Troubleshooting OSI Layers 4–7

In this two-part white paper series, learn to quickly locate and resolve problems across the OSI layers using the Troubleshooting Cheat Sheet.

The root cause of application anomalies don't stop at Layer 3 of the Open Systems Interconnection (OSI) model. In fact, some of the most difficult to diagnose service issues are rooted in, or manifest themselves at the Transport Layer (Layer 4) or higher.

Since the Transport layer is responsible for building and maintaining sessions between devices, it serves to connect the lower layers (1-3) to the higher layers (5-7). It can also be the point of referred pain, when the real source of the problem lies elsewhere.

A methodical process to resolving service issues that reside at Layers 4-7 can clear up ambiguities and accelerate problem resolution. Mike Motta, NI University instructor and troubleshooting expert, and Tony Fortunato Senior Network Performance Specialist and Instructor with the Technology Firm, place typical user complaints into three categories:

- Slow Network
- Inability to Access Network Resources
- Application-Specific Issue

Top 3 user complaints

- Slow network
- Inability to access network resources
- Application-specific issues

Based upon the answers to the questions outlined in the following Troubleshooting Cheat Sheet, you'll gain a better understanding of the symptoms and be able to isolate the issue to the correct layer of the OSI model.

Complaint	What to Ask	What It Means
Slow Network	· What type of application is being used? Is it web-based? Is it commercial, or a homegrown application?	· Determines whether the person is accessing local or external resources.
	· How long does it take the user to copy a file from the desktop to the mapped network drive and back?	· Verifies they can send data across the network to a server, and allows you to evaluate the speed and response of the DNS server.
	· How long does it take to ping the server of interest?	· Validates they can ping the server and obtain the response time.
	· If the time is slow for a local server, how many hops are needed to reach the server?	· Confirms the number of hops taking place. Look at switch and server port connections, speed to the client, and any errors.
Inability to Access Network Resources	· What task is the user attempting to perform?	· Indicates whether the action is limited to a specific resource such as a mapped drive on a server or multiple network resources.
	· What type of application is the user attempting to access?	· Similar to the above question on application type, this may point to a problem with multiple internal servers.
Application-Specific Issues	· What's the 3-way handshake time?	· Identifies potential points where a slowdown might be occurring.
	· What's the server processing time?	· Points to whether the server is taking too long to process data.
	· How much data are you pulling from the application and sending across the network?	· Assesses whether the application is sending data in an expected and efficient way.

For additional information on application-specific issues, check out Mike Motta's Tech Tips.

Continuing Up the Stack: The Second 4 Layers

With these questions answered, working through the OSI model is a straightforward process. With the exception of Layer 1, each layer of the OSI model relies on the next lower layer to provide services as specified. Requests drop down and are completed, as every layer interacts with the next layer, both above and below.

When dealing with different layers, understanding how each delivers data and functions impacts how you will troubleshoot.

Layer Highlights and Functions

Transport Layer

- End-to-end connection and connectionless data delivery management
- Ensures reliable packet delivery caused by network congestion & errors
- Congestion avoidance and data transmission flow control

Note: The majority of the functions below for Layers 5-7 are logically combined into the "Application Layer" for the purposes of discussion in this paper. Some capabilities, the application connectivity of the session layer in particular, can be thought of as residing within Layer 4.

Session Layer

- Establishes, manages, and terminates application connections
- Manages data transfer permissions and records upper layer errors

Presentation Layer

- Application, system, and network independent data formatting
- Data conversion, compression, encryption, and decryption

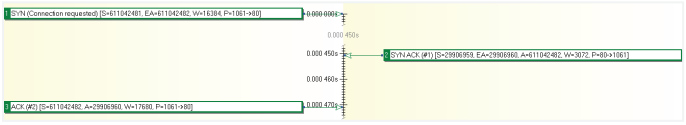
Application Layer

- Supports application and end-user processes
- Offers application services
- User identification, authentication, and privacy

The TCP three-way handshake is the foundation on which the application session is built.

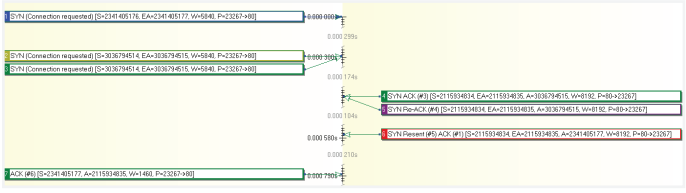
Steps to Troubleshooting Success: Transport Layer Checklist

1. **Is the TCP three-way handshake successful?**
   The TCP three-way handshake is the foundation on which the application session is built. "Without it, your service is DOA before it even gets out of the gate," says Mike Motta.



