pathSolutions

WHITE PAPER

Root-Cause Network Troubleshooting

To truly fix a problem, you have to know the root-cause of it

(877) 748-1777
Sales@PathSolutions.com
www.PathSolutions.com

Contents

(877) 748-1777
Sales@PathSolutions.com
www.PathSolutions.com

Why is Troubleshooting So Hard?

When it comes to troubleshooting network problems, you'll need more than monitoring solutions to get the job done. Most organizations have monitoring software that is designed to monitor the condition of their network environment. For example: Generate an alert if there are outages or resource constraints like high CPU utilization on a virtual machine or low disk space on a disk array. Or monitor utilization on WAN links.

With monitoring software, you'll know that everything in your environment is reachable. But you won't know if packets are being dropped or delayed anywhere in the network. In this case, troubleshooting is very difficult as engineers must manually log into switches and routers in the infrastructure to determine *where* and *why* a problem is happening. And if engineers don't have specific knowledge of how the network is built, *finding* and *resolving* the exact cause of a problem can take hours, days, or weeks. And if the problem is intermittent, it may never be resolved.

What's missing from this is the ability to *troubleshoot* the *root-cause* of network problems.

All engineers engage in troubleshooting but getting to the root-cause of a problem is key. This is a different activity and requires different information to achieve its goal. Organizations that rely solely on monitoring software end up having problems when it comes to troubleshooting issues in their environment.

This White Paper details a troubleshooting methodology, applies it to three specific case studies, and considers how you can optimize the troubleshooting process.

Troubleshooting Methodology Key to Resolving Problems

To truly fix a problem, you have to know the root-cause of it. Instead of relying on assumptions, implementing a good troubleshooting methodology is required. Follow these steps to ensure that problems are efficiently solved:

1) **Collect Information:** This is where the majority of troubleshooting time should be spent, as it's the "clue discovery" part of the process. The more valid information brought to bear, the more accurate the results. Typically, not enough time is spent collecting the right information, and this leads to a lot of guessing and assumptions—not the best way to operate when troubleshooting critical problems.
2) **Develop a Hypothesis:** Based on the data collected, develop a hypothesis. Again, if there is difficulty in creating a hypothesis, go back to step #1 and collect more information.

3) **Test the Hypothesis:** For each conclusion, develop a test to determine if the hypothesis is correct or not. If it is not correct, go back to step #1 and collect more information.

4) **Implement the Fix:** Once you have tested your hypothesis under the same conditions as when the problem first occurred based on the information you collected, schedule the fix with the user community. This may require downtime that has to be scheduled after hours, or it may be something that must be immediately implemented.

5) **Verify that the Problem was Solved:** Make sure that you have solved the original problem. But you must also verify you have not introduced other problems due to the fix. Making one set of users happy at the cost of another set is rarely a good tradeoff.

6) **Notify Users:** If you don't notify the users, they won't know that it was fixed. Seems simple, but if this step is missed, you may have large groups of idle users because they are waiting for notification of the fix.

7) **Document the Fix:** In a time-constrained environment, this is another overlooked area. It's important that you document what happened, what the root-cause problem was, and how it was fixed. Often, you may come across similar problems, and this documentation may speed resolution by others on the team, as well as be an input to the information collection process.

Troubleshooting is an iterative process—the more data you collect and analyze, the higher the likelihood of developing a correct hypothesis. Monitoring software may contribute one or two elements to the picture, but there are many other sources of information that should be evaluated before developing and testing a hypothesis. The following case studies explore how a troubleshooting methodology can be employed to resolve some common and uncommon network problems.

Case Study #1: Virtual Machine Weirdness

The server team has a strange problem that's affecting their ability to scale appropriately and is also causing downtime for their services:

- Anytime they move certain virtual servers to a remote data center, the server move fails, and the virtual machine crashes.
- Also, there are significant unused server resources in the remote data center and overutilized server resources in the local data center that are causing customer slowdowns.

Many organizations would see this as a problem for the server team, *not* the network team. After all, the monitoring software indicates that the network is fully operational, and some virtual machines CAN be moved between the data centers without a problem. What's causing this—is it a server issue or could it be the network?

