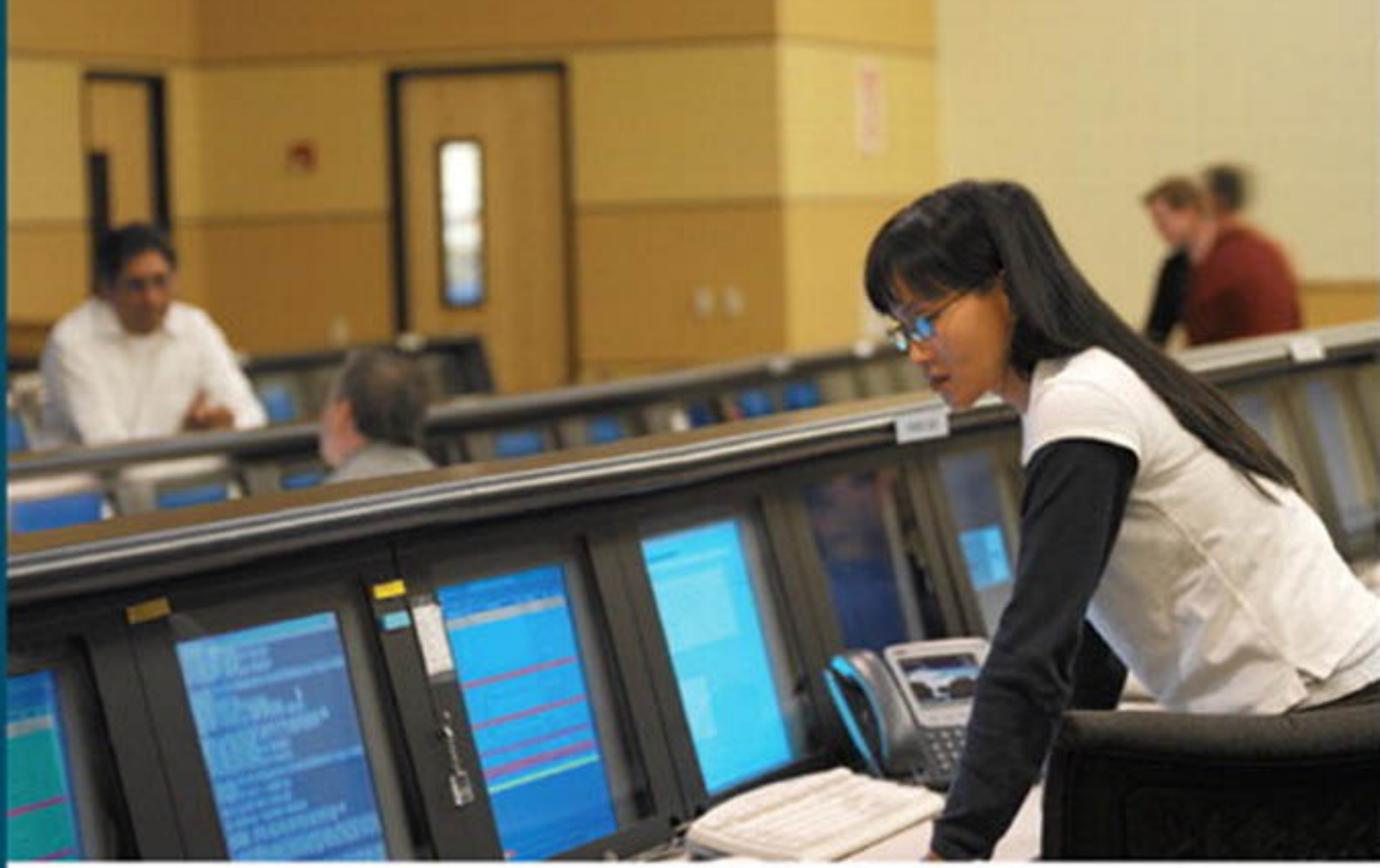
Network Management Fundamentals

A guide to understanding how network management technology really works



ciscopress.com

Alexander Clemm, Ph.D.

Network Management Fundamentals

Alexander Clemm, Ph.D.

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 November 2006

LIBRARY OF CONGRESS CATALOG CARD NUMBER: 2004110268

ISBN: 1-58720-137-2

Warning and Disclaimer

This book is designed to provide information about network management. 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.

