SNMP 协议

larkguo@gmail.com

2007-05-26

目录

1 简介

1.1 概况

1. SNMP 是为网络管理服务而定义的应用协议，在 1988 年 8 月首次定义，由 Internet 工程任务组织(Internet Engineering Task Force)(IETF)的研究小组为了解决 Internet 上的路由器管理问题而提出的，很快就在 RFC1157 中达到了正式标准。
2. SNMP 是 NMS 和代理之间的异步请求和相应协议。
3. SNMP 是由一系列协议组和规范组成的，它们提供了一种从网络上的设备中收集网络管理信息的方法。
4. SNMP 被设计成与协议无关，所以它可以在 IP，IPX，AppleTalk，OSI 以及其他用到的传输协议上被使用。
5. 从被管理设备中收集数据有两种方法：一种是轮询（polling-only）方法，另一种是基于中断（interrupt-based）的方法。
6. SNMP 消息全部通过 UDP 端口 161 接收，只有 Trap 信息采用 UDP 端口 162。

1.2 名词

SNMP，Simple Network Management Protocol：简单网络管理协议

它是一个标准的用于管理 IP 网络上结点的协议。此协议包括了监视和控制变量集以及用于监视设备的两个数据格式：SMI 和 MIB。

MIB，Management Information Base：管理信息库

由网络管理协议访问的管理对象数据库，它包括 SNMP 可以通过网络设备的 SNMP 管理代理进行设置的变量。

SMI，Structure of Management Information：管理信息结构

用于定义通过网络管理协议可访问的对象的规则。SMI 定义在 MIB 中使用的数据类型及网络资源在 MIB 中的名称或表示。

ASN.1，Abstract Syntax Notation One：抽象语法定义

用于定义语法的正式语言，在 SNMP 中它用于定义 SNMP 协议数据单元和对象的格式。

PDU，Protocol Data Unit：协议数据单元

在网络中传送的数据包。

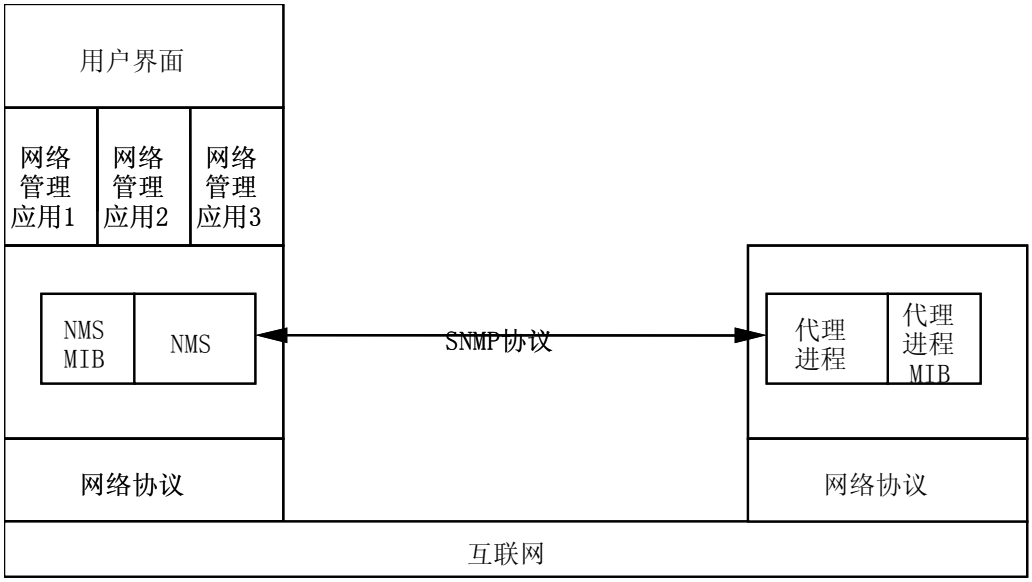
BER，basic encoding rules：基本编码规则

由 CCITT (X.209)和 ISO (ISO 8825)指定的编码规则，它描述了如何将 ASN.1 类型表示为字符串。

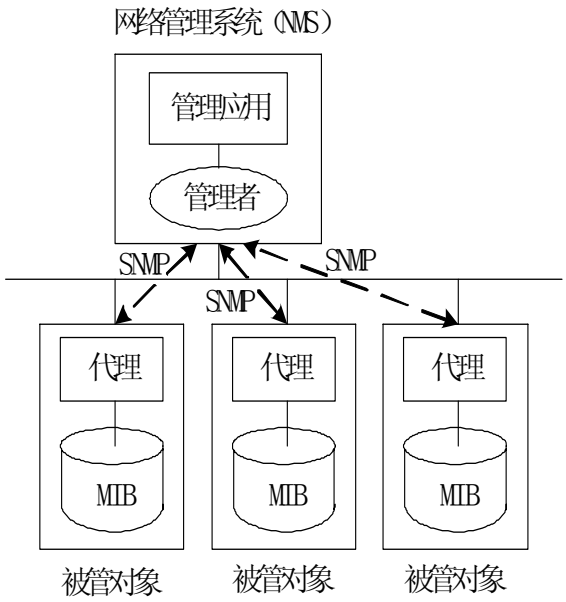
1.3 模型

SNMP 参考模型由以下 4 个主要部分构成：互联网络，网络协议，网络管理进程和被管网络

实体，如图所示：

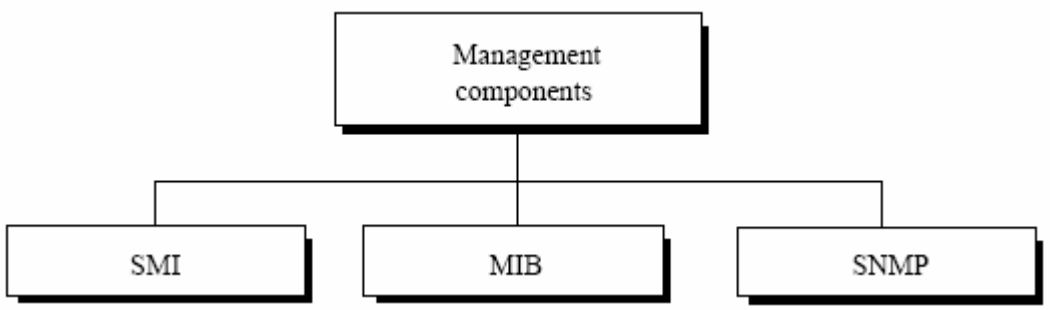


其中的核心内容是 SNMP 管理模型，如图所示：



2 组成

3 个主要的组成部分：SMI、MIB、协议.

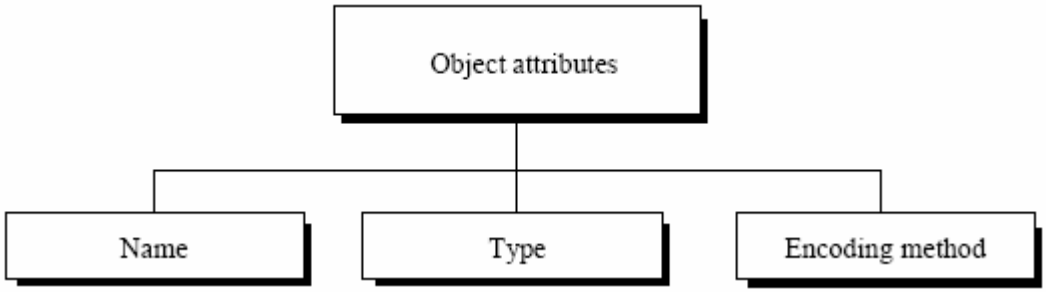


2.1 SMI

管理信息结构是管理信息库中对象定义和编码的基础。SMI 是对公共结构和一般类型的描述，和标识方法在一起，在现实中都要用到。

SMI 经常被比作数据库的模式，它描述 MIB 中的对象。

SMI 中最关键的原则是管理对象的形式化定义要用抽象语法记法（ASN.1）来描述。管理对象在现实中都是作为特定的 MIB 严格定义的，在 SMI 称为为对象类型。SNMP 的对象类型有 3 个用来描述其特征的最基本属性：名,语法,编码。



