Trabalho Prático Nº.1 – Protocolos da Camada de Transporte

Duração: 2 aulas

Este trabalho deve ser realizado com recurso à máquina virtual **XubunCORE_7_5** que está disponibilizada em http://marco.uminho.pt/ferramentas/CORE/xubuncore.html (user: *core* password: *core*)

Nota: *Não siga o guião cegamente, com copy/paste dos comandos, sem se interrogar do que está a fazer! Tente perceber os comandos, e o que é suposto fazerem, antes de os fazer! Não perca os objetivos de vista!*

RELATÓRIO

O relatório do TP1 deve incluir uma página de identificação (UC, ano letivo, identificação do Grupo e dos alunos do grupo, identificação do trabalho e data de submissão), um índice do conteúdo e uma secção com as **respostas diretas às questões incluídas na Parte B: Questões**, incluindo sempre os dados que recolheu durante a realização do trabalho e que justificam as respostas dadas.

O relatório, escrito em formato livre, deve ser submetido na plataforma de ensino da universidade, usando a funcionalidade de transferência de ficheiros do grupo, com o nome **CC-TP1-G<Turno>.*<Grupo>*.pdf** (por exemplo, CC-TP1-G1.03.pdf para o grupo 03 do turno PL1) no final do dia da aula prevista para a conclusão do trabalho (verifique as datas exatas nas regras de funcionamento das aulas práticas na área de "Informações" ou "Conteúdo" na plataforma de ensino da universidade dedicada à UC).

PARTE A: Instalação, configuração e utilização de serviços de transferência de ficheiros

Recomenda-se que utilize a topologia `CC-Topo-2022-2023.imn` que pode descarregar da plataforma de *e-learning* e que siga os seguintes passos:

- **[Xubuncore Linux]:** verificar se o software servidor está instalado; instalar se necessário;
- **[Xubuncore Linux]:** preparar uma pasta com os ficheiros a transferir; um ficheiro de texto e um ficheiro binário;
- **[Xubuncore Linux]:** executar o core com a topologia virtual;
- **[Topologia virtual, Servidor1]:** configurar servidores de modo a darem acesso a essa pasta;
- **[Topologia virtual, Portatil1]:** usar o software cliente respetivo para transferir os ficheiros disponibilizados.

Verificar se o software (cliente e servidor) está instalado e instalar se necessário. *[Máquina XubunCORE Linux (host principal), usando linha de comando (bash), user core, password core]*	
Comandos	Observações
`$ sudo apt install openssh-server` `$ sudo apt install openssh-client`	O software SSH já deve estar instalado de raiz no Linux e o serviço SSH/SFTP já está configurado e ativo por omissão em todas as topologias virtuais criadas pelo CORE; Não deverá ser necessário fazer nada de especial para usar SSH.
`$ sudo apt install vsftpd`	Existem vários packages de software FTP para Linux. Neste exercício sugere-se a utilização do "vsftpd". O cliente ftp já existe no Linux e não é preciso instalar.
`$ sudo apt install atftpd` `$ sudo apt install atftp`	Para software servidor TFTP propõe-se o uso do "atftpd", que é um servidor tftp avançado e também do respetivo cliente "atftp". Não existe nenhum servidor ou cliente pré-instalado.

`$ sudo apt install mini-httpd`
`$ sudo apt install wget`
`$ sudo apt install lynx`
	O software "mini-httpd" foi escolhido por ser um servidor web simples e que usa poucos recursos. Já o "lynx" e o "wget" são clientes Web, para consola, muito usados e poderosos!

`$ sudo dpkg --status openssh-server`
`$ sudo dpkg --status vsftpd`
`$ sudo dpkg --status atftpd`
`$ sudo dpkg --status atftp`
`$ sudo dpkg --status mini-httpd`
…
	Opcional. O comando "*dpkg*" é o gestor de pacotes Debian. Neste caso está a ser usado para verificar o estado dos pacotes que instalámos previamente.

