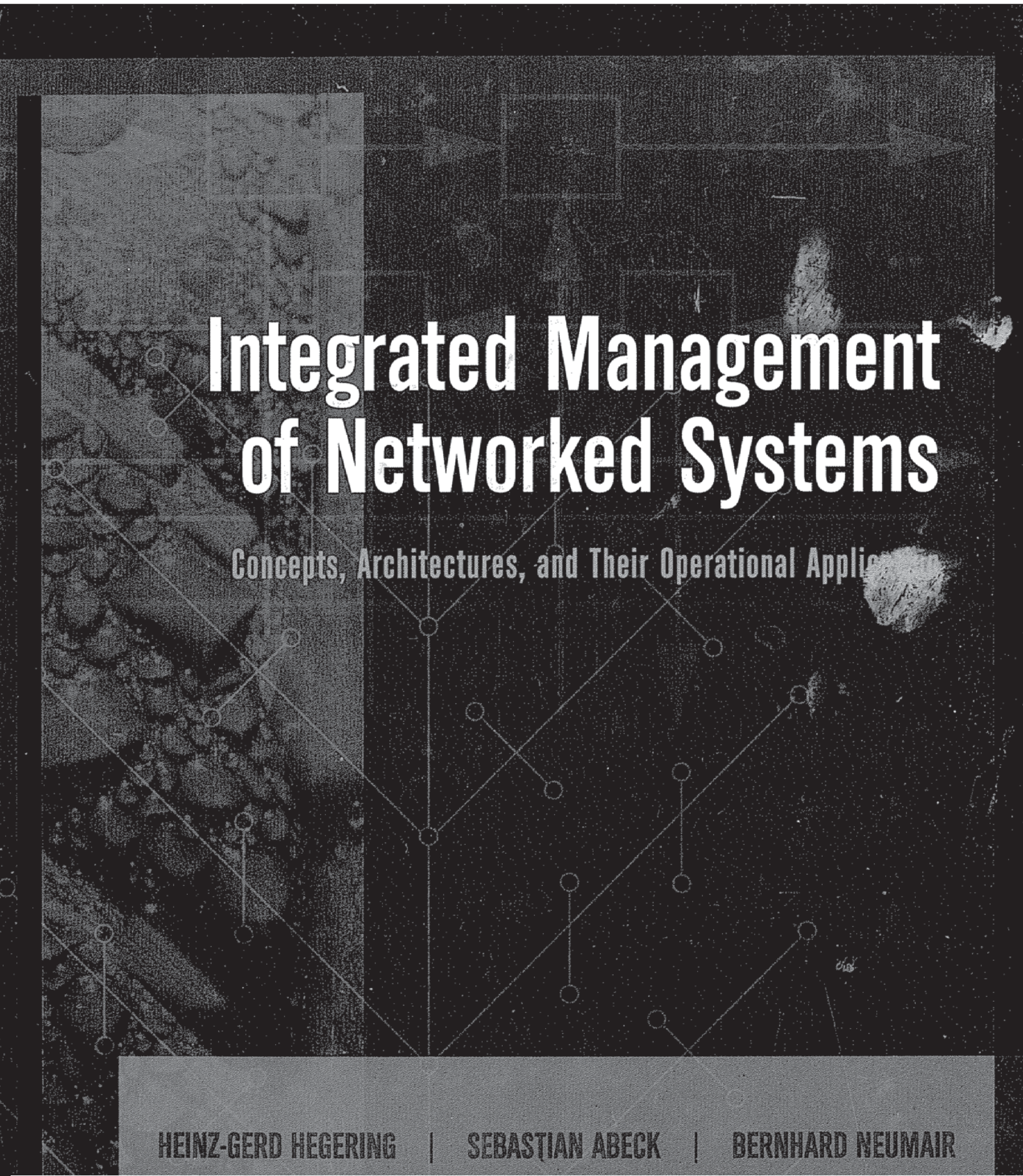

Integrated Management of Networked Systems

Concepts, Architectures, and Their Operational Application

HEINZ-GERD HEGERING | SEBASTIAN ABECK | BERNHARD NEUMAIR

The Morgan Kaufmann Series in Networking

Series Editor, Dave Clark

- *Integrated Management of Networked Systems: Concepts, Architectures, and Their Operational Application*
  Heinz-Gerd Hegering, Sebastian Abeck, and Bernhard Neumair
- *Virtual Private Networks*
  Dennis Fowler
- *Networked Applications: A Guide to the New Computing Infrastructure*
  David G. Messerschmitt
- *Modern Cable Television Technology: Video, Voice, and Data Communications*
  Walter Ciciora, James Farmer, and David Large
- *Switching in IP Networks: IP Switching, Tag Switching, and Related Technologies*
  Bruce S. Davie, Paul Doolan, and Yakov Rekhter
- *Wide Area Network Design: Concepts and Tools for Optimization*
  Robert S. Cahn
- *Optical Networks: A Practical Perspective*
  Rajiv Ramaswami and Kumar Sivarajan
- *Practical Computer Network Analysis and Design*
  James D. McCabe
- *Frame Relay Applications: Business and Technology Case Studies*
  James P. Cavanagh
- *High-Performance Communication Networks*
  Jean Walrand and Pravin Varaiya
- *Computer Networks: A Systems Approach*
  Larry L. Peterson and Bruce S. Davie

Integrated Management of Networked Systems

Concepts, Architectures, and Their Operational Application

Heinz-Gerd Hegering
University of Munich
Sebastian Abeck
University of Karlsruhe
Bernhard Neumair
DeTeSystem GmbH

Morgan Kaufmann Publishers
San Francisco, California

Senior Editor: Jennifer Mann
Editorial Assistant: Karyn Johnson
Director of Production & Manufacturing: Yonie Overton
Production Editor: Sarah Burgundy
Cover Design: Wall-to-Wall Studios, Pittsburgh, PA
Cover Image: © A. Child/The Stock Solution
Text Design: Rebecca Evans and Associates
Illustration: Dartmouth Publishing, Inc.
Composition: Ed Sznyter, Babel Press
Copyeditor: Robert Fiske
Proofreader: Jennifer McClain
Indexer: Steve Rath
Printer: Courier Corporation

Morgan Kaufmann Publishers, Inc.
Editorial and Sales Office
340 Pine Street, Sixth Floor
San Francisco, CA 94104-3205
USA

Telephone	415/392-2665
Facsimile	415/982-2665
Email	*mkp@mkp.com*
WWW	*http://www.mkp.com*
Order toll free	800/745-7323

Title of German original: Integriertes Management vernetzter Systeme

ISBN: 3-932588-16-9

Printed in the United States of America
03 02 01 00 99 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Hegering, Heinz-Gerd, 1943–
[Integriertes Management vernetzter Systeme. English]
Integrated management of networked systems: concepts, architectures, and their operational application / Heinz-Gerd Hegering, Sebastian Abeck, Bernhard Neumair
p. cm. — (Morgan Kaufmann series in networking)
Includes bibliographical references and index.
ISBN 1-55860-571-1
1. Computer networks—Management. 2. Computer network architectures. I. Abeck, Sebastian. II. Neumair, Bernhard. III. Title. IV. Series.
TK5105.5.H43313 1999
004.6—dc21 99-35160
CIP

Contents

6

Internet Management

Internet management architecture is defined by a large number of standards that also refer to several different versions, thereby making it difficult to provide a structured description. Our presentation of the material is structured as follows. Section 6.1 provides a general overview. Section 6.2 presents the Internet information model (SMI), broken down into version 1 and version 2. Section 6.3 then presents the communication model (SNMP), also broken down into version 1 and version 2. Section 6.4 deals with RMON, and section 6.5 outlines some of the newer developments that have not yet been completely standardized.

6.1 Overview

The Internet management architecture—frequently also called SNMP management because of its management protocol—clearly forms the basis for the majority of multivendor management solutions in the data communications environment. This dominance is due, on the one hand, to the widespread use of IP-based protocols as a result of the Internet but also to the fact that, compared to OSI and OMG, Internet management concepts are simpler and lend themselves to implementation by relatively small components and inclusion within inexpensive products at a reasonable cost. Another reason is that the standardization process in the Internet area is less complicated and more open to informal participation. Even ordinary users are able to familiarize themselves quickly with the development of a standard using the request-for-comments (RFC) procedure. This procedure is controlled by the Internet Architecture Board (IAB) through the Internet Research Task Force (IRTF) and the Internet Engineering Task Force (IETF). Workgroups on different subject areas produce recommen-

dations for standardization that, after appropriate review processes, become either a proposed or a full standard category draft. The documents are assigned a corresponding RFC number. IETF is largely vendor driven, whereas ISO and ITU are largely carrier driven.

Historical development of Internet management

Management protocols first came on the scene in the late 1980s with the development of the host monitoring protocol (HMP), the high-load entity management system (HEMS), the simple gateway monitoring protocol (SGMP), and CMOT (i.e., CMIP over TCP/IP). The one that enjoyed the greatest success, however, was the simple network management protocol (SNMP), a further development of SGMP.

This basic SNMP, now also called SNMPv1 as a result of further development, is used widely today. Because of various weaknesses in the original design (relating to the transmission of high-volume management data and to security requirements such as authentication and encryption), the first follow-up version SNMPv2 was introduced. Although appropriate products were available from the outset, version 2 had difficulty gaining full acceptance due in part to its controversial security concept. It may also be mentioned that SNMPv2 suffered from the success of SNMPv1. Vendors saw little market motivation to incur cost of migration because the market was largely still happy with SNMPv1. In the meantime, yet another version, SNMPv3, is at an advanced stage in the standardization process.

Chapter overview

The Internet management architecture does not incorporate a very distinctive organizational or functional model. Compared to OSI management, even the information model (section 6.2) is simpler and not truly object oriented. The development goal was to produce uncomplicated concepts and therefore products. Flexibility and functionality are therefore not incorporated into the information model but are achieved through the management applications. Despite or because of their simplicity, Internet MIBs and SNMP offer considerable advantages. Although the lack of special inheritance techniques restricts the reuse of specifications and makes it less convenient to carry out systematic object manipulation in the MIB (e.g., efficient MIB browsing or scoping), it makes it easier for developers and users to understand and implement the concept, often assisted by freeware and publicly available tools. This explains why the Internet approach has caught on quickly, especially in the LAN area.

In section 6.2.1, we describe the information model in terms of the way it was defined for version 1 of SNMP. Section 6.2.2 looks at the extensions to the model that were introduced with SNMP version 2.

What also makes the Internet communication model (section 6.3) less complicated is the fact that the management protocol task pro-

vides access to the MIB according to the information model. SNMP can run over any transport network but is usually deployed over the connectionless UDP protocol. Version 1 of the management protocol is introduced in section 6.3.1.

In section 6.3.2, we describe the further development of the Internet management protocol as it exists with SNMPv2, which was published in 1993. Several SNMPv2 dialects have been developed further since then, essentially differing in the security concept used. We cover the variants SNMPv2p and SNMPv2c.

Section 6.4 is dedicated to remote monitoring MIB I and II (RMON-MIB). This is an approach in which the managed component already carries out and controls the standardized collection and preprocessing of management information, thereby reducing the workload of the manager and the network. This illustrates a function model defined for use with SNMP.

We conclude in section 6.5 by briefly describing the latest state of development, namely, SNMPv3 and DISMAN. An attempt is made in SNMPv3 to achieve a convergence of the different SNMPv2 variants and to design a modular, expandable architecture for SNMP entities. DISMAN relates to an Internet approach for management by delegation.

An enormous number of documents (RFCs) are available for Internet management, but only a small number of them have the status of a full standard. This is partly because the IETF will obsolete or downgrade full standards if replaced by later standards or found to be lacking in deployment. The main standards for SNMPv1 are RFC 1155 (describing the information model), RFC 1213 (describing the original Internet MIB), and RFC 1157 (describing the management protocol SNMP). The text makes reference to other RFCs. All RFCs and current designs for further development are accessible to the public through electronic means (*http://www.ietf.org*).

It should be mentioned that there is the need for coexistence of Internet management and OSI management. This makes necessary some mediation architecture that performs the translation of, for example, the different information models. This topic will be discussed in Chapter 10.

6.2 Internet Information Model (SMI and MIB)

The Internet information model (structure of management information, SMI) is available in two versions. SNMPv1-SMI is the original

Internet management structure

version and continues to be the most widely used one. In the course of the development of a successor version of the SNMP management protocol, the information model has also been extended to SNMPv2-SMI. The basic concepts of version 1 have been retained, however.

6.2.1 SNMPv1-SMI

Internet management is based on the client–server principle in which the client is usually designated as the management station (in short, manager) and the server as the management agent (in short, agent). In this asymmetric cooperation model (Figure 6.1), the manager is the carrier system for all management applications; it provides an interface to human operators and maintains a (logical) database for all relevant data of the network it monitors. Using the SNMP management protocol, a manager communicates with the monitored resources (managed nodes), more specifically, with the "management representative" of the resource, the agent. Through the agent, the manager has read or write access to the agent MIB (management information base). This *agent MIB* is a collection of variables that are characteristic of the behavior of the managed node. With Internet management, these MIB variables are also called *managed objects*, although they do not incorporate any kind of object orientation in the sense of encapsulation, inheritance, or the like. The variables represented by Internet MOs are similar to the MOC attributes of OSI.

