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 18-1 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 18-2 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 18-3 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	

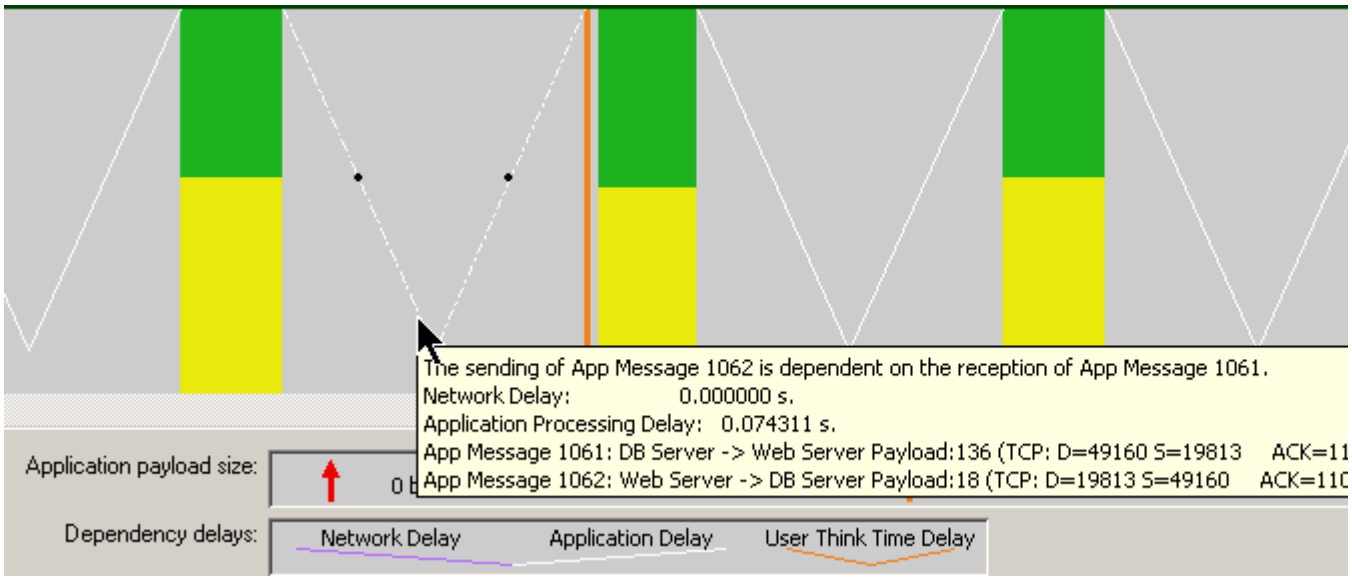
Table 18-4 Application Task Delays (Total per Transaction Analyzer Model File)

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 18-1 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

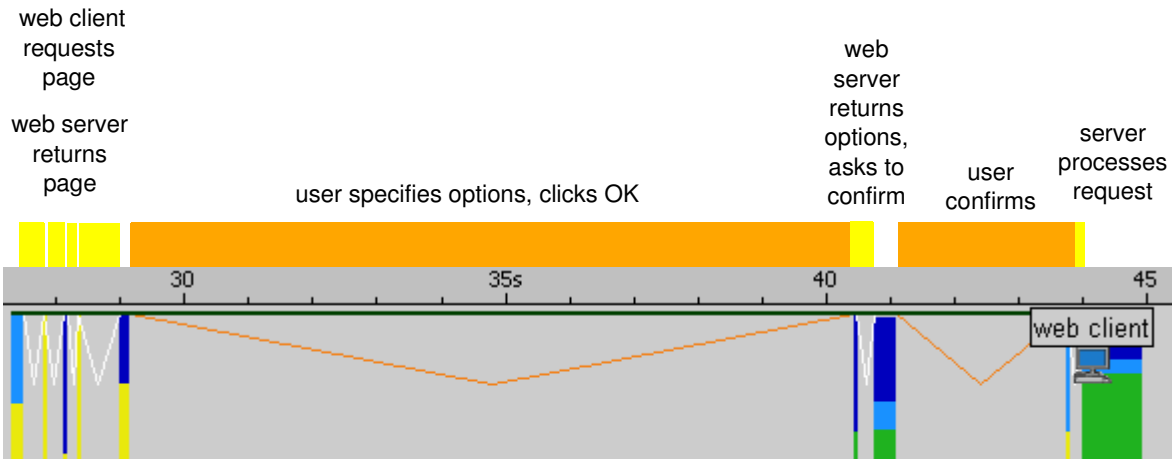
In the context of AppTransaction Xpert, *user think time* refers to the delay incurred when a client-side application requires user input before it can proceed. One complete user-level transaction might consist of several user-level actions that are separated by think time. In Figure 18-2, the server 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 18-2 Example: Application Delay and User Think Time at Client Tier



The following features enable you to view user think time:

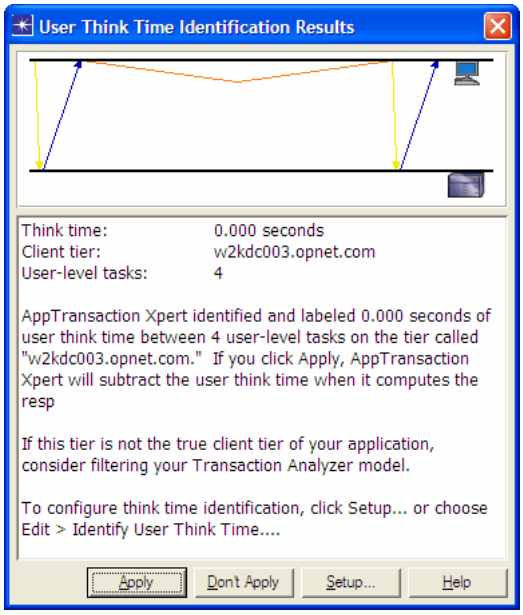
- Application Message Dependencies—Shows the User Think Time between two messages
- AppDoctor Summary of Delays—Shows the total User Think Time as a percentage of the total response time
- AppDoctor Statistics—Shows the total User Think Time at the client tier, in seconds

- 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 18-3 "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:

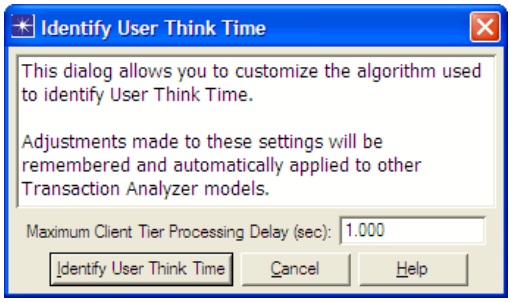
- Identifying Possible User Think Time Automatically
- Identify User Think Time (One Client-Side Dependency)

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 18-4 "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 18-5 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 18-1 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 18-1

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
- Application Example 1: Serial Request/Responses
- Application Example 2: Parallel Request/Responses
- Example 2: Conclusion
- Analyzing Delays in the “Parallel Effects” Category
- Viewing the Individual Components of Parallel Effects
- Parallel Effects Category: Summary

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:

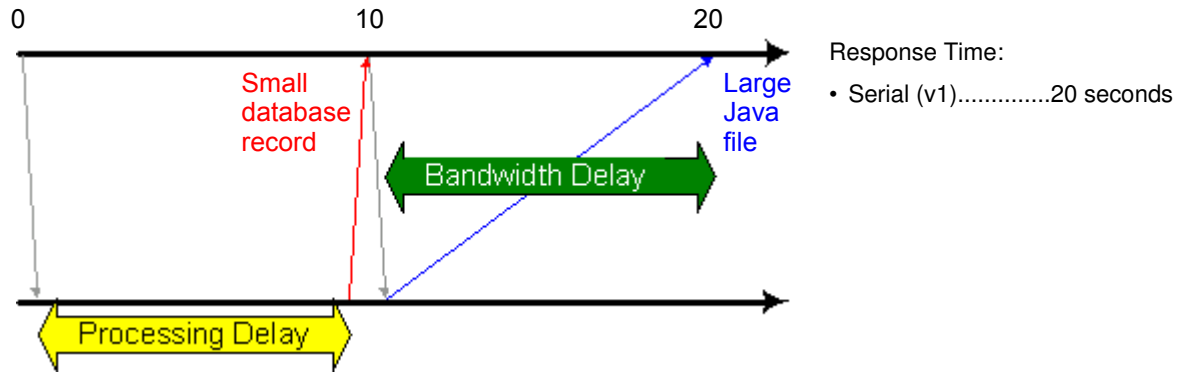
- Exclude the irrelevant traffic in AppTransaction Xpert
- Re-import the packet traces and filter the traffic more carefully during the import process.