Uma vez instalado o software necessário, e ainda antes de iniciar o core com a topologia virtual, sugere-se que prepare uma única pasta com dois ficheiros (um ficheiro de texto e um ficheiro binário) para serem disponibilizados pelos vários servidores (SFTP, FTP, TFTP e HTTP). **Relembra-se que a pasta estará acessível e visível em todos os nós da topologia virtual, pois todos partilham exatamente o mesmo sistema de ficheiros!**

Preparar uma pasta com os ficheiros a transferir; um ficheiro de texto e um ficheiro binário. *[Máquina XubunCORELinux (host principal), usando linha de comando (bash), user core, password core]*	
Comandos	Observações

`$ sudo mkdir -p /srv/ftp`
 	
`$ sudo usermod -d /srv/ftp ftp`
 	
`$ sudo cp /etc/hosts /srv/ftp/file1`
 	
`$ sudo cp /bin/ls /srv/ftp/file2`
	O servidor FTP instala um novo utilizador no sistema com *username* "ftp" sem password para poder servir ficheiros da *home* desse utilizador de forma anónima a qualquer cliente FTP. A pasta a criar chama-se *"/srv/ftp".* O comando *mkdir* criará a pasta se ela não existir (e todas as incluídas no path que forem necessárias – opção "-p"). O comando *usermod* faz dela a "*home*" do user "*ftp*".
	Depois são copiados para lá dois ficheiros: o *"/etc/hosts"* que é um ficheiro de texto pequeno e que vai ser o "*file1*" e o ficheiro executável *"/bin/ls"* que será o ficheiro binário (executável) "*file2*". Pode optar por colocar ou editar outros ficheiros nessa pasta. Tudo o que estiver lá ficará acessível.

Emular a topologia do ficheiro `CC-Topo-2022-2023.imn` no *core.*	
Comandos	Observações
`$ wget http://marco.uminho.pt/disciplinas/CC-LEI/CC-Topo-2022.imn`	Obtem o ficheiro "`CC-Topo-2022.imn`" da página da disciplina com o comando de linha "*wget*".
`$ sudo systemctl start core-daemon.service` `$ sudo core-gui CC-Topo-2022.imn`	Executa o *core* com a referida topologia. O daemon já deve estar previamente ativado, no entanto o primeiro comando inicia o *daemon* do *core.* O segundo comando lança a interface gráfica (cliente *core-gui*) com poderes de *root*.
Topologia virtual, `Servidor1`, botão do rato do lado direito, *Shell Windows → bash*.	Obter uma *bash shell* no nó `Servidor1`
Topologia virtual, `Portatil1`, botão do rato do lado direito, *Shell Windows → bash*.	Obter uma *bash shell* no nó `Portatil1`
Topologia virtual, `Grilo`, botão do rato do lado direito, *Shell Windows → bash*.	Obter uma *bash shell* no nó `Grilo`
Topologia virtual, `Router1`, botão do rato do lado direito, *Wireshark → Eth2*.	Lançar um processo *Wireshark* no `Router1` para capturar todos os pacotes que passam pelo seu interface `eth2` (certifique-se de que esse é o local certo para a captura!).

A.1 PING

Verificar conetividade dos nós **Portatil1 (LAN1)** e **Grilo (LAN4)**

```
root@Portatil1$ ping -c 20 10.2.2.1 | tee file-ping-output
```
Nota: … o resultado irá ficar também armazenado em "file-ping-output" …

```
root@Portatil1$ less file-ping-output
…
```

(Repetir para o nó Grilo)

Observe e analise as perdas e duplicações de pacotes.

A.2 SFTP

Verificar se o servidor SSH no nó **Servidor1 (LAN2)** está ativo na **porta 22**

```
root@Servidor1$ ps -ef | grep ssh
```
… verificar se existe um processo sshd em execução…

```
root@Servidor1$ netstat -n -a
```
… verificar se está alguém à escuta na porta 22…

Transferir por SFTP o ficheiro **file1** do **Servidor1 (LAN2)** para o **Portatil1 (LAN1)** e **Grilo (LAN4)**, a partir destes.

```
root@Portatil1$ rm /root/.ssh/known_hosts
```
Nota: … a identidade dos nós da topologia virtual está sempre a mudar… pelo que é preciso apagar a lista de hosts bem conhecidos do SSH

```
root@Portatil1$ sftp core@10.2.2.1
```
Nota: … não se esqueça que a password do utilizador "core" usado no exemplo é "core" … se usar outro user terá que usar outra password…

```
sftp> pwd
sftp> cd /srv/ftp
sftp> dir
sftp> get file1
sftp> quit
…
```

(Repetir para o nó Grilo)

Observe e analise as diferenças entre as duas transferências.