Internet MIBs and agent MIB

The *Internet MIBs* define the structure of the agent MIB. These specify which managed objects are even allowed to appear in an agent MIB, how they are structured, which meaning they incorporate, and how they can be identified. Internet MOs are related hierarchically and named according to the same registration tree used by ISO and ITU. The agent MIB contains the MOs provided by the managed nodes. Each variable is a leaf on the registration tree with a dependency on a typically concrete (can be logical, too) resource. The MO variables of the agent MIB can be queried about current values; if need be, a value can be changed, which then has an effect on the represented resource. An agent MIB therefore represents the instantiation of (sometimes more than one) Internet MIBs (Figure 6.2): The latter defines the "object types" that are allowed in principle; the former contains the "object entities" that are supported by the respective concrete agent.

Internet registration tree

An important objective associated with the specification of multi-vendor management information is the development of a unique form of identification and description of information worldwide. To do jus-

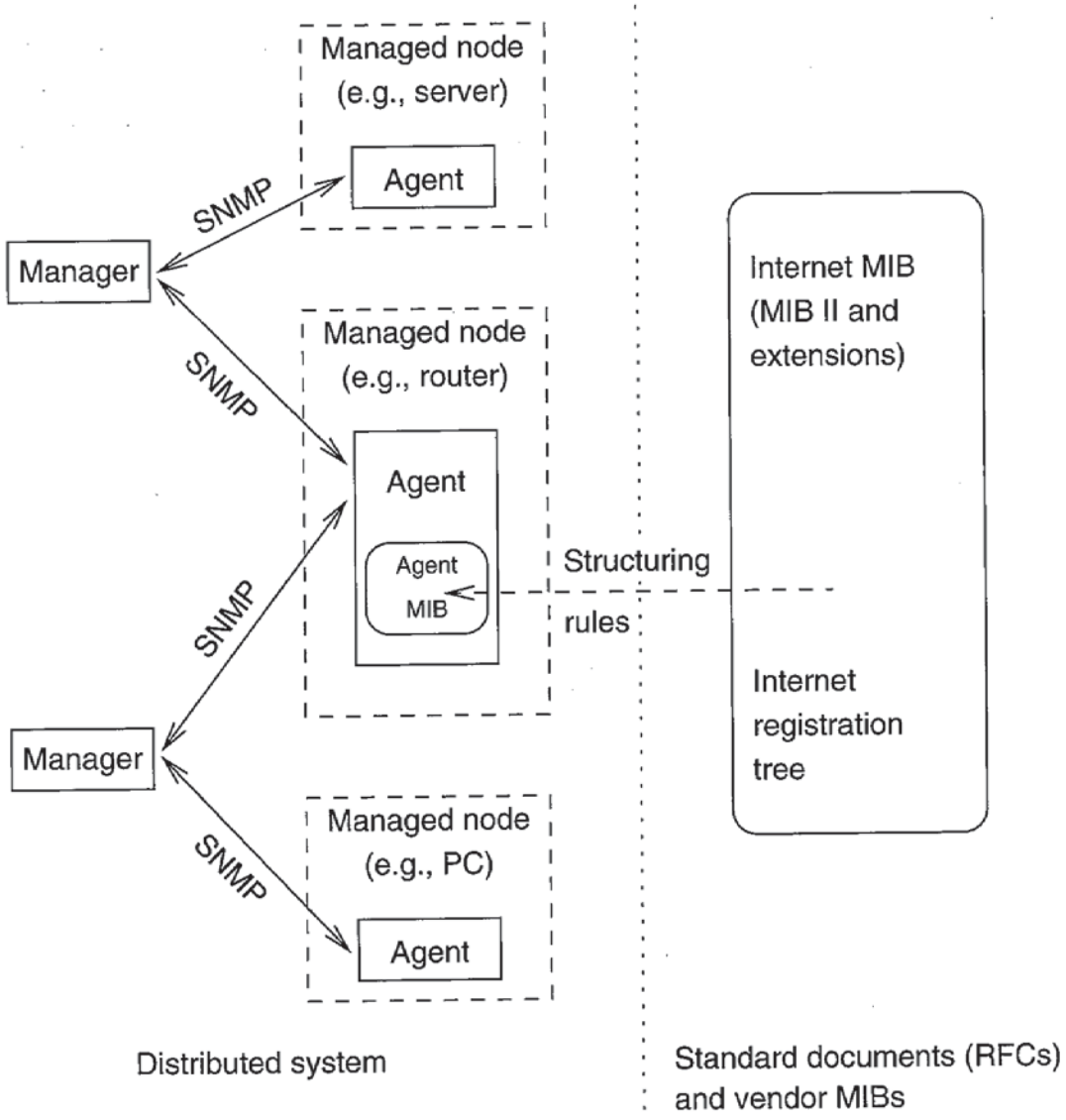

Figure 6.1
Internet architecture model and MIBs

tice to this requirement, the ISO and the ITU-T have introduced a global *object identifier tree*, which permits any objects, not just those in the management domain, to be assigned to a unique worldwide identifier. The assignment of the object identifiers is extremely simple: The name space is organized like a tree in which each tree node is assigned a name and a number, with the numbers beginning with 1 enumerating all the nodes belonging to a parent node. This allows delegation of naming authority (e.g., the IETF is responsible for numbering all nodes registered under "1.3.6.1"). Figure 6.3 shows the organization of the Internet name tree (i.e., the part of the registration tree below the node with object identifier "1.3.6.1" and the name "iso.org.dod.internet").

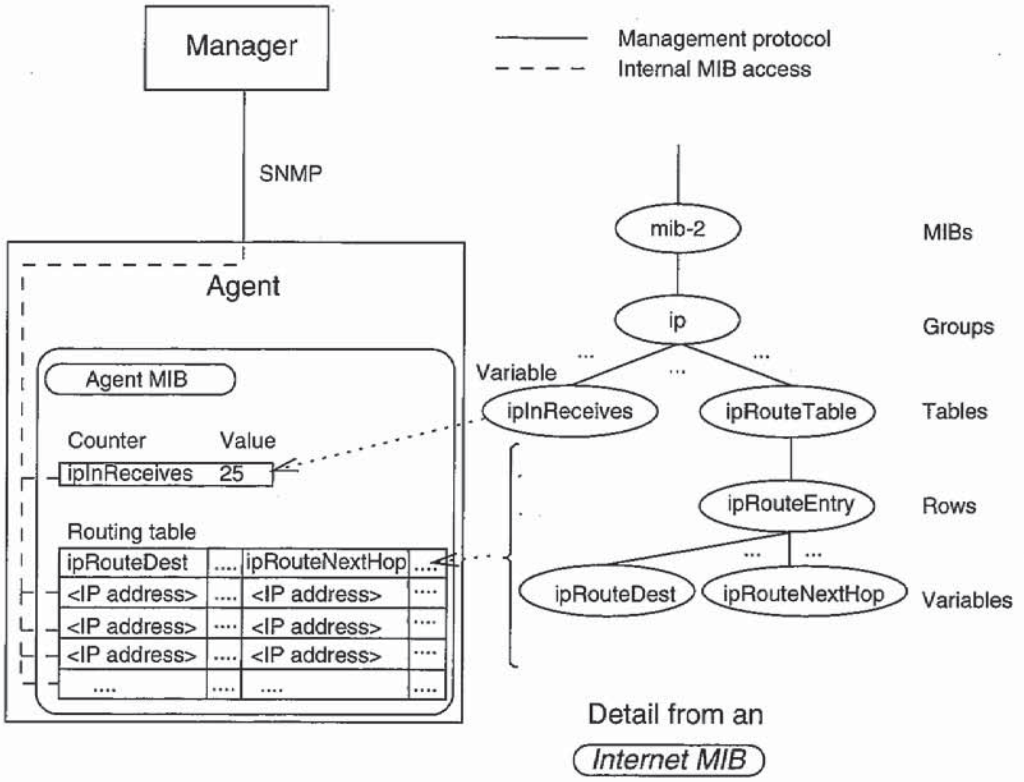

Figure 6.2
Example of an agent and Internet MIB

Management information is scattered over several subtrees of the registration tree

- Below the "mgmt" node with object identifier "1.3.6.1.2," currently only one node, "mib-2," is defined, below which is the standard MIB (currently MIB-II defined by RFC 1213). Each management agent is required to have the capability of interpreting MIB II. We will look at its content later.
- The *experimental MIBs* are incorporated below the "experimental" subtree. This generally relates to technology-specific management information for FDDI, frame relay, the DS3 interface, ISO CLNS objects, X.25, ATM, UNI, and SONET—a total of more than 70 MIBs [MIL97]. The management information contained in these MIBs is a candidate for extending future standard MIBs. In practice, many MIBs become mature without moving to the "management" branch.
- The "enterprise" subtree is reserved for enterprises that want to define manufacturer-specific management information in accordance with the Internet information model. More than 3000 enterprise-specific subtrees already exist [MIL97]. The explosion of proprietary *manufacturer MIBs* is having an effect on the interoperability of management systems because MOs are addressed through their position in the registration tree and this has to be reflected in the management applications. Moreover, many MIBs contain hun-

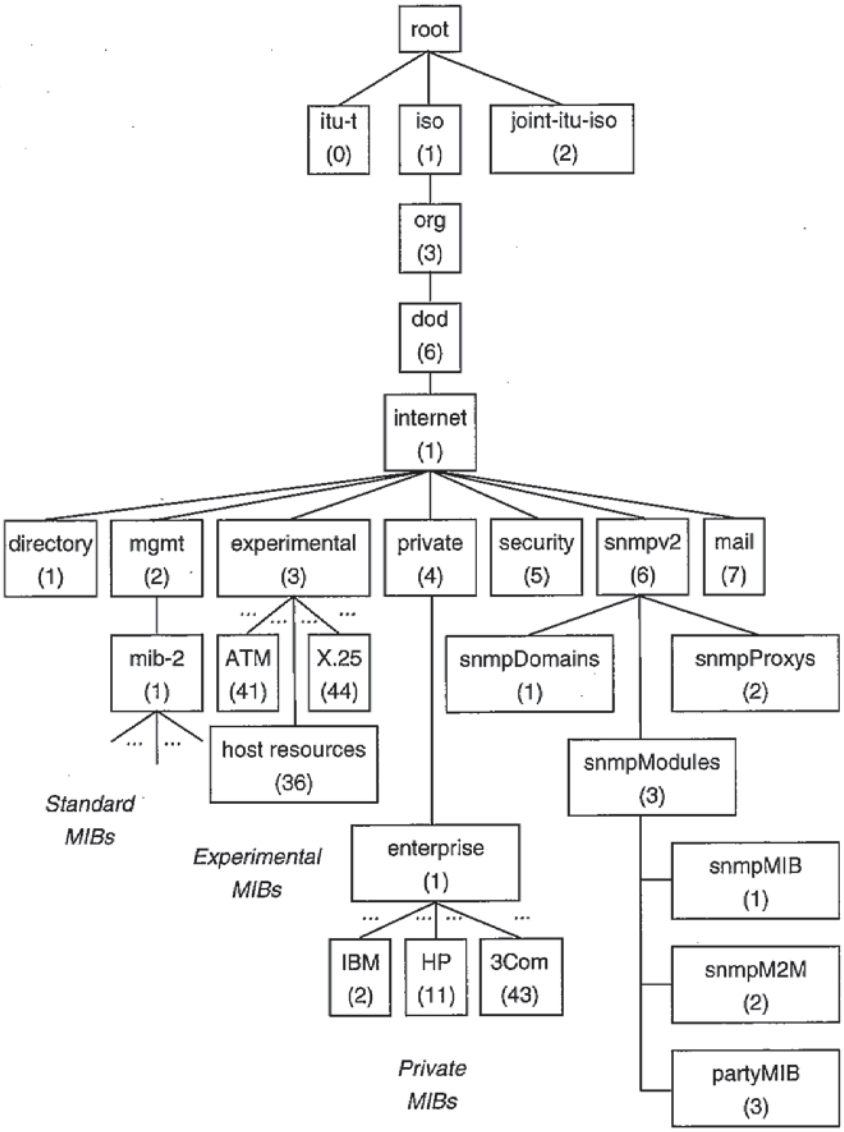

Figure 6.3
Organization of Internet registration tree

dreds of MIB variables, and it is often difficult to understand the true relevance of these variables to management. The type and number of MIBs implemented are gradually becoming product-differentiating characteristics in the immense set of management product offerings.

Initially, only the first four subtrees appeared below the "internet" node; the others were added later as a result of further developments. The internal nodes of the registration tree are pure structuring nodes and do not contain supplementary information. They are used exclusively for registration and object identification; the actual management information is located only in the leaves. In a certain way, the struc-

turing nodes fill the role of class definition, the attributes of which are the leaves. However, their position in the tree induces neither a containment (at least not in every case) nor an inheritance hierarchy.

Description of object types, generic object type macro

The IETF defined an ASN.1 OBJECT-TYPE macro for the nodes that contain the "real" management information; this macro allows the specification of the following information:

- Name and object identifier of the node
- Syntax of the management information referenced by or available through accessing the node in the form of an ASN.1 data type
- Informal description of the semantics of this management information
- Specification status information that determines whether the management information is mandatory, optional, or obsolete because it has been made superfluous as a result of a new specification for the MIB

Nodes that contain this information are designated as *object types.* The instantiation of object types, the *object instances,* produces the actual *managed objects* (MOs) in the agent MIB. All the managed objects of the different Internet subtrees per Figure 6.3 are specified in accordance with the preceding scheme.

Let us use an IP packet counter as an example. The concrete syntax in the form it was used for the object `ipInReceives` in the original standard MIB (RFC 1213) reads:

```
ipInReceives OBJECT TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
>>The total number of input datagrams received from
interfaces, including those received in error.<<
::= { ip 3 }
```

The specification clarifies that the object is a "counter" that can be accessed only through reading and must be supplied by an SNMP agent. An informal description of what is to be counted by the object is also supplied. The description concludes with details on how the object can be identified. It also identifies the branch the object is located on relative to the parent node; in the example, it is branch 3 below the node "ip."

Allowable ASN.1 data types, generic object types

Counter is one of the following six ASN.1 data types predefined in the Internet information model (RFC 1155, structure of management information, SMI) to describe the syntax of Internet objects:

- *Network address* enables the selection of a protocol family. Initially, this selection was limited to Internet protocols.
- *IpAddress* is used in the representation of the 32-bit-long Internet address.
- *Time ticks* allow the specification of periods of time with each time tick comprising 1/100 second, relative to the last boot time at the agent.
- *Gauge* defines a counting object that can assume a value between 0 and $2^{32} - 1$, can be increased and decreased, and does not loop.
- *Counter* defines a counting object that can assume a value between 0 and $2^{32} - 1$ and can only be increased. The counter is cyclical (modulo numbering).
- A variable of the type *opaque* contains the value of any ASN.1 type desired (i.e., this allows late binding of types).

Tables as object types

In addition to these specifically defined ASN.1 types in the Internet information model, the simple types `INTEGER`, `OCTET STRING`, `OBJECT IDENTIFIER`, and `NULL` are also supported. `SEQUENCE` and `SEQUENCE OF` are permitted as ASN.1 types for combined objects also referred to as "constructor types"; these two data types are used to construct Internet tables. Tables are the only combined objects that occur in an Internet MIB. They are constructed on the basis of the following principle: The object type that describes an entire table consists of a "`SEQUENCE OF` ⟨table row⟩" with each row of a table in turn consisting of a "`SEQUENCE` ⟨table column⟩." The two different ASN.1 types `SEQUENCE OF` and `SEQUENCE` take into account the fact that the entries in a table can be added or removed dynamically as table rows, whereas the columns that describe the MIB variables stored in the table must be established when the table is defined. (Note: This restriction no longer applies to the expanded information model SNMPv2-SMI [RFC 1902], sometimes also referred to as SMIv2, and therefore table columns can also be expanded through subsequent refinement, but still not added to at runtime.)

Example of a table structure

We demonstrate the Internet approach to table descriptions on the basis of the specification for the routing table (i.e., the description of the managed object `ipRouteTable`). We have chosen this example for didactic reasons because the description is still based on the

conventions of SNMPv1-SMI. In a now obsolete version of MIB II (RFC 1213), different object types were used to describe the table. Because space is limited, we are listing only 2 of the 13 total table columns:

Structure of the whole table

```
ipRouteTable OBJECT-TYPE
SYNTAX SEQUENCE OF IpRouteEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
>>This entity's IP Routing table.<<
::= { ip 21 }

IpRouteEntry OBJECT-TYPE
SYNTAX IpRouteEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
>>A route to a particular destination<<
INDEX { ipRouteDest }
::= { ipRouteTable 1 }
```

Structure of a row

```
ipRouteEntry ::=
SEQUENCE {
ipRouteDest IpAddress,
...
ipRouteNextHop IpAddress,
...
}
```

A column element

```
ipRouteDest OBJECT-TYPE
SYNTAX IpAddress
ACCESS read-write
STATUS mandatory
DESCRIPTION
>>The destination IP address of this route. An entry with ...<<
::= { ipRouteEntry 1 }
```

Another column element

```
ipRouteNextHop OBJECT-TYPE
SYNTAX IpAddress
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The IP address of the next hop of this route...."
::= { ipRouteEntry 7 }
```


According to the description, access is not possible to the table object ipRouteTable or the table row object ipRouteEntry itself. In other words, a manager cannot retrieve the entire table or table row—it must retrieve individual columnar variables. In terms of how they are described, the object types ipRouteDest and ipRouteNextHop, which are columns of a table row, seem no different from the so-called simple MIB variables, such as the counter ipInReceives. The differences are first evident when the respective instantiations are referenced. This also applies to the INDEX entry that appears in the table row object ipRouteEntry. At least one appropriate OBJECT-TYPE, which has to appear as one of the columns in the table, is specified as an index for each table. This is called an *index object type* (ipRouteDest in the example). If access to a specific table row is desired, then the current value of the index object type in this line is appended to the identifier for the object itself to form a "name." The operation "getnext" mentioned later is utilized to "walk through" table rows by following this name structuring.

It should be pointed out that the example given was used only to provide a general description of a table. As we have already mentioned, the example selected ipRouteTable is now obsolete as a managed object. ipRouteTable from RFC 1213 was replaced by *ipForwardTable* in RFC 1354, which in turn has now been replaced by RFC 2096 to allow different options for specification. The Internet information model in the extended SNMPv2-SMI version (which we cover in section 6.2.2) is already being used for description purposes. The new routing table structure is provided by the combined object type ipCidrRouteTable (an excerpt will be given in section 10.3.1), which is registered in the IP group under ipForward (4) in MIB II. This table, which even has an index consisting of four object types, can be used to support policy-controlled interdomain routing.

MIB II is the currently valid basic specification of management information. Eleven other structural nodes called *groups of MIB II* (Figure 6.4) are attached to structuring node 1.3.6.1.2.1 assigned to mib-2 itself. The rule is that if one device (agent) supports an MIB group, then all objects (leaves of the tree) must support this group.

Groups in MIB II

- *Group system*: The objects in this group supply general information about the managed node (sysDescr, sysObjectID, sysLocation, sysUpTime), information about contact partners and system names (sysContact, sysName), and information coded in an integer number identifying which services of which protocol layer are supported by the managed node (sysServices).

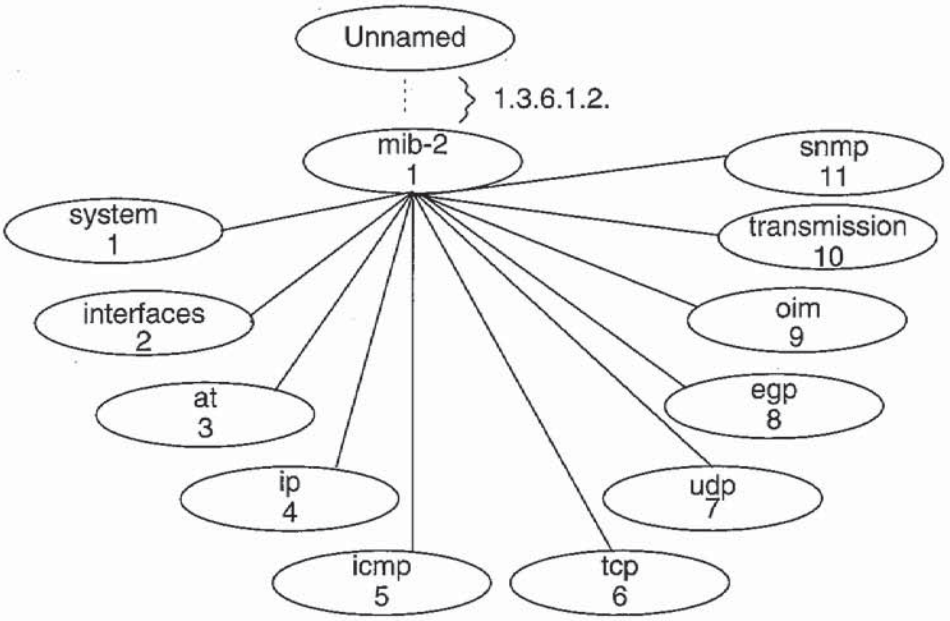

Figure 6.4
MIB II groups

- *Interfaces group*: This group consists of the number of network interfaces (`ifNumber`) provided by the managed node plus a table (`ifTable`) containing the following information for each interface: type (`ifType`) and description of the interface (`ifDescr`), information on the configuration of the interface (`ifSpeed`, `ifMtu`, `ifPhysAddress`), status information (`ifAdminStatus`, `ifOperStatus`), and finally statistics counters (`ifInOctets`, `ifInUnknownProtos`, `ifOutUcastPkts`, `ifOutErrors`, etc.). The latest extension to this information can be found in RFC 2233.
- The *address translation* (at) *group* was a part of MIB I but was assigned the status "deprecated" in MIB II, meaning that it is being continued only for compatibility with MIB I. It was a table for mapping IP addresses to hardware addresses. In the future, each protocol group is being defined with its own address mapping tables.
- *Protocol groups ip, icmp, tcp, udp, egp, snmp*: Statistics counters appear in each group and are used to count different types of incoming and outgoing protocol data units as well as different kinds of error situations. Other tables that contain specific information about routing, connections, and neighboring nodes for each protocol are also available. We have already referred to the current valid structure of the routing table (RFC 2096).

- *Transmission group*: This group contains management information for a number of different transmission protocols and network interfaces such as X.25, Ethernet, Token Ring, FDDI, LAPB, DS1, E1, PPP, DS3, SDMS, frame relay, RS232, ATM, and SONET. The DS1 interface alone contains almost 100 MIB variables known as DS1-MOs. Each transmission protocol is represented by an MIB beneath this group.
- *OIM group*: OIM, which stands for OSI Internet Management MIB, is already considered a "historic" group and contains managed objects for CMOT, thus CMIP over TCP.

Details about object types and their precise specification will not be included in the context of this discussion. In-depth knowledge of all protocols and network technologies is required in order to develop a full understanding of the information defined in these groups.

But to reiterate, the original Internet SMI information model is not object oriented, which means that it does not contain object classes or inheritance. The large number of MIB catalogs that exist in the Internet area would suggest that the simple registration tree (which does not provide a complete or flexible representation of containment) does not allow an adequate structuring of management information. The lack of a mechanism for refining and reusing object definitions particularly results in logically related management information being maintained in totally separate subtrees within the registration tree. This is obvious, for example, when you need all the available information for a particular interface card; the information is dispersed over all sorts of different tables of MIB II, of the vendor MIB, and of the RMON-MIB. In this sense, too, the many private MIBs are largely redundant. This is where the hidden heterogeneity surfaces again: The management applications must be able to recognize these many MIBs since MOs of different subtrees have to be addressed separately. Related MOs are often specified in different MIBs or registered in different branches of the registration tree; besides, object identifiers (OIDs) in the tree are used as names and the SNMP protocol is structured only to traverse OIDs, but not relationships.

MIBs for systems and application management

The MIBs mentioned so far have all been network oriented (i.e., they have all concerned the network management of the Internet, its protocols, components, underlying networks, and access networks). In recent years, Internet management has also recognized the importance of not limiting integrated management to specifics of the networks but also applying the concept to aspects of systems management and application management. Definitions of management information based on the principles of the SMI information model have therefore

also been available in these areas. Examples of MIBs that can be used beyond pure Internet network management include:

- Host resources MIB (RFC 1514)
- Mail monitoring MIB (RFC 2249)
- X.500 directory monitoring MIB (RFC 1567)
- DNS server/resolver MIB extensions (RFC 1611/1612)
- Network services monitoring MIB (RFC 2248)
- Printer MIB (RFC 1759)
- Uninterruptible power supply MIB (RFC 1628)
- Relational database management system MIB (RFC 1697)
- System-level MOs for applications (RFC 2287)
- Application MIB [STB98, IET98]

The host resources MIB enables hosts to specify management information that characterizes the system; this includes information indicating characteristic sizes of the different memories, describing attached peripherals, documenting installed software, and relating to running processes. From the standpoint of a systems administrator, this MIB information is, of course, still too limited [GUN95, HKN96]. But the RFCs mentioned also signal an opening to other areas of application for Internet management. We will talk about the RMON-MIB again in section 6.5.

6.2.2 SNMPv2-SMI

This section will discuss the development of the Internet information model and the resulting enhancements to SNMPv1-SMI. The first set of SNMPv2-RFCs (RFC 1441 to 1452) was published in 1992; it was relieved by the current specifications RFC 1901 to 1908.

Extension of information model, new subtrees in the registration tree

The original Internet registration tree contained only subtrees (1) to (4) under 1.3.6.1 (compare Figure 6.3); the subtrees `security` (5) and `SNMPv2` (6) were added during the development of `SNMPv2`. The nodes `snmpDomains` (1), `snmpProxys` (2), and `snmpModules` (3) appear under `SNMPv2`. The structural node snmpModules contains the subtrees `snmpMIB` (1), `snmpM2M` (2), and `partyMIB` (3). `SnmpMIB` (1) largely replaces the subtree snmp (1) in MIB II and refers to SNMPv2, which is not directly interoperable with SNMPv1. The `system` group and other MIB II groups have been extended by several MOs. The `snmpDomains` group refers to the different versions of SNMPv2 on underlying protocol

stacks. Details about the MIB extensions can be found in RFCs 1906 and 1907 and in [MIL97, PEM97].

New object types in SNMPv2

The ASN.1 data types allowed in the *object type macro* (compare section 6.2.1) have also been extended. New ones that have been added since version 1 include nonnegative integers ("Unsigned32"), larger counters ("Counter64"), OSI addresses ("Nsap address"), and BITS, a construct that represents an enumeration of named bits.

