Enterprise Network Testing

The Role and Applications of Testing in Pre-Deployment, Migration, and Post-Deployment Network Operations

Andy Sholomon
Tom Kunath

ciscopress.com

Enterprise Network Testing

Andy Sholomon
Tom Kunath

Cisco Press
800 East 96th Street
Indianapolis, IN 46240

Enterprise Network Testing

Andy Sholomon, Tom Kunath

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 April 2011

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

ISBN-13: 978-1-58714-127-0

ISBN-10: 1-58714-127-2

Warning and Disclaimer

This book is designed to provide information about enterprise network testing. 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.

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.

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.

Publisher: Paul Boger

Associate Publisher: Dave Dusthimer

Executive Editor: Mary Beth Ray

Managing Editor: Sandra Schroeder

Senior Project Editor: Tonya Simpson

Editorial Assistant: Vanessa Evans

Book Designer: Louisa Adair

Cover Designer: Sandra Schroeder

Composition: Mark Shirar

Manager, Global Certification: Erik Ullanderson

Business Operation Manager, Cisco Press: Anand Sundaram

Development Editor: Kimberley Debus

Copy Editor: Bill McManus

Technical Editors: Tyler Pomerhn and Don Sautter

Indexer: Tim Wright

Proofreader: Sheri Cain

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.**

About the Authors

Andy Sholomon, CCIE No. 15179, works as a Network Consulting Engineer (NCE) in Cisco's Central Engineering Performance and Validation Testing team. He routinely plans and performs network testing for some of Cisco's largest Enterprise customers. In his six years at Cisco, Andy has been involved in both planning and deploying some of the largest enterprise data centers in the United States. He has also worked with some of Cisco's large service provider customers. Before joining Cisco, Andy worked as a Network Engineer in the global financial industry, spending 5 years at UBS in multiple roles, including security engineering, and worked as a Systems Engineer at Spear, Leeds & Kellogg (now a part of Goldman Sachs Group). Andy has been a speaker at the Cisco Live Networkers Conference. Besides the CCIE, Andy holds multiple industry certifications, including the CISSP and MCSE. Andy lives with his wife, daughter, and Great Dane in Chapel Hill, North Carolina.

Tom Kunath, CCIE No. 1679, is a Solutions Architect in Cisco's Advanced Services Central Engineering team, where he works as a design and test consulting engineer. With nearly 20 years in the networking industry, Tom has helped design, deploy, and operate many of Cisco's largest Enterprise and Financial customer networks. Before joining Cisco, Tom worked at Juniper Networks' Professional Services Group as a Resident Engineer supporting several service provider IP and MPLS backbones, and prior to that as a Principal Consultant at International Network Services (INS). In addition to his CCIE, Tom holds several industry certifications, including a Juniper JNCIS and Nortel Networks Router Expert. Tom lives in Raleigh, North Carolina, with his wife and two children.

About the Technical Reviewers

Tyler Pomerhn, CCIE No. 6676 (Routing/Switching, SNA/IP, Security, Service Provider), is an engineer with Cisco Systems within the Central Engineering Performance and Validation Testing Services (PVTS) group based in Research Triangle Park, North Carolina. He has worked in PVTS and the Customer Proof of Concept (CPOC) testing organizations for six years within Cisco, testing all manner of topologies and technologies for Fortune 100 companies to ensure their deployments were a success. Prior to working with testing groups inside Cisco, he worked with the Inside Sales team within Cisco in RTP, providing in-depth engineering resources to sales teams in the Federal Channels organization. Tyler holds a bachelor's degree in electrical engineering from SUNY Buffalo, as well as a bachelor's degree in physics from SUNY Fredonia, and has a total of 13 years of experience with computer networking.

Don Sautter, CCIE No. 13190 (Routing and Switching), is a Network Engineer at Cisco Systems within the Central Engineering Performance and Validation Testing Services (PVTS) group based in Research Triangle Park, North Carolina. He has worked for Cisco Systems for 10 years, the last 4 within PVTS performing systems solution testing and design validation. Don has 30 years of networking experience, during which he has performed a wide variety of engineering functions and held various positions within the industry.

Dedications

This book is dedicated to our loving families and our Cisco customers—the network engineers and managers who challenge us to provide them with the truth and offer them the simplest solution to meet their most complex problems.

"All fixed set patterns are incapable of adaptability or pliability. The truth is outside of all fixed patterns."

—Bruce Lee

Acknowledgments

We'd like to give special recognition to all of the Cisco engineers who contributed valuable content to this book: Gery Czirjak, for helping to write Chapter 3, "Testing and Lab Strategy Development;" Yenu Gobena, for helping to write Chapter 15, "IPv6 Functionality Test Plan;" Connie Varner, for sharing her insight on working in a large test organization and using the right test tools to get the job done; Tejas Suthar, who, as a network architect, understands first hand the role and value of structured testing in validating design; Varghese Thomas, for providing a case study on network readiness testing for VoIP; and our technical editors, Don Sautter and Tyler Pomerhn, who are also seasoned network test engineers in their day jobs, for keeping us honest and on track.

We'd also like to recognize our test tool vendors, in particular Ixia Networks and Spirent Communications, for their outstanding products and technical support; and Thomas Maufer, for an excellent contribution on application simulation, and the Mu Dynamics automated approach of creating test cases with live packet captures.

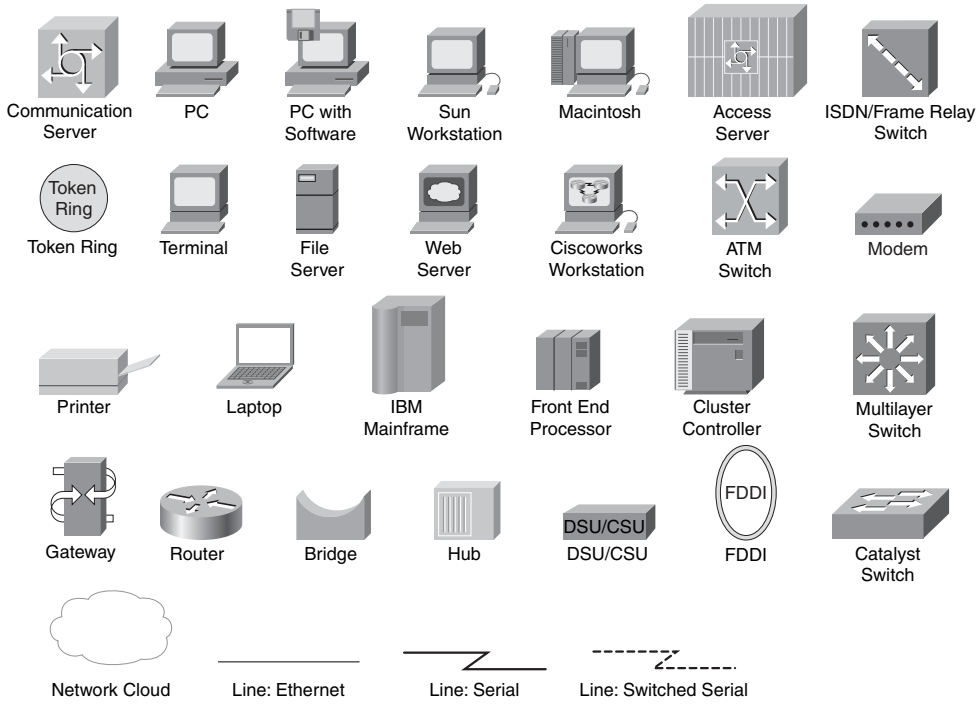
A quadruple "thumbs up" goes out to the production team for their help with this book. All of them have been incredibly professional and a pleasure to work with. Thank you for giving us the flexibility to finish this book while attending to the needs and timeframes of our own customer testing projects.

Finally, to our wives, for their support and encouragement with this project. Thank you both for picking up the "parenting slack" that we left during all the nights and weekends that we spent hunkered around our computers to get this done.

Contents at a Glance

Contents

Icons Used in This Book



Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in the *Cisco IOS Command Reference*, which describes these conventions as follows:

- **Boldface** indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that are manually input by the user (such as a **show** command).
- *Italics* indicate arguments for which you supply actual values.
- Vertical bars (|) separate alternative, mutually exclusive elements.
- Square brackets [ ] indicate optional elements.
- Braces { } indicate a required choice.
- Braces within brackets [{ }] indicate a required choice within an optional element.

Introduction

As many as 17 billion devices are projected to be connected to the Internet by 2014, fueled by more and more computing tasks now being handled online, from phone calls to personalized searches to downloading entertainment. In an effort to enhance the value of user transactions, corporations are increasingly transforming their network infrastructures from "packet plumbing" into "business platforms," converging ever more application and network functions along the way. This transformation has placed unprecedented pressures on network managers, now charged with meeting application service-level agreements for uptime and performance dictated to them by leaders of the business. Once considered an optional activity, network testing has become mandatory in many organizations and is a critical step toward meeting the expectations of near-zero downtime.

Goals and Methods

There is currently a void in publications that address test methodologies as they relate to the enterprise network lifecycle, particularly in the area of advanced technologies. Existing test publications, such as IETF RFCs and vendor test tool documentation, focus on test procedures for particular products and technologies, as opposed to complete network systems. While these are well known and used throughout the industry, they do not offer a complete blueprint to an organization that wants to know when, what, and exactly how to test products, solutions, and advanced technologies to keep its business up and running.

The primary goal of this book is to help you understand how you can develop effective test methods to discover in your network designs flaws or weaknesses that could potentially bring down your network. The intent is that this will be accomplished through the following methods:

- Establishing the importance of structured systems testing as a fundamental component of an enterprise architecture strategy
- Explaining the different types of testing that complement decision making during the various phases of a network's lifecycle
- Outlining a business and technical blueprint for developing a testing organization and lab facility
- Providing a series of customer case studies that reinforces the benefits of testing in the various phases of the networks lifecycle
- Providing test plan templates for various technical solutions that can be customized and used by readers in their own testing

Who Should Read This Book?

This book is intended to be read by network professionals who want to understand what structured system testing is, and how to effectively test complex network systems and technologies. The sample test plans included in this book are intended to be used as a reference, and can be customized for individual use.