A.3 FTP

Configurar e arrancar manualmente o servidor FTP no **Servidor1 (LAN2)**

```
root@Servidor1$ chmod a-w /srv/ftp
```
... directoria não pode ter acesso para escrita... por questões de segurança...

```
root@Servidor1$ vsftpd /etc/vsftpd.conf -osecure_chroot_dir=/srv/ftp -oanonymous_enable=YES
```
... este comando vai manter-se em execução no terminal sem passar para background...
... pode ser enviado para background com um "control-Z", que suspende o processo, seguido do comando "bg", que o envia para background
... para mais informações sobre os parâmetros escrever "man vsftpd" ou "man vsftpd.conf" num terminal

Transferir por FTP o ficheiro **file1** do **Servidor1 (LAN2)** para o **Portatil1 (LAN1)** e **Grilo (LAN4)**, a partir destes.

```
root@Portatil1$ ftp 10.2.2.1
```
... entrar com username anonymous e qualquer password (aconselha-se o e-mail)
... se não conseguir aceder de forma anónima utilize o par username/password por defeito, i.e., "core"/"core"
```
ftp> status
ftp> pwd
ftp> dir
ftp> get file1
ftp> quit
…

(Repetir para o nó Grilo)
```

Observe e analise as diferenças entre as duas transferências.

A.4 TFTP

Configurar e arrancar manualmente o servidor TFTP no **Servidor1 (LAN2)**

```
root@Servidor1$ chmod –R 777 /srv/ftp
```
... directoria tem de ter acesso para escrita para todos...
```
root@Servidor1$ touch atftpd.log
```
... se houver problemas podemos ver neste ficheiro de log o que se passou...
```
root@Servidor1$ atftpd --verbose=3 --user root.ftp --logfile atftpd.log \
                       --bind-address 10.2.2.1 --daemon --no-fork /srv/ftp/
```
... atenção que a barra \ serve para continuar o comando noutra linha e não é necessária se escrever tudo na mesma linha ...
... este comando vai manter-se em execução no terminal sem passar para background...
... pode ser enviado para background com um "control-Z", que suspende o processo, seguido do comando "bg", que o envia para background
... para mais informações sobre os parâmetros escrever "man atftpd" num terminal

Transferir por TFTP o ficheiro **file1** do **Servidor1 (LAN2)** para o **Portatil1 (LAN1)** e **Grilo (LAN4)**, a partir destes.

```
root@Portatil1$ atftp 10.2.2.1
ftp> status
ftp> get file1
ftp> quit
…

(Repetir para o nó Grilo)
```

Observe e analise as diferenças entre as duas transferências.

A.5 HTTP

Configurar e arrancar manualmente o servidor HTTP no **Servidor1 (LAN2)**

```
root@Servidor1$ mini_httpd -d /srv/ftp/
…
root@Servidor1$ ps –ef
```
… para verificar se o daemon ficou em execução…

Transferir por HTTP os ficheiros **file1** e **file2** do **Servidor1 (LAN2)** para o **Portatil1 (LAN1)** e **Grilo (LAN4)**, a partir destes. Pode usar-se o comendo **wget** ou o comando **lynx**, pois ambos utilizam o HTTP como transporte de transporte.

```
root@Portatil1$ wget http://10.2.2.1/file1
root@Portatil1$ wget http://10.2.2.1/file2
…

(Repetir para o nó Grilo)
```

Observe e analise as diferenças entre as duas transferências.

A.6 Uso da camada de transporte por parte das aplicações

Numa linha de comando da máquina virtual **XubunCORE**, mas fora do emulador, sem ativar o **core-gui**, execute:

```
$ sudo Wireshark
```

Capture o tráfego nos instantes que considere adequados, observando atentamente como as várias aplicações utilizam os serviços da camada inferior…

- ✓ Transfira (*download*) ficheiros via HTTP (*wget* ou *lynx*) do site **speedtest.tele2.net**
- ✓ Transfira (*upload* & *download*) ficheiros utilizando o servidor FTP do mesmo site
- ✓ Tente aceder via *telnet* ao serviço **rainmaker.wunderground.com**
- ✓ Explore o DNS usando o comando *nslookup* **www.uminho.pt**
- ✓ Teste a conetividade para **www.google.pt** com o *ping*
- ✓ Teste a conetividade para **www.fccn.pt** com o *traceroute*

…

(Use outras aplicações que conheça e que possa explorar)

Observe e analise as diferenças entre as duas transferências.

