使用PySNMP获取网络数据

学院：信息工程学院

教师：张迁

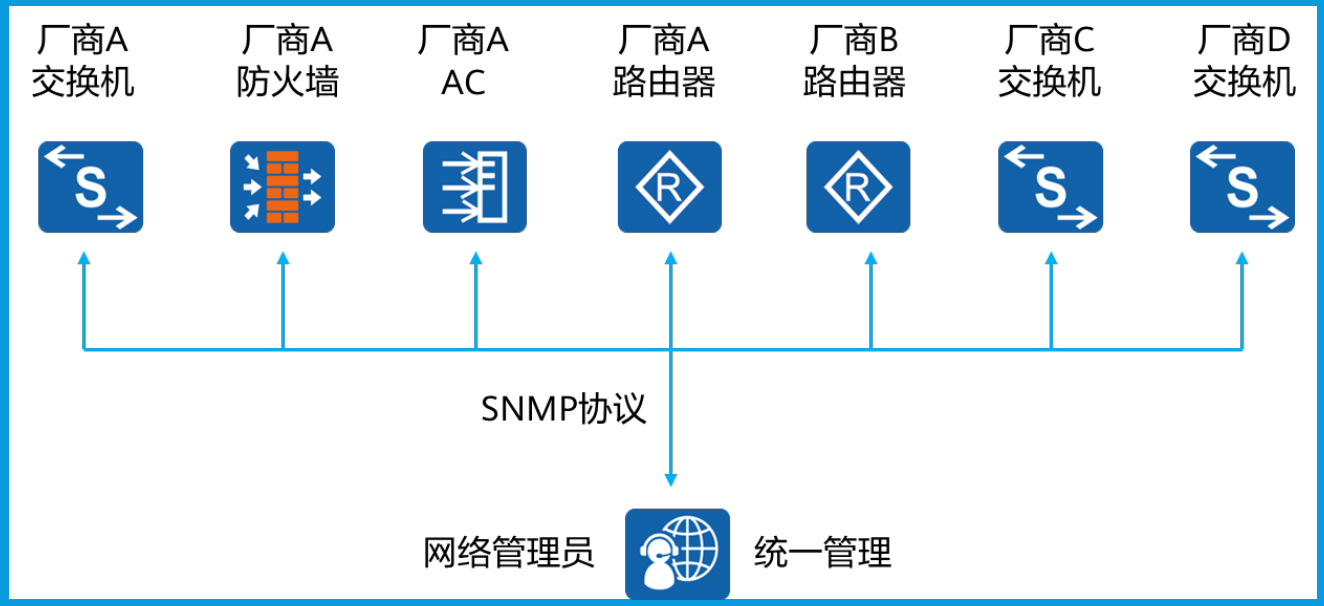
目录

1 SNMP基本概念

1. 随着网络规模越来越庞大，网络中不同种类、不同厂家、不同型号设备共存，网络管理繁琐低效。为解决这个问题，SNMP（Simple Network Management Protocol，简单网络管理协议）应运而生，规范了网络管理的接口和协议，实现对网络中所有设备的统一管理。
2. 为应对网络中复杂多变的运维需求，工程师有必要具备代码编程能力以提高运维效率。Python中的PySNMP模块可以实现SNMP功能，是工程师提升运维能力的有力工具。

1 SNMP基本概念

- 互联网工程任务组（IETF，Internet Engineering Task Force）定义了SNMP（简单网络管理协议），以此实现设备统一管理。
- 所有支持SNMP协议的网络设备，都可将其统一纳入管理。



1 SNMP基本概念

- SNMP的发展经历了SNMPv1，SNMPv2c，SNMPv3，这是一个不断完善改进的过程。

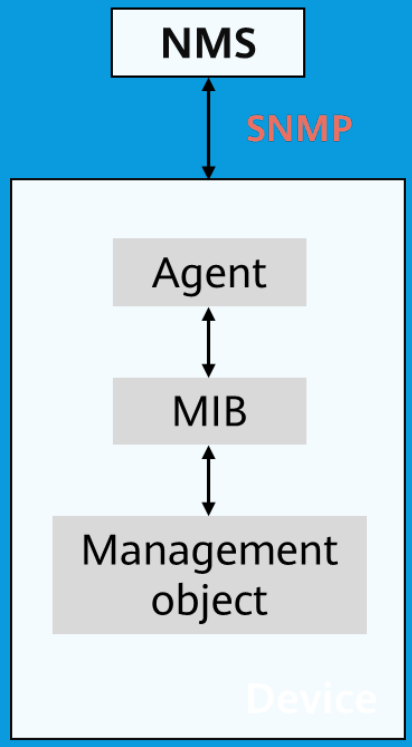
- SNMPv1协议SNMP的最初版本，容易实现且成本低。
- 缺少大量读取数据的能力，没有足够的安全机制。
- 适合规模较小，设备较少，安全性要求不高或本身就比较安全的网络，如校园网，小型企业网。

- SNMPv2扩充了SNMPv1的功能，增加GetBulk和inform操作。
- 没有足够的安全机制。
- 适合规模较大，设备较多，安全性要求不高或本身就比较安全，但业务比较繁忙，有可能发生流量拥塞的网络。

- SNMPv3增加了身份验证和加密处理。
- 新的SNMP体系结构，适应性强、方便管理、扩展性好。
- 适合各种规模，尤其是对安全性要求较高，只有合法的管理员才能对设备进行管理的网络。

