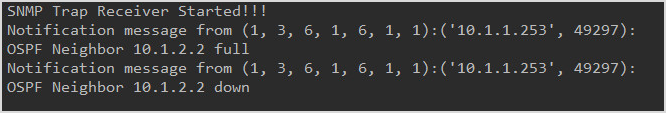
**经典网络协议 第十天作业**

1. **使用Python监控SNMP Trap(v2/v3均可)!监控OSPF邻居状态!**

**最终结果如下图, Python对OSPF邻居状态改变的SNMP Trap进行分析,并打印!**



**提示一: 路由器SNMP Trap配置**

snmp-server enable traps ospf state-change  
snmp-server enable traps ospf errors

**提示二: OSPF SNMP Trap Oid**

[整体OID摘要](http://cric.grenoble.cnrs.fr/Administrateurs/Outils/MIBS/?oid=1.3.6.1.2.1.14.10.1)

https://cric.grenoble.cnrs.fr/Administrateurs/Outils/MIBS/?oid=1.3.6.1.2.1.14.10.1  
[邻居状态改变](http://cric.grenoble.cnrs.fr/Administrateurs/Outils/MIBS/?oid=1.3.6.1.2.1.14.10.1.6)

https://cric.grenoble.cnrs.fr/Administrateurs/Outils/MIBS/?oid=1.3.6.1.2.1.14.10.1.6  
[邻居IP地址](http://cric.grenoble.cnrs.fr/Administrateurs/Outils/MIBS/?oid=1.3.6.1.2.1.14.10.1.1)

https://cric.grenoble.cnrs.fr/Administrateurs/Outils/MIBS/?oid=1.3.6.1.2.1.14.10.1.1  
[邻居Router ID](http://cric.grenoble.cnrs.fr/Administrateurs/Outils/MIBS/?oid=1.3.6.1.2.1.14.10.1.3)

https://cric.grenoble.cnrs.fr/Administrateurs/Outils/MIBS/?oid=1.3.6.1.2.1.14.10.1.3

代码：

from get\_ip\_netifaces import get\_ip\_address  
from pysnmp.carrier.asynsock.dispatch import AsynsockDispatcher  
from pysnmp.carrier.asynsock.dgram import udp, udp6  
from pyasn1.codec.ber import decoder  
from pysnmp.proto import api  
from pprint import pprint  
  
  
def analysis(info):  
 # 分析Trap信息字典函数  
 # 下面是这个大字典的键值与嵌套的小字典  
 # 1.3.6.1.2.1.1.3.0 {'value': 'ObjectSyntax', 'application-wide': 'ApplicationSyntax', 'timeticks-value': '103170310'}  
 # 1.3.6.1.6.3.1.1.4.1.0 {'value': 'ObjectSyntax', 'simple': 'SimpleSyntax', 'objectID-value': '1.3.6.1.6.3.1.1.5.4'}  
 # 1.3.6.1.2.1.2.2.1.1.2 {'value': 'ObjectSyntax', 'simple': 'SimpleSyntax', 'integer-value': '2'}  
 # 1.3.6.1.2.1.2.2.1.2.2 {'value': 'ObjectSyntax', 'simple': 'SimpleSyntax', 'string-value': 'GigabitEthernet2'}  
 # 1.3.6.1.2.1.2.2.1.3.2 {'value': 'ObjectSyntax', 'simple': 'SimpleSyntax', 'integer-value': '6'}  
 #  
 if '1.3.6.1.2.1.14.10.1.6' in info.keys():  
 if info["1.3.6.1.2.1.14.10.1.6"]['integer-value'] == '1':  
 print('OSPF Neighbor ' +info["1.3.6.1.2.1.14.1.1"]['ipAddress-value'] + " Down")  
 elif info["1.3.6.1.2.1.14.10.1.6"]['integer-value'] == '8':  
 print('OSPF Neighbor ' +info["1.3.6.1.2.1.14.1.1"]['ipAddress-value']+ " Full")  
 # pprint(info, indent=4) # 打印字典信息  
  
  
def cb\_fun(transport\_dispatcher, transport\_domain, transport\_address, whole\_msg): # 处理Trap信息的函数  
 while whole\_msg:  
 msg\_ver = int(api.decodeMessageVersion(whole\_msg)) # 提取版本信息  
 if msg\_ver in api.protoModules: # 如果版本兼容  
 p\_mod = api.protoModules[msg\_ver]  
 else: # 如果版本不兼容，就打印错误  
 print('Unsupported SNMP version %s' % msg\_ver)  
 return  
 req\_msg, whole\_msg = decoder.decode(  
 whole\_msg, asn1Spec=p\_mod.Message(), # 对信息进行解码  
  
