CHAPTER 5

Management Functions and Reference Models: Getting Organized

This chapter picks up right where we left off in our discussion of management dimensions in the previous chapter. Specifically, it takes an in-depth look at the function dimension of network management, a big topic that deserves its own chapter. This concerns the range of functionality that management applications and operational support systems need to cover. We discuss these functions along the lines of several management *reference models*, which do a great job of organizing these functions. Reference models are conceptual frameworks, introduced for the purpose of organizing in a systemic manner the different functions of a system or a technology—network management in this case.

After reading this chapter, you will be able to

- Explain what the most established functional reference model, FCAPS, stands for and what it consists of
- Outline the wide range of management functions in network management and explain what these functions entail
- Describe the OAM&P model, an alternative functional reference model that is popular with telecommunications service providers
- Explain the limitations of reference models
- Describe how different functional reference models relate

Of Pyramids and Layered Cakes

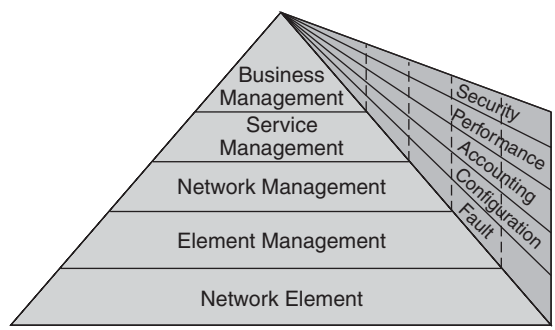
As mentioned previously, management reference models serve as conceptual frameworks for organizing different tasks and functions that are part of network management. The emphasis is on the word *conceptual*. In reality, reference models are, in many cases, not literally followed—management systems and operational support environments can be structured in different ways

that, for various reasons, do not reflect the same breakdown in functionality as suggested by any particular reference model. However, a reference model can be used for guidance and helps provide a sense of orientation in the following ways:

- It makes it easier to check a management system or operations support infrastructure for completeness. It forces the person applying the model to differentiate the different tasks that must be addressed. For the reference models presented here and for the purposes of this chapter, this aspect is the most important.
- It helps categorize and group different functions, and identify which ones are closely related and belong together and which ones do not.
- It helps to identify scenarios and use cases that need to be collected, and to recognize interdependencies and interfaces between different tasks. ("Use cases" are a part of use case analysis, a software engineering methodology that is used to derive functional requirements for systems by analyzing in a systemic manner different ways in which the system might be used—hence the term *use case*.)

A few reference models have been widely established. One of them is the Fault, Configuration, Accounting, Performance, Security model, commonly referred to as FCAPS (pronounced: "eff-caps"—rhymes with *snaps*). As its name indicates, it divides management functions into five categories—fault management, configuration management, accounting management, performance management, and security management. It is actually part of a larger management reference model that we introduced in the previous chapter: the TMN reference model. As mentioned earlier, the TMN reference model covers much more than just management layers; the FCAPS categorization of management functions is one of the concepts that it introduces. We can redraw the TMN pyramid from Figure 4-5 with more refinement, as in Figure 5-1, to show the functional dimension in addition to the layering. Whereas the layering dimension of the TMN model was discussed in the previous chapter, this chapter takes an in-depth look at its functional dimension.

Figure 5-1 *TMN Reference Model Refined with FCAPS*



Another reference model is the Operations, Administration, Maintenance, and Provisioning (OAM&P) model. Here, management functions are categorized a bit differently. Ironically, although the FCAPS model originated from TMN and was intended to standardize network management for telecommunications, large telecommunications service providers traditionally favor the OAM&P model. We take a look at this model as well.

Although these are the most prominent reference models, they are not the only ones. Network management can be organized in a thousand ways. Still, discussion of these reference models teaches important lessons regarding established ways to think about network management. Even more important for the purposes of this chapter, discussing reference models provides a great opportunity to explain in detail the different functions that are associated with managing a network.

TMN and similar models are sometimes criticized as being overly complex—looking like multilayered cakes of many slices when perhaps a doughnut would do. However, in many cases, much of this criticism stems from an improper understanding of what these models are all about. A number of points should be kept in mind when considering reference models:

- A reference model is conceptual—that is, an abstract partitioning of a problem space. In general, there is no need for an actual system to follow the structure of a reference model literally.
- A specific application or operational support system may be designed with very specific constraints in mind. A reference model, however, has to be generally applicable and cannot be optimized for any one specific case. Otherwise, there would be a risk that the model might break in other scenarios—that is, the model might lose its generality.
- Generally, it is advantageous to be able to slice up a problem space for many of the same reasons that make component-based systems more attractive than monoliths. Different functions can always be combined later; breaking up is what's hard to do.

FCAPS: The ABCs of Management

To get a handle on the wide range of management functions that are required in an operational support environment, people often group them into a set of broad categories that are known as Fault, Configuration, Accounting, Performance, Security (FCAPS). This is the categorization that we use to go over the various functions.

In many cases, function categories can be addressed independently of each other, in terms of both the systems supporting them and the organization performing these functions. For instance, fault management activities such as monitoring, diagnosing, and troubleshooting devices are very

different in nature from configuration management activities that deal with the configuration and turn-up of devices.

The subsections that follow introduce each of the FCAPS function categories in more detail. The subsections contain a number of enumerations that might appear lengthy but that are intended to convey the breadth of the spectrum of functions associated with the management of networked systems. The functions are presented mainly from a user perspective—functionality that is at the disposal of a service provider organization and network managers within that organization.

F Is for Fault

Fault management deals with faults that occur in the network, such as equipment or software failures, as well as communication services that fail to work properly. Fault management is therefore concerned with monitoring the network to ensure that everything is running smoothly and reacting when this is not the case. Effective fault management is critical to ensure that users do not experience disruption of service and that when they do, disruption is kept to a minimum.

Fault management functionality includes but is not limited to the following:

- Network monitoring, including basic alarm management as well as more advanced alarm processing functions
- Fault diagnosis, root cause analysis, and troubleshooting
- Maintaining historical alarm logs
- Trouble ticketing
- Proactive fault management

Network Monitoring Overview

Network monitoring includes functions that allow a network provider organization to see whether the network is operating as expected, to keep track of its current state, and to visualize that state. This functionality is fundamental to being able to recognize and react to fault conditions in the network as they occur.

The most important aspect of network monitoring concerns the management of alarms. Alarms are unsolicited messages from the network that indicate that some unexpected event has occurred, which in some cases requires operator intervention. Those unexpected events can actually be about anything—from a router that detects that one of its line cards is no longer working to (literally) a fire alarm, from a sudden drop in signal quality on a wireless link to a suspected intrusion into the network by an unauthorized user. Like other alarms in the real world, an alarm in a network might actually set off a bell just like a fire alarm or result in the automatic dispatch of operating

personnel, similar to alarms set off by home intrusion-detection systems. In most cases, however, they simply result in a message being sent to a management system, which lets an application or an operator decide what to do.

Alarm management is such a large area that the term is sometimes used synonymously with *fault management*, although there is more to fault management than alarms. Alarm management includes many functions that we classify into basic functions, such as alarm collection and visualization, and more advanced functions that involve processing alarms to perform filtering and correlation tasks.

Basic Alarm Management Functions

We start the discussion of alarm management with the more basic functions—collecting alarms, maintaining accurate and current lists of alarms, and visualizing alarms and network state.

The most basic and, at the same time, most important task—the task that everything else builds on—consists of simply collecting alarms from the network and making sure that nothing important is missed. This includes receiving the alarms and storing them in memory so that they can be further processed by an application or a human operator who decides how to react. Basic alarm management functionality also includes persisting alarms—that is, writing them to disk or storing them in a database, to build a historical record of alarms that occurred.

In more sophisticated cases, alarm collection can include additional, more advanced mechanisms that can check that no alarms were lost and that can request replay of alarms, as long as the network provides such capabilities. In practice, alarms can be lost in many ways, even when this is not supposed to happen—after all, in most cases, the event that is being alarmed was not supposed to happen, either. For example, the underlying transport might not be reliable and the alarm information might thereby be dropped on the way to the management application. Another reason alarm information might fail to reach its destination is that the network is congested and alarm messages simply cannot get through (remember that this situation was one of the selling points for a dedicated management network). In a third scenario, the alarm information might actually have reached the host of the management application but was still not properly collected because the application or database was not functioning properly or was being restarted when the alarm message arrived.

