جامعة تشرين

كلية الهندسة الميكانيكية و الكهربائية

مشروع في برمجة و إدارة الشبكات

بإشراف الدكتور مهند عيسى

إعداد :هيا حسن العلي 2300
حسن غسان خيربك 1128
محياة أحمد محلا 1720

بعنوان

SNMP PROTOCOL USING PYTHON CODE

تعريف بلغة البايثون

هي لغة برمجية عالية المستوى سهلة التعلم مفتوحة المصدر قابلة للتوسيع تعتمد أسلوب البرمجة كائنية التوجه (OOP).

وهي لغة مفسرة ومتعددة الاستخدامات تستخدم بشكل واسع في العديد من المجالات كبناء البرامج المستقلة باستخدام الواجهات الرسومية, وفي تطبيق الويب, ويمكن استخدامها كلغة برمجية نصية للتحكم في أداء العديد من البرمجيات ,ويمكن استخدامها لعمل برامج بسيطة للمبتدئين .

مميزات لغة باليثون

1. دعم للبرمجة الوظيفية
2. سهلة التعلم
3. حرة ومفتوحة المصدر
4. لغة برمجة عالية المستوى
5. محمولة
6. كائنية التوجة
7. قابلة للامتداد

ملخص عن المشروع:

- مقدمة عن الحاجة ل بروتوكول SNMP
- تعريف ال SNMP
- استخدماته
- إصدارته
- آلية عمله
- كود بايثون