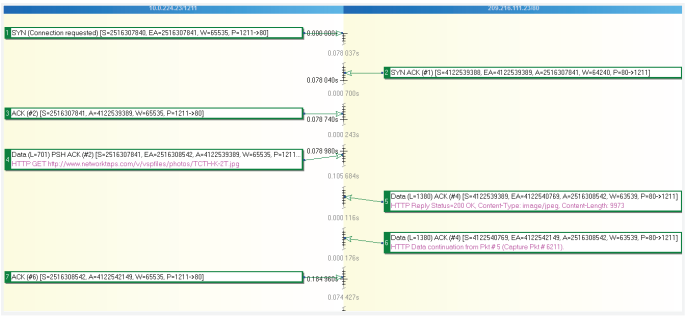
At this point the session client-server connection is complete and the application layer activities can begin. Occasionally, the session request is delayed or fails as the application socket or port is too busy to support it (or if there are lower layer network issues). In this case, rather than responding with a SYN-ACK, the server will reply with an RST (or simply ignore the SYN).

Below is an example of an abnormal three-way handshake. You can see the client needs to say "Are you there?" (Transmitting a SYN) three times before the server finally responds (with a SYN-ACK). Definitely not a good way to start a conversation (and if seen repeatedly should be investigated).



Assuming the three-way handshake is successful, Motta states that, "One important piece of data you can glean is the total network roundtrip time. It's measured from the SYN to the ACK and is useful for assessing the ability of the underlying network to service devices, as it does not include any application processing overhead. Think of it as the "best time" you can expect to attain from a responsiveness standpoint between client and server. Given this, it can be useful in measuring the infrastructure's ability to support lower-latency applications."

Client application requests can then begin, below is an example with HTTP:

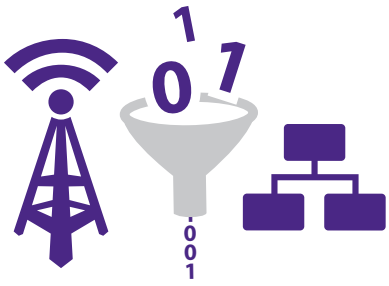


2. Is TCP Repeating Itself Too Often?

The Pain of Excessive Retransmissions

It's important to emphasize here that as a connection-based transport, TCP is a highly robust protocol that can tolerate the re-sending of packets. In fact, even the healthiest network will drop some packets.

TCP is also great at shielding the application from this activity at low levels of packet loss, unless it becomes excessive. The potential root causes of too many retransmissions are varied, from problems with TCP itself (e.g. checksum or sequence number generation errors) to degraded physical connectivity (very common, due to bad cabling or switch port, etc...).



Did You Know:

The three top communication disruptors are excessive retransmissions, flow control issues, and congestion.

Retransmissions: Real or Fast?

"Retransmissions come in two varieties," says Tony Fortunato.

"The first I will refer to as Real. In this case the TCP timeout value is reached before the data is received and the packet must be retransmitted (after considerable delay) which may impact app performance."

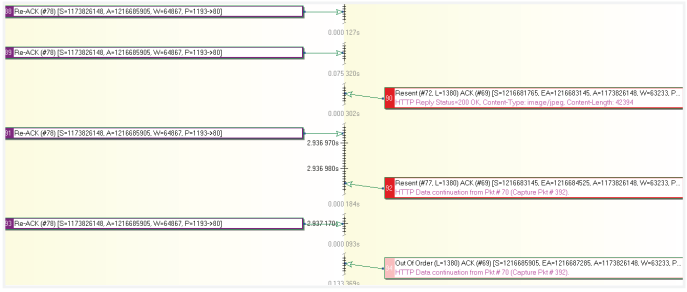
"The second is a Fast retransmission. In this scenario, the receiving TCP stack continues to resend duplicate ACKs to the sender for the last contiguous sequenced packet number received as each new out-of-order packet arrives. This is much better as the sender can retransmit the packet that is assumed lost without waiting for the timeout to occur. This usually minimizes the impact to the app and vividly illustrates the power of TCP as it sorts out an occasional lost packet," says Fortunato.

Assuming neither of these are an issue, the next most frequent causes are an overloaded link or TCP server Stack Busy condition.

The former is caused when a switch or router is simply overwhelmed with the amount of traffic passing through it. At some point the deluge exceeds the processing capability and packets must be discarded. Likewise, if the application server workload exceeds the processing capability, its TCP stack will ignore or delay a client request. This will be viewed by the client as packets dropped or lost, and a retransmission request will be initiated.

The solution in both cases is to reduce the load to the devices and/or increase the switch/router or server processing performance to support the heightened workloads.

