,

# Universidade do Minho Departamento de Sistemas de Informática

**MIETI, UMINHO 2022/2023**

**Módulo de Gestão Redes**

**Relatório do Projeto de Grupo**

Grupo:



José Gomes

A93XXX

Diogo Cerqueira

A93108

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Índice

[1. Introdução 6](#_Toc125119599)

[2. Requisitos 8](#_Toc125119600)

[3. Implementação 9](#_Toc125119601)

[3.1 Linguagem de Programação 9](#_Toc125119602)

[3.2 MIB 10](#_Toc125119603)

[3.3 Classes 13](#_Toc125119604)

[4. Teste, Resultados e Discussão 22](#_Toc125119658)

[4.1 Discussão de Resultados 24](#_Toc125119659)

[5. Autoavaliação 25](#_Toc125119660)

[6. Referências 26](#_Toc125119661)

[7. Conclusão 27](#_Toc125119662)

# Introdução

No âmbito da Unidade Curricular de Gestão de Redes foi proposto a realização de um trabalho que tem como objetivo final o desenvolvimento de um projeto de um agente proxy SNMPv2c utilizando o conhecimento adquirido ao longo do semestre.

Este relatório tem comos objetivos principais apresentar uma breve descrição do problema, uma formulação deste, e iremos descrever todas as decisões tomadas por o grupo, e explicar os resultados obtidos.

De forma a realizar o trabalho com sucesso, anteriormente, foi importante relembrar a linguagem de programação Java, cuja foi a linguagem de programação escolhida pelo grupo para a realização deste projeto.

# Requisitos

De forma a garantir o planeamento prévio e adequado da exequibilidade do presente trabalho torna-se essencial identificar claramente os seus requisitos, uma vez que os mesmos servirão de base ao trabalho executado posteriormente.

* 1. **Requisitos funcionais**

Para que o sistema trabalhe com o funcionamento desejado é necessário que cumpra os seguintes requisitos:

* + - Possuir um pacote freeware instalado com suporte a SNMP – o grupo decidiu utilizar o Net-SNMP;
    - Acrecentar mais algum
  1. **Requisitos não funcionais**

Os próximos requisitos, que serão mostrados em baixo são não funcionais o que significa que os mesmo não estão diretamente ligados às funcionalidades do sistema, mas sim relacionados com o tempo de respostas e de fiabilidade. Os requisitos são definidos por:

* Código simples e otimizado, que levará a um menor tempo de resposta do código;
* Código bem comentado para ser fácil a sua compreensão

# Lista de Figuras

[Figura 1 - Esquema que representa a MIB 7](file:///C:\Users\diogo\Downloads\Relatorio_GR.docx#_Toc125387289)

[Figura 2 - Exemplo de um snmpget 10](file:///C:\Users\diogo\Downloads\Relatorio_GR.docx#_Toc125387290)

[Figura 3 - Exemplo do envio de um comando pela Socket 11](file:///C:\Users\diogo\Downloads\Relatorio_GR.docx#_Toc125387291)

[Figura 4 - Funções de encriptação e desencriptação 13](file:///C:\Users\diogo\Downloads\Relatorio_GR.docx#_Toc125387292)

# Implementação

# 4.1 Linguagem de Programação

De forma a efetuar a realização do código do projeto, foi decidido a utilização da linguagem de programação Java, pois, além de ser a linguagem para a qual os membros do grupo dão preferência, foi possível descobrir através de uma pesquisa que esta possui bibliotecas que facilitam a implementação da primitiva SNMPv2c.