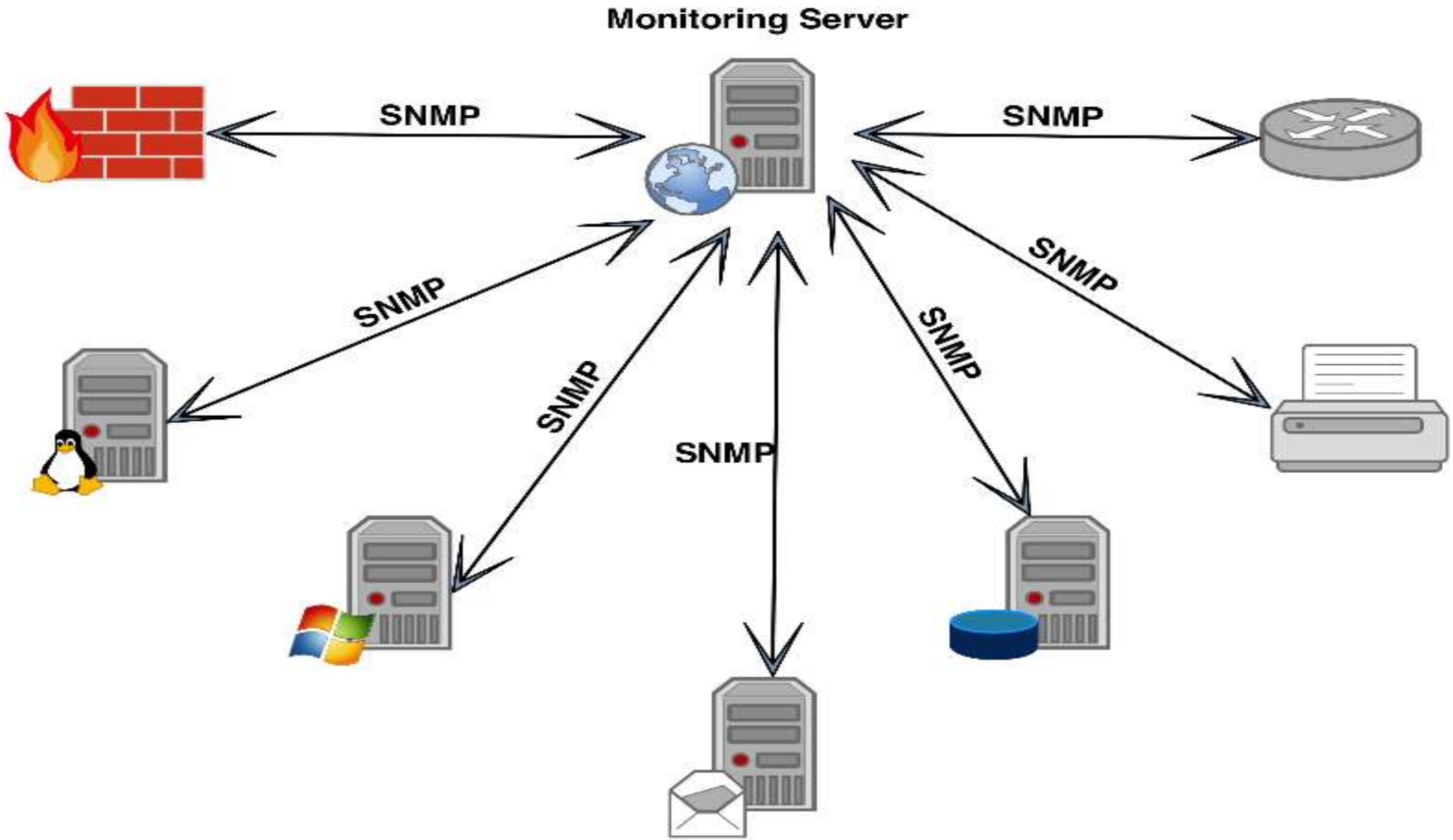
مقدمة

- في ظل الانفتاح الكبير على الانترنت وعلى الشبكات و زيادة نسبة الأجهزة التي تقوم بعملية إدارة الشبكات مثل الراوترات والسويتشات وزيادة فعاليتها يوم بعد يوم وحتى يستطيع مهندسو الشبكات مراقبة أجهزتهم و طريقة أدائها فكان لابد لهم من إيجاد بروتوكول خاص للمراقبة عن بعد يعطيهم بيانات دائمة لكفاءة عمل الأجهزة على الشبكة وبما فيها كفاءة عمل المعالج والرامات وكمية نقل البيانات ضمن الشبكة والكثير من خصائص المراقبة الهامة .

- تم البدء في تطوير بروتوكول ال SNMPعام 1998 ليقوم بعملية المراقبة وهو بروتوكول مطور من بروتوكول آخر تم تطويره عام 1987 اسمه SGMP (Simple Gateway Management Protocol) وجاء بعده بروتوكول آخر ظن الجميع انه سوف يحل مكان SNMPوهو CMIPلكن الآخير لم يدوم كثيرا لأن ال SNMP أثبت فعاليته بشكل أقوى على الساحة كونه يعمل على نطاقات واسعة وقابل للعمل مع جميع أنواع مكونات الشبكة .

ما هو SNMP Protocol

هو اختصار ل Simple Network Management Protocol أو بروتوكول إدارة الشبكة البسيط وهو بروتوكول انترنت Standard لجمع وتنظيم المعلومات حول الأجهزة الموجودة على الشبكة ويعمل في طبقة ال Application Layerمن نموذج ال OSIولتعديل تلك المعلومات وتغيير سلوك الجهاز وتشمل الأجهزة التي تدعم SNMP :
راوترات , سويتشات , سيرفرات , أجهزة الكومبيوتر , طابعات .


Monitoring Server
SNMP
SNMP
SNMP
SNMP
SNMP
SNMP
SNMP

استخدامات بروتوكول SNMP:

يستخدم على نطاق واسع في إدارة الشبكة ومراقبتها Network Management and Monitoring وتحليل الشبكة والأجهزة الموجودة بها ويتكون من مجموعة من معايير الشبكة Internet Protocol Suite بما في ذلك Application Layer و الاسكيم الخاصة بقاعدة البيانات Database Scheme و مجموعة من عناصر البيانات .