After alarms have been collected, an accurate list of current alarms needs to be maintained. This list answers the questions, which current conditions in the network require management attention, and which issues are currently being experienced with the services provided by the network. The alarm list also informs the operator of the current state of each of the various entities that are being managed and whether a particular device, for instance, is having problems. To maintain the alarm list, it is not sufficient to simply append alarms to the list as they occur. Alarms can also clear—

that is, the underlying condition that caused the alarm can be resolved. The current list of alarms needs to be updated accordingly, removing alarms when they no longer apply.

It is also important to understand how the alarms and indeed the network state are visually presented to the user. In its most basic and most important form, visualization can occur simply through textual lists. Each alarm results in an entry in the list, containing information about the alarm. Those lists can be searched, sorted, and filtered according to many different criteria, such as alarm severity, the type of alarm, the network element (or range of network elements) affected, the type of network element affected, the time of day when the alarm occurred, and many more.

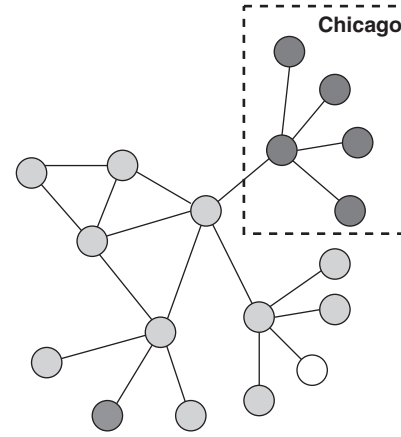
However, visualization can occur in many ways. One popular method uses topology maps. Icons on the map represent devices and can be animated to indicate the current alarm state. The alarm state corresponds to the highest severity of any alarm condition on a device, of which there might be several. Likewise, connections can be represented on the topology map, as indicated by lines that connect the icons. Like the icons, the lines can be animated with different colors to indicate the current state of that particular connection. Not all alarms indicate a complete failure of a device or the services running over it. In fact, in most cases, they do not. The severity of an alarm indicates the degree of impact of its underlying alarm condition—the higher the severity, the greater its impact on the network, services, and end users. For example, red might be used for devices on the map with a critical alarm, orange for major alarms, yellow for minor alarms, and green for no alarms. Gray might be used to indicate lost management connectivity to the device. Please note that red would be inappropriate in the case of lost connectivity because the device itself might in fact be functioning properly. Of course, icons can be used not just to represent individual devices, but to represent groupings of devices that can be "zoomed into." In that case, the color for the grouping of devices reflects the most severe alarm of any of the devices contained within it.

Topology maps can make monitoring systems look attractive and, hence, become good sales tools for those systems. In addition, they provide a good overall picture of the health of a network and a good way to indicate geographical "hot spots" and identify where the real problems lie. If everything in area Chicago suddenly turns red, including the single access router through which all those devices are connected to the core network, chances are, the resolution for the problem lies with that router, and that's where the troubleshooting should start. As Figure 5-2 illustrates, this way the graphical representation of topology on a map makes information much easier to correlate than if the same information were merely represented in a flat list of alarms, interspersed with alarms from other parts of the network.

Figure 5-2 *Visualization of Alarm Information (a) Through a List and (b) Through a Topology Map*

Node	Sev	Time	Event	Info
ruby	cr	16:00:42	sysdn	…
jbee	cr	16:00:42	sysdn	…
M3660-sjs	mn	16:00:33	qostc	…
M3660-sjn	mn	16:00:25	l0exc	…
Pep-7600	mj	16:00:20	dropn	…
txsouth	cr	16:00:05	sysdn	…
blubber	cr	16:00:05	sysdn	…
Hlee-7569	cr	16:00:04	pwrfl	…
snorkel88954	cr	15:59:58	sysdn	…

(a)



(b)

Many reasons exist for maintaining historical alarm data in addition to the list of current alarms. This requires simply logging and archiving alarms as they occur, which is actually simpler than maintaining the current list. After all, items are appended only at the end of the list. Historical alarm data is not required for monitoring the network but is useful in many other ways. For example:

- Historical alarm data can be mined to help with future diagnosis and correlation (we come to these topics in a moment). Basically, this can be helpful to identify alarm patterns that have occurred in similar form on past occasions. Recognizing such patterns and recalling their past resolution can help resolve future problems faster.
- It can be used to establish trends, to see how alarm rates and types of alarms reported have evolved over time.
- It can be analyzed in conjunction with other historical data, such as changes that have been performed on the network—for example, the introduction of new network elements—and its impact on historical alarm patterns, or correlation of alarms with certain usage patterns of the network.

Advanced Alarm Management Functions

Beyond those basic alarm management functions, in any network of meaningful size, additional functions to manage alarms are required.

Some of those functions provide network managers with greater flexibility in processing alarms. For example, an alarm-forwarding function might send alarms to the pager of an operator to allow

for an automatic dispatch, much as a home intrusion detection system automatically calls the local police station.

Another function allows network operators to acknowledge alarms, meaning that they confirm that they have seen the alarm and are taking care of it. The function might allow network operators to open a *trouble ticket* (as first mentioned in Chapter 2, "On the Job with a Network Manager," and explained in more detail later in this chapter) that is based not on customer complaints, but on event messages from the network that point to trouble.

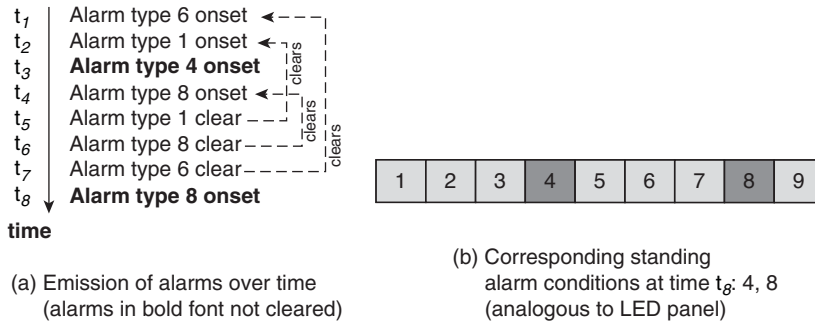
A third function handles the clearing of alarms: recognizing when an alarm is no longer current or, more precisely, matching failure onset and failure remission conditions. Most alarms have an underlying alarm condition (if they do not, they belong to a special category of alarms called *transient alarms*). An alarm message is sent to report the onset of this alarm condition. At some later point in time, a second alarm message might be sent to indicate that the alarm condition no longer exists.

It is important to maintain at the management system level accurate lists of the current, or *standing*, alarm conditions, without needing to query the network device for what those conditions are. More often than not, the device does not have such a query capability. Besides, it would be a bad idea to continuously poll the device for information that can be derived from information that it has sent already.

You can think of the alarm condition in terms of a conceptual panel of light emitting diodes (LEDs), one for each distinct alarm condition that exists on the device. (Of course, because the multitude of alarm conditions that exist, providing a comprehensive panel is impractical in reality because it would require too many LEDs.) LEDs light up when the condition comes into effect and remain lit while the condition holds. LEDs go off when their underlying alarm condition no longer holds. An alarm is sent whenever a LED just went on—an operator might not be watching the LED panel all the time because there are so many to watch in the network. Likewise, an alarm clear is sent to indicate that the LED went off again. The list of standing alarms is simply the list of LEDs that are currently lit.

Figure 5-3 illustrates this concept. The left side of the diagram depicts a chronological list of alarm messages. Some messages indicate the onset of an alarm condition of a certain type; others indicate the remission of the same condition—the alarm has cleared. The list of current alarms includes only those messages that have not cleared—that is, those for which no matching "clear" indication has been received. The right side of the diagram depicts a fictitious LED panel that indicates which of the alarm conditions are current, corresponding to the alarm list.

