Relatório:

2º Trabalho Prático

Licenciatura em Engenharia Informática e computação

Faculdade de Engenharia da Universidade do Porto

No âmbito da unidade curricular: **Redes de computadores**

Trabalho realizado por:

- José Leandro Rodrigues da Silva – up202008061
- José Pedro Teixeira Ramos - up202005460

Índice

Sumário

Este relatório tem como objetivo documentar o segundo trabalho realizado no 3º ano do curso LEIC em Redes de Computadores. O trabalho pode ser dividido em duas partes, a aplicação de download e as experiências das aulas práticas, sendo que a aplicação de download foi feita em horas extraordinárias.

Introdução

No âmbito da unidade curricular, Redes de Computadores, do 3º ano do curso de Licenciatura em Engenharia Informática e Computação (LEIC) no ano letivo 2022/23, foi realizado um 2º trabalho prático. Este relatório servirá então para descrever, discutir e partilhar o trabalho desenvolvido durante e fora das aulas práticas para este trabalho prático. Este trabalho consiste em 2 partes lógicas, numa primeira parte desenvolveu-se uma aplicação para download de ficheiros usando FTP e na segunda parte realizaram-se várias experiências em laboratório para configurar e analisar uma rede de computadores.

1st Part – Download Application

Description

Nesta parte do trabalho prático foi desenvolvida uma aplicação para download de ficheiros usando FTP (File Transfer Protocol). Para desenvolver esta aplicação utilizou-se a linguagem C e foi necessário um trabalho prévio de preparação para primeiro compreender a fundo como funciona o protocolo FTP e depois como conseguir implementar este protocolo usando conexões a sockets TCP.

Ao desenvolver este trabalho foi definido que este deveria dar alguma liberdade ao utilizador no que toca à sintaxe dos comandos (login); deveria ser resistente a erros e no caso destes acontecerem deveria-se fornecer explicações para a sua ocorrência na linha de comandos; deveria, também, fazer algumas verificações tanto de dados recebidos e dados enviados (IP e número de bytes); por fim as funções construídas deveriam ser o mais gerais possíveis para o futuro reaproveitamento e reutilização do código desenvolvido.

Para utilizar a aplicação, dá-se como argumento um comando com a seguinte formatação:

download ftp://[<user>:<password>@]<host>/<url-path>

Como se pode verificar, o login é opcional, e a sua presença é identificada pela presença do caráter ‘@’ e na sua ausência assume-se as credenciais default:

User: “anonymous” | Pass: “”

Adicionalmente assumiu-se que na omissão que qualquer uma das credenciais é utilizada a credencial default para a credencial em falta.

Para testar a aplicação utilizaram-se os seguintes comandos:

- download ftp://ftp.up.pt/pub/kodi/timestamp.txt
- download ftp://anonymous:@ftp.up.pt/pub/parrot/index.db
- download ftp://rcom:rcom@netlab1.fe.up.pt/files/crab.mp4

Architecture

As funções desenvolvidas/utilizadas para esta aplicação foram as seguintes:

- int parse_arguments(char **args, parameters* params)
- int getIP(char *host, char *host_ip)
- int open_connect_TCP_socket(int *sockfd, char *server_ip, int port)
- int parse_pasv_response(char *response, int *port, char* ip)
- int parse_retr_response(char *response, int *bytes)
- void print_socket_response(int sockfd)
- int get_cmd_response(int sockfd, char* response)
- int send_cmd_to_socket(int sockfd, char *cmd, char *arg, char* response)
- int download_file(int sockfd, char* filename)

Foi também utilizada a struct:

```
typedef struct
{
    char command[MAX_ARG_SIZE];
    char protocol[MAX_ARG_SIZE];
    char username[MAX_ARG_SIZE];
    char password[MAX_ARG_SIZE];
    char host[MAX_ARG_SIZE];
    char url_path[MAX_ARG_SIZE];
    char filename[MAX_ARG_SIZE];
    char ip[MAX_ARG_SIZE];
} parameters;
```

Esta estrutura contém todos os dados necessários para completar o download de um ficheiro e grande parte destes argumentos são dados no comando de argumento

Passo agora a explicar cada um destas funções tendo em conta o workflow da aplicação e a utilidade de cada uma.