Changed structure of object-type macro

Even the macro itself defined by SNMPv1 has been altered in SNMPv2 (RFC 1902) and now contains the parts `SYNTAX`, `UnitsParts`, `MAX-ACCESS`, `STATUS`, `DESCRIPTION`, `ReferPart`, `IndexPart`, and `DefValPart`. The extended object types appear in `SYNTAX`; `UnitsParts` contains as optional text "dimensions" such as "seconds" and "notifications" associated with the MIB variable. In a (semantically) orderly form, `MAX-ACCESS` has the values `not-accessible`, `accessible-for-notify`, `read-only`, `read-write`, and `read-create`. The new `STATUS` values are `current`, `deprecated`, and `obsolete`. `ReferPart` can contain reference text to other modules. `IndexPart` permits flexible table descriptions, and `DefValPart` the definition of default values. It is important to mention the problem caused by having different versions of the Internet information model. The addition of new MIB groups or variables means adding new branches to the registration tree; however, if variables are no longer necessary because of new versions, they still have to remain in the MIB (once registered, the number is forever assigned to the original object).

New SNMPv2 macros

In SNMPv1, events were simply defined by the protocol and not by individual MIBs. SNMPv2-SMI (SMIv2) introduced new macros, such as the `NOTIFICATION-TYPE` macro (RFC 1905), which permits a flexible specification of traps. Macros incorporating text conventions (RFC 1903) for specifying frequently used reference types are also available. The `AGENT-CAPABILITY` macro describes the capabilities of SNMPv2 agents in the form of supported modules, MOs, and values. It roughly corresponds to OSI management knowledge. And, lastly, the `MODULE-COMPLIANCE` macro can be used to describe the minimum requirements for an MIB implementation. This replaces the use of "mandatory" and "optional" in the `OBJECT-TYPE` macro itself.

To sum up, SMIv2 advances management information modeling. The object type macro allows finer granularity when defining MOs. The text conventions allow more formal specification of behavior and reusability of specification. Also table handling has become more flexible. Formalizing "row status" allows manager create and delete table rows; also a certain refinement of table rows is allowed in version

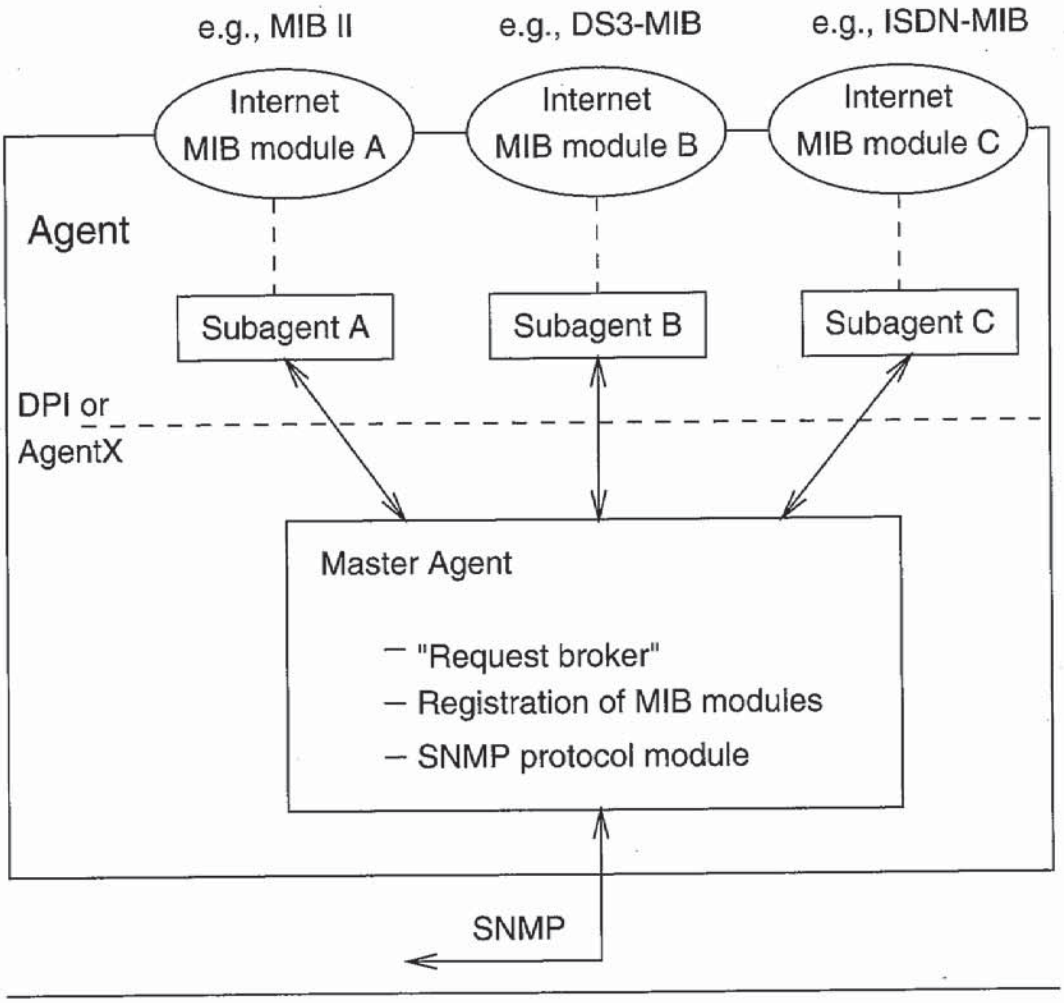

Figure 6.5
AgentX and DPI

2. For more details with respect to SMIv2-tables, see RFC 1902 or [STA96].

Although the SNMP version 2 management protocol has not caught on, the same does not apply to the expanded information model SNMPv2-SMI (abbreviated to SMIv2). According to IETF specification rules, object type macros based on SMIv2 are being used to describe all newer Internet MIBs. Since SMIv2 is largely a superset of SMIv1, publicly available translators can be used to convert a v2-based MIB for use in a v1-product or agent or manager. Some loss of functionality does occur.

Coexistence of independent MIBs on an agent through DPI

The existence of so many MIBs from different sources has inevitably resulted in a move to support the coexistence of several MIB models on one agent. This possibility is described in RFC 1592. Subagents that communicate with the (main) agents of a managed node over the *distributed protocol interface* (DPI 2.0) are allocated to the individual MIB modules (see Figure 6.5). The master agent distributes the requested

operations of the manager transmitted by the management protocol SNMP (see section 6.3) among the subagents. The subagents collect the management information relating to their MIB modules and pass on the results to the master agent. DPI allows subagents and MIBs to be added and removed dynamically without the need for a recompilation of the master agent. The local MIB modules are addressable by names since the DPI contains a registration mechanism. The agent systems are therefore more flexible and easier to scale because not all subagents have to run at the same time.

AgentX as a further development of DPI

The *agent extensibility (agentX) protocol* is a further development of DPI (RFC 2257). This protocol makes an even more distinct separation between master agent and subagent. According to the RFC, "The master agent is MIB ignorant and SNMP omniscient, while the subagent is SNMP ignorant and MIB omniscient for the MIB variables it instantiates." The master agents interpret the SNMP protocol operations and transform them into agentX protocol operations; the subagents themselves are responsible for any management-relevant instrumentation, such as connections to concrete resources. Subagents can be developed with agentX with no need to consider other subagents that might already exist on the component.

6.3 Internet Communication Model (SNMP)

The core of the Internet communication model is the simple network management protocol (SNMP). So far, two versions of SNMP exist—version 1 and several variants of version 2. We describe these two versions in the following subsections. In section 6.5, we will sketch a third version, SNMPv3, which is still in the design process.

6.3.1 SNMP Version 1

In Internet management, resources are also managed, in other words, supervised and controlled, through access to the characteristic values of the MIBs representing the resources. As demonstrated in section 6.2, managed objects are the leaves in the agent MIBs. Communication between manager and agent is over the simple network management protocol (SNMP), which was initially specified in RFC 1157.

Description of SNMP operations

The manager is able to access the remote agent MIB using SNMP operations. An SNMP agent receives the manager's requests, carries out the actions required, and generates an appropriate response. During

this process, the protocol works asynchronously, meaning that the manager can initiate a request without having to wait for the agent to supply a response (implementations of SNMP may or may not "block" while a request is outstanding). In addition to various manager-initiated operations, SNMP also provides a trap message that can be used by agents to transmit information to the manager without having received a prior request from the manager. SNMP provides a total of four protocol interactions, which are shown in Figure 6.6.

Get-request-operation

- Read access (get-request-operation): The manager generates a *GetRequest* PDU in which it enters the object instance values from the agent MIB that should be transmitted in a *GetResponse* PDU from the agent. This operation is atomic with SNMPv1: Either all the values are retrieved or none is. The identifier given in the PDU structure (Figure 6.7) identifies an MO entity. With simple MIB variables, the identifier is formed by a "0" being attached to the object identifier in the Internet MIB. As explained earlier, access to table values is controlled by a yet-to-be-specified index column of the table; this specifies the index object type. Access to the *n*th entity (therefore the *n*th table row) of object type X in a table is through an access identifier ⟨object ID of X⟩, ⟨value of the index object type in the *n*th table row⟩. The question arises concerning the calculation of the actual value of the index object type in connection with this access method: Looking at the example in section 6.3, how is it possible to access entries in the table `ipRouteTable` if the IP destination address contained in the `ipRoute Dest` is not known? The answer to this question is closely linked to the SNMP operation `get-next`, which allows table columns to be retrieved row by row even if the precise access identifier is not known.

Get-next-operation

- Browsing the agent MIB (`get-next`-operation): The get-next PDU is almost identical to the get-request PDU; it has the same PDU exchange pattern and the same format. In the get-request PDU, each variable in the variable-bindings list refers to an object instance whose value is to be returned. With get-next there is a difference. The manager generates a *GetNextRequest* PDU, in which it enters the object identifier of an object type or an object instance. In a *GetResponse* PDU from the agent, it receives the access identifier and the value, in lexicographic order, of the "next" object instance for the object identifier indicated in the request. More than one object identifier can be entered in a `get-next-request` PDU, and the next object instance for each object identifier is determined by the agent. Typically, the get-next operation is used to access the

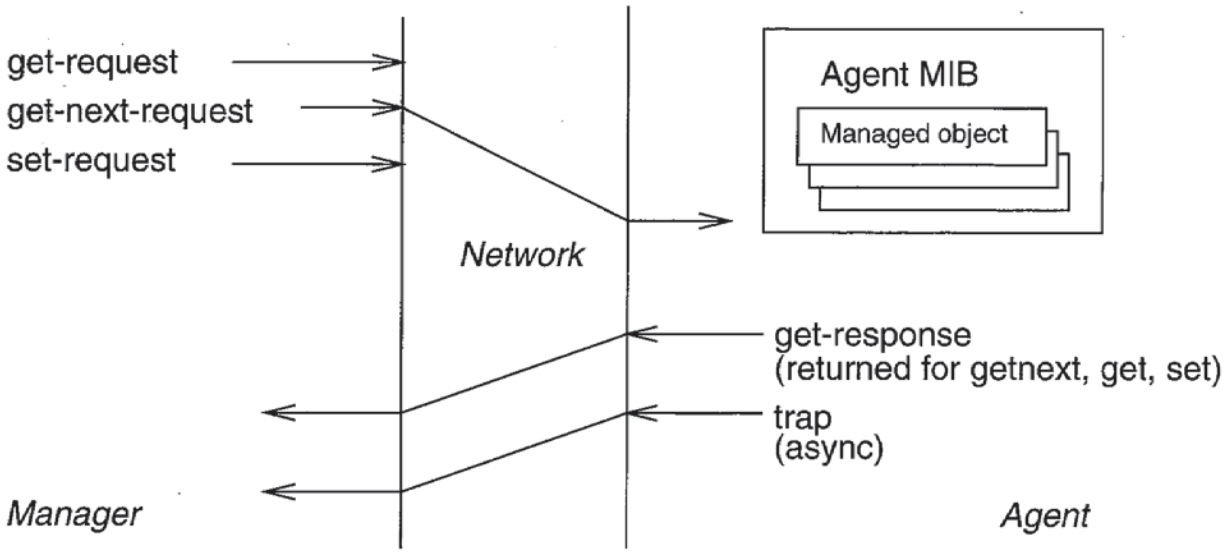

Figure 6.6
SNMP operations

next instance of an object in the MIB. It could access the next object in the database, or it could access the next row entry in a table. The get-next PDU invokes the accompanying get response. The keys to the search in the database for the get-next are the MO names of the previous get. There are several possible outcomes of a get-next operation depending on what is requested (e.g., object ID of a simple object in MIB, object-instance ID of a simple object in MIB, OID of table, OID of table entry, columnar-object ID, columnar-object-instance ID, OID that does not match). Get-next provides an elegant way for retrieving unknown objects and accessing table values. For example, to get the first row of a table, perform a get-next with every object in the table, identified by ⟨object ID of X⟩.0. To get the next row, resend the response returned to the first get-next, and so on.

- Write access (`set`-operation): The manager generates a *SetRequest* PDU for the agent in which it enters the object instances together with the new values to be set. The agent indicates the success or failure of the write action in a *GetResponse* PDU. (Note: There are no *SetResponse* PDUs.)
- Notification by agents (`trap`-operation): The agent generates a trap PDU in which it notifies the manager of certain events without having received a prior request from the manager to do so. In SNMPv1, these were limited to a set of seven trap types.

Structure of SNMP protocol data units

The different interactions are described in more detail in the description of the PDUs used in SNMP. The exact syntactic PDU structure is defined in the form of an ASN.1 specification in RFC 1157. `GetRequest`, `GetNextRequest`, `GetResponse`, and `SetRequest` have the same structure and consist of the following parts (Figure 6.7):

- *request-id* is used for the unique identification of a request (i.e., for correlation of get response with request).
- *error-status* enables the agent to notify the manager of possible errors that occur during the execution of a request. The following error states have been standardized: `noError`, `tooBig`, `noSuchName`, `badValue`, `readOnly`, `genErr`.
- *error-index* can be used by the agent in the case of an error to identify the first variable responsible for the error.
- *variable-binding* consists of a sequence of pairs of variable names and variable values. It is here that the manager enters the variable (object) names appearing in the request; with write access the value of the variable is the value to be set; otherwise, the variable value for a request is set to 0. In its response, the agent transmits the variable names provided by the manager with the corresponding values (i.e., it fills in values by modifying the request buffer for simplified implementation).