2.1.1.1 数据类型

(1) INTEGER：有些整形变量没有范围限制，有些整形变量定义为特殊的数值。
(2) OCTER STRING:0 或多个 8 位字节，每个字节值在 0－255 之间。
(3) DisplayString:0 或多个 8 位字节，每个字节必须是 ASCII 码，所有该类型的变量不能超过 255 个字符。
(4) OBJECT IDENTIFIER:一个任意长的非负整数序列，用于标记对象.
(5) NULL:相关变量还没有值。
(6) IPAddress:4 字节的 OCTER STRING，以网络序表示的 IP 地址，每个字节代表 IP 地址的一个字段。
(7) PhyAddress:OCTER STRING 类型，代表物理地址。
(8) Counter:非负整数，范围为 0－4294976295，达到最大后从 0 开始。
(9) Gauge:非负整数，范围为 0－4294976295，或增或减，达到最大值后锁定，直到复位。
(10) TimeTicks:时间计数器，以 0.01 秒递增，但是不同的变量可以有不同的递增幅度，所以定义该类变量时必须指定递增幅度。

(11) SEQUENCE:类似于结构，包括 0 个或多个元素，相当于 C 语言的结构体 struct
(12) SEQUENCE OF:向量，其所有元素具有相同的类型，相当于 C 语言的数组 array

2.1.1.2 Message

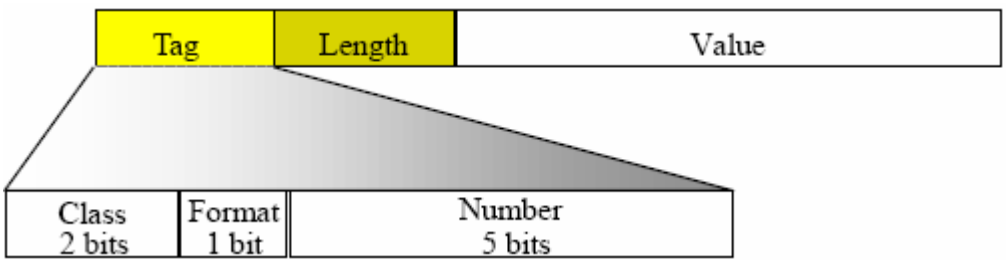
一条 SNMP 消息由版本号、SNMP 共同体名和协议数据单元(PDU)构成。The length of SNMP messages should not exceed 484 octets.


```
Message ::=
    SEQUENCE {
        version         INTEGER {version-1(0)},
        community       OCTET STRING,
        data                ANY
}
```

- **版本识别符（version identifier）：**确保 SNMP 代理使用相同的协议，每个 SNMP 代理都直接抛弃与自己协议版本不同的数据报。
- **团体名（Community Name）：**用于 SNMP 从代理对 SNMP 管理站进行认证；如果网络配置成要求验证时，SNMP 从代理将对团体名和管理站的 IP 地址进行认证，如果失败，SNMP 从代理将向管理站发送一个认证失败的 Trap 消息。共同体为一个字符串，这是管理进程和代理进程之间的口令，是明文格式，默认为 public。
- **协议数据单元（PDU）：**其中 PDU 指明了 SNMP 的消息类型及其相关参数。

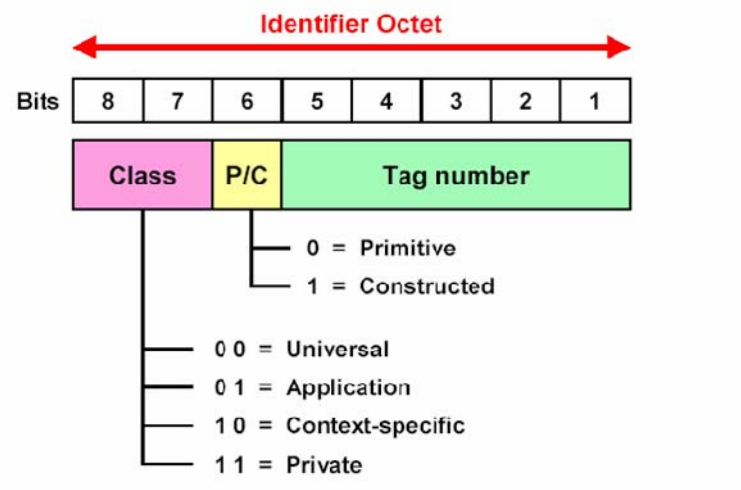
2.1.1.3 BER

- BER - 基本编码规则(Basic Encoding Rules)
- ITU-T(X.690)和 ISO(8825-1)标准
- 一种编码规格说明
- 描述如何将 ASN.1 类型的值编码成字节串(string of octets)的方法
- 基于一种称为 type-length-value (TLV)结构的方法，在 ASN.1 中，也称 identifier-length-content(ILC)

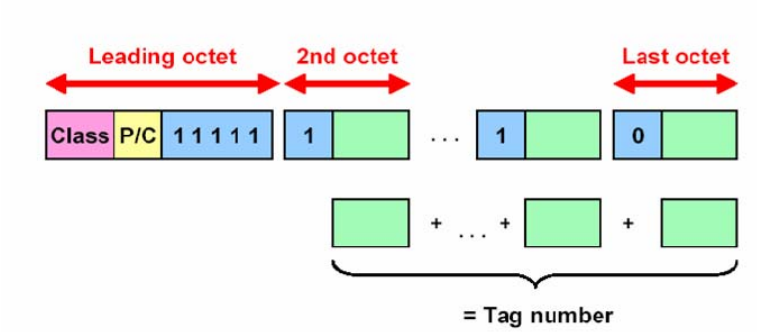


2.1.1.3.1 Identifier 字段

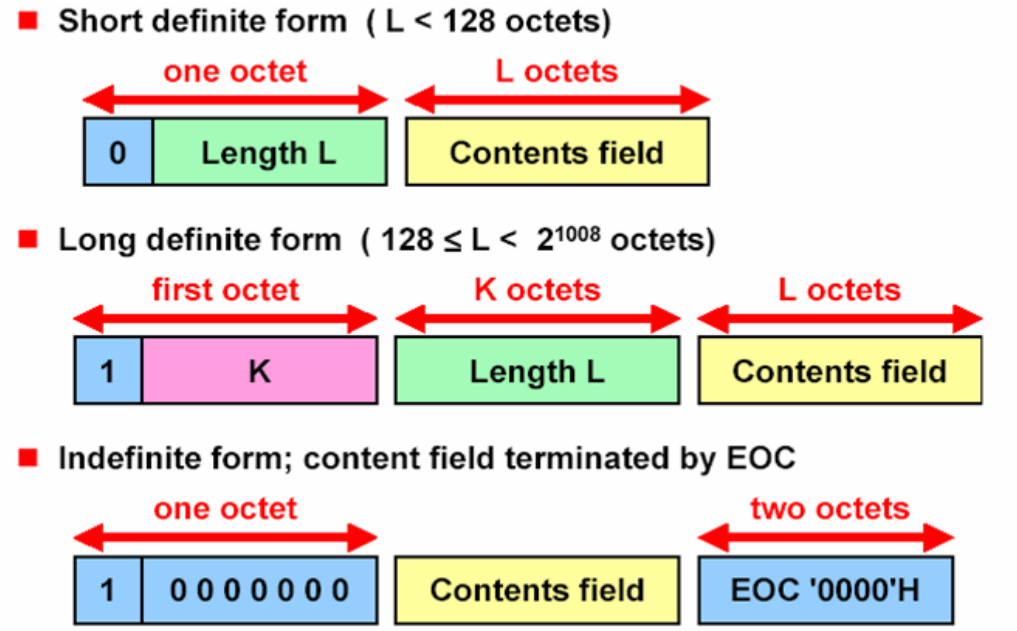
Tag number < 31:



Tag number >= 31:



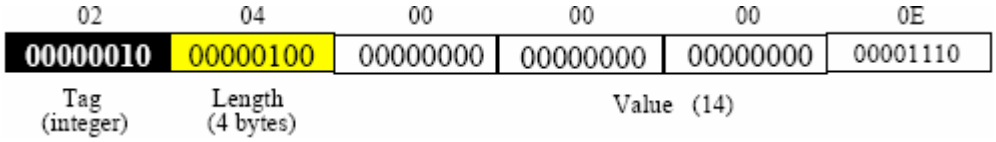
2.1.1.3.2 Length 字段



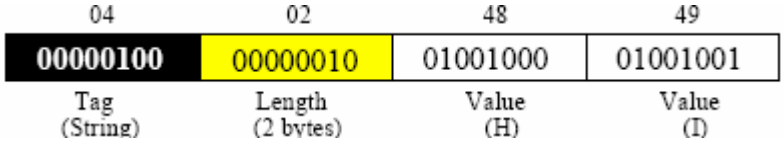
- 短格式
  - 既可用于基本类型，也可用于内容长度不超过 128 的构造类型
- 长格式
  - 既可用于基本类型，也可用于构造类型
  - 通常内容长度大于或等于 128
- 不定长格式
  - 仅用于构造类型
  - EOC 字节可看作是 tag 为 0 的基本类型，内容长度为 0

例

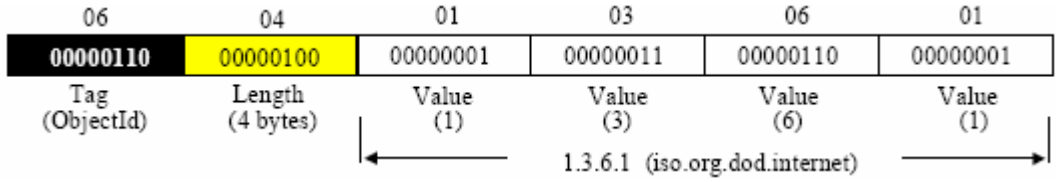
Example 1, integer 14



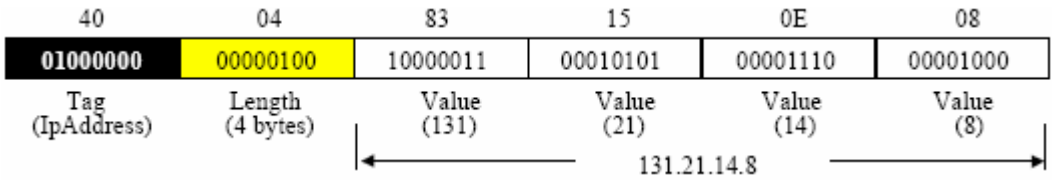
Example 2, string “HI”