Corporate and Government Sales

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

For sales outside of the U.S. please contact: **International Sales** 1-317-581-3793 international@pearsontechgroup.com

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Readers' feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book or otherwise alter it to better suit your needs, you can contact us through e-mail at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Trademark Acknowledgments

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

and "There are flames," it is much more efficient to send one correlated message that says "The kitchen is on fire." The correlated alarm might still contain references to the original, uncorrelated, "raw" alarms, in the rare case that this information is still needed. It might also be marked as a correlated alarm so that an end user can distinguish between the conclusions drawn by the alarm correlation function and the original alarm data.

Correlation can have varying degrees of sophistication. Simple forms of correlation can occur at the level of the managed device (for example, if a card fails, let the device suppress alarms indicating that its ports have failed as well). More complex forms of correlation might involve sophisticated algorithms, inference engines, or expert system technology. The use of the term *alarm correlation* easily raises expectations that highly sophisticated and complex correlation is performed, whereas in reality simple forms of correlation are far more common. In fact, *correlation* can be considered an overused term. In many cases, it is incorrectly applied to refer to any function that reduces the volume of alarms, even if that function is not a correlation but perhaps simply a filtering function.

Note that alarm correlation is different from root cause analysis, although, again, sometimes both terms are used liberally and interchangeably. Alarm correlation focuses on identifying which alarms are likely different symptoms that are all related to the same root cause, without actually identifying the root cause that initiated the symptoms. Its goal is to intelligently filter and reduce the amount of alarm information that is reported. The correlated alarm information still must be analyzed for what caused it. This is precisely the subject of root cause analysis.

Fault Diagnosis and Troubleshooting

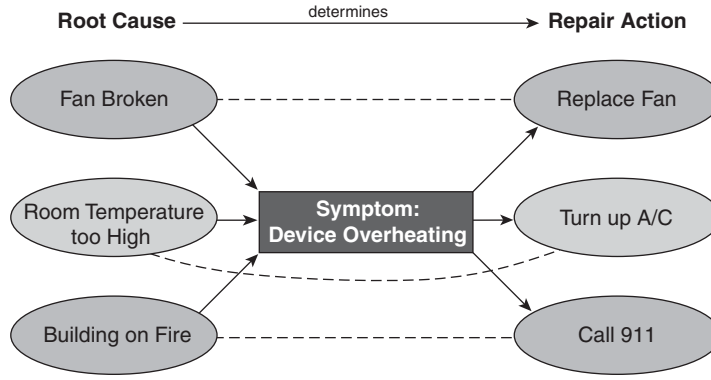
Alarm management is a significant aspect of fault management—so significant, in fact, that the two terms are often used synonymously. However, there is more to fault management than alarms. One other aspect concerns fault diagnosis and troubleshooting.

Network diagnosis is conceptually not much different from medical diagnosis. The difference, of course, is the type of patient. To reach a medical diagnosis for a set of symptoms (for example, a rash), the doctor might want to take a look at additional monitoring data (for example, by taking the patient's temperature and blood pressure) and might conduct his or her own series of tests, such as testing the reflexes or asking the patient to breathe deeply while listening with a stethoscope.

When a fault occurs in a network, the capability to diagnose the problem—that is, to quickly identify what caused it, is key to minimizing its impact on users. The proper diagnosis then is the basis for selecting the proper repair action. The analysis process that leads to a diagnosis is often also referred to as root cause analysis. An alarm generally alerts you only to a symptom, not what caused it.

For example, assume that you receive an alarm "Device overheating," as Figure 5-6 illustrates. How do you find out what actually caused the alarm? Was it because the device fan failed? Is the room temperature in general too high? Or is the building on fire? Of course, you might simply walk over to the device and check for yourself. But remember that you might be sitting in a network operations center 50 miles away and have to diagnose the problem remotely. And only after it has been properly diagnosed can you determine what the proper repair action should be: Should you dispatch a technician to replace the fan? Do you need to turn up the air-conditioning? Or should you call 911?

Figure 5-6 *Symptom, Root Cause, and Repair Action*



Diagnosis is often supported by troubleshooting functions. Troubleshooting can involve simply retrieving additional monitoring data about a device, data that was not conveyed as part of the alarms. In addition, the capability to inject tests into a network or a device for troubleshooting purposes provides essential support for diagnosis activities. With networks, there are many examples of such tests: For instance, loopback tests are common in telecommunications. Those tests involve setting up a connection to a remote endpoint that is automatically "looped back" to where it originated—short-circuited, if you will. By comparing data that is sent and received over the looped connection, important conclusions can be made. For example, loopback tests can be used to verify that communication paths are indeed intact. As a side benefit, they can also be used to measure certain quality-of-service parameters, such as delay. Likewise, phone calls might be generated to test voice connections.

