SNMP (Simple Network Management Protocol)

Disciplina: Gerência de redes

Grupo: André Arthur,

Bartolomeu Nunes,

Cristiliano Negreiros,

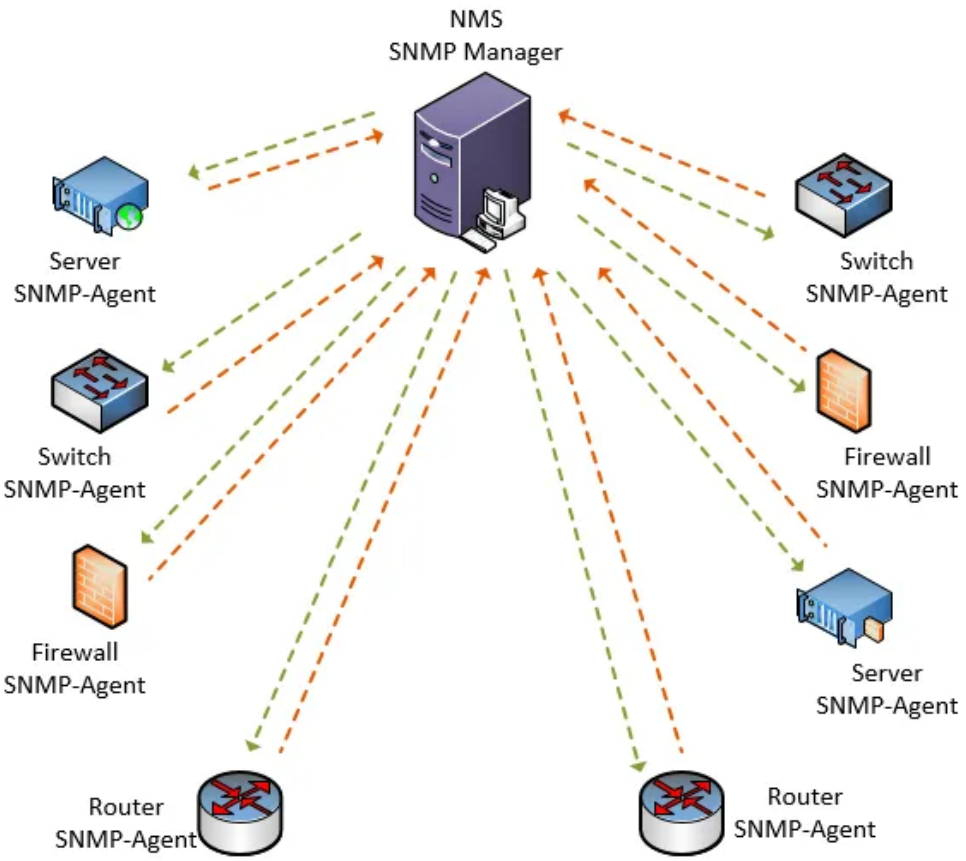
João Nogueira

Universidade de Pernambuco

Recife - 2022

SNMP:

É um protocolo padrão, utilizado para realizar o gerenciamento de dispositivos dentro de uma rede, o gerenciamento é feito de forma que o gerente(aplicação que roda no sistema de gerenciamento) vai ser capaz de coletar informações sobre o dispositivo com o agente(software que vai monitorar o dispositivo gerido) e essas informações vão servir para auxiliar no gerenciamento destes dispositivos, como por exemplo intensidade do tráfego, sistema operacional, utilização do cpu, temperatura, entre outros.

