Redes Avançadas - T2

Gabriela Zorzo, Lucas Andreotti e Joana Figueredo

Enunciado

1. Escolha do elemento de rede

O elemento pode ser real ou simulado.

2. Escrita da MIB

A MIB deve ser escrita usando SMIv2 e deve conter de 10 a 15 informações sobre o elemento gerenciado com no mínimo 1 tabela. Obrigatoriamente devem ser implementados novos objetos, não sendo permitido o uso apenas de objetos já existentes na MIB-II ou em outras MIBs. Devem ser incluídos objetos com acesso para somente leitura e leitura/escrita.

3. Implementação do agente

O agente pode ser implementado usando o pacote net-snmp (http://www.net-snmp.org/) ou outra biblioteca/framework para desenvolvimento de agentes SNMP.

Net-snmp manpage (http://www.net-snmp.org/docs/man/snmpd.conf.html) - seção “EXTENDING AGENT FUNCTIONALITY”

O agente deverá permitir a execução de operações get, getnext e set.

Implementação

Elemento gerenciado

O elemento gerenciado é uma cafeteria simulada através do aquivo *cofeeshop.py*. A cafeteria possui os seguintes elementos:

- *revenue*: valor total das vendas.
- *totalOrders*: número total de pedidos.
- *coffee*: número de cafés pedidos.
- *tea*: número de chás pedidos.
- *soda*: número de refrigerantes pedidos.
- *muffin*: número de muffins pedidos.
- *sandwich*: número de sanduíches pedidos.
- *pie*: número de tortas pedidos.
- *employees*: lista de funcionários.
- *status*: status da cafeteria (aberta ou fechada).

O elemento gerenciado, quando executado, escreve o valor de cada elemento em um arquivo json chamado *logs.json*. A atualização do arquivo ocorre a cada 10 segundos, simulando um novo pedido aleatório sendo solicitado.

Para executar o simulador da cafeteria, basta abrir um terminal e executar:

```
python coffeeshop.py
```

Objetos da MIB

A implementação da COFFEESHOP MIB está no arquivo *COFFEESHOP.TXT*. A MIB está localizada no ramo *experimental* e possui 10 itens, sendo 8 destes do tipo *Integer32*, 1 do tipo *DisplayString* e 1 tabela. A seguir, um exemplo de cada tipo de objeto implementado.

Integer32

O objeto *revenue* pode ser acessado pelo OID = .1.3.6.1.3.1234.1.1.0.

```
revenue OBJECT-TYPE
     SYNTAX  Integer32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
             "The total revenue of the coffeeshop."
     ::= { items 1 }
```

DisplayString

O objeto *status* pode ser acessado pelo OID = .1.3.6.1.3.1234.1.9.0. Além disso, esse objeto pode receber a operação *set* pois permite a escrita.

```
status OBJECT-TYPE
     SYNTAX DisplayString (SIZE(0..255))
     MAX-ACCESS read-write
     STATUS current
     DESCRIPTION
        "The status of the coffeeshop (open or closed)."
    ::= { items 9 }
```

Tabela

O objeto *employeesTable* é uma tabela composta por uma sequência de objetos *Employee*. Cada objeto *Employee* possui um identificador *employeeId* do tipo *Integer32* e um nome *employeeName* do tipo *DisplayString*.

```
employeesTable OBJECT-TYPE
    SYNTAX SEQUENCE OF Employee
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "The cofeeshop employees list."
    ::= { items 10 }

employee OBJECT-TYPE
    SYNTAX Employee
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "An employee."
    INDEX { employeeId }
    ::= { employeesTable 1 }

Employee ::= SEQUENCE {
    employeeId Integer32,
    employeeName DisplayString (SIZE(0..255))
}
```

Agente

O agente implementado faz a leitura do arquivo *logs.json* para retornar o valor solicitado do elemento gerenciado de acordo com o OID do objeto. Ele suporta os comandos de *get*, *getnext* e *set*.

Os elementos gerenciados, seus respectivos OIDs:

- *revenue*: .1.3.6.1.3.1234.1.1.0
- *totalOrders*: .1.3.6.1.3.1234.1.2.0
- *coffee*: .1.3.6.1.3.1234.1.3.0
- *tea*: .1.3.6.1.3.1234.1.4.0
- *soda*: .1.3.6.1.3.1234.1.5.0
- *muffin*: .1.3.6.1.3.1234.1.6.0
- *sandwich*: .1.3.6.1.3.1234.1.7.0
- *pie*: .1.3.6.1.3.1234.1.8.0
- *status*: .1.3.6.1.3.1234.1.9.0
- *employeesTable*: .1.3.6.1.3.1234.1.10
- *employee*: .1.3.6.1.3.1234.1.10.1
- *employeeId*: .1.3.6.1.3.1234.1.10.1.1
- *employeeName*: .1.3.6.1.3.1234.1.10.1.2

Todos os elementos suportam os comandos *get* e *getnext*. Para acessar os elementos da tabela *employeeTable* é necessário acessar o objeto final, dado pelo número do objeto *employee*, o número do parâmetro desejado (*employeeId* ou *employeeName*), seguido do número da linha: .1.3.6.1.3.1234.1.10.1.{*employeeId*|*employeeName*}.{*numLinha*}. O elemento *status* suporta também o comando *set*.

Para executar somente o agente, podem ser executados os seguintes comandos no terminal:

get

```
python agent.py -g .1.3.6.1.3.1234.1.9.0
```

getnext

```
python agent.py -n .1.3.6.1.3.1234.1.9.0
```

set

```
python agent.py -s .1.3.6.1.3.1234.1.9.0 s Close
```

Exemplos execução agente com comandos SNMP

O passo-a-passo para execução do trabalho está implementado no arquivo *SNMP.ipynb*.

Antes de executar, atualizar o caminho absoluto dos arquivos que serão consultados.

SNMP.ipyn (Agente)

- pass .1.3.6.1.3.1234.1 /usr/bin/python3 {path}/agent.py - {path} deve ser substituído pelo caminho absoluto da pasta no GitHub.

agent.py e coffeeshop.py

- PATH = {path} - {path} deve ser substituído pelo caminho absoluto da pasta no GitHub.

EXEMPLOS DE EXECUÇÃO OID

get

```
snmpget -v2c -c public localhost .1.3.6.1.3.1234.1.1.0

# retorno: SNMPv2-SMI::experimental.1234.1.1.0 = INTEGER: 104
```

getnext

```
# primeiro ínidice da tabela
snmpgetnext -v2c -c public localhost .1.3.6.1.3.1234.1.10

# retorno: SNMPv2-SMI::experimental.1234.1.10.1.1 = INTEGER: 1
```

set

```
snmpset -v2c -c private localhost .1.3.6.1.3.1234.1.9.0 s Close

# retorno: SNMPv2-SMI::experimental.1234.1.9.0 = STRING: "Close"
```

EXEMPLOS DE EXECUÇÃO NOME OBJETO

get

```
snmpget -v2c -c public -M +. -m +COFFEESHOP localhost revenue.0

# retorno: COFFEESHOP::revenue.0 = INTEGER: 104
```

getnext

```
# primeiro ínidice da tabela
snmpgetnext -v2c -c public -M +. -m +COFFEESHOP localhost employeesTable

# retorno: COFFEESHOP::employeeId = INTEGER: 1
```

set

```
snmpset -v2c -c private -M +. -m +COFFEESHOP localhost status.0 s Open

# retorno: COFFEESHOP::status.0 = STRING: Open
```

Mais exemplos podem ser vistos no arquivo *SNMP.ipynb*.