes only for SQL data.

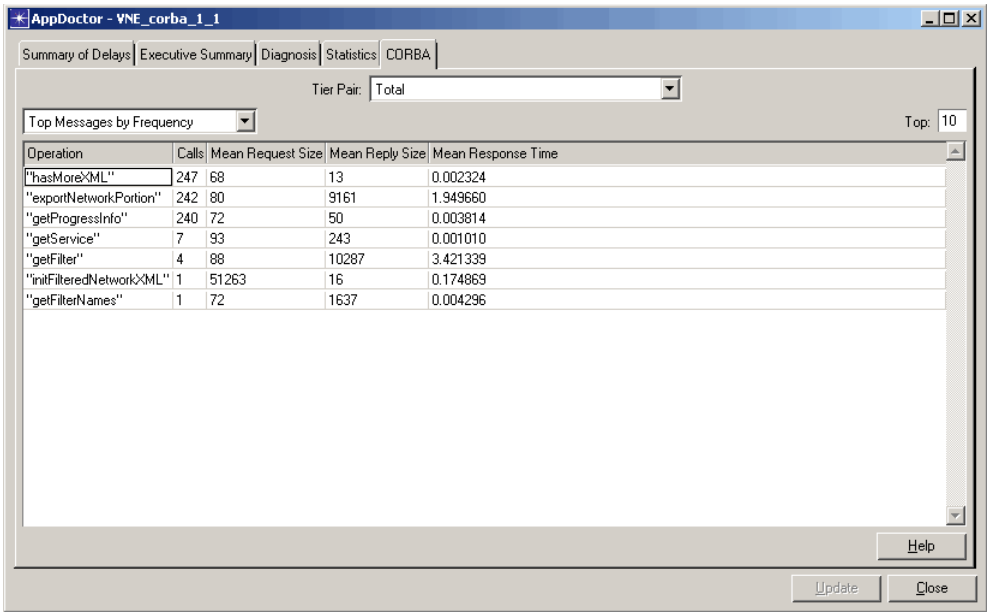
CORBA Transaction Analysis

If a Transaction Analyzer model contains CORBA traffic, the AppDoctor window incudes a CORBA tabbed page with summary and statistic information. CORBA decode summaries are applicable for request and reply messages only. When working with decode summaries, it is important to remember that the ID number of the request message decode must match the ID number of the corresponding reply message decode.

The CORBA tabbed page summarizes information about CORBA operation invocations contained in the Transaction Analyzer model. There are multiple summary tables available on this page, so you can view the top CORBA operations based on a specific threshold (such as frequency or response time).

The following figure shows a typical CORBA tabbed page.

Figure 129 CORBA Page



The drop-down menu in the upper-left provides four sorting options:

- Top messages by frequency
- Request size (in bytes)
- Reply size (in bytes)
- Speed (with the slowest messages at the top of the list)

Using the "Tier Pair" drop-down menu, you can select whether summary tables are computed across all tier pairs (Total) or for an individual tier pair.

The main screen shows a list of the function calls. The default setting is 10 operations. You can edit the number in the Top: text box to show more or fewer operations.

VoIP Analysis

AppTransaction Xpert Decode Module can analyze Voice-over-IP transactions. The Tree View page shows each VoIP call as a separate transaction, and organizes each transaction into the following subtransactions:

- Call Setup—which runs over an application-signaling protocol such as SIP. Each SIP subtransaction is further organized into separate subtransactions based on SIP branch.
- Data Transmission—which runs over RTP. The treeview further organizes RTP connections into separate subtransactions.
- RTCP Connections—which includes information about simple packet loss, jitter, and endpoint times.

If a Transaction Analyzer model contains VoIP traffic, the AppDoctor window includes a VoIP tabbed page that shows the following information for each VoIP call: summary information, packet statistics, and call statistics (such as call setup time, max jitter, out-of-sequence packets, packet loss, max delta, media type, and sampling rate).

Related Topics

- Protocol Decodes

Redecoding Traffic

The “Redcode As” operation allows you to redecode traffic to a specific port as a specific protocol. Note that only a single port for a single tier can be redecoded at a time, and it is applied to all of the selected connections to that port.

The “Redecode As” operation is useful when:

- The Transaction Analyzer model contains traffic that is not automatically decoded when the model is created.
- Updating the external decoder used by AppTransaction Xpert. If the Transaction Analyzer model was created with a previous decoder, you can redecode selected traffic so that the Protocol Decodes Viewer shows decodes from the most recent decoder.

  Note: If using the most recent decodes is a priority, it is best to re-create the Transaction Analyzer model so that all decode information is up-to-date.

Hint: If the results of the redecode is not as expected, perform the redecode procedure again, but select the other port of the connection from the “Decode traffic to” pull-down menu.

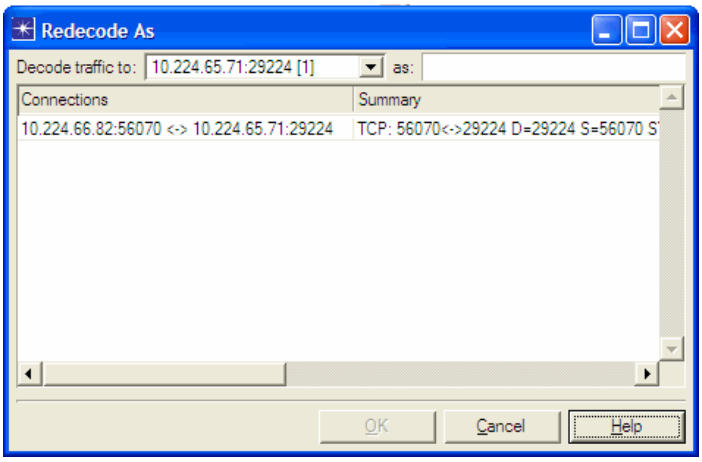
Procedure 62 Redecoding Traffic

1. In the Tree View or Tier Pair Circle page, select one or more connections.

   Hint: Hold down the Ctrl key to select individual connections. Hold down the Shift key to select a range of connections.

2. Right-click on a selected connection and choose “Redecode As…”.

   ➢ The “Redecode As” dialog box appears.



   The dialog box includes the following options and fields:

   - “Decode traffic to” pull-down menu—Lists the selected ports (including tier name and port number). The number in the square brackets represents the number of connections selected that match the tier and port. The list is sorted by the number of matching connections.
   - “as” pull-down menu—Lists the protocols to which you can redecode traffic.

- Connections column—Lists all of the connections that will be redecoded when OK is selected.
- Summary column—Displays a summary for the connection.

3. From the "Decode traffic to" pull-down menu, select the server port.

   Selecting the correct server port will result in more accurate decodes, particularly for database protocols.

4. From the "as" pull-down menu, select the TCP/UDP protocol that you want to redecode the traffic as. Alternately, select "Edit…" and manually enter the protocol name.

5. Click OK.

   ➢ The selected traffic is redecoded.

End of Procedure 62

Note: The system configuration files "adm_redecode_as.tcp.ace_dict" and "adm_redecode_as.udp.ace_dict" control the list of items that are displayed in the "as" pull-down menu. To add or delete an item from the list, edit the appropriate file in a text editor. For information about how to persistently force decoding, see Knowledge Base S21363 on the Support Website.

Related Topics

- Protocol Decodes

Troubleshooting: Incorrectly Identified Protocols

Protocols are identified by the communication port. AppTransaction Xpert includes a list of default ports and their associated protocols. If your organization uses a different port other than the default port, then the protocols may be misidentified and decoded incorrectly.

To view the default ports associated with protocols, open the <install_dir>\sys\configs\protocols_v2.ace_dict file. If your organization uses a different port than the port specified in the file, then edit and save the file.

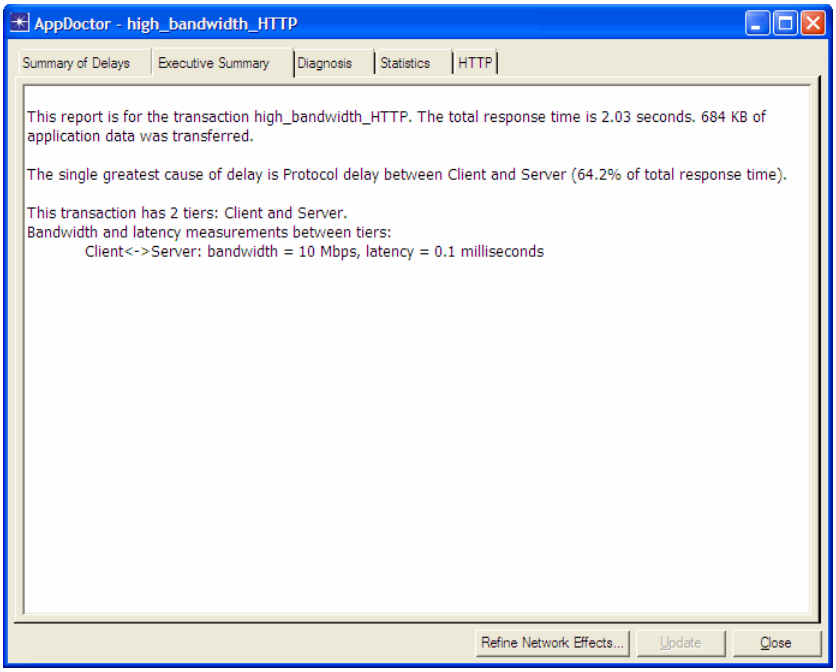
Related Topics

- Protocol Decodes

CHAPTER 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 130 Executive Summary Tabbed Page in the AppDoctor Window