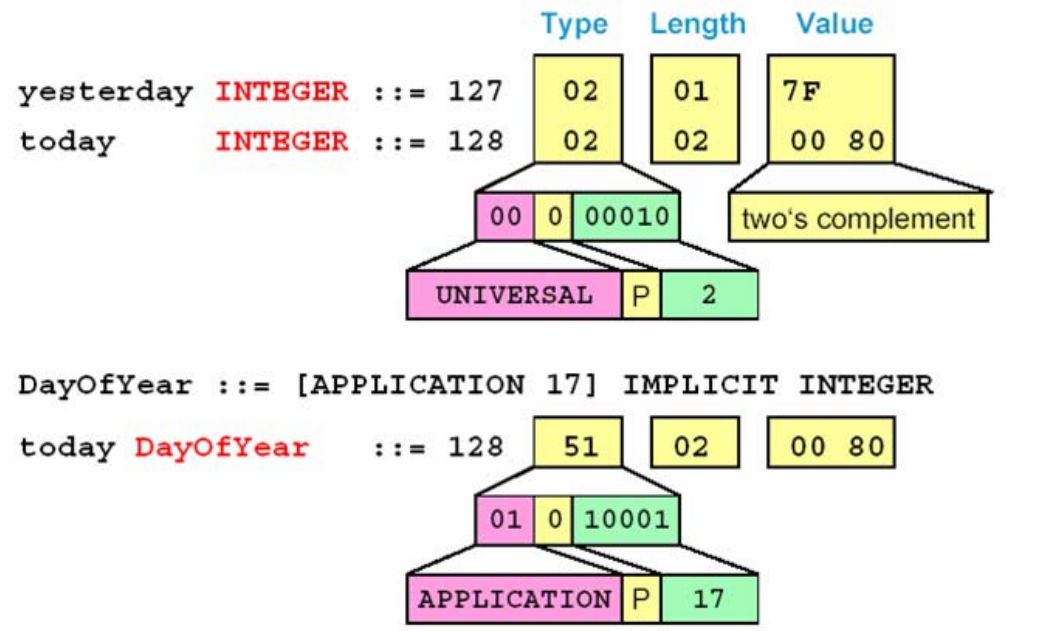
Example 3, ObjectIdentifier 1.3.6.1



Example 4, IPAddress 131.21.14.8



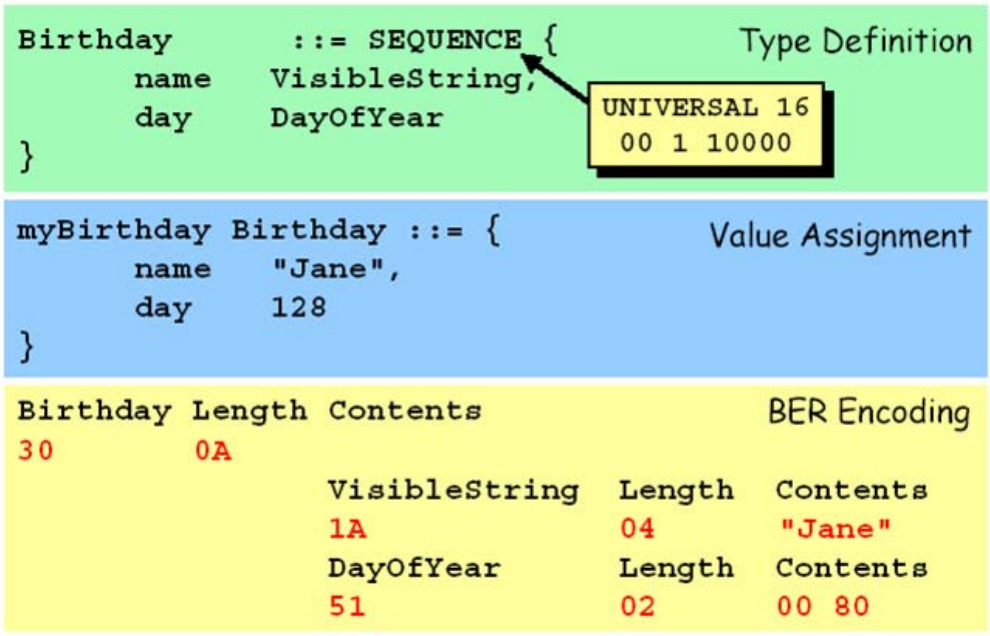
Example 5, INTEGER



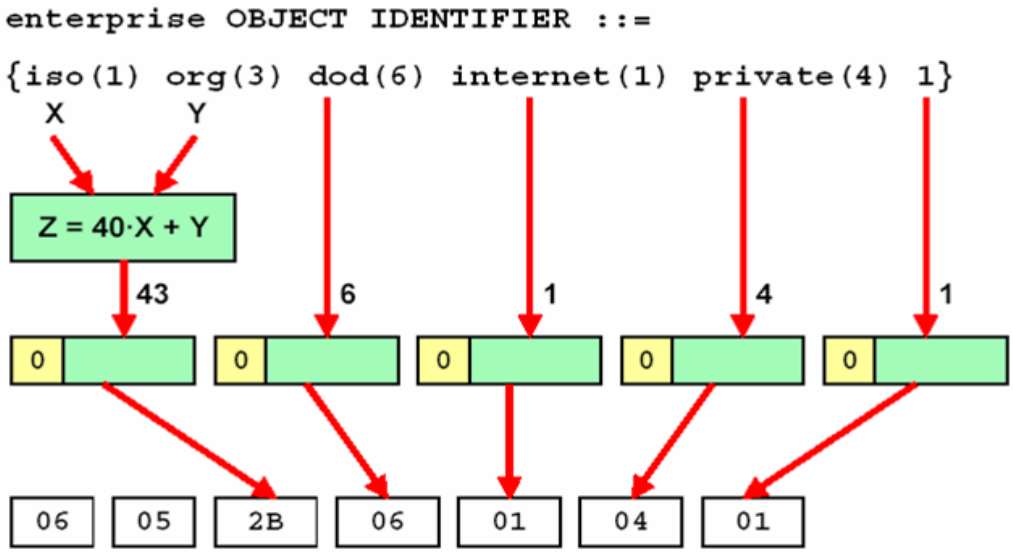
Example 6, INTEGER

- BER coding of two's complement integers
  - -129: 1111 1111 0111 1111 = 02 02 FF 7F
  - -128: 1111 1111 1000 0000 = 02 01 80
  - -127: 1111 1111 1000 0001 = 02 01 81
  - -1: 1111 1111 1111 1111 = 02 01 FF
  - 0: 0000 0000 0000 0000 = 02 00
  - 1: 0000 0000 0111 1111 = 02 01 01
  - 127: 0000 0000 0111 1111 = 02 01 7F
  - 128: 0000 0000 1000 0000 = 02 02 00 80
  - 129: 0000 0000 1000 0001 = 02 02 00 81

Example 7, SEQUENCE



Example 8,OBJECT IDENTIFIER



Example 9,OBJECT IDENTIFIER

- Coding of OID Root
  - ccitt(0): Z = Y {0..39}
  - iso(1): Z = 40 + Y {40..79}
  - joint-iso-ccitt(2): Z= 80 + Y {80..119}
- Coding of OID nude numbers
  - 类似于 Identifier 字段中的 Tag 编码
  - Range {0..127}:
    - 0XXX XXXX
  - Range {128..16383}:
    - 1XXX XXXX 0XXX XXXX

- Range {16384..2097151}:
  - 1XXX XXXX 1XXX XXXX 0XXX XXXX

- 例：RSA 数据安全公司 {1 2 840 113549}
  - 06 06 2a 86 48 86 f7 0d

2.1.1.3.3 BER 编码存在的问题

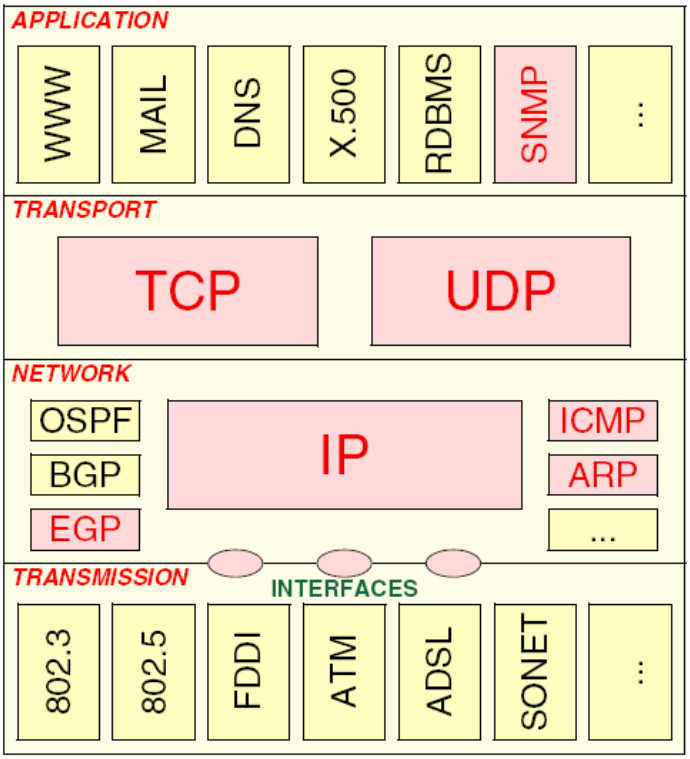
- 不唯一
  - 同一个值可能有超过 1 种合法的编码; 灵活,但容易造成混淆
- 示例
  - BIT STRING 值 01101110 01011101 11
  - BIT STRING 编码中,在长度字段后的第一个字节给出了最后一个字节中未用到的位数(本例中为 6)
  - 03 04 06 6e 5d c0（用 0 填充,短格式）
  - 03 04 06 6e 5d e0（用 100000 填充）
  - 03 81 04 06 6e 5d e0（长格式）
  - 23 09 （构造式编码）

03 03 00 6e 5d （“0110111001011101”+”11”）

03 02 06 c0

- 解决方案(两种方向)
  - DER: BER 子集,只使用定长编码
  - CER: BER 子集,基于不定长编码

2.2 MIB



MIB 定义了可以通过网络管理协议进行访问的管理对象的集合。第一组 RFC 定义的 MIB 成为 MIB-I。接下来的 RFC1213 中又添加了新的对象，成为了正式的标准，称为 MIB-II。

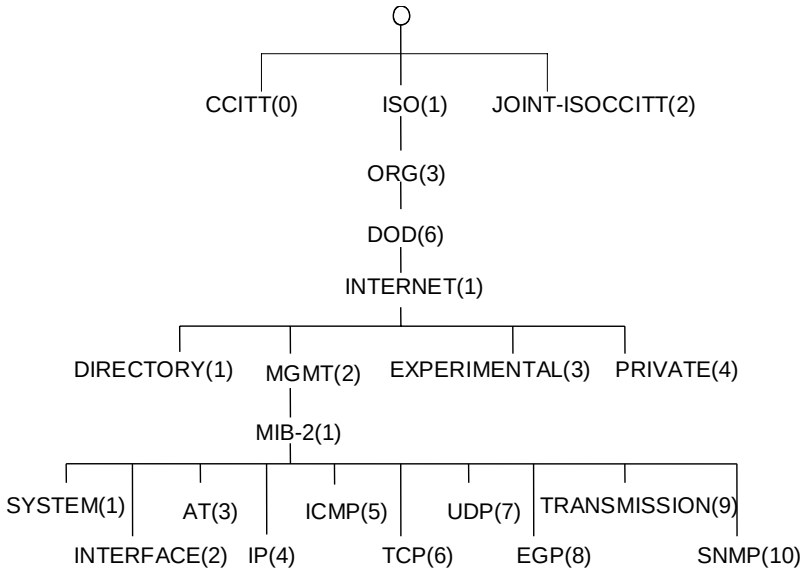
MIB 经常被当作管理对象的虚拟的数据库。

MIB 可以描述为一棵树，各个数据项构成了树的叶结点。每个 MIB 对象都有一个唯一的对象标识符（OID）来标识和命名，这个标识符取决于 MIB 对象在树中的位置，而对象的实例也有标识符，由对象类的对象标识符加上实例标识符构成的。

SNMP MIB 的对象标识符结构定义了三个主要分支：

- 国际电报电话咨询委员会 CCITT（现在的国际电信联盟的电信部门 ITU-T）负责管理分支 0
- 国际标准化组织 ISO 负责管理分支 1，
- CCITT 和 ISO 联合机构管理分支 2。

