Anexos

ANEXO A: ACRÓNIMOS

- **AH:** Authentication headers
- **CSV:** Comma Separated Values
- **DIFS:** DCF Interframe Space
- **EID:** Endpoint identifier
- **IETF:** Internet Engineering Task Force
- **IP:** Internet protocol
- **IPSec:** IP Security protocol
- **LISP:** Locator/Identifier Separation Protocol
- **LSB:** Less significant bits
- **MS/MR:** Map server / map register
- **MSB:** Most Significant Bits
- **MTU:** Maximum Transfer Unit
- **NMS:** Network Monitoring Systems
- **OTT:** Over The Top
- **QoS:** Quality of Service
- **RLOC:** Routing LOCator
- **ROHC:** RObust Header Compression
- **SA:** Security associations
- **SIFS:** Short Interframe Space
- **SSH:** Secure Shell
- **TCP:** Transmission control protocol
- **TLS:** Transport Layer Security
- **UDP:** User datagram protocol
- **VoIP:** Voice over IP.
- **VPN:** Virtual private network

ANEXO B: INSTALACIÓN Y CONFIGURACIÓN DEL ENTORNO DE PRUEBAS

Raspbian es la versión de Debian adaptada a la Raspberry Pi. Puede descargarse la imagen del sistema en formato .ISO en descarga directa o a través del protocolo Torrent desde la página oficial de Raspberry Pi de manera gratuita:

- https://www.raspberrypi.org/downloads/raspbian

Para instalar Raspbian en una Raspberry Pi deberemos copiar la imagen en una tarjeta micro SD. Para ello, la propia página oficial de la distribución recomienda usar el programa *Win32DiskImager* disponible para su descarga desde *Sourceforge* bajo licencia Open Source:

- https://sourceforge.net/projects/win32diskimager

Los pasos para realizar la instalación pueden consultarse en detalle en página de documentación de Raspberry Pi:

- https://www.raspberrypi.org/documentation/installation/installing-images/README.md

Una vez grabado, introducimos la tarjeta micro SD en la Raspberry Pi y la conectamos a la alimentación. El sistema arrancará y necesitaremos las siguientes credenciales de acceso:

> **Usuario:** *pi*
> **Contraseña:** *raspberry*

Una vez hayamos accedido al sistema, debemos habilitar el reenvío de paquetes IP para ser capaces de utilizar las Raspberry Pi como equipos enrutadores. Además, es útil configurar este parámetro de manera permanente ya que de lo contrario se resetearía con un reinicio del equipo. Para ello abrimos en un editor el fichero de configuración situado en */etc/sysctl.conf* con permisos de administrador:

```
> sudo nano /etc/sysctl.conf
```

Buscamos el comentario *"# Uncomment the next line to enable packet forwarding for IPv4"* y descomentamos la línea situada justo debajo quitando la almohadilla del principio de línea:

```
net.ipv4.ip_forward = 1
```

Salimos con la combinación *crtl + x* y guardamos el archivo.

Ahora vamos a pasar a configurar el direccionamiento IP. Para ellos editamos el fichero */etc/network/interfaces* como administrador:

```
> sudo nano /etc/network/interfaces
```

En este archivo debemos configurar tanto la interfaz inalámbrica como la Ethernet añadiendo las siguientes líneas. Esta sería la configuración para el equipo rpi-1.

```
auto wlan0
iface wlan0 inet static
       address 192.168.70.1
       netmask 255.255.255.0
       wireless-channel 11
       wireless-essid LAB
       wireless-mode ad-hoc

auto eth0:4
iface eth0:4 inet manual
       pre-up /sbin/config/ eth0:4 192.168.4.171 netmask 255.255.255.0 up
```

Para los demás equipos realizaríamos un procedimiento análogo según el siguiente esquema de direccionamiento:

	LAN-4	*LAN-5*	*LAN-70*
Red	192.168.4.0	192.168.5.0	192.168.70.0
Máscara	255.255.255.0	255.255.255.0	255.255.255.0
IP host-1	.229	-	-
IP host-2	-	.230	.3
IP R-4	.171	-	.1
IP R-5	-	.172	.2
IP MS/MR	-	-	.3