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

- Executive Summary page (Figure 130) 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:

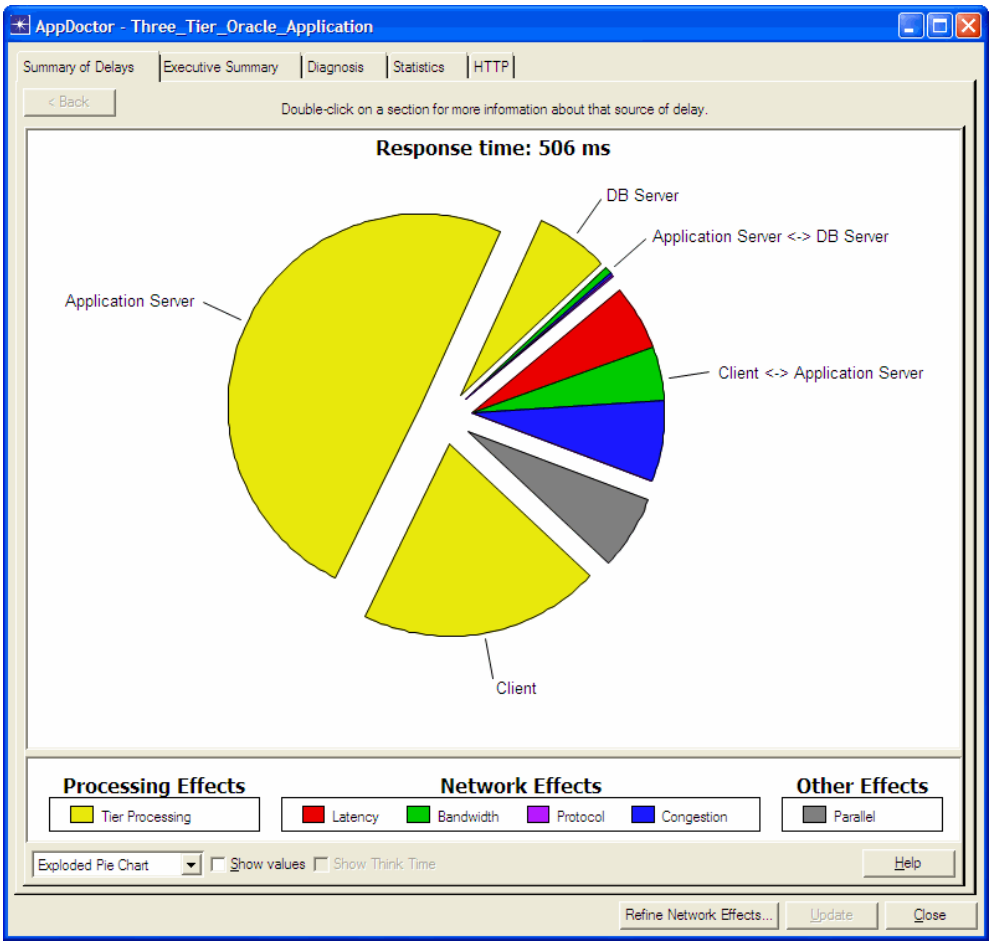
- Processing Effects—Application processing at each tier
- "User Think Time—"—The total delay incurred whenever a client-side application waits for user input before it can proceed. If the application includes think time, this window shows the following:
  - If Show Think Time is selected, the window 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 131 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" on page 499.

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 58 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 59 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
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.

Table 59 AppDoctor Tier-Pair Statistics (Continued)

Statistic	Description
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.