The Culprit

It turns out that the network is the culprit as the monitoring software is not aware that a couple of interfaces in the remote data center are dropping a significant number of FrameTooLarge packets because the MTU on the interfaces is not set properly. This is what the monitoring software doesn't know:

- When a virtual machine is moved, if it has a local virtual layer-2 network, it can operate completely on its own and be fully functional on a local hypervisor or a remote hypervisor.
- If the virtual machine is moved and it has a virtual layer-2 network or virtual layer-3 network that is shared between the data centers, it requires the MTU for all involved interfaces to be large enough to handle the encapsulated traffic. If the MTU is set at the default 1500 bytes, these packets will be dropped, and the server move will fail.

In this case, the root-cause of the server problem lies with the network—if the network team was aware of it, they could be the hero of the day, solving the problem in just a few minutes.

The Solution

While this is not a common problem with all networks, it can occur when virtual machines are configured with a specific configuration. To solve it, you would investigate all involved interfaces for the correct MTU size. This is not too difficult to do if you know what to look for. The problem is that it's just one variable amongst hundreds that would need to be investigated. How do you know that you should check it?

If you have a troubleshooting solution, the way in which you would go about identifying and resolving the problem would be very different and much faster. For example, if FrameTooLarge errors are tracked on all interfaces, you could run a correlation to determine the interfaces, switches, and routers involved between the two points. Then the problem would be easy to spot as an "MTU exceeded" problem on the specific devices that were dropping the packets.

Case Study #2: Bandwidth Blues

Your network has a nice, fat 1gig WAN connection to a remote data center. But it's always over-utilized and runs at near capacity for most of the day. Why?

Your first inclination might be to see if you have a bandwidth hog and in this case, it makes sense to use your monitoring software. After doing some analysis, you determine that the usage is legitimate as it's comprised of database synchronizations and web transactions. Perhaps adding more bandwidth will alleviate the problem—but the additional bandwidth could end up costing thousands of dollars. What if extra bandwidth doesn't resolve the problem?

The Culprit

Additional bandwidth is often the go-to solution for situations like this. If you have a nice fat WAN connection, and your users are not hogging bandwidth for non-business pursuits, it could *logically* follow that more bandwidth fixes the problem. But have you considered other possibilities?

Before spending money, let's dig down into the error counters collected on the LAN switch upstream of the WAN router: 18% of the packets were being dropped on this interface. Why?

- If you were able to analyze the error counters, you would see that a layer-1 cabling problem in the LAN caused the WAN to become overloaded due to retransmits.
- A quick look at the cabling identifies the real cause of the bandwidth blues: a cable is pinched and flattened between two server cabinets.

Fixing that cable costs $5 in comparison to the thousands of dollars you would have spent on more bandwidth.

The Solution

In this case, a monitoring solution points you in the wrong direction which leads you to the wrong conclusion. No matter the bandwidth, that pinched cable would impact capacity—you are not collecting and analyzing enough data to discover the root-cause. But if FCS Errors, Alignment Errors, and Collisions were collected on all the upstream and downstream interfaces and analyzed with a heuristics engine, you would discover that you had a cabling problem on a specific switch and interface.

Case Study #3: Stuttering Video

Your company has deployed some nice high-end video conferencing units around the country, and the project is almost complete. But some of the sites are reporting bad video artifacts and audio clipping issues. Your monitoring software indicates that everything is fine on the network. What's going on?

If everything is fine on the network, you might jump to the conclusion that the WAN queues on the MPLS network are misconfigured and are causing problems which leads you to have a heated discussion with your WAN provider.

The Culprit

Here's the issue: Your monitoring solution has not provided enough knowledge about the network to solve the problem.

But if the configuration and error counters were collected on the switch connection where the video conference units were connected, you could see that a few of the sites had duplex mismatches and were dropping 10% of their packets. A quick configuration change to make them run at full-duplex on both ends of the conversation resolves the issue in less than five minutes.

