21 Troubleshooting with Patterns

When troubleshooting application issues with AppTransaction Xpert, you may notice patterns when viewing application data in data exchange charts, tree views, statistic graphs, and simulation results. The patterns can be found in many sources, including repetitive delays, unexpected delays, packet sizes, TCP connections, and network throughput. Identifying patterns can help to provide clues to resolving application issues.

Look for patterns. When you see a pattern, ask yourself:

- Why does the pattern exist?
- Why was the pattern broken?

The answer often points to a potential application problem.

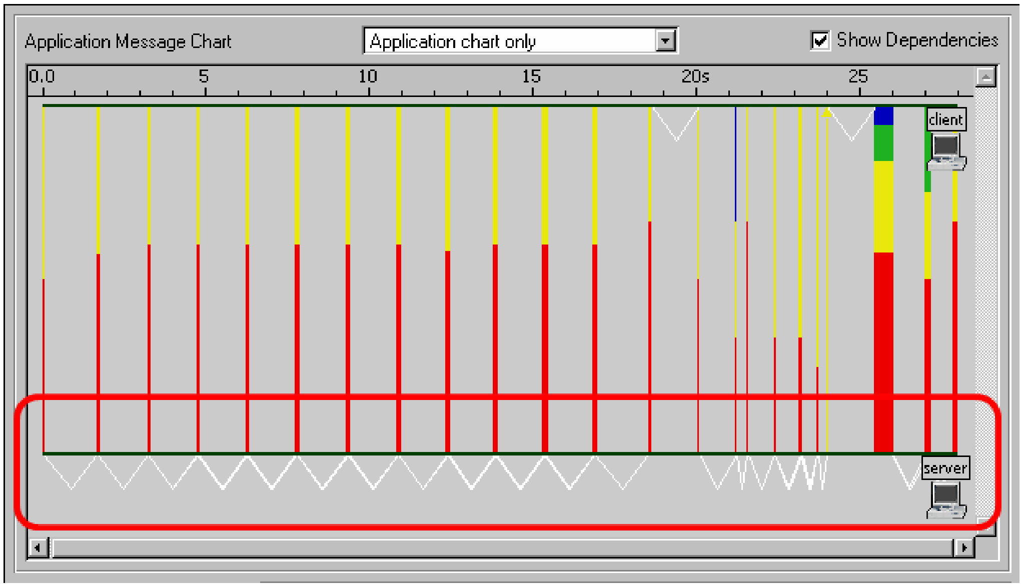
Important—Don't get carried away. Not all problems manifest themselves as patterns. And, not all patterns indicate a problem. But, if you see a pattern, investigate it.

Review the following sample patterns identified by AppTransaction Xpert users:

- Pattern #1: Server Delays
- Pattern #2: Network Delays
- Pattern #3: Small TCP Connections
- Pattern #4: TCP "In-Flight" Data

Pattern #1: Server Delays

Issue:
In the following Data Exchange Chart, notice the pattern of “application delay” at the server. Each delay is approximately 1.5 seconds.

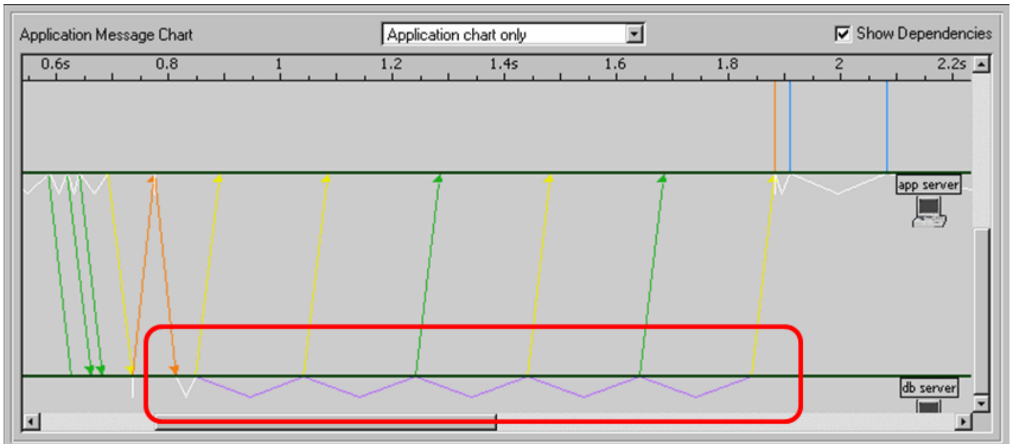


Solution:
An application timer turned out to be the primary source of delay. A timer was set to poll a request buffer 2,400 times per hour; requests could only be serviced every 1.5 seconds.

Pattern #2: Network Delays

Issue:

In the following Data Exchange Chart, notice the pattern of “network delay” at the database. Each delay is about 200 ms.