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.

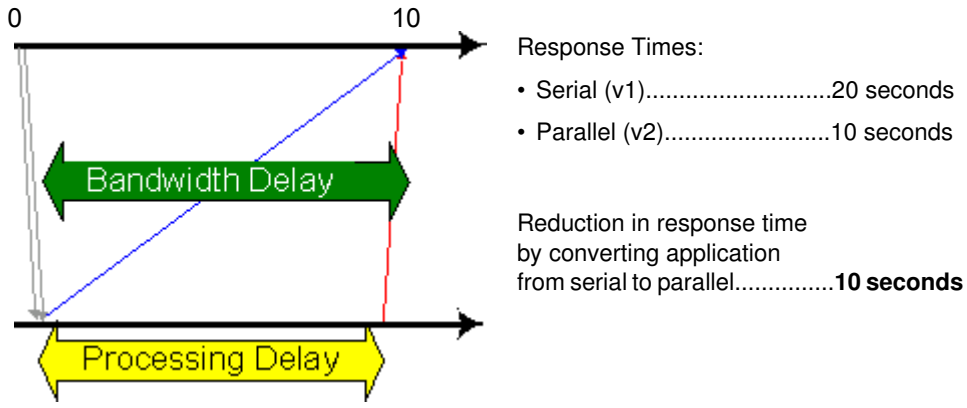
Figure 18-6 Serial Application: Processing Delay, then Bandwidth Delay



Application Example 2: Parallel Request/Responses

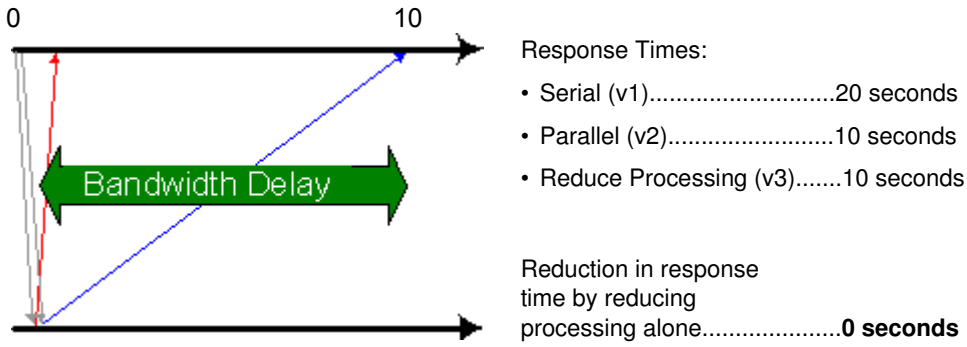
Suppose the application developer rewrites the application so that the client sends both requests at once. Now there are two different types of delay occurring at once. Clearly it would be incorrect to assign the application delay to one effect or another; in this example, it is a mixture of two simultaneous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