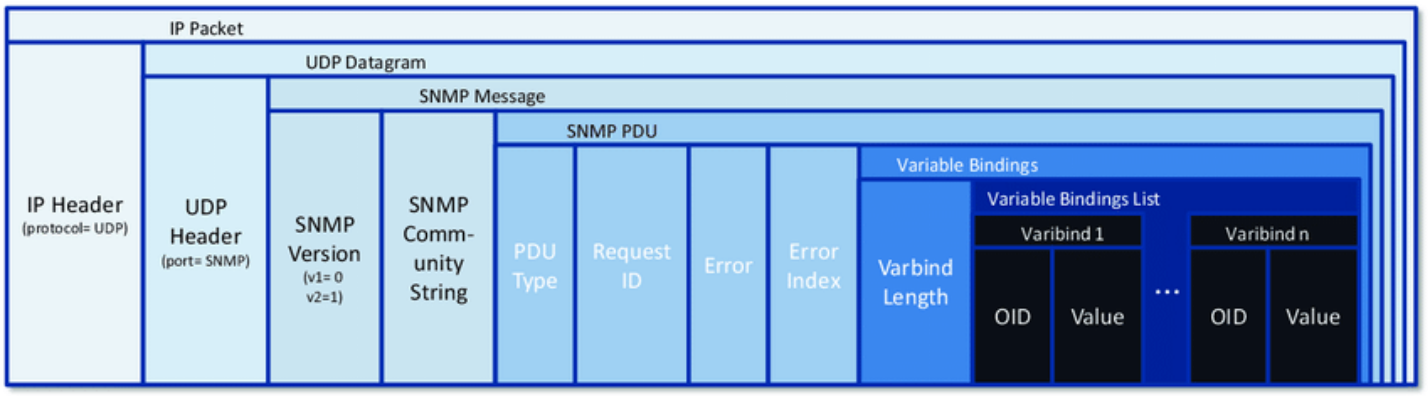


https://getlabsdone.com/what-is-snmp-and-how-it-works/

Detalhes do Protocolo:

O SNMP opera na camada de aplicação do modelo OSI, onde todas as informações são transportadas por meio do protocolo UDP. Os agentes SNMP recebem as requisições UDP na porta 161 e podem responder na porta de origem do gerente. O gerente recebe notificações (trap ou informRequest) na porta 162.
As unidades de dados de protocolo(PDU) SNMP são construídas da seguinte forma:

IP header	UDP header	version	community	PDU-type	request-id	error-status	error-index	variable bindings



https://www.researchgate.net/figure/IP-packet-encapsulation-of-SNMP-version-2c_fig37_262493920

Tipos de PDUs:

Os 7 tipos de PDUs identificados no campo *PDU-type* acima são os seguintes.

GetRequest
usado para retirar um pedaço de informação de gerenciamento.

SetRequest
usado para fazer uma mudança no subsistema gerido.

GetNextRequest
usado interativamente para retirar sequências de informação de gerenciamento.

GetBulkRequest
usado para retirar informações de um grupo de objetos.

Response
Retorna associações de variáveis e reconhecimento do agente para o gerente para GetRequest, SetRequest, GetNextRequest, GetBulkRequest e InformRequest.

Trap
usado para reportar uma notificação ou para outros eventos assíncronos sobre o subsistema gerido.

InformRequest
Notificação assíncrona reconhecida com confirmação do recebimento da PDU de trap.

Community string:

Uma **community string** pode ser configurada no gerenciador SNMP e a mesma string deve ser configurada no agente para se comunicar com o gerenciador. Por padrão, a maioria dos equipamentos de rede usa a **community string** padrão public. Se a **community string** não corresponder em ambos os lados, o gerente e o agente não poderão se comunicar. Você pode ter uma **community string** que seja somente leitura ou

leitura-gravação. **Community string** somente leitura pode visualizar apenas as informações dentro do sistema, enquanto a leitura-gravação pode fazer as alterações no dispositivo.

