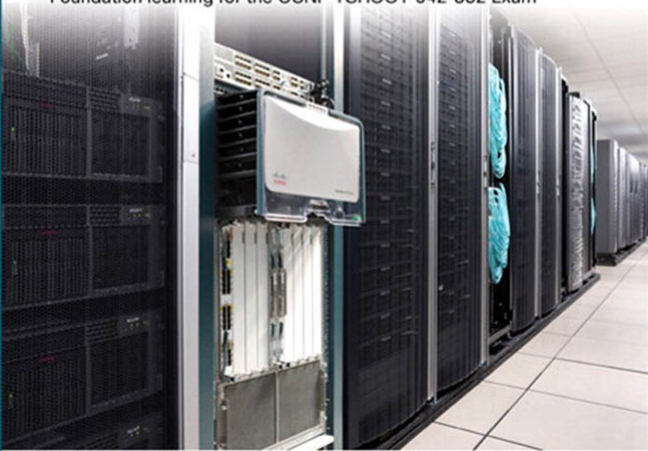
CISCO

Troubleshooting and Maintaining Cisco IP Networks (TSHOOT)

Foundation Learning Guide

Foundation learning for the CCNP TSHOOT 642-832 Exam



ciscopress.com

Amir Ranjbar, CCIE No. 8669

Troubleshooting and Maintaining Cisco IP Networks (TSHOOT) Foundation Learning Guide

Foundation learning for the CCNP TSHOOT 642-832

Amir Ranjbar, CCIE No. 8669

Published by:

Cisco Press

800 East 96th Street

Indianapolis, IN 46240 USA

Printed in the United States of America 1 2 3 4 5 6 7 8 9 0

First Printing March 2010

Library of Congress Cataloging-in-Publication Number is on file.

ISBN (10-digit): 1-58705-876-6

ISBN (13-digit): 978-1-58705-876-9

Warning and Disclaimer

This book is designed to provide information about the Troubleshooting and Maintaining Cisco IP Networks (TSHOOT) course as a reference in preparation for TSHOOT Exam 642-832 for the CCNP certification. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an “as is” basis. The authors, Cisco Press, and Cisco Systems, Inc., shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the discs or programs that may accompany it.

The opinions expressed in this book belong to the author and are not necessarily those of Cisco Systems, Inc.

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community. Readers' feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book or otherwise alter it to better suit your needs, you can contact us through e-mail at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Cisco Press or Cisco Systems, Inc., cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Corporate and Government Sales

Cisco Press offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact: U.S. Corporate and Government Sales, 1-800-382-3419 or corpsales@pearsontechgroup.com.

For sales outside the U.S., please contact: International Sales, internatioal@pearsoned.com.

Publisher: Paul Boger

Associate Publisher: Dave Dusthimer

Executive Editor: Mary Beth Ray

Managing Editor: Patrick Kanouse

Copy Editor: Keith Cline

Editorial Assistant: Vanessa Evans

Cover Designer: Louisa Adair

Composition: Mark Shirar

Business Operation Manager, Cisco Press: Anand Sundaram

Manager Global Certification: Erik Ullanderson

Senior Development Editor: Christopher Cleveland

Project Editor: Jennifer Gallant

Technical Editors: Elan Beer, Sonya Coker, Jeremy Creech, Rick Graziani, David Kotfila, Wayne Lewis, Jim Lorenz, Snezhy Neshkova, Allan Reid, Bob Vachon

Proofreader: MPS Limited, A Macmillan Company

Indexer: WordWise Publishing Services

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV
Amsterdam, The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**

Introduction

This book's content is based on the Cisco Systems TSHOOT course that has recently been introduced as part of the CCNP curriculum; it provides troubleshooting and maintenance knowledge and examples in the area of Cisco routing and switching. It is assumed that the reader possesses as much Cisco routing and switching background as that covered in the Cisco ROUTE and SWITCH courses. The content of this book is enough to prepare the reader for the TSHOOT exam, too. Note that the e-learning content of the Cisco TSHOOT course has been integrated into this book.

Teaching troubleshooting is not an easy task. This book introduces the reader to many troubleshooting methodologies and identifies the benefits of different techniques. Technical routing and switching topics are briefly reviewed, but the emphasis is on troubleshooting commands, and most important, presenting many troubleshooting examples. Chapter review questions help readers evaluate how well they absorbed the chapter content. The questions are also an excellent supplement for exam preparation.

Who Should Read This Book?

Those individuals who want to learn about modern troubleshooting methodologies and techniques and desire to see several relevant examples will find this book very useful. This book is most suitable for those who have some prior routing and switching knowledge but would like to learn or enhance their troubleshooting skill set. Readers who want to pass the Cisco TSHOOT exam can find all the content they need to successfully do so in this book. The Cisco Networking Academy CCNP TSHOOT course students will use this book as their official textbook.

Cisco Certifications and Exams

Cisco offers four levels of routing and switching certification, each with an increasing level of proficiency: Entry, Associate, Professional, and Expert. These are commonly known by their acronyms CCENT (Cisco Certified Entry Networking Technician), CCNA (Cisco Certified Network Associate), CCNP (Cisco Certified Network Professional), and CCIE (Cisco Certified Internetworking Expert). There are others, too, but this book focuses on the certifications for enterprise networks.

For the CCNP certification, you must pass exams on a series of CCNP topics, including the SWITCH, ROUTE, and TSHOOT exams. For most exams, Cisco does not publish the scores needed for passing. You need to take the exam to find that out for yourself.

To see the most current requirements for the CCNP certification, go to Cisco.com and click Training and Events. There you can find out other exam details such as exam topics and how to register for an exam.

The strategy you use to prepare for the TSHOOT exam might differ slightly from strategies used by other readers, mainly based on the skills, knowledge, and experience you have already obtained. For instance, if you have attended the TSHOOT course, you might take a different approach than someone who learned troubleshooting through on-the-job training. Regardless of the strategy you use or the background you have, this book is designed to help you get to the point where you can pass the exam with the least amount of time required.

How This Book Is Organized

Although this book can be read cover to cover, it is designed to be flexible and allow you to easily move between chapters to cover only the material with which you might need additional remediation. The chapters can be covered in any order, although some chapters are related and build upon each other. If you do intend to read them all, the order in the book is an excellent sequence to follow.

Each core chapter covers a subset of the topics on the CCNP TSHOOT exam. The chapters cover the following topics:

- **Chapter 1, "Planning Maintenance for Complex Networks":** This chapter presents and evaluates commonly practiced models and methodologies for network maintenance, introduces the processes and procedures that are fundamental parts of any network maintenance methodology, and identifies and evaluates tools, applications, and resources that support network maintenance processes.
- **Chapter 2, "Troubleshooting Processes for Complex Enterprise Networks":** This chapter explains the benefits of structured troubleshooting and how to implement troubleshooting procedures. Furthermore, the generic troubleshooting processes and their relation to network maintenance processes are analyzed, along with the role of change control and documentation.
- **Chapter 3, "Using Maintenance and Troubleshooting Tools and Applications":** This chapter reviews the built-in Cisco IOS tools and commands, plus some specialized tools and applications used for network troubleshooting and maintenance.
- **Chapter 4, "Maintaining and Troubleshooting Campus Switched Solutions":** This chapter reviews prominent campus multilayer switching technologies such as VLANs, Spanning Tree Protocol, inter-VLAN routing, and first-hop redundancy protocols, and it focuses on resolving problems related to these technologies.
- **Chapter 5, "Maintaining and Troubleshooting Routing Solutions":** This chapter's focus is on troubleshooting network layer connectivity. Troubleshooting EIGRP, OSPF, BGP, and route redistribution are presented in sequence.
- **Chapter 6, "Troubleshooting Addressing Services":** This chapter consists of two parts. The first part discusses how to identify and correct common IPv4 addressing service issues (NAT and DHCP specifically), and the second part does the same for common IPv6 routing issues.
- **Chapter 7, "Troubleshooting Network Performance Issues":** This chapter has three main sections. The first section presents troubleshooting network application services, and the second and third sections focus on troubleshooting performance issues on routers and switches.

- **Chapter 8, "Troubleshooting Converged Networks":** This chapter discusses troubleshooting topics that relate to proper operation of wireless, unified communications, and video applications.
- **Chapter 9, "Maintaining and Troubleshooting Network Security Implementations":** This chapter starts by explaining the troubleshooting challenges in secure networks. Next, troubleshooting the management plane, control plane, and data plane are discussed in sequence. Troubleshooting branch office connectivity is the final topic of this chapter.
- **Chapter 10, "Review and Preparation for Troubleshooting Complex Enterprise Networks":** This chapter reviews the key maintenance and troubleshooting concepts and tools, and concludes with a brief discussion about applying maintenance and troubleshooting concepts and tools.

There is also an appendix that has answers to the "Review Questions" questions found at the end of each chapter.

Chapter 2

Troubleshooting Processes for Complex Enterprise Networks

This chapter covers the following topics:

- Troubleshooting principles and approaches
- Implementing troubleshooting processes
- Integrating troubleshooting into the network maintenance process

Most modern enterprises depend heavily on the smooth operation of their network infrastructure. Network downtime usually translates to loss of productivity, revenue, and reputation. Network troubleshooting is therefore one of the essential responsibilities of the network support group. The more efficiently and effectively the network support personnel diagnose and resolve problems, the lower impact and damages will be to business. In complex environments, troubleshooting can be a daunting task, and the recommended way to diagnose and resolve problems quickly and effectively is by following a structured approach. Structured network troubleshooting requires well-defined and documented troubleshooting procedures.

