

CHAPTER

9

Network Management



Most Important Ideas and Concepts from Chapter 9

Although Chapter 9 is the shortest chapter in our textbook, network management is an important topic nonetheless with important ideas and concepts that one should take away from this chapter. We discuss only five important ideas below (rather than our usual ten), not only to reflect the chapter's short length, but also to assure you that the concepts below are indeed important. Rather than forcing ourselves to choose an arbitrary number of topics (ten), we list and discuss the network management topics that we believe are truly important and most deserving to be singled out for emphasis.

- ◆ **Network management framework: managed device (managed objects, management information base, management agent), managing entity, and network management protocol.** Figure 9.2 on page 735 of the textbook illustrates the conceptual framework for network management. First and foremost, there are *managed devices*—the network equipment, both hardware and software, that make up the network. Each managed device may have many components, or managed objects. A managed object may be a physical part of the device (for example, an interface card) or a piece of software (for example, a protocol such as RIP or TCP). Information about a managed object is contained in a *management information base (MIB)* in the managed device. A MIB might contain the number of checksum errors encountered in frames on an Ethernet interface, or the number of UDP segments received by a host. The *management agent* is a software process associated with a device that communicates with the network's *managing entity*. The managing entity is also a software process. It receives information from the managed devices and, together with humans-in-the-loop, issues commands that control the managed devices. The protocol used for communication between the managing entity and each managed device is the *network management protocol*. Often, the term “network management protocol” is considered synonymous with “network management.” However, now that we've studied Chapter 9, we know that the network management protocol is only a part (albeit a very important part) of the larger network management framework.
- ◆ **SNMP protocol.** The SNMP protocol is the Internet Standard protocol (RFC 3416) for communication between managed devices and the managing entity. SNMP can be used in two different ways: In *request-response mode*, the managing entity sends an SNMP request message to the agent in the managed device. This SNMP request is typically used to query or set (for example, initialize) a MIB object value in the managed device. The agent in the managed device then replies to the SNMP request message with an SNMP response message. In *trap-mode*, a managed device sends an SNMP trap message to the managing entity on its own initiative, that is, without having first received a request message from the managing entity. A trap message typically informs the managing entity that an exceptional condi-

tion has occurred (for example, an interface has gone down). In this sense, a trap message is much like a hardware or software interrupt in a computer—it informs the controlling entity (the operating system in the case of the computer or the managing entity in the case of a managed network) that a noteworthy event has occurred. On receiving the interrupt/SNMP trap, the computer/managing entity can then take appropriate actions in response to the interrupt/trap.

- ◆ **The need for presentation services.** If all computers represented and stored information exactly the same way, there would be no need for a presentation service. But alas, in spite of standards, different computers have different ways of representing and storing information internally. Perhaps the most well-known example of such a difference is the fact that some computers store the bytes that make up an integer value in most-significant-byte-first order; other computers store the bytes that make up an integer in exactly the opposite order—in least-significant-byte-first order!

But what does byte ordering in computers have to do with communication? Recall that when a communication protocol sends data from one computer to another, it is simply sending a stream of bytes. Thus, if a sender transmits the bytes representing an integer that is stored most-significant-byte-first at the sender to a receiver that interprets/stores an integer least-significant-byte-first, then the integer value sent by the sender will be interpreted as a very *different* integer value by the receiver. The role of presentation services is to solve such problems, ensuring that the *meaning* of the information communicated (for example, the numerical value of an integer) rather than the underlying representation of the information (for example, most-significant-byte-first, or least-significant-byte-first order) is preserved. Figures 9.7 and 9.8 on page 755 of the textbook provide a more whimsical illustration of the presentation problem, using the analogy of an aging hippie, a grandmother, and a teenager, each of whom has a very different way of internally representing the idea that something is pleasing.

- ◆ **Type, Length, Value (TLV) encoding.** Suppose that two computers want to exchange a piece of information, while preserving the meaning of this information (for example, in our example above, the value of an integer rather than the verbatim internal representation of this integer). A natural way for them to do this is to agree on a predefined data format. The so-called Type, Length, Value (TLV) encoding is one way to achieve this. The TLV encoding of a piece of data indicates the *type* (T) of the data (which must be one of a set of predefined data types); the number of bytes needed to express the value of the data (that is, the *length* (L) of the data), and the *value* (V) of the data itself (in some predetermined format, agreed to by all). TLV encoding is sometimes referred to as *self-identifying encoding*, since every TLV-encoded piece of data explicitly carries its data type (that is, is self-identifying) as part of its encoding.

