Teste Gestão de Redes

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Gestão e Virtualização de Redes

Gestão de Redes

🗓️ : **15 de janeiro de 2021**

👤 : **Joana Isabel Afonso Gomes - a84912**

QUESTÃO I

a) O mecanismo de alarmes veio diminuir a dependência do polling intensivo nas versões prévias do SNMP (as aplicações de gestão fazerem polling à informação que está nos agentes), passando a responsabilidade de processamento para os agentes (as aplicações de gestão deixam de precisar de fazer pedidos). Os alarmes são enviadas do agente para a aplicação de gestão para avisar de eventos extraordinários como mudanças no estado da interface de rede, mau funcionamento da memória, limite de temperatura ultrapassado, entre outros.

Para além disso, melhorou a escalabilidade pobre de versões prévias, sendo que as maiores desvantagens do SNMP passavam (e continuam a passar) pelos problemas escalabidade e a falta de segurança, sendo este um aspeto relevante a melhorar.

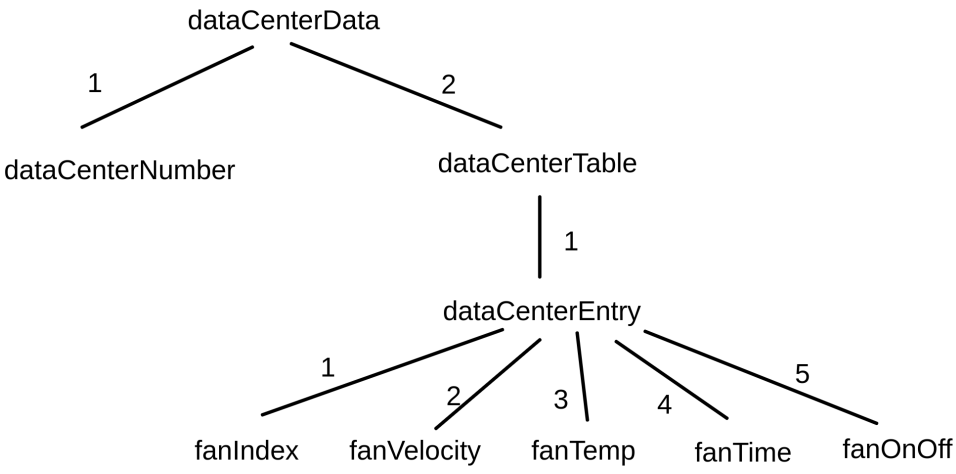
b)

```
void rebootInterfaces() {
    String add = "192.168.1.255:161" //endereço da porta
    String numberOID = ".1.3.6.1.2.1.2.1" // OID do obj com o nr de interf no sistema
    String ifAdminStatus = ".1.3.6.1.2.1.2.2.1.7" // OID do obj com o estado desired
                                                  //das interfaces
    String communityString = "community-string"

    int countInterfaces = snmpget -v2c -c communityString add numberOID

    for(int i=1; i<=countInterfaces; i++){
      snmpset -v2c -c communityString add ifAdminStatus.i 2 //por todos down
      snmpset -v2c -c communityString add ifAdminStatus.i 1 //por todos up
```

```
    }
}
```

QUESTÃO II

a)




```
dataCenterNumber OBJECT_TYPE
  SYNTAX INTEGER
  ACCESS read-write
  STATUS mandatory
  DESCRIPTION
    "Numero de ventoinhas"
  ::= { sensoresData 1 }

dataCenterTable OBJECT_TYPE
  SYNTAX SEQUENCE OF DataCenterEntry
  ACCESS read-write
  STATUS mandatory
  DESCRIPTION
    "Tabela"
  ::= { dataCenter 2 }

dataCenterEntry OBJECT_TYPE
  SYNTAX DataCenterEntry
  ACCESS not-accessible
  STATUS mandatory
  DESCRIPTION
    "Entrada da tabela"
  INDEX {fanIndex}
  ::= { dataCenterTable 1 }

DataCenterEntry :: SEQUENCE {
```

```
  fanIndex INTEGER,
  fanVelocity INTEGER,
  fanTemp INTEGER,
  fanTime TIMETICKS
  fanOnOff INTEGER
}

fanIndex OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS read-write
  STATUS mandatory
  DESCRIPTION
    "Indice da ventoinha na tabela"
::= { dataCenterEntry 1}

fanVelocity OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS read-write
  STATUS mandatory
  DESCRIPTION
    "Velocidade da ventoinha. 1-off , 2-lento, 3-medio, 4-maximo"
::= { dataCenterEntry 2}

fanTemp OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS read-write
  STATUS mandatory
  DESCRIPTION
    "Temperatura da Ventoinha"
::= { dataCenterEntry 3}

fanTime OBJECT-TYPE
  SYNTAX TIMETICKS
  ACCESS read-write
  STATUS mandatory
  DESCRIPTION
    "Tempo para cada nível"
::= { dataCenterEntry 4}

fanOnOff OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS read-write
  STATUS mandatory
  DESCRIPTION
    "Estado da ventoinha. 1-ON 2-OFF"
::= { dataCenterEntry 5}
```

b)

```
void monitoringFan(int maxTime, int minTime){
  int minTemp = 20;
  int maxTemp = 30;

  int fanNumber = snmpget("dataCenterData.1"); //nr de ventoinhas
```

```
  for(int i=0; i<=fanNumber; i++){

    int temp = snmpget("dataCenterData.2.1.3"+i) //temp da ventoinha

    if(temp <= minTemp){ //desligar a ventoinha
      snmpset("dataCenterData.2.1.5"+i,2);
    }

    else if(temp >=maxTemp){ //ligar a ventoinha e passar a maxTime
      snmpset("dataCenterData.2.1.5"+i,1);
      snmpset("dataCenterData.2.1.4"+i,maxTime);
    }

    else{ //in between minTemp e maxTemp ventoinha ligada e passar a minTime
      snmpset("dataCenterData.2.1.5"+i,1);
      snmpset("dataCenterData.2.1.4"+i,minTime);
    }

  }
}
```