Tabla 1 – Esquema de direccionamiento del montaje de laboratorio

ANEXO C: INSTALACIÓN, CONFIGURACIÓN Y USO DE HERRAMIENTAS

LISPmob con SimpleMux

Para instalarlo, primeramente debemos obtener el código fuente de LISPmob con SimpleMux desde el repositorio del proyecto en Github. La dirección del repositorio del proyecto es la siguiente:

- https://github.com/Simplemux/lispmob-with-simplemux

Para instalar LISPmob, en primer lugar, necesitamos tener instalado Git. Si es necesario podemos

obtenerlo instalando el paquete *git-all* desde el gestor de paquetes de Aptitude. Es conveniente, antes de instalar cualquier paquete, actualizar la información disponible de los repositorios de Debian:

```
>  sudo apt-get update
>  sudo apt-get install git-all
```

Vamos a instalar ahora la versión modificada de LISPmob que incluye SimpleMux, para ello nos bajamos la versión más reciente del proyecto desde Github mediante el siguiente comando:

```
>  git clone git://github.com/Simplemux/lispmob-with-simplemux.git
```

Debemos instalar además, los paquetes de dependencias que necesita LISPmob para funcionar correctamente:

```
>  apt-get install gengetopt
>  apt-get install libzmq3 libzmq3-dev
>  apt-get install libxml2-dev
>  apt-get install libconfuse-dev
```

También debemos tener instalado ROHC para ello hay que descomprimir rohc-1.7.0.tar.xz

```
>  tar -xf rohc-1.7.0.tar.xz
```

Entramos en el directorio *rohc-1.7.0* y ejecutamos

```
>  ./configure   #esta instrucción instala en /usr/local/bin y /usr/local/lib
>  export LD_LIBRARY_PATH=/usr/local/lib
>  make all
>  make check
>  make install
>  make
```

Una vez instalado el software, podremos ya lanzar LISP en el dispositivo. La mejor manera de configurar LISP es pasando una ruta a un fichero de configuración como parámetro a la hora de ejecutarlo. Para ello vamos a crear un fichero de configuración:

```
>  sudo nano lisp.conf

   operating-mode=xTR
   map-resolver={192.168.70.3}
   map-server {
          address=192.168.70.3
          key-type=1
          key=lispmob
          proxy-reply=off
   }
   database-mapping {
          eid-prefix=192.168.4.0/24
          rloc-address{
                address=192.168.70.1
                priority=1
                weight=100
```

```
        }
    }
```

Para ejecutar LISP se ha preparado un pequeño script que comprueba que no existan ya lanzados procesos de la aplicación y de haberlos, los cierra y lanza LISP de nuevo. Esto soluciona posibles errores de ejecución.

```
> ./launch-lisp.sh
```

IPsec-tools

Una de las maneras de utilizar IPSec en Linux es a través del comando *setkey*. Esta utilidad se encuentra dentro del paquete ipsec-tools. Este paquete es instalable mediante el gestor de paquetes de Debian, Aptitude; aunque es común que se suministre por defecto en la mayoría de distribuciones de Linux.

```
> sudo apt-get install ipsec-tools
```

Para su utilización debemos crear un fichero de configuración especificando las características de nuestra política de IPSec y las condiciones de aplicación de las mismas, es decir, unas reglas que especifiquen a qué conexiones se aplican la seguridad IPSec y a que conexiones no se les aplica.