How This Book Is Organized

Although this book could be read cover to cover, it is designed to be flexible and to allow you to easily move between chapters and sections of chapters to cover just the material that you need more work with. Part I, "Introduction to Enterprise Network Testing" (Chapters 1 through 5), is an introduction to systems testing, covering fundamental concepts with a focus on the relationship of testing to an enterprise architecture and design process. These chapters are mainly nontechnical, setting the stage for the case studies (Part II) and test plans (Part III) that follow in Chapters 6 through 18, which are the core chapters and can be covered in any order. If you intend to read them all, the order in the book is an excellent sequence to use.

Chapters 1 through 18 cover the following topics:

- **Chapter 1, "A Business Case for Enterprise Network Testing"**—This chapter introduces fundamental concepts of network testing and its critical role in validating design and making sound deployment decisions. The chapter begins with a discussion of why IT dollars should be spent on testing, and the evolution of the network as a platform for business. This is followed by a discourse on the cost of network downtime to the business, and how testing can be used to improve availability by validating design and reducing human error. The chapter concludes with an introduction to the different types of testing and a discussion about a structured approach to testing.
- **Chapter 2, "Testing Throughout the Network Lifecycle"**—This chapter builds upon the concepts introduced in Chapter 1 by explaining how a structured testing program complements the architecture and design process of the enterprise. An introduction to the Cisco Lifecycle Services approach of Plan, Prepare, Design, Implement, Operate, and Optimize (PPDIOO) follows, with examples of the different kinds of test activities that would commonly occur in each phase.
- **Chapter 3, "Testing and Lab Strategy Development"**—This chapter examines many of the business and technical considerations when developing an organizational testing strategy. The chapter includes a business cost analysis of building, staffing, and operating a lab, presenting the reader with various possible funding models. Best practices for test lab facility design and an estimate of the resources (equipment, tools, and people) that are necessary to sustain it are presented in depth, so that the reader can make an intelligent decision about whether it makes sense to build a lab or outsource testing.

- **Chapter 4, "Crafting the Test Approach"**—This chapter walks through the details of a structured approach to handling, scoping, and planning for different types of test requests. It begins with a suggested approach for assessing and scoping a test project, and offers guidance on how to identify the test scenarios, design and build a lab topology, select appropriate test tools, and write a detailed and concise test plan.
- **Chapter 5, "Executing the Test Plan"**—This chapter delves into many of the low-level details associated with system-level testing. Best practices for building a functional prototype of a network design are discussed, including methodologies for accommodating scale testing with the minimal amount of equipment. An introduction to several commercial, free, and Cisco IOS test tools is included, with tips on how to best leverage them for different types of testing.
- **Chapter 6, "Proof of Concept Testing Case Study"**—This chapter walks through a case study of how a financial customer leveraged proof of concept (POC) testing to gain confidence in a new network architecture that was proposed as part of a data center centralization/consolidation strategy.
- **Chapter 7, "Network Readiness Testing Case Study"**—This chapter walks through a case study of how a software development company leveraged network readiness testing on its production network to identify gaps and gauge readiness for a planned Unified Communications deployment.
- **Chapter 8, "Design Verification Testing Case Study"**—This chapter walks through a case study of how a university leveraged design verification testing to validate and refine a low-level design (LLD) for a new MPLS backbone infrastructure.
- **Chapter 9, "Migration Plan Testing Case Study"**—This chapter walks through a case study of how a university leveraged testing to validate the low-level steps and device configurations necessary to incrementally migrate its legacy IP network to a new MPLS/VPN network.
- **Chapter 10, "New Platform and Code Certification Case Study"**—This chapter walks through a case study of how a financial organization leveraged predeployment acceptance testing to certify new hardware, operating systems, and software features as part of a corporate change management compliance policy.
- **Chapter 11, "Network Ready for Use Testing Case Study"**—This chapter walks through a case study of how network ready for use (NRFU) testing was used as a final check to certify that a newly opened sports and entertainment complex was functional and ready to offer IP services to the staff and public on opening day.
- **Chapter 12, "Inter-Organization Secure Data Center Interconnect: Firewall Test Plan"**—This chapter introduces a technical solution for securely interconnecting the data centers of two separate enterprise networks, and then presents a detailed test plan for validating its performance and scalability.
- **Chapter 13, "Site-to-Site IPsec Virtual Private Networking: DMVPN and GET VPN Test Plans"**—This chapter discusses the motivation and details of two different site-to-site VPN designs based on IPsec technologies, and then presents detailed test plans to validate the functionality and scale of each.

- **Chapter 14, "Data Center 3.0 Architecture: Nexus Platform Feature and Performance Test Plan"**—This chapter discusses the low-level details of a next-generation data center solution built upon on the Nexus family of switches. A test plan is provided to validate the platform and system functionality of the solution components, which include: Nexus 5000 End-of-Row (EoR) Switches, Nexus 2000 Top-of-Rack (ToR) Fabric Extenders, Nexus 7000 core switches, and MDS 9500 Director-class SAN switches.
- **Chapter 15, "IPv6 Functionality Test Plan"**—This chapter includes an IPv6 technology primer and functionality test plan for some of its basic features.
- **Chapter 16, "MPLS/VPN: Scalability and Convergence Test Plan"**—This chapter discusses the low-level details of a hierarchical MPLS/VPN design that securely segments a global enterprise network. A systems test plan is provided to validate the solution, focusing on fast convergence, scalability, and high availability features.
- **Chapter 17, "WAN and Application Optimization: Performance Routing and Wide Area Application Services Test Plan"**—This chapter discusses a solution that includes PfR and WAAS features to optimize application performance across a WAN. A test plan is provided to validate the feature functionality and scalability, and to quantify the performance gains of deploying PfR and WAAS on the WAN.
- **Chapter 18, "Using the Lab for Hands-on Technology Training: Data Center 3.0 Configuration Lab Guide"**—This chapter illustrates how an enterprise lab can be used as a field enablement resource for hands-on training. A sample lab guide showing step-by-step Nexus 7000, MDS, and Unified Computing System provisioning tasks is provided as an example of how training materials should be structured to facilitate self-study using a custom-built lab topology.

This page intentionally left blank

Chapter 4

Crafting the Test Approach

This chapter covers the following topics:

- Motivations for Different Types of Testing
- Test Scoping
- Test Planning
- Choosing the Right Test Tools
- Writing the Test Plan

Chapter 1, "A Business Case for Enterprise Network Testing," stressed the importance of assessing the business reasons for testing as your first step in crafting an effective test approach. In the same way that a network designer would be foolish to specify equipment or make technical recommendations without prior knowledge of customer requirements, a test engineer would be misguided to attempt writing a test plan without first understanding the triggers, scope, motives, and expectations for the test initiative. By rushing ahead and skipping this critical step, you risk missing the mark in your testing, focusing on the wrong types of tests, or capturing erroneous results. This will waste precious time and resources as you continuously redefine your test plan; add, remove, or modify equipment to your lab topology; rerun your test cases; and generate reports. Taking time to identify the objectives and outline an assessment is critical before you ever step foot into the lab. Only after the following questions are answered should you begin to write a detailed test plan or build a lab topology:

- What are the test triggers?
- Who is requesting the test and what are their motives?
- How much testing is necessary and what constitutes success?

- What is the impact of test failure and what are the known risks?
- What are the resources (people, lab equipment, and test tools) required to execute the test?

As discussed in Chapter 2, "Testing Throughout the Network Lifecycle," a complimentary relationship between network testing and design functions exists in organizations that execute enterprise architecture effectively. We explained how structured testing complements and validates design deliverables, by providing examples of the different types of test requests that you can expect throughout the network's lifecycle.

This chapter will begin to fill in the practical details of what is necessary to build an effective approach toward different types of test requests. It begins with a suggested approach for assessing and scoping a test project, and offers guidance and best practices for the following considerations:

- How to identify test case scenarios
- How to develop a lab prototype
- How to choose the proper test tools necessary to execute the different types of tests
- How to write a detailed test plan

As with most technical undertakings, there is no absolute right way to approach systems testing. We do not promote ours as the only way to conduct successful testing. However, this is a proven method that will improve your chances of getting it right the first time.

Motivations for Different Types of Testing

The first step in assessing the objective and scope of a test effort is to understand the reasons for why it was requested, and the motives of the people or organization that requested it. In some instances, your client may be able to clearly tell you why they want testing and what they expect from testing, while others may only be able to tell you that their proposed deployment "is critical to the business and must be tested." In cases of the latter, you will need to rely on knowledge of your client, personal experience, and industry best practices to determine the objective and scope of the test effort. Following are some of the most common triggers and motivations associated with the different types of testing.

Proof of Concept Testing

Proof of concept (POC) testing is normally conducted during the Plan Phase of a new network design, or prior to the introduction of a new technology or service into an operational network. A network architect will often request that a POC test be completed to ensure that a new product or technology will work as expected in the context of their design. Successful POC testing is often the criteria for purchasing or moving into the low-level design (LLD) phase of a project, and in some cases POC testing is a mandatory

milestone to be completed before purchasing approval will be granted. In general, POC testing should be conducted systematically but persist only as long as necessary to prove that a proposed solution will work as expected. An exception to this general rule is when POC testing is used as a means to differentiate between similar products as part of a "bake-off" test. These types of tests often require extensive scale and feature testing in order to provide the necessary data to differentiate between competing products.

Network Readiness Testing

Network readiness testing is often included as part of a network assessment to determine whether a production network can meet the needs of a new application or service, and to identify any gaps that may hinder it. This type of testing is commonly conducted prior to deploying a Cisco Unified Communications (UC) solution, to help an enterprise determine whether its network will be able to meet the stringent requirements associated with real-time applications. Network readiness testing for UC often involves test tool injection and measurement of synthetic application traffic across a live network to predict how the actual application will perform when network elements are running in steady-state conditions, during day-to-day operations. Success criteria for this type of testing is easy to define because the SLA requirements with respect to delay, jitter, and loss are well understood for UC applications. Careful planning and coordination is often necessary when this type of network readiness testing is conducted so that production service disruption can be avoided.

Design Verification Testing

As the name suggests, this type of testing occurs during the Design Phase of a network's lifecycle. Design verification testing is similar to POC testing in that both are performed in order to gain confidence in a proposed design or solution prior to deployment. Design verification testing is typically more extensive than POC testing, however, as it often represents the last opportunity before implementation to fully examine whether all aspects of a design will function as expected when subjected to various stress conditions. Design verification testing is focused on performance, scalability, failover, operations, and manageability elements of a design. The output from this type of testing often feeds into the software recommendations, hardware specifications, and device configuration templates of an LLD document.

Hardware Certification Testing

Hardware certification testing often occurs during the Optimize Phase of a network's lifecycle as new platforms are introduced into existing operational networks to provide enhanced capabilities, better performance, or to replace equipment that is reaching end-of-life (EOL) status from a vendor supportability standpoint. Engineering and operations groups of an enterprise often require that hardware certification testing be completed before a product can be deployed in the production network. While it is generally accepted that equipment vendors will subject new platforms to a variety of tests during the

product development cycle, there is no substitute for customized, enterprise-specific testing to uncover defects or feature limitations that would not be found otherwise. It is nearly impossible for an equipment manufacturer to predict how a customer might deploy every feature, or the level of stress that a platform might be subjected to in an operational network with unique requirements. Likewise, it would be impractical for an equipment vendor to perform interoperability testing with every other vendor's equipment that might be deployed on a customer network. Hardware certification tests generally are simple in nature and shorter in duration as compared to other tests because they focus mainly on "unit level" test cases that can be conducted on relatively small lab topologies.

Network Operating System Testing

This type of testing is often required by the operations teams responsible for OS upgrades and is similar in scope to hardware certification testing. Network OS testing is often performed during the Optimize Phase of a network's lifecycle, as operating software reaches its end of life, or when new features or bug fixes are needed. Overall, there are many different levels of network OS testing that can be undertaken, some of which are only appropriate during the product development phase by the equipment vendor test groups. The most common types of tests conducted by clients are software acceptance tests, which are a customized suite of tests executed to verify general feature functionality. Regression tests are a variant of software acceptance tests, in which critical features that worked in the past are retested to ensure that they are still functioning properly in the new OS. The scope of network OS testing ranges from small, short-duration tests (such as bug fix verifications), to longer-duration, multithreaded tests that involve multiple features to be verified in parallel.

Migration Plan Testing

One of the most challenging and critical aspects of a networking project is the migration of users and services to a new network infrastructure. Even the best network designs are destined for failure if they cannot be implemented without causing extended service outages. Yet despite the risks, many network architects spend a disproportionate amount of time focused on the "end state" of their network designs, developing migration plans as an afterthought, if at all. A good migration plan should address how routing protocols and applications will interact when the network is partially migrated, providing success indicators and a backout plan when unexpected behavior is encountered during a migration. Testing of a migration plan is an essential part of the design process for networking projects of any scale. It is sometimes a requirement of the implementation or operations groups responsible for making changes to the network. In some instances, a migration plan can be developed during a design verification lab test effort by repeating the baseline and performance test scripts on the interim topology consisting of the old and new networks.

A high-level migration test plan approach for a new network backbone might look something like this:

Step 1. Build a prototype of the old and new network backbone topologies.

Step 2. Run a baseline test using known traffic patterns from the existing and new networks.

Step 3. Physically and logically interconnect the old and new network backbone topologies, as they will be connected during the migration. If this will take multiple steps, each interim topology should be tested.

Step 4. Run the same set of baseline tests on the interim network that you ran on the old network.

Step 5. Simulate device and circuit failure scenarios in each interim step of the migration in order to understand the impact on test traffic and whether any collateral damage occurs.

Step 6. Disconnect the old portion of the network or reprovision it on the new backbone. This should be done the same way as the migration plan will be done. If this will be done in multiple steps in your plan, you should test each one of them.

Step 7. Repeat the set of baseline tests.

Step 8. Run a set of new tests that exercise any new features or services to be offered by the new network.

A migration test would be considered successful when the baseline test results meet or exceed the performance of the old network and the features offered by the new network are verified.

Network Ready for Use Testing

A network ready for use (NRFU) test typically is executed on a new greenfield network infrastructure as a last step in certifying that it is ready to carry production traffic. During an NRFU test, network devices are methodically checked to ensure that they have been implemented according to the design specifications and are running in an error-free state.

Some of the tests commonly associated with NRFU testing include the following:

- Device tests (hardware/software inventory, power, syslog error checking)
- Circuit tests (throughput, delay, jitter, errors)
- Routing tests (adjacencies, routing table consistency)
- Traffic tests (end-to-end traffic testing)
- Network service tests (multicast, QoS, WAN acceleration)

- Application tests
- Management/NMS/security tests

In some cases, the NRFU testing extends to a limited production pilot where a low-risk site or portion of the network is cut over to the new network and monitored closely for a "probationary" period of time.

Test Scoping

Two questions you can expect from your clients when discussing potential test engagements are

- "How long will this testing take?"
- "How much will it cost?"

You would be wise to refrain from giving an answer to either question until you have a good understanding of the scope of what you will be testing. Defining the test scope will help you estimate the extensiveness of the test process and help you forecast costs. For example, a bug fix verification test is a test with a narrow scope and would not normally require a complicated lab topology to be built or an extended set of test cases to complete. The test scope would broaden as more devices, features, and test cases are added to the requirements.

It is sometimes difficult to define the scope of a test when you do not have a technical understanding of the solution or design you are being asked to verify. For this reason, consider involving your most senior technical people during the scoping exercise, despite their protestations to avoid it. Whenever possible, you should spend time with the network architects to review the proposed network design prior to scoping a test so that you can accurately estimate the necessary tools, people, and equipment.

The sections that follow describe some considerations when scoping a test engagement, the steps for which are as follows:

Step 1. Categorize the type of test to be completed.

Step 2. Identify project stakeholders.

Step 3. Identify indicators of test success.

Step 4. Estimate the resources required to complete the test.

Step 5. Identify risks.

Step 6. Identify the timeline for completion.

Step 1: Categorize the Type of Test to Be Completed

Identify the reasons for testing and try to determine whether the type of test fits into one of the categories described earlier in the chapter. Try to clearly articulate the trigger and objectives in one sentence, such as:

- "*Proof of concept test*, with the goal of gaining confidence and experience with the new voice gateway platform."
- "*Network ready for use test* to certify the new data center in Syracuse is operationally ready to be cut over into production and carry live customer data."
- "*Design verification test* to ensure that the next-generation WAN design will work and support company business needs for the next 3 to 5 years."

Categorizing a test as shown in the previous examples will allow you to consider and compare a potential test with similar engagements you may have completed in the past.

Step 2: Identify Project Stakeholders

Stakeholders are people or groups who have a vested interest in the outcome of the test initiative. They may include the company leadership, representatives from the business units, application developers, network architects, network operations staff, and contracted professional services firms. Early in the project, work with your project sponsor to create a list of all possible stakeholders so that you can develop an effective communications plan. The goal is to gain and sustain commitment to the test initiative, and to avoid negative behaviors such as stalling or undermining from people who demand to be "in the loop." It is a good idea to solicit input to the test plan from a wide variety of stakeholders so that a comprehensive approach can be taken.

For very large test efforts requiring input from multiple stakeholders, it may be helpful to assign and designate them using a well-known method from organizational design known as RACI (Responsible, Accountable, Consulted, Informed):

- **Responsible:** This person is responsible for completing a task.
- **Accountable:** This person will be called to account if the task is not completed and may manage the person who is responsible for completing the task. Project managers often have this role.
- **Consulted:** Though not accountable or responsible for completion, this person is consulted about aspects of the task.
- **Informed:** The holder of this passive role is kept informed but isn't accountable or responsible for tasks.

Step 3: Identify Indicators of Test Success

It is critical to get agreement from the stakeholders on what constitutes testing success; otherwise, exiting the test may become difficult. You don't need to identify success criteria for each and every test case during the Scoping Phase—that will come later during test planning. What you need to understand at this point is what constitutes overall test success, so that you have a good idea of which elements are important to build test cases around. Here are some examples of success criteria for various types of tests.

Network Design Verification Test

- Test Pass Criteria:
  - 100 percent of test cases have been executed.
  - Network design meets customer requirements with respect to feature functionality, performance, and convergence around failures.
  - No Severity 1 or 2 defects encountered.
  - All Severity 3 defects (if any) have been documented/filed and workarounds have been provided.
- Test Fail Criteria:
  - Severity 1 defects are found with no workaround in an area critical to the solution.
  - Severity 2 defects are found that can put in jeopardy the on-time deployments of the solution.
  - One or more "key" features are missing, are not operational, or don't meet customer-specific requirements.

Network Ready for Use Test

- Test Pass Criteria:
  - 100 percent of test cases have been executed.
  - All device hardware and line cards power up properly and successfully pass self-test diagnostics.
  - All devices configured with the hardware and Cisco IOS Software specified in the LLD document.
  - No device crashes observed during testing.
  - All circuits passing traffic in an error-free state.
  - Test traffic reroutes around failures within the constraints of the SLA specified in the LLD.

    - All devices can be monitored and managed by the NMS platforms in the NOC.
- Test Fail Criteria:
    - Device crashes observed during testing that cannot be correlated to faulty hardware.
    - Circuit errors incrementing.
    - Excessive errors seen in device logs that cannot be explained.
    - Test traffic does not converge around failures within the constraints of the SLA specified in the LLD.
    - Devices unreachable by the NMS platforms in the NOC.

Step 4: Estimate the Resources Required to Complete the Test

An estimation of the people, lab equipment, and test tools needed to complete the testing activities will be necessary to develop a reasonably accurate project timeline, and to provide pricing for the test if procurement is necessary. When estimating the people required, be sure to account for the skill level and number of people required for each of the necessary tasks; for example:

- Test plan development (high skill level)
- Equipment cabling and lab assembly (low skill level)
- Test plan execution (medium to high skill level, depending on technology and scale of test)
- Test results development (medium to high skill level, depending on technology and scale of the test)

It will be difficult to accurately estimate equipment resources needed for testing unless fairly complete design documentation is available for review. At a minimum, you should request a high-level network topology diagram so that you can identify any major components lacking in your lab inventory that will need to be procured. This will have an impact on the price and timeline for test execution. Be sure to consider that special software licenses may be needed if you are conducting applications testing. If new equipment will need to be installed, consider the space, power, and cooling impact to existing facilities. When scoping test engagements where unfamiliar platforms or new technology is involved, it may be necessary to allocate additional time for staff training or even account for assistance from outside consultants.

Test tools are also lab assets and need to be considered when estimating resources. Choosing the proper tool for a particular test is an art in itself, as explained in a later section, aptly titled “Choosing the Right Test Tools.” Work with your test tool vendors to help you understand whether their products have the capabilities to perform the types of

tests you will be executing. They are an invaluable resource, and many of the larger vendors often lease test tools for the duration of a test engagement if necessary.

Step 5: Identify Risks

A risk is an event or condition that, if it occurs, could negatively impact your test project. Sometimes risks are outside your control, such as when an equipment or software manufacturer cannot ship critical hardware or tools on time. This type of risk can be mapped to external dependencies. In other cases, risks can be mapped to internal dependencies, as in the case of having to hire staff with the appropriate skill set to complete a test. Tests that do not have clear objectives or well-documented designs present a risk of going over budget as the test team wastes time testing the wrong features or design scenarios. A few of the most common risk factors to consider when planning test schedules include the following:

- Shipping issues with equipment
- Third-party equipment requiring special skills from an outside vendor
- Lack of responsiveness from customer technical leaders or decision makers
- Contention for equipment due to a critical outage in another test bed or production area of the company
- Personnel without appropriate skills
- Personnel leaving the group/company

Step 6: Identify the Timeline for Completion

Large testing projects often require contributions from multiple people across different departments and even companies. Good teamwork, clear communications, and dedication to the project are necessary so that testing of a particular design or solution does not last forever. Because network testing is commonly triggered by a proposed network deployment, deadlines are often readily available and documented in an overall project plan. A challenge you will often face is developing a realistic test timeline that allows you to execute your testing thoroughly, while at the same time meeting the deadlines of the project. It is often helpful to allocate time to each of the various test phases when constructing your timeline; for example:

- Assessment of Objectives: 1 week
- Test Case Planning: 2 weeks
- Test Lab Setup: 2 weeks
- Test Execution: 2 weeks
- Test Documentation and Analysis: 1 week

Understand that there are dependencies that may affect your timeline, causing it to slip past the eight calendar weeks given in the preceding example. For example, you will not be able to move into the test execution phase if you are waiting for critical equipment or test tools to be procured, as identified in the assessment phase. Also, it is extremely important to obtain stakeholder feedback and signoff on your test plan prior to moving into your lab setup. Otherwise, you risk the chance of last-minute test changes or additions being requested by the stakeholders. A clear and continuous communications plan during this process is necessary to maintain an accurate test timeline.

Once the goals, objectives, and test scenarios are clearly stated and acknowledged by the stakeholders, you should be able to define success criteria, execute the tests, deliver the results, and exit the engagement.

Test Planning

Now that you and your client clearly understand and agree on the test scope, objectives, and criteria for success, it is finally time to roll up your sleeves and start working on the test plan. As always, it is important to collaborate with the stakeholders on the test plan to determine specifics regarding the application characteristics, behaviors, and new features that are expected of the new system. The prototype network system, equipment specifications, test cases, test tools, data to be collected, and results format must also be discussed and agreed upon. This is an important step in the process because it requires many decisions and significant teamwork.

Design the Functional Prototype Network System

For most types of tests, a working prototype network system of the intended design will serve as the platform upon which functionality, operation, and performance of the new system will be evaluated. A prototype network system is commonly illustrated in a set of network topology diagrams that represent a miniaturized version of the end-state network.

Note NRFU and network readiness tests typically do not require working prototypes to be built because testing and evaluation are conducted on the preproduction or operational network.

Designing the working prototype for the lab requires experience, creativity, ingenuity, and a deep understanding of the network design, solutions, and technologies to be evaluated. An awareness of the test triggers and motivations of the clients requesting the test is also necessary so that the right aspects of the design are represented and modeled. The first challenge you will face when designing the prototype is how much of the network system will need to be implemented to convince your client that the design meets business and technical requirements. The working prototype must be functional and able to demonstrate performance under stress and scale conditions, but it rarely needs to be a full-scale implementation of the new system. The design and scale of a prototype will

vary, depending on the type of test you are conducting and what features or components of the design must be showcased for your clients. How little or much of the network is represented in the prototype will be bounded by the people, money, equipment, and time you have to complete the test.

Use the information you collected during the Scoping Phase to help you determine the size and design of the network prototype, particularly the input and requirements from the key stakeholders. Whereas the design architect may be primarily focused on technical issues, such as validating a routing design or gaining performance benchmarks, network operations may be concerned only with the manageability and usability benefits of the new design. Pay attention to corporate politics and organizational structure with the client. For example, the primary stakeholder funding the project may be someone from the business side of an organization, possibly having rejected designs that the stakeholder considered overbuilt and expensive. The stakeholder's motivation might be to evaluate a "bronze standard" version of the design, composed of less costly components.

Constructing a High-Level Lab Topology Diagram

Early in the Plan Phase, you need to process all the input gathered from the Scoping Phase and develop a high-level topology diagram to start the test plan dialog with your clients. The high-level topology diagram is your initial draft of what the final lab network topology will look like, and it should be one of the first documents you share with your customer when discussing your approach. This diagram should include the major devices, including routers, switches, firewalls, servers, workstations, telephony, video, WAN simulators, test equipment, and so forth, that will be necessary to conduct the tests. Connections between the various devices should be shown, using "clouds" where appropriate to represent elements that will provide connectivity to the system, but not necessarily be under evaluation, such as a provider Multiprotocol Label Switching (MPLS) backbone or the Internet.

Figure 4-1 illustrates an example of a high-level lab topology diagram.

Note the relative lack of detail as compared to a typical network topology diagram. There are no IP addresses, equipment hardware configuration details, circuit speed indicators, or port numbering information. This is completely acceptable for a high-level test diagram considering that its purpose at this point is simply to gain consensus on the topology that you are proposing, and to serve as a reference to the test cases you will be developing. You will add details to this diagram to facilitate the lab build, and you most likely will be adding test-specific diagrams that illustrate particular test cases when it comes time to finalize the test plan.

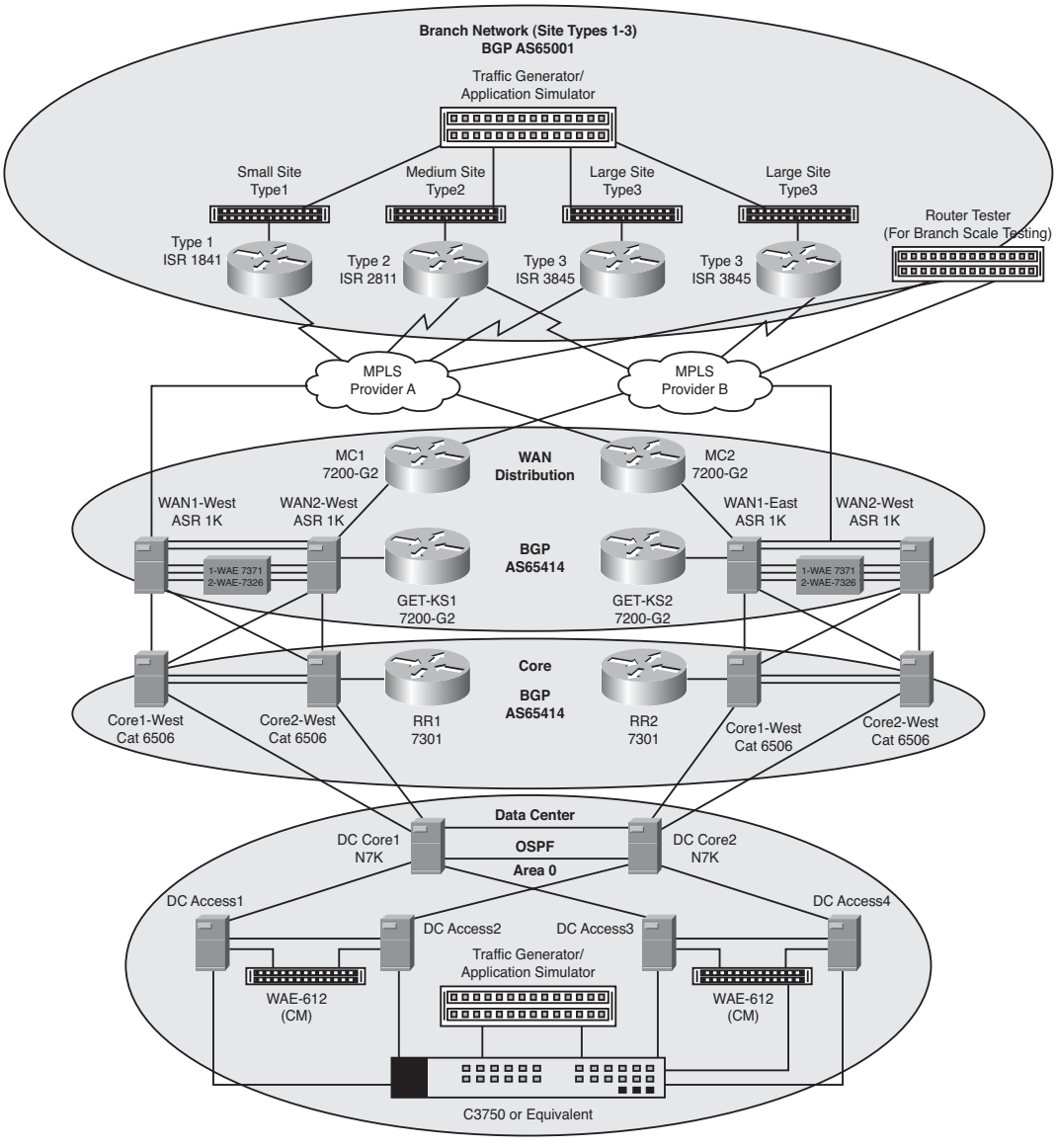


Figure 4-1 *Example High-Level Lab Topology Diagram*

Identifying the Test Suites and Test Cases

A test case in the context of internetworking is a set of conditions or variables under which a tester will determine whether or not a design element (network service, component, or feature) is working correctly. Test cases are the essence of the test plan, and they are sometimes collected or aggregated into test suites. Identifying the right test cases and expected output is an art form in itself, often requiring a series of conversations with the project stakeholders, architects, and operators of the network.

A simple description of the test cases you expect to run, and accompany the network topology diagram you have prepared, is sufficient to begin the test plan dialog. Your client may already have some ideas of the kinds of tests they want to see, so you should request their input right away. However, some clients have no idea on how testing is conducted; in these situations, you need to rely on your own testing experience and an understanding of the design goals, test triggers, and motivations for the testing.

A sample list of test suites and cases would look something like Table 4-1.