Num primeiro momento é necessário dar parse do comando enviado como argumento e para isso é utilizada a função **parse_arguments** que recebe esse comando, guarda os dados relevantes numa struct params e, no caso de erro devolve o valor 1.

Tendo o nome do host, é necessário obter o seu IP address e para o efeito utiliza-se a função **getIP**, que calcula o endereço IP para o host name dado no argumento. Esta função foi reaproveitada do código de exemplo disponibilizado pelo corpo docente.

Agora pode-se estabelecer uma conexão com um socket TCP utilizando o IP calculado e a porta 21 (porta default para FTP). Este socket (control socket) servirá para controlo, ou seja, será o meio para envio de comandos e receção de resposta a esses comandos. Para abrir esta conexão utiliza-se a função **open_connect_TCP_socket** que também foi reaproveitada do código de exemplo disponibilizado pelo corpo docente. Esta devolve um file descriptor.

A função **print_socket_response** é utilizada para imprimir para a linha de comandos a resposta de várias linhas dada inicialmente pelo control socket recentemente aberto.

Agora é necessário enviar comandos para o control socket utilizando a função **send_cmd_to_socket** que recebe duas strings uma com o comando a ser enviado e outra (opcional) com um possível argumento do comando, se não for necessário um argumento, pode ser deixada em branco. Esta função escreve para o control socket o comando e utiliza a função **get_cmd_response** para obter a resposta do socket ao comando enviado e imprimi-la para a linha de comandos. O valor retornado corresponde ao response status code que pode ser 5xx, 4xx, 3xx, 2xx, 1xx dependendo do efeito do comando. A resposta do server é também devolvida através dos parâmetros.

Assim para fazer o login bastam duas chamadas à função **send_cmd_to_socket**:

- send_cmd_to_socket(control_socketfd, "user", params.username, response)
- send_cmd_to_socket(control_socketfd, "pass", params.password, response)

E para entrar em modo passivo utiliza-se:

- send_cmd_to_socket(control_socketfd, "pasv", params.password, response)

A resposta ao comando “pasv” deve ser analisada para obter a porta, que será usada para abrir um novo socket que permita a transferência de dados (data socket). Para o efeito é usada a função **parse_pasv_response** que constrói um array de bytes a partir da resposta onde os primeiros 4 bytes correspondem ao IP do host e os últimos 2 bytes são utilizados para calcular o nº da porta (bytes[4]*256 + bytes[5]). O IP do host é enviado através dos parâmetros para comparar com o IP obtido, dentro da função, e confirmar a ausência de erros. A porta calculada é devolvida através dos parâmetros.

De seguida a porta calculada é utilizada para abrir um nova conexão com o TCP data socket usando uma chamada à função já referida:

- open_connect_TCP_socket(&data_socketfd, params.ip, port)

Tendo agora o data socket aberto falta enviar o comando **retr** com o caminho para o ficheiro como argumento (url_path):

- send_cmd_to_socket(control_socketfd, "retr", params.url_path, response)

A resposta do socket a este comando deve também ser analisada para obter o nº de bytes enviados para posteriormente confirmar a integridade do ficheiro e ausência de erros. Para tal é utilizada a função **parse_retr_response,** esta função analisa a resposta do socket e devolve o nº de bytes enviados.

Por fim, basta utilizar a função **download_file** que lê byte a byte os dados enviados para o data socket e armazena-os num ficheiro no mesmo diretório do executável com o mesmo nome do ficheiro a transferir. Essencialmente, realiza o download do ficheiro desejado. O nº de bytes lidos é contabilizado e devolvido pela função. Se o nº de bytes enviados for diferente dos recebidos, então ocorreu algum erro e por isso esta verificação é realizada.

O workflow de chamadas a funções da aplicação (e algumas verificações de erros) é realizado na função main da aplicação.

2st Part – Configuration and Study of a Computer network

Experiência 1 - Configure an IP network

O objetivo principal desta experiência é configurar as máquinas tux3 e tux4 com IPs e máscaras, de maneira a que possam se comunicar entre elas. Também usamos o wireshark para ver como as duas máquinas se comunicam entre si.

Principais comandos usados:

Linux: ifconfig <carta_de_rede> <endIP>/<mascara>;

Linux: route add -net <destino>/<mask> gw <endDestino>;

Questões:

What are the ARP packets and what are they used for?

ARP ou Address Resolution Protocol é usado para mapear IPs e endereços MAC. Quando um computador tenta enviar um pacote para outro que esteja na mesma rede local, ele vai usar ARP para descobrir qual das outras máquinas tem o IP que ele quer. E quando o ARP chegar ao computador que quer, o destinatário vai enviar outro ARP para a rede que enviou o pedido inicialmente, mas com o seu endereço MAC, para que seja possível fazer a transferência de dados.

What are the MAC and IP addresses of ARP packets and why?

Quando o tux1(IP - 172.16.40.1; MAC - 00:21:5a:61:2f:d4) tenta enviar pacotes para o tux4(IP - 172.16.40.254; MAC - 00:21:5a:5a:7b:ea), ele vai mandar ARPs em broadcast, ou seja para toda a rede local. O ARP vai ter o seguinte, o IP do tux3 IP - 172.16.40.1, o MAC - 00:21:5a:61:2f:d4, o ip do tux4 IP - 172.16.40.254 e o MAC - 00:00:00:00:00:00, o MAC é assim porque ele não sabe o MAC do tux4 ainda. Quando o tux4 receber esse ARP ele vai enviar o ARP, mas com o seu MAC e não vai ser em broadcast Portanto o ARP tem o IP e MAC da máquina que envia e o IP e MAC da máquina que recebe.

What packets does the ping command generate?

O comando ping gera pacotes ARP para saber o endereço MAC e depois gera comandos ICMP echo request para o endereço especificado, no qual depois vai receber uma resposta de echo reply packet.

What are the MAC and IP addresses of the ping packets?

Se fizermos ping do tux3 para o tux4:

Pacote **request**:

- Enderesso IP do emissor: 172.16.40.1
- MAC do emissor: 00:21:5a:61:2f:d4
- Enderesso IP do receptor: 172.16.40.254
- MAC do receptor: 00:21:5a:5a:7b:ea

Pacote **reply**:

- Enderesso IP do emissor: 172.16.40.254
- MAC do emissor: 00:21:5a:5a:7b:ea
- Enderesso IP do receptor: 172.16.40.1
- MAC do receptor: 00:21:5a:61:2f:d4

How to determine if a receiving Ethernet frame is ARP, IP, ICMP?

Se a ethernet header for 0x0806 então a frame é um ARP e se for 0x0800 é um IP. Se no IP header estiver 1, então é um ICMP.

How to determine the length of a receiving frame?

O tamanho da frame pode ser visto no wireshark.

What is the loopback interface and why is it important?

Uma interface de loopback é uma interface virtual que está sempre ativa e acessível, no qual deixa o computador receber respostas de si mesmo. Como resultado, uma interface de loopback é útil para tarefas de debug, pois seu endereço IP sempre pode receber ping.

Experiência 2 - Implement two bridges in a switch

O objetivo desta experiência foi criar duas bridges//Virtual LAN, uma no tux3 e tux4 e outra no tux2. Os principais pontos de destaque nesta experiência foi a criação e configuração das bridges, e como afeta a ligação entre as outras máquinas.

Principais comandos usados:

Linux: ifconfig <carta_de_rede> <endIP>/<mascara>;

GTK: /interface bridge add name=bridgeY0

GTK: /interface bridge port remove [find interface =ether1]

GTK: /interface bridge port add bridge=bridgeY0 interface=ether1

Questões:

How to configure bridgeY0?