Tests can be used not only in troubleshooting after a problem has already occurred, but also proactively, to be able to recognize any fault conditions or deterioration in quality of service before it becomes noticeable to a user. The best fault management, after all, is to avoid faults altogether.

Proactive Fault Management

Most fault management functionality, such as alarm management, is, by nature, reactive—it deals with faults after they have occurred. However, proactive fault management is also possible—that is, taking steps to avoid failure conditions before they occur. This includes, for instance, the previously mentioned injection of tests into the network to detect deterioration in the quality of service and impending failure conditions early, before they occur. Proactive fault management can also include alarm analysis that recognizes patterns of alarms caused by minor faults that point to impending bigger problems.

Trouble Ticketing

Another problem to mention concerns management of the fault management process itself, from detection to resolution of problems. A larger network might easily serve tens of thousands of users. In such networks, it is possible for hundreds of problems requiring follow-up to occur daily. Hopefully, none or only very few of the problems will be catastrophic in the sense of large-scale network outages. Nevertheless, individual users might still be experiencing problems that are serious enough for them, such as sluggish network response time or loss of dial tone. Given the scale of today's networks, it is quite easy to lose track of things.

Trouble tickets are one way in which a network provider organization can keep track of the resolution of network (or service) problems that typically require human intervention. Those problems might have been reported by the network itself through certain types of alarms, or they might have been reported by a customer experiencing a problem. When certain problems are encountered or reported by users, a trouble ticket is issued to describe the problem. Trouble tickets are assigned to operators, who are responsible for resolving the trouble ticket—that is, taking care of the problem. The trouble ticket system helps keep track of which trouble tickets are still outstanding. It can automatically escalate a problem if it is not resolved in time. The system can also help communicate a problem between different operators by automatically attaching the entire history of the problem and its resolution to the trouble ticket.

Not every alarm results in a trouble ticket because issuing that many tickets would quickly overwhelm operations personnel. Instead, trouble tickets are issued generally only when the reported alarms and other observed conditions indicate that the capability to deliver service could be affected, and for alarm conditions whose resolution likely requires human intervention that the network provider organization needs to track.

C Is for Configuration

We now turn to the second letter in FCAPS, *C*, which stands for configuration management. For the network to do what it is supposed to do, it might need to be first told what to do—that is, configured. This is similar to having to initially set up a VCR so that it tunes to the proper channels, to select the proper input for connections from a video console, and later needing to program the

VCR to record a particular show. Depending on the type of network equipment, its configuration can be much more involved than that of a VCR. In addition, in a network, you might have a large number of devices, all of which need to be configured in a coordinated manner to be capable of singing in tune, so to speak.

Configuration management includes functionality to perform operations that will deliver and modify configuration settings to equipment in the network. This includes the initial configuration of a device to bring it up—that is, to be properly connected to the network—as well as ongoing configuration changes. For example, to provide a new employee with phone service in an enterprise network, the network needs to be configured so that it will recognize the new user's phone number and be capable of directing calls to that phone, as well as ensure that the collection of billing records associated with the new user is turned on so that his department can be properly charged.

Performing configuration operations alone is not enough; you also need to keep track of what you have in your network. The write operations must be complemented by read operations, so to speak. Although in a small network keeping track of what's in it seems trivial, as you start scaling your network to thousands or tens of thousands of devices and users, it becomes more difficult—how do you know that all equipment is really where you expect it to be? How can you be sure that a user did not unplug one of your routers and plug it in somewhere else, altering your physical network topology that had been fine-tuned to offer well-balanced performance? Or what if someone simply connected another piece of equipment on his own, unwittingly making the network vulnerable to attacks?

By the same token, you need to also know what has been configured—for example, what services are running over which equipment, and which users are associated with the equipment—so that you know who might be affected if you need to perform maintenance operations. Accordingly, configuration management also includes auditing the network to retrieve its current configuration and making sure that the management system's information about the network is current.