Figure 5-3 *Alarms and Alarm Conditions*



(a) Emission of alarms over time (alarms in bold font not cleared)

(b) Corresponding standing alarm conditions at time t_8: 4, 8 (analogous to LED panel)

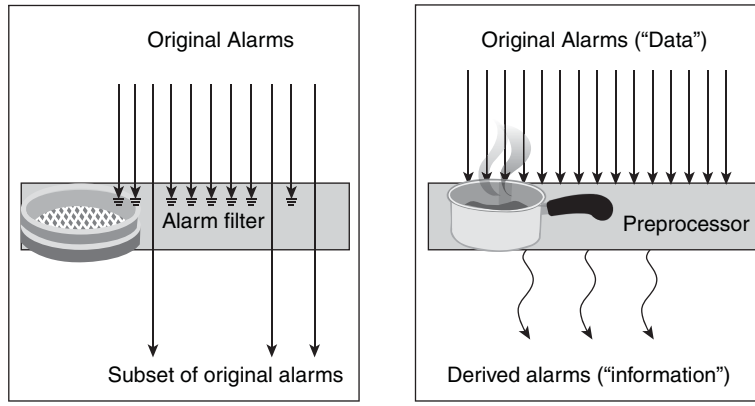
The concepts of clearing alarms and acknowledging alarms are often confused; sometimes "clearing of alarms" is mistakenly used to mean acknowledging an alarm. When you encounter these terms, make sure you understand how they are being applied.

When an alarm is cleared, it means that the underlying condition that caused the alarm has ended. This is different from acknowledging an alarm. *Acknowledging* merely means that the alarm was noted and, presumably, action is being taken. To use a physical picture, if the alarm sets off a bell that keeps ringing, turning off the bell corresponds to acknowledging the alarm. The problem might still be indicated by a LED that is still lit and that will remain lit until the underlying condition that caused the alarm ends. Only at that point will the alarm be cleared.

Another category of functions that is related to managing alarms concerns trying to reduce the amount of information that human operators and higher-level management applications are exposed to. It is possible in large networks, such as those of large telecommunications service providers, for hundreds of thousands of alarms to occur per day—so many alarms that alarm-processing capabilities of alarm management applications are often measured in hundreds of alarms per second. Of course, not every alarm indicates something major or an impending catastrophe of its own. Most are not really "alarms" in the narrower sense of the word, but event messages. Still, they need to be looked at. The problem is, therefore, how to reduce the volume of information that must be evaluated.

Generally two techniques deal with potential event information overload. One technique is filtering. Its goal is to remove event information that is deemed unimportant or redundant, to allow the receiver to focus on the more relevant event information. The other is correlation. Its goal is to preprocess and aggregate data from events and alarms, and distill it into more concise and meaningful information. Figure 5-4 illustrates and contrasts both techniques. We discuss each of these techniques in the sections that follow.

Figure 5-4 *Alarm Filtering vs. Preprocessing*



(a) Alarm filtering (b) Alarm preprocessing

Alarm and Event Filtering

Let us first turn our attention to filtering, not just of alarms, but of events in general. To focus an operator's or a management application's attention on those events that really matter, it is important to block out as many irrelevant or less important events as possible. This is analogous to the way in which the human brain is able to deal with the massive flow of data that it is constantly exposed to, such as sounds, visual images, and sensory data. To focus, it filters out massive amounts of data that would otherwise be distracting, for example, background noise when following a conversation.

One way to enable filtering is to allow users (operators or management applications) to subscribe only to those alarms and events that are of potential relevance to them and what they need to accomplish, as specified by some criteria. This way, users receive only events that meet those criteria. Here are some examples of using this technique effectively: Users might choose to subscribe only to alarms that involve a particular system or subsystem. For instance, they could be concerned with always ensuring that the company's CEO receives excellent communication service and, therefore, subscribe specifically to alarms that affect the port through which the company's CEO's office is connected. Users might also choose to subscribe only to alarms of a certain type. For instance, operating personnel for voice services might be interested only in alarms that indicate problems that are related to voice service. Finally, users might choose to receive only alarms that have a certain severity. They might decide to receive only critical alarms and to have everything else discarded (well, perhaps not discarded, but simply stored in a logfile so that it can be used for analysis when needed, as opposed to being brought directly to their attention). This could be important when high alarm volumes occur, so they can avoid the small stuff and ensure that high-impact items are dealt with.

Another way to filter alarm concerns *deduplication* of alarms. In some cases, the same alarm condition might cause the same alarm to be sent repeatedly. Because each new instance of the same alarm contains no new information, the new instances might simply be thrown away. The process of discarding the redundant alarms is referred to as deduplication. A similar scenario to which similar considerations apply is that of oscillating alarms. In that case, there is an underlying oscillating alarm condition, causing alarms to be sent and then cleared again immediately before occurring again in rapid succession multiple consecutive times. Although oscillating alarms relate to only a singe condition and are hence relatively easy to spot, they can lead to a high overall alarm volume that drowns out other events that are happening in the network. Therefore, the alarms should be turned off.

An infamous example concerns the "door open" alarm. Such an alarm can often be sent by equipment that can be installed in publicly accessible locations whenever a sensor detects that its door is opened. Having a door to a piece of equipment opened can indicate a serious problem because it could mean that an unauthorized person might be tampering with the equipment. The problem in this case is that thousands of alarms could be generated per hour when the sensor on a particular piece of equipment is faulty and mistakenly detects that the door is open, only to correct itself by reporting that it is closed, every other second. Until the faulty sensor is fixed, the oscillating alarms need to be filtered.

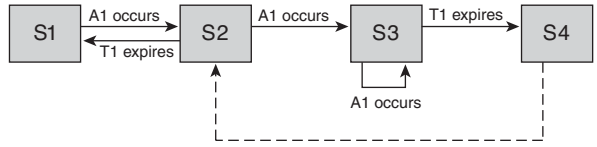
Of course, with oscillating alarms, it could still be useful to know the frequency with which oscillation occurs, or, with redundant alarms, how many duplicates there are. For example, is the door reported open three times in an hour? If so, the door might really have been opened three times because someone is in fact tampering with the equipment, perhaps while performing maintenance. Or is the door reported open 3,000 times in an hour, in which case a sensor or contact is probably bad? If the repeated occurrences of the alarm are simply filtered, this information is lost. A better solution is to record the information that duplicates or oscillations have been observed, along with how many there were, but to throw away the alarms themselves. Of course, if the rate at which oscillations or seemingly redundant alarms occur drops below a certain threshold, it might even be advisable to not filter those alarms at all.

For example, here is one technique that can be applied in the case of redundant alarms: The first occurrence of the alarm message needs to be forwarded without delay to the intended recipient. If duplicates of the same alarm occur, at least 1 minute should be allowed to pass before notifying the recipient of the same alarm again. At this point, the alarm message is sent again, annotated with a counter that tells the recipient how many instances of the alarm message have actually occurred.

Figure 5-5 also illustrates this. If alarm A1 occurs in the initial state, S1, of the system, it is forwarded immediately and another state, S2, is entered. If no more alarm A1s are received within the minute time period, the system reverts back to its initial state. However, if additional A1 alarms occur, they are not forwarded immediately. Instead, the system enters another state, S3, in which the duplicate counter is increased for each occurrence of A1. Eventually, the minute timer expires.

At that point, the system enters state S4, in which it sends the alarm A1 along with the count of its number of occurrences. It then immediately enters again state S2, waiting for more duplicates of A1 or, if no more are received within the minute interval, reverts back to the initial state.

Figure 5-5 *Deduplication of Alarms*



S1: Initial state; wait for A1 to occur
S2: Send A1; start timer T1; initialize duplicate count to 0
S3: Increment duplicate count by 1
S4: Send A1 annotated with number of occurrences

Of course, strictly speaking, we are now no longer simply filtering messages. Although it is true that we throw away many of the duplicates, we maintain a little counter for the number of occurrences and add this counter to the duplicate alarm message that we sent. This means that we have actually started to aggregate and preprocess information across alarm messages—what we have here is really a very simple form of *correlating* alarms, which leads us to the next topic.

Alarm and Event Correlation

Generally, alarm correlation refers to an intelligent filtering and preprocessing function for alarms. Alarm messages are intercepted and analyzed and compared to identify which alarms are likely related. For example, alarms could be related because they report the same symptom or because they probably have the same root cause. Depending on the sophistication of the correlation function, different aspects can be taken into account—information contained in the alarms themselves, context information such as knowledge of the network topology, or time context, such as the delay encountered between different messages.