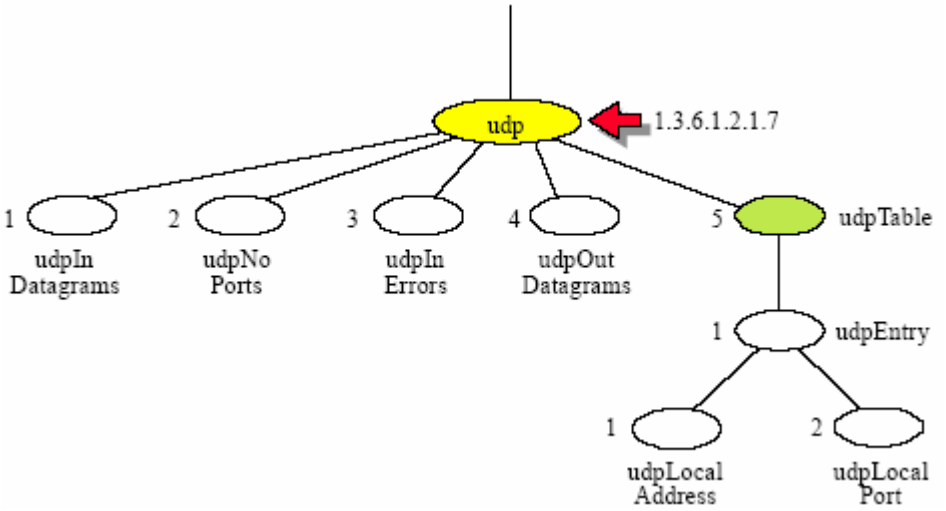
Internet 的 MIB 都在分支 1，属于 ISO.ORG.DOD.INTERNET 子树，具体如下图所示。



RFC1213 中定义了 Internet 标准的 MIB 和 MIB-II，共包含 171 个对象，分成 11 组，下表对各组总体信息作了具体解释。

MIB类型	类型说明
System	主机或网关信息
Interface	网络接口信息
AT	地址转换信息
IP	网际信息
ICMP	网际控制报文协议
TCP	传输控制协议
UDP	用户数据报协议
EGP	外部网关协议
Transmission	传输媒体信息
SNMP	简单网络管理协议

2.2.1UDP 组



UDP 组中包含几个变量和一个表格。变量为：

udpInDatagram(1) 1.3.6.1.2.1.7.1
udpNoPorts(2) 1.3.6.1.2.1.7.2
udpInErrors(3) 1.3.6.1.2.1.7.3
udpOutDatagram(4) 1.3.6.1.2.1.7.4
udpTable(5) 1.3.6.1.2.1.7.5

名称	数据类型	R/W	描述
udpInDatagram	Counter	R	UDP 数据报输入数
udpNoPorts	Counter	R	没有发送到有效端口的 UDP 数据报个数
udpInErrors	Counter	R	接收到的有错误的 UDP 数据报个数
udpOutDatagram	Counter	R	UDP 数据报输出数

在 udpTable 中有 2 个变量：

UDP 监听表，索引＝<udpLocalAddress>.<udpLocalPort>			
名称	数据类型	R/W	描述
udpLocalAddress	IpAddress	R	监听进程的本地 IP 地址，0.0.0.0 代表接收任何接口的数据报
udpLocalPort	[0..65535]	R	监听进程的本地端口号

2.2.1.1实例标识

对 MIB 变量进行操作，必须对 MIB 的每个变量进行标识。只有叶子节点是可操作的 SNMP 没法处理表格的一整行或一整列.

- 简单变量

对于简单变量的处理是通过在其对象标识后面添加".0"处理。例如对象标识是 1.3.6.1.2.1.7.1，则实例标识是 1.3.6.1.2.1.7.1.0。

- 表格

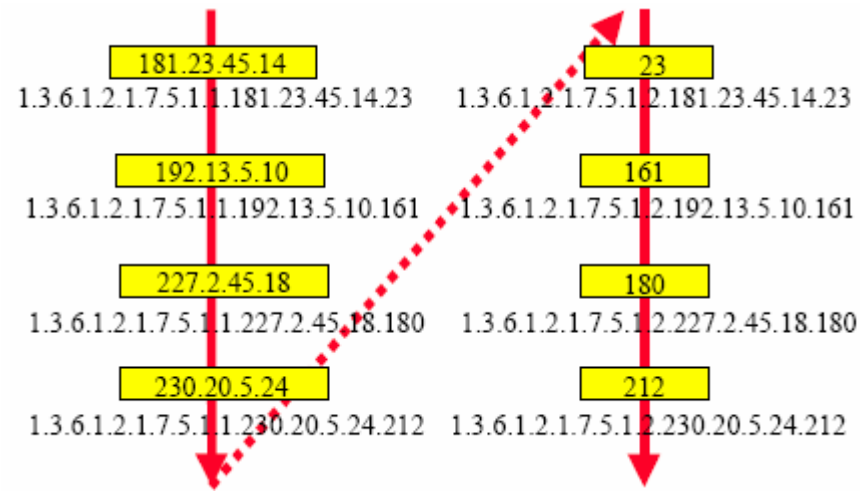
每个 MIB 中的索引都包含一个以上的索引。对于 UDP 监听表来说，MIB 定义了包含两个变量的联合索引。假定 UDP 监听表中有 3 行具体成员：

0. 0. 0. 0 67
0. 0. 0. 0 161
0. 0. 0. 0 520

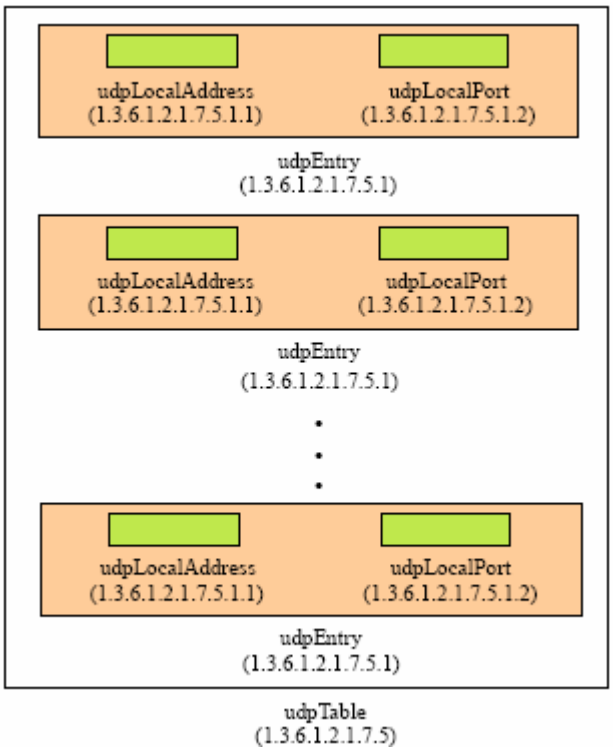
这表明系统将从端口 67、161 和 520 接收来自任何接口的 UDP 数据报。这三行数据处理后为：

行	对象标识	简称	值
1	1.3.6.1.2.1.7.5.1.1.0.0.0.0.67 1.3.6.1.2.1.7.5.1.2.0.0.0.0.67	UdpLocalAddress.0.0.0.0.67 UdpLocalPort.0.0.0.67	0.0.0.0 67
2	1.3.6.1.2.1.7.5.1.1.0.0.0.0. 161 1.3.6.1.2.1.7.5.1.2.0.0.0.0. 161	UdpLocalAddress.0.0.0.0. 161 UdpLocalPort.0.0.0. 161	0.0.0.0 161
3	1.3.6.1.2.1.7.5.1.1.0.0.0.0. 520 1.3.6.1.2.1.7.5.1.2.0.0.0.0. 520	UdpLocalAddress.0.0.0.0. 520 UdpLocalPort.0.0.0. 520	0.0.0.0 520

- 字典式排序



MIB 中按照对象标识进行排序有一个隐含规则，MIB 表格是根据其对象标识按照字典的顺序进行排序的。上面表格排序后如下所示：



行	对象标识	简称	值
1	1.3.6.1.2.1.7.5.1.1.0.0.0.0.67 1.3.6.1.2.1.7.5.1.1.0.0.0.0.161 1.3.6.1.2.1.7.5.1.1.0.0.0.0.520	UdpLocalAddress.0.0.0.0.67 UdpLocalAddress.0.0.0.0.161 UdpLocalAddress.0.0.0.0.520	0.0.0.0 0.0.0.0 0.0.0.0
2	1.3.6.1.2.1.7.5.1.2.0.0.0.0. 67 1.3.6.1.2.1.7.5.1.2.0.0.0.0. 161 1.3.6.1.2.1.7.5.1.2.0.0.0.0.	UdpLocalPort.0.0.0. 67 UdpLocalPort.0.0.0. 161 UdpLocalPort.0.0.0. 520	67 161 520