The Solution

This is just one example how duplex mismatches can cause network problems. The goal should be to correct all of them to prevent problems like this from occurring. With monitoring software alone, this is impossible to do because they do not collect and analyze the "right" information. However, if interface configurations as well as FCS Errors, Alignment Errors, and Collisions were collected on all involved interfaces and analyzed, duplex mismatches could be automatically identified as well how to fix them. All of your video conferencing problems could be solved system-wide *within 15 minutes.*

Optimizing Your Troubleshooting Methodology

These case studies are examples of how problems can be identified and resolved faster when the appropriate criteria are evaluated: Collecting the right type of information, within the right timeframe, with the right correlation, and then have analysis that speeds understanding of the situation.

In almost all cases, timeframe is critical: If you knew what your network was doing at the time of the event, problems could be solved in minutes, rather than hours or days. When employing a troubleshooting methodology, most of your time is spent collecting and analyzing information, so optimizing the collection of the information is the key to speeding *understanding* of the situation. Finally, being able to analyze and correlate that information speeds *resolution* of the problem.

Right Types of Information

There are a number of different types of information that should be brought into the troubleshooting picture. Logfiles, netflow records, packet captures, error counters, configuration information, utilization information, are all possible sources that can be tapped, but some have drawbacks as well:

- Logfiles may be of limited value since they are generated typically for device-centric faults like reboots, power supply failures, and memory failures.
- Netflow records can't help because they are designed to show who used a link and thus have no information about packet loss or buffering problems.
- Packet captures may not be as useful because all they can do is confirm there is packet loss—there is no way to determine where or why the packets were lost.

In most cases, network faults leave traces in error counters and can be further confirmed by analyzing configuration information. For example, every packet lost or delayed (buffered) leaves a trace in a device's SNMP error counters. While incredibly valuable, it represents a significant amount of data, as the error counters need to be collected from every device and interface in the infrastructure—on a regular basis.

Timeframe Analysis

The information collected must be timeframe-specific. For example, an error counter that shows 12,000 FCS errors since the device was rebooted six months ago is not helpful because you have no idea if the 12,000 errors happened a few minutes ago when an event occurred, or 5 months ago. You must be able to collect and store the error counter information with appropriate timestamps.

Correlation

Correlation enables you to narrow down the scope of information, identifying the path through the network, or group of devices, where the problem originated. You can then narrow scope further—only looking at the involved interfaces, switches, and routers in the session.

Analysis

Instead of spending valuable time manually interpreting the meanings of specific error counters, they should be automatically analyzed to produce plain-English answers. This significantly speeds up problem resolution.

Why Monitoring Software Can't Get the Job Done

The ability to access, collect, and analyze error counter data, configuration information, and performance information is key to optimizing the troubleshooting process. It's easy to assume that monitoring software can be configured to collect SNMP error counters on devices to help with information collection. However, that is not the case as monitoring software is not designed to make this easy or even possible. Instead, you must manually:

1) Determine which error counters should be collected from network devices.
2) Determine (through trial and error) which error counters are supported by different OS versions running on these devices.
3) Configure your monitoring software to collect this information. This, alone, may take weeks to months of effort to accomplish on smaller networks. It is almost impossible to do on larger networks.

In addition, monitoring software has several drawbacks to effective troubleshooting:

- **Re-configuration.** If your network changes in that time period, a significant re-configuration effort must be undertaken.
- **Flooding.** Monitoring software tends to send a flood of individual SNMP OID packets on the network to try to collect information. This can flood network links and cause CPU spikes on network devices during the collection activity because packet optimization and device CPU protection is not employed.
- **Scalability.** Monitoring software typically can't scale to handle the amount of error counter information collected, ending up with limitations on the number of elements that can be collected from a single collector. This means that multiple collectors need to be set up to support many networks.
- **Database Expansion.** Since monitoring software is not optimized for collecting error counter information, database growth must be closely monitored. For example, the quantity of collected information can be *ten times* the previously collected information.