This chapter explains the benefits of structured troubleshooting and identifies the leading principles that are at the core of all troubleshooting methodologies. Implementing troubleshooting procedures is the next topic, with a discussion on gathering and analyzing information and solving the problem. Finally, the generic troubleshooting processes and their relation to network maintenance processes are analyzed along with the role of change control and documentation.

Troubleshooting Methodologies

Troubleshooting is not an exact science, and a particular problem can be diagnosed and sometimes even solved in many different ways. However, when you perform structured troubleshooting, you make continuous progress, and usually solve the problems faster than it would take using an ad hoc approach. There are many different structured troubleshooting approaches. For some problems, one method might work better, whereas for others, another method might be more suitable. Therefore, it is beneficial for the troubleshooter to be familiar with a variety of structured approaches and select the best method or combination of methods to solve a particular problem.

Troubleshooting Principles

Troubleshooting is the process that leads to the diagnosis and, if possible, resolution of a problem. Troubleshooting is usually triggered when a person reports a problem. Some people say that a problem does not exist until it is noticed, perceived as a problem, and reported as a problem. This implies that you need to differentiate between a problem, as experienced by the user, and the actual cause of that problem. The time a problem is reported is not necessarily the same time at which the event causing the problem happened. Also, the reporting user generally equates the problem to the symptoms, whereas the troubleshooter often equates the problem to the root cause. For example, if the Internet connection fails on Saturday in a small company, it is usually not a problem, but you can be sure that it will turn into a problem on Monday morning if it is not fixed before then. Although this distinction between symptoms and cause of a problem might seem philosophical, you need to be aware of the potential communication issues that might arise from it.

Generally, reporting of a problem triggers the troubleshooting process. Troubleshooting starts by defining the problem. The second step is diagnosing the problem during which information is gathered, the problem definition is refined, and possible causes for the problem are proposed. Eventually this process should lead to a hypothesis for the root cause of the problem. At this time, possible solutions need to be proposed and evaluated. Next, the best solution is selected and implemented. Figure 2-1 illustrates the main elements of a structured troubleshooting approach and the transition possibilities from one step to the next.

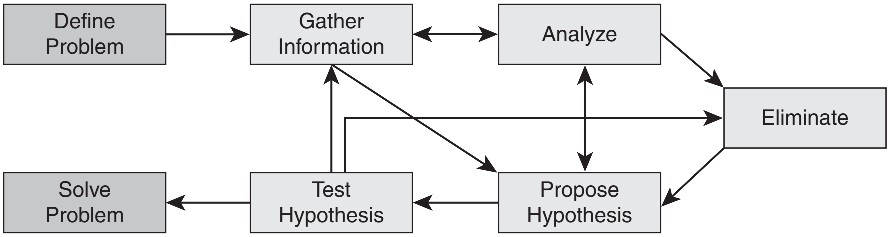


Figure 2-1 *Flow Chart of a Structured Troubleshooting Approach*

It is noteworthy, however, that the solution to a network problem cannot always be readily implemented and an interim workaround might have to be proposed. The difference between a solution and a workaround is that a solution resolves the root cause of the problem, whereas a workaround only alleviates the symptoms of the problem.

Although problem reporting and resolution are definitely essential elements of the troubleshooting process, most of the time is spent in the diagnostic phase. One might even believe that diagnosis is all troubleshooting is about. Nevertheless, within the context of network maintenance, problem reporting and resolution are indeed essential parts of troubleshooting. Diagnosis is the process of identifying the nature and cause of a problem. The main elements of this process are as follows:

- **Gathering information:** Gathering information happens after the problem has been reported by the user (or anyone). This might include interviewing all parties (user) involved, plus any other means to gather relevant information. Usually, the problem report does not contain enough information to formulate a good hypothesis without first gathering more information. Information and symptoms can be gathered directly, by observing processes, or indirectly, by executing tests.
- **Analyzing information:** After the gathered information has been analyzed, the troubleshooter compares the symptoms against his knowledge of the system, processes, and baselines to separate normal behavior from abnormal behavior.
- **Eliminating possible causes:** By comparing the observed behavior against expected behavior, some of the possible problems causes are eliminated.
- **Formulating a hypothesis:** After gathering and analyzing information and eliminating the possible causes, one or more potential problem causes remain. The probability of each of these causes will have to be assessed and the most likely cause proposed as the hypothetical cause of the problem.
- **Testing the hypothesis:** The hypothesis must be tested to confirm or deny that it is the actual cause of the problem. The simplest way to do this is by proposing a solution based on this hypothesis, implementing that solution, and verifying whether this solved the problem. If this method is impossible or disruptive, the hypothesis can be strengthened or invalidated by gathering and analyzing more information.

All troubleshooting methods include the elements of gathering and analyzing information, eliminating possible causes, and formulating and testing hypotheses. Each of these steps has its merits and requires some time and effort; how and when one moves from one step to the next is a key factor in the success level of a troubleshooting exercise. In a scenario where you are troubleshooting a complex problem, you might go back and forth between different stages of troubleshooting: Gather some information, analyze the information, eliminate some of the possibilities, gather more information, analyze again, formulate a hypothesis, test it, reject it, eliminate some more possibilities, gather more information, and so on.

If you do not take a structured approach to troubleshooting and go through its steps back and forth in an ad hoc fashion, you might eventually find the solution; however, the process in general will be very inefficient. Another drawback of this approach is that handing the job over to someone else is very hard to do; the progress results are mainly lost. This can happen even if the troubleshooter wants to resume his own task after he has stopped for a while, perhaps to take care of another matter. A structured approach to troubleshooting, regardless of the exact method adopted, yields more predictable results in the long run. It also makes it easier to pick up where you left off or hand the job over to someone else without losing any effort or results. A troubleshooting method that is commonly deployed both by inexperienced and experienced troubleshooters is the shoot-from-the-hip method. Using this method, after a very short period of gathering information, the troubleshooter quickly makes a change to see if it solves the problem. Even though it may seem like random troubleshooting on the surface, it is not. The reason is that the guiding principle for this method is knowledge of common symptoms and their corresponding causes, or simply extensive relevant experience in a particular environment or application. This technique might be quite effective for the experienced troubleshooter most times, but it usually does not yield the same results for the inexperienced troubleshooter. Figure 2-2 shows how the "shoot from the hip" goes about solving a problem, spending almost no effort in analyzing the gathered information and eliminating possibilities.

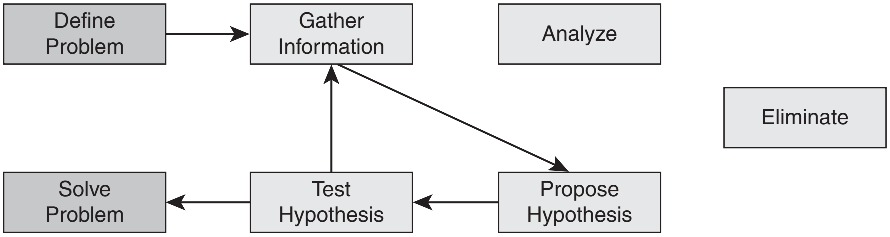


Figure 2-2 *The Shoot-from-the-Hip Troubleshooting Method*

Assume that a user reports a LAN performance problem and in 90 percent of the past cases with similar symptoms, the problem has been caused by duplex mismatch between users' workstation (PC or laptop) and the corresponding access switch port. The solution has been to configure the switch port for 100-Mbps full duplex. Therefore, it sounds reasonable to quickly verify the duplex setting of the switch port to which the user connects and change it to 100-Mbps full duplex to see whether that fixes the problem. When it works, this method can be very effective because it takes very little time. Unfortunately, the downside of this method is that if it does not work, you have not come any closer to a possible solution, you have wasted some time (both yours and users'), and you might possibly have caused a bit of frustration. Experienced troubleshooters use this method to great effect. The key factor in using this method effectively is knowing when to stop and switch to a more methodical (structured) approach.

Structured Troubleshooting Approaches

A structured troubleshooting method is used as a guideline through a troubleshooting process. The key to all structured troubleshooting methods is systematic elimination of hypothetical causes and narrowing down on the possible causes. By systematically eliminating possible problem causes, you can reduce the scope of the problem until you manage to isolate and solve the problem. If at some point you decide to seek help or hand the task over to someone else, your findings can be of help to that person and your efforts are not wasted.

Commonly used troubleshooting approaches include the following:

- **Top down:** Using this approach, you work from the Open Systems Interconnection (OSI) model's application layer down to the physical layer.
- **Bottom up:** The bottom-up approach starts from the OSI model's physical layer and moves up to the application layer.
- **Divide and conquer:** Using this approach, you start in the middle of the OSI model's stack (usually the network layer) and then, based on your findings, you move up or down the OSI stack.
- **Follow the path:** This approach is based on the path that packets take through the network from source to destination.
- **Spot the differences:** As the name implies, this approach compares network devices or processes that are operating correctly to devices or processes that are not operating as expected and gathers clues by spotting significant differences. In case the problem occurred after a change on a single device was implemented, the spot-the-differences approach can pinpoint the problem cause by focusing on the difference between the device configurations, before and after the problem was reported.
- **Move the problem:** The strategy of this troubleshooting approach is to physically move components and observe whether the problem moves with the components.

The sections that follow describe each of these methods in greater detail.

Top-Down Troubleshooting Method

The top-down troubleshooting method uses the OSI model as a guiding principle. One of the most important characteristics of the OSI model is that each layer depends on the underlying layers for its operation. This implies that if you find a layer to be operational, you can safely assume that all underlying layers are fully operational as well. So for instance, if you are researching a problem of a user that cannot browse a particular website and you find that you can establish a TCP connection on port 80 from this host to the server and get a response from the server, you can typically draw the conclusion that the transport layer and all layers below must be fully functional between the client and the server and that this is most likely a client or server problem and not a network problem. Be aware that in this example it is reasonable to conclude that Layers 1 through 4 must be fully operational, but it does not definitively prove this. For instance, non-fragmented packets might be routed correctly, while fragmented packets are dropped. The TCP connection to port 80 might not uncover such a problem. Essentially, the goal of this method is to find the highest OSI layer that is still working. All devices and processes that work on that layer or layers below are then eliminated from the scope of the problem. It might be clear that this method is most effective if the problem is on one of the higher OSI layers. This approach is also one of the most straightforward troubleshooting methods, because problems reported by users are typically defined as application layer problems, so starting the troubleshooting process at that layer is an obvious thing to do. A drawback or impediment to this method is that you need to have access to the client's application layer software to initiate the troubleshooting process, and if the software is only installed on a small number of machines, your troubleshooting options might be limited.

Bottom-Up Troubleshooting Method

The bottom-up troubleshooting approach also uses the OSI model as its guiding principle with the physical layer (bottom layer of the OSI stack) as the starting point. In this approach you work your way layer by layer up toward the application layer, and verify that relevant network elements are operating correctly. You try to eliminate more and more potential problem causes so that you can narrow down the scope of the potential problems. A benefit of this method is that all of the initial troubleshooting takes place on the network, so access to clients, servers, or applications is not necessary until a very late stage in the troubleshooting process. Based on experience, you will find that most network problems are hardware related. If this is applicable to your environment, the bottom-up approach will be most suitable for you. A disadvantage of this method is that, in large networks, it can be a time-consuming process, because a lot of effort will be spent on gathering and analyzing data and you always start from the bottom layer. The best bottom-up approach is to first reduce the scope of the problem using a different strategy and then switch to the bottom-up approach for clearly bounded parts of the network topology.

Divide-and-Conquer Troubleshooting Method

The divide-and-conquer troubleshooting method strikes a balance between the top-down and bottom-up troubleshooting approaches. If it is not clear which of the top-down or bottom-up approaches will be more effective for a particular problem, an alternative is to start in the middle (typically the network layer) and perform some tests such as ping. Ping is an excellent connectivity testing tool. If the test is successful, you can assume that all lower layers are functional, and so you can start a bottom-up troubleshooting starting from this layer. However, if the test fails, you can start a top-down troubleshooting starting from this layer. Whether the result of the initial test is positive or negative, this method will usually result in a faster elimination of potential problems than what you would achieve by implementing a full top-down or bottom-up approach. Therefore, the divide-and-conquer method is considered a highly effective troubleshooting approach.

Follow-the-Path Troubleshooting Method

The follow-the-path approach is one of the most basic troubleshooting techniques, and it usually complements one of the other troubleshooting methods such as the top-down or the bottom-up approach. The follow-the-path approach first discovers the actual traffic path all the way from source to destination. Next, the scope of troubleshooting is reduced to just the links and devices that are actually in the forwarding path. The principle of this approach is to eliminate the links and devices that are irrelevant to the troubleshooting task at hand.

Spot-the-Differences Troubleshooting Method

Another common troubleshooting approach is called spotting the differences. By comparing configurations, software versions, hardware, or other device properties, links, or processes between working and nonworking situations and spotting significant differences between them, this approach attempts to resolve the problem by changing the nonoperational elements to be consistent with the working ones. The weakness of this method is that it might lead to a working situation, without clearly revealing the root cause of the problem. In some cases, you are not sure whether you have implemented a solution or a workaround. Example 2-1 shows two routing tables; one belongs to Branch2, experiencing problems, and the other belongs to Branch1, with no problems. If you compare the content of these routing tables, as per the spotting-the-differences approach, a natural deduction is that the branch with problems is missing a static entry. The static entry can be added to see whether it solves the problem.

Example 2-1 *Spot the Differences: One Malfunctioning and One Working Router*

```
— — — — — — — — — — — — — - Branch1 is in good working order — — — — — — — — — — —
Branch1# show ip route
<...output omitted...>
     10.0.0.0/24 is subnetted, 1 subnets
C       10.132.125.0 is directly connected, FastEthernet4
C    192.168.36.0/24 is directly connected, BVI1
S*   0.0.0.0/0 [254/0] via 10.132.125.1
— — — — — — — — — — — — — - Branch2 has connectivity problems — — — — — — — — — — —
Branch2# show ip route
<...output omitted...>
     10.0.0.0/24 is subnetted, 1 subnets
C       10.132.126.0 is directly connected, FastEthernet4
C    192.168.37.0/24 is directly connected, BVI1
```

To further illustrate the spotting-the-differences approach and highlight its shortcomings, assume that you are troubleshooting a connectivity problem with a branch office router and you have managed to narrow down the problem to some issue with the DSL link. You have not discovered the real culprit, but you notice that this branch's router is an older type that was phased out in most of the other branch offices. In the trunk of your car, you have a newer type of router that must be installed at another branch office next week. You decide to copy the configuration of the existing malfunctioning branch router to the new router and use the new router at this branch. Now everything works to your satisfaction, but unfortunately, the following questions remain unanswered:

- Is the problem actually fixed?
- What was the root cause of the problem?
- What should you do with the old router?

- What will you do for the branch that was supposed to receive the new router you just used?

In a case like this, the default settings (and behavior) of the old and the newer operating systems (IOS) could be different, and that explains why using the newer router solves the problem at hand. Unless those differences are analyzed, explained, and documented (that is, communicated to others), merely changing the routers is not considered a solution to the problem, and the questions in the preceding list remain unanswered.

Obviously, the spotting-the-differences method has a number of drawbacks, but what still makes it useful is that you can use it even when you lack the proper technological and troubleshooting knowledge and background. The effectiveness of this method depends heavily on how easy it is to compare working and nonworking device, situations, or processes. Having a good baseline of what constitutes normal behavior on the network makes it easier to spot abnormal behavior. Also, the use of consistent configuration templates makes it easier to spot the significant differences between functioning and malfunctioning devices. Consequently, the effectiveness of this method depends on the quality of the overall network maintenance process. Similar to the follow the path approach, spot the differences is best used as a supporting method in combination with other troubleshooting approaches.

Move-the-Problem Troubleshooting Method

Move the problem is a very elementary troubleshooting technique that can be used for problem isolation: You physically swap components and observe whether the problem stays in place, moves with the component, or disappears entirely. Figure 2-3 shows two PCs and three laptops connected to a LAN switch, among which laptop B has connectivity problems. Assuming that hardware failure is suspected, you must discover if the problem is on the switch, the cable, or the laptop. One approach is to start gathering data by checking the settings on the laptop with problems, examining the settings on the switch, comparing the settings of all the laptops, and the switch ports, and so on. However, you might not have the required administrative passwords for the PCs, laptops, and the switch. The only data that you can gather is the status of the link LEDs on the switch and the laptops and PCs. What you can do is obviously limited. A common way to at least isolate the problem (if it is not solved outright) is cable or port swapping. Swap the cable between a working device and laptop B (the one that is having problems). Move the laptop from one port to another using a cable that you know for sure is good. Based on these simple moves, you can isolate whether the problem is cable, switch, or laptop related.

Just by executing simple tests in a methodical way, the move-the-problem approach enables you to isolate the problem even if the information that you can gather is minimal. Even if you do not solve the problem, you have scoped it to a single element, and you can now focus further troubleshooting on that element. Note that in the previous example if you determine that the problem is cable related, it is unnecessary to obtain the administrative password for the switch, PCs, and laptops. The drawbacks of this method is that you are isolating the problem to only a limited set of physical elements and not gaining

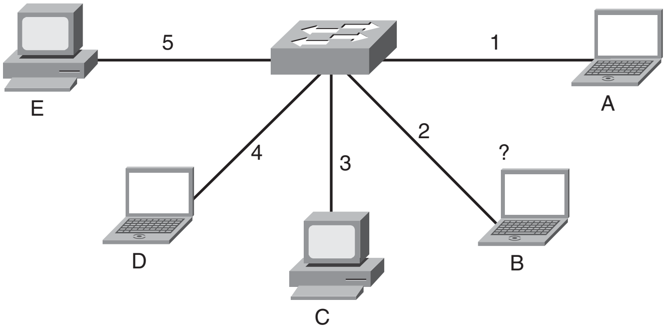


Figure 2-3 *Move the Problem: Laptop B Is Having Network Problems*

any real insight in what is happening, because you are gathering only very limited indirect information. This method assumes that the problem is with a single component. If the problem lies within multiple devices, you might not be able to isolate the problem correctly.

Troubleshooting Example: Methodologies

An external financial consultant has come in to help your company's controller with an accounting problem. He needs access to the finance server. An account has been created for him on the server, and the client software has been installed on the consultant's laptop. You happen to walk past the controller's office and are called in and told that the consultant can't connect to the finance server. You are a network support engineer and have access to all network devices, but not to the servers. Think about how you would handle this problem, what your troubleshooting plan would be, and which method or combination of methods you would use.

What possible approaches can you take for this troubleshooting task? This case lends itself to many different approaches, but some specific characteristics can help you decide an appropriate approach:

- You have access to the network devices, but not to the server. This implies that you will likely be able to handle Layer 1–4 problems by yourself; however, for Layer 5–7, you will probably have to escalate to a different person.
- You have access to the client device, so it is possible to start your troubleshooting from it.
- The controller has the same software and access rights on his machine, so it is possible to compare between the two devices.

What are the benefits and drawbacks of each possible troubleshooting approach for this case?