	520		

在表格中，一个给定变量的所有实例都在下个变量的所有实例之前显示。这意味表格的操作顺序是先行后列的。表格中对行的排序和表格中索引的值有关。

2.2.2system 组

system 组包含 7 个变量，没有表格，分别是：

```
sysDescr      .1.3.6.1.2.1.1.1.0
sysObjectID   .1.3.6.1.2.1.1.2.0
sysUptime     .1.3.6.1.2.1.1.3.0
sysContact    .1.3.6.1.2.1.1.4.0
sysName       .1.3.6.1.2.1.1.5.0
sysLocation   .1.3.6.1.2.1.1.6.0
sysServices
```

2.2.3interface 组

interface 组只定义了一个简单变量，是系统的接口数量。该组还有一个表格变量，有 22 列。

2.2.4at 组

at 组是地址转换组，在该组中仅有一个由 3 列组成的表格变量。

2.2.5ip 组

ip 组定义了很多简单变量和 3 个表格变量（地址表、路由表、地址转换表）。

2.2.6icmp 组

icmp 组包含 4 个普通计数器变量（ICMP 报文的输出和输入数量以及 ICMP 差错报文的输入和输出数量）和 22 个其他 ICMP 报文数量的计数器，11 个输出计数器，11 个输入计数器。

2.2.7tcp 组

tcp 组包含 14 个简单变量，主要为 TCP 状态。还包含 1 个表格变量，即 TCP 连接表。

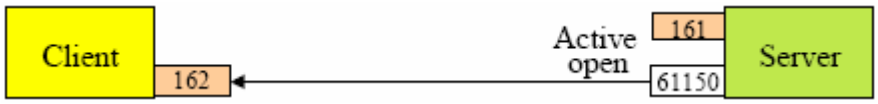
2.3 SNMP

2.3.1端口

Passive open by both client and server:

Exchange of request and response messages:

Sending trap messages by the server:

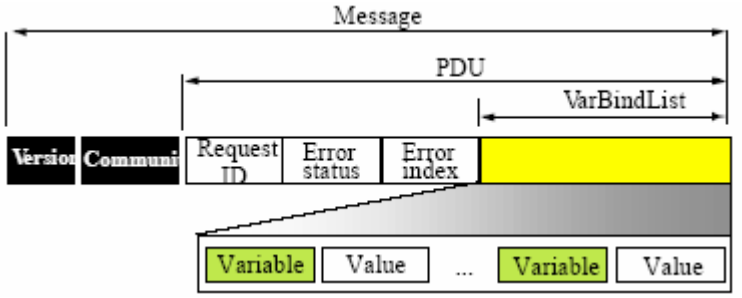


2.3.2结构

```
PDUs::= CHOICE {
get-request [0]        IMPLICIT PDU,
get-next-request [1]   IMPLICIT PDU,
get-response [2]          IMPLICIT PDU,
set-request [3]        IMPLICIT PDU,
trap [4]               IMPLICIT Trap-PDU
}
```

GetRequest, GetNextRequest, SetRequest

PDU type	request-id	0	0	variable-bindings



GetResponse

PDU type	request-id	error-status	error-index	variable-bindings

variable-bindings

name	value	name	value	. . .	name	value

```
VarBindList ::= SEQUENCE OF VarBind

VarBind ::= SEQUENCE {

name ObjectName,

value ObjectSyntax

}

ObjectName ::= OBJECT IDENTIFIER

ObjectSyntax ::= CHOICE {

simple SimpleSyntax,

application-wide ApplicationSyntax

}

SimpleSyntax ::= CHOICE {

number INTEGER,

string OCTET STRING,

object OBJECT IDENTIFIER,
```

empty NULL

}

ApplicationSyntax::= CHOICE {

address NetworkAddress,

counter Counter,

gauge Gauge,

ticks TimeTicks,

arbitrary Opaque

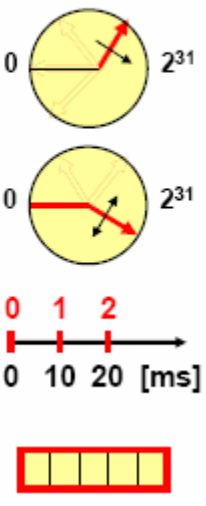
}

NetworkAddress::= CHOICE {

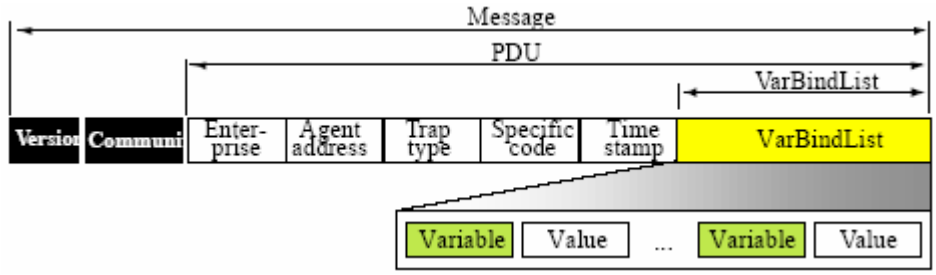
internet IpAddress

}

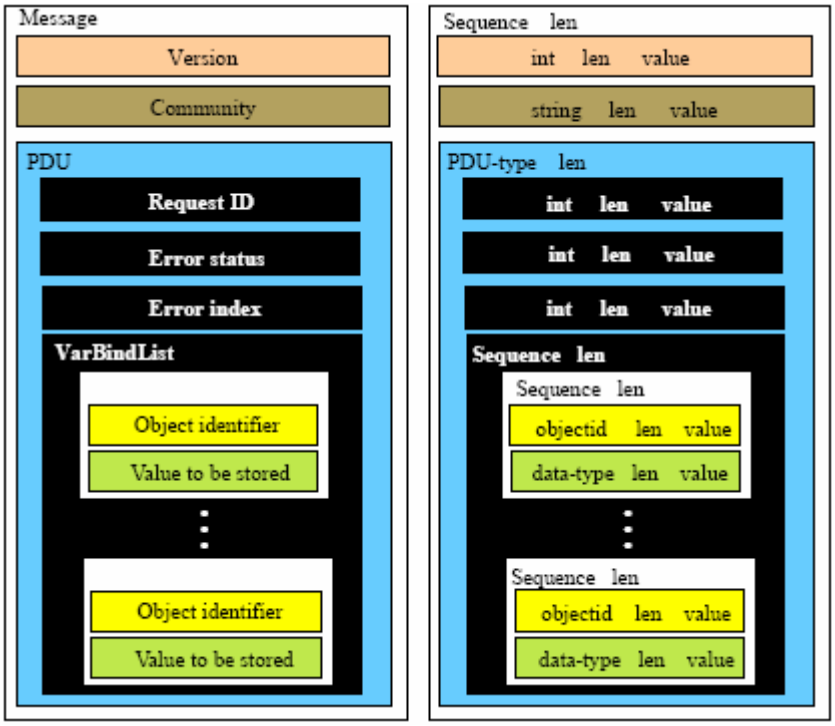
```
IpAddress ::= [APPLICATION 0] IMPLICIT
                 OCTET STRING (SIZE (4))

Counter   ::= [APPLICATION 1] IMPLICIT
                 INTEGER (0..4294967295)

Gauge     ::= [APPLICATION 2] IMPLICIT
                 INTEGER (0..4294967295)

TimeTicks ::= [APPLICATION 3] IMPLICIT
                 INTEGER (0..4294967295)

Opaque    ::= [APPLICATION 4] IMPLICIT
                 OCTET STRING
```



Message format for trap:



Encoding SNMP message using BER



Field	Description
version	SNMP version; RFC 1157 is version 1.
community	A pairing of an SNMP agent with some arbitrary set of SNMP application entities. The name of the community functions as a password to authenticate the SNMP message.
request-id	Used to distinguish among outstanding requests by providing each request with a unique ID.
error-status	Used to indicate that an exception occurred while processing a request. Values are: noError (0), tooBig (1), noSuchName (2), badValue (3), readOnly (4), genErr (5)
error-index	When error-status is nonzero, error-index may provide additional information by indicating which variable in a list caused the exception. A variable is an instance of a managed object.
variable-bindings	A list of variable names and corresponding values. In some cases (e.g., GetRequest-PDU, the values are null.