PARTE B: Questões

1. De que forma as perdas e duplicações de pacotes afetaram o desempenho das aplicações? Que camada lidou com esses problemas: transporte ou aplicação? Responda com base nas experiências feitas e nos resultados observados.
2. Obtenha a partir do *Wireshark*, ou desenhe manualmente, um diagrama temporal para a transferência do ficheiro *file1* por FTP realizada em A.3. Foque-se apenas na transferência de dados [ftp-data] e não na conexão de controlo (o FTP usa mais que uma conexão em simultâneo). Identifique, se aplicável, as fases de início de conexão, transferência de dados e fim de conexão. Identifique também os tipos de segmentos trocados e os números de sequência usados tanto nos dados como nas confirmações.
3. Obtenha a partir do *Wireshark*, ou desenhe manualmente, um diagrama temporal para a transferência do ficheiro *file1* por TFTP realizada em A.4. Identifique, se aplicável, as fases de início de conexão, transferência de dados e fim de conexão. Identifique também os tipos de segmentos trocados e os números de sequência usados tanto nos dados como nas confirmações.
4. Compare sucintamente as quatro aplicações de transferência de ficheiros que usou, tendo em consideração os seguintes aspetos: (i) identificação da camada de transporte; (ii) eficiência; (iii) complexidade; (iv) segurança.
5. Com base no trabalho realizado, construa uma tabela informativa identificando, para cada aplicação executada (*ping, traceroute, telnet, ftp, tftp, wget/lynx, nslookup, ssh,* etc.), qual o protocolo de aplicação, o protocolo de transporte, a porta de atendimento e o *overhead* de transporte.

ANEXO I: Emulador de Rede "CORE"

Para além de compreender os conceitos teóricos subjacentes, um engenheiro informático deve ser capaz de conceber topologias de rede, configurar os diversos equipamentos que a integram e verificar o seu correto funcionamento. Uma outra componente importante é o desenvolvimento de programas que integrem os diversos componentes nos vários níveis da pilha protocolar.

Estas competências, para poderem ser adquiridas, dependem da existência de Laboratórios onde existam os diversos tipos de equipamento que podem integrar uma rede tais como *routers*, *switches*, *hubs*, *hosts* (servidores e clientes) através de diferentes formas de interligação.

Para alguns trabalhos, a exemplificação de determinadas técnicas, protocolos ou tecnologias torna indispensável o uso de uma topologia de rede de determinada dimensão. Atendendo à rápida evolução da área de estudo, o uso de simuladores de redes para investigação e para ensino é prática comum. Todavia, num simulador normalmente utiliza-se um modelo simplificado da realidade e os conceitos e a experiência prática acumulada não podem ser transportados para os equipamentos reais com muita facilidade.

Com o desenvolvimento do conceito de virtualização e a implementação de máquinas virtuais com as mesmas potencialidades dos sistemas de computação, dos sistemas operativos mais populares e dos próprios equipamentos de rede deu-se um passo decisivo para a emulação de redes.

Um dos problemas tradicionais com a virtualização é a falta de escalabilidade, particularmente visível quando o número de máquinas virtuais aumenta. O problema complica-se mais quando se pretende ter uma topologia de rede com um conjunto heterogéneo de máquinas virtuais. Este problema pode ser eventualmente minimizado através de uma implementação leve, isto é, mais simplificada, da máquina virtual.

Neste contexto, apresenta-se como plataforma de apoio às aulas a plataforma CORE - *The Common Open Research Emulator*. O CORE proporciona um ambiente de emulação *open-source* que usa técnicas existentes de virtualização de sistemas operativos para construir redes com fios e sem fios onde os protocolos e as aplicações podem correr sem qualquer modificação. Sendo um emulador em tempo real, essas redes podem estar conectadas a redes reais estendendo os ambientes de teste a equipamento real. A arquitetura CORE inclui uma GUI para desenhar facilmente topologias de rede, uma camada de serviço para instanciar máquinas virtuais e uma API para orquestrar tudo isto. As topologias podem ser também criadas através de *scripts* em Python. O CORE permite trabalhar com topologias de múltiplos nós, contudo o número de nós virtuais suportado depende da capacidade computacional do sistema em questão.