Below is an example of multiple retransmissions for a web session:



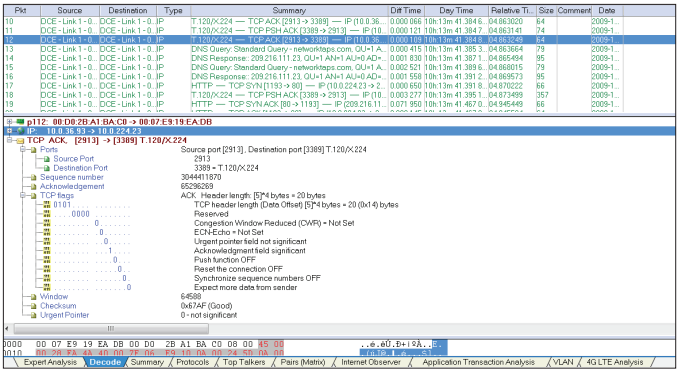
There will always be some lost packets (and hence retransmissions). As long as these are not excessive, other frequent TCP issues are probably related to flow and congestion control.

3. Is TCP Transmitting Too Slowly?

Incorrect Flow Control Can Throttle App Performance

Incorrect window size can have a significant impact on application performance. Window size is the end-to-end TCP flow control protocol utilized to ensure the sender does not transmit data faster than the receiver can receive and process. Its value is expressed in bytes.

The graphic below shows that the receiving device's window size is 64,588 bytes. The window size should be large enough to adequately deliver sufficient amounts of data to the service or application for acceptable user performance without exceeding the receiving TCP packet's buffering capacity (otherwise packets will be discarded and need to be re-transmitted).



Once the sender reaches the maximum amount of data advertised in the window size, it must wait for an ACK with an updated window size from the receiver before proceeding with more data.

Generally, assuming a reasonable value, window sizing is not an issue on a local network. However, higher levels of network latencies—often associated with poor WAN performance—can starve an app.

Care and Feeding of Applications

Considering the round-trip time of the network, whenever a sender transmits data and the window size reaches zero, it must stop and wait for its data to:

- Traverse the network
- Reach the receiver
- Get the receiver's ACK (with an updated window size)

As the travel time grows longer and larger via latency, applications can be left waiting for TCP to request more data. The solution here can vary. If possible and the application can support it, simply increase the window size.

You can also re-architect the application host deployment so as to reduce the latency between tiers. Of course, lowering network latency is another way to solve the issue if your budget supports the associated costs in higher WAN speeds and/or faster infrastructure devices.

Closely related to window sizing are the topics of chatty apps and application read/write buffer sizing.

3 ways to resolve latency

- Increase the window size
- Re-architect app deployment
- Upgrade infrastructure

4. Is TCP Experiencing Roadblocks?

– Congestion Control Can Point to the Solution

Network congestion can significantly degrade performance. Fortunately, recent additions to the TCP standards revolve around congestion control. As network complexity and speeds continue to grow, the ability of TCP to effectively manage congestion has increased.

There are four algorithms that are used simultaneously:

- Slow-start
- Congestion avoidance
- Fast retransmit
- Fast recovery.

If the previously discussed concepts fail to improve performance and assuming you suspect the issue remains within the transport layer please refer to RFC 5681 for additional details.

Optimize TCP Performance

Fortunato and Motta offer three frequently overlooked ways to improve your TCP transport layer performance.

Segment Size

The optimal segment size (not counting TCP header and packet overhead) is 1460 bytes (assuming 10/100 Ethernet). Motta calls this out as a potential waste of network resources because sending out lots of small packets exacerbates the connection-orientated nature of TCP (which in MS Windows requires an ACK for every two packets or 200ms).

"I constantly remind my clients that if their app can support it, this is the segment size that best leverages their network assets and offers a great way to improve service performance," says Motta.

Windows Scaling

Since the window size control field is limited to no more than 65,535 bytes, a TCP window scale option (defined in RF 1323) can be used to increase the maximum window size up to a gigabyte. This is a great way to significantly increase TCP throughput.

"However, before doing this be sure to confirm your devices can support the added required buffering, otherwise it can corrupt your drivers," says Fortunato. "Also, your app must be able to handle the increased amount of possible incoming data."

Selective Acknowledgement

The TCP protocol as initially designed can lead to inefficiencies because of the cumulative ACK scheme. Basically this means that a receiver is unable to say it received later data if it failed to receive earlier bytes. Thousands of bytes may be received but if the first 1,000 bytes is missing, all the data may have to be re-sent.

RFC 2018 defines an optional selective acknowledgement (SACK), enabling the receiver to ACK discontinuous chunks of packets that were successfully received. The receiver communicates the beginning and end of a contiguous range of packets that it has (via the sequence numbers), allowing the sender to simply resend the lost packets. "I've worked with many clients that have not implemented this great TCP feature," says Fortunato. "To me it's a simple way to improve overall service performance, especially since even the most robust networks will drop packets."