Para configurar a brigeY0 primeiro nós temos de criar a bridge, para fazer isto acedemos ao GTK e escrevemos o seguinte comando /interface bridge add name=bridgeY0, de seguida fazemos /interface bridge port remove [find interface =ether1], neste passo é preciso ver a onde o tux3 e tux4 estão ligados no MicroTick, o que este comando faz é eliminar a porta no ether1 para que depois podermos adicionar essa porta à bridge, fazemos isto para a porta do tux3 e tux4. No final adicionamos as portas que eliminamos à bridge com o comando /interface bridge port add bridge=bridgeY0 interface=ether1.

How many broadcast domains are there? How can you conclude it from the logs?

Existem 2 domínios de broadcast, quando fazemos broadcast apenas as portas que pertencem à bridge são abrangidas.

Experiência 3 - Configure a Router in Linux

Nesta experiência o que era pedido era configurar o tux4 para ficar como um router para o tux3 e tux2, desta maneira é esperado que seja possível fazer uma conexão entre o tux3 e o tux2 sendo assim possível a partilha de dados

Principais comandos usados:

Linux: ifconfig <carta_de_rede> <endIP>/<mascara>;

Linux: route add -net <destino>/<mask> gw <endDestino>;

Linux: echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

Linux: echo 0 > /proc/sys/net/ipv4/ip_forward

GTK: /interface bridge port remove [find interface =ether1]

GTK: /interface bridge port add bridge=bridgeY0 interface=ether1

Questões:

What routes are there in the tuxes? What are their meaning?

Todos os tux têm rotas geradas automaticamente, sendo que o tux3 tem rota para 172.16.40.0/24, o tux2 tem para 172.16.41.0/24 e o tux4 tem para o 172.16.40.0/24 e o 172.16.41.0/24 todas as rotas têm gateway igual 0.0.0.0.

O tux3 tem uma rota do 172.16.41.0 com gateway 172.16.40.254, ou seja, sempre que queiramos enviar um pacote para a Virtual Lan 41 ele vai usar a gateway 172.16.40.254 para fazer passagem para a Virtual Lan 41.

O tux2 tem uma rota do 172.16.40.0 com gateway 172.16.41.253, ou seja, sempre que queiramos enviar um pacote para a Virtual Lan 40 ele vai usar a gateway 172.16.41.253 para fazer passagem para a Virtual Lan 40.

O tux4 tem uma rota do 172.16.40.0 com gateway 172.16.41.253, ou seja, sempre que queiramos enviar um pacote para a Virtual Lan 40 ele vai usar a gateway 172.16.41.253 para fazer passagem para a Virtual Lan 40 e tem uma rota do 172.16.41.0 com gateway 172.16.40.254, ou seja, sempre que queiramos enviar um pacote para a Virtual Lan 41 ele vai usar a gateway 172.16.40.254 para fazer passagem para a Virtual Lan 41.

What information does an entry of the forwarding table contain?

A forwarding table contém o destino, gateway, genmask, flags, metric, use e interface.

- O destino é o IP onde queremos chegar.

- A gateway é o IP do computador que vai receber o pacote, e depois vai mandá-lo para o IP do destino.
- A flag é informação sobre a rota.
- Metric é o custo da rota.
- Use é o contador de usos da rota.
- Interface é a carta de rede (eth0).

What ARP messages, and associated MAC addresses, are observed and why?

Quando queremos fazer uma troca de pacotes entre o tux3 e o tux2, o que vai acontecer é o seguinte, o tux3 não sabe o MAC do tux2, portanto vai enviar um ARP em broadcast com o seu IP, MAC e IP do tux2, sendo que em vez de ter o MAC do tux2 vai ter 00.00.00.00.00.00, quando o tux2 receber esta mensagem, ele vai enviar outra igual mas com o seu MAC em vez de 00.00.00.00.00.00 e não vai ser em broadcast mas sim diretamente para o tux3. Isto acontece sempre que tentamos estabelecer ligação entre 2 computadores, porque o computador nunca sabe o MAC da rede de destino.

What ICMP packets are observed and why?