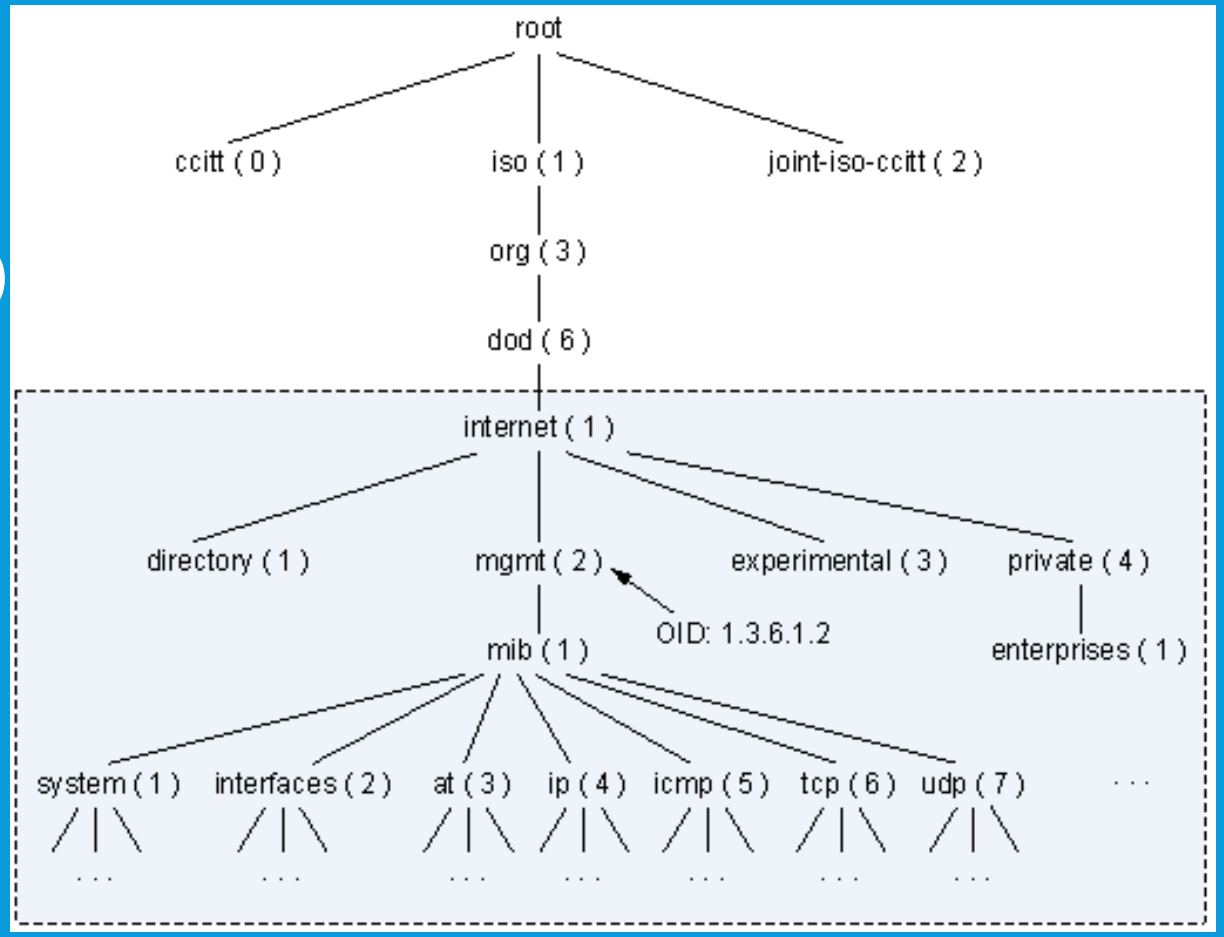
1 SNMP基本概念

- NMS（Network Management System）：采用SNMP协议对网络设备进行管理的系统。
- Agent：被管理设备中的一个代理进程，与NMS进行交互。
- MIB（Management information base）：被管理设备所维护的变量（能够被Agent查询和设置的信息）构成的数据库。
- Management object：被管理对象，如设备中的某个硬件（如接口板），也可以是在硬件或软件上配置的参数集合。
- Device：被管理设备。



1 SNMP基本概念

- MIB是一个数据库，指明了被管理设备所维护的变量（即能够被Agent查询和设置的信息）。MIB在数据库中定义了被管理设备的一系列属性：
  - 对象标识符（Object IDentifier，OID）
  - 对象的状态
  - 对象的访问权限
  - 对象的数据类型等
- MIB给出了一个数据结构，包含了网络中所有可能的被管理对象的集合。因为数据结构与树相似，MIB又被称为对象命名树。



1 SNMP基本概念

- NMS通过SNMPv3向被管理设备下发查询和设置操作指令，并接收操作响应信息，同时监听被管理设备发送的告警信息。

功能	SNMPv3操作类型	描述
查询	Get	从Agent中提取一个或多个参数值。
	GetNext	从Agent中按照字典顺序提取下一个参数值。
	GetBulk	对Agent进行信息群查询。
设置	Set	通过Agent设置一个或多个参数值。
告警	Trap	Agent主动向NMS发出信息，告知被管理设备出现的情况。
	Inform	作用与Trap相同，但需要NMS进行接收确认，会占用较多系统资源。
响应消息	Response	Agent对Get/Set操作的响应消息，NMS对Inform的响应消息。

1 SNMP基本概念

- SNMP在执行Get、Set等基本操作时面临安全威胁，SNMP v3版本较v1，v2c在安全性方面做了提升。SNMPv1\v2c使用团体名进行安全认证。存在以下安全风险：大多数网络产品出厂时设定只读团体名缺省值为“Public”，读写操作团体名缺省值为“Private”。许多网络管理人员从未修改过

基本概念：

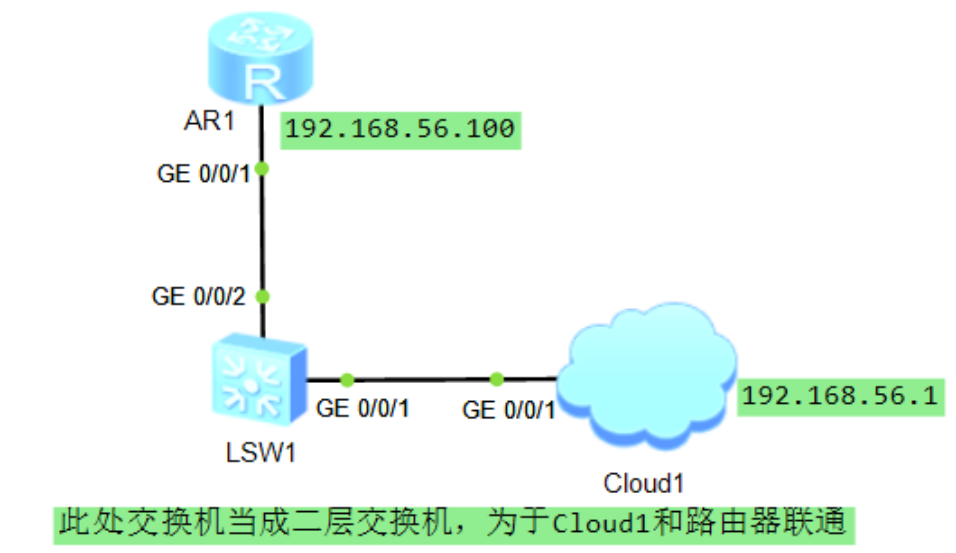
- 用户组（Group）：拥有特定安全级别属性的一个用户集合。安全级别：
  1级：privacy（鉴权且加密）
  2级：authentication（只鉴权）
  3级：noauthentication（不鉴权不加密）
- 视图（View）：允许用户访问的mib节点集合。