Una vez creado un de configuración, lanzaremos *setkey* con la opción *-f*, que indica el paso por parámetro de un fichero de configuración del túnel IPSec, en nuestro caso llamado **ipsec.conf**. La manera de crear un túnel IPSec entonces es utilizando el siguiente comando:

```
> sudo setkey -f ipsec.conf
```

Lo primero que haremos al crear un fichero de configuración de IPSec es escribir las instrucciones para borrar las tablas tanto de políticas como de filtrado que hubiera en ese momento. Esto lo hacemos debido a que si lo ejecutamos sin borrar las anteriores, lo que haríamos sería añadir nuevas políticas y filtrado. Desde el punto de vista de un entorno de test creamos una situación inestable, porque dependemos de lo que hubiese en ese momento configurado en la máquina. De modo que, para obtener resultados reproducibles conviene borrar primero cualesquiera reglas IPSec haya configuradas en ese momento.

```
#! /usr/sbin/setkey -f
# Iniciar las SAD y SPD
flush; #Borrar las reglas existentes de IPsec
spdflush; #Borrar las políticas existentes de IPsec

# Ahora creamos nuevas reglas
add    192.168.70.1    192.168.70.2    esp    0x201    -m    tunnel    -E    cast128-cbc
0x12345678901234567890123456789012;
add    192.168.70.1    192.168.70.2    ah    0x200    -m    tunnel    -A    hmac-sha256
0x1234567890123456789012345678901234567890123456789012345678901234;
add    192.168.70.2    192.168.70.1    esp    0x301    -m    tunnel    -E    cast128-cbc
0x12345678901234567890123456789012;
```

```
add 192.168.70.2 192.168.70.1 ah 0x300 -m tunnel -A hmac-sha256
0x12345678901234567890123456789012345678901234567890123456789012345678901234;

# add 192.168.70.1 192.168.70.2 ah 0x200 -m tunnel -A aes-xcbc-mac
0x123456789012345678901234567890123;
# add 192.168.70.2 192.168.70.1 ah 0x300 -m tunnel -A aes-xcbc-mac
0x123456789012345678901234567890123;

# Ahora creamos nuevas políticas
spdadd 192.168.4.0/24 192.168.5.0/24 any -P out ipsec esp/tunnel/192.168.70.1-
192.168.70.2/require ah/tunnel/192.168.70.1-192.168.70.2/require;
spdadd 192.168.5.0/24 192.168.4.0/24 any -P in ipsec esp/tunnel/192.168.70.2-
192.168.70.1/require ah/tunnel/192.168.70.2-192.168.70.1/require;
```

D-ITG

D-ITG puede ser descargado desde la página del proyecto, en el sitio web de la Universidad de Nápoles:

> http://traffic.comics.unina.it/software/ITG/download.php

También podemos descargarlo directamente en el sistema Linux por consola de comandos:

```
> wget http://traffic.comics.unina.it/software/ITG/codice/D-ITG-2.8.1-r1023-
  src.zip
```