Application Layers

At this point in the debug process, you've hopefully eliminated any Layers 1 - 4 issues. Due to the complexity and number of applications in a modern data center, it is important to realize that the concepts provided are limited to basic fundamentals.

HTTP, a protocol used for many application front-ends, and of course web-based traffic is used for illustrative purposes of the process of debugging an app issue.

Fast Facts:

HTTP status codes are divided into five groups:

1XX– Informational

2XX– Success

3XX– Redirection

4XX– Client Error

5XX– Server Error

Application Read/Write Block Buffer Sizing

Block buffer sizing is an important metric that is often overlooked or not visible to the network engineer (who frequently is not familiar with app details).

The block buffer size is the maximum amount of data in bytes that the app can support queued up. This information is what's passed to the stack for use in TCP window sizing. If the value is too low, it will effectively throttle the overall app performance, potentially dramatically. What looks on the surface like a slow network—and will show up as zero window problems—is actually a pure application layer issue.

Chatty Apps

If an application requires packet sizing that is appreciably lower than the optimal segment size mentioned above (1460 bytes), it is considered chatty. After multiple clients complained about degraded performance, even after checking out their networks and finding them fine, Motta explained the effects of the chatty applications.

"When clients re-architect their application deployments, applications that once worked fine when hosted all in-house (with low latency) suddenly act up when increased network latencies (associated with WAN links) bring the application's low payload limits to the surface," Motta says.

Motta recommends updating the app to support larger segments. If this is not an option, reduce the network latency to levels that will enable the service to run within acceptable user performance parameters.

Status Codes

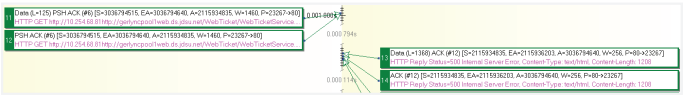
Digging into application-specific issues requires the ability to perform payload-level analysis to assess exactly how an app is responding to users' requests. Depending on the service, reason, error, status, or condition, codes can be captured and translated to infer how the application is performing. There are a number of performance monitoring vendors that offer these capabilities.

Check out a comprehensive list to see individual values.

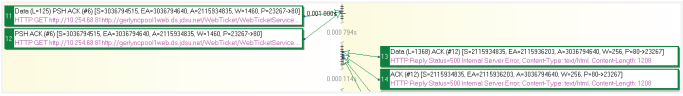
Here are the most common:

Status Code	What It Means	What to Do About It
200 Ok	A successful HTTP request, your world would be happier if this was the only code you ever saw	Nothing
304 Not Modified	The client already has the information (which has not changed, hence the web server does not need to retransmit; this is also a good response	Nothing
404 Not Found	The user has requested a URL that does not exist	Nothing, unless you see too many of them to the same URL, then it could be someone attempting to gain access either innocently (e.g. perhaps an incorrect URL was provided customers?) or for nefarious reasons. Further investigation may be merited
500 Internal Server Error	Something bad has happened inside the web server, "Houston we have a problem" with a system	Get in touch with the web server team if more than a few of these are noted as it could be an early warning of underlying problems

Below are examples of web server responses. The first is the expected 200 OK. This means all is well.



On the other hand, as you can see in the diagram below, a 500 Internal Server Error is bad news.



As long as you primarily see 200 OKs (and of course assuming acceptable Layers 1 - 4 health), your web-based service should be delivering acceptable user performance. The one caveat is that you will want to setup adequate baseline response times to ensure that data is received in sufficient time to maintain high-levels of customer satisfaction.

While most applications use similar type messaging to communicate their ability to effectively service clients' requests, the format and meanings will be unique. The key is to work with your application teams to understand what these various codes are, and when errors are detected to make the specific group aware so the issue can be quickly resolved.

Conclusion

The life of the network engineer or administrator gets more interesting and challenging every day. As the first responder, network staff will often play a pivotal role in fixing the problem. What that means is they will need to have a solid understanding of all layers of the OSI model, correcting those issues that reside within the network and assisting the application or systems teams when possible.

Also, modern applications and the more complex hosting strategies that distribute app tiers around the world make detailed transport and application layer awareness ever-more important. Use the methodologies and suggestions in this paper as a starting point to ensure that your network resources are ready to support today's modern apps and solve problems when they occur.

Contact Us **+1 844 GO VIAVI**
(+1 844 468 4284)

To reach the Viavi office nearest you,
visit viavisolutions.com/contacts.

troubleshootingosilayers4–7-wp-ec-ae
30176217 901 0315