- SNMPv3安全性增强：

- 用户安全模块USM（User-based Security Model）：
  身份验证：Agent或NMS接到信息时首先必须确认信息是否来自有权限的NMS或Agent，并且信息在传输过程中未被改变。
  数据加密：通过对称密钥系统，NMS和Agent共享同一密钥对数据进行加密和解密。
- 基于视图的访问控制模块VACM：对用户组实现基于视图的访问控制

1 SNMP基本概念

如下拓扑，配置路由器AR1



接口配置

```
<Huawei>system-view
Enter system view, return user view with Ctrl+Z.
[Huawei]sysname R1
[R1]int go/o/1
[R1-GigabitEtherneto/o/1]  ip address 192.168.56.100 24
[R1-GigabitEtherneto/o/1]  quit
```

1 SNMP基本概念

配置路由器SNMP

```
# 使能SNMP功能，配置版本为v3
[R1]snmp-agent
[R1]snmp-agent sys-info version v3

# 配置SNMPv3组名为test，加密认证方式为privacy
[R1]snmp-agent group v3 test privacy

# 创建SNMPv3用户，名为R1同时配置认证和加密密码为Huawei12#$
[R1]snmp-agent usm-user v3 R1 test authentication-mode md5 Huawei12#$ privacy-mode aes128
Huawei12#$

# 创建名为param的Trap参数信息，securityname为sec
[R1]snmp-agent target-host trap-paramsname param v3 securityname sec privacy

# 设置SNMP告警主机地址为192.168.56.100
[R1]snmp-agent target-host trap-hostname nms address 192.168.56.100 trap-paramsname param

#打开告警开关，设置发送告警的源接口为GE0/0/1，最后的命令是打开设备的所有告警开关
[R1]snmp-agent trap source GigabitEthernet 0/0/1
[R1]snmp-agent trap enable
```

2. PySNMP模块基本概念

- PySNMP是python的第三方模块，实现了SNMP v1/v2c/v3的所有功能，最新版本为v4.4.12。使用者可以使用python语言，利用该模块实现SNMP的所有操作。
- PySNMP提供了简单易用的高层封装接口，用来简化工程师的编码过程，提高其效率。
- PySNMP的主要功能：