Figure 18-7 Parallel Application: Processing and Bandwidth Delay



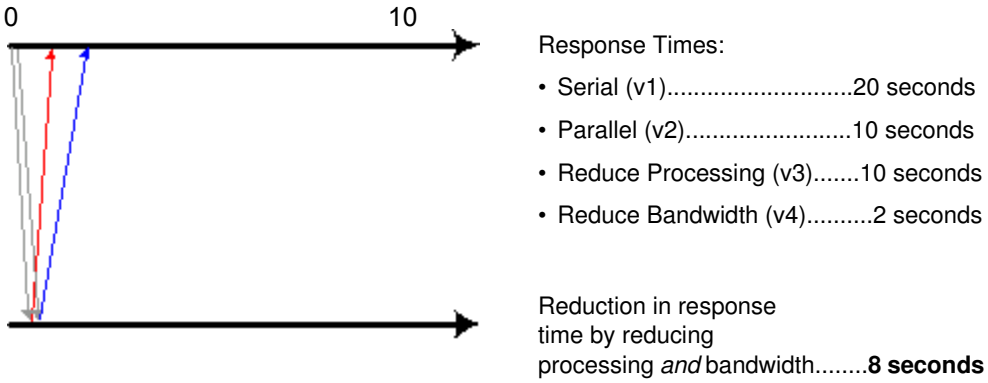
Suppose the developer optimizes the server code so that it transmits the small database record almost immediately. Processing delay is nearly zero, but the response time is unchanged. Why? Because the previous version had two delays occurring in parallel, but only one delay was reduced. The large file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

Figure 18-8 Parallel Application: Processing Delay Reduced



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 18-9 Parallel Application: Bandwidth Delay Reduced



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

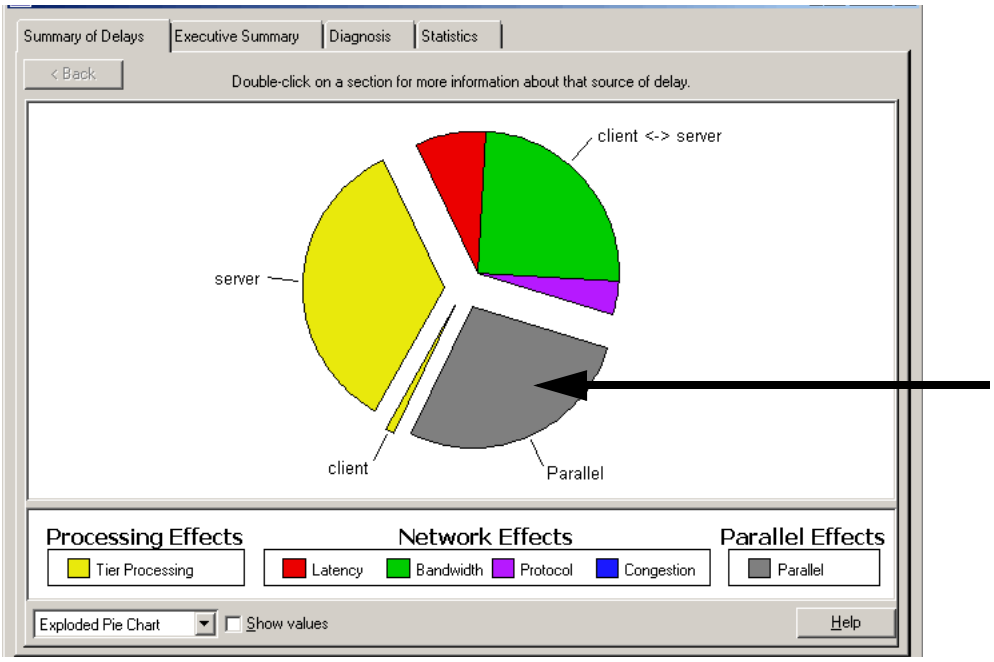
When AppDoctor finds multiple, simultaneous delays in a Transaction Analyzer model, it assigns the time incurred to the Parallel Effects delay category. Two features are especially useful for analyzing these simultaneous delays:

- 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

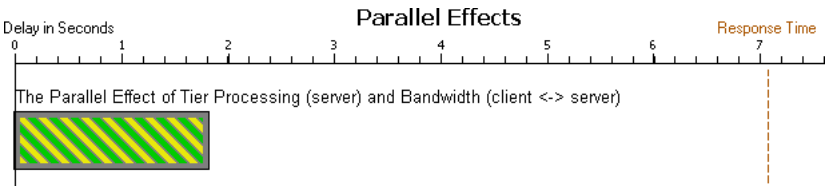
If the AppDoctor Summary of Delays shows a significant amount of delay in the Parallel Effects category, double-click on the gray section of the pie chart to display the Parallel Effects drill-down page (Figure 18-12).

Figure 18-10 AppDoctor Summary of Delays with Parallel Effects



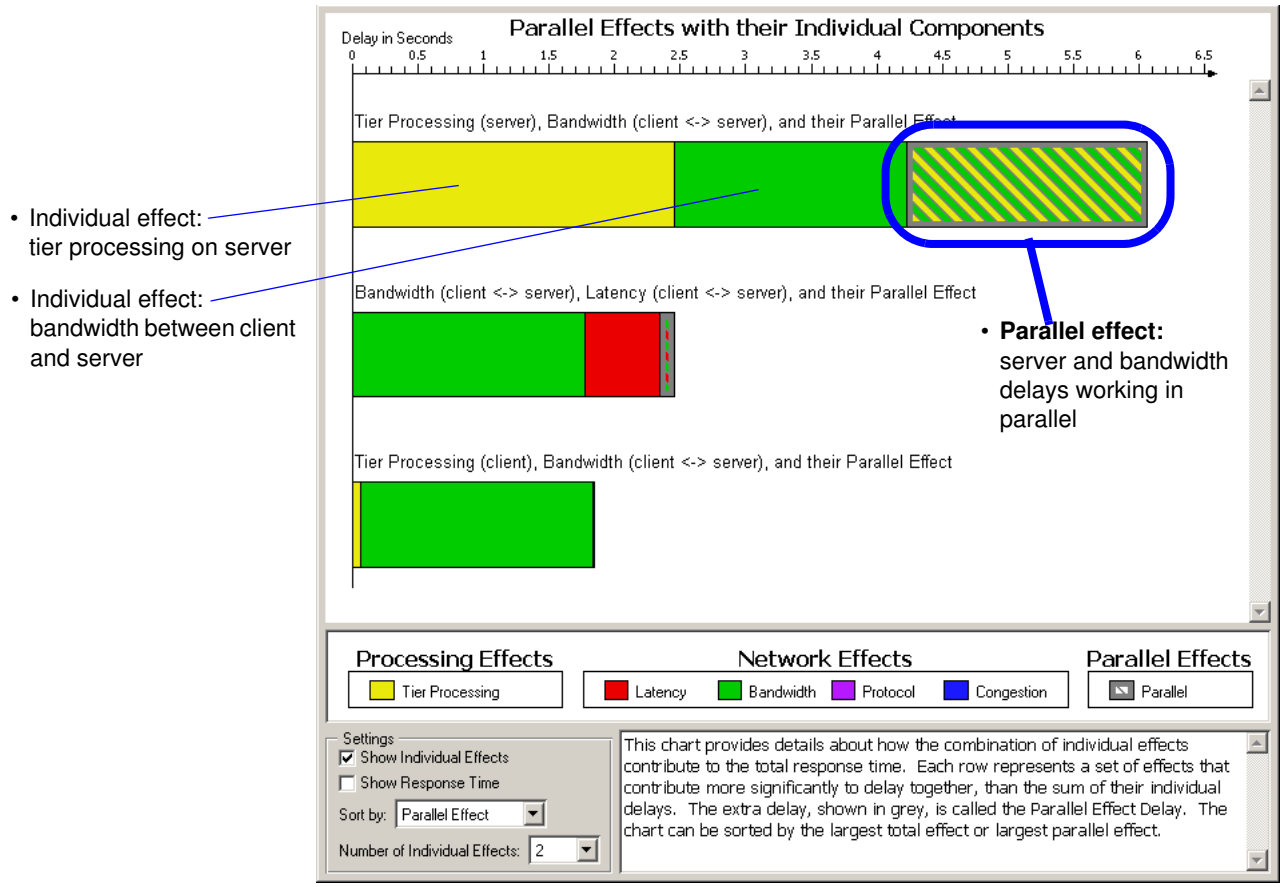
In the following figure, the graph shows that processing delay and bandwidth are working in parallel, and that this “parallel effect” is responsible for ~1.8 seconds of delay (out of a total response time of ~7.1 seconds).

Figure 18-11 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 18-12 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 18-11), 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.)