The idea behind event and alarm correlation is that instead of forwarding and reporting many individual alarm or event messages, only a few (ideally, only one) messages need to be sent that aggregate and summarize the information from across multiple "raw" events. This way, the number of alarm messages that are reported to other alarm management applications and to human users can be significantly decreased, often by orders of magnitude. At the same time, the semantic content of those messages can be dramatically increased—that is, the actual information that is conveyed with each message. This prevents users from becoming overwhelmed and allows them to focus on the most relevant information instead of wasting their energy or processing cycles on alarms that could be easily discounted as noise. To give a simple analogy, instead of sending alarms "There is a funny smell," "The windows are fogging up," "Visibility is getting poor," "More funny smell," "It is uncomfortably warm," "It is really getting hot," "There is a crackling noise,"

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

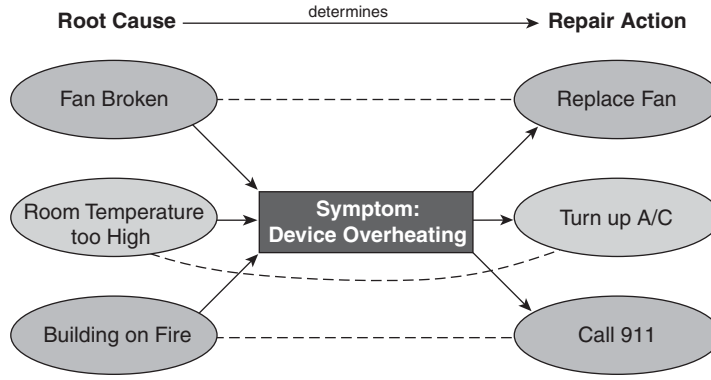
Alarm management is a significant aspect of fault management—so significant, in fact, that the two terms are often used synonymously. However, there is more to fault management than alarms. One other aspect concerns fault diagnosis and troubleshooting.

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.

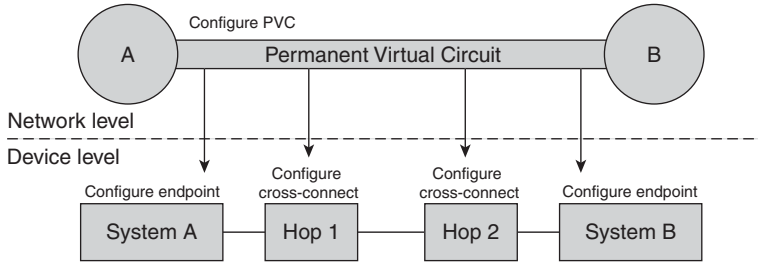
We dive into configuration management functions in more detail in the following subsections and cover the following topics:

- 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

typically associated with the other FCAPS functional areas, such as fault (used to retrieve monitoring data that helps in troubleshooting a problem), performance (used to collect statistics), or security (for example, to detect suspicious patterns in network usage that could indicate a denial-of-service attack).

In addition to auditing your network, you also might want to be able to discover what is in your network. The need for such a function might not be obvious at first. After all, if you as a network provider keep proper track of your network, you would not expect any surprises. However, discovery is still a very important function for many reasons. For example:

- Inventory records might not be accurate.
- Personnel might change things in the network and might not always record those changes properly.
- Discovering the network might be more efficient than having to enter the information about the network into a management application.
- Finally, depending on the management scenario, in many cases inventory records might not be available because keeping an inventory might not be appropriate in the first place. Consider, for example, system management scenarios that involve devices that are mobile or that roam across the network, with people moving managed end systems such as computer workstations, disconnecting and connecting them to the network at arbitrary locations. Trying to keep an inventory database with information on what is supposed to be connected where in the network might not be a good idea in such an environment. However, managing the network and monitoring those devices is still required, so the capability to discover them is important.

A word of caution: in some cases, auditing functionality is misleadingly dubbed as discovery. However, what is "discovered" in those cases is not a device or something unexpected whose presence in the network was previously unknown. Instead, it is already known that the device is there; information merely is retrieved about its configuration. To be able to refer to actual discovery functionality when the term *discovery* is already occupied for auditing functionality, the term *autodiscovery* is frequently used. So whenever you encounter the claim that an application supports functionality to discover a network, be sure to check that the term is not confused with auditing and that the functionality that is referred to is indeed discovery.

Synchronization

Each time you or your management application needs to know your network's configuration, you do not want to first have to audit the equipment or discover the network. That would be much too inefficient and slow. Instead, you expect your management system to maintain a cache of information about your network, probably stored in a database. At least, this is the case for information that is relatively slow to change, such as which equipment is deployed in the network and how it has been configured. After all, configuration information is not the same as state or statistical information that rapidly changes, in which case you have no choice but to retrieve it on demand when you need a real-time view. However, as with any cache, you run into the problem of how to ensure that your cache does not get stale—that is, how to ensure that the information in your management system is indeed an accurate reflection of the information in the network.

Therefore, functions are needed to help management systems maintain an accurate and consistent management view of the network. Those functions are fundamentally concerned with the notion that there are two representations of management information: the network itself and the management system's view of it. Whenever there are two views of the same information, the question arises how to keep them from contradicting each other and, if contradictions occur, how to resolve them—in other words, how to synchronize the information.

For synchronization to take place properly, a key question is which set of information should be considered the "master" of the information in question. The master is also referred to as the golden store.

- One view is that the network should be considered the master—the network ultimately is the reality, and this reality needs to be reflected by the management system (see Figure 5-8). The management information maintained by the management system is nothing other than a cache that needs to be kept from going stale. This is the more common approach and the approach that enterprises generally apply.
- Another view is that the management system should be considered the master, and the network needs to be built toward the information maintained in the management system (see Figure 5-9). A discrepancy between network and management system indicates that an error occurred in setting up the network and that the network is wrong—well, not wrong, but not what it is supposed to be. This approach is less common but can be found with large telecommunication service providers.

Figure 5-8 *Network as Golden Store*

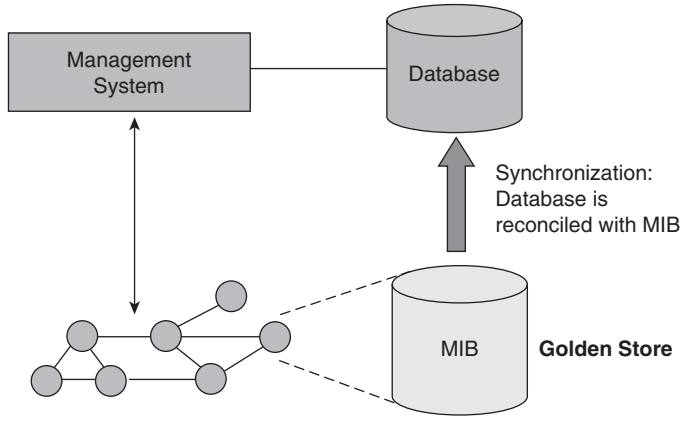
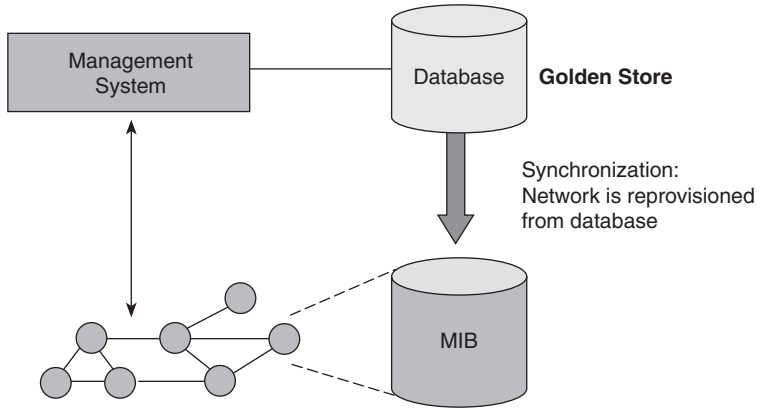


Figure 5-9 *Management System as Golden Store*



Depending on which view is taken, one of the following functions is used to synchronize management information:

- **Reconciliation**—The network is considered the master, and the information of the management system should reflect what is actually in the network. Information is therefore synchronized from the network to the management system (management information reflects the network as *built*). As mentioned, this view is the most common; hence, most of the time, synchronization of network information occurs through reconciliation.