SNMP代理支持：允许开发人员编写自己的SNMP代理应用程序。

SNMP管理器支持：可以用来查询网络设备的状态信息，设置设备配置，或者接收来自设备的陷阱（trap）消息。

MIB解析：支持MIB（Management Information Base）文件解析，使得开发者可以使用MIB定义的对象来查询或设置设备状态。

命令生成器：提供了命令生成器函数，如getCmd(), nextCmd(), setCmd()等，用于发送SNMP请求并处理响应。

2. PySNMP模块基本概念

右图解释了PySNMP的功能结构：

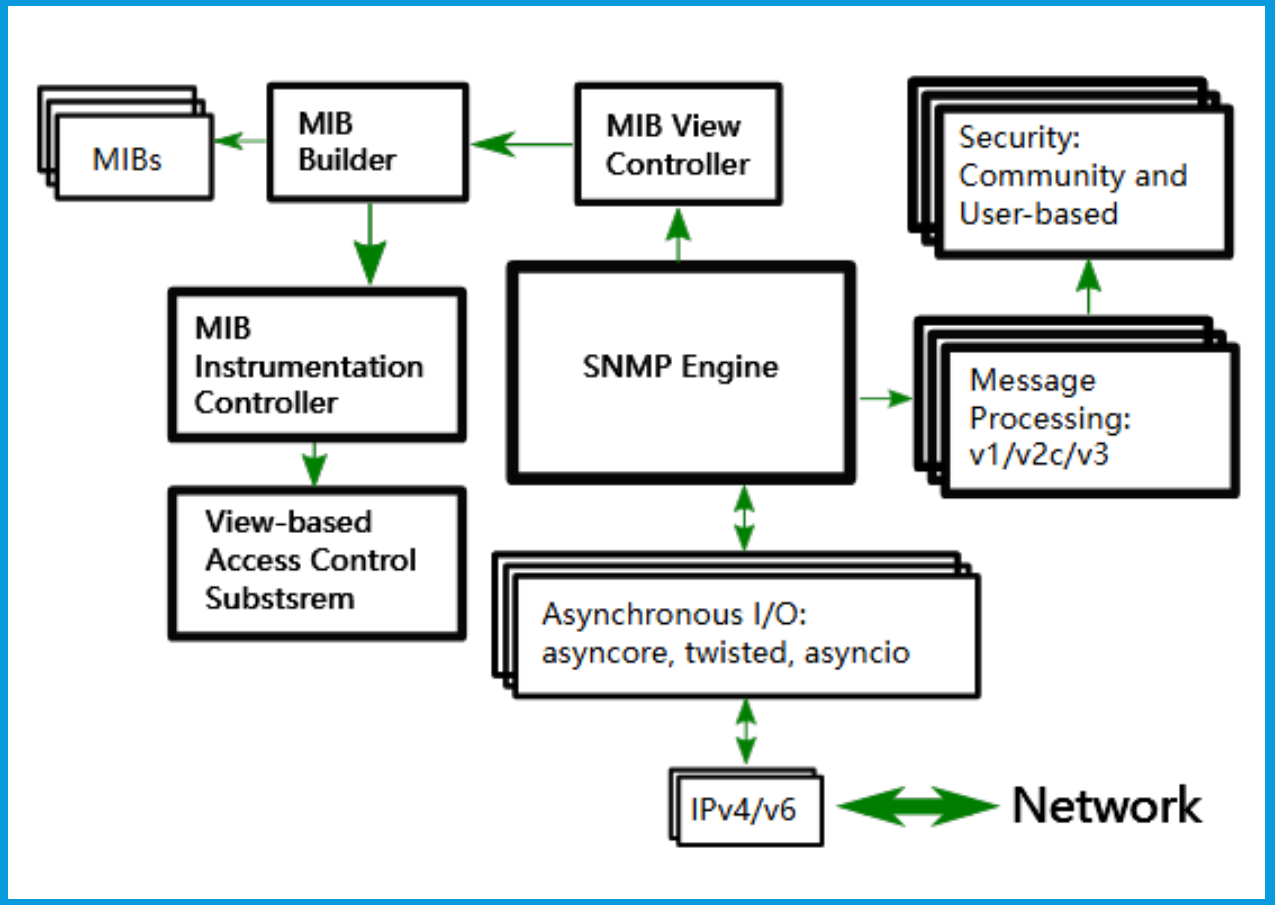
1、SNMP引擎是核心，是保护伞，它控制SNMP系统其它组件；

2、 I/O子系统用来传输或接收SNMP消息，其由一个抽象的分发器(Dispatcher)和一个(或多个)抽象Transport类组成

3、Message and PDU Dispatcher是SNMP消息处理活动的地方，它的主要任务包括：把SNMP应用从不同子系统收集的PDU向下传输给Transport Dispatcher，并把来自于网络的SNMP消息向上传输到SNMP应用；

4、消息处理模块为当前和未来可能版本的SNMP协议处理消息层级的协议操作；

5、消息安全模块处理消息认证和加密，访问控制系统使用LCD(Local Configuration Datastore)来认证对被管理对象的远程访问。而一系列MIB模块和对象集合被称之为LCD。



2. PySNMP模 块安 装

- PIP安装PySNMP：

```
pip install pysnmp
pip install pysnmp-mibs
```

- 查看PySNMP版本，当前版本为4.4.12

```
C:\Users\Administrator>python
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pysnmp
>>> pysnmp.__version__
'4.4.12'
```