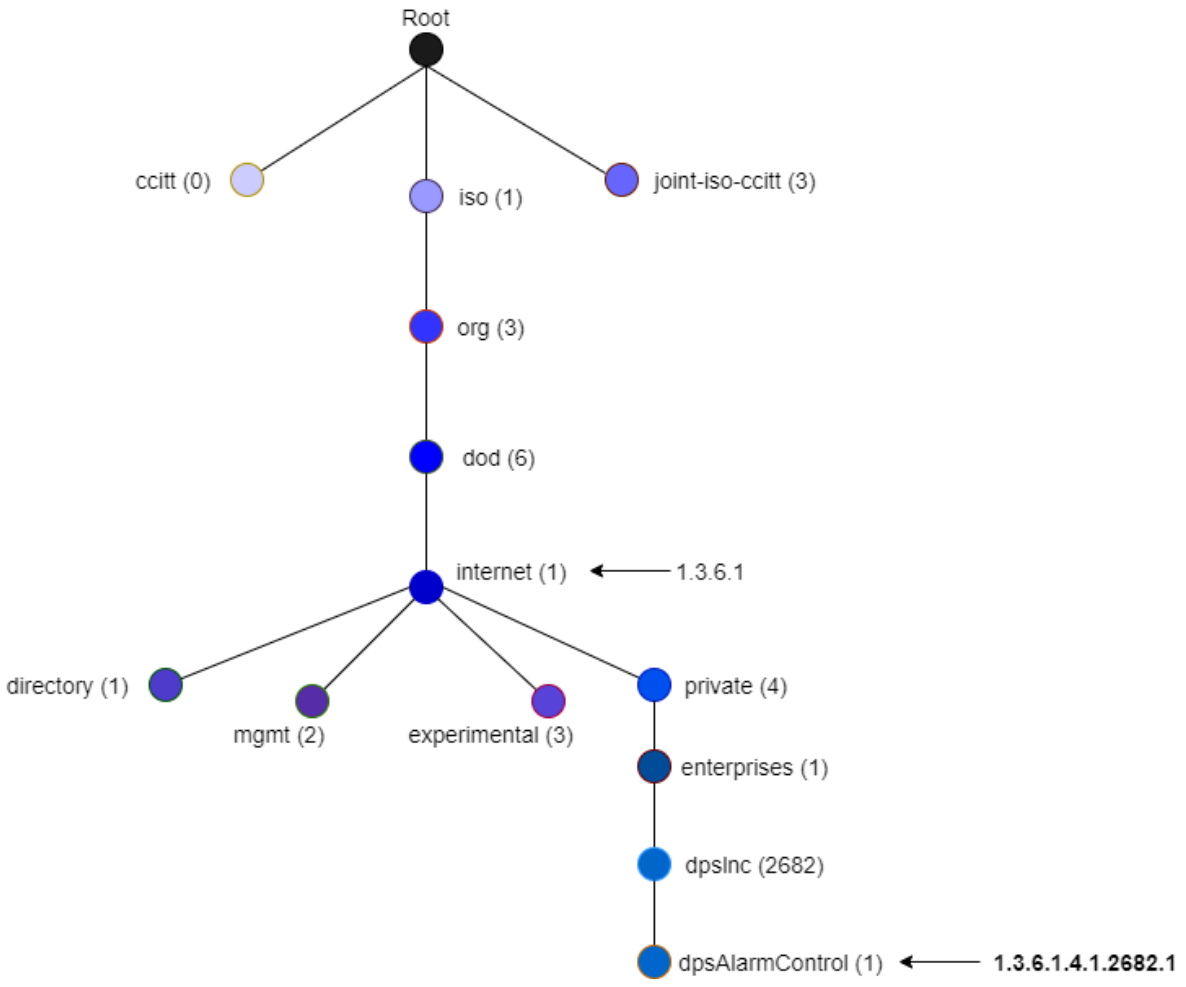
MIB:

MIB (Management Information Base) é uma coleção de informações organizadas hierarquicamente. Eles são acessados usando um protocolo como SNMP. MIBs são coleções de definições que definem as propriedades do objeto gerenciado dentro do dispositivo a ser gerenciado.

OID:

OIDs significa identificadores de objetos. Os OIDs identificam exclusivamente objetos gerenciados em uma hierarquia MIB. Isso pode ser descrito como uma árvore, cujos níveis são atribuídos por diferentes organizações. Os IDs de objeto MIB de nível superior (OIDs) pertencem a diferentes organizações padrão. Os fornecedores definem ramificações privadas, incluindo objetos gerenciados para seus próprios produtos.

Como mostrado na figura abaixo, podemos encontrar o OID desejado com base na árvore MIB.



https://www.dpstele.com/snmp/what-does-oid-network-elements.php

Código:

Definindo o OID , Community, Ip destino e porta.

```python
from struct import pack
import socket

### Definindo parâmetros do SNMP

OID = "1.2.1.1.1.0"
COMM = b'public'

### Definindo parâmetros do gerente

ipDest = b'127.0.0.1' #b"192.168.1.7"
portDest = 161

## Criando conexão no socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

#verifica criação do socket
if(s == -1):
    print('\n\nNão foi possível criar o socket')
else:
    print('\n\nsocket foi criado em ', s, ('\n'))