- **Top down:** You have the opportunity to start testing at the application layer. It is good troubleshooting practice to confirm the reported problem, so starting from the application layer is an obvious choice. The only possible drawback is that you will not discover simple problems, such as the cable being plugged in to a wrong outlet, until later in the process.
- **Bottom up:** A full bottom-up check of the whole network is not a very useful approach because it will take too much time and at this point, there is no reason to assume that the network beyond the first access switch would be causing the issue. You could consider starting with a bottom-up approach for the first stretch of the network, from the consultant's laptop to the access switch, to uncover potential cabling problems.
- **Divide and conquer:** This is a viable approach. You can ping from the consultant's laptop to the finance server. If that succeeds, you know that the problem is more likely to be with the application (although you have to consider potential firewall problems, too). If the ping fails, you are definitely dealing with a network issue, and you are responsible for fixing it. The advantage of this method is that you can quickly decide on the scope of the problem and whether escalation is necessary.
- **Follow the path:** Similar to the bottom-up approach, a full follow-the-path approach is not efficient under the circumstances, but tracing the cabling to the first switch can be a good start if it turns out that the link LED is off on the consultant's PC. This method might come into play after other techniques have been used to narrow the scope of the problem.
- **Spot the differences:** You have access to both the controller's PC and the consultant's laptop; therefore, spot the differences is a possible strategy. However, because these machines are not under the control of a single IT department, you might find many differences, and it might therefore be hard to spot the significant and relevant differences. Spot the differences might prove useful later, after it has been determined that the problem is likely to be on the client.
- **Move the problem:** Using this approach alone is not likely to be enough to solve the problem, but if following any of the other methods indicates a potential hardware issue between the consultant's PC and the access switch, this method might come into play. However, merely as a first step, you could consider swapping the cable and the jack connected to the consultant's laptop and the controller's PC, in turn, to see whether the problem is cable, PC, or switch related.

Many combinations of these different methods could be considered here. The most promising methods are top down or divide and conquer. You will possibly switch to follow-the-path or spot-the-differences approach after the scope of the problem has been properly reduced. As an initial step in any approach, the move-the-problem method could be used to quickly separate client-related issues from network-related issues. The bottom-up approach could be used as the first step to verify the first stretch of cabling.

Implementing Troubleshooting Procedures

The troubleshooting process can be guided by structured methods, but it is not static, and its steps are not always the same and may not be executed in the exact same order every time. Each network is different, each problem is different, and the skill set and experience of the engineer involved in a troubleshooting process is different. However, to guarantee a certain level of consistency in the way that problems are diagnosed and solved in an organization, it is still important to evaluate the common subprocesses that are part of troubleshooting and define procedures that outline how they should be handled. The generic troubleshooting process consists of the following tasks:

Step 1. Defining the problem

Step 2. Gathering information

Step 3. Analyzing the information

Step 4. Eliminating possible problem causes

Step 5. Formulating a hypothesis about the likely cause of the problem

Step 6. Testing that hypothesis

Step 7. Solving the problem

It is important to analyze the typical actions and decisions that are taken during each of these processes and how these could be planned and implemented as troubleshooting procedures.

The Troubleshooting Process

A network troubleshooting process can be reduced to a number of elementary subprocesses, as outlined in the preceding list. These subprocesses are not strictly sequential in nature, and many times you will go back and forth through many of these subprocesses repeatedly until you eventually reach the solving-the-problem phase. A troubleshooting method provides a guiding principle that helps you move through these processes in a structured way. There is no exact recipe for troubleshooting. Every problem is different, and it is impossible to create a script that will solve all possible problem scenarios. Troubleshooting is a skill that requires relevant knowledge and experience. After using different methods several times, you will become more effective at selecting the right method for a particular problem, gathering the most relevant information, and analyzing problems quickly and efficiently. As you gain more experience, you will find that you can skip some steps and adopt more of a shoot-from-the-hip approach, resolving problems more quickly. Regardless, to execute a successful troubleshooting exercise, you must be able to answer the following questions:

- What is the action plan for each of the elementary subprocesses or phases?
- What is it that you actually do during each of those subprocesses?

- What decisions do you need to make?
- What kind of support or resources do you need?
- What kind of communication needs to take place?
- How do you assign proper responsibilities?

Although the answers to these questions will differ for each individual organization, by planning, documenting, and implementing troubleshooting procedures, the consistency and effectiveness of the troubleshooting processes in your organization will improve.

Defining the Problem

All troubleshooting tasks begin with defining the problem. However, what triggers a troubleshooting exercise is a failure experienced by someone who reports it to the support group. Figure 2-4 illustrates reporting of the problem (done by the user) as the trigger action, followed by verification and defining the problem (done by support group). Unless an organization has a strict policy on how problems are reported, the reported problem can unfortunately be vague or even misleading. Problem reports can look like the following: “When I try to go to this location on the intranet, I get a page that says I don’t have permission,” “The mail server isn’t working,” or “I can’t file my expense report.” As you might have noticed, the second statement is merely a conclusion a user has drawn perhaps merely because he cannot send or receive e-mail. To prevent wasting a lot of time during the troubleshooting process based on false assumptions and claims, the first step of troubleshooting is always verifying and defining the problem. The problem has to be first verified, and then defined by you (the support engineer, not the user), and it has to be defined clearly.

A good problem description consists of accurate descriptions of symptoms and not of interpretations or conclusions. Consequences for the user are strictly not part of the problem description itself, but *can* be helpful to assess the urgency of the issue. When a problem is reported as “The mail server isn’t working,” you must perhaps contact the user and find out exactly what he has experienced. You will probably define the problem as “When user *X* starts his e-mail client, he gets an error message saying that the client can not connect to the server. The user can still access his network drives and browse the Internet.”

After you have clearly defined the problem, you have one more step to take before starting the actual troubleshooting process. You must determine whether this problem is your responsibility or if it needs to be escalated to another department or person. For example, assume the reported problem is this: “When user *Y* tries to access the corporate directory on the company intranet, she gets a message that says permission is denied. She can access all other intranet pages.” You are a network engineer, and you do not have access to the servers. A separate department in your company manages the intranet servers. Therefore, you must know what to do when this type of problem is reported to you as a network problem. You must know whether to start troubleshooting or to escalate it to the server department. It is important that you know which type of problems is

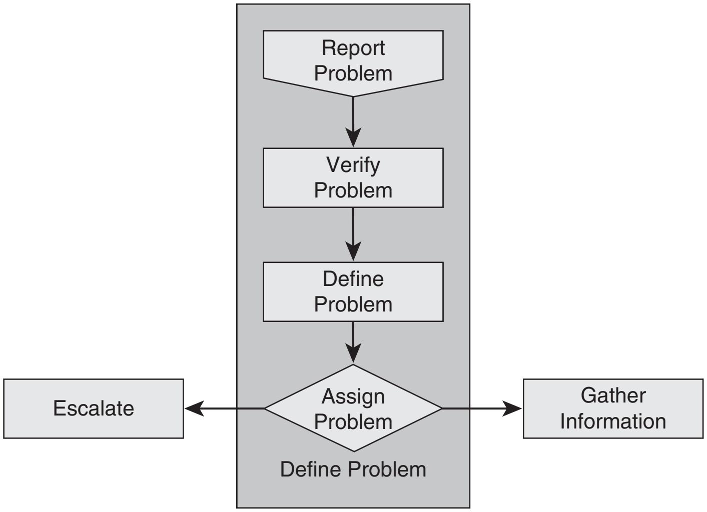


Figure 2-4 *A Reported Problem Must First Be Verified and Then Defined by Support Staff*

your responsibility to act on, what minimal actions you need to take before you escalate a problem, and how you escalate a problem. As Figure 2-4 illustrates, after defining the problem, you assign the problem: The problem is either escalated to another group or department, or it is network support's responsibility to solve it. In the latter case, the next step is gathering and analyzing information.

Gathering and Analyzing Information

Before gathering information, you should select your initial troubleshooting method and develop an information-gathering plan. As part of this plan, you need to identify what the targets are for the information-gathering process. In other words, you must decide which devices, clients, or servers you want to collect information from, and what tools you intend to use to gather that information (assemble a toolkit). Next, you have to acquire access to the identified targets. In many cases, you might have access to these systems as a normal part of your job role, but in some cases, you might need to get information from systems that you cannot normally access. In this case, you might have to escalate the issue to a different department or person, either to obtain access or to get someone else to gather the information for you. If the escalation process would slow the procedure down and the problem is urgent, you might want to reconsider the troubleshooting method that you selected and first try a method that uses different targets and would not require you to escalate. As you can see in Figure 2-5, whether you can access and examine the devices you identified will either lead to problems escalation to another group or department or to the gathering and analyzing information step.