Elements of a trap PDU

With the SNMP PDUs described so far, the manager can carry out polling only. Polling is based on a request–response scheme like the one that occurs in the PDU structure described. The trap PDU is the only protocol data unit in SNMP that allows the transmission of information that has been initiated by an agent and therefore is not based on the polling principle. Many SNMP implementations use the so-called trap-directed polling: A trap tells the manager to get object values (i.e., poll them), for example, in case of an event having occurred. This is in contrast to OSI events; here notifications issued asynchronously by an agent carry all affected attribute values with them. This trap PDU is structured as follows:

- *PDU-type* for trap PDU is characterized by type = 4.
- *enterprise* contains the object identifier of the object that has produced the trap.
- *agent-addr* describes the network address of the SNMP agent that sent the trap.

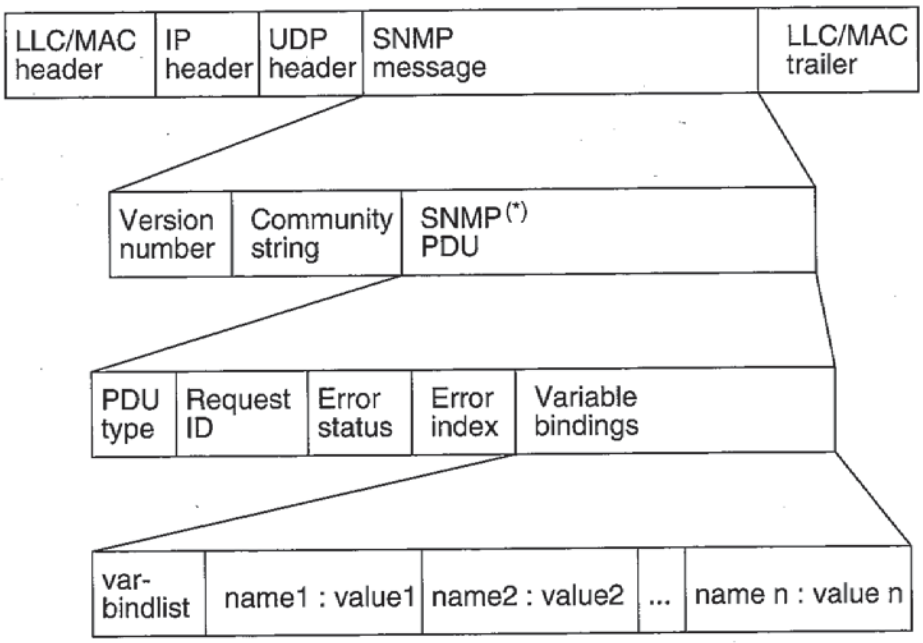

Figure 6.7
SNMP message format

- *generic trap* provides a rough identification of the trap based on the following standardized trap types: coldStart (e.g., if agent's configuration or SNMP entity was altered—typically, an unexpected restart), warmStart (e.g., after reinitializing—typically routine restart), linkDown (e.g., failure in one of the agent's links—variable-bindings field refers to related interface), linkUp, authenticationFailure, egpNeighborLoss (signals that an EGP neighbor is now down), and enterpriseSpecific (signals that an enterprise-specific event occurred).
- *specific trap* allows a further enterprise-specific classification of "enterpriseSpecific" type traps beyond the seven generic types.
- *time stamp* contains the time ticks that had passed since the last initialization of the network, at the time this event occurred at the agent.
- *variable-bindings* offers additional information relating to the trap. The significance of this field is implementation specific.

Traps are therefore similar to "predefined weak interrupt conditions." The manager can but does not have to react. On the agent side, some threshold monitoring can be carried out before a trap is sent. A manager will normally respond to a trap using selective polling.

Figure 6.8 sums up the SNMP definitions as specified in RFC 1157. An SNMP entity constructs the appropriate PDU using the ASN.1 structure given in the definitions. Then the PDU is passed to an authentication service (if present) together with the source and destination transport addresses and a community name. The authentication service may perform encryption or any other transformation. Then the SNMP message is constructed according to Figure 6.7. This new ASN.1 object is then encoded, using the basic encoding rules, and passed to the transport service (e.g., UDP). At the destination, an SNMP entity does a syntax check of the SNMP message. Then it verifies the version number. If there is an authentication service, the message is delivered to it. If authentication succeeds, the SNMP PDU is returned as an ASN.1 object. After some syntax check, the SNMP PDU is processed. Figure 6.9 illustrates the receipt of the SNMP PDUs on the agent side.

Community string as an SNMP security mechanism

The security mechanism used in SNMP is an extremely simple concept: In SNMP, each protocol data unit to be transmitted is packed into an SNMP message (Figure 6.7), which in addition to the PDU itself, contains a version number and a community string relevant to the security aspect. The community string serves as a type of password for MIB access since the agent processes only those SNMP messages that match the string configured in the agent, received from authorized managers.

Because of the simple concept, this form of authentication is also called a *trivial authentication algorithm.* A major disadvantage of the procedure is that by listening in on the network traffic a nonauthorized user (or a router on the path between manager and agent) can easily determine the community strings (the permitted passwords) configured in the SNMP agents and therefore carry out any number of operations on an agent MIB. The procedure also lacks the mechanism to cope with other threats, such as the falsification of SNMP messages (alterations of bridge filter settings), the repetition of SNMP messages (e.g., reboot requests to components), and the unauthorized disclosure of the contents of SNMP messages that have been listened in on (e.g., passwords for terminal servers).

In the IAB's view, extending SNMP to include appropriate security procedures such as the cryptographic checksum algorithm MD5 (RFC 1321) and the data encryption standard (DES) in cipher block chaining mode [FIPS-PUB46-2 and 81] was an urgent priority in its efforts to improve new SNMP versions (e.g., SNMPv2p). However, IP itself has since been extended to provide both confidentiality and packet authentication for application protocols, not just for SNMP. This simplifies the

```
RFC1157-SNMP DEFINITIONS ::= BEGIN

IMPORTS
    ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
        FROM RFC1155-SMI;

-- top-level message
Message ::= SEQUENCE {version INTEGER {version-1(0) },      -- version-1 for this RFC
          community OCTET STRING,                           -- community name
          data ANY          -- e.g., PDUs if trivial authentication is being used

-- protocol data units
PDUs ::= CHOICE { get-request          GetRequest-PDU,
                  get-next-request     GetNextRequest-PDU,
                  get-response         GetResponse-PDU,
                  set-request          SetRequest-PDU,
                  trap                 Trap-PDU}


-- PDUs
GetRequest-PDU ::=      [0] IMPLICIT PDU
GetNextRequest-PDU ::=  [1] IMPLICIT PDU
GetResponse-PDU ::=     [2] IMPLICIT PDU
SetRequest-PDU ::=      [3] IMPLICIT PDU

PDU ::= SEQUENCE { request-id INTEGER,
                   error-status INTEGER {                   -- sometimes ignored
                                 noError(0),
                                 tooBig(1),
                                 noSuchName(2),
                                 badValue(3),
                                 readOnly(4),
                                 genErr(5)},
                   error-index INTEGER,                     -- sometimes ignored
                   variable-bindings VarBindList}           -- values are sometimes ignored

Trap-PDU ::= [4] IMPLICIT SEQUENCE {
                   enterprise OBJECT IDENTIFIER,            -- type of object generating
                                                            -- trap, see sysObjectID in RFC 1155
                   agent-addr NetworkAddress,               -- address of object generating trap
                   generic-trap INTEGER {                   -- generic trap type
                                 coldStart(0),
                                 warmStart(1),
                                 linkDown(2),
                                 linkUp(3),
                                 authenticationFailure(4),
                                 egpNeighborLoss(5),
                                 enterpriseSpecific(6)},
                   specific-trap INTEGER,                   -- specific code, present even
                                                            -- if generic-trap is not enterpriseSpecific
                                                            -- enterpriseSpecific
                   time-stamp TimeTicks,                    -- time elapsed between the last
                                                            -- (re)initialization of the network
                                                            -- entity and the generation of the trap
                   variable-bindings VarBindList}           -- "interesting" information

-- variable bindings
VarBind ::= SEQUENCE {name ObjectName,
                      value ObjectSyntax }

VarBindList ::= SEQUENCE OF VarBind

END
```

Figure 6.8
SNMP definition (RFC 1157)

```
procedure receive-getrequest;
begin
    if object not available for get then
        issue getresponse (noSuchName, index)
    else if generated PDU too big then
        issue getresponse (tooBig)
    else if value not retrievable for some other reason then
        issue getresponse (genErr, index)
    else issue getresponse (variablebindings)
end;

procedure receive-getnextrequest;
begin
    if no next object available for get then
        issue getresponse (noSuchName, index)
    else if generated PDU too big then
        issue getresponse (tooBig)
    else if value not retrievable for some other reason then
        issue getresponse (genErr, index)
    else issue getresponse (variablebindings)
end;

procedure receive-setrequest;
begin
    if object not available for set then
        issue getresponse (noSuchName, index)
    else if inconsistent object value then
        issue getresponse (badValue, index)
    else if generated PDU too big then
        issue getresponse (tooBig)
    else if value not settable for some other reason then
        issue getresponse (genErr, index)
    else issue getresponse (variablebindings)
end;
```

Figure 6.9
Receipt of SNMP PDUs

special security requirements that must still be addressed by extending SNMP.

Figure 6.7 also illustrates the normal embedding of SNMP in the Internet protocol world. SNMP is typically based on the connectionless transport protocol UDP. SNMP PDUs are therefore coded in accordance with a simplified subset of the ASN.1 basic encoding rules (ISO 8825-1). SNMP traps are normally received at UDP port 162, the other SNMP messages at UDP port 161.

In addition to the most frequently used mapping of SNMP to UDP, SNMP can now also support OSI protocol stacks, Ethernet, Novell-IPX, and AppleTalk. SNMP proxies, or management gateways, are then used to manage the resources that do not support SNMP. In order to ease the use of SNMP functionality when writing SNMP-based management applications, SNMP APIs were developed, such as SNMP++ (see Chapter 15).

6.3.2 SNMP Version 2

As mentioned earlier, SNMPv1 is the dominant management protocol in the data communications environment. It can even be implemented in relatively simple resources. Almost all devices support SNMP, at least to the extent that interoperability is guaranteed within the scope of MIB II. Many management applications are available, and SNMP has proved itself in many areas of application.

Nevertheless, a number of disadvantages and limitations have already surfaced:

Weaknesses in SNMP version 1

- SNMPv1 incorporates a very weak security concept. Community strings are "clear text" transmitted passwords.
- SNMPv1 does not support the efficient transmission of large volumes of management data.
- SNMPv1 supports only a polling scheme that is always initiated only by the manager. However, an agent can initiate polling through the use of traps (trap-directed polling). All management information must be requested explicitly by the manager.
- SNMPv1 allows only primitive support of asynchronous events. There are very few permanent and globally defined traps. The trap mechanism functions only for previously specified events. Supplementary management information is not specified for traps, and traps are not related to MIB definitions of objects that generate them.
- The reliance on connectionless UDP results in the danger of data loss.

Work on proposals aimed at improving SNMPv1 therefore commenced in 1992, and a preliminary version of SNMPv2, including initial prototype implementations (RFC 1441 to 1452), was proposed as a set of draft standards in April 1993. This SNMPv2 version, described later, is also called SNMPv2 Classic or sometimes, because of its party-

based security model, SNMPv2p. However, because of the extensive and complicated security mechanisms, the new proposals were not well received in the market. Attempts at simplifying the proposal were undertaken by the SNMPv2 working group in 1994–95 and recorded in draft standards RFC 1902 to 1908 in 1996. Other developments were produced under the names SNMPv2*, SNMPv2u, and SNMPv2c. The last is a version that uses community strings (compare with SNMPv1). SNMPv2u is based on a user-oriented security model, and SNMPv2* attempts to present an approach that is a mixture of v2p and v2u. Needless to say, this proliferation resulted in market confusion.

New PDU types and protocol operations

The new SNMPv2-PDU structure is the same for all SNMPv2 variants. It has the same structure for all protocol operations. The new PDU types *GetBulkRequest, InformRequest,* and *SNMPv2-trap* are particularly singled out for mention. Added to the already existing SNMPv1-PDU types GetRequest, GetNextRequest, GetResponse, and SetRequest, this makes seven PDU types that are available. Another one, the report PDU, is under consideration. *GetBulk* requires only one call for a readout of a whole table (or parts of a table). The purpose of this PDU is to minimize the number of protocol exchanges required to retrieve a very large amount of management information. This means it takes only one PDU to be executed instead of the previously necessary sequence of lexicographically ordered GetNextRequest PDUs. One GetBulk identifies the max-repetitions and is handled by the agent as though that number of GetNext's had been received in sequence. This means that the GetBulkRequest operation uses the same selection principle as the GetNextRequest operation. That is, selection is always of the next object in lexicographic order. The difference now is that multiple lexicographic successors can be selected. In essence, GetBulk works as follows. It includes a list of (N+R) variable names in the variable-binding list. For each of the first N names, retrieval is done in the same fashion as for GetNext; for each of the last R names, multiple lexicographic successors are returned. The GetBulk is controlled by two fields: nonrepeaters (the number of variables for which a single lexicographic successor is to be returned) and max-repetitions (the number of successors to be returned for the remaining variables in the variable-binding list that has the length L). The following holds: N = max(min(nonrepeaters, L), 0); M = max(max repetitions, 0); and R = L – N. The total number of variable-binding pairs that can be produced is $N + (M \times R)$.