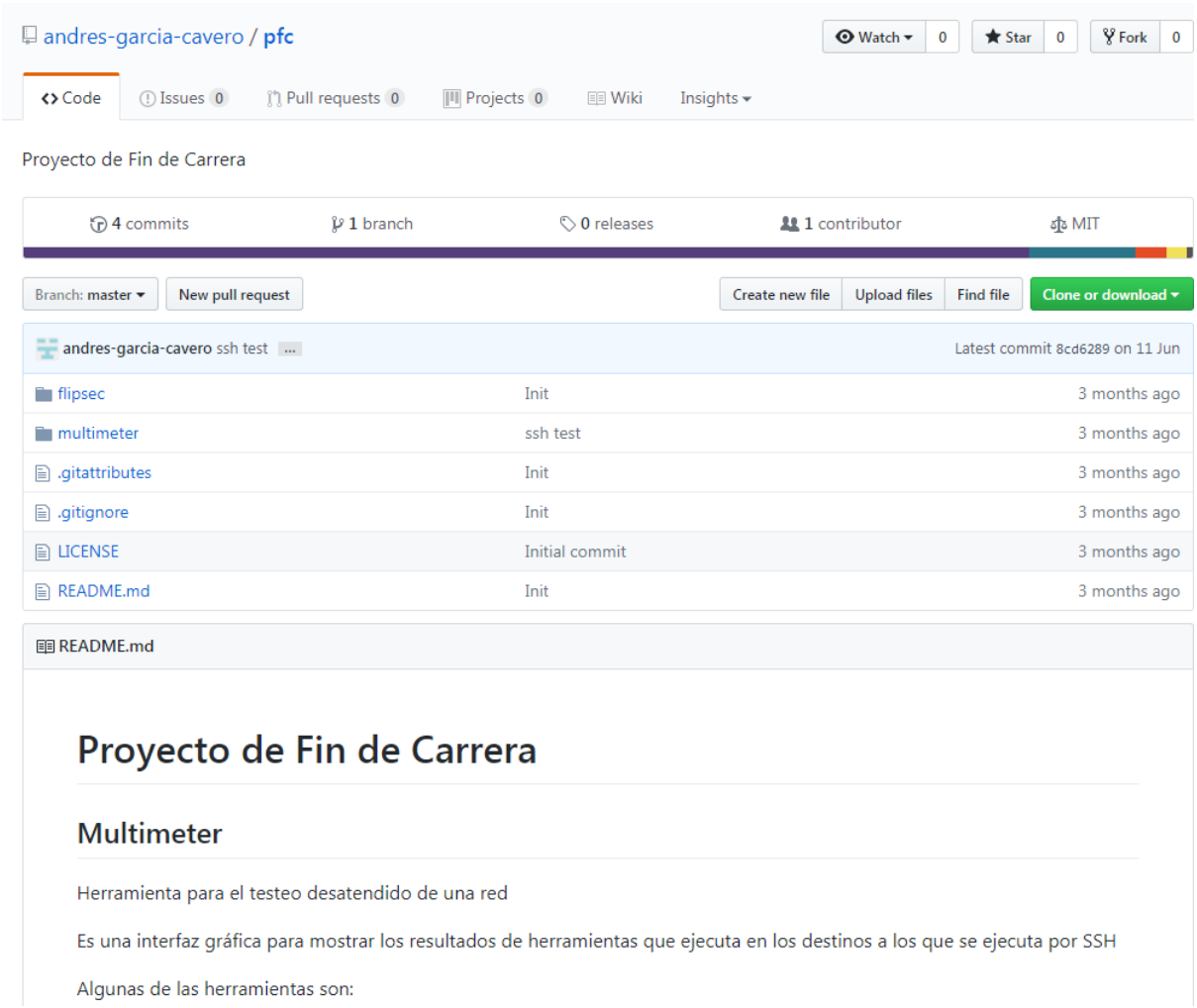
Una vez descargado, lo descomprimimos y procedemos a su instalación:

```
> unzip D-ITG-2.8.1-r1023-src.zip
> cd D-ITG-2.8.1-r1023/src
> make
> sudo make install
```

Una vez terminado el proceso de compilación, los archivos binarios generados serán copiados en la ruta *D-ITG-2.8.1-r2058M/bin.*

ANEXO D: INSTALACIÓN, CONFIGURACIÓN Y USO DE LA HERRAMIENTA MULTIMETER

La aplicación Multimeter no necesita ahora mismo de instalación ya que se encuentra en estado de desarrollo Alpha. Por lo tanto, la manera de trabajar con ella es descargando el código desde el repositorio del proyecto en GitHub: https://github.com/andres-garcia-cavero/pfc



Para ello podemos utilizar Git o descargarlo en formato .ZIP desde GitHub.

Para hacerlo mediante Git:

```
> git clone https://github.com/andres-garcia-cavero/pfc
```

Para poder ejecutarla en nuestro Sistema, debemos tener instalado Node JS.

Es posible descargarlo desde la página oficial: https://nodejs.org/es/

Dispone de versión para todas las plataformas.

Una vez instalado Node JS instalaremos los paquetes utilizados en el proyecto y sus dependencias situando una consola en el directorio raíz de la descarga e introduciendo el comando de NPM (Node Package Manager, el gestor de paquetes de Node JS):

```
>   npm install
```

NPM descargará todos los ficheros necesarios y los guardará en el directorio *node_modules* dentro de nuestra raíz.