- **Reprovisioning**—With reprovisioning, the management system is the master of management information; synchronization flows from the management system to the network, resulting in configuration changes to network devices as needed so that they reflect the information in the management inventory (management information reflects the network as *planned*). Until the network devices report that the appropriate changes have been made, the management system maintains a flag indicating that they are out of synch.

- **Discrepancy reporting**—With discrepancy reporting, discrepancies are simply detected and flagged for the user. The management application does not make a decision about the direction in which synchronization is to take place. This decision is the responsibility of the user and must be performed on a case-by-case basis. When the user decides that the management system should reflect the information that is out there, he will ask for reconciliation. When the user decides that he wants the configuration of the network to correspond to what is currently reflected in the information stored by the management system, he triggers a configuration operation.

Note that sometimes within the same network provider organization, both views of what should constitute the golden store are valid for different management functions: Monitoring certainly needs a view of what is actually in the network; the network, in this case, is clearly considered the master of management information. However, for network inventory functions in a large service provider, what is kept in the inventory should indeed be considered the master as the network is carefully engineered; network devices should not just "pop up" on the network, but should be the result of careful planning.

Of course, you need to keep track of things beyond information that is already reflected in the network—in addition to maintaining a cache of management information, there is a need for a true inventory of information that is nowhere reflected in the network but that is needed for management purposes. For example, you might want to keep track of the tasks you have assigned to the resources in your network, such as which services and end users they should support. This enables you to distinguish between network resources that have already been committed for a particular purpose and those that can still be assigned. With this information, you can avoid situations such as accidentally reassigning a port to a customer when it is already in use by someone else, or assigning the same IP address twice, which leads to all kinds of confusion and can disrupt service for existing users. In addition, keeping track of those assignments enables you to anticipate potential capacity shortages and react in a timely manner. For example, if you keep track of how network ports have been allocated, when the percentage of allocated ports exceeds a certain threshold, you will still have time to increase network capacity before being hit by a shortage. On the other hand, by knowing that sufficient ports have not yet been assigned, you can avoid overcapacities in the network and hence dead capital that would result from adding capacities too early.

Backup and Restore

If you are a PC user (and, chances are, you are), you are aware of the importance of protecting your data by performing regular backups. You never know when your hard disk will bite the dust or whether your PC will contract a virus that could destroy your PC's file system. Having a backup of your data in such cases enables you to recover. With backups in place, contracting a virus or needing to replace your hard drive is still annoying, but it beats by far simply being wiped out.

Likewise, the need for backup and restore functionality applies to your network. Here, your user data is not Word files or Excel spreadsheets, but the configuration data of your network. This data is very critical and needs to be protected, just as you would protect the accounting data and customer database of your company. Imagine some catastrophic event taking down a portion of your network and wiping out configurations, possibly affecting thousands of end users or customers, who might be getting more disgruntled and impatient by the minute. There would be no time to reconfigure network equipment one by one and reprovision every service. This would simply be too inefficient and would take too much time. Instead, the quickest, simplest, and most reliable way to bring things back up would be to simply restore your network to the last working configuration. As with PCs, having to restore the network is a function that you will hopefully never have to invoke. Still, it is a critical function—if you ever encounter a situation in which you need to restore a network, you will be glad to have such a capability in place.

Image Management

As with PCs, network equipment vendors occasionally issue new software revisions. Such revisions might be new feature releases, or they might simply be patches that contain bug fixes. In these cases, you need to be able to upgrade your network. The problem is that now you are not dealing with a single PC, but with hundreds or thousands pieces of equipment scattered across your network. To do so effectively, you need to be able to keep track of which software images are installed on which network devices, and have a way to deliver new images to those devices where the upgrade applies and install them without disrupting service. This functionality is referred to as *image management*. Despite the name, image management has nothing to do with managing your image in the public relations sense of the word; it involves managing software images running on network equipment.

A Is for Accounting

Organizations that offer communication services over a network ultimately need to generate revenue for the services they provide. After all, this is how they make their living. If they do not bill for the services they provide, they will not stay in business for long—notwithstanding some dotcom businesses that might give a service away but compensate for it through some other means, such as advertisements. Even if the organization is not a service provider but, say, an internal IT department providing those services to its own company, measuring the actual services provided and consumed is still required. This is necessary to be able to assess the cost/benefit ratio of

running those services, to keep cost under control relative to the services that are actually provided, and to use firm data for decisions on whether to perform services in-house or outsource them. After all, if an outside vendor could provide certain services as well as or better than your IT department at a lower cost, chances are, you will at least consider outsourcing.

Accounting management is all about the functions that allow organizations to collect revenue and get credit for the communication services they provide, and to keep track of their use. It is hence at the core of the economics of providing communications services. Obviously, accounting management needs to be highly robust; highest availability and reliability standards apply. After all, if accounting data is not properly collected, the service provider is actually giving away free services, translating directly into lost revenue.

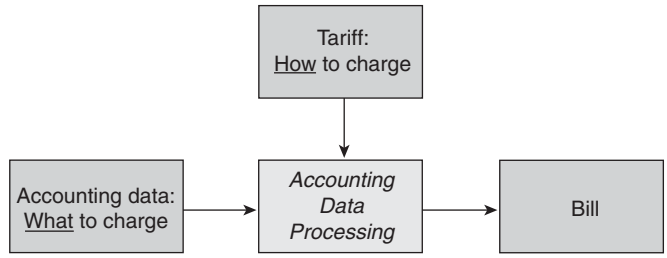
Earlier we used the analogy of network management and the medical field, comparing the diagnosis of faults in a network with the medical diagnosis of a patient. A cynic might say that the analogy extends to accounting management—after all, the hospital will want to send a bill as well.

On the Difference Between Billing and Accounting

Accounting management is often associated simply with billing, which is actually only one aspect. Billing is a common function that is performed for most businesses, whether they are rental car agencies, house-cleaning services, or restaurants. The business in this case is, of course, providing communication services. Writing the bills themselves, keeping track of customer data, and sending payment reminders is pretty similar for all these business. The domain specifics come in with regards to how to account for use of the service—that is, measuring what was consumed, by whom, and when. After all, unless you provide services at a flat fee ("all you can eat" or "all you can communicate"), you can send someone a proper bill only when you know what they have consumed, how much, and at what time. In other words, you need to *account* for the services and goods that your customer received.

Consider how you would account for rental car services—this would involve knowing the type of car, how many days the customer rented it, and whether the customer returned it with a full gas tank. Of course, this is not enough to write a bill. For this, you also need to know what tariff to apply. The tariff defines the rules on how to charge for the accounted services. In many cases, what to charge is determined not only by the actual service provided (in this case, the duration of the rental and the type of car); it might also depend on when the service was provided (weekday or weekend, for example) and to whom (regular or corporate or gold customer rate). Therefore, to produce a bill, the accounting data needs to be processed and the proper tariff needs to be applied to it. Figure 5-10 illustrates the relationship between accounting data, tariff, and bill.

Figure 5-10 *Accounting vs. Billing*



As indicated earlier, it is certainly possible to also simply charge a flat fee (think "all you can eat" in a restaurant, or "all you can communicate" in the networking context). Flat fees for networking services are not uncommon because customers often prefer simple and predictable pricing—think of flat-fee Internet service, for example. It also makes the task of billing easier for communication service providers, although it does not do away with the need to account for service usage entirely, as you shall see when we revisit flat fees in the section after next.

Accounting for Communication Service Consumption

To track the consumption of network services, meters must be set up that collect usage data. In the case of some services, usage data is automatically generated. For example, in the case of voice, call detail records (CDRs) are automatically generated by the network as part of call setup and teardown. Of course, these records still need to be collected, making sure that none are lost. In addition, because communication services often are provided across a network, duplicates must be eliminated. For example, if for a connection or a call the source and the destination each generate their own record, they need to be matched and consolidated into one.