As bibliotecas utilizadas foram:

* **Crypto** – usada para facilitar a implementação da cifra AES que faz a encriptação dos dados comunicados entre o Agent e o Manager
* **Socket** – usada para a criação da socket que permite a comunicação entre o Agent e o Manager
* **LocalDateTime** – usada para funcionalidades que envolvam o tempo
* **Objects** – usada para facilitar a utilização dos objetos do programa

# 4.2 MIB

A nossa MIB é constituída por uma tabela (OperTable) cuja é constituída por variadas entradas. Na figura seguinte é possivel observar a sua constituição:

Diagrama

Descrição gerada automaticamente

Figura 1 - Esquema que representa a MIB

**Objetos da MIB:**

* **OperTable**: Conjunto que possui as entradas
* **OperEntry:** Conjunto que possui os objetos
* **idOper:** é o identificador da operação recebida pelo agente. É a chave da tabela.
* **typeOper:** mostra o tipo de operação SNMP.
* **operArg1:**
* **operArg2:**
* **idSource:** identificador da fonte do request.
* **idDestination:** identificador do destido do request e onde a operação será executada.
* **idArg:** OID do objeto da MIB e cujo é o argumentoda operação a ser executada pelo agente.
* **valueArg:** valor do objeto referido pelo idArg e cujo é o resultado recebido no agente proxy vindo do agente SNMP.
* **typeArg:** tipo de dados do valueArg.
* **sizeArg:** tamanho em bytes do valueArg
* **ttl:** tempo de vida restante da entrada na tabela. Quando este objeto possuir o valor zero a entrada é retirada e o Manager não poderá aceder aos valores.
* **status:** é um número que corresponde ao estado do request (accepted, created, entre outros)

Falar da MIB, Como alteramos as coisas e vamos buscar

# 4.3 Classes

Este trabalho possui 6 classes:

# Agent

# Nesta classe é criado um Agent que irá comunicar com o Manager.

# É nesta classe que é criado a socket “server” a qual os managers poderão aceder para mandar os dados.

# Enquanto o Manager não comunicar “exit”, o agent vai esperar pelos requests. Depois de recebidos, os dados são desencriptados e de seguida são analisados. Feita a análise, o Agent vai buscar à MIB os dados pedidos pelo manager.

# Na figura seguinte é possível os dados retornados a um comando “snmpget”

# Texto Descrição gerada automaticamente

Figura 2- Exemplo de um snmpget

# 

# Manager

# Nesta classe é criada uma Socket que vai conectar à Socket Server do Agente (Modelo Cliente-Servidor) . Estas 2 Sockets conectam-se devido a estarem a usar a mesma porta. Esta classe recebe os dados escritos pelo utilizador através do terminal, encripta-os e envia-os para o Agent.

# Texto Descrição gerada automaticamente

Figura 3- Exemplo do envio de um comando pela Socket

# StatusCodes

# Esta classe os valores (códigos) que vão ser inseridos no objeto “status” para descrever o estado do request. Os valores possíveis são:

# OK – 200

# Created – 201

# Accepted – 202

# Bad Request – 400

# Unauthorized – 401

# Forbidden – 403

# Not Found - 404

# OperEntry

# 

# Nesta classe é criado um Objeto OperEntry que irá possuir vários atributos que se referem aos objetos que caracterizam o request.

# Possui uma função que compara 2 OperEntry e verifica se são iguais. Além disto, possui as tradicionais funções de get() e set() dos seus atributos e a toString().

# MIBProxy

# Nesta classe é

# CipherAES

# Nesta classe são implementados os métodos de criptografia usados para encriptar o canal de comunicação entre o Manager e o Agent. A cifra escolhida para este processo foi o AES. Esta classe é muito importante pois possui o IV e a chave da cifra. Também possui as funções de encriptação e desencriptação, cujas podemos observar na figura seguinte.

# Texto Descrição gerada automaticamente

Figura 4- Funções de encriptação e desencriptação

# 

# 4.4 Canal de Comunicação

# A comunicação entre o Manager e o Agent realiza-se através de Sockets com base num modelo de Cliente - Servidor. É criada uma Socket Server no Agent que espera que a Socket Client criada no Manager se conecte a esta. A porta usada é a 5000 mas pode ser alterada, desde que a escolhida esteja livre, ou seja, não esteja a ser usada.

# Existe uma proteção do tráfego pois os dados são encriptados através da utilização de técnicas de criptografia, mais concretamente, a cifra AES. Esta é uma cifra simétrica (usa a mesma chave para encriptar e desencriptar o tráfego). Esta possui um vetor aleatório, denominado IV, que traz aleatoridade ao criptograma. Decidimos escolher esta cifra pois é uma cifra considerada inquebrável e que não traz uma sobrecarga computacional elevada ao processo.