实训6-（一）：PySNMP模块的安装和验证

【任务目标】掌握Python的PySNMP模块的安装和导入方法。

（一）安装 pysnmp 并验证版本（请按要求填写命令，粘贴结果图）

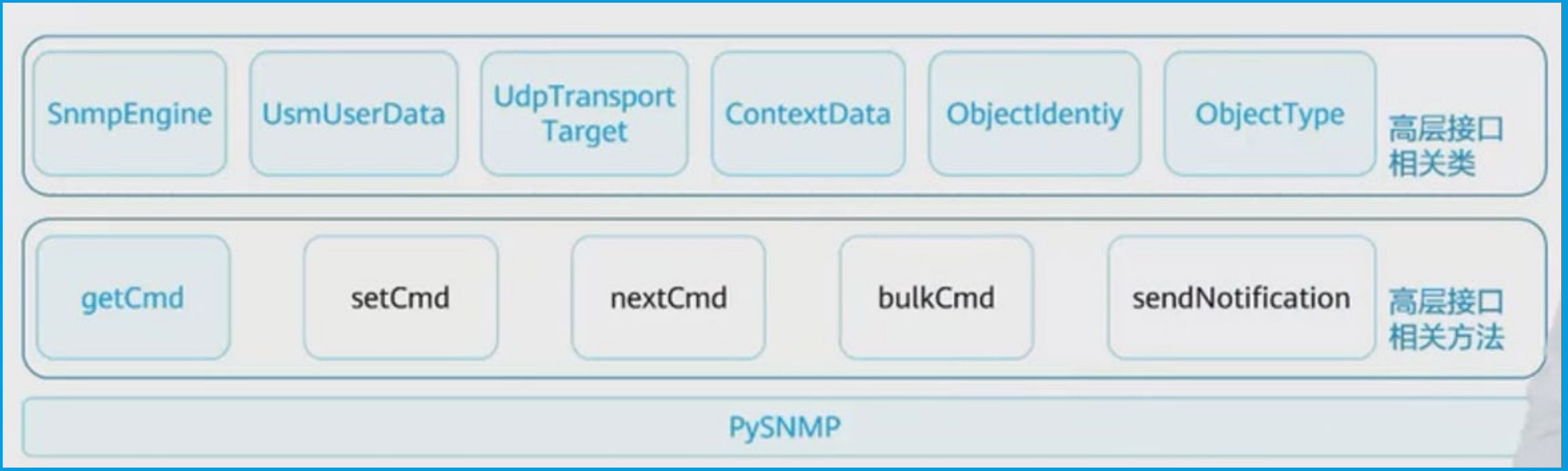
1. 通过 Anaconda Prompt 在虚拟环境 ensp_py 下安装 pysnmp 及 pysnmp-mibs 包，并验证其版本信息。

执行命令截图：

验证结果图：

3 PySNMP模块的6个高层接口类

- PySNMP高层接口实现SNMPv3基本操作涉及的类有SnmpEngine类，UsmUserData类，UdpTransportTarget类， ContextData类， ObjectIdentity类， ObjectType类；
- 涉及的方法有getCmd，setCmd，nextCmd，bulkCmd，sendNotification。



3 PySNMP模块的6个高层接口类

- SnmpEngine类：PySNMP模块中的一个核心对象，PySNMP实现所有SNMP操作都涉及SnmpEngine类实例。创建SNMP引擎对象后，可以使用它来发送SNMP消息，发送消息需要指定目标设备的IP地址及、SNMP版本及其他操作等。创建SNMP类实例的使用方法：

```
engine = SnmpEngine()
```

- UsmUserData类：PySNMP模块对SNMPv3用户安全模块USM的实现。可以利用该类创建SNMPv3用户及其对应的认证与加密算法。使用方法：

```
userData= UsmUserData(
                        'admin',                                  # 用户名
                        authKey = 'Admin@123',                    # 认证密钥
                        privKey = 'Huawei@123',                   # 加密密钥
                        authProtocol = usmHMACSHAAuthProtocol,    # 认证算法
                        privProtocol = usmAesCfb128Protocol       # 加密算法
                        )
```

3 PySNMP模 块的6个 高层 接口 类

PySNMP支持的认证算法和加密算法：

认证算法	加密算法
usmNoAuthProtocol usmHMACMDAuthProtocol usmHMACSHAuthProtocol usmHMAC128SHA224AuthProtocol usmHMAC192SHA256AuthProtocol usmHMAC256SHA384AuthProtocol usmHMAC384SHA512AuthProtocol	usmNoPrivProtocol usmDESPrivProtocol usm3DESEDEPrivProtocol usm3AesCfb128PrivProtocol usm3AesCfb192PrivProtocol usm3AesCfb256PrivProtocol

3 PySNMP模块的6个高层接口类

- UdpTransportTarget类：包含被管理设备IP地址和端口号的类。使用方法：

```
target = UdpTransportTarget((“192.168.56.100”, 161))    # (host, port)是包含被管理设备IP和Port
                                                          # 的元组
```

- ContextData类：表示SNMP上下文信息的类。使用方法：

```
context = ContextData()                    # 初始化类实例时参数为空则为‘empty’上下文对象
```

- ObjectIdentity类：表示MIB节点OID的类。使用方法：

```
oid1 = ObjectIdentity(‘SNMPv2-MIB’,‘sysName’,0)    # sysName节点对象实例
oid2 = ObjectIdentity(‘1.3.6.1.2.1.1.5.0’)          # 使用sysName节点OID字符串进行初始化
```

- ObjectType类：表示MIB节点的类，使用ObjectIdentity对象进行初始化。使用方法：

```
obj1 = ObjectType(ObjectIdentity(‘SNMPv2-MIB’,‘sysName’,0))  #实例化sysName的ObjectType对象
```

4. PySNMP模块的5个接口函数

- getCmd方法：实现SNMP Get操作类型的方法，返回值是一个生成器。方法声明：

  getCmd(snmpEngine, authData, transportTarget, contextData, *varBinds)。

参数	说明
snmpEngine	SnmpEngine类实例
authData	UsmUserData类实例
transportTarget	UdpTransportTarget类实例
ContextData	ContextData类实例
*varBinds	ObjectType类实例

- 使用方法：

```
g = getCmd(snmpEngine, authData, transportTarget, contextData, *varBinds)  # g是一个生成器
# 使用next方法，会产生一个get操作，获取的结果保存在varBinds中，其余三个返回值指示出错的情况
errorIndication, errorStatus, errorIndex, varBinds =next(g)
```

4. PySNMP模块的5个接口函数

- bulkCmd方法：用于实现SNMP getbulk()操作并获取批量MIB值，它是对OID值的遍历，返回值是一个生成器。
- bulkCmd()适用于节点下还有叶子节点或者一个叶子节点有许多值的情况，如查询设备的接口，设备有多个接口，每个接口都有相关信息，使用父节点ifEntry查询涉笔所有接口信息。
- 方法声明如下：

bulkCmd(snmpEngine, authData, transportTarget, contextData, nonRepreaters, maxRepetitions, *varBinds)。

其中参数nonRepeaters和maxRepetitions用于控制MIB对象的批处理个数。例如将前者设为0，后者设为50，则每个SNMP响应最多可以包含50个变量绑定(varBinds)。