ما هي إصدارات وأنواع بروتوكول ال SNMP

تم تطوير ونشر ثلاث إصدارات مهمة من بروتوكول SNMP و هي الإصدار الأول

SNMPv_1إصدارين أحدث هما SNMP v_2و SNMP_v3و تتميز الإصدارات الأحدث بتحسينات في الأداء والمرونة والأمان .

كيف يعمل ال SNMP

لنتفق أولا على الأشياء الأساسية التي يجب معرفتها وهي أن هذا البروتوكول هو أحد بروتوكولات الطبقة السابعة ويستخدم ال UDP/IP للإرسال ومن خلال البورت 161و162 وهو يستخدم خمس أنواع من الرسائل للتواصل بين السيرفر والعميل وهي :

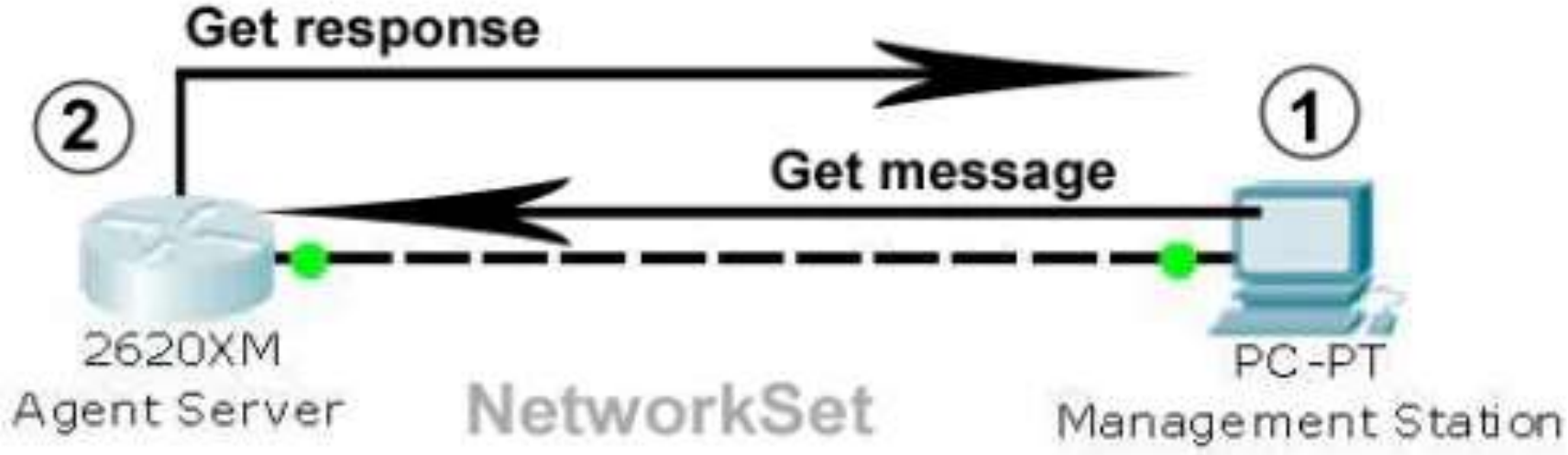
1. TRAP
2. SET
3. GET_RESPONSE
4. GET _NEXT
5. MESSAGE GET

- SNMP Manager:وهو الجهاز المسؤول عن مراقبة و إدارة الأجهزة المتصلة بالشبكة ويتم إدارة و مراقبة هذه الأجهزة عن طريق برنامج يستخدم بروتوكول SNMPويتم تنزيله على الجهاز المسؤول عن ذلك (SNMP Manager).
- SNMP Agent: هي الأجهزة التي تتم إدارتها من خلال SNMP Manager وهذه الأجهزة إما أن تكون Router, Switch , Server, PC ويقوم ال SNMP Agentبإرسال رسالة تسمى Trap Messageترسل أي تغير يحدث في الأجهزة إلى ال SNMP Manager .

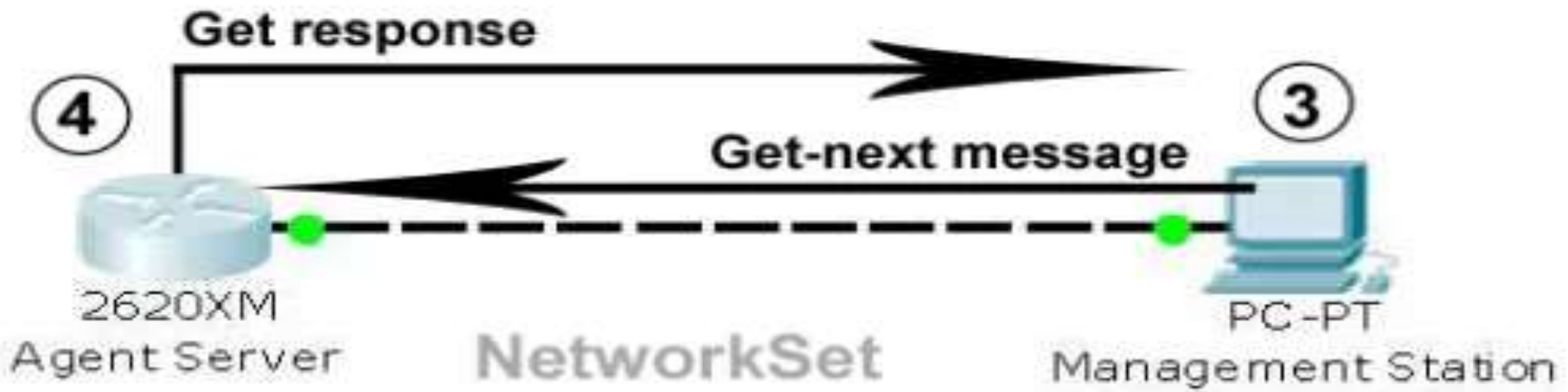
يستخدم بروتوكول SNMP البورت رقم 161
لإنشاء اتصال بين SNMP Manager and
SNMP Agent.

PROCESS	PROTOCOL	PORT NUMBER
Request receipt by the agent	UDP	161
Manager's communication with the agent	UDP	161
Notification receipt by the manager	UDP	162
Agent's notification generation		Any available port
Request receipt	TLS/DTLS	10161
Notification receipt	TLS/DTLS	10162

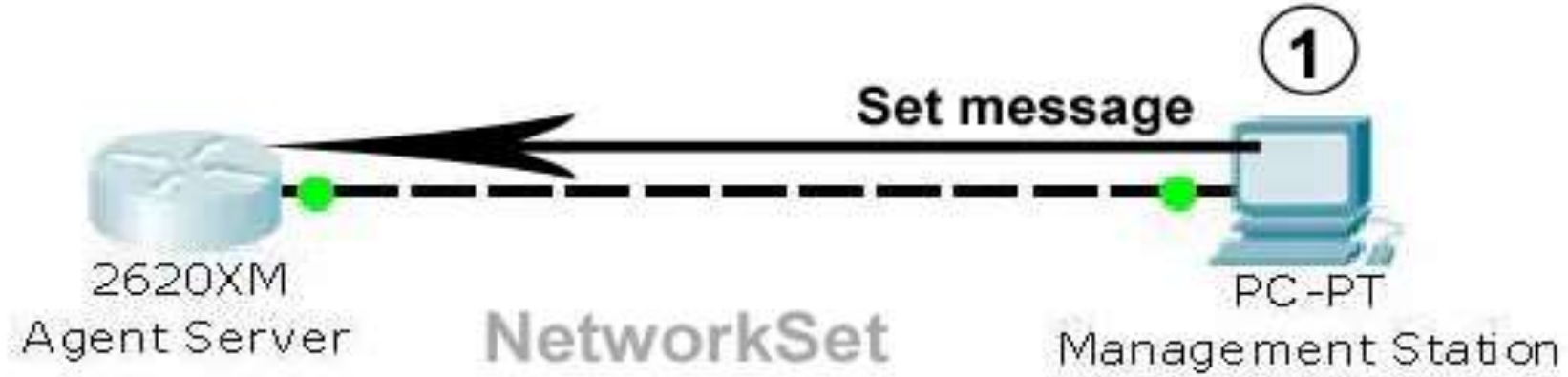
فعندما يريد العميل أن يبدأ بالمراقبة يقوم بإرسال GET Message إلى ال Agent وهو بدوره يرسل المطلوب على شكل GET RESPONSE
وكما نرى من الصورة القادمة