Get-bulk-operation

The InformRequest PDU is used for manager-to-manager communication (not agent-to-manager). All PDU types, including the trap

PDU, now have a uniform syntactical basic structure. Compared to SNMPv1, the new PDUs offer much more diversity in the error codes.

Extensive security concept with authentication and encryption for SNMPv2p variant

The controversy surrounding the entire version 2 development centers on the security concept. There was an interest in countering the security deficiencies mentioned in section 6.3—masquerades, information modification, message stream modification, disclosure. Although the security concepts of SNMPv2 were not successful, they will be described here because the same problems exist as before. Even if an accepted standard fails to incorporate an adequate security specification, this has to be considered relevant as far as actual implementations are concerned because of the impact on the security of management. Although SNMPv2p has now been classified as historic, it is the SNMPv2 variant that offers the most comprehensive security concept. It uses the following security mechanisms:

- The data encryption standard (DES) is applied as the encryption cipher algorithm.
- Packet source and content authentication is implicit through the use of a shared secret key based on the MD5 (message digest no. 5) algorithm.
- A time stamp procedure using weakly synchronized clocks is offered. This enables the monitoring of sequences that are out of order (i.e., prevents replay attacks) and also offers clock synchronization.

These three mechanisms allow the implementation of three different security levels that are activated in accordance with the relevant management transaction per the coding in the SNMPv2 message:

- No authentication; no encryption (`nonsecure`)
- Authentication using MD5; no encryption (`not private` or `confidential`)
- MD5 authentication; DES encryption (`authenticated, private`)

The security-relevant context, called party, of an SNMPv2p entity must always be entered in the header of an SNMP message for the sender and the receiver (compare Figure 6.10). Different parties can be used between the same manager and agent to define authorized views of particular MIBs or objects. An SNMPv2p message contains the format fields privDst, authinfo, dstParty, srcParty, and context in front of the actual SNMPv2 PDU. *dstParty* and *srcParty* contain the security-relevant role (context, party) of the receiver or the sender. *Context* indicates the MIB view (section of the MIB). *Authinfo* provides

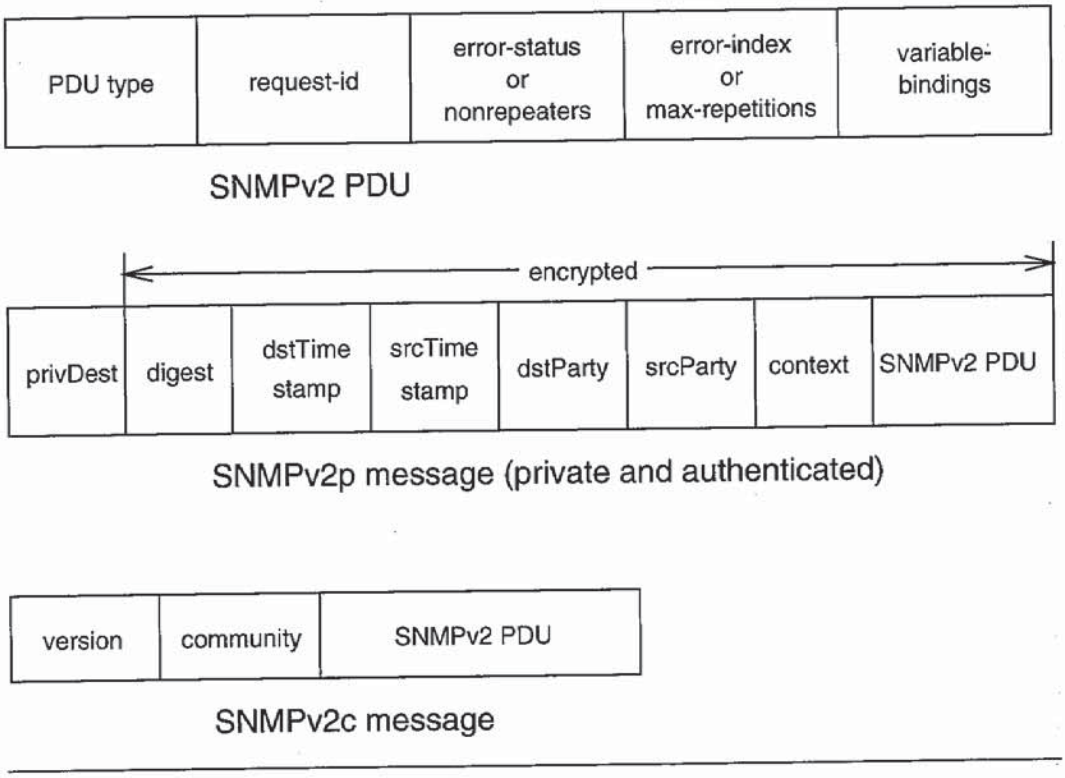

Figure 6.10
Structure of an SNMPv2 PDU and SNMPv2 message

information (e.g., digest, time stamp) that is relevant to the authentication protocol. *PrivDst* repeats the object identifier of dstParty. If encryption (privacy) is selected, then all fields of the SNMPv2p message except privDst are coded using DES. Refer to RFCs 1445 to 1447 or to [STA96] for information on how security procedures are actually carried out.

The SNMPv2c variant uses the community string for security

The controversy over security (e.g., complexity of implementation and administration, limitation to single authentication and encryption algorithm was considered too weak but required for export) is avoided in SNMPv2c although it also uses SNMPv2 PDUs (compare Figure 6.10) to take advantage of the new protocol operations. The security provisions in the SNMPv2 message of this variant are limited to the community string familiar from SNMPv1.

Introduction of intermediate managers and manager-to-manager communication

The last modification to version 2 that deserves mention is the one to the organizational model. As illustrated in Figure 6.1, SNMPv1 incorporates only manager-to-agent communication. It has long been recognized that substructures are required for the management of large networks in order to keep certain activities as local as possible. Version 2 supports hierarchical management structures (compare Figure 4.4). Managed nodes can have a dual role (dual-mode nodes)—as managers for nodes at a lower level in the hierarchy and simultaneously as agents

to a superior manager. The manager-to-manager MIB (M2M-MIB) and inform-PDU were developed to enable this communication between managers. However, as a result of subsequent developments, M2M-MIB has achieved the status of historic and therefore is not currently valid.

It should be obvious from the discussion that SNMPv1 and SNMPv2 are not interoperable. This fact and the complicated security mechanisms that impact the implementation and time budget of agents are the reasons why SNMPv2—despite existing products—has not yet been successful in the market; only very few firms have supported SNMPv2p or the later SNMPv2c. The market is waiting for the standardization of SNMPv3 to be completed (see section 6.5). Yet the expanded SMIv2 information model is being widely used already. Coexistence with SNMPv1 is essential for a migration strategy to SNMPv2; this is dealt with by RFC 1908. The latter recommends two solutions for protecting SNMP investments: a proxy agent or managers that can cope with both versions (bilingual managers). Recommendations are also available for bilingual agents (RFC 2089).

6.4 Remote Monitoring MIBs

The remote monitoring (RMON) standard was developed on the basis of experience gained with proprietary network probes. Probes are measurement logging components used to monitor LAN traffic in different LAN segments; the logged values are then fed to an evaluator component, usually a LAN analyzer. A certain amount of data preprocessing can take place in the probes.

The RMON-MIB represents a standardized probe

The RMON-MIB (RFC 1757) describes the managed objects of a standardized remote network monitoring device and to a certain extent represents an abstraction of a probe. The demands placed by its objects on the supporting agents are higher than those of a standard MIB II because RMON also defines the results of statistical calculations as managed objects. In this sense, RMON can be interpreted as the first extension of Internet management in the direction of OSI systems management functions. The network traffic resulting from polling is reduced somewhat because of the additional preprocessing that takes place in the agents. Furthermore, the quantity of SNMP-standardized manageable resources increases due to the RMON-MIB because RMON is particularly oriented toward LAN supervision, thus also below the IP layer. So long as sufficient agent resources are available, supervisory data can continue to be collected even if the SNMP connection to the

manager is broken off. Moreover, in principle, RMON agents can be managed by more than one manager.

We are describing the RMON-MIBs in detail because of their important function in actual network management compared to the other MIBs mentioned at the end of section 6.2.1. RMON is the first standard to provide more flexible monitoring. It also illustrates a kind of functional model defined for use with SNMP.

The RMON-MIB is subtree mib-2.16 in the Internet registration tree

The RMON-MIB (RFC 1757) occupies subtree 16 below structural node MIB II (i.e., node 1.3.6.1.2.16) in the Internet registration tree and consists of ten MIB groups (see Figure 6.11). The RMON2-MIB (RFC 2021), which was developed later, currently consists of additional groups that are attached as additional subtrees below mib-2.16; the first ten groups are also referred to as RMON1. Overall, RMON comprises more than 200 managed objects. All RMON groups are optional, which means that "RMON-conformant" products do not have to support all groups. This is taken into account when interoperability is being considered for heterogeneous environments. However, certain dependencies exist: the `event group` is a prerequisite of the `alarm group`; the `filter group` is necessary for the `packetCapture group`; and the `hostTopN group` needs the `host group`.

The RMON-MIB consists of ten MIB groups

The `statistics group` supplies the following management information (per segment): number of packets, bytes, broadcasts, multicasts and their distribution, information about lost packets, and five other error types (e.g., collisions, CRC errors, packets that are too small or too large). Some of the same parameters also exist in the interface group of MIB II, but there they relate to individual devices, whereas the focus here is on traffic load and the error rates of LAN segments. The granularity of the collected data depends, of course, on the quality of the underlying LAN attachment units.

The `history group` provides the basis for a trend analysis that allows the recording of monitoring information from the statistics group. Monitoring frequency and measurement intervals for individual interfaces are specified in the `controlTable`. The results are then filed in other tables (`historyTable`). Table management (conventions, arithmetic, changes and deletions to the table rows) in the SNMP area is very complicated and cannot be covered in detail in this book. We direct you to the original documents (RFCs 1757 and 2021) or to the literature (e.g., [STA96]). There is some analogy to the OSI summarization function.

The `alarm group` uses threshold analyses to define alarms. A hysteresis mechanism referring to absolute and relative changes in measurement values is specified using the upper and lower threshold values

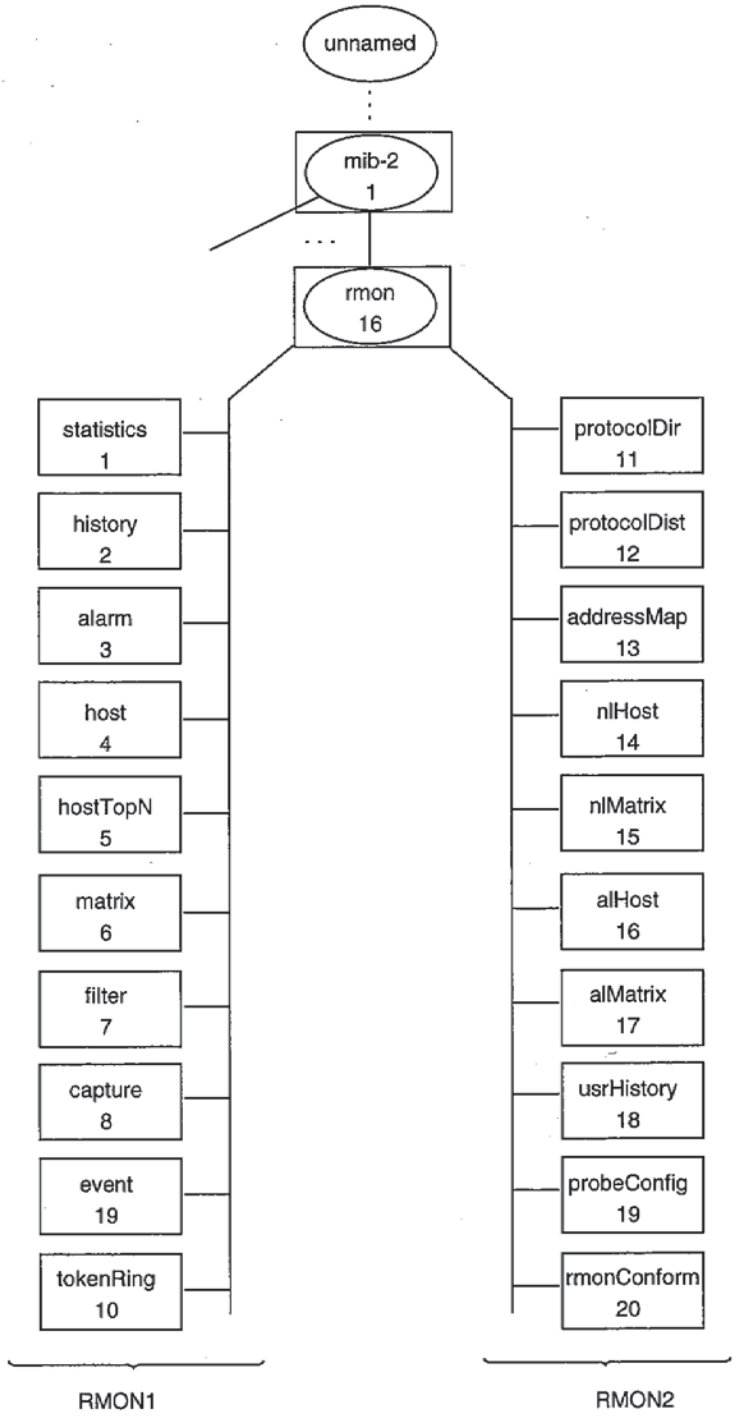

Figure 6.11
RMON-MIB

assigned to individual measurement values (load values, error rates). The alarms are then transmitted in trap notifications from SNMP to the manager.