# Teste, Resultados e Discussão

Os testes seguintes foram realizados para o ficheiro de texto “Circuito do Mónaco”

**BFS**

Custo = 24

**DFS**

Custo = 26

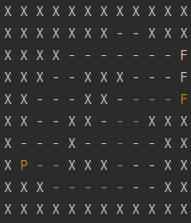
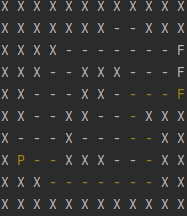


Fig 6 – Percurso da Pesquisa BFS

Fig 5 – Percurso da Pesquisa DFS

**Greedy**

Custo = 26

**A\***

Custo = 24

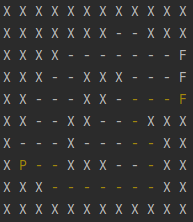
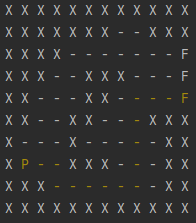
****

Fig 8 – Percurso da Pesquisa A\*

Fig 7 – Percurso da Pesquisa Greedy

# Discussão de Resultados

Como é possível observar pelas figuras 5, 6 , 7 e 8, para este circuito as melhores pesquisas são a BFS e a A\* pois possuem um menor custo final. Também é importante realçar que todas as pesquisas acabaram com uma velocidade final de 17 e que em nenhum momento tiveram uma colisão com os limites de pista.

Através destes resultados, também foi interessante observar que as pesquisas BFS e A\* apesar de possuírem um custo final igual, estas tiveram percursos diferentes.

Também, tal como esperado, conseguimos concluir que a noção geral de melhor ou pior pesquisa não existe, pois essa noção depende do problema proposto. Por exemplo, a pesquisa DFS e Greedy foram as “piores” pesquisas para o Circuito do Mónaco, mas para os circuitos de Melbourne e Suzuki já acabaram com um custo final igual à A\* e à BFS.

# Conclusão

A realização deste projeto permitiu ao grupo consolidar o conhecimento relativo aos protocolos, mecanismos e filosofias da arquitetura de gestão do INMF e dos conceitos principais sobre ameaças de segurança em aplicações e sobre as estratégias que permitem implementar uma maior segurança ao sistema. Também nos permitiu aumentar profundamente o conhecimento sobre o protocolo SNMP, atarvés da observação de como este funciona na prática. É de realçar que a execução de testes experimentais, foi fundamental para uma análise crítica dos resultados obtidos.

O grupo acabou por ter algumas dificuldades na encriptação do canal de comunicação, mais em concreto na transformação de byte para texto limpo mas após um maior estudo sobre este tema foi possível ultrapassar esta dificuldade..

# Autoavaliação

* + - Diogo Cerqueira

Neste projeto estive envolvido no desenvolvimento do código e no desenvolvimento do relatório. A maior dificuldade que encontrei durante esta fase foi na elaboração do código da função BFS.

* + - José Gomes

Neste projeto estive envolvido no desenvolvimento do código e no desenvolvimento do relatório. A maior dificuldade que encontrei foi na implementação dos códigos de pesquisa.

# Referências

[1] Russell and Norvig (2009). Artificial Intelligence - A Modern Approach, 3rd edition,

ISBN-13: 9780136042594;

[2] Costa E., Simões A., (2008), Inteligência Artificial-Fundamentos e Aplicações, FCA,

ISBN: 978-972-722-34

[3] Maria do Carmo Nicoletti, Estevan R. Hruschka, (2017), Fundamentos da Teoria dos Grafos para Computação

Fazer

# Conclusão

De modo a finalizar a fase deste relatório, gostávamos de referir que, apesar das dificuldades encontradas consideramos que conseguimos alcançar um bom resultado, pois alcançámos todos os objetivos propostos.

Este trabalho permitiu-nos compreender melhor o funcionamento dos algoritmos de pesquisa e ajudou-nos muito a melhorar a nossa capacidade de trabalhar com a linguagem de programação Python.