Table 4-1 *Example Test Suites and High-Level Test Cases*

Test Suite #	Test Suite	Test Case
1	Open Shortest Path First (OSPF) Routing	ABR Summarization Default Route Announce OSPF NSSA Redistribution BGP to OSPF Timer Optimizations
2	Border Gateway Protocol (BGP) Routing	Data Center iBGP Optimizations CE-PE eBGP Optimizations BGP Aggregation BGP Policy (Communities and Local-Pref)
3	Quality of Service (QoS)	Marking Queuing Traffic Shaping Policing Remarking at CE-PE to MPLS QoS Transparency Across MPLS CoPP
4	Cisco Wide Area Application Services (WAAS)	WCCP Redirects CIFS/FTP Acceleration
5	LAN	Campus Switching Branch Switching HSRP
6	Multicast	PIM Sparse Mode AnyCast RP with MSDP MVPN–MPLS

Table 4-1 *Example Test Suites and High-Level Test Cases*

Test Suite #	Test Suite	Test Case
7	Cisco Performance Routing (PfR)	Fast Reroute Load Balancing Interop with WAAS
8	Cisco Group Encrypted Transport VPN (GET VPN)	Group Member at Branch and WAN Distribution Cooperative Key Servers
9	Network Management	SNMP SSH AAA NTP Logging NetFlow
10	Performance/Scalability	Branch Router Performance (RFC 2544) WAN Route Saturation WAAS Scale
11	Negative Testing	Circuit Path Failover Line Card Failover Route Processor Failover Power Failures MPLS Cloud Hard Failures MPLS Cloud Soft Failures BGP Flapping

Choosing the Right Test Tools

> "Take these three items—some WD-40, a vise grip, and a roll of duct tape. Any man worth his salt can fix almost any problem with this stuff alone."
> —Clint Eastwood as Walt Kowalski in the movie *Gran Torino*

In contrast to simple home repair, system testing can be very complicated. In most cases, completing system testing without the assistance of sophisticated test tools is not practical or even feasible. Choosing the right tools and knowing how to use them is extremely important to effectively guide and streamline test execution and reporting of results. There are myriad test tools to choose from, and, as with anything else in networking, no single tool is right for every test or for every tester. While it would not be practical to mention every product or type of test tool in this book, the following sections describe some of the common categories of tools and which type of tests they would commonly be used for.

Stateless Packet Generators (Bit Blasters)

Packet generators are one of the simplest, but most important, types of test tool. Many different vendors sell packet generators, and the products differ from each other by many criteria such as the type of interface or interfaces supported, packets per second (pps) capacity, packet and traffic manipulation capability, results reporting, and automation.

Interfaces

Test tool vendors have historically differentiated themselves competitively by producing multiple interface cards for their products. When the world of low-speed WANs included multiple physical interface specifications (such as RS-232, V.35, and X.21), vendors were at the ready with interfaces, adaptor cables, and the like, to allow direct connection to the WAN circuit. With the growing popularity of Ethernet, including Gigabit Ethernet (GE) and 10-Gigabit Ethernet (10GE) offerings from service providers, many vendors now focus their product offerings on these interfaces, with the assumption that if the network under test includes a WAN link, the packets needed for testing can be forwarded to it via a router or switch over a Gigabit or 10-Gigabit Ethernet interface. Thus, the days of specialized interfaces on packet generators are essentially over, and test tool vendors are focused more specifically on the port density (feeds) and capacity (speeds) of their products.

A packet generator with Gigabit Ethernet interfaces generally has an RJ-45 (Category 5-style) interface, and you need to pay attention to the physical media settings of Gigabit or 100-Mbps operation and half-duplex, full-duplex, or duplex autonegotiation. When using a packet generator with a 10-Gigabit Ethernet interface, you are faced with choices about media adapters (XFPs, SFPs, etc.) and single- or multimode fiber cables.

Tool Power/Capacity

Protocol analyzers, such as Wireshark, which can be installed on a laptop computer, are capable of generating a low-speed stream of packets, which may be adequate for a basic feature test. An example of this usage would be to verify that a packet actually traverses a device or network under test, or that it is blocked, for instance, by an access list. Analyzer-based generators may also have the capability to replay a captured stream of packets back into the network. Analyzer-based generators are especially useful for field technicians, who are dispatched to network trouble sites and who are usually troubleshooting either Layer 1/physical-type problems or device misconfigurations.

For lab-based testing, more powerful devices are generally needed; depending on the vendor, you'll find generators of varying capacity. Packet generators at this level are usually chassis based, and you purchase different types of cards (GE or 10GE, etc.) for the chassis. The cards have varying numbers of generating ports (for instance, ten 1-GE ports or four 10GE ports) and you should pay attention to any caveats regarding capacity that the vendor lists in its documentation. For example, a four-port card may be capable of generating at full line rate on only two ports simultaneously, or a port may be able to support

full line rate but is limited as to the number of unique streams it can send; you need to know this as you plan your tool use in the test bed. Generally, these types of generators are also capable of packet capture and inspection (protocol analysis) when they are not being used for generation. Again, check the documentation to assess the product capabilities, preferably before you commit to the purchase!

Packet/Traffic Manipulation

There are many ways in which manipulation of the packets being generated is important to network testing. The address fields (both MAC and IP addresses) need to be easily modifiable so that the packets traverse the network as expected. The Layer 4 header can also be modified to specify a certain TCP or UDP port in order to simulate a certain type of network traffic, or test access lists. The packet payload can also be populated with specific content.

Most packet generators have the capability to send errored packets—for instance, packets that are too long, too short, or have bad checksums. This can be important if the purpose of a test is to verify that the device under test (DUT) does not forward an errored packet and, perhaps more importantly, that it does not experience operational stress, such as a lockup, high CPU, and so on, if it receives a steady stream of errored packets. In addition, the generator may allow the engineer to set the packet size within the range of valid sizes, either as fixed or incremental, or variable in accordance with some particular algorithm.

Other popular packet manipulations include the capability to set QoS bits within a packet, such as those used for Type of Service (ToS) or Differentiated Services Code Point (DSCP). These capabilities make it straightforward to test classification or queuing features in a switch or router.

For manipulating the traffic itself, most packet generators provide the capability to send traffic continuously or to send only a specified number of packets, which can be useful if the focus of the test is to determine if there is any packet loss. Other traffic rate possibilities include

- Sending by percentage of line rate
- Send rate in pps
- Send rate in bps
- Send rate determined by interpacket gap

Almost all packet generators on the market today have these packet- and traffic-manipulation capabilities, although the look and feel of the user interface makes some easier to learn and use than others.

Results

In some test scenarios, test results are collected exclusively from the device or network under test. However, the packet generator provides useful information regarding what it is doing, displaying the traffic being generated in metrics such as

- Transmit rate in bps
- Transmit rate in pps
- Number of packets, in total, transmitted
- Number of bytes, in total, transmitted

Most test tools have the further capability to coordinate information such as packets sent and received, and to use timestamps applied to the traffic being generated, across two or more ports in the same chassis, giving the tester important data such as packets lost, flow latency, and, for convergence tests, packet loss duration.

Traffic generators also generally have the capability to save and/or export these statistics in various formats, the most basic of which is a comma-separated values (CSV) file. The more sophisticated test tools are capable of generating graphs or pie charts within their own software, which can be very useful for producing result reports on the fly as tests are completed.

Automation

Some testing is conducted manually, with single-threaded test cases being executed and observed by the test engineer, and with data collected and conclusions drawn as the tool is in use. Other types of testing, such as regression testing, require a longer duration of tool use, with the expectation that test cases are run sequentially, and device and tool configurations are changed automatically as individual tests within a greater test suite are completed. For automated testing, the test tool must have some type of scripting interface. The most popular scripting language in the world of test engineers is Tool Command Language (TCL). Almost all packet generators available today have the necessary support for scripted operation via TCL. Further, most test tools have their own proprietary automation interface. An advantage to the use of TCL for automation is that it is a global application—for instance, with TCL, you could kick off a router configuration change, and then change the test tool configuration and restart the traffic flows. With the test tool's proprietary automation application, it's not likely that you could reconfigure a router.

When to Use Stateless Packet Generators

Packet generators are useful for many different types of tests. They can be used to create a certain level of background traffic as other application-specific tests are run. They can be used for stress testing, where the desired result is something like a performance chart for a DUT—for instance, pps compared with processor CPU. They are also useful for

QoS testing, because the tool allows you to set the ToS or DSCP bits, in addition to manipulating the packets sent in the other ways previously discussed.

Packet Generator Vendors

At press time, the North American market for packet generators is essentially shared by two major corporations—Spirent Communications and Ixia—although there are other "niche market" players:

- **Spirent:** The "classic" Spirent tool is called SmartBits; as additional software capabilities such as stateful traffic generation (see the next section) were added to the basic packet generator, new product names were rolled out. These new names are used to differentiate the SmartBits tool from the one with the new features, even though all products run on the same physical chassis. Spirent's present flagship test platform is TestCenter.
- **Ixia:** Like Spirent, the "first generation" Ixia product and the subsequent "feature-enhanced" software packages for IP traffic run on the same underlying chassis. The packet generator engine is called IxExplorer. Subsequent software packages such as stateful traffic generation are, as of this writing, called IxNetwork and IxLoad. Ixia currently supports another popular packet generator, the Agilent N2X, as Ixia acquired Agilent in 2009. Ixia has renamed the N2X product line IxN2X.

Stateful Packet Generators (Application Simulators)

For many years, packet generators verged on the capability of doing stateful traffic. Examples of what might be deemed "semistateful" capabilities include

- Capability to form a routing protocol (BGP, OSPF, etc.) neighbor relationship with a router and inject routes or other routing information such as OSPF link-state advertisements (LSA) into the network
- Capability to inject a multicast stream at one point in the network, and have a different test port send a "Join" for that stream, thus allowing multicast to be tested without a real multicast source in the test bed

What was lacking in these early tools was the capability to respond to network conditions, as a real TCP-enabled network node does. If a "real" network device running an application over TCP experiences packet loss, the normal behavior is to request a resend of the packets that were not received and then "window" the packet rate down, to avoid future packet loss. A stateless packet generator simply cannot do this, which is a reason why it is sometimes referred to as a "packet blaster."

Another benefit of most stateful tools is their capability to interact with devices in the network, as opposed to simply interacting with other test tool ports. Examples of this

capability include tools that can send traffic through firewalls, and those that can work with external servers or server load balancers.

Stateful Generation Tool Vendors

The first commercial stateful packet generator was a product called Chariot, and it consisted of endpoint software, installed on Microsoft Windows or UNIX/Linux computers, and console software, installed on a separate machine. The console computer controlled the endpoints (directing them to send application test traffic) and collected results. The Chariot endpoint software was acquired by Ixia, which integrated the endpoint software into the Ixia chassis line cards, with the display console running on a GUI called IxChariot. The current version of this stateful traffic generator is called IxLoad.