In general, usage data is based on volume, duration, and/or quality. Examples of accounting measures are megabytes of data traffic, minutes of phone calls, number of service transactions, and use of premium or guaranteed services versus best-effort services. The data that needs to be collected must be put in terms and units that are relevant to the particular service and, hence, tend to be service specific. For example, to perform accounting for a voice service, it will not be interesting to know how many bits of voice payload were transported. However, the duration of a call is important. On the other hand, for a database-backup service the volume of data is very important, but not necessarily the duration of the backup. Sometimes other factors need to be taken into account, such as the distance of a phone call—although in recent years, the notion of distance has become much less relevant.

Accounting data is often collected only for offline processing. For example, this is typical if you send a subscriber a monthly bill. However, sometimes accounting data processing is also required in real time or near–real time. A good example is prepaid voice services: calling cards. The calling card customer can talk only as long as her minutes have been paid for. When the prepaid credit

runs out, the prepaid voice service provider will want to be able to disconnect the call. Of course, this imposes additional requirements and the need for a feedback cycle between the network that is providing the service and accounting management. In some cases, this blurs the line between management and control—a management function becomes a part of the communication service itself.

Although it should go without saying, it does need to be mentioned that it is not sufficient to merely measure communication service consumption; consumption also must be properly attributed to the user of the service. Therefore security functions, such as authentication to identify a user, often need to accompany and complement the collection of accounting data. This does not require users to provide a login and password each time; it can simply be based on the port through which a user connects to a service—a data port in an office or a phone jack in a home, for example.

Related to attributing communication service to the proper user is another important function of accounting management: fraud detection. Fraud detection is concerned with tracking down and preventing theft of communication services, such as unauthorized users hacking into a network to receive free Internet access or making free phone calls, or—worse—assuming the identity of legitimate users to steal services. Fraud is a big concern to communication service providers. It causes revenue leakage—that is, lost revenue for communication services that were provided but not paid for. It can also impede the quality of the service that legitimate users receive because communication resources are unexpectedly not available. And of course, no customer will accept being billed for services that were not actually received.

Accounting Management as a Service Feature

To simplify accounting and to simplify communication products, in many instances, flat-fee instead of usage-based models are offered. As mentioned earlier, flat-fee Internet service is one example. Of course, although flat fees ease some of the requirements of tracking precise use, other aspects, such as the need to attribute service use to authorized users—who are known to the service provider and not delinquent on their bills—remain.

In addition, accounting management can serve as an additional feature of the service itself, the very service that it provides accounting for. For example, viewing service use and billing information online makes a service more convenient and transparent, resulting in greater customer satisfaction and perceived ease in purchasing and paying for the service. The capability to view service use differentiated by different accounts of the same primary customer (example: wife, husband, and each of the children) constitutes additional service features that could be sold at a premium; at the same time, it opens up new ways to bundle service offerings and target specific market segments.

Flexibility in accounting management can lead to very sophisticated service offerings, such as having different charges for “family and friends” or different charges for calls that are made

between customers on the same network versus to customers on other networks (on-net and off-net calls), to name a few examples. These are examples taken from telephony services that can be commonly found today but that were made possible only by advances in accounting management.

P Is for Performance

When you buy a car and look at different choices, you assume that the cars you are looking at can all transport you from point A to point B. Each of the choices might also offer automatic transmission, power door locks, air-conditioning, and perhaps even a navigation system. However, those functional properties alone might not tell the whole story, and you might even take them for granted. To make a decision, you also take a look at nonfunctional properties, most important of which is performance. Does the car accelerate from 0 to 60 mph in 5 seconds, or in 25? Does it get 40 miles per gallon, or only 10? The point is, performance makes a big difference, and it is no different with communication networks.

Performance Metrics

Performance of networks is characterized by a multitude of performance characteristics, measured according to metrics. Some examples of performance metrics are these:

- Throughput, measured by a number of units of communication performed per unit of time. The units of communication depend on the layer, type of network, and networking service in question. Examples are as follows:
  - At the link layer, the number of bytes, or octets, that are transmitted per second
  - At the network layer, the number of packets that are routed per second
  - At the application layer for a web service, the number of web requests that are serviced per second
  - At the application layer for a voice service, the number of voice calls, or call attempts, that can be processed per hour

  As a side note, closely related to throughput is utilization. Whereas throughput is an absolute number (such as number of bytes per second), utilization is a relative number that expresses throughput as a percentage of the theoretical maximum capacity of the underlying system.
- Delay, measured in a unit of time. Again, you can measure different kinds of delay, depending on what layer or networking service you are dealing with. Examples are as follows:
  - At the link layer, the time that it takes for an octet that is transmitted to reach its destination at the other end of the line
  - At the network layer, the time that it takes for an IP packet to reach its destination

  - — At the application layer for a web service, the time that it takes for a request to reach its destination at the host servicing the request after the request has been issued
  - — At the application layer for a voice service, the time it takes to receive a dial tone after you have lifted the receiver

- Quality is in many ways also performance related and can be measured differently, depending on the networking service
  - — At the link layer, the number or percentage of seconds during which errors in transmission occurred
  - — At the network layer, the number or percentage of packets dropped
  - — At the application layer for a web service, the number or percentage of web requests that could not be serviced
  - — At the application layer for a voice service, the number or percentage of voice calls that were dropped or abnormally terminated

As the examples point out, the same performance concept (such as throughput and delay) can be applied at different layers of the communication hierarchy. It should be mentioned that what is measured at each layer is nevertheless fundamentally different and not just a matter of which unit is applied—for example, whether throughput is expressed in kilobytes or megabytes per second. Instead, what is measured at each layer is different, and the measurements observed at one layer give no indication of what might be observed at a different layer. For example, the number of bytes transmitted at the link layer provides no indication of the number of voice calls that are successfully serviced at the application layer, nor can they be computed from one another.

Monitoring and Tuning Your Network for Performance

Performance management deals with monitoring and tuning your network for its performance. This includes a wide variety of functions.

At the most basic level, you want to be able to retrieve a snapshot of the current performance. This corresponds with taking a look at the speedometer of your car to see how fast you are going. Of course, in the case of network management, the speedometer is replaced by packet counters, delay measures, and gauges that indicate utilization percentages.

For a more sophisticated analysis, you might want to observe some of these parameters over time. For example, you might want to plot a histogram of some performance values on a screen, with a new sample taken (and point plotted) every second, or every 5 minutes, or whatever time interval suits you. Doing this gives an absolute reading of a particular value, and you also can observe how the values change over time. This way, you will be able to distinguish between a sudden drop or spike in value from a value that is within the ordinary.

Some patterns might indicate that a problem is about to occur—for example, an increase in utilization of an interface might precede an increase in the number of packets that are dropped, which, in turn, might precede users experiencing application sessions timing out. Monitoring the performance often allows you to anticipate problems and take care of them before they occur.

When observing the values over time, you might be able to determine a trend—whether the utilization continues to go up, for example. In this case, you can get a head start on planning for an upgrade. You might be able to spot bottlenecks in your network—areas that seem to be constantly congested, as well as areas that seem to be underutilized, where equipment might be put to better use elsewhere. All this can be valuable information for adjusting and tuning your network configuration to get the optimum performance and value from your equipment.

Collecting Performance Data

When you sit in front of a screen, you can monitor the performance of only a very small portion of your network—for example, of a hot spot where there appears to be a problem. However, you might be interested in recording performance data from all over the network, even if you cannot constantly monitor it. It can sometimes be useful to have the option of looking at the data later if you discover a problem, to see if there are any indications in the data of how the problem developed, or to just use the data for general analysis. In many cases, such analysis does not have to occur in real time; it is even possible to perform the analysis offline. This means that you need to collect statistical performance data. Periodic snapshots need to be taken and stored somewhere in a file system or database.

Constant polling of performance data from devices can quickly bring a management system to its knees, not to mention the network and devices being polled. Imagine that you have a network with 10,000 devices, and you are interested in 10 performance parameters on each. If you wanted to collect data on a per-minute basis, it would require 100,000 polling cycles per minute! Fortunately, there are more intelligent ways of collecting performance data.

One popular way of obtaining performance data is by having it reported as what amounts to a stream of events—for example, using protocols such as Netflow or IP Flow Information Export (IPFIX). This way, the request to poll performance data is no longer required.