The example that follows demonstrates how information gathering can be influenced by factors out of your control, and consequently, force you to alter your troubleshooting

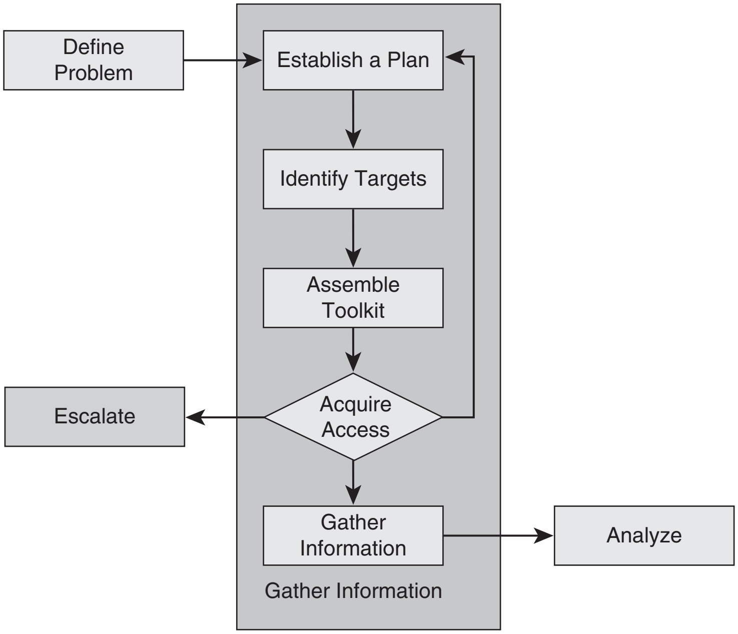


Figure 2-5 *Lack of Access to Devices Might Lead to Problem Escalation to Another Group*

approach. Imagine that it is 1.00 p.m. now and your company's sales manager has reported that he cannot send or receive e-mail from the branch office where he is working. The matter is quite urgent because he has to send out a response to an important request for proposal (RFP) later this afternoon. Your first reaction might be to start a top-down troubleshooting method by calling him up and running through a series of tests. However, the sales manager is not available because he is in a meeting until 4:30 p.m. One of your colleagues from that same branch office confirms that the sales manager is in a meeting, but left his laptop on his desk. The RFP response needs to be received by the customer before 5:00 p.m. Even though a top-down troubleshooting approach might seem like the best choice, because you will not be able to access the sales manager's laptop, you will have to wait until 4:30 before you can start troubleshooting. Having to perform an entire troubleshooting exercise successfully in about 30 minutes is risky, and it will put you under a lot of pressure. In this case, it is best if you used a combination of the "bottom-up" and "follow-the-path" approaches. You can verify whether there are any Layer 1–3 problems between the manager's laptop and the company's mail server. Even if you do not find an issue, you can eliminate many potential problem causes, and when you start a top-down approach at 4:30, you will be able to work more efficiently.

Eliminating Possible Problem Causes

After gathering information from various devices, you must interpret and analyze the information. In a way, this process is similar to detective work. You must use the facts and evidence to progressively eliminate possible causes and eventually identify the root of the problem. To interpret the raw information that you have gathered, for example, the output of **show** and **debug** commands, or packet captures and device logs, you might need to research commands, protocols, and technologies. You might also need to consult network documentation to be able to interpret the information in the context of the actual network's implementation. During the analysis of the gathered information, you are typically trying to determine two things: What is happening on the network and what should be happening. If you discover differences between these two, you can collect clues for what is wrong or at least a direction to take for further information gathering. Figure 2-6 shows that the gathered information, network documentation, baseline information, plus your research results and past experience are all used as input while you interpret and analyze the gathered information to eliminate possibilities and identify the source of the problem.

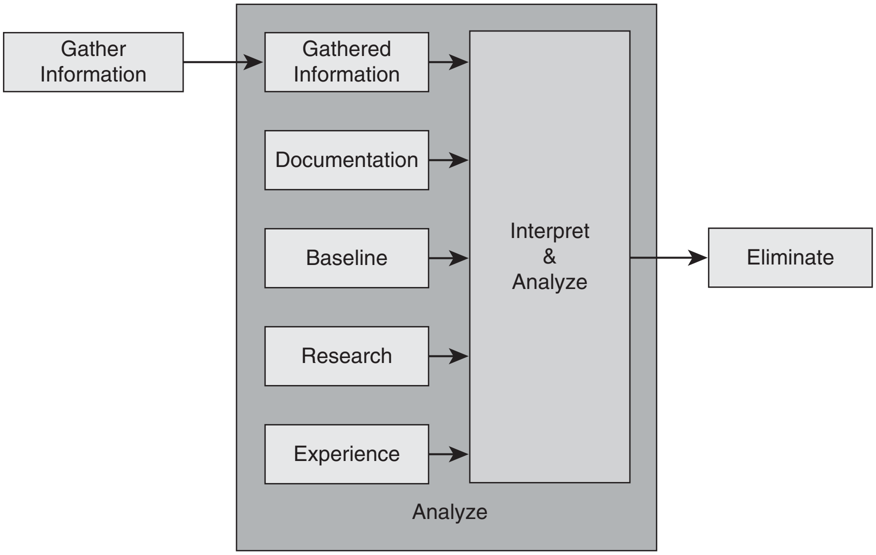


Figure 2-6 *Useful Factors That Can Feed and Support the Interpret and Analyze Task*

Your perception of what is actually happening is usually formed based on interpretation of the raw data, supported by research and documentation; however, your understanding of the underlying protocols and technologies also plays a role in your success level. If you are troubleshooting protocols and technologies that you are not very familiar with, you will have to invest some time in researching how they operate. Furthermore, a good baseline of the behavior of your network can prove quite useful at the analysis stage. If you know how your network performs and how things work under normal conditions, you can spot anomalies in the behavior of the network and derive clues from those deviations. The benefit of vast relevant past experience cannot be undermined. An experienced network engineer will spend significantly less time on researching processes, interpreting raw data, and distilling the relevant information from the raw data than an inexperienced engineer.

Formulating/Testing a Hypothesis

Figure 2-7 shows that based on your continuous information analysis and the assumptions you make, you eliminate possible problem causes from the pool of proposed causes until you have a final proposal that takes you to the next step of the troubleshooting process: formulating and proposing a hypothesis.

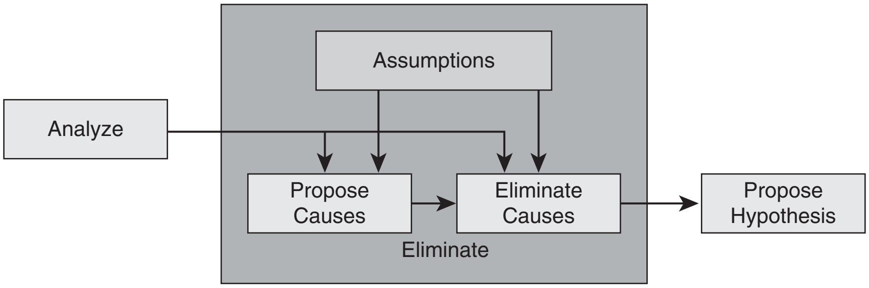


Figure 2-7 *Eliminating Possibilities and Proposing a Hypothesis Based on*

After you have interpreted and analyzed the information that you have gathered, you start drawing conclusions from the results. On one hand, some of the discovered clues point toward certain issues that can be causing the problem, adding to your list of potential problem causes. For example, a very high CPU load on your multilayer switches can be a sign of a bridging loop. On the other hand, you might rule out some of the potential problem causes based on the gathered and analyzed facts. For example, a successful ping from a client to its default gateway rules out Layer 2 problems between them. Although the elimination process seems to be a rational, scientific procedure, you have to be aware that assumptions play a role in this process, too, and you have to be willing to go back and reexamine and verify your assumptions. If you do not, you might sometimes mistakenly eliminate the actual root cause of a problem as a nonprobable cause, and that means you will never be able to solve the problem.

An Example on Elimination and Assumptions

You are examining a connectivity problem between a client and a server. As part of a follow-the-path troubleshooting approach, you decide to verify the Layer 2 connectivity between the client and the access switch to which it connects. You log on to the access switch and using the **show interface** command, you verify that the port connecting the client is up, input and output packets are recorded on the port, and that no errors are displayed in the packet statistics. Next, you verify that the client's MAC address was correctly learned on the port according to the switch's MAC address table using the **show mac-address-table** command. Therefore, you conclude that Layer 2 is operational between the client and the switch, and you continue your troubleshooting approach examining links further up the path.

You must always keep in mind which of the assumptions you have made might need to be reexamined later. The first assumption made in this example is that the MAC address table entry and port statistics were current. Because this information might not be quite fresh, you might need to first clear the counters and the MAC address table and then verify that the counters are still increasing and that the MAC address is learned again. The second assumption is hidden in the conclusion: Layer 2 is operational, which implies that the client and the switch are sending and receiving frames to each other successfully in both directions. The only thing that you can really prove is that Layer 2 is operational from the client to the switch, because the switch has received frames from the client.

The fact that the interface is up and that frames were recorded as being sent by the switch does not give you definitive proof that the client has correctly received those frames. So even though it is reasonable to assume that, if a link is operational on Layer 2 in one direction it will also be operational in the other direction, this is still an assumption that you might need to come back to later.

Spotting faulty assumptions is one of the tricky aspects of troubleshooting, because usually you are not consciously making those assumptions. Making assumptions is part of the normal thought process. One helpful way to uncover hidden assumptions is to explain your reasoning to one of your colleagues or peers. Because people think differently, a peer might be able to spot the hidden assumptions that you are making and help you uncover them.

Solving the Problem

After the process of proposing and eliminating some of the potential problem causes, you end up with a short list of remaining possible causes. Based on experience, you might even be able to assign a certain measure of probability to each of the remaining potential causes. If this list still has many different possible problem causes and none of them clearly stands out as the most likely cause, you might have to go back and gather more information first and eliminate more problem causes before you can propose a good hypothesis. After you have reduced the list of potential causes to just a few (ideally just one), select one of them as your problem hypothesis. Before you start to test your proposal, however, you have to reassess whether the proposed problem cause is within your area of responsibilities. In other words, if the issue that you just proposed as your hypothesis causes the problem, you have to determine whether it is your responsibility to solve it or you have to escalate it to some other person or department. Figure 2-8 shows the steps that you take to reach a hypothesis followed by escalating it to another group, or by testing your hypothesis.