Os pacotes ICMP que podemos observar são, ICMP request e ICMP reply. Já não aparece ICMP host unreachable porque temos rotas para todas as redes.

What are the IP and MAC addresses associated to ICMP packets and why?

Os IPs e MACs associados com fazemos envio de pacotes na mesma interface, por exemplo do tux3 eth0 com o tux4 eth0, os IPs vão ser do tux3 e tux4 eth0 e os MACs vão ser do tux3 e tux4 eth0.

Quando enviamos pacotes para interfaces diferentes, por exemplo do tux3 para o tux2, o que vai acontecer é o seguinte, o tux3 vai enviar um pedido para o tux4 eth0, os IPs vão ser do tux3 e tux4 eth0 e os MACs vão ser do tux3 e tux4 eth0, depois o tux4 vai enviar um pedido para o tux2, os IPs vão ser do tux3 e tux2 e os MACs vão ser do tux4 eth1 e tux2.

Experiência 4 - Configure a Commercial Router & Implement NAT

Como nas experiências anteriores só conseguimos fazer ligações dentro da lan, agora queremos ter acesso à internet com um router comercial, para fazer isto vai ser preciso configurar o router com o NAT. Esta experiência teve o objetivo de perceber os redirecionamentos de pacotes ICMP e a influência da NAT na conexão à internet.

Principais comandos usados:

Linux: route add -net <destino>/<mask> gw <endDestino>;

Linux: route add -net default gw <endDestino>;

Linux: echo 0 > /proc/sys/net/ipv4/conf/eth0/accept_redirects

Linux: echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects

GTK: /interface bridge port remove [find interface =ether1]

GTK: /interface bridge port add bridge=bridgeY0 interface=ether1

GTK: /ip address add address=<destino>/<mask> interface=ether1

GTK: /ip firewall nat disable 0

Questões:

How to configure a static route in a commercial router?

Primeiro precisamos ligar a bancada à entrada de configuração do router. Depois no GTK fazemos /ip route add dst-address=<destino><mascara> gateway=<gateway>

What are the paths followed by the packets in the experiments carried out and why?

Antes de chegarmos ao exercício 4 o caminho usado pelos pacotes é o caminho normal, isto é, se tiver uma rota direta ele vai por esse caminho, caso contrário é dirigido ao router por rotas default. No exercício 4 desativamos os redirects e o RC é a default do tux2 e tux4. Quando fazemos ping do tux2 para o tux3, como o tux4 é o único que consegue comunicar com as duas interfaces e o tux2 não tem caminho direto para o tux4, ele vai usar o default gateway para conectar-se ao RC, depois o RC conecta-se ao tux4 e por final o tux4 conecta-se ao tux3.

How to configure NAT in a commercial router?

Para configurar o NAT é preciso primeiro de tudo configurar a porta série no switch:

GTK: /interface bridge port remove [find interface =ether1]

GTK: /interface bridge port add bridge=bridgeY1 interface=ether1

Depois para configurar o router é preciso conectar o configurador do router e fazer a configuração do ip e das rotas:

/ip address add address=172.16.1.Y9/24 interface=ether2

/ip address add address=172.16.1.254/24 interface=ether1

/ip route add dst-address=0.0.0.0/0 gateway=172.16.1.254

/ip route add dst-address=172.16.40.0/24 gateway=172.16.41.253

What does NAT do?

NAT ou Network Address Translation é uma maneira de mapear múltiplos endereços privados para públicos antes de começar a fazer transferência de pacotes, ou seja, temos muitos computadores ligados a usarem o mesmo IP público, mas os IP privados continuam iguais. Se nós não usarmos o NAT, não vamos conseguir fazer uma ligação à internet, porque o servidor no qual estamos a tentar contactar não vai saber para onde mandar os pacotes de resposta, pela simples razão de estarmos a usar um endereço privado em vez de um público. Adicionalmente também ajuda-nos a proteger e ter privacidade na nossa máquina, porque impede fontes externas de acederem aos endereços privados.