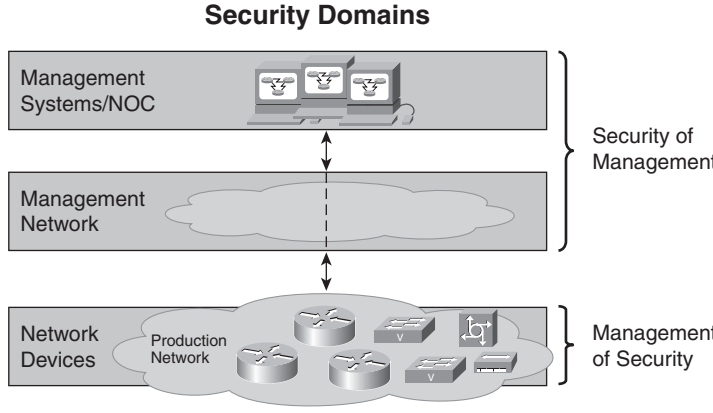
Another option is popular whenever the collection of performance data does not have to occur in real time: The data collection is simply set up at the device. A management application tells the device what type of performance data it is interested in. Internally, the device then takes a snapshot of this data over predetermined time intervals, such as every 15 minutes starting on the hour. The device logs this data in a file on flash memory or hard disk (if it supports this function, it will almost certainly have one). This collects the data into "buckets," dripping in an additional drop of data at every time interval until the bucket is emptied (that is, retrieved by the management application) or until it is full. Once a day, perhaps in the middle of the night, when the overall processing load is low, the management application retrieves the files containing the performance

data from the devices. Then management applications that know how to crunch large volumes of numbers go over those files, trying to establish trends or whatever else a user is interested in.

S Is for Security

The final letter in FCAPS, "S, stands for security—that is, management aspects that are related to securing your network from threats, such as hacker attacks, the spread of worms and viruses, and malicious intrusion attempts. Two aspects need to be distinguished: security of management, which ensures that the management itself is secure, and management of security, which manages the security of the network. Those aspects are depicted in Figure 5-11 and are explained in detail in the following subsections.

Figure 5-11 *Security of Management vs. Management of Security*



Security of Management

Security of management deals with ensuring that management operations themselves are secure. A big part of this concerns ensuring that access to management is restricted to authorized users.

For example, access to the management interfaces of the devices in the network needs to be secured to prevent unauthorized changes to network configurations. Also, the management network needs to be secured to prevent disruption to management traffic.

In addition, access to the management applications themselves needs to be secured properly—devices generally authorize on the basis of a management application, not on the basis of the user of a management application. Therefore, securing access to the management interfaces and management network without securing the management applications is akin to locking the door of your house to keep thieves out but leaving the windows open. Clearly, improper access to management applications can cause considerable damage. After all, if you can use those

applications to modify configurations of devices in the network to provision services and to tune network performance, you could also abuse them to disrupt services, degrade network performance, or provision services illegitimately to give users who are not authorized (or have not paid) access to the network.

Note that management needs to be secured not only against attacks from the outside. You need to also account for the possibility that security breaches occur from within. Accordingly, although managing access privileges properly is a necessary ingredient to ensure secure management, it is not sufficient by itself. Another important function concerns maintaining tamper-proof security audit trails that record any management operations that are performed on the network. If mechanisms to safeguard the network and its management fail, an audit trail enables you to reproduce what has actually happened, possibly identify culprits, and more easily recover from the security breach.

As a general rule, security threats from the inside are harder to defend against than threats from the outside. However, by performing the following tasks, you can go a long way in defending against the worst threats and preventing disruptions to the operation of your network:

- Set up proper processes and procedures to ensure orderly operations
- Assign access privileges only to those who actually need these privileges for their immediate job function
- Require "secure" passwords that cannot easily be cracked
- Require that passwords be changed at regular intervals
- Establish audit trails, themselves secured properly
- Set up proper facilities for backup and restore of critical management data

Management of Security

Management of security involves managing security of the network itself, as opposed to security of its management. Unfortunately, as we all know, in these days, online security threats are all too common. In many cases, security threats target not so much the network, but devices connected to the network—PCs of end users, for example, or systems that host websites for corporations. In addition, the network infrastructure itself might come under attack. Common security threats include but are by no means limited to the following:

- Hacker attacks of individuals who try to obtain improper control of a system that is connected to the network.

- Denial-of-service (DOS) attacks that try to overload portions of a network by generating illegitimate traffic, preventing legitimate network traffic from getting through. A variant is distributed denial-of-service (DDOS) attacks, which coordinate those attacks from multiple sources, making them harder to defend against.
- Viruses and worms that attempt to corrupt and possibly destroy systems along with their file systems, which are connected to the network or which are network devices themselves. Related to this are Trojan horses, malicious code that masquerades as a useful and innocent program that, when opened by a user, can wreak havoc.
- Spam, also considered a security problem because its volume can overwhelm a network and its servers.

Management of security provides functions to deal with and protect against these and other security threats. This involves some of the same functions that provide security for management, such as ensuring that management interfaces of network devices are not open to people from the outside, as well as maintaining security audit trails that record all operations—and attempted operations—on network elements.

In addition, management of security involves other functions. All of those functions can be components of a comprehensive security management strategy. For example:

- Intrusion detection involves monitoring traffic on the network to detect suspicious traffic patterns that could indicate an ongoing attack. One technique that can help guard against the spread of viruses involves inspecting traffic payload to see what is carried inside it, and then discarding or marking content that is apparently intended to compromise the network's security. Methods that involve inspection can sometimes be ineffective, however, because in many cases payload is transported over secure connections that are encrypted. In those cases, inspection fails because it cannot decrypt the traffic that is being transported.
- Another technique that can help protect a network consists of applying policies that limit or allow to only gradually increase the amount of traffic that is geared toward a particular destination or that originates from a particular source. If an attack resulted in a sudden burst of traffic, this technique allows for a more graceful degradation of the network and its services if they come under attack.
- The capability to "blacklist" ports and network addresses at which suspicious traffic patterns are observed and through which suspected offenders may enter the network constitutes another important safeguard. Those ports and addresses can be put under additional scrutiny and monitored for suspicious activity so that they can be quickly shut down if an attack is suspected.

- So-called "honey pots" are a more recent technique to gather information about security vulnerabilities in a network to help better defend it. A honey pot is a piece of equipment or a host system that appears to be a part of the regular network but that, unbeknownst to the attacker, is actually isolated and specially secured. It serves as a trap. Because the honey pot is not an actual part of the production network, any traffic that is directed at the honey pot can with reasonable certainty be regarded as malicious. Analyzing such traffic yields important information about attacks on the network and allows you to better fend off such attacks.

Limitations of the FCAPS Categorization

The notion of FCAPS is tremendously useful in providing a simple framework that is easy and intuitive to understand. It provides structure to discussions of management functionality and establishes a common terminology. However, it is important to note that it also constitutes somewhat of an oversimplification. Many cases of functionality cannot be easily categorized because they can be used for different purposes that fall under different function categories.

One example is the capability to test the functioning of a given service, often used for troubleshooting—in other words, fault management. However, there are other uses of the same capability, such as to validate that provisioning steps have had the desired effect (configuration management), or to use the same test to simultaneously take performance measurements (performance management).

Another example concerns the capability to log and report events—that is, messages that are emitted by network devices. This capability is generally associated with fault management because it clearly relates to alarms. However, it can also support other FCAPS management function categories as follows:

- Performance management, such as when the crossing of a certain threshold is reported, perhaps when utilization reaches a certain level
- Configuration management, such as when events indicate certain changes in the network's configuration
- Security management, such as in conjunction with security-related events, perhaps unsuccessful logon attempts into network devices or activities that smack of fraud

The following sections examine some other ways to organize management functionality.

OAM&P: The Other FCAPS

The previous section described at length the various functions that are provided by management organized along the FCAPS model. Although FCAPS is probably the best-known functional reference model, it is by no means the only one. Management functions can also be organized in

other ways. Of course, the functions that ultimately need to be performed remain the same, independent of how they are categorized. What might change is the way in which those functions are grouped and organized, the way in which the functions need to interface with each other, the way in which information flows, and (if mapped to an actual network provider organization) the way in which responsibilities are assigned.