Tras ello podemos ejecutar desde la raíz del proyecto el comando:

```
>   ng serve
```

Este comando realizar la compilación del código de la aplicación de Angular y levanta un servidor web con el resultado.

Tras ello podemos lanzar la aplicación mediante el comando:

```
>   electron
```

Cabe nombrar que todos estos comandos solo son necesarios debido a que la aplicación se encuentra aún en estado de desarrollo, en una aplicación de producción empaquetada, únicamente habría que ejecutar un archivo .EXE que iniciaría un proceso de instalación análogo al de cualquier otro software del mercado.

Las instrucciones de uso de la herramienta se proporcionan a través de los distintos ficheros README.md del repositorio del proyecto en GitHub.

ANEXO E: SCRIPTS UTILIZADOS EN LOS TESTS

saturation-sweep.sh

```
#!/bin/bash
# c = size in bytes
sizeStart=100 #bytes
sizeStep=100 #bytes
sizeEnd=2000 #bytes
# C = packets per second
rateStart=100 #packets/second
rateStep=100 #packets/second
rateEnd=2000 #packets/second
#First loop is size
#Second loop is rate
for (( size=sizeStart; size<=sizeEnd; size=size+sizeStep ))
do
        for (( rate=rateStart; rate<=rateEnd; rate=rate+rateStep ))
        do
                echo "log_size_${size}_rate_${rate}"
```

```
                sudo ITGSend -a 192.168.70.2 -c $size -C $rate -t 1000 -x
"log_size_${size}_rate_${rate}"
        done
done
```

data-decoder.sh

```
#!/bin/bash
sizeStart=100 #bytes
sizeStep=100 #bytes
sizeEnd=2000 #bytes
rateStart=100 #packets/second
rateStep=100 #packets/second
rateEnd=2000 #packets/second
for (( size=sizeStart; size<=sizeEnd; size=size+sizeStep ))
do
        for (( rate=rateStart; rate<=rateEnd; rate=rate+rateStep ))
        do
                sudo ITGDec "log_size_${size}_rate_${rate}" -c 10000
"decoded_log_${size}_rate_${rate}"
        done
done
```

Este script extrae todas las mediciones almacenadas por ITGRecv en los archivos *log_** y los almacena en los archivos *decoded_log_** en formato TSV. El comando *–c* indica que queremos todos los parámetros medidos y el número que sigue a continuación indica cada cuantos segundos queremos que se calculen dichos parámetros. Como queremos un único valor por test, lo configuramos a un valor mayor que el tiempo de test (1000 ms del script *saturation-sweep.sh*) durante el cual hemos realizado el envío de datos.

join-in-csv.js

```
var fs = require('fs');
var dirname = 'decode_log_/';
var csvFile = 'data.csv';
var os = require('os');
fs.writeFile(csvFile,'"Size","Rate","Time","Bitrate","Delay","Jitter","Packe
t loss"'+os.EOL,(error)=>{
       if (error) console.log(error);
});
var count = 0;
fs.readdir(dirname, (error, filenames) => {
       if (error) { console.log(error); }
       filenames.forEach((filename)=>{
              fs.readFile(dirname + filename, 'utf-8', (error, content) => {
                     if (error) { console.log(error); }
                     var filenamePartsArray = filename.split('_');
                     var size = filenamePartsArray['2'];
                     var rate = filenamePartsArray['4'];
                     var    csvString    =    size+','+rate+','+content.split('
').join(',');
                     fs.appendFile(csvFile,csvString,(error)=>{
                            if (error) console.log(error);
```

```
                        count++;
                        process.stdout.write("Files: " + count + "\r");
                    });
                });
            });
        });
```