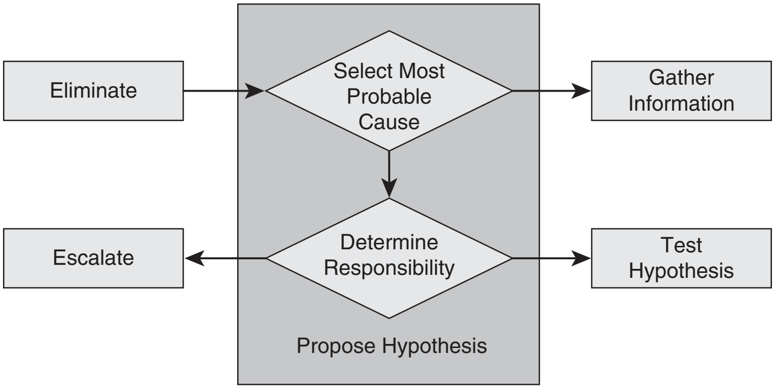


Figure 2-8 *Formulating a Hypothesis Is Followed by Escalation or Testing the Hypothesis*

If you decide to escalate the problem, ask yourself if this ends your involvement in the process. Note that escalating the problem is not the same as solving the problem. You have to think about how long it will take the other party to solve the problem and how urgent is the problem to them. Users affected by the problem might not be able to afford to wait long for the other group to fix the problem. If you cannot solve the problem, but it is too urgent to wait for the problem to be solved through an escalation, you might need to come up with a workaround. A temporary fix alleviates the symptoms experienced by the user, even if it does not address the root cause of the problem.

After a hypothesis is proposed identifying the cause of a problem, the next step is to come up with a possible solution (or workaround) to that problem, and plan an implementation scheme. Usually, implementing a possible solution involves making changes to the network. Therefore, if your organization has defined procedures for regular network maintenance, you must follow your organization's regular change procedures. The next step is to assess the impact of the change on the network and balance that against the urgency of the problem. If the urgency outweighs the impact and you decide to go ahead with the change, it is important to make sure that you have a way to revert to the original situation after you make the change. Even though you have determined that your hypothesis is the most likely cause of the problem and your solution is intended to fix it, you can never be entirely sure that your proposed solution will actually solve the problem. If the problem is not solved, you need to have a way to undo your changes and revert to the original situation. Upon creation of a rollback plan, you can implement your proposed solution according to your organization's change procedures. Verify that the problem is solved and that the change you made did what you expected it to do. In other words, make sure the root cause of the problem and its symptoms are eliminated, and that your solution has not introduced any new problems. If all results are positive and desirable, you move on to the final stage of troubleshooting, which is integrating the solution and documenting your work. Figure 2-9 shows the flow of tasks while you implement and test your proposed hypothesis and either solve the problem or end up rolling back your changes.

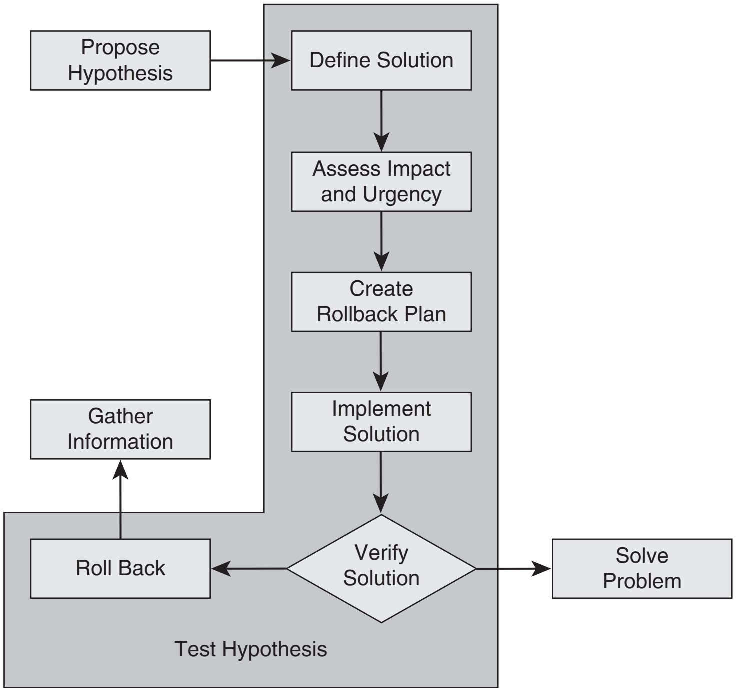


Figure 2-9 *Testing a Proposed Hypothesis*

You must have a plan for the situation if it turns out that the problem was not fixed, the symptoms have not disappeared, or new problems have been introduced by the change that you have made. In this case, you should execute your rollback plan, revert to the original situation, and resume the troubleshooting process. It is important to determine if the root cause hypothesis was invalid or whether it was simply the proposed solution that did not work.

After you have confirmed your hypothesis and verified that the symptoms have disappeared, you have essentially solved the problem. All you need to do then is to make sure that the changes you made are integrated into the regular implementation of the network and that any maintenance procedures associated with those changes are executed. You will have to create backups of any changed configurations or upgraded software. You will have to document all changes to make sure that the network documentation still accurately describes the current state of the network. In addition, you must perform any other actions that are prescribed by your organization's change control procedures. Figure 2-10 shows that upon receiving successful results from testing your hypothesis, you incorporate your solution and perform the final tasks such as backup, documentation, and communication, before you report the problem as solved.

The last thing you do is to communicate that the problem has been solved. At a minimum, you will have to communicate back to the original user that reported the problem, but if you have involved others as part of an escalation process, you should communicate

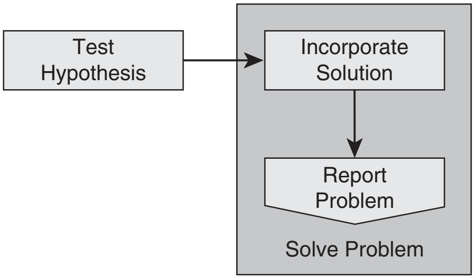


Figure 2-10 *The Final Step: Incorporate the Solution and Report the Problem as Solved*

with them, too. For any of the processes and procedures described here, each organization will have to make its own choices in how much of these procedures should be described, formalized, and followed. However, anyone involved in troubleshooting will benefit from reviewing these processes and comparing them to their own troubleshooting habits.

Integrating Troubleshooting into the Network Maintenance Process

Troubleshooting is a process that takes place as part of many different network maintenance tasks. For example, it might be necessary to troubleshoot issues arisen after implementation of new devices. Similarly, it could be necessary to troubleshoot after a network maintenance task such as a software upgrade. Consequently, troubleshooting processes should be integrated into network maintenance procedures and vice versa. When troubleshooting procedures and maintenance procedures are properly aligned, the overall network maintenance process will be more effective.

Troubleshooting and Network Maintenance

Network maintenance involves many different tasks, some of which are listed within Figure 2-11. For some of these tasks, such as supporting users, responding to network failures, or disaster recovery, troubleshooting is a major component of the tasks. Tasks that do not revolve around fault management, such as adding or replacing equipment, moving servers and users, and performing software upgrades, will regularly include troubleshooting processes, too. Hence, troubleshooting should not be seen as a standalone process, but as an essential skill that plays an important role in many different types of network maintenance tasks.

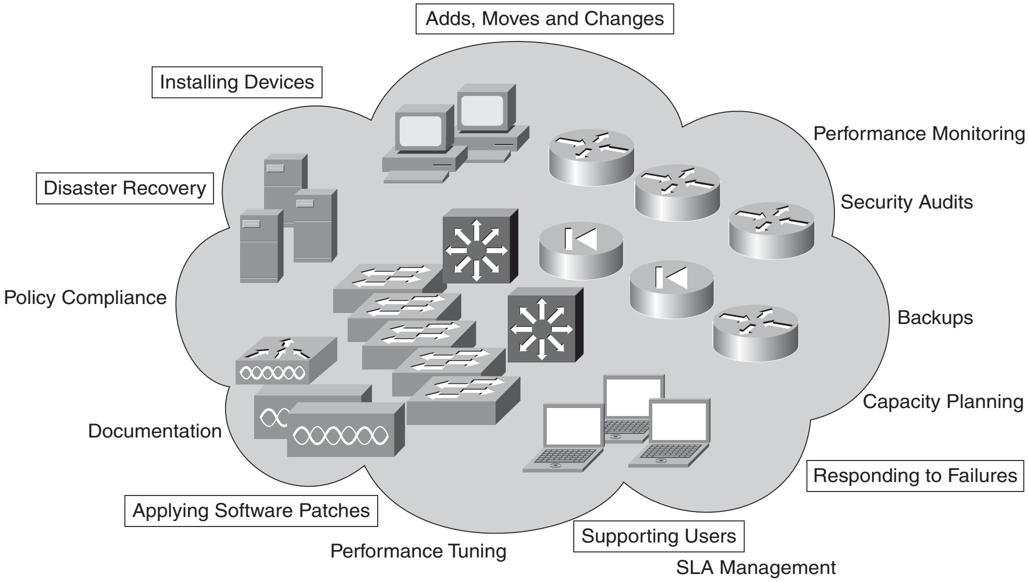


Figure 2-11 *Troubleshooting Plays an Important Role in Many Network Maintenance Tasks*

To troubleshoot effectively, you must rely on many processes and resources that are part of the network maintenance process. You need to have access to up-to-date and accurate documentation. You rely on good backup and restore procedures to be able to roll back changes if they do not resolve the problem that you are troubleshooting. You need to have a good baseline of the network so that you know which conditions are supposed to be normal on your network and what kind of behavior is considered abnormal. Also, you need to have access to logs that are properly time stamped to find out when particular events have happened. So in many ways, the quality of your troubleshooting processes depends significantly on the quality of your network maintenance processes. Therefore, it makes sense to plan and implement troubleshooting activities as part of the overall network maintenance process and to make sure that troubleshooting processes and maintenance processes are aligned and support each other, making both processes more effective.