Experiência 5 - DNS

O objetivo desta experiência foi de configurar um DNS (Domain name system) na rede construída até ao momento. Este sistema vai permitir a qualquer host da rede obter endereços IP para um determinado host name indicado, já que um DNS permite a tradução entre estes argumentos. Assim qualquer host da nossa rede vai poder comunicar com hosts externos.

Principais comandos usados:

Para configurar o DNS apenas foi necessário editar entradas no ficheiro resolv.conf presente no diretório /etc de cada um dos tuxes (tux2, tux3, tux4).

Questões:

How to configure the DNS service at an host?

Para configurar o DNS em cada host é necessário aceder ao ficheiro resolv.conf presente no diretório /etc da máquina em questão, limpar as entradas lá presentes e adicionar a entrada com o endereço IP do servidor DNS do laboratório onde se encontra a nossa bancada. No nosso caso a entrada a adicionar foi:

- nameserver 172.16.1.1

What packets are exchanged by DNS and what information is transported

Cada host da rede envia para o DNS um pacote com um hostname e o servidor DNS do laboratório responde com um pacote contendo o endereço IP para o hostname identificado.

Experiência 6 - TCP Connections

Para esta experiência o objetivo foi apenas a captura, observação (usando o WireShark) e análise dos pacotes relacionados com as conexões TCP estabelecidas durante a execução da aplicação de download desenvolvida na primeira parte deste trabalho.

Principais comandos usados:

Os comandos usados foram apenas os de compilação e execução da aplicação de download, possíveis comandos de execução já foram explicitados na primeira parte.

Questões:

How many TCP connections are opened by your ftp application?

A aplicação de download desenvolvida durante a sua execução abre duas conexões TCP, uma para comunicar informações de controlo e outra para a transferência de dados. Daí serem utilizados dois socket (control socket e data socket).

In what connection is transported the FTP control information?

A conexão TCP utilizada para envio e receção de informação de controlo relacionada com comandos para login, entrada em modo passivo e para transferência de ficheiros é a conexão de controlo, representada na aplicação pelo control socket.

What are the phases of a TCP connection?

Numa conexão TCP podem ser identificadas 3 fases:

- Estabelecimento da ligação: Ambas aplicações sincronizam-se e a transferência de dados é preparada.
- Transferência de dados: Os dados são divididos em vários pacotes ordenados e são enviados um a um.
- Terminação da ligação: Confirma-se o término da transmissão e a ligação é fechada.

How does the ARQ TCP mechanism work? What are the relevant TCP fields? What relevant information can be observed in the logs?

ARQ ou (Automatic Repeat reQuest) é utilizado pelo protocolo TCP para confirmar que a transmissão de pacotes ocorreu sem erros. Para isso, o recetor envia uma mensagem ACK (acknowledgement) para cada pacote recebido corretamente. No caso do TCP, é utilizada uma “sliding window” onde, caso um ACK não seja recebido, dentro de um intervalo de tempo (timeout) o respetivo pacote, inserido na janela definida, é reenviado. Nos logs pode-se verificar que nas mensagens ACK do TCP são identificados os campos:

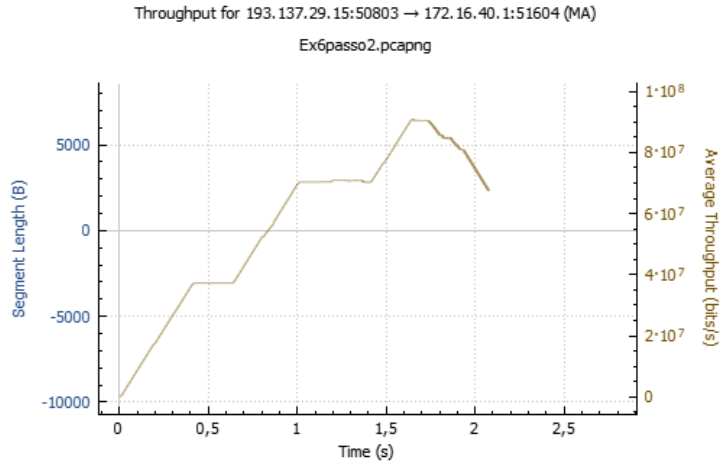
- Sequence number (Seq): número do pacote a ser enviado
- ACK number (Ack): número de sequência do próximo pacote esperado
- Window size (Win): tamanho da congestion window
- Outros campos (timestamps, len, ...)

How does the TCP congestion control mechanism work? What are the relevant fields. How did the throughput of the data connection evolve along the time? Is it according the TCP congestion control mechanism?

A técnica de controlo de congestão do TCP utiliza uma janela de congestão (congestion window) que é uma variação do método “sliding window” que tem em conta a congestão da conexão. Esta técnica utiliza alguns mecanismos tipo:

- Slow Start: Inicialmente, o tamanho da window aumenta lentamente.
- Additive increase/multiplicative decrease: Sempre que ocorrem perdas o tamanho da window reduz para metade. E quando não ocorrem perdas a window aumenta de tamanho por um MSS (Max Segment Size).
- Fast retransmit: Sempre que ocorrem perdas os pacotes são retransmitidos rapidamente, pois o tempo de espera do sender é reduzido

Nos Logs da experiência 6 (Ex6passo2.pcapng), após a conexão ser estabelecida pode-se verificar um aumento gradual do throughput quando não ocorrem perdas (exemplo: linhas nº 107, 109, 111, 113, …) e uma quebra quando ocorrem.



Tendo em conta a fórmula de throughput, este vai aumentando com o tempo, pois a window também aumenta. Analisando os logs verifica-se uma conformidade com a técnica de controlo de congestão e os outros mecanismos descritos.

Is the throughput of a TCP data connections disturbed by the appearance of a second TCP connection? How?

A taxa de transferência da conexão inicial é afetada, pois TCP utiliza um princípio de fairness (Max-Min), onde todas as conexões concorrentes devem ter acesso a partes iguais de “bottleneck”, ou seja, a bandwidth é distribuída de forma igual a todos os flows. Assim, com o aparecimento da segunda conexão, a primeira conexão deixa de ter acesso a toda a bandwidth e distribui-a com a nova conexão. Por isso o throughput da 1ª conexão deve baixar após o estabelecimento da 2ª conexão.

Conclusão

Com este trabalho prático foram obtidas competências e conhecimentos ao nível da construção e desenvolvimento de uma rede de computadores, compreensão do funcionamento de conexões TCP, experiência adquirida no uso de sockets associados a conexões TCP usando a linguagem C e familiaridade com aplicações FTP. A realização das várias experiências, cada uma focada em tópicos diferentes, permitiu e exigiu a obtenção de uma área alargada de conhecimentos de redes, tipo IP, DNS, TCP connections, bridges, ports, comandos comuns, packets, WireShark, protocolos,...

Ambos os autores encontram-se satisfeitos com o trabalho desenvolvido, pois ao longo da realização deste trabalho procurou-se aplicar uma metodologia de melhoria constante, incremental e de correção de erros, o que resultou, dentro do possível, num trabalho organizado, rigoroso e pedagógico.

Referências

- https://www.emnify.com/iot-glossary/transmission-control-protocol
- https://www.geeksforgeeks.org/tcp-fairness-measures/
- https://en.wikipedia.org/wiki/TCP_congestion_control
- https://www.cs.utexas.edu/users/lam/396m/slides/Sliding_window+congestion_control.pdf
- https://www.comptia.org/content/guides/what-is-network-address-translation
- https://www.ibm.com/docs/en/zos-basic-skills?topic=layer-address-resolution-protocol-arp
- https://www.techtarget.com/searchnetworking/definition/ping
- https://techhub.hpe.com/eginfolib/networking/docs/switches/common/15-18/5998-8158_bog/content/ch06s03.html

Slides das Teóricas

Anexos

Pode-se encontrar o código-fonte da aplicação de download, logs das várias experiências e um documento com alguns dos comandos utilizados nas experiências no seguinte repositório:

- https://github.com/Gambuzin0/trab2_RC