Application Message Chart

Application chart only

☑ Show Dependencies

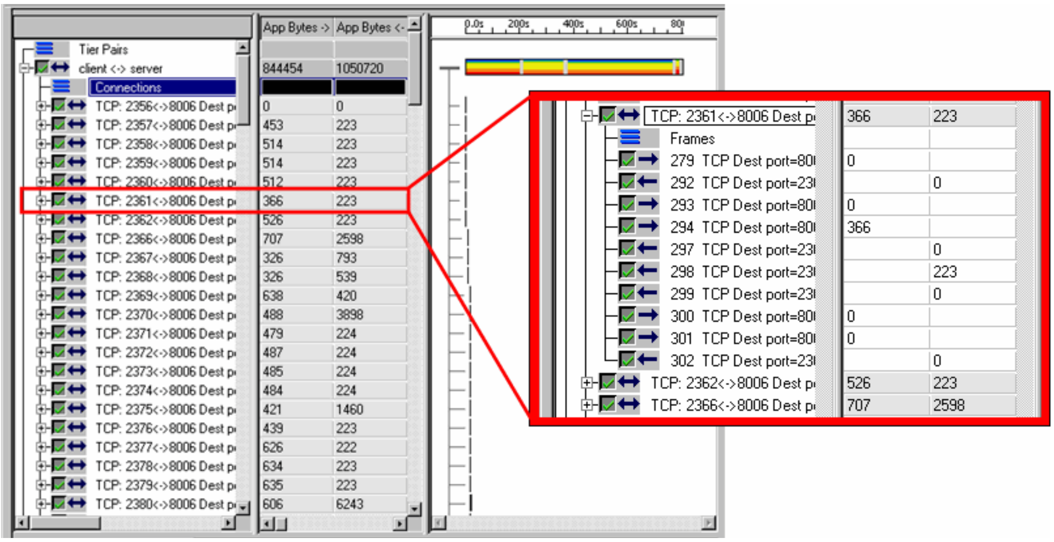
Cause:

A clue here is that 200 ms is a common value for the TCP Delayed ACK timer. Notice the small messages surrounding the delays. This delay is caused by the interaction between Nagle’s Algorithm and TCP delayed ACK.

Pattern #3: Small TCP Connections

Issue:

The following Tree View shows an application that is very slow on the WAN. Notice the small TCP connections.



	App Bytes ->	App Bytes <-
Tier Pairs		
client <-> server	844454	1050720
Connections		
TCP: 2356<->8006 Dest p	0	0
TCP: 2357<->8006 Dest p	453	223
TCP: 2358<->8006 Dest p	514	223
TCP: 2359<->8006 Dest p	514	223
TCP: 2360<->8006 Dest p	512	223
TCP: 2361<->8006 Dest p	366	223
TCP: 2362<->8006 Dest p	526	223
TCP: 2366<->8006 Dest p	707	2598
TCP: 2367<->8006 Dest p	326	793
TCP: 2368<->8006 Dest p	326	539
TCP: 2369<->8006 Dest p	638	420
TCP: 2370<->8006 Dest p	488	3898
TCP: 2371<->8006 Dest p	479	224
TCP: 2372<->8006 Dest p	487	224
TCP: 2373<->8006 Dest p	485	224
TCP: 2374<->8006 Dest p	484	224
TCP: 2375<->8006 Dest p	421	1460
TCP: 2376<->8006 Dest p	439	223
TCP: 2377<->8006 Dest p	626	222
TCP: 2378<->8006 Dest p	634	223
TCP: 2379<->8006 Dest p	635	223
TCP: 2380<->8006 Dest p	606	6243

TCP: 2361<->8006 Dest p	366	223
Frames		
→ 279 TCP Dest port=80	0	
← 292 TCP Dest port=23		0
→ 293 TCP Dest port=80	0	
→ 294 TCP Dest port=80	366	
← 297 TCP Dest port=23		0
← 298 TCP Dest port=23		223
← 299 TCP Dest port=23		0
→ 300 TCP Dest port=80	0	
→ 301 TCP Dest port=80	0	
← 302 TCP Dest port=23		0
TCP: 2362<->8006 Dest p	526	223
TCP: 2366<->8006 Dest p	707	2598

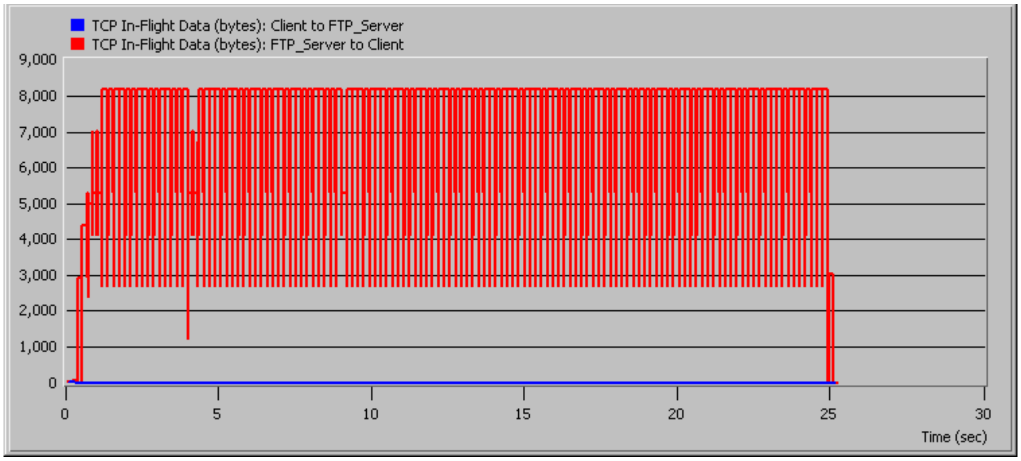
Cause:

This is the result of a bad application design. There are 306 TCP connections, most with only two data packets. This inefficient use of TCP connections adds hundreds of extra application turns.

Pattern #4: TCP “In-Flight” Data

Issue:

In the following Statistic Graph, note that the “In-Flight” data seems to plateau at 8k.



Cause:

This is a classic TCP window problem. One of the TCP windows is set to 8k, limiting throughput.