Documentation

Having accurate and current network documentation can tremendously increase the speed and effectiveness of troubleshooting processes. Having good network diagrams can especially help in quickly isolating problems to a particular part of the network, tracing the flow of traffic, and verifying connections between devices. Having a good IP address schematic and patching administration is invaluable, too, and can save a lot of time while trying to locate devices and IP addresses. Figure 2-12 shows some network documentation that is always valuable to have.

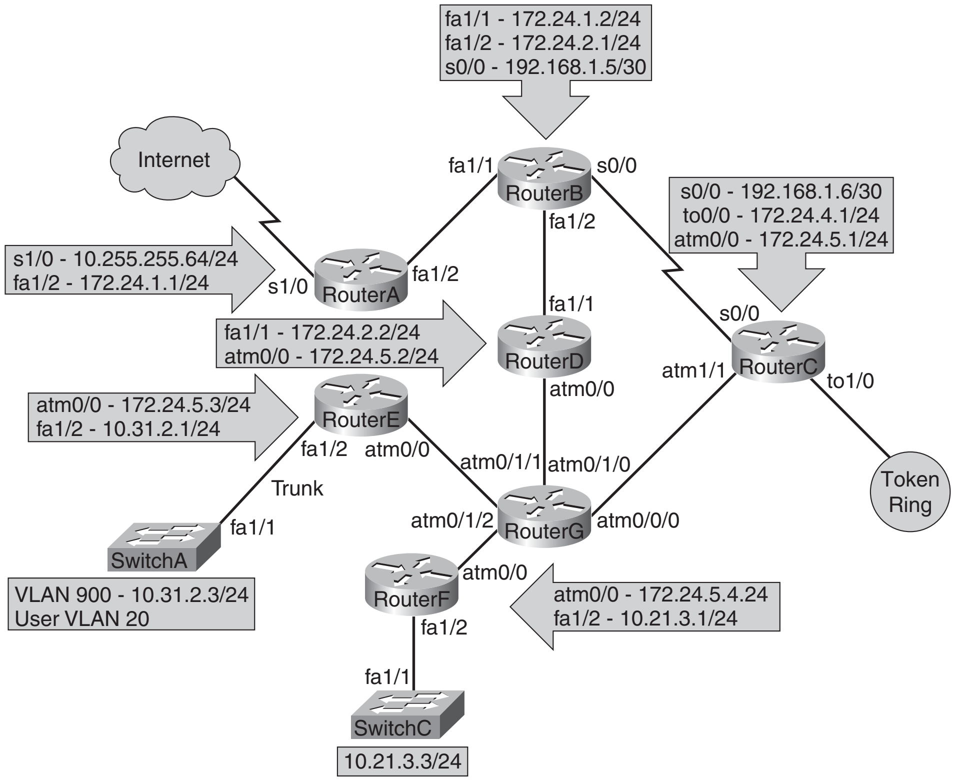


Figure 2-12 *Network Documentation Increases Troubleshooting Efficiency*

On the other hand, documentation that is wrong or outdated is often worse than having no documentation at all. If the documentation that you have is inaccurate or out-of-date, you might start working with information that is wrong and you might end up drawing the wrong conclusions and potentially lose a lot of time before you discover that the documentation is incorrect and cannot be relied upon.

Although everyone who is involved in network maintenance will agree that updating documentation is an essential part of network maintenance tasks, they will all recognize that in the heat of the moment, when you are troubleshooting a problem that is affecting network connectivity for many users, documenting the process and any changes that you are making is one of the last things on your mind. There are several ways to alleviate this problem. First, make sure that any changes you make during troubleshooting are handled in accordance with normal change procedures (if not during the troubleshooting process itself, then at least after the fact). You might loosen the requirements concerning authorization and scheduling of changes during major failures, but you have to make sure that after the problem has been solved or a workaround has been implemented to restore connectivity, you always go through any of the standard administrative processes like updating the documentation. Because you know that you will have to update the documentation afterward, there is an incentive to keep at least a minimal log of the changes that you make while troubleshooting.

One good policy to keep your documentation accurate, assuming that people will forget to update the documentation, is to schedule regular checks of the documentation. However, verifying documentation manually is tedious work, so you will probably prefer to implement an automated system for that. For configuration changes, you could implement a system that downloads all device configurations on a regular basis and compares the configuration to the last version to spot any differences. There are also various IOS features such as the Configuration Archive, Rollback feature, and the Embedded Event Manager that can be leveraged to create automatic configuration backups, to log configuration commands to a syslog server, or to even send out configuration differences via e-mail.

Creating a Baseline

An essential troubleshooting technique is to compare what is happening on the network to what is expected or to what is normal on the network. Whenever you spot abnormal behavior in an area of the network that is experiencing problems, there is a good chance that it is related to the problems. It could be the cause of the problem, or it could be another symptom that might help point toward the underlying root cause. Either way, it is always worth investigating abnormal behavior to find out whether it is related to the problem. For example, suppose you are troubleshooting an application problem, and while you are following the path between the client and the server, you notice that one of the routers is also a bit slow in its responses to your commands. You execute the **show processes cpu** command and notice that the average CPU load over the past 5 seconds was 97 percent and over the last 1 minute was around 39 percent. You might wonder if this router's high CPU utilization might be the cause of the problem you are troubleshooting. On one hand, this could be an important clue that is worth investigating, but on the other hand, it could be that your router regularly runs at 40 percent to 50 percent CPU and it is not related to this problem at all. In this case, you could potentially waste a lot of time trying to find the cause for the high CPU load, while it is entirely unrelated to the problem at hand.

The only way to know what is normal for your network is to measure the network's behavior continuously. Knowing what to measure is different for each network. In general, the more you know, the better it is, but obviously this has to be balanced against the effort and cost involved in implementing and maintaining a performance management system. The following list describes some useful data to gather and create a baseline:

- **Basic performance statistics such as the interface load for critical network links and the CPU load and memory usage of routers and switches:** These values can be polled and collected on a regular basis using SNMP and graphed for visual inspection.
- **Accounting of network traffic:** Remote Monitoring (RMON), Network Based Application Recognition (NBAR), or NetFlow statistics can be used to profile different types of traffic on the network.

- **Measurements of network performance characteristics:** The IP SLA feature in Cisco IOS can be used to measure critical performance indicators such as delay and jitter across the network infrastructure.

These baseline measurements are useful for troubleshooting, but they are also useful inputs for capacity planning, network usage accounting, and SLA monitoring. Clearly, a synergy exists between gathering traffic and performance statistics as part of regular network maintenance and using those statistics as a baseline during troubleshooting. Moreover, once you have the infrastructure in place to collect, analyze, and graph network statistics, you can also leverage this infrastructure to troubleshoot specific performance problems. For example, if you notice that a router crashes once a week and you suspect a memory leak as the cause of this issue, you could decide to graph the router's memory usage for a certain period of time to see whether you can find a correlation between the crashes and the memory usage.

Communication and Change Control

Communication is an essential part of the troubleshooting process. To review, the main phases of structured troubleshooting are as follows:

Step 1. Defining the problem

Step 2. Gathering facts

Step 3. Analyzing information

Step 4. Eliminating possibilities

Step 5. Proposing a hypothesis

Step 6. Testing the hypothesis

Step 7. Solving the problem

Figure 2-13 shows several spots where, while performing structured troubleshooting, communication is necessary if not inevitable.

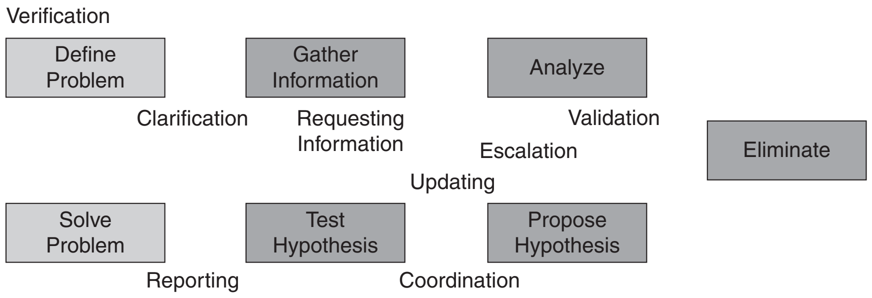


Figure 2-13 *Communication Plays a Role in All Phases of Structured Troubleshooting*

Within each phase of the troubleshooting process, communication plays a role:

- **Defining the problem:** Even though this is the first step of the structured troubleshooting, it is triggered by the user reporting the problem. Reporting the problem and defining the problem are not the same. When someone reports a problem, it is often too vague to act on it immediately. You have to verify the problem and gather as much information as you can about the symptoms from the person who reported the problem. Asking good questions and carefully listening to the answers is essential in this phase. You might ask questions such as these: "What do you mean exactly when you say that something is failing? Did you make any changes before the problem started? Did you notice anything special before this problem started? When did it last work? Has it ever worked?" After you communicate with the users and perhaps see the problems for yourself, and so on, you make a precise and clear problem definition. Clearly, this step is all about communication.
- **Gathering facts:** During this phase of the process, you will often depend on other engineers or users to gather information for you. You might need to obtain information contained in server or application logs, configurations of devices that you do not manage, information about outages from a service provider, or information from users in different locations, to compare against the location that is experiencing the problem. Clearly, communicating what information you need and how that information can be obtained determines how successfully you can acquire the information you really need.
- **Analyzing information and eliminate possibilities:** In itself, interpretation and analysis is mostly a solitary process, but there are still some communication aspects to this phase. First of all, you cannot be experienced in every aspect of networking, so if you find that you are having trouble interpreting certain results or if you lack knowledge about certain processes, you can ask specialists on your team to help you out. Also, there is always a chance that you are misinterpreting results, misreading information, making wrong assumptions, or are having other flaws in your interpretation and analysis. A different viewpoint can often help in these situations, so discussing your reasoning and results with teammates to validate your assumptions and conclusions can be very helpful, especially when you are stuck.
- **Proposing and testing a hypothesis:** Most of the time, testing a hypothesis involves making changes to the network. These changes may be disruptive, and users may be impacted. Even if you have decided that the urgency of the problem outweighs the impact and the change will have to be made, you should still communicate clearly what you are doing and why you are doing it. Even if your changes will not have a major impact on the users or the business, you should still coordinate and communicate any changes that you are making. When other team members are working on the same problem, you have to make sure that you are not both making changes. Any results from the elimination process might be rendered invalid if a change was made during the information-gathering phase and you were not aware of it. Also, if two changes are made in quick succession and it turns out that the problem was resolved, you will not know which of the two changes actually fixed it. This does not mean

that you cannot be working on the same problem as a team, but you have to adhere to certain rules. Having multiple people working on different parts of the network, gathering information in parallel or pursuing different strategies, can help in finding the cause faster. During a major disaster, when every minute counts, the extra speed that you can gain by working in parallel may prove valuable. However, any changes or other disruptive actions should be carefully coordinated and communicated.

- **Solving the problem:** Clearly, this phase also involves some communication. You must report back to the person who originally reported the problem that the problem has been solved. Also, you must communicate this to any other people who were involved during the process. Finally, you will have to go through any communication that is involved in the normal change processes, to make sure that the changes that you made are properly integrated in the standard network maintenance processes.

Sometimes it is necessary to escalate the problem to another person or another group. Common reasons for this could be that you do not have sufficient knowledge and skills and you want to escalate the problem to a specialist or to a more senior engineer, or that you are working in shifts and you need to hand over the problem as your shift ends. Handing the troubleshooting task over to someone else does not only require clear communication of the results of your process, such as gathered information and conclusions that you have drawn, but it also includes any communication that has been going on up to this point. This is where an issue-tracking or trouble-ticketing system can be of tremendous value, especially if it integrates well with other means of communication such as e-mail.

Finally, another communication process that requires some attention is how to communicate the progress of your troubleshooting process to the business (management or otherwise). When you are experiencing a major outage, there will usually be a barrage of questions from business managers and users such as "What are you doing to repair this issue? How long will it take before it is solved? Can you implement any workarounds? What do you need to fix this?" Although these are all reasonable questions, the truth is that many of these questions cannot be answered until the cause of the problem is found. At the same time, all the time spent communicating about the process is taken away from the actual troubleshooting effort itself. Therefore, it is worthwhile to streamline this process, for instance by having one of the senior team members act as a conduit for all communication. All questions are routed to this person, and any updates and changes are communicated to him; this person will then update the key stakeholders. This way, the engineers who are actually working on the problem can work with a minimal amount of distraction.

Change Control

Change control is one of the most fundamental processes in network maintenance. By strictly controlling when changes are made, defining what type of authorization is required and what actions need to be taken as part of that process, you can reduce the frequency and duration of unplanned outages and thereby increase the overall uptime of your network. You must therefore understand how the changes made as part of troubleshooting fit into the overall change processes. Essentially, there is not anything different between making a change as part of the maintenance process or as part of troubleshooting. Most of the actions that you take are the same. You implement the change, verify that it achieved the desired results, roll back if it did not achieve the desired results, back up the changed configurations or software, and document/communicate your changes. The biggest difference between regular changes and emergency changes is the authorization required to make a change and the scheduling of the change. Within change-control procedures, there is always an aspect of balancing urgency, necessity, impact, and risk. The outcome of this assessment will determine whether a change can be executed immediately or if it will have to be scheduled at a later time.

The troubleshooting process can benefit tremendously from having well-defined and well-documented change processes. It is uncommon for devices or links just to fail from one moment to the next. In many cases, problems are triggered or caused by some sort of change. This can be a simple change, such as changing a cable or reconfiguring a setting, but it may also be more subtle, like a change in traffic patterns due to the outbreak of a new worm or virus. A problem can also be caused by a combination of changes, where the first change is the root cause of the problem, but the problem is not triggered until you make another change. For example, imagine a situation where somebody accidentally erases the router software from its flash. This will not cause the router to fail immediately, because it is running IOS from its RAM. However, if that router reboots because of a short power failure a month later, it will not boot, because it is missing the IOS in its flash memory. In this example, the root cause of the failure is the erased software, but the trigger is the power failure. This type of problem is harder to catch, and only in tightly controlled environments will you be able to find the root cause or prevent this type of problem. In the previous example, a log of all privileged EXEC commands executed on this router can reveal that the software had been erased at a previous date. You can conclude that one of the useful questions you can ask during fact gathering is "Has anything been changed?" The answer to this question can very likely be found in the network documentation or change logs if network policies enforce rigid documentation and change-control procedures.

Summary

The fundamental elements of a troubleshooting process are as following:

- Gathering of information and symptoms
- Analyzing information
- Eliminating possible causes
- Formulating a hypothesis
- Testing the hypothesis

Some commonly used troubleshooting approaches are as follows:

- Top down
- Bottom up
- Divide and conquer
- Follow the path
- Spot the differences
- Move the problem

A structured approach to troubleshooting (no matter what the exact method is) will yield more predictable results in the long run and will make it easier to pick up the process where you left off in a later stage or to hand it over to someone else.

The structured troubleshooting begins with problem definition followed by fact gathering. The gathered information, network documentation, baseline information, plus your research results and past experience are all used as input while you interpret and analyze the gathered information to eliminate possibilities and identify the source of the problem. Based on your continuous information analysis and the assumptions you make, you eliminate possible problem causes from the pool of proposed causes until you have a final proposal that takes you to the next step of the troubleshooting process: formulating and proposing a hypothesis. Based on your hypothesis, the problem might or might not fall within your area of responsibility, so proposing a hypothesis is either followed by escalating it to another group or by testing your hypothesis. If your test results are positive, you have to plan and implement a solution. The solution entails changes that must follow the change-control procedures within your organization. The results and all the changes you make must be clearly documented and communicated with all the relevant parties.

Having accurate and current network documentation can tremendously increase the speed and effectiveness of troubleshooting processes. Documentation that is wrong or outdated is often worse than having no documentation at all.

To gather and create a network baseline, the following data proves useful:

- Basic performance statistics obtain by running **show** commands
- Accounting of network traffic using RMON, NBAR, or NetFlow statistics
- Measurements of network performance characteristics using the IP SLA feature in IOS

Communication is an essential part of the troubleshooting process, and it happens in all of the following stages of troubleshooting:

- Reporting the problem
- Gathering information

- Analyzing and eliminating possible causes
- Proposing and testing a hypothesis
- Solving the problem

Change control is one of the most fundamental processes in network maintenance. By strictly controlling when changes are made, defining what type of authorization is required and what actions need to be taken as part of that process, you can reduce the frequency and duration of unplanned outages and thereby increase the overall uptime of your network. Essentially, there is not much difference between making a change as part of the maintenance process or as part of troubleshooting.

Review Questions

1. Which three of the following processes are subprocesses or phases of a troubleshooting process? (Choose three.)

   a. Elimination

   b. Testing

   c. Termination

   d. Problem definition

   e. Calculation

   f. Compilation

2. Which four of the following approaches are valid troubleshooting methods? (Choose four.)

   a. Top down

   b. Bottom up

   c. Follow the path

   d. Seek-and-destroy

   e. Divide and conquer

3. Which three of the following troubleshooting approaches use the OSI reference model as a guiding principle? (Choose three.)

   a. Top down

   b. Bottom up

   c. Follow the path

   d. Spot the differences

   e. Move the problem

   f. Divide and conquer

4. Which of the following troubleshooting methods is most appropriate to find a bad cable?

a. Top down

b. Bottom up

c. Follow the path

d. Spot the differences

e. Move the problem

f. Divide and conquer

5. Which conditions make troubleshooting by spotting the differences more effective?

6. Which of the following has a clear problem definition?

a. I cannot order printer cartridges because the Internet is down.

b. My e-mail does not work.

c. I cannot log on to the network because the server is down.

d. When I try to access http://www.cisco.com, my Internet Explorer says that it cannot display the web page.

7. Which two of the following resources will help in interpreting and analyzing information gathered during troubleshooting? (Choose two.)

a. Documentation

b. Network baseline

c. Packet sniffers

d. Assumptions

8. Which of the following steps are parts of testing a hypothesis? (Choose four.)

a. Defining a solution

b. Creating a rollback plan

c. Implementing the solution

d. Defining the problem

e. Assessing impact and urgency

9. During which three of the troubleshooting phases could it be necessary to escalate a problem to a different department? (Choose three.)

   a. Defining the problem

   b. Gathering information

   c. Analyzing the facts

   d. Eliminating possible causes

   e. Formulating a hypothesis

   f. Solving the problem

10. Which of the following technologies can be deployed to measure critical network performance indicators such as delay and jitter?

   a. NetFlow

   b. RMON

   c. IP SLA

   d. NBAR

11. Which of the following phases of the troubleshooting process does not have communication as a major component?

   a. Defining the problem

   b. Solving the problem

   c. Eliminating causes

   d. Gathering information

Index