 )  
 print('Notification message from %s:%s: ' % (  
 transport\_domain, transport\_address # 打印发送TRAP的源信息  
 )  
 )  
 req\_pdu = p\_mod.apiMessage.getPDU(req\_msg)  
 if req\_pdu.isSameTypeWith(p\_mod.TrapPDU()):  
 if msg\_ver == api.protoVersion1: # SNMPv1的特殊处理方法,可以提取更加详细的信息  
 print('Enterprise: %s' % (  
 p\_mod.apiTrapPDU.getEnterprise(req\_pdu).prettyPrint()  
 )  
 )  
 print('Agent Address: %s' % (  
 p\_mod.apiTrapPDU.getAgentAddr(req\_pdu).prettyPrint()  
 )  
 )  
 print('Generic Trap: %s' % (  
 p\_mod.apiTrapPDU.getGenericTrap(req\_pdu).prettyPrint()  
 )  
 )  
 print('Specific Trap: %s' % (  
 p\_mod.apiTrapPDU.getSpecificTrap(req\_pdu).prettyPrint()  
 )  
 )  
 print('Uptime: %s' % (  
 p\_mod.apiTrapPDU.getTimeStamp(req\_pdu).prettyPrint()  
 )  
 )  
 var\_binds = p\_mod.apiTrapPDU.getVarBindList(req\_pdu)  
 else: # SNMPv2c的处理方法  
 var\_binds = p\_mod.apiPDU.getVarBindList(req\_pdu)  
  
 result\_dict = {} # 每一个Trap信息,都会整理返回一个字典  
 # 下面是这个大字典的键值与嵌套的小字典  
 # 1.3.6.1.2.1.1.3.0 {'value': 'ObjectSyntax', 'application-wide': 'ApplicationSyntax', 'timeticks-value': '103170310'}  
 # 1.3.6.1.6.3.1.1.4.1.0 {'value': 'ObjectSyntax', 'simple': 'SimpleSyntax', 'objectID-value': '1.3.6.1.6.3.1.1.5.4'}  
 # 1.3.6.1.2.1.2.2.1.1.2 {'value': 'ObjectSyntax', 'simple': 'SimpleSyntax', 'integer-value': '2'}  
 # 1.3.6.1.2.1.2.2.1.2.2 {'value': 'ObjectSyntax', 'simple': 'SimpleSyntax', 'string-value': 'GigabitEthernet2'}  
 # 1.3.6.1.2.1.2.2.1.3.2 {'value': 'ObjectSyntax', 'simple': 'SimpleSyntax', 'integer-value': '6'}  
 for x in var\_binds: # 打印详细Trap信息  
 result = {}  
 for x, y in x.items():  
 # print(x, y.prettyPrint()) # 最原始信息打印  
 # 处理信息到字典  
 if x == "name":  
 id = y.prettyPrint() # 把name写入字典的键  
 else:  
 bind\_v = [x.strip() for x in y.prettyPrint().split(":")]  
 for v in bind\_v:  
 if v == '\_BindValue':  
 continue  
 else:  
 result[v.split('=')[0]] = v.split('=')[1]  
 result\_dict[id] = result  
 # 把字典传到分析模块进行分析  
 analysis(result\_dict)  
  
 return whole\_msg  
  
  
def snmp\_trap\_receiver(ifname, port=162):  
 if\_ip = get\_ip\_address(ifname)  
 transport\_dispatcher = AsynsockDispatcher() # 创建实例  
  
 transport\_dispatcher.registerRecvCbFun(cb\_fun) # 调用处理Trap信息的函数  
  
 # UDP/IPv4  
 transport\_dispatcher.registerTransport(  
 udp.domainName, udp.UdpSocketTransport().openServerMode((if\_ip, port)) # 绑定到本地地址与UDP/162号端口  
 )  
  
 transport\_dispatcher.jobStarted(1) # 开始工作  
 print("SNMP Trap Receiver Started!!!")  
 try:  
 # Dispatcher will never finish as job#1 never reaches zero  
 transport\_dispatcher.runDispatcher() # 运行  
 except Exception:  
 transport\_dispatcher.closeDispatcher()  
 raise  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 # 使用Linux解释器 & WIN解释器  
 snmp\_trap\_receiver("ens33")

运行结果：

