

**Rede de Computadores**

***2º Trabalho Laboratorial***

1. **Aplicação de download**
2. **Configuração e análise de uma rede**

Mestrado Integrado em Engenharia Informática e Computação

André Esteves [up201606673@fe.up.pt](mailto:up201606673@fe.up.pt)

Luís Diogo Silva [up201503730@fe.up.pt](mailto:up201503730@fe.up.pt)

Francisco Friande [up201508213@fe.up.pt](mailto:up201508213@fe.up.pt)

**Índice**

[**Sumário** 2](#_Toc532293065)

[**Introdução** 2](#_Toc532293066)

[**Parte 1 – Aplicação de download** 3](#_Toc532293067)

[**Arquitetura da aplicação** 3](#_Toc532293068)

[**Exemplo de transferência de um ficheiro** 3](#_Toc532293069)

[**Parte 2 – Configuração e análise de uma rede** 3](#_Toc532293070)

[**Experiência 1** 3](#_Toc532293071)

[**Experiência 2** 3](#_Toc532293072)

[**Experiência 3** 3](#_Toc532293073)

[**Experiência 4** 3](#_Toc532293074)

[**Experiência 5** 3](#_Toc532293075)

[**Experiência 6** 3](#_Toc532293076)

[**Experiência 7** 4](#_Toc532293077)

[**Conclusões** 4](#_Toc532293078)

[**Referências** 4](#_Toc532293079)

[**Anexo** 4](#_Toc532293080)

[**Makefile** 4](#_Toc532293081)

# **Sumário**

Este relatório foi elaborado no âmbito da unidade curricular de Rede de Computadores. O trabalho em causa consistia em duas partes. A primeira requer a elaboração de uma aplicação que faz o download de um ficheiro dado o seu URL e a segunda parte consiste na configuração e análise de uma rede no decorrer das aulas práticas sendo ao todo sete experiências.

O trabalho foi realizado no seu todo no ambiente disponibilizado, sendo concluído com sucesso em todos os aspetos, cumprindo os objetivos pedidos.

# **Introdução**

O propósito deste relatório é expor os aspetos mais teóricos da realização do projeto.

……….

O relatório está composto pela seguinte forma:

* Parte 1 – Aplicação para download de um ficheiro:
  + Arquitetura da aplicação.
  + Relatório de transferência de um ficheiro ftp.
* Parte 2 – Configuração e análise de uma rede:
  + Para cada experiência (1 a 7):
    - Arquitetura da rede;
    - Objetivos da experiência;
    - Comandos principais.

# **Parte 1 – Aplicação de download**

A primeira parte do trabalho consistia do desenvolvimento de uma aplicação de linha de comandos em C que fizesse um download de um ficheiro de um servidor FTP, como está descrito em RFC959. O programa recebe do utilizador um URL com o formato ***ftp://[<user>:<password>@]<host>/<url-path>***, como está descrito em RFC1738.

## **Arquitetura da aplicação**

A aplicação começa por fazer interpretar o URL inserido pelo utilizador, fazendo uso da função parseInfo, colocando esta informação numa instância da struct Info:

|  |
| --- |
| struct Info { |
|  | char hostname[100]; |
|  | char path[150]; |
|  | char\* filename; |
|  | char user[50]; |
|  | char password[50]; |
|  | }; |

Caso o utilizador tenha optado por não inserir o username e a password na linha de comandos, como está previsto em RFC1738, estes são obtidos através do input do utilizador, fazendo uso da função getUserInfo.

De seguida, é criada uma instância da struct hostent (definida em netdb.h), na qual é guardado o IP do servidor, fazendo uso da função getHostInfo. Esta informação é usada para criar o socket que vai ser utilizado para comunicar com o utilizador. Isto é feito através da função connectTCP. Depois desta ligação, é lida a resposta do servidor (através da função readResponseCode), de forma a certificarmo-nos que a ligação foi feita com sucesso. De forma a concluir esta etapa inicial de conexão, é enviada ao servidor a informação de login, que está armazenada na struct Info, através da função sendLoginInfo.

Estando estabelecida a ligação, é pedido ao servidor para entrar em modo passivo. Isto é feito através da função getServerPort. Esta função retorna a porta que vai ser utilizada pelo servidor para o envio do ficheiro. Esta a informação é usada para criar um novo canal de comunicação com o servidor, na forma de um outro socket, fazendo uso, novamente, da função connectTCP.

Finalmente, o ficheiro é recebido por parte do utilizador após a chamada à função retrieveFile. Após esta chamada, os canais de comunicação são fechados e a aplicação termina.