An alternative to the FCAPS categorization of management functions is known as *OAM&P*—Operations, Administration, Maintenance, and Provisioning. The OAM&P model is popular in particular with large telecommunications service providers, whose internal organization OAM&P often reflects much better than FCAPS, which is more popular with enterprises and data providers.

Without reiterating the individual management functions that we discussed earlier in the chapter, the OAM&P categories cover the management ground as follows:

- Operations involves the day-to-day running of the network—specifically, coordinating activities among administration, maintenance, and provisioning as required. It also includes monitoring the network to ensure that things run properly, although, in many cases, monitoring activities are also conducted as part of maintenance. This further illustrates the point that, in the end, any categorization is somewhat arbitrary, and different functional organizations might work best for different network providers.

- Administration covers the support functions that are required to manage the network and that do not involve performing changes (configuring, tuning) to the running network itself. Administration includes activities such as designing the network, tracking network usage, assigning addresses, planning upgrades to the network, taking service orders from end users and customers, keeping track of network inventory, collecting accounting data, and billing customers.

- Maintenance includes functionality that ensures that the network and communication services operate as they are supposed to. This involves diagnosing, troubleshooting, and repairing things that do not work as planned, to keep the network in a state in which it can be continuously used and provide proper service.

- Finally, provisioning is concerned with the proper setting of configuration parameters on the network so that the network functions as expected. Depending on what gets provisioned, different types of provisioning are distinguished. Equipment provisioning is concerned with updating equipment configuration parameters and installing and turning up equipment. Service provisioning is concerned with configuring the network end-to-end to provide or disable a service for end customers at the proper service level.

FAB and eTOM: Oh, Wait, There's More

Yet another functional management reference framework has been established by the Telemanagement Forum (TMF), a consortium of companies in the telecommunications space that includes service providers, equipment vendors, and system integrators. This framework is known as the Telecoms Operations Map (TOM) and has the concept of a management lifecycle at its center; in a sense, it competes with the older OAM&P model, and because it is newer, it is not yet as established. Fundamentally, TOM distinguishes among three lifecycle stages, each with its own unique set of management requirements: Fulfillment—Assurance—Billing (FAB). TOM applies these lifecycle stages at different layers that are clearly distinguished:

- **Network and systems management**—Roughly corresponding to the element and network management layers in TMN
- **Service development and operations**—Roughly corresponding to the service management layer
- **Customer care**—Roughly corresponding to the business management layer

For example, applied to the management of a particular service:

- Fulfillment ensures that a service order that was received from a customer is carried out properly. This involves turning up any newly required equipment (for example, customer premises equipment such as a cable modem), performing required equipment configurations, and reserving required resources in the network, such as bandwidth or ports.
- Assurance includes all activities required to ensure that a service runs smoothly after it has been fulfilled. Services need to be monitored to ensure that quality-of-service guarantees are met. Any faults that occur in the network need to be diagnosed and repaired to keep their impact on the service to a minimum.
- Billing involves making sure that the services provided and resources consumed are accounted for properly and can be billed to the user. This is a very important step because, without the ability to bill properly, any service provider would quickly go out of business.

More recently, TOM has been extended into eTOM, the enhanced Telecom Operations Map. eTOM widens TOM's scope and aims to include all aspects of business management, incorporating aspects as diverse as supply chain management, human resources management, financial asset management, and so forth. However, there are no additional aspects with respect to the FAB categorization of management functions.

There is much more to eTOM than can be reasonably described here. For example, specific functions at the various layers and lifecycle stages are called out, along with the interfaces and interactions between those functions, all of which eTOM specifies in great detail. eTOM thus goes

beyond being a mere reference framework by defining a comprehensive set of standards that enables systems in an operational support environment to interact and interoperate with each other, and to collectively support a service provider's business processes.

How It All Relates and What It Means to You: Using Your Network Management ABCs

With so many functional reference models to choose from, which one is the best? Ultimately, it comes down to a matter of preference and to which model suits your network best. These reference models are, after all, virtual; the way in which you actually organize your management functions may look different altogether. You can cut things diagonally, horizontally, or vertically. The result in each case should be that you have partitioned the task of managing your network into smaller chunks that are much easier to tackle and digest than trying to conquer the entire task at once. The different models not only cut things differently, but they also apply a different number of cuts—yet each is perfectly valid and provides valuable orientation for how network management can be organized overall. In addition, they all provide a common terminology that can be used when discussing groups of management functions. However, someone well versed in the FCAPS model will have difficulty relating to someone who "grew up" in the OAM&P world, and vice versa. So how do the different models relate?

Table 5-1 provides a rough sketch of how FCAPS and OAM&P relate and effectively map to each other. An *X* in a cell indicates that the functions are closely related. An *(X)* indicates that the functions are still related, but to a lesser extent. An — indicates that the functions are only loosely related, if at all.

Table 5-1 *Relationship Between FCAPS and OAM&P*

	F	C	A	P	S
O	(X)	—	—	(X)	—
A	—	—	X	(X)	(X)
M	X	(X)	—	X	X
P	—	X	—	—	—

A word of caution: This mapping attempts to paint the big picture in broad strokes but is not entirely precise. For example, it should not be misinterpreted to mean that *configuration* is a synonym for *provisioning*. OAM&P provisioning is related to other FCAPS areas, such as security, in that it affects how security-related parameters will be provisioned. Likewise, FCAPS configuration plays a role in OAM&P administration because networks might need to be audited for their inventory, an aspect that, strictly speaking, is not part of provisioning.

With similar caveats, Table 5-2 provides a rough sketch of the relationship between FCAPS and FAB. Very roughly speaking, fulfillment encompasses configuration; assurance encompasses fault, performance, and security; and billing corresponds to accounting.

Table 5-2 *Relationship Between FCAPS and FAB*

	F	C	A	P	S
F	—	X	—	—	—
A	X	—	—	X	X
B	—	—	X	—	—

Chapter Summary

This chapter took a closer look at functional reference models. We took a tour of the most important management functions using the FCAPS model.

Fault management consists of functions to monitor the network to ensure that everything is working properly. Dealing with alarms and the large volume of events that are constantly being generated is one of the challenges that fault management addresses. However, it encompasses other functions as well, such as troubleshooting and diagnosis.

Configuration management is concerned with how the network is configured. This involves setting configuration parameters in such a way that the network can provide the services that it is supposed to. Configuration management also involves functions that enable users to audit a network and discover what's in it.

Accounting management deals with collecting and recording data about how the network is used and about the consumption of its services by end users. It is at the heart of being able to collect revenues and to be able to quantify the value that is derived from the network.

Performance management is all about collecting statistics from the network to assess performance and tune the network. The goal is to allow for proper allocation of resources in the network, such as removing bottlenecks, providing forecasts as input for network planning, and delivering the best possible quality of service with the given means.

Finally, security management is concerned with managing security-related aspects of the network. It is geared toward averting various kinds of security threats that a network and its management infrastructure are exposed to.

FCAPS is not the only way in which management functions can be categorized. Another model that is popular in particular with telecommunications service providers is Operations, Administration, Maintenance, and Provisioning (OAM&P), and more models exist. Each model

reflects a different way in which the various functions that are required to manage a network can be grouped and organized. However, regardless of which model you prefer, at the end of the day, the functions that need to be performed when managing a network remain the same.

Chapter Review

1. What does FCAPS stand for?
2. What does OAM&P stand for?
3. What is the difference between alarm filtering and alarm correlation?
4. The management functions discussed in this chapter pertain not only to the element management layer that deals with individual pieces of equipment, but really to any management layer. Give an example of a fault at the element management layer, an example of a fault at the network management layer, and an example of a fault at the service management layer.
5. Give an example of a configuration operation at the element management layer, a configuration operation at the network management layer, and a configuration operation at the service management layer.
6. Give an example of an event sent by a network device that supports an accounting management function. Give an example of an event that supports a security management function.
7. Provide a technical reason, not a marketing reason, for why a service provider might choose to provide flat-rate billing.
8. Performance and accounting management are similar, in that both are interested in collecting usage data from the network. Describe an important way in which the use of this data and the requirements for its collection differ.
9. "I have no need for security management functions because I am using a dedicated and secure management network." Please comment on this statement.
10. Provide a rough sketch of how OAM&P and FAB relate.