差错状态字段是一个整数，由代理进程设置，指明有错误发生

差错状态	名称	描述
0	NoError	没有错误
1	TooBig	代理进程无法把响应放在一个 SNMP 消息中发送
2	NoSuchName	操作一个不存在的变量
3	BadValue	Set 操作的值或语法有错误
4	ReadOnly	管理进程试图改变一个只读变量
5	genErr	其他错误

2.3.2.1 PDU

PDU 为协议数据单元，即分组。

```
PDU ::= SEQUENCE {
    request-id      INTEGER,
    error-status    INTEGER {
                        noError(0),
```

```
                    tooBig(1),
                    noSuchName(2),
                    badValue(3),
                    readOnly(4)
                    genErr(5)},
    error-index     INTEGER,
    variable-bindings
                SEQUENCE OF {
                    name    ObjectName,
                    value   ObjectSyntax
                }
}
```

PDU 类型	名称
0	Get-request
1	Get-next-request
2	Get-response
3	Set-request
4	Trap

2.3.2.2 Trap-PDU

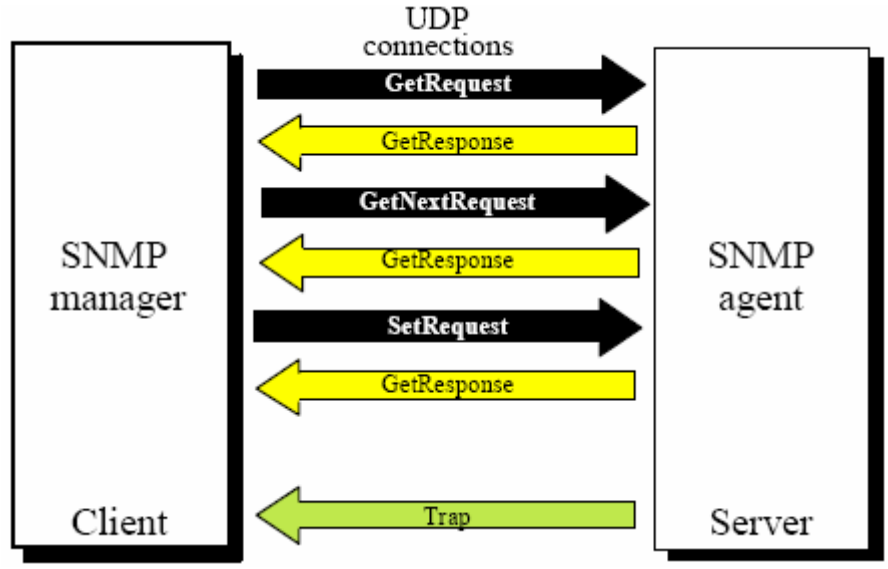
PDU type	enterprise	agent-addr	generic-trap	specific-trap	time-stamp	variable-bindings

```
specific-trap INTEGER,
time-stamp TimeTicks,
variable-bindings VarBindList
} -- enterprise OID equals sysObjectID
Trap-PDU ::= [4]
    IMPLICIT SEQUENCE {
        enterprise      OBJECT IDENTIFIER,
        agent-addr      NetworkAddress,
        generic-trap    INTEGER {
                         coldStart(0),
                         warmStart(1),
                         linkDown(2),
                         linkUp(3),
```

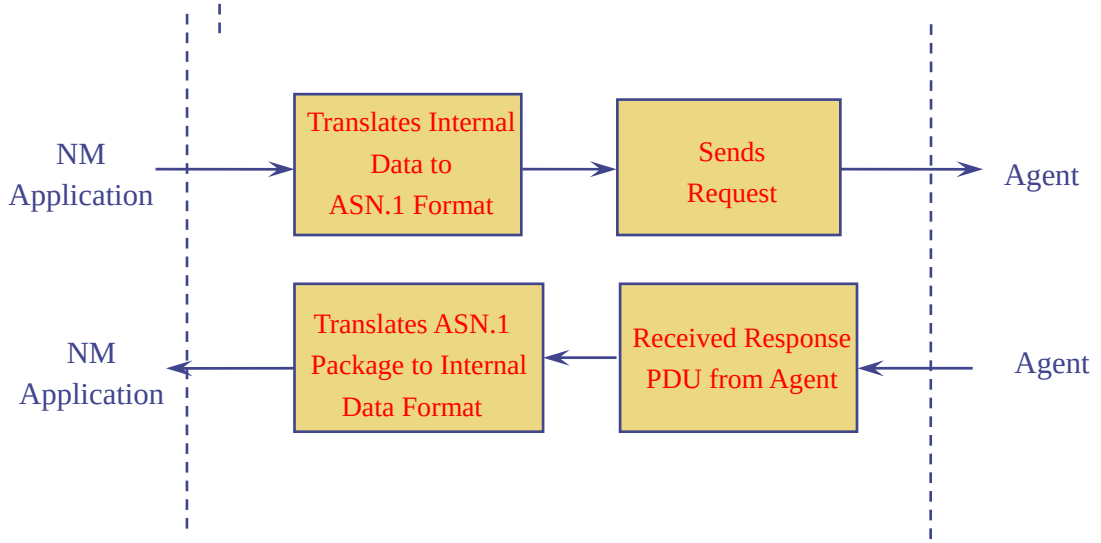
```
                    authenticationFailure(4),
                    egpNeighborLoss(5),
                    enterpriseSpecific(6)},
        specific-trap    INTEGER,
        time-stamp       TimeTicks,
        variable-bindings        VarBindList
            }
```

Enterprise: Type of Object generating trap.
Agent Address: Address of object generating trap.
Generic Trap: Generic trap type.
Specific Trap: Enterprise specific trap.
Time Stamp: Time elapsed between the last initialization of the network entity and the generation of the trap.
Variable Bindings: "Interesting" information

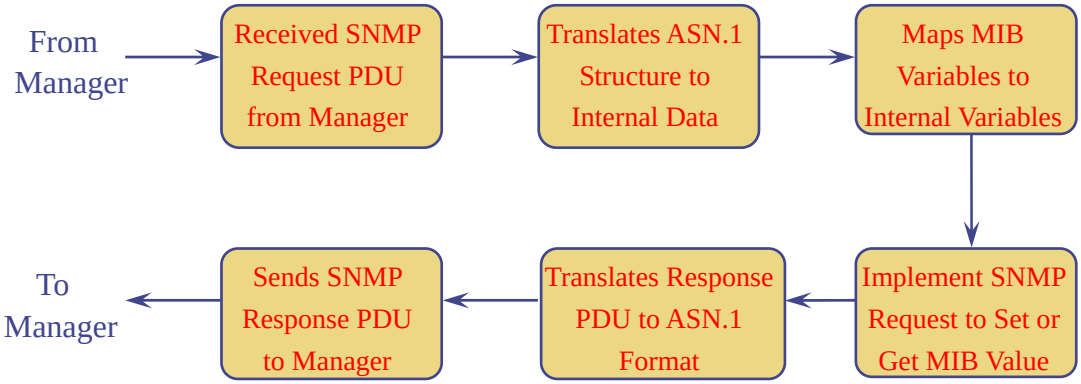
2.3.3过程



2.3.3.1 Manager



2.3.3.2 Agent



Main Loop of Agent:

1. Agent waits for an incoming datagram in Port 161
2. Reads the datagram from UDP and notes the transport address of the sending entity.
3. Increments the QUANTUM to keep track of the logical **request-id** being processed by agent
4. De-serializes the datagram into an ASN.1 structure. If error occurs, log error and discard packet.
5. The ASN.1 structure is translated into SNMP message. If error occurs, log error and discard packet.
6. Check on **VERSION**-NUMBER field. If error occurs, log error and

discard packet.