For each host initiating network traffic in the segment, the `host group` contains statistical information derived from the monitoring of

MAC addresses in the packets. The host group (see Figures 6.12 and 6.13; the latter taken from [STA96]) for its part consists of tables. A control table (`hostControlTable`) defines which hosts are to be monitored and the measurement period. The `hostTable` contains the measurement data (e.g., packets sent or received, broadcasts, defective packets) organized according to MAC addresses, whereas the `hostTimeTable` essentially contains the same information but organized by the time when it originated.

The `hostTopN group` enables statistics to be obtained from those hosts that maintain statistics lists on projected measurement values. Yet this evaluation is carried out by the agent and not by the manager for the projected measurement interval, thereby producing a reduction in SNMP traffic.

The `matrix group` allows direction-related traffic matrices to be established for the source–sink pairs of MAC addresses (i.e., the hosts).

The `filter group` provides for the definition and application of filters (i.e., the conditions for recording packets). Data filters specify packet selection on the basis of bit patterns in the packets; status filters define the selection using status information such as CRC errors and packet length violations. Complex filter conditions can be formulated using condition expressions (AND/OR expressions). A packet stream that successfully passes a filter test is called a *channel.* It is possible for counters and events to be associated with a channel. There is some functional analogy to an ISO discriminator.

The `packetCapture group` permits the definition per channel of buffer hierarchies for the packet streams received from the filters.

Lastly, the `event group` supports the definition of events that, if necessary, trigger actions that are defined elsewhere in the MIB. The conditions for the events could also be defined in other RMON-MIB groups.

As mentioned earlier, the RMON-MIB was originally developed for Ethernet LANs. RFC 1513 contains extensions for Token Ring LANs (IEEE 802.5) that include the addition of specific table objects in the statistics and history groups and in the supplementary tokenRing group.

RMON2 extends the original RMON-MIB to include ten more groups

Whereas RMON1 primarily carries out a protocol analysis of OSI levels 1 and 2, RMON2 (RFC 2021) extends this analysis to include levels 3–7. This extended analysis improves the monitoring of the internetworking as well as of the application level (email, file transfer, WWW) because important information for the manager is already processed on the RMON agent, in other words, on remote systems. Further, the agent can process data that represents more closely a

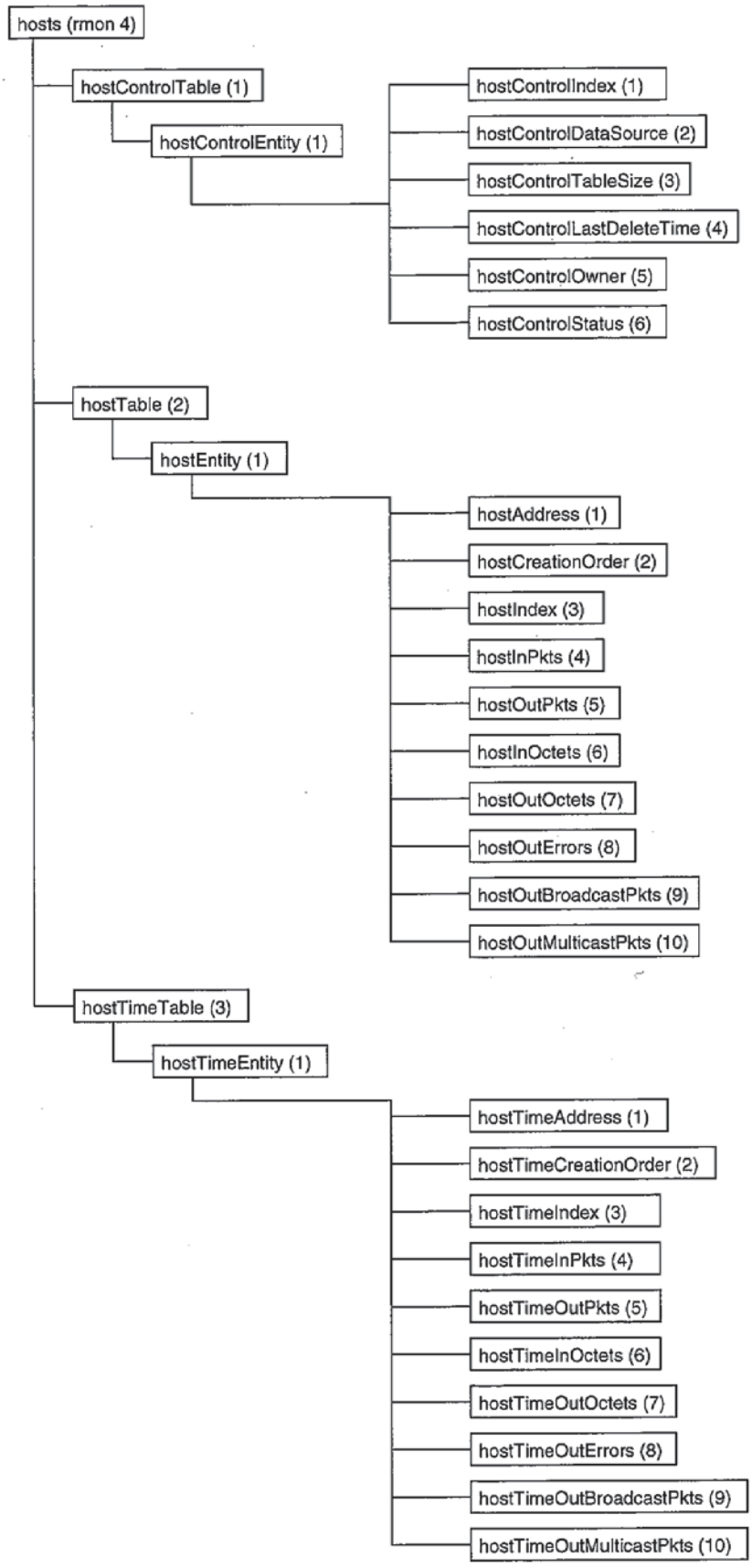

Figure 6.12
Host group of RMON

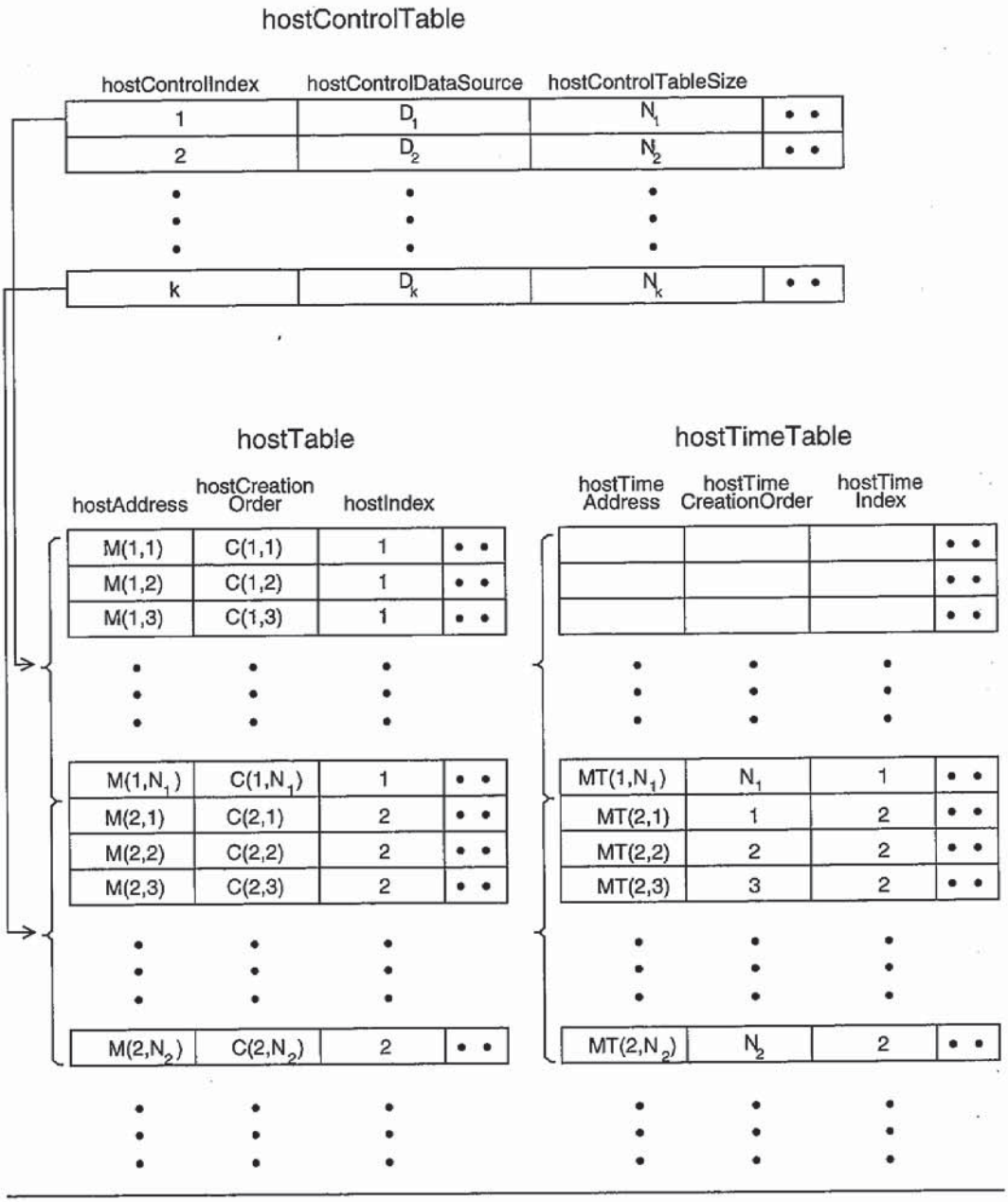

Figure 6.13
Example of hostTables

temporal "snapshot." A manager retrieving data, even with GetBulk, has only data that is not collected at the same instant and can be difficult to relate.

Compared to RMON1, RMON2 offers more flexibility in the use of tables. First, external objects can also be used as index types when object type macros are specified; second, time filters are available for indexing. As a result, only those measurement values that have changed since the last polling cycle are addressed and then transmitted to the manager.

RMON2 extends RMON by ten further MIB groups (11–20) below the Internet structural node mib-2.rmon (see Figure 6.11).

The `protocolDir group` lists in a table all the protocols that can be monitored by the agent. The group also supplies information about monitored protocol parameters, PDU structures, and so forth.

One of the tasks of the `protocolDistribution group` is to collect protocol-related statistical data about packets. This enables bandwidths to be established according to the respective layer.

The `addressMap` contains mappings between MAC and network addresses and supports the generation of topology maps and autodiscovery.

The `nlHost group` keeps track of the network traffic (especially IP traffic) related to a specific network address (IP address), allowing the requested data to be made available over an appropriate control table. The `nlMatrix group` incorporates the traffic matrix at the network level. The `alHost` and `alMatrix groups` are the groups for the application layer that can be queried for specific management information relating to certain hosts and in accordance with application protocols supported by the `protocolDir group`.

The `usrHistory group` permits application-specific data to be collected and stored with the assistance of the alarm and history groups.

The `probeConfig` and `rmonConformance groups` describe information that is needed in order to evaluate and manage the interoperability of RMON implementations. This includes knowledge about the MIB groups supported by agents and the values of the configuration parameters for the probes as RMON agents.

Evaluation of RMON

Compared to the MIBs outlined in earlier sections of the book, the RMON-MIB is all and all considerably more advanced when it comes to the important task of remote network monitoring. But it has to be borne in mind that RMON is standardized only for Ethernet and Token Ring but not, for example, for Fast Ethernet or FDDI. The RMON-MIB offers a number of advantages such as a reduction in network traffic, data preprocessing, proactive management, offline data entry, and the availability of several managers for the RMON agents. The prerequisite is that the probes are well placed in the network. The advantages are offset by complicated table management and the need for complex agents. As a consequence, dedicated devices may be needed to cope with this aspect. However, a number of network components (such as hubs and switches, or even interface cards for PCs) already integrate RMON agents, but often only the first four groups (statistics, history, hostTable, alarm) are supported. Any evaluation of RMON products

must take into account the number of supported RMON groups, the maximum number of nodes that can be monitored, necessary CPU and storage capacity for filtering, the recording and storage of traffic information, packet throughput (maximum clearance time), and which RMON applications are available on a specific management platform. The latter applies to follow-up procedures for RMON data; these particularly include procedures for demand assessment, QoS management, capacity planning, and reporting systems. Although RMON probes are capable of collecting all types of data, the managers are responsible for all management-relevant responses through the use of appropriate applications.

6.5 SNMP Version 3 and Other Developments

SNMPv3 as convergence of SNMP proposals

There have been various efforts in recent years to integrate a new security concept and other improvements into Internet management. Owing to a considerable difference of opinion within the SNMPv2 working group, no final proposals have yet been drawn up. Since 1997, the SNMPv3 working group has been working toward a convergence of previous proposals. Since the work is still ongoing, we can present only some of the known objectives.

The goal of the working group is to produce a uniform security and management framework for SNMP that will enable the implementation of a secure "set," in other words, a secure controlling management, using the simplest means possible and incorporating modularity for adaptability to different areas of application. Modularity and coexistence are required to enable agents to operate several different security models, encryption mechanisms, and SNMP message formats simultaneously and to allow the replacement or expansion of agent modules. It has now been recognized that a static allocation of management functionality is not useful against the background of the functional diversity and dynamics of management requirements, and that this kind of approach is also not scalable. These considerations also dictate that it should be possible to apply a modular structure to an MIB so that the modules can more readily be used to deal with processing needs, specifically that they can be allocated to management-relevant functional areas. This is another case where an object-oriented approach would be more effective in satisfying the respective requirements!

The current results of the working group have been compiled in the overview document "An Architecture for Describing SNMP Management Frameworks" (RFC 2271). The architecture presented is modular

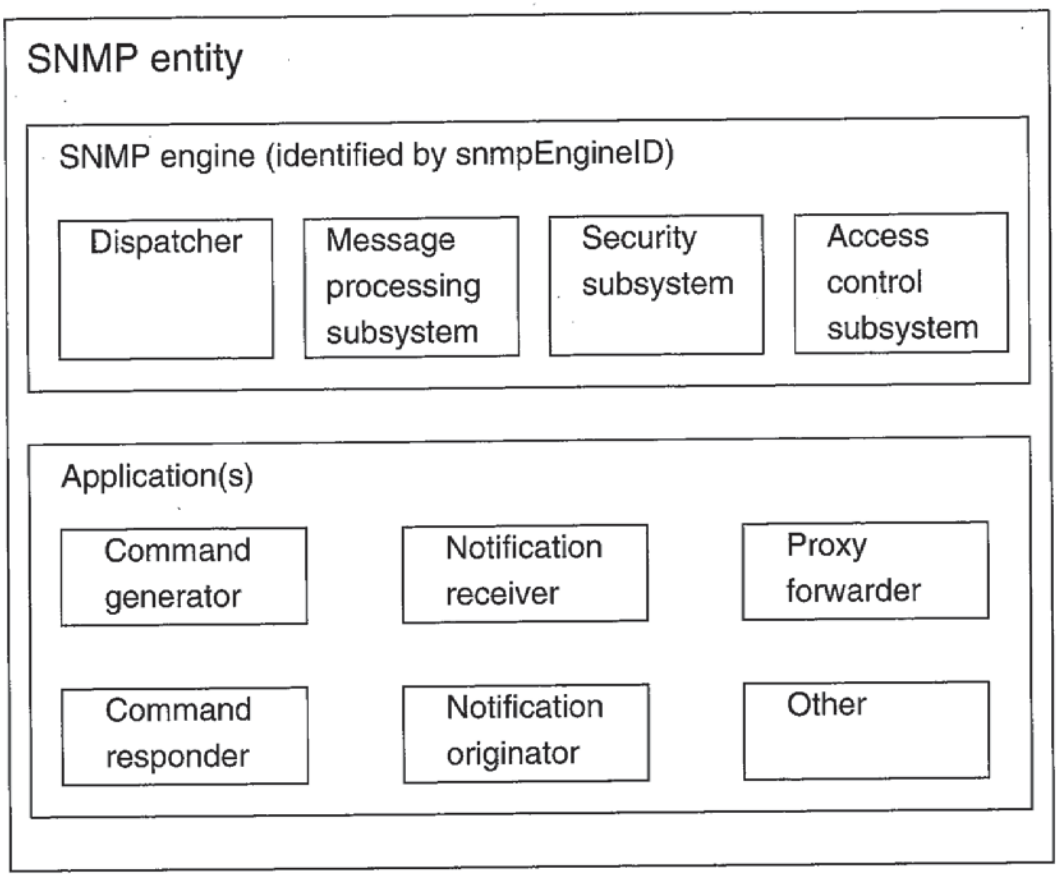

Figure 6.14
Structure of an SNMPv3 entity

and flexible in structure, enabling it to incorporate other standard developments. At the core and consisting of several subsystems is an *SNMPEngine* on which all different kinds of SNMP applications can be run (Figure 6.14). The subsystems in turn each consist of different modules capable of coexisting with one another.

The SNMP management system developed within this new SNMPv3 framework overall consists of:

- Several nodes each with an SNMP entity containing a *notification originator* and a *command responder* application. These correspond to the previous agents.
- At least one SNMP entity that contains a *command generator* or a *notification receiver* application. This corresponds to the traditional manager.
- A management protocol for exchanging information between SNMP entities.

It is obvious that the fundamental structure is the same for all SNMP entities, irrespective of whether their traditional role has been that of a manager, a midlevel manager, a proxy, or an agent.

The SNMP engine is the basic system for all SNMP entities

The engine consists of a dispatcher and the message processing, security, and access control subsystems. The *dispatcher* coordinates internal communication between the subsystems and is able to differentiate between the different modules coexisting within a subsystem. It directs incoming PDUs to the correct applications and maps outgoing PDUs to the appropriate underlying transport systems. The dispatcher provides applications with a service interface for which a whole range of service primitives has been specified.

The *message processing subsystem* handles incoming and outgoing SNMP messages. In so doing, it supports a number of coexisting modules each corresponding to a different message format or protocol version (e.g., SNMPv3, SNMPv1, SNMPv2c). The fact that modules can be replaced or added lays the migration groundwork for future protocol developments (RFC 2272).

The *access control subsystem* deals with issues controlling access to MIB objects and the corresponding access rights. This is another area where there is fundamentally support for more than one module; the standard module is based on the view-based access control model, for which modularized MIBs are a prerequisite (RFC 2275). This is a functional replacement for SNMPv2 parties.

The *security subsystem* also is designed to allow for the coexistence of different security modules that are controlled by the message processing subsystem. For example, one module could be based on SNMPv1 communities; another could implement the user-based security model (USM, RFC 2274) approach preferred by SNMPv3. The USM attempts to answer the following questions: Is the SNMP message genuine as well as on time, and who is requesting the SNMP operation? The USM therefore concerns itself with the threats "modification of information," "message stream modification," "masquerade," and "disclosure," but not with "denial of service" or "traffic analysis." The user for whom a security-relevant service is provided is called the "principal"; this user is identified by a *userName* that is mapped to a *securityName.* Hash functions first transform user passwords into nonlocal codes; from this code each SNMP engine is able to extrapolate its own local code (public and private keys) for the user. The USM also specifies algorithms for key update and authentication. Two protocols are defined for the latter: MD5 and secure hash algorithm (SHA, FIPS-PUP 180-1). Perhaps this will be replaced by the use of IPSEC. As with SNMPv2, loosely synchro-

nized clocks are used to monitor the timeliness of SNMP messages. DES is used for data encryption.

The *applications* in Figure 6.14 are management-relevant applications. In particular, these also contain the basic applications for an SNMP entity that are linked to the SNMP operations. For example, the SNMP operations SET and GET then are forwarded to the command responder application. This then has local access to different contexts supported by different MIB modules (RFC 2273).

Overall, SNMPv3 promises to be a flexible framework that will allow previous as well as new Internet management versions to coexist within one management architecture. Owing to the importance of compatibility, the SNMPv3 working group has also addressed this issue in a separate Internet draft document, "Coexistence Between Version 1, Version 2, and Version 3 of the Internet Standard Network Management Framework" (August 1998).

AgentX development

Another new approach that promotes flexible and dynamic agent structures is *agentX*, which was covered at the end of section 6.2.

DISMAN—a concept for management by delegation

New developments are also evident in another area. In section 4.4, we mentioned that a static allocation of management functionality is not appropriate for very large and dynamically distributed systems because of the lack of scalability. We referred to the development of management by delegation. The DISMAN working group is also looking at this concept for use in the Internet world. An Internet draft *Distributed Management Framework* has been drawn up by the group. To the DISMAN group, distributed management does not mean management functionality distributed in a statically set way according to one of the different possible forms of arrangement, but something that is "movable." This approach will allow the "distributed management" application to keep pace (scalability) with the changing needs of large distributed systems. This kind of dynamic adaptability is also optimal for dealing with changes in conditions (such as organizational changes and changed processes).

The idea behind the DISMAN framework is that the manager as the "DISMAN user" turns over jobs backed up with "credentials" to a "distributed manager" that executes the applications in a distributed form in entities called "management targets." A number of relevant services are recommended for, among other things, domain formation, describing management targets, and the safe delegation of credentials and their control. Domains are groups of systems that are subject to a management policy according to certain criteria. A description of the target systems of a delegation is required to enable criteria to be applied in determining whether a target system is even in a position

to carry out a particular task. The function of "credentials" is precisely as the name implies, and these are indispensable for security. Lastly, the delegated task itself must be described in a script language and its actual delegation controlled as a management action.

DISMAN-MIBs

Proposals for several related MIBs already exist. Among these are DISMAN-SERVICES-MIB, TARGET-MIB (to express targets for traps and script transfer), EVENT-MIB (based on the RMON alarm and event groups and intended as a replacement of those groups; it is also the successor and update of the SNMPv2's manager-to-manager MIB), Notification LOG-MIB, Expression MIB (to provide custom objects for the EVENT MIB), Schedule MIB (definitions of MOs for scheduling management operations), Remote Operations MIB, and SCRIPT-MIB (defines a standard interface for the delegation of management functions based on the Internet management framework; this comprises capabilities to transfer management scripts; for initiating, suspending, resuming, and terminating scripts; to transfer arguments and results; and to monitor and control running scripts). The respective documents have the status "Internet draft" (i.e., working papers of the IETF and its working groups, published in the last quarter of 1998). They are not yet standardized, but these efforts signal the first step the Internet has taken toward implementing management as a dynamically distributed application.

It is still not clear whether the high implementation cost will have an effect on the acceptance of the powerful DISMAN concept. It is also not definite whether DISMAN will require an SNMPv3 implementation or can also be used in SNMPv1 environments.

Internet management is undisputedly the dominant management architecture in the data communications world, particularly in the LAN and intranet areas. But the original simple concepts also have their limitations in terms of scalability, modeling complex managed objects, and the complexity of intelligent agents. In this respect object-oriented approaches offer definite advantages. But the Internet world is very dynamic, which is also evident from the Internet management concepts currently being developed in many different areas.

Anyone who wants to be kept up-to-date on the developments outlined in this chapter can do so over WWW. The general URL is *http://www.ietf.org/html.charters*. Information on the individual working groups can be obtained by adding */disman-charter.html*, *agentx-charter.html*, or *snmpv3-charter.html*.

6.6 Chapter Summary

Internet management is dominant in the field of classic data communication networks. The Internet management architecture is conceptually simpler compared to the OSI one.

The information model (SMI) is not truly object oriented; this means that the inheritance principle cannot be used for things like class definitions and allomorphism in the specification of managed objects. The Internet information model is oriented toward data type. The data types (referred to as object types) permitted for management information are simple ASN.1 data types or tables. The Internet MOs are specified as object types (grouped in MIBs) in the leaves of the Internet registration tree. The object entities (managed objects) are accessed through the identifier in the registration tree. The registration tree expresses only partial containment relationships. Agent MIBs implement entities of sections from the Internet MIBs. The object types are described on the basis of simple ASN.1 macros. MIB II is currently the standard MIB. A range of technology-specific MIBs as well as a large number of manufacturer-specific MIBs are also available. Although most MIBs have been oriented toward network management, MIBs are now increasingly available for systems and applications management.

RMON was introduced in order to take advantage of the increasing processing capability of agents, to handle the preprocessing of management information in agents, and to relieve the workload on the network and the management system. RMON allows management data to be collected and evaluated in agents on a table-controlled basis. To a degree, RMON thereby carries out the tasks handled by some of the systems management functions (SMFs) in OSI.

There are two versions of the Internet SMI. Version 2 adds several new subtrees to the registration tree. The macro that is used to define an object type has been changed slightly. Overall SMIv2 offers advancements in the modeling of management information. The formalization of row status allows table lines to be added or deleted (a rough and limited analogy to CMIS M-Create/M-Delete). The refinement of table lines that is possible has a slight analogy to ISO inheritance. The inclusion of text conventions allows a more formal specification of behavior and permits the reusability of specifications. SMIv2 is therefore the only version being used in new documents.

The Internet communication model defines the management protocol SNMP that provides access to agent MIBs. The use of simple operations enables read (get) or write (set) access to values of object

instances. GetNext can be used to browse through MIBs or tables. In version 1, data access by the manager consists of atomic operations with the response providing either all the data requested or none of it. There are few predefined traps that the agents are able to send asynchronously. These traps have been defined for SNMP, and in contrast to OSI notifications, they cannot be defined as object specific. SNMP version 1 is not a truly secure protocol; the community strings it uses are "clear text" passwords. SNMP version 2 offers many different variants of SNMP messages that differ from one another in the security features offered. The now standardized PDU structure, which also includes a trap PDU, is common to all the variants. GetBulk is a new PDU type and allows large quantities of management information to be requested and transmitted in a single operation. (However, the new PDU is not able to offer the flexibility of CMIS scoping and filtering.) Because of the success of SNMPv1 and because of the discrepancies existing in its security concept, SNMPv2 has not been able to make an impact on the market.

New developments in the Internet management environment are focusing on a general and flexible protocol platform for the coexistence of SNMP variants (SNMP engine of SNMPv3) and the aspect of management by delegation (DISMAN). However, none of these developments has been standardized as of yet.