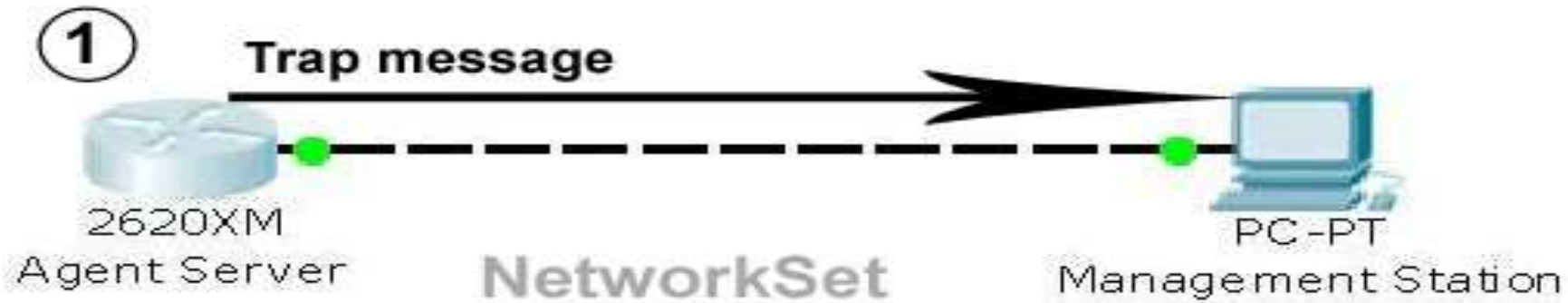
أما بالنسبة للرسالة GET _Next MESSAGE فهي عندما يريد أن يتابع المراقبة ويرغب في الحصول على المزيد من المتغيرات :



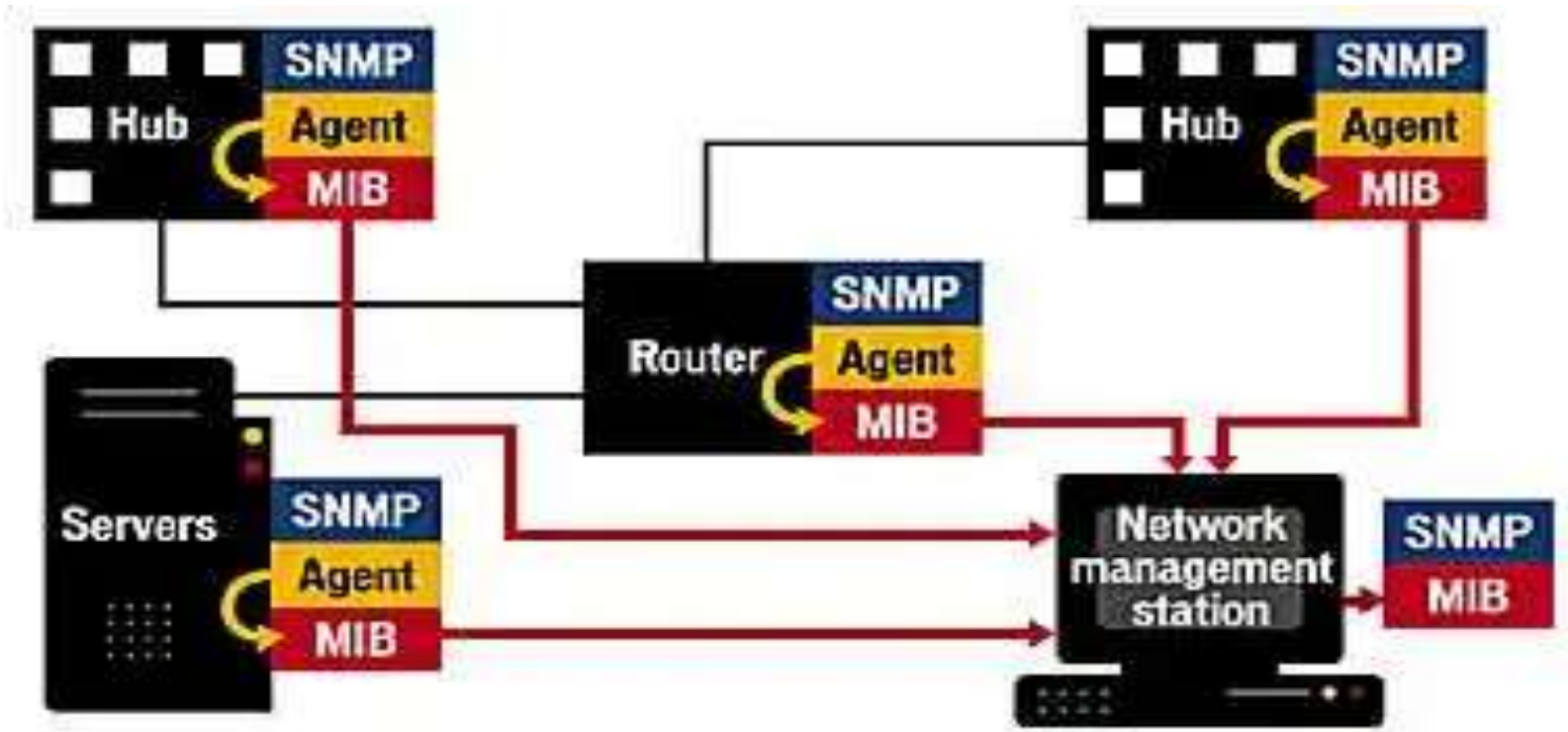
رسالة ال SET ترسل من قبل العميل لكي يطلب من ال Agent تحديد الشيء الذي تم تغييره على السيرفر (تغيير قيمة) :



رسالة ال Trap ترسل من قبل ال Agentفي حال حدوث شيء ما في جهاز المراقب مثلا توقف بورت عن العمل (Link down/up) وفي هذه الحالة يرسلها على البورت 162 بينما باقي الرسائل ترسل على البورت 161 .



هذه الصورة تشرح لنا عمل بروتوكول ال SNMPمن الداخل :



يتضح لنا أن ال AGENT يأخذ الطلب من العميل ويضعه في MIB (Management Information Base وبدوره يقوم ال MIB بإرسال المعلومات المطلوبة إلى العميل .

```
# importing the required module
from pysnmp import hlapi

# defining the get() function
def get(
    target,
    oids,
    credentials,
    port = 161,
    engine = hlapi.SnmpEngine(),
    context = hlapi.ContextData()
    ):
    handler = hlapi.getCmd(
        engine,
        credentials,
        hlapi.UdpTransportTarget((target, port)),
        context,
        *construct_object_types(oids)
    )
```

- من مقتطفات الكود أعلاه يمكننا الاستفادة من واجهة برمجة التطبيقات عالية المستوى ل pySNMP . حددنا وظيفة بسيطة متل GET()والتي تتطلب عنوان أو اسم جهاز بعيد .
- يحتاج إلى قائمة معرفة الكائنات (OIDS) التي نحتاج للحصول عليها وبعد ذلك مجموعة من بيانات الاعتماد لمصادقة الجلسة ويمكننا تحديد منفذ UDP مميز إذا احتجنا.

```
def construct_object_types(listOfOids):
    objectTypes = []
    for oid in listOfOids:
        objectTypes.append(hlapi.ObjectType(hlapi.ObjectIdentity(oid)))
    return objectTypes
```