Spirent's Avalanche 3100 appliance solution is similar to Ixia's IxLoad product, with the capability to generate stateful traffic at high speeds for application and performance testing. The Avalanche 3100 is a line-rate, 1-Gbps and 10-Gbps Layer 4–7 multiprotocol stateful traffic performance solution with multi-10-Gbps capacity for stateful application traffic generation.

Results Reporting

To be useful for most types of testing, stateful tools must also provide the same basic types of results as packet generators, such as packets sent/received, packet rate sent/received, loss, and latency. However, stateful tools generally have a more sophisticated format for test results. Because stateful tools generally seek to mimic various types of applications, such as a DNS lookup or an HTTP Get, they need to provide measurements that are reflective of the "user experience" for an application. Examples of such metrics include the following:

- Response time (for instance, on an HTTP Get)
- Throughput (for instance, for an FTP Put)
- Transactions per second (for highly interactive applications with small-packet payloads)
- Voice quality (simulations of Voice over IP, with Mean Opinion Scores being derived from delay, jitter, and loss measurements)
- Connection setup rate or maximum concurrent connections for firewall

As with stateless tools, the product must offer some capability to export results data from the tool, and the GUI for transferring these results should not be overly complex.

When to Use Stateful Packet Generators

Stateful generators are required when testing devices or technologies that do not respond correctly to stateless packets, and they add value to many other kinds of tests. They can be used for many of the same types of tests as packet generators—to provide a level of

background traffic, or to stress test a device such as a stateful firewall. More information on choosing stateful or stateless traffic is available in the "Understanding the Different Types of Test Traffic" section of Chapter 5.

When used as a routing protocol generator, injecting routes and traffic that follows them, a stateful tool is well suited to testing the scalability and resilience of a particular network design. A stateful generator can also be used for determining the capacity of a given design, particularly if it includes a low-speed link (such as a WAN link), by drawing on user experience results such as response time, or Mean Opinion Score for VoIP. There are many newer technologies, such as Cisco Performance Routing (PfR), that are best tested with stateful flows; for example, with PfR, the PfR Master Controller makes decisions about managing traffic based on delay and throughput metrics that it can collect from real TCP-based transactions. A stateful generator is also useful for QoS testing, where the test is expected to produce user experience results—for instance, where low-priority TCP flows are expected to back off in the face of congestion, allowing preferred service to other types of traffic.

Network Delay and Impairment Tools

Another important tool in the tester's toolkit is a device, sometimes referred to as a "black box," that allows the injection of delay or impairment into a network under test. When it is crucial to understand how a technology, an application, a device, or a design copes with impairment or delay, this kind of tool is invaluable.

Delay

There are many sources for delay in a network, including serialization delay, propagation or transit delay, and queuing delay with network devices. Delay is especially troublesome if a network includes low-speed links, and if a user application is delay-sensitive. Far too often, applications are tested on a single LAN, with no thought given to how they may perform with a client device located miles away from a server. To properly assess the user experience in the face of delay, a test lab should include the capability to simulate network delay. This is commonly achieved using appliance-based impairment tools, or in some cases extremely (several kilometers) long spools of fiber.

Impairment

As with delay, impairment at the physical layer of the network is another stress that should be considered when rolling out a new design, technology, or application. When bit-level losses cause packet corruption, TCP-based applications detect packet loss and require acknowledged retransmission, which slows down application throughput and user response time. In the face of packet loss, UDP-based applications simply lose integrity; the most obvious instance of this being VoIP traffic, where packet loss is noted by the end users in the form of poor voice quality (for instance, "choppiness" or a sense of the voice dropping in and out). An imprecise way to introduce errors at the physical layer is to wiggle connectors or use cables that are known to have bad conductors or internal

shorts. However, it is very difficult to do this scientifically; a cable that is simply a little bit bad one day may be completely "shot" the next, making it impossible to produce the same error condition. It is far better to use a tool that introduces a programmable and predictable amount of impairment.

Note Fortunately, most devices that function as delay generators also function as impairment generators.

Network Modeling and Emulation Tools

In some instances, it is simply not practical or feasible to send test traffic into a prototype network system built with actual network equipment. Network architects are often asked to evaluate one or more alternatives to a proposed network design, comparing the functionality and estimated performance of various topologies or routing protocol configurations. In these situations, it is often preferable to leverage a modeling tool, which uses software and mathematical models to analyze the behavior of a network, or an emulation tool, which duplicates the functions of a network device by implementing its operating software on another computer.

Network Modeling Tools

Network modeling tools offer the capability to create visual simulations of network topologies by using actual production device configurations as the "seed" files. These tools rely on network device libraries that model the behavior of major networking devices (routers, switches, firewalls) and simulate the interconnection speeds and performance of circuits so that network capacity, resiliency, and application performance can be estimated in a variety of conditions. Network modeling tools are more easily set up than real test topologies and can allow alternatives to be more quickly compared and easily evaluated. Typical applications of network modeling tools include simple tasks such as device configuration audits, to complex sophisticated functions such as capacity planning and application performance analysis. Modeling tools should be used for early decision making on a design. They may point out obvious issues before you build out your test bed, and thus save you valuable time.

Network Modeling Tool Vendors

Cisco Advanced Services teams use several different network modeling tools. They can be invaluable for getting a quick evaluation of the effects a change will have on a large network topology. Three of them are discussed in the following sections.

OPNET Technologies

One of today's market leaders of network modeling tools is OPNET Technologies (www.opnet.com), which provides several products that enable enterprise and service provider planners to analyze how network devices, protocols, applications, and servers operate. OPNET's products model Cisco's and other vendors' device configurations, and

they include tools for taking captured data and analyzing which components are the bottlenecks.

Shunra Software

Another one of today's market leaders of modeling tools is Shunra Software (http://www.shunra.com), which offers a suite of products known as Virtual Enterprise Solutions. Like the others, Shunra provides tools that have the capability to simulate network topologies and conditions, allowing users to predict application performance and play "what-if" scenarios prior to deployment. In addition to its software tools, Shunra also offers a hardware-based appliance that offers the capability to plug actual multimedia devices into it so that voice and video can be evaluated subjectively by users, rather than relying solely on numerical presentation. This enables performance to be evaluated against real applications as delay, packet loss, jitter, or reduced bandwidth is introduced.

Analytical Engines

NetRule 7.1, from Analytical Engines (www.analyticalengines.com), is another software modeling tool that allows users to simulate networks and predict application performance. According to the Analytical Engine website, NetRule costs far less than other predictive network applications on the market.

Application Simulation Tools

In the early days of networks, the network infrastructure was managed as its own entity, and the applications were managed separately. Typical IT organizations were often broken into three groups: Network Engineering/Support, Desktop Support, and Application Development/Support. However, the network has evolved to be more application-aware, and the applications have evolved to be more network-aware, creating organizational gray areas on one level, and requiring much more application intelligence in the testing of the network.

As a reflection of the trends toward the blending of applications and infrastructure, testing has evolved from simple single-protocol bit blasting at Layer 2 and Layer 3, to full-fledged stateful multiprotocol, multihost application simulation. Similarly, application testing has evolved from UI-oriented testing, such as that offered by Mercury Interactive and others, to approaches that are much more network-aware.

Application simulation is distinguished from protocol testing because it recognizes that protocols don't exist in a vacuum. Applications are increasingly IP based, complex, and changing at ever-faster rates. Real applications involve vendor proprietary extensions, multiple protocols, and multiple cooperating systems. Testing based on actual network traffic, which includes many not-quite-standard protocol extensions, is much more "real world" than testing based on whatever the standards bodies say should be on the network.

Real application traffic is much different from what is in those standards (after all, standards bodies focus on protocols; very few applications are standardized), so it's essential that application simulation start from reality, not from the ivory tower. Even if there were

a standard for applications, the configuration fingerprint of an application in one particular enterprise environment would be unique, and moreover, these configurations change frequently (as components are upgraded, new components are added, new code is rolled out, new users are defined, etc.).

Mu Dynamics (www.mudynamics.com) has taken an approach that is based on actual application traffic, which isolates the application from the transport. The key thing that differentiates application simulation from bit blasting, or from packet replay, is that Mu's approach involves re-creating the same stateful application flow over a dynamically created transport. Application state is maintained by tracking important session variables (cookies, tokens, initial values, sequence numbers, call IDs, etc.) throughout the session, so it is indistinguishable from a session created by a real client device (or server device). In fact, many applications depend on concurrent usage of multiple protocols, involving multiple cooperating systems.

Application simulation testing in these dynamic environments must capture and represent this unique configuration so that the test harness is as close as possible to the behavior of real clients talking to real servers. This is the core reason why Mu Dynamics' Test Suite turns packets into test cases, within an automation framework that magnifies the productivity of test teams by at least an order of magnitude. Application simulation depends on accurately reproducing application flows so that the test traffic is indistinguishable from real application traffic.

Security Testing Tools

Security and penetration test cases are often required components of hardware and software certification test plans. There are several security and penetration test tools available, some of which are marketed commercially, others developed and distributed as freeware by "hackers" on the Internet. In either case, your company's information security team and your code of business conduct should be consulted before any security tool is used on your network.

The best known and most commonly used security tools are referred to as port scanners. Port scanners, such as the open source and free Nmap utility, operate by exploring an address range for active hosts using ping (ICMP ECHO and REPLY) packets. Once active hosts have been identified, they are scanned for open TCP and UDP ports, which can provide indication of the network services operating on that host. A number of these tools support different scanning methods and have different strengths and weaknesses; these are usually explained in the scanner documentation. For example, some are better suited for scans through firewalls, and others are better suited for scans that are internal to the firewall. Some port scanners are capable of gathering additional information, such as the target operating system, using a method known as system fingerprinting. For example, if a host has TCP ports 135 and 139 open, it is most likely a Windows host. Other items such as the TCP packet sequence number generation and responses to ICMP packets (for example, the Time To Live, or TTL, field) also provide a clue to identifying the operating system. Operating system fingerprinting is not foolproof; firewalls can be

configured to filter certain ports and types of traffic, and system administrators can configure their systems to respond in nonstandard ways to camouflage the true operating system.