Assim, um emulador de redes como o CORE permite emular a topologia de uma rede num computador pessoal. Desta forma, pode-se estudar o funcionamento da rede e correr nos diversos nós da topologia o mesmo software que corre na rede real. Esta situação permite que a experiência acumulada possa ser transportada com facilidade para redes com equipamentos de rede reais.

O CORE é bastante flexível, escalável e simples de usar, permitindo também correr "código real" nos equipamentos definidos.

A plataforma CORE inclui uma interface gráfica que permite definir com facilidade a topologia de rede e configurar os equipamentos ativos da rede (*routers*, *switches*, *access points*, etc.). Esta plataforma foi desenvolvida por um grupo de técnicos da Boeing, usando como ponto de partida uma ferramenta (*IMUNES*) desenvolvida na Universidade de Zagreb. Utiliza ou FreeBSD ou Ubuntu como sistema base e encaminhamento baseado nos sistemas *Quagga/Zebra* para os routers virtuais.

Instalação e Experimentação

Verifique se tem o CORE instalado no seu computador. Caso isso não aconteça pode utilizar a máquina virtual Linux disponibilizada para o efeito no seguinte link:

http://marco.uminho.pt/ferramentas/CORE/xubuncore.html

Após a instalação, teste o seu funcionamento abrindo um terminal e executando o comando **core-gui** na linha de comandos.

Seguem-se algumas perguntas que pode tentar responder (ou tentar encontrar as respostas) como forma de consolidação dos conhecimentos sobre esta plataforma:

1- Relativamente aos menus da GUI do CORE, verifique (se necessário, recorra ao manual do CORE):

   a. Que tipo de equipamentos se podem interligar em topologias de rede?

   b. Quais as diferenças básicas entre um *hub* e um *switch*?

   c. Quais as diferenças entre um *switch* e um *router*?

   d. Quais são as possibilidades de configuração nos *links* de ligação física?

2- Defina em modo de edição uma topologia com quatro *routers* como nós. Faça uma ligação do nó 1 para o nó 2, deste para o nó 3, e deste para o nó 4.

   a. Verifique que são atribuídos automaticamente endereços de rede IPv4 e IPv6 aos vários nós...

   b. Qual o débito nominal das ligações que interligam os nós? Coloque esse débito a 10 Mbps.

   c. Passe para o modo de emulação. Faça um clique duplo no nó 1 e verifique que processos estão em execução...

   d. O *ping* é um utilitário que permite testar a conectividade IP entre dois nós na topologia de rede. Existe conectividade entre o nó 1 e o nó 4?

3- A partir do modo de edição considere modificar a topologia de rede da seguinte forma: (i) remova a ligação entre o nó 2 e o nó 3 e entre este e o nó 4; (ii) insira um hub; (iii) ligue o nó 2, o nó 3 e o nó 4 a esse hub; (iv) ligue um servidor ao nó 1 e (v) dois clientes ao nó 4.

   a. Verifique as alterações ocorridas a nível de endereçamento quando introduziu o *hub* na topologia da rede...

   b. Teste a conectividade entre o nó 7 e o servidor...

   c. Estabeleça uma ligação *ssh* ao servidor a partir do nó 7...

Instalação em máquinas Windows 10

Verifique o guia "Tutorial - Instalar CORE no WSL" disponibilizado na área de conteúdo do BB da UC para realizar uma instalação do Core em ambiente windows subsistem for Linux.

Além disso, também pode aceder a uma imagem *docker* com a última versão do CORE instalado (~245Mb), para poder correr o emulador em diversas arquiteturas (o arquivo contem um ficheiro README com instruções):

https://uminho365-my.sharepoint.com/:u:/g/personal/id8321_uminho_pt/EePAcHBJE0lKuNLP41XsuDkBr1kgQ002GWgWkW0VaNUvTA

Referências

- J. Ahrenholz, *Comparison of CORE Network Emulation Platforms*, Proceedings of IEEE MILCOM Conference, 2010.
- Manual do CORE, https://coreemu.github.io/core/

ANEXO II: Analisador de Tráfego "Wireshark"

Como complemento ao emulador de redes CORE, pode ser usada também uma ferramenta que permite a captura e a análise de tráfego que circula nas redes emuladas (ou em redes reais), vulgarmente designada de Analisador de Tráfego (ou s*niffer*). Um *sniffer* de rede permite analisar em deferido tráfego capturado (anteriormente) em tempo real, não gerando tráfego adicional para a rede.