7. **Community** name is looked up. If community is unknown to agent, agent send AUTHENTICATION trap to Manager station in Port 162; log error and discard packet.
8. Agent loops through list of **variables** in the request. If no prototype is found, return a GET-RESPONE with error noSuchName and discard package. Once prototype is fund, operation is checked against community profile. If mismatch occurs, return get-respone with error noSuchName or readOnly and discard package.Otherwise, agent invokes access routine to perform the desired operation.

3 应用

3.1 snmpget

命令：

```
snmpget [options] node variable [...]
       query a node using SNMP Get request
```

例：

```
>>snmpget -d 10.144.18.118 .1.3.6.1.2.1.1.1.0
Transmitted 41 bytes to 10.144.18.118 port 161:
Initial Timeout: 0.80 seconds
     0:  30 27 02 01 00 04 06 70 75 62 6c 69 63 a0 1a 02     0'.....public...
    16:  02 18 bc 02 01 00 02 01 00 30 0e 30 0c 06 08 2b     .........0.0...+
    32:  06 01 02 01 01 01 00 05 00 -- -- -- -- -- -- --       .............
     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:   GET-REQUEST-PDU (0xa0): 26 bytes
    15:         INTEGER REQUEST-ID (0x2) 2 bytes: 6332
    19:         INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:         INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:         SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:         SEQUENCE VARBIND (0x30): 12 bytes
    29:                 OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    39:                 NULL (0x5) 0 bytes

Received 69 bytes from 10.144.18.118 port 161:
     0:  30 43 02 01 00 04 06 70 75 62 6c 69 63 a2 36 02     0C.....public.6.
    16:  02 18 bc 02 01 00 02 01 00 30 2a 30 28 06 08 2b     .........0*0(..+
    32:  06 01 02 01 01 01 00 04 1c 53 75 6e 20 53 4e 4d     .........Sun SNM
```

```
    48:  50 20 41 67 65 6e 74 2c 20 53 55 4e 57 2c 55 6c     P Agent, SUNW,Ul
    64:  74 72 61 2d 31 -- -- -- -- -- -- -- -- -- -- --       tra-1.....
     0:  SNMP MESSAGE (0x30): 67 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    RESPONSE-PDU (0xa2): 54 bytes
    15:         INTEGER REQUEST-ID (0x2) 2 bytes: 6332
    19:         INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:         INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:         SEQUENCE VARBIND-LIST (0x30): 42 bytes
    27:         SEQUENCE VARBIND (0x30): 40 bytes
    29:             OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    39:             OCTET-STR (0x4) 28 bytes: "Sun SNMP Agent, SUNW,Ultra-1"
system.sysDescr.0 : DISPLAY STRING- (ascii):  Sun SNMP Agent, SUNW,Ultra-1
```

3.2 snmptrap

命令：

```
snmptrap [-d] [-p port] [-c community] node enterprise agent-addr  generic-trap
specific-trap time-stamp variable type value [variable type value...]
        issue an SNMP Version 1 Trap
options:[-d] [-t timeout] [-r retries] [-p port] [-c community] [-v version]
```

例：

```
    >>snmptrap -d manager .1.3.6.1.4.1.612.1.1 10.144.18.116 6 99999 0 .1.3.6.1.1
    octetstringascii "Trap test"
Transmitted 64 bytes to manager (10.144.18.100) port 162:
     0:  30 3e 02 01 00 04 06 70 75 62 6c 69 63 a4 31 06     0>.....public.1.
    16:  09 2b 06 01 04 01 84 64 01 01 40 04 0a 90 12 74     .+.....d..@....t
    32:  02 01 06 02 03 01 86 9f 43 01 00 30 13 30 11 06     ........C..0.0..
    48:  04 2b 06 01 01 04 09 54 72 61 70 20 74 65 73 74     .+.....Trap test
     0:  SNMP MESSAGE (0x30): 62 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    V1-TRAP-PDU (0xa4): 49 bytes
    15:         OBJ-ID ENTERPRISE (0x6) 9 bytes: .1.3.6.1.4.1.612.1.1
    26:         IPADDRESS AGENT-ADDR (0x40) 4 bytes: 10.144.18.116 (manager2)
    32:         INTEGER GENERIC-TRAP (0x2) 1 bytes: 6
    35:         INTEGER SPECIFIC-TRAP (0x2) 3 bytes: 99999
    40:         TIMETICKS TIME-STAMP (0x43) 1 bytes: 0 (0x0)
    43:         SEQUENCE VARBIND-LIST (0x30): 19 bytes
    45:             SEQUENCE VARBIND (0x30): 17 bytes
    47:                 OBJ-ID (0x6) 4 bytes: .1.3.6.1.1
```

53: OCTET-STR (0x4) 9 bytes: "Trap test"

4 附录

4.1 其他网络管理

从 1985 年开始，CCITT 致力于开发和制定电信网的网络管理标准，并且在 1988 年－1992 年的研究期间，引进了许多 OSI 管理思想进行了重写，目前已经形成了较为完善的电信网络管理推荐标准，即 TMN 网络管理。由于前面一段的发展经历，实际上现在的 OSI 管理标准与 TMN 建议可以互为补充。

八十年代后，Internet 发展迅猛，IETF 采用了基于 OSI 的 CMIP 协议作为 Internet 的管理协议，并对它作了修改，修改后的协议被称作 CMOT(Common Management Over TCP/IP)。但 CMOT 迟迟未能出台，IETF 决定把已有的 SGMP(简单网关监控协议)进一步修改后，作为临时的解决方案。这个在 SGMP 基础上开发的解决方案就是著名的简单网络管理协议（SNMP)。由于 SNMP 存在一些不足，IETF 已经提出了改进版本 SNMPv2，SNMPv3，在多个方面进行改进和强化。

RMON
TMN
CMIP

4.2 SNMP 版本比较

SNMP v1 基本上没有什么安全性可言，在安全方面 SNMP v1 存在以下主要的安全问题：SNMP 数据包的修改：指一个未经验证的用户捕获到 SNMP 数据包后，修改其信息，又把数据包发送到目的站。而接收设备不能得知数据的改变，于是就响应包里的信息，导致安全问题。

SNMP v2 在原有的 Get、GetNext、Set、Trap 等操作外增加了 GetBulk 和 Inform 两个新的协议操作。其中 GetBulk 操作快速获取大块数据。Inform 操作允许一个 NMS 向另一个 NMS 发送 Trap 信息，并接收一个响应消息

SNMP v2 安全标准对数据修改、假冒和数据包顺序改变等安全问题提出了比较满意的解决方案，进一步为安全标准提出了一系列的目标，提出了分级的安全机制以及验证机制和使用 DES 标准加密算法。

SNMP v3 并不是一个自成体系，用以取代 SNMP v1 和 SNMP v2 的协议。SNMP v3 定义了安全方面的扩展能力，用来和 SNMP v1 及 SNMP v2c 相连接。在 SNMP v3 工作组定义的五个 RFC 中，2271 描述了现行的 SNMP 使用的体系结构，2275 描述了一种接入控制的方法，它同 SNMP v3 的核心功能是独立开的，只有 2272—2274 三个 RFC 才是真正有关安全方面的建议。

4.3 参考

Version	RFC	Description
v1	RFC 1155	Structure and Identification of Management Information for TCP/IP-based Internets.
	RFC 1157	Simple Network Management Protocol.
	RFC 1213	MIB II Implementation for Linux and VxWorks.
	RFC 1215	Convention for defining traps for use with the SNMP
v2	RFC 1901	Introduction to Community-based SNMPv2.
	RFC 1907	MIB for SNMPv2.
v3	RFC 2571	SNMP Framework MIB.
	RFC 2572	SNMP Message Processing and Dispatching (MPD).
	RFC 2573	SNMP Target MIB and SNMP Notification MIB.
	RFC 2574	SNMP User-Based Security Model (USM) MIB.
	RFC 2575	SNMP View-Based Access Control Model (VACM) MIB.
	RFC 2576	SNMP Co-existence between Version 1, Version 2, and Version 3 MIB.

网络管理论坛　http://snmp.xiloo.com/index.html
网络管理技术　http://www.im.ncnu.edu.tw/~ycchen/nm/