- ◆ **Abstract Syntax Notation One (ASN.1).** ASN.1 is an ISO standard for data representation. ASN.1 has a set of predefined data types (for example, Boolean, integer, real, character string, object name) and a set of rules that define the TLV encoding for these data types. Data to be transmitted in ASN.1 format is encoded using ASN.1's TLV encoding rules (known as Basic Encoding Rules, in ASN.1 parlance) before being sent from sender to receiver. The receiver, knowing that the data it is receiving is ASN.1-encoded (because the protocol *requires* all data to be encoded using ASN.1), can use the ASN.1 syntax to decode the data, and then locally store the data according to its own local data format/storage conventions.



Review Questions

This section provides additional study questions. Answers to each question are provided in the next section.

1. **Request-Response versus trap mode.** Suppose that a network operator wants to be notified immediately if a link interface goes down. Should this notification be performed using SNMP in request-response mode, or in trap-mode? Why?
2. **TLV or TVL coding?** Why do you think the length precedes the value in a TLV encoding (rather than the length following the value)?
3. **ASN.1.** What is the ASN.1 object identifier for the TCP protocol? (See Figure 9.3 on page 744 of the textbook.)
4. **ASN.1 (more).** What is the ASN.1 object identifier for the MIB variable that counts “the total number of [TCP] segments received, including those received in error. This count includes segments received on currently established connections.” [RFC 1213]? (To answer this question, refer to RFC 1213, or better yet, use the online interactive ASN.1 object identifier tree at <http://www.alvestrand.no/objectid>.)
5. **SNMP.** Suppose that the network managing entity wants to send an SNMP message to a host to query that host for the “the total number of [TCP] segments received, including those received in error. This count includes segments received on currently established connections.” [RFC 1213]. Consider Figure 9.4 on page 748 of the textbook. What are the values in the fields of the SNMP message sent to the host? (In order to answer this question, you will need to do some background investigation. Consult RFC 3416, page 6 to determine the type value for the SNMP GET request. To get the size and range of possible request ID values, and the values for the Error Status and Error Index field, see RFC 3416, page 7. The name field of the SNMP MIB variable being queried contains the ASN.1 object identifier tree value of that SNMP MIB variable; therefore, see the answer to Review Question 4.)
6. **ASN.1 BER.** Consider Figure 9.9 on page 757 of the textbook. What is the ASN.1 BER encoding of {weight, 129} {lastname, “Chopin”}?
7. **ASN.1 BER (more).** Consider Figure 9.9 again. What is the ASN.1 BER encoding of {weight, 262} {lastname, “Beethoven”}?



Answers to Review Questions

1. Trap mode. In request-response mode, the operator would have to probe the device continuously, checking that the interface is up. If the operator sends a probe every T time units, then on average, the operator would find out that a link has gone down $T/2$ time units after the event has occurred. Also, if the link is normally up, there is a high overhead to this polling—most SNMP request and response messages would be sent simply to determine that the link is (still) up. In trap mode, a single SNMP message will be generated as soon as the link goes down.
2. The length precedes the value so that after determining the length, the receiver knows exactly how many of the subsequent bytes constitute the value. Recall that some data types (for example, character strings) can be of arbitrary length; therefore, the receiver needs to know where in the ensuing stream of bytes the value ends.
3. 1.3.6.1.2.1.6
4. 1.3.6.1.2.1.6.10. See <http://www.alvestrand.no/objectid/1.3.6.1.2.1.6.html>
- 5.

```
PDU type: 0      /* the GET request has type 0 )
Request Id: <an integer value put here by the managing entity>
              /* per RFC 3416, range is (-214783648..214783647) */
Error Status: 0 /* this field not used for a GET */
Error Index: 0 /* this field not used for a GET */
Object identifier 1: 1.3.6.1.2.1.6.10
Value: NULL     /* NULL is an ASN.1-defined data type */
```

6. 4 6 C h o p i n 2 1 '10000001'.
The 4 indicates that the first data item is an octet string (see Table 9.5 on page 756 of the textbook). The 6 indicates that there are six letters following. This is the ASN.1 BER encoding of the character string “Chopin”. The 2 indicates that the next item is an integer. The 1 indicates that the integer value is one byte long. The binary value ‘1000001’ is the binary representation of the number 129.
7. 4 9 B e e t h o v e n 2 2 '00000001' '00000110'.
The last two bytes are the binary representation of the number 262.