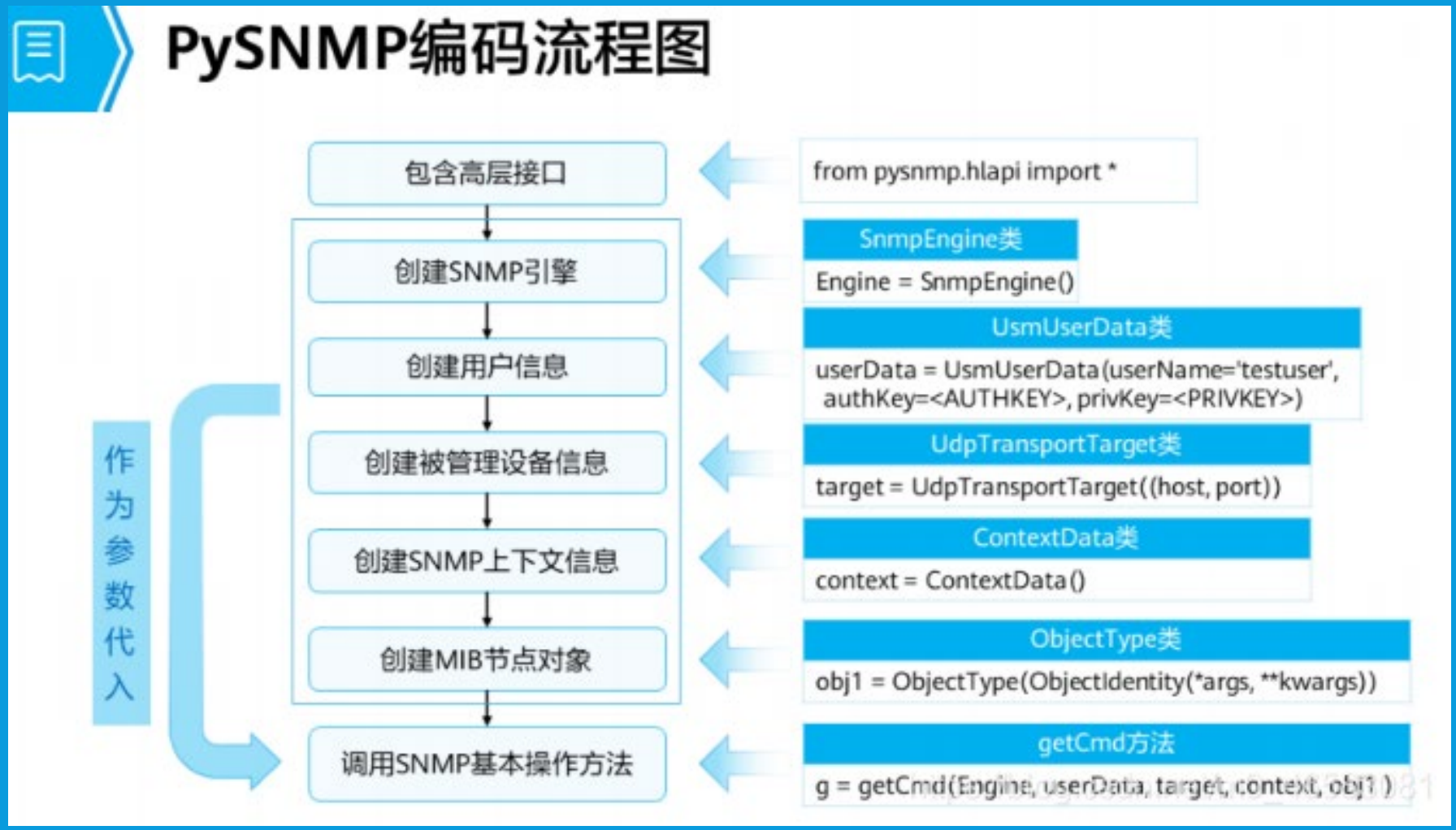
4. PySNMP模块的5个接口函数

- setCmd方法：用于实现SNMP set操作，返回值是一个生成器。方法声明如下：
  setCmd(snmpEngine, authData, transportTarget, contextData, *varBinds)。
- SNMP允许你请求一个MIB对象，通过next获取下一个值。通过这种方式，你可以提前读取你不知道的MIB对象。MIB对象根据它们的OID进行概念排序。这个功能由 nextCmd() 函数实现：
  nextCmd(snmpEngine, authData, transportTarget, contextData, *varBinds)。
- sendNotification 函数用于发送 SNMP 通知（通常是 TRAP 或者 INFORM），这通常用于当网络设备发生重要事件时向网络管理系统（NMS）报告。
  sendNotification(snmpEngine, CommunityData, transportTarget, contextData, NotificationType, *varBinds)。

其中CommunityData()：定义发送方的身份认证信息，如社区字符串。NotificationType()：定义要发送的通知类型。

4. PySNMP模块的5个接口函数

PySNMP模块编码流程



5.实训6-（二）：使用PySNMP获取网络数据

【任务目标】参考实验指导说明书，基于指导教师给的网络拓扑图Ensp文件，通过PySNMP获取路由器SZ1和SZ2数据，包括每台路由器的sysname、接口数目、接口类型、接口IP地址和掩码、路由目标、路由下一跳。需要完成的任务如下。

（1）配置SNMPv3。

（2）通过MIB管理工具获取OID。

（3）编写Python脚本。

（4）运行Python脚本。