In addition to discovering a target operating system, some scanners are also capable of discovering whether host applications that leverage well-known TCP ports are running. For example, if a scanner identifies that TCP port 80 is open on a host, it can be assumed that the host is running a web service. Even more sophisticated scanners are capable of identifying which vendor's web server product is installed, which can be critical for identifying vulnerabilities that can be exploited. For example, the vulnerabilities associated with Microsoft's IIS server are very different from those associated with Apache web server. This level of application identity can be accomplished by "listening" on the remote port to intercept the "banner" information transmitted by the remote host when a client (a web browser in this example) connects. Banner information is generally not visible to the end user; however, when it is transmitted, it can provide a wealth of information, including the application type, application version, and even operating system type and version. Again, this is not foolproof, because a security-conscious administrator can alter the transmitted banners. The process of capturing banner information is sometimes called banner grabbing.

Many rudimentary security tests on network gear consist of conducting a port scan to check that only the expected ports—those that have services active such as Telnet or SSH—are open. For these basic requirements, a freeware tool such as Nmap should be adequate. When doing more complete or specific security and vulnerability testing, there are several good freeware and commercially available tools on the market. These tend to be much more complex and can require some security and scripting expertise to operate.

Packet-crafting tools such as HPING can be used to assemble and send custom packets to hosts, with the goal of obtaining information from the received replies. These kinds of tools are often used when trying to probe hosts behind a firewall by spoofing a well-known application's TCP port. A tool such as HPING can also be used to create malformed protocol packets in an attempt to launch a denial-of-service (DoS) attack. One common example of such an attack would be to send fragmented BGP packets to a router with the intention of creating a software crash.

Yersinia (www.yersinia.net) is a network tool designed to take advantage of some weaknesses in different network protocols. Attacks for the following network protocols are implemented:

- Spanning Tree Protocol (STP)
- Cisco Discovery Protocol (CDP)
- Dynamic Trunking Protocol (DTP)

- Dynamic Host Configuration Protocol (DHCP)
- Hot Standby Router Protocol (HSRP)
- IEEE 802.1Q
- IEEE 802.1X
- Inter-Switch Link (ISL) Protocol
- VLAN Trunking Protocol (VTP)

You can use a tool like Yersinia to harden your Layer 2 design and implementation.

Other security tools that can be used during testing are common password crackers, wireless scanners, and DoS tools, just to name a few.

Network Protocol Analysis Tools

These tools are often referred to as "sniffers." The best-known one is a free tool called Wireshark (formerly known as Ethereal, found at www.wireshark.org/), which can decode hundreds of protocols, including OSPF, BGP, SNMP, Telnet, and just about any other protocol your production network may have running. Sniffers used with a spanned port on a Cisco switch are invaluable tools for troubleshooting.

Writing the Test Plan

After you and your client have agreed upon the scope of the prototype and the test suites to be carried out, it is time to write a plan that describes exactly how you will test them. A test plan should address the following topics, which will be described in detail in the next few sections of this chapter:

- Overall project scope and objectives
- Test objectives and success criteria
- Test resources required (people, hardware, software, test tools)
- Test schedule
- Developing detailed test cases

Overall Project Scope and Objectives

A brief description of the overall project scope serves as a primer for stakeholders who are unfamiliar with the triggers and motivations for the testing project, in addition to guiding the testers' efforts as they create meaningful test cases. The following are some examples of specific project objectives that were written for particular customers:

- First Integrity Financial plans to build two new data centers in 2010 and is in the process of selecting a networking vendor with the best possible solution. Based on the customer's requirements for the new data centers, the account team has proposed a design that will be proof of concept tested in the Independent Network Services testing facility. Results of the tests will be presented to First Integrity as input into the vendor selection process.
- Spacely Sprockets is in the process of building a next-generation WAN to meet an increased, intergalactic demand for its superior bicycle components. A low-level design developed by Spacely Sprockets' network architects will be verified in the Independent Network Services testing facility to ensure that any weaknesses or limitations are found prior to deployment. Findings during the test effort will be documented and sent to the architects for LLD refinement.

Test Objectives and Success Criteria

Test objectives and s uccess criteria should be developed based on a client's business and technical goals for the network design, and they should include any known SLAs associated with applications or services. The test objectives should simply be to measure the outcome of the test case, and they should be based, as much as possible, on industry standards for all relevant technologies and services. For example, VoIP quality can be measured quantitatively using a Mean Opinion Score.

A MOS score is a subjective test of a call quality that was originally designed by the Bell Companies to quantify the quality of a voice call, with 1 being unacceptable and 5 being superlative.

This information will help the test plan developer define relevant test cases with clearly identifiable success or failure metrics that can be agreed upon by the tester and client. The following are examples of test objectives and success criteria that were written for a particular customer:

- Measure the response time for the Trading application Delta when the network path is 45 percent loaded, which is the average estimated load during trading hours. The acceptance criteria, per the SLA, for Trading application Delta is that the response time must be 300 ms or less.
- Measure the throughput for the Trading application Delta when the network is 90 percent loaded, which is the peak estimated load during a failure scenario in the primary path. The acceptance criteria, per the SLA for Trading application Delta, is that the throughput must be at least 1 Mbps
- Measure the impact to test traffic when various components in the WAN path are failed over. The availability SLA for Trading application Delta specifies that less than .1 percent loss be encountered on a flow running at 1000 pps during a failover event.

Test Resources Required

The people, hardware, software, and test tools necessary to complete the test should be included in the test plan for resource estimation, test build guidance, and historical recording purposes. It is very important to accurately document the exact hardware and software versions of the components that will be tested, as even small variations in hardware or software versions can produce different results with certain test scenarios. This information will provide a valuable baseline should operational issues occur further down the road.

Table 4-2 is an example of how equipment details can be captured in the test plan.

Table 4-2 *Example Hardware Equipment to Be Tested*

PE Router—Generic Configuration

Product	Description	Qty
XR-12000/10	Cisco XR 12000 Series Original Router	4
12410	Cisco XR 12000 Series 10-Slot Router	1
12416	Cisco XR 12000 Series 16-Slot Router	1
12816	Cisco XR 12000 Series 16-Slot Router	1
12406	Cisco XR 12000 Series 6-Slot Router	1
XR-PRP-2	Cisco XR 12000 Series Performance Router Processor 2	5
12000-SIP-601	Cisco XR 12000 and 12000 Series SPA Interface Processor-601	11
SPA-1X10GE-L-V2	Cisco 1-Port 10GE LAN-PHY Shared Port Adapter	2
XFP-10GLR-OC192SR	Multirate XFP module for 10GBASE-LR and OC192 SR-1	2
SPA-2X1GE-V2	Cisco 2-Port Gigabit Ethernet Shared Port Adapter	5
SPA-8X1GE-V2	Cisco 8-Port Gigabit Ethernet Shared Port Adapter	1
SPA-8X1FE-TX-V2	Cisco 8-Port Fast Ethernet (TX) Shared Port Adapter	3
SPA-4XOC3-POS-V2	Cisco 4-Port OC-3 POS Shared Port Adapter	4
SFP-GE-S	1000BASE-SX SFP (DOM)	4
GLC-T	1000BASE-T SFP	16
SFP-OC3-IR1	OC-3/STM-1 pluggable intermediate-reach 15 km trans	4
SPA-10X1GE-V2	Cisco 10-Port Gigabit Ethernet Shared Port Adapter	3

If applicable, it is also a good idea to provide per-node details of how the line cards are to be installed in modular node chassis. This will assist with the test build and remove any ambiguity regarding the exact hardware that was tested if questions arise during test results analysis. Figure 4-2 shows an example of an equipment slot configuration diagram that can be added to the test plan.

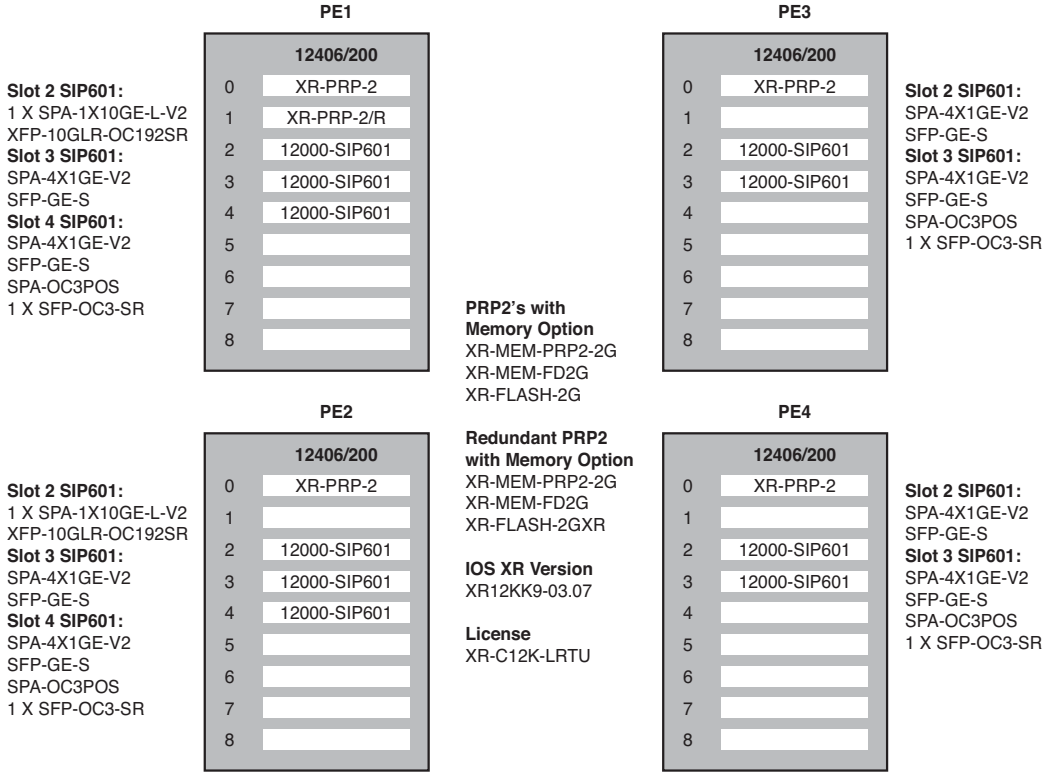


Figure 4-2 *Equipment Slot Configuration Diagram*

The exact software feature set and version should be recorded for each device type and role in the network, as shown in Table 4-3.