O *Wireshark* é uma ferramenta para observação das mensagens trocadas entre duas entidades protocolares. Permite capturar os pacotes que circulam na rede enviados e recebidos através da *interface* de rede do seu computador pessoal.

Como ilustrado na Figura 1, um *sniffer* é um software que implementa dois processos principais, a captura e a análise de unidades protocolares de dados, ou *Protocol Data Units* (PDUs). A biblioteca de captura obtém uma cópia de todos os PDUs (a partir do nível dois – nível de ligação) que são enviados e recebidos numa dada interface de rede.

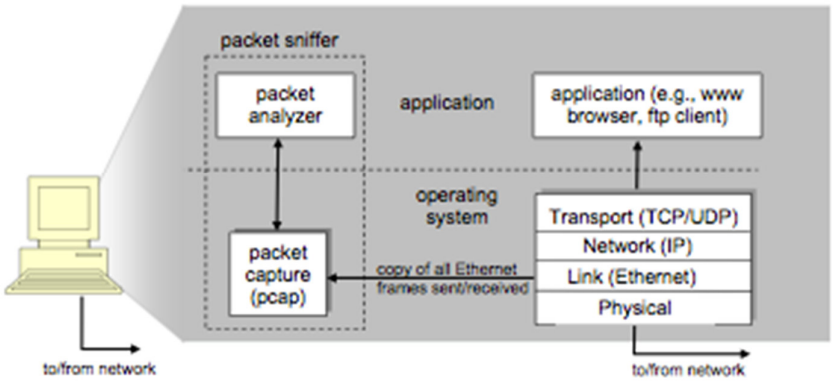

Figura 1: Estrutura típica de um *sniffer*

A comunicação entre sistemas remotos traduz-se na troca de mensagens geradas por protocolos de alto nível (normalmente associados a aplicações ou serviços dos utilizadores), que sofrem um processo de encapsulamento protocolar até ao nível da tecnologia de ligação disponível. O conceito de encapsulamento protocolar, explicado nas aulas teóricas, é ilustrado na Figura 2.

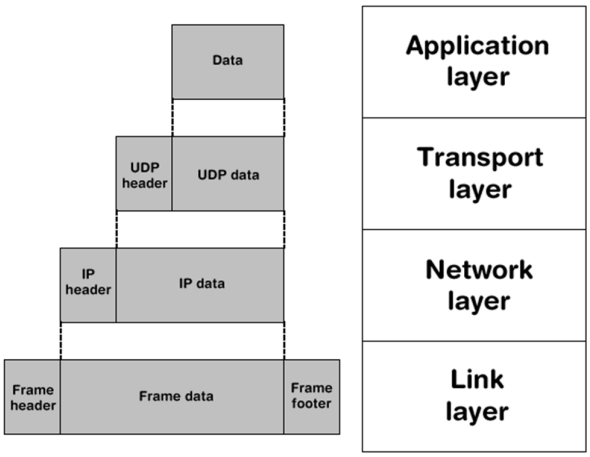

Figura 2: Encapsulamento protocolar

O segundo componente dum *sniffer* é um analisador de PDUs que mostra o conteúdo de todos os campos duma mensagem protocolar. Para esse efeito, o analisador de protocolos tem que compreender a estrutura das mensagens trocadas pelas entidades protocolares. Por exemplo, na troca de informação entre um cliente e um servidor Web, se os PDUs trocados forem capturados, o analisador de tráfego tem de perceber o formato da trama (*frame)* Ethernet (camada 2), do datagrama IP (camada 3) dentro da trama, do segmento TCP (camada 4) dentro do datagrama e, por fim, da unidade protocolar HTTP dentro do segmento TCP (camada de aplicação).

Quando se corre o *Wireshark*, após uma captura de tráfego, é mostrada uma interface equivalente à apresentada na Figura 3.