```

Montando a mensagem SNMP com os campos de Value,OID, Error Index, Error

```python
### Montando mensagem SNMP de trás pra frente

# Value field
val = b'mensagem de teste do snmp'
snmpVal = pack("bb{}s".format(len(val)),4,len(val),val) #byte 0x05  - data type nulo

# Object Field
OID = OID.split(".")  #System Description
snmpOid = pack("2b",0x2b, 0x06) + bytearray(int(x) for x in OID ) # adicionando iso.3 ao OID

# Sequence / Varbind Type Field
snmpVarbind = snmpOid + snmpVal
snmpVarbindSeq = pack('b',6) + len(snmpOid).to_bytes(1,'little') + snmpVarbind

# Sequence / Varbind List Field
snmpVarbindList = pack('b',0x30) + len(snmpVarbindSeq).to_bytes(1,"little") + snmpVarbindSeq
snmpVarbindList = pack('b',0x30) + len(snmpVarbindList).to_bytes(1,"little") +  snmpVarbindList

# Error Index
ErrIndex = pack("b",0x0)
pdu = pack("b",0x02) + len(ErrIndex).to_bytes(1,"little") +  ErrIndex + snmpVarbindList

# Error
Err = pack("b",0x0)
pdu = pack('b',2) + len(Err).to_bytes(1,"little") + Err + pdu
```

Com a mensagem SNMP montada podemos enviar para o ip destino e esperar uma resposta.

```python
# Mensagem SNMP Final
snmpMessage = pack('b',0x30) + len(snmpProtocolHeader).to_bytes(1,"little") + snmpProtocolHeader

print(f'Mensagem SNMP raw bytes:\n{snmpMessage}')

## Transmitindo mensagem

print("\n\nTransmitindo mensagem\n\n")
s.sendto(snmpMessage,(ipDest,portDest))

try:
    Rxbuf = s.recv(2000)
    print (f'Resposta Recebida!\n\nString Resposta:\n\n{Rxbuf.decode("utf-8", errors="ignore")}')

except socket.timeout:
    print ('time out!!!')

s.close()

print ('\n\nFim da Operacao. Socket fechado.')
```

Como resultado obtivemos a seguinte string de resposta.

```
socket foi criado em  <socket.socket fd=1780, family=AddressFamily.AF_INET, type=SocketKind.SOCK_DGRAM, proto=0>

Mensagem SNMP raw bytes:
b"0?\x02\x01\x00\x04\x06public\xa02\x02\x01\x02\x02\x01\x00\x02\x01\x000'0%\x06\x08+\x06\x01\x02\x01\x01\x01\x00\x04\x19mensagem de
teste do snmp"


Transmitindo mensagem


Resposta Recebida!

String Resposta:

0□□ □□public□□□□□ □□ 00□□+□□□□□□ □Hardware: Intel64 Family 6 Model 142 Stepping 9 AT/AT COMPATIBLE - Software: Windows Version 6.3
(Build 19043 Multiprocessor Free)


Fim da Operacao. Socket fechado.
```

Código Utilizando o Pysnmp:

O mesmo processo descrito acima utilizando o pysnmp.

```python
from pysnmp.entity.rfc3413.oneliner import cmdgen

def snmpget(oid,ip):
    cmdGen = cmdgen.CommandGenerator()

    errorIndication, errorStatus, errorIndex, varBinds = cmdGen.getCmd(
    cmdgen.CommunityData('public'),
    cmdgen.UdpTransportTarget((ip,161)),
    oid)

    if errorIndication:
        return (errorIndication)
    else:
        if errorStatus:
            return ('%s at %s' %{errorStatus.prettyPrint(),errorIndex and varBinds[int(errorIndex)-1] or '?'})
        else:
            for name,val in varBinds:
                return ('%s = %s' %(name.prettyPrint(), val.prettyPrint()))