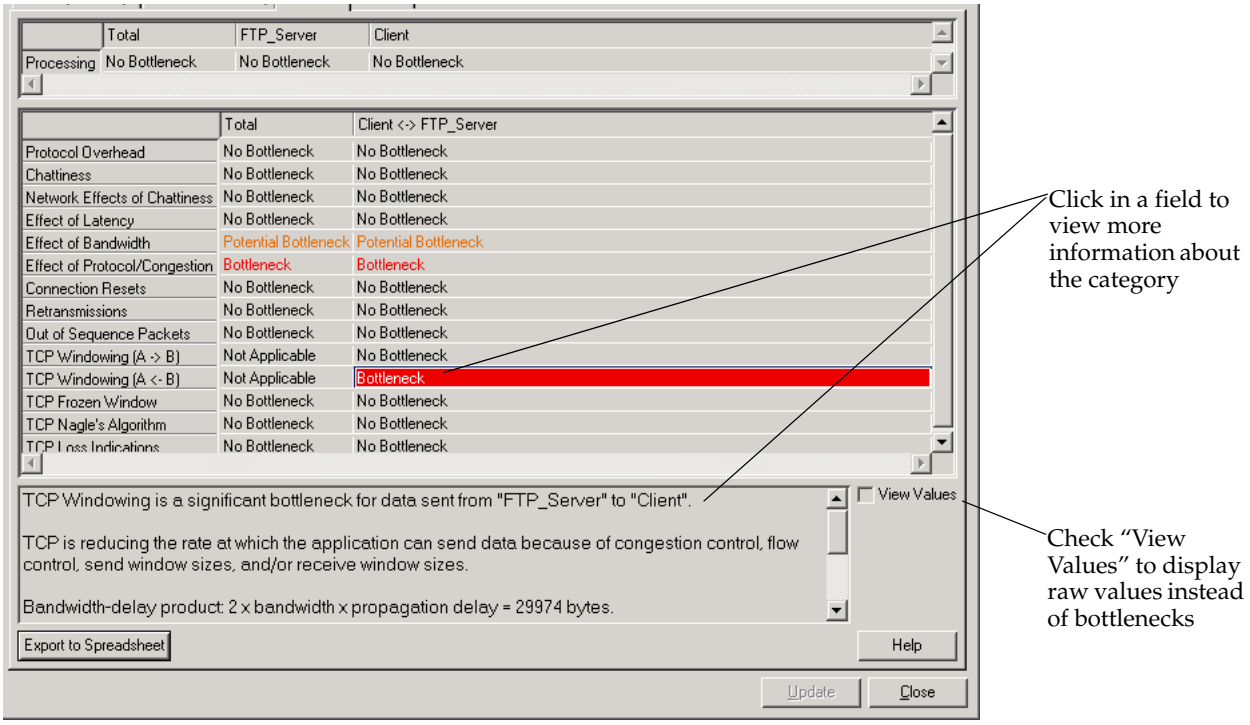
AppDoctor uses specific thresholds to determine bottlenecks and potential bottlenecks. To change the default threshold levels, choose Edit > Preferences to open the Edit Preferences dialog box; then search on the string “app”. AppTransaction Xpert has threshold preferences for each AppDoctor diagnosis 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 132 Diagnosis Tabbed Page in AppDoctor Window



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 60 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.
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 acknowledgment 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.

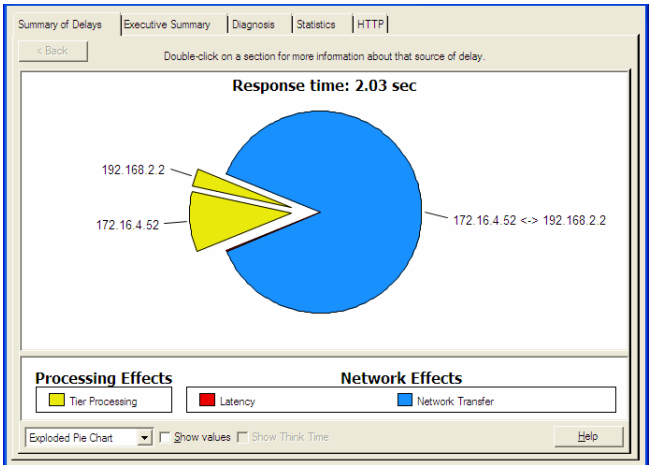
Explanation

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.

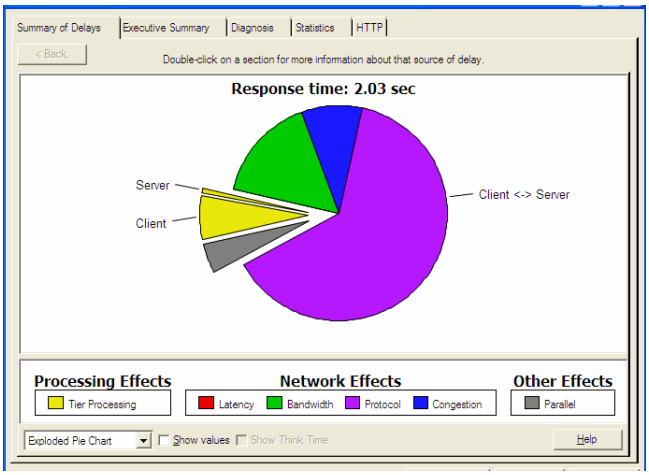
Figure 133 Example: Defining Bandwidth for Detailed Network Analysis Results

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 Acknowledgments 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