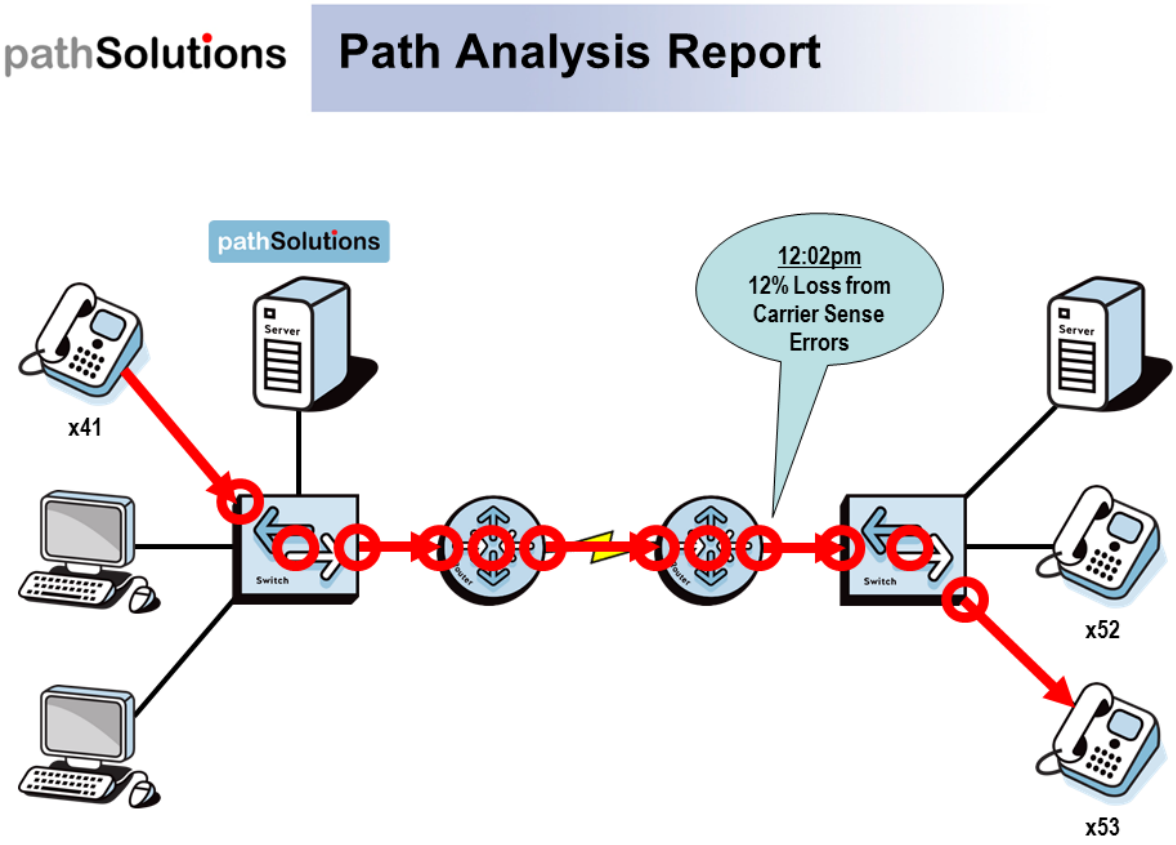
- **Lack of Correlation.** Monitoring software doesn't focus on the correlation of large amounts of data as they aren't designed to typically collect large amounts of data.
- **Lack of Analysis.** With no understanding of the information collected, coming to useful conclusions is difficult. As a result, you must research and manually interpret results and hope you reach a correct conclusion—a time consuming and resource intensive process.

It's clear that a monitoring solution, alone, cannot support a robust troubleshooting methodology.

The Key to Root-Cause Troubleshooting: Total Network Visibility®

PathSolutions TotalView optimizes the root-cause troubleshooting process and addresses the drawbacks of traditional monitoring solutions:

- It's easily deployable on any network within minutes.
- It collects SNMP error counters, configuration, and performance information on every link, switch, and router in the entire infrastructure every five minutes. The information is collected with a high-degree of packet optimization, so links don't get flooded and device CPUs aren't affected.
- It provides timeframe analysis to do evaluations of what happened when events occurred and correlates the information to identify the links, switches, and routers involved between any two endpoints.
- It has a heuristics engine that analyzes the error counters to produce plain-English answers of the problems.



The result is that you can easily know what happened on the network when a problem occurred – and get the root-cause of the problem fixed quickly and efficiently.

Troubleshooting is hard. TotalView makes it easy.

Additional Resource:

Webinar: Finding Packet Loss in the Network

(877) 748-1777
Sales@PathSolutions.com
www.PathSolutions.com