dot-graph.js

```
        var tolerance = 1.03;
        var dataset;
        var svgWidth = 800;
        var svgHeight = 800;
        var margin = 50;
        var maxY = 2000;
        var maxX = 2000;
        // X Axis =  Rate
        var y = d3.scaleLinear().domain([maxX,0]).range([0,svgWidth]);
        // Y Axis = Size
        var x = d3.scaleLinear().domain([0,maxY]).range([0,svgHeight]);
        d3.csv("data.csv", (data) => {
          var svg = d3.select("body")
      .append("svg")
      .attr("width", margin + svgWidth + margin)
      .attr("height", margin + svgHeight + margin);
            svg
                .selectAll("circle")
                .data(data)
                .enter()
                .append("circle")
                .attr("cx", (d,i) => {
                    return x(d.Rate);
                })
                .attr("cy", (d,i) => {
                    return y(d.Size);
                })
                .attr("r", 7)
                .attr('data-size', (d) => { return d.Size;})
                .attr('data-rate', (d) => { return d.Rate;})
                .attr('fill',(d,i)=>{

                    var bitrateCalc = d.Size * d.Rate * 8;
                    var  measuredBitrate  =  Number(d.Bitrate)  *
1000;

                    console.log(bitrateCalc);
                    console.log(typeof bitrateCalc);
                    console.log(measuredBitrate);
```

```
                                        console.log(typeof measuredBitrate);

                                    if (bitrateCalc > (tolerance * measuredBitrate) )
{
                                        return 'red';
                                    } else {
                                        return 'green';
                                    }
                                });
```

});

launch-lisp.sh

```
#!/bin/bash
# Cleaning previously launched instances
rm /var/run/lispd.pid
# Launch LISP
cd /home/pi/andres/lispmob-with-simplemux/lispd
./lispd -f /home/pi/multimeter/lisp.conf
```

ANEXO F: REFERENCIAS

REDES

[SALDANA] Saldana, J., Forcén, I., Fernández-Navajas, J., & Ruiz-Mas, J. (2015, October). Improving Network Efficiency with Simplemux. In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on* (pp. 446-453). IEEE.

LISP

[CISCO-LISP] Cisco Systems Inc. http://lisp.cisco.com/lisp_over.html.
[WIK-ILISP] Wikipedia. https://en.wikipedia.org/wiki/Locator/Identifier_Separation_Protocol.
[IETF-RFC6830] Internet Engineering Task Force. https://tools.ietf.org/html/rfc6830.
[FARINACCI] Farinacci, D., Lewis, D., Meyer, D., & Fuller, V. (2013). The locator/ID separation protocol (LISP).

LISPmob

[LISPMOB] Open Overlay Router project. http://www.openoverlayrouter.org/lispmob/.
[GITHUB-LISPMOB] GitHub Lispmob project repository. https://github.com/LISPmob/lispmob

SimpleMux

[IETF-SALDANA] Internet Engineering Task Force. https://tools.ietf.org/html/draft-saldana-tsvwg-simplemux-08 (work in progress)
[GITHUB-LISPMOB-SMUX] GitHub Lispmob with SimpleMux project repository.
https://github.com/Simplemux/lispmob-with-simplemux

IPSec

[SEO] Seo, K., & Kent, S. (2005). Security architecture for the internet protocol.

D-ITG

[DITG] *A. Botta, A. Dainotti, A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios", Computer Networks (Elsevier), 2012, Volume 56, Issue 15, pp 3531-3547.* http://wpage.unina.it/a.botta/pub/COMNET_WORKLOAD.pdf

TCPDUMP

[TCPDUMP] Jacobson, Van, Craig Leres, and Steven McCanne. "TCPDUMP public repository." *Web page at http://www. tcpdump. org* (2003).

NODE JS

[NODEJS] Tilkov, S., & Vinoski, S. (2010). Node. js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing*, *14*(6), 80-83.

ELECTRON

[ELECTRON] Fang, D., Keezer, M., Williams, J., Kulkarni, K., Pienta, R., & Chau, D. H. (2017, April). Carina: Interactive million-node graph visualization using web browser technologies. In *Proceedings of the 26th International Conference on World Wide Web Companion* (pp. 775-776). International World Wide Web Conferences Steering Committee