- يعرض مقتطف الشفرة أعلاه قائمة يمكن توسيعها عن طريق إضافة * مسبقًا ، كما فعلنا في طريقةget ().

```
def fetch(handler, count):
    res = []
    for i in range(count):
        try:
            error_indication, error_status, error_index, var_binds = next(handler)
            if not error_indication and not error_status:
                items = {}
                for var_bind in var_binds:
                    items[str(var_bind[0])] = cast(var_bind[1])
                res.append(items)
            else:
                raise RuntimeError('Got SNMP error: {0}'.format(error_indication))
        except StopIteration:
            break
    return res
```

- في مقتطف الكود أعلاه ، أنشأنا طريقة try-exceptلإيقاف التكرار لسبب محدد. في الحالات التي يحدد فيها المستخدم عددًا أكبر من عدد العناصر التي لدينا بالفعل

```
def cast(val):
    try:
        return int(val)
    except (ValueError, TypeError):
        try:
            return float(val)
        except (ValueError, TypeError):
            try:
                return str(val)
            except (ValueError, TypeError):
                pass
    return val
```

```
hlapi.UsmUserData(
    'testuser',
    authKey = 'authenticationkey',
    privKey = 'encryptionkey',
    authProtocol = hlapi.usmHMACSHAAuthProtocol,
    privProtocol = hlapi.usmAesCfb128Protocol
    )
```

Output:

```
{'1.3.6.1.2.1.1.5.0': 'R1.sdn.local'}
```

```
def get_bulk(
    target,
    oids,
    credentials,
    count,
    start_from = 0,
    port = 161,
    engine = hlapi.SnmpEngine(),
    context = hlapi.ContextData()):
    handler = hlapi.bulkCmd(
        engine,
        credentials,
        hlapi.UdpTransportTarget(( target, port )),
        context,
        start_from, count,
        *construct_object_types( oids )
    )
    return fetch(handler, count)
```

```
def get_bulk_auto(
    target,
    oids,
    credentials,
    count_oid,
    start_from = 0,
    port = 161,
    engine = hlapi.SnmpEngine(),
    context = hlapi.ContextData()):
    count = get(
        target,
        [count_oid],
        credentials,
        port,
        engine,
        context
        )[count_oid]
    return get_bulk(target, oids, credentials, count, start_from, port, engine, context)
```

```
ele = get_bulk_auto(
    '10.0.0.1',
    ['1.3.6.1.2.1.2.2.1.2 ', '1.3.6.1.2.1.31.1.1.1.18'],
    hlapi.CommunityData('JAVATPOINT'),
    '1.3.6.1.2.1.2.1.0')
for i in ele:
    for x, y in i.items():
        print("{0} = {1}".format(x, y))
    print('')
```

Output:

```
1.3.6.1.2.1.2.2.1.2.1=FastEthernet1/0
1.3.6.1.2.1.31.1.1.1.18.1=
1.3.6.1.2.1.2.2.1.2.2=FastEthernet0/0
1.3.6.1.2.1.31.1.1.1.18.2=
1.3.6.1.2.1.2.2.1.2.3=FastEthernet0/1
1.3.6.1.2.1.31.1.1.1.18.3=Test Desc
1.3.6.1.2.1.2.2.1.2.4=Serial2/0
1.3.6.1.2.1.31.1.1.1.18.4=
1.3.6.1.2.1.2.2.1.2.5=Serial2/1
1.3.6.1.2.1.31.1.1.1.18.5=
1.3.6.1.2.1.2.2.1.2.6=Serial2/2
1.3.6.1.2.1.31.1.1.1.18.6=
1.3.6.1.2.1.2.2.1.2.7=Serial2/3
1.3.6.1.2.1.31.1.1.1.18.7=
1.3.6.1.2.1.2.2.1.2.9=Null0
1.3.6.1.2.1.31.1.1.1.18.9=
```