**Exemplo de um download bem-sucedido**

Na secção Anexos, estão documentados a compilação da aplicação (Figura 1) e dois downloads bem-sucedidos. No primeiro caso (Figura 2), o utilizador colocou o username e a password no argumento passado na linha de comandos. No segundo caso (Figuras 3 e 4), o utilizador optou por não inserir estes dados na linha de comandos. A figura 3 mostra o programa à espera que o utilizador insira o username. A figura 4 mostra o programa após a sua execução. Na figura 5, é possível ver o ficheiro transferido do diretório onde foi executada a aplicação.

# **Parte 2 – Configuração e análise de uma rede**

## **Experiência 1 – Configurar um IP de rede**

O objetivo desta experiência é ligar o tux 1 ao tux 4 utilizando o switch.

1. O que são os pacotes ARP? Para o que são usados?

O ARP (*Address Resolution Protocol)* é um protocolo de comunicação que serve

1. Quais são os endereços MAC e IP dos pacotes ARP e porquê?
2. Quais os pacotes gerados pelo command ping?
3. Quais são os endereços MAC e IP dos pacotes ping?
4. Como determinar se a trama recetora Ethernet é ARP, IP, ICMP?
5. Como determinar o comprimento de uma trama recetora?
6. O que é a interface *loopback* e porque é importante?

## **Experiência 2 – Implementar duas LAN’s virtuais no switch**

Nesta experiência

1. Como configurar vlany0?
2. Quantos domínios de transmissão existem? O que se pode concluir a partir dos registos?

## **Experiência 3 – Configurar o Router no Linux**

Nesta experiência.

1. Quais rotas existem nos tuxes? Qual é o seu significado?
2. Que informação é que uma entrada da tabela de *forwarding* contém?
3. Quais mensagens ARP e endereços MAC associados que são observados e porquê?
4. Quais pacotes ICMP são observados e porquê?
5. Quais são os IP e MAC endereços associados aos pacotes ICMP e porquê?

## **Experiência 4 – Configurar um Router Comercial e Implementar NAT**

Nesta experiência.

1. Como configurar um router estático num router comercial?
2. Quais são as rotas seguidas pelos pacotes nas experiências anteriormente seguidas e porquê?
3. Como configurar o NAT num router comercial?
4. O que faz o NAT?

## **Experiência 5 – DNS – Domain Name System**

Nesta experiência.

1. Como configurar o serviço DNS num *host*?
2. Que pacotes são trocados pelo DNS e que informações são transportadas?

## **Experiência 6 – Conexões TCP**

Nesta experiência.

1. Quantas conexões TCP são abertas pela aplicação FTP?
2. Em qual conexão é transportado a informação de controlo FTP?
3. Quais são as fases de uma conexão TCP?
4. Como é que o mecanismo ARQ TCP funciona? Quais são os campos TCP relevantes? Qual informação relevante pode ser observada nos *logs*?
5. Como é que o mecanismo de controlo de congestão TCP funciona? Quais são os campos relevantes? Como é que o fluxo da conexão de dados evoluiu ao longo do tempo? Está de acordo com o mecanismo de controlo de congestão TCP?
6. De que forma é afetada a conexão de dados TCP pelo aparecimento de uma segunda conexão TCP? Como?

# **Conclusões**

Em suma, podemos dizer que os objetivos que nos foram propostos foram cumpridos. Foi desenvolvida uma aplicação que é capaz de enviar ficheiros através de uma ligação de porta de série, conseguindo ser resiliente a possíveis erros de comunicação, devido ao protocolo implementado. Os dois níveis que dividem o projeto – o nível de aplicação e o nível de ligação lógica – trabalham de forma completamente independente entre si. Por exemplo, o nível de aplicação não sabe detalhes sobre como é feita a transmissão dos pacotes, como *stuffing e destuffing*, e o nível de ligação não sabe como são interpretados os pacotes que transmite. Este tipo de estruturação foi um dos objetivos de aprendizagem que julgamos ter sido alcançado.

# **Referências**

# **Anexo**

## **Makefile**

|  |  |
| --- | --- |
|  | **all: sender receiver** |
|  |  |
|  | **sender: sender.c** |
|  | **gcc -o sender -Wall sender.c protocol.c llopen.c llwrite.c llclose.c** |
|  | **receiver: receiver.c** |
|  | **gcc -o receiver -Wall receiver.c protocol.c llopen.c llread.c llclose.c -lm** |
|  | **clean :** |
|  | **rm sender receiver \** |