Table 4-3 *Example Software Versions to Be Tested*

Platform	Role	Cisco IOS Software Version	Image/Feature Set
2811	CE Router	12.3(14)T7	c2800nm-adventerprisek9-mz.123-14.T7.bin
2821	CE Router	12.3(14)T7	c2800nm-adventerprisek9-mz.123-14.T7.bin
4500/Sup III	L3 Switch	12.2(25)	cat4000-i5k91s-mz.122-25.EWA14.bin
4500/Sup 6E	L3 Switch	12.2(46)	cat4500e-entservicesk9-mz.122-46.SG.bin
C3750	L2 Switch	122-25.SEB4	c3750-ipbase-mz.122-25.SEB4.bin

Large test organizations often tackle several projects simultaneously, some of which are long term, requiring a team approach. An estimate of the resources allocated to a particular test should be included in the test plan, as shown in Table 4-4.

Table 4-4 *People, Roles, and Time Allocation*

Role	Name	Resource Allocation
Program Manager	Cosmo Spacely	As required
Test Manager	George Jetson	25%
Test Lead	Joseph Barbara	100%
Test and Documentation	Henri Orbit	100%
	George O'Hanlon	50%

Test Schedule

A test schedule designates work to be done and specifies deadlines for completing milestones and deliverables. Test entrance and exit criteria should be clearly defined so that everyone understands what tasks must be completed prior to the start of testing, and when testing is considered to be complete. An example of test entrance criteria may be that a client must approve the test plan, at which point no more changes will be allowed without a redefinition of the test scope. Test exit criteria may include running all of the planned tests, identifying or filing bugs for any defects found, and/or reviewing test results with the customer.

Table 4-5 shows a sample test schedule.

Table 4-5 *Sample Test Schedule*

Date	Milestones	Deliverables/Comments
10/1/2009	Test Plan Start	High-level test case review with customer and account team
10/5/2009	Test Plan—Review & Approval	Test Plan document review with customer and account team
10/6/2009	Entrance Criteria (EC) Approval	Project Execution Commit with sponsors
10/6/2009	Test Start	Dependent on test entrance criteria documented in EC
10/13/2009	Test Complete	Completion of all test cases
10/20/2009	Test Result Report Complete	Final test results report complete
10/23/2009	Internal Test Document Review	Review test document with internal team prior to customer review
10/26/2009	Test Document Review with Customer	Customer review of test document
11/2/2009	Lab Topology Teardown	Test Project complete

Developing the Detailed Test Cases

As explained earlier, test cases are the essence of the test plan, as they ultimately will be followed to produce results that will determine whether the device, feature, or system under test has passed or failed. As the test plan writer, you must be very concise when specifying the set of preconditions, steps, expected output, and method of data collection that should be followed. This is particularly important when the people executing the tests have not been involved in the development of the test plan, or are working on several different tests concurrently. When the time comes for test execution, engineers need to understand

- What they are testing
- Why they are testing it
- How they are going to test it
- What information they need to capture
- The format in which they need to record results

Test cases are often classified as being either formal or informal.

Formal test cases can be directly mapped to test requirements with success criteria that are measurable through quantifiable metrics. Formal test cases have a known input and an expected output, which are worked out before the test is executed. For example, a formal test case could be developed to verify a vendor's claim that a particular firewall product can support 64,000 concurrent connections. The expected output might be that the platform should be able to forward traffic at a particular pps rate with the specified preconditions that it must be performing stateful inspection and filtering on 1 to 64,000 sessions. This type of formal case would be considered a "positive" test. A "negative" test could similarly be defined where the number of concurrent sessions was gradually increased above 64,000 at a rate of 1000 per second so that the effect on packet forwarding rate, CPU, memory, and general device health could be observed and measured. Formal test cases such as this should be linked to test requirements using a traceability matrix.

For features or network services without formal requirements or quantifiable success criteria, test cases can be written based on the accepted normal operation of features or services of a similar class. For example, an informal test case could be written to demonstrate the capability of a WAN acceleration appliance (such as Cisco WAAS WAE) to improve performance on a particular TCP application. As there are no industry standards that quantify "WAN acceleration," the informal test case could simply measure the time it takes to transfer a file via FTP from a remote server with, and then without, WAN acceleration enabled. The expected output could simply be that the time to retrieve the file should be "less" when WAN acceleration is enabled, which would then be recorded as a benchmark.

Understanding System Test Execution Methodologies

Chapter 1 introduced a four-phased approach to systems testing that has proven to be effective in replicating a customer's network design and in modeling application traffic characteristics. This approach includes developing a comprehensive set of test cases categorized as baseline, feature, negative, or scalability.

This section introduces a few common test methodologies that can be used to help develop test cases for each phase. These include conformance tests, functional and interoperability tests, and performance and scalability tests.

Conformance Testing

Conformance testing is used to verify compliance with standards and is often a key component of network hardware and software certification test plans. These types of tests are often challenging to develop because many network protocols are difficult to implement consistently between different vendors. Despite the existence of RFCs and IETF standards, implementations often have subtle differences because the specifications are typically informal and inevitably contain ambiguities. Sometimes there are even changes in implementation between different code levels within the same vendor's products.

Conformance tests are usually made up of both positive and negative test cases to verify how network devices comply with specific protocol standards. Conformance testing tools perform their tests as a dialog by sending protocol-specific packets to the device under test, receiving the packets sent in response, and then analyzing the response to determine the next action to take. This methodology allows conformance test tools to test complicated scenarios much more intelligently and flexibly than what is achievable by simple packet generation and capture devices.

When conducting conformance tests, keep in mind that even the test tool makers must interpret an RFC, and, as mentioned earlier, there may be differences in implementation between the test tool and the network equipment under test. If you see discrepancies, record them and work with the vendors to find a feasible workaround. Often times, these differences have been seen before.

A BGP conformance test plan is provided in Chapter 6, "Proof of Concept Testing Case Study of a Cisco Data Center 3.0 Architecture," as an example.

Functional and Interoperability Testing

Functional and interoperability tests are geared toward evaluating specific device features as they would be implemented in a "realistic" setup, and as such these tests are commonly seen in POC and design verification testing. Interoperability testing is a critical aspect of testing IP services that determines if elements within the architecture interact with each other as expected, to deliver the desired service capability. In contrast with conformance testing, which provides proof of RFC-defined protocols working between a few devices, generally two tests—functional and interoperability—allow engineers to expand the test coverage from a simple, small lab setup, to a more realistic, real-world configuration.

Functional and interoperability testing is the determination through a larger systems test of whether the behavior of a network architecture, in specific scenarios, conforms to the test requirements. For example, when you enable that QoS feature on your WAN edge network, will it reduce jitter for your voice traffic, or will it cause CPU spikes, fill your interface queues, and cause your routing protocol to drop? In this type of test, you will have multiple features enabled and competing for resources.

Functional and interoperability testing is often conducted as part of baseline testing, where all of the network features are enabled together. Only when all the features that will be working in conjunction in your network are combined with all of the types of hardware and software you will be using, will you be able to have a real view of how they will all interact together. Using the preceding QoS example, the routing protocol may work perfectly by itself, and the QoS policy may be doing exactly what you expect; but when you combine them together with the correct Cisco IOS Software and hardware, as well as some SNMP polling, you may see an issue. This combination of complex features, hardware, and software is what functional and interoperability tests are all about.

While the functional and interoperability tests do not specifically test for conformance, they sometimes help you identify conformance issues. For example, if you connect a new

router to an existing lab network, you may find that the OSPF neighbors end up stuck in the Exstart/Exchange State. This problem occurs frequently when attempting to run OSPF between a Cisco router and another vendor's router. The problem occurs when the maximum transmission unit (MTU) settings for neighboring router interfaces don't match.

Performance and Scalability Testing

Performance and stress tests take the architecture to the next level. Assuming that everything is working as expected in your test environment under various test scenarios, including negative or failure tests, the next question is how well the network will work under different scenarios with an increased traffic load. There are many performance metrics you should collect, as well as stress scenarios that you should try out, before the network is deployed into production and required to support revenue-generating traffic.

Performance and stress tests are actually two different things. In performance testing, you are trying to create a baseline for how the network will behave during typical and increased loads, as well as during failover scenarios. The goal of performance testing is to find and eliminate bottlenecks and establish a roadmap for future regression testing. To conduct performance testing is to engage in a carefully controlled process of measurement and analysis, until you hit a predetermined threshold, be it CPU, memory, interface utilization, or something else.

Stress testing, on the other hand, tries to break the system under test by overwhelming its resources or by taking resources away from it, in which case it is sometimes called *negative testing*. The main purpose behind this is to make sure that the system fails and recovers gracefully, as well as to find the point at which the system will become inoperable.

When conducting a performance test, you would want to see, for example, how long it takes a router to bring up 15 OSPF neighbors each advertising 1000 routes. In a stress test, you would check how many OSPF neighbors advertising 1000 routes would cause the router to start behaving incorrectly. Both of these types of testing tend to require very expensive and extensive test gear.

Format for Written Test Case

There are several articles written, and even commercial software products available, to help you develop written test cases. While there is no absolute right way to write a test case, experience and best practices suggest that it should be written clearly, simply, with good grammar. It is recommended that the following information should be included at a minimum:

- **Test ID:** The test case ID must be unique and can also be associated with the test logs and other collected data.
- **Node List:** The list of the actual hardware being tested in this test case.
- **Test Description:** The test case description should be very brief.

- **Test Phase:** Baseline, Feature, Negative, or Scalability.
- **Test Suite:** If applicable, include the feature or service that this test case will be used to verify. Examples may include OSPF, QoS, High Availability, or VoIP.
- **Test Setup:** The test setup clearly describes the topology, hardware, logical configurations, test tools, applications, or other prerequisites that must be in place before the test can be executed. For complex tests, it is often helpful to include a diagram to help illustrate exactly how the test should be set up.
- **Test Steps:** The test steps are the step-by-step instructions on how to carry out the test. These should be very detailed so that testers with minimum experience can execute the tests.
- **Expected Results:** The expected results are those that describe what the system must give as output or how the system must react based on the test steps.
- **Observed Results:** The observed results are those outputs of the action for the given inputs or how the system reacts for the given inputs.
- **Pass/Fail:** If the expected and observed results are the same, then the test result is Pass; otherwise, it is Fail.

Summary

This chapter covered the motivations for different types of testing and the best way to scope and plan a well-thought-out test. The chapter also went over some of the most often used test tools and where they should be used in your testing. Finally, this chapter explained how to write an effective test plan. The next chapter provides examples of written test cases and gives you tips to help you be successful in your testing.

This page intentionally left blank

Index

Numerics

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q-R

S

T

U

V

W-X-Y-Z