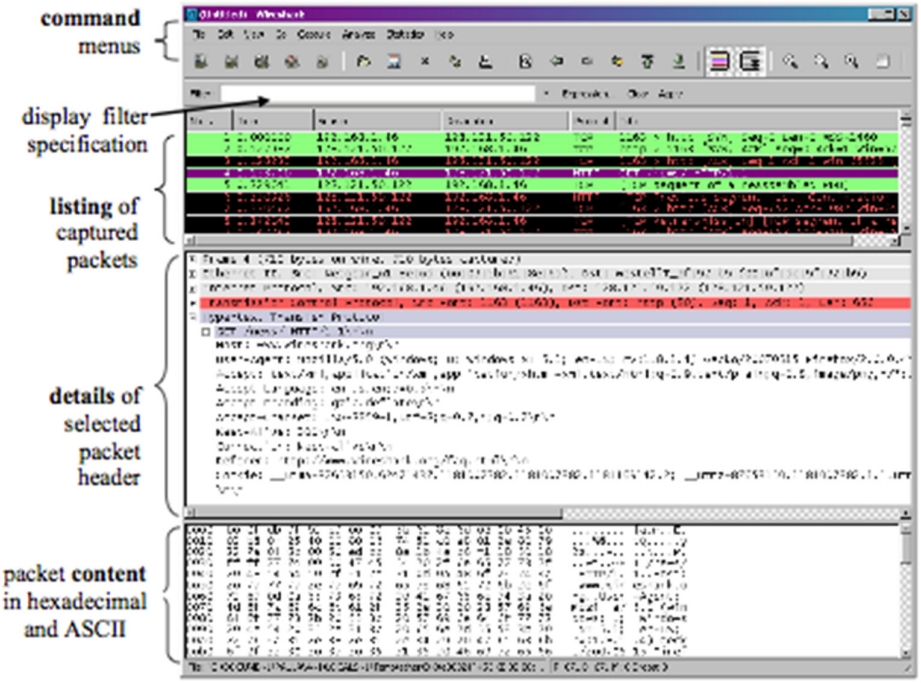

Figura 3: Interface gráfica do Wireshark

A interface do *Wireshark* tem cinco componentes principais:

- O menu de comando onde se destacam o *File* e o *Capture*. O *File* permite guardar uma determinada captura de tráfego ou abrir um ficheiro com informação previamente armazenada. O menu *Capture* permite iniciar e terminar a captura de tráfego.
- A janela de listagem de pacotes mostra uma linha com um sumário de cada pacote capturado, incluindo um número de pacote (atribuído pelo *Wireshark*), tempo de captura, origem, destino, protocolo, tamanho e tipo de informação.
- A janela de detalhe de cabeçalhos de pacote disponibiliza detalhes sobre o pacote selecionado na janela anterior. Podem-se obter detalhes sobre a trama *Ethernet*, o datagrama IP e os protocolos de nível superior.
- A janela do conteúdo do pacote mostra o conteúdo inteiro do pacote capturado.
- Junto do topo da *Wireshark* GUI existe um campo de filtragem dos pacotes a visualizar, onde podem ser colocadas expressões que permite filtrar os dados capturados. Podem-se definir filtros variados tais como definir o nome do protocolo, o endereço de origem e/ou destino, etc..

Experimentação

Assumindo que o seu computador está com acesso à Internet:

1. Ative um *web browser*...
2. Arranque o *Wireshark* na sua máquina nativa. Deve obter uma janela inicial, similar à da Figura 1...
3. Escolha *Capture* *Options*. Em *Display Options* desative a opção *Hide Capture Info Dialog*. Se o seu computador tiver mais que uma interface deve seleccionar a pretendida (assuma a selecionada por defeito pelo *Wireshark*)...
4. Inicie a captura de tráfego. Deverá aparecer uma janela com o sumário da captura de pacotes realizada. Pode também explorar a opção *Statistics* para verificar os tipos de protocolo envolvidos no tráfego capturado...
5. Com o *Wireshark* a correr, aceda à página www.fccn.pt a partir o seu browser. Pare a captura...

6. As mensagens HTTP trocadas com o servidor (e muito outro tráfego) vão aparecer na captura realizada. Filtre o tráfego *http* para facilitar a análise...
7. Selecione a primeira mensagem HTTP GET. Inspecione o processo de encapsulamento protocolar, observando a informação envolvida em cada um dos níveis (trama *Ethernet*, datagrama IP, transporte, etc.)...
8. Identifique quais os endereços IP em uso na sua máquina e na máquina destino (servidor www.fccn.pt).

Referências

- Manual do *Wireshark*, https://www.Wireshark.org/#learnWS