```

Como resultado obtivemos as seguintes mensagens para os OIDs passados

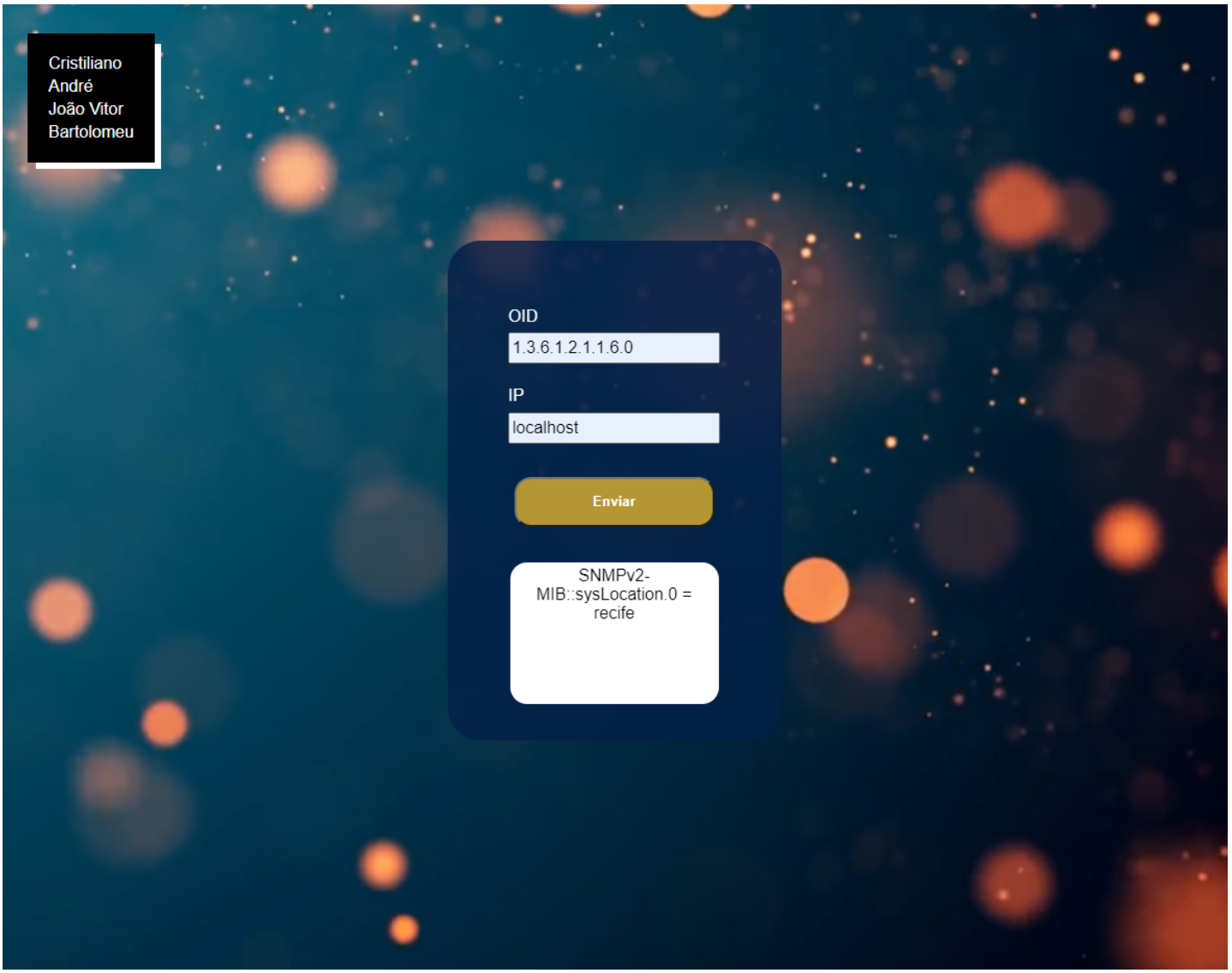
```python
snmpget('1.3.6.1.2.1.1.1.0','localhost') #SystemDescr
snmpget('1.3.6.1.2.1.1.5.0','localhost') #SystemName
snmpget('1.3.6.1.2.1.1.6.0','localhost') #SystemLocation
```

Python

```
SNMPv2-MIB::sysDescr.0 = Hardware: Intel64 Family 6 Model 142 Stepping 9 AT/AT COMPATIBLE - Software: Windows Version 6.3 (Build 19043
Multiprocessor Free)
SNMPv2-MIB::sysName.0 = LAPTOP-TISLCRU2
SNMPv2-MIB::sysLocation.0 = Brasil
```

Resultado Final:

Resultado do código desenvolvido para realizar o envio da mensagem SNMP com um front end em angular, api em flask e um server para comunicação via api.


Cristiliano
André
João Vitor
Bartolomeu
OID
1.3.6.1.2.1.1.6.0
IP
localhost
Enviar
SNMPv2-MIB::sysLocation.0 = recife