Configuration management is at the core of setting up a network so that it can deliver service; it is really at the core of network management in general. Configuration management is fundamentally tied to provisioning and to fulfillment—but those are functions used in other categorizations of the management function space, namely OAM&P, as well as Fulfillment, Assurance, Billing (FAB), discussed later in this chapter. Without effective configuration management, a network provider will have a hard time keeping track of what is actually deployed in a network or providing even basic functions such as turning up a service. However, other management functions depend on configuration management as well. For example, in fault management, many networking problems cannot be properly diagnosed without accurate knowledge of the network's configuration.

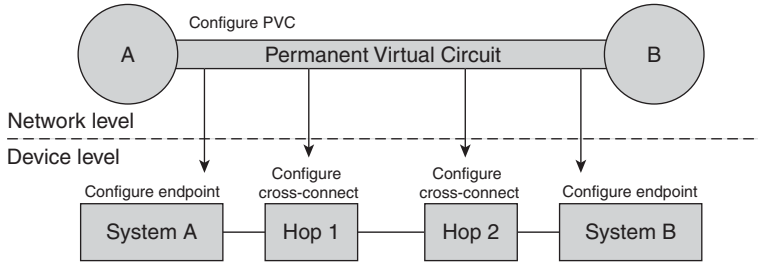
We dive into configuration management functions in more detail in the following subsections and cover the following topics:

- Configuring managed resources, whether they are network equipment or services running over the network
- Auditing the network and discovering what's in it
- Synchronizing management information in the network with management information in management applications
- Backing up network configuration and restoring it in case of failures
- Managing software images running on network equipment

Configuring Managed Resources

At the core of configuration management are the activities and operations used to configure what is being managed. Ultimately, this involves sending commands to network equipment to change its configuration settings. In some cases, this involves only one device in isolation, such as configuring an interface on a port. In other cases, configuration operations that are performed on the devices are simply part of a bigger operation at the network level that involves changing the configuration of multiple devices across the network. An example is setting up a connection across the network, such as a static route or an ATM permanent virtual circuit (PVC). This requires configurations to be performed on each hop along the connection to, in essence, cross-connect incoming and outgoing interfaces along the path, as Figure 5-7 illustrates.

Figure 5-7 *Network-Level vs. Device-Level Configuration*



Above the element and network management layers, configuration management also includes functionality to perform configurations that are necessary for the network to provide a service for an end user—the managed resource, in this case, is simply the service. Configuration management at the service level is generally referred to as *service provisioning*, borrowing terminology from the OAM&P reference model that we discuss in the next section.

Provisioning a service involves being able to turn up the service, to modify certain service parameters, and to tear it down. The latter aspect is often forgotten but is just as important as setting up the service. For example, if an employee leaves your company, you do not want that employee to still have access to the company's VPN. Likewise, if you are a telecommunications service provider and have a customer who isn't paying, you want to be able to cut off his service.

It is important to be able to describe the service in terms that relate to the service, not in terms that relate to the network over which the service is provisioned. For example, you might want to be able to order a service that provides a new employee, John, with VPN service, e-mail with a mailbox of certain size, and phone service with voice mail, call forwarding, but no authority to place international calls. It is up to a service provisioning application, not an end user, to break down the instruction to configure this particular type of service into the detailed configuration operations that need to be sent down to the networking equipment so that the service can go into effect. For example, the application would need to assign a phone number and configure the voice-mail servers, e-mail servers, switch ports, and IP PBX accordingly. The capability to provision services rapidly, correctly, and efficiently is of utmost importance to service providers and their competitiveness: Being able to roll out services faster decreases the time to collect revenue and could therefore actually increase revenue. In addition, it minimizes operational cost and increases customer satisfaction.

Auditing, Discovery, and Autodiscovery

Being able to configure your network is important, but not enough. You need to also be able to query the network to find out what actually has been configured—you need a read in addition to the write. This is referred to as *auditing*. Many reasons exist for auditing devices in the network. For example, you might want to verify that the configuration of the network is indeed what you expect it to be. You might want to see if configuration commands that you sent down indeed took. Without this function, a service provider would have a very hard time understanding what is going on in a network and why it is going on.

Closely related to auditing devices for configuration data is querying devices for other data that is not related to configuration. This includes information about the current state of the device as well as performance data, such as the number of packets that are currently being dropped or the current use of device ports. The basic mechanisms to query nonconfiguration data on the device are generally the same as for configuration data. The only difference is that, in the case of configuration data, the queried data is in general persisted on the device (stored in nonvolatile memory or on hard disk), whereas this normally is not the case with state information. State information will not survive a reboot, for example. However, retrieving nonconfiguration data is