Creació d'un sniffer gràfic en entorn GNU

Arnau Lamarca Lasierra
Grau en Enginyeria Informàtica

Maria Isabel March Hermo

07/01/2018

Llicències alternatives (triar alguna de les següents i substituir la de la pàgina anterior)

A) Creative Commons:

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-CompartirIgual 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-SenseObraDerivada 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-CompartirIgual 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement 3.0 Espanya de Creative Commons

B) GNU Free Documentation License (GNU FDL)

Copyright © ANY EL-TEU-NOM.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free

C) Copyright

FITXA DEL TREBALL FINAL

Títol del treball:	***Creació d'un sniffer gràfic en entorn GNU***
Nom de l'autor:	***Arnau Lamarca Lasierra***
Nom del consultor:	***Maria Isabel March Hermo***
Data de lliurament (mm/aaaa):	***01/2018***
Àrea del Treball Final:	***Xarxes de computadors***
Titulació:	***Enginyeria Informàtica***

Resum del Treball (màxim 250 paraules):

Des de l'aparició d'Internet, la considerada xarxa de xarxes, la popularitat dels sistemes informàtics en general i les aplicacions que requereixen l'ús d'una xarxa en particular ha crescut exponencialment. En l'actualitat, hi ha milers de milions d'usuaris que, sia des de la comoditat de la seva llar o mentre es mouen o fan esport amb les noves tendències d'ubiqüitat, usen aplicacions que es recolzen, en diverses formes i diferents graus, en l'ús i intercanvi d'informació per mitjà d'una xarxa, per regla general, pública.

La seguretat d'aquestes xarxes esdevé, doncs, una qüestió cabdal i és en aquest context on aplicacions com els analitzadors de xarxa prenen una gran importància per a la seva capacitat d'analitzar el flux de dades present en el si d'una xarxa, el que aporta una potencial habilitat de poder assegurar-ne la seguretat, capacitar-se com a expert en xarxes o deduir el funcionament d'un determinat sistema estudiant les comunicacions que amb ell s'estableixen, entre moltes altres coses.

Un analitzador de xarxa és un programa que captura paquets en una xarxa, els descodifica i els mostra per tal que se'n pugui realitzar l'anàlisi. El present treball cerca la creació d'una eina d'aquest tipus, desenvolupada en un entorn GNU/Linux.

El present desenvolupament es presenta com a una bona forma de millorar i créixer com a professional, alhora que cerca proporcionar una eina de lliure distribució que qualsevol persona pugui emprar i estendre per tal d'adaptar-la a les seves necessitats personals.

Abstract (in English, 250 words or less):

Since the Internet, defined as the network of networks, appeared, the popularity of computer systems in general and applications that require the use of a network in particular has grown in an exponential way. Nowadays, there are thousands of millions of users that, either from their home's comfort or while they are on the go or even while they are doing sport with the new ubiquity trend, use applications that rely, in different ways and degrees, on the use and the information exchanged through a network, generally speaking, a public one.

Those networks's security is, thus, a really important subject and is precisely in this context where applications such as network sniffers have a great importance because of their ability to analyze the data flow on a network, which gives the potential ability to secure it, to qualify as a network professional or even to deduce how a certain system works by just analyzing the data flow, among many other things.

A sniffer is a program that captures packets in a network, decodes them and show them so they can be analyzed. The current work tries to create a sniffer, fully developed on a GNU/Linux environment.

The current development is seen as a good way to grow and improve my skills as professional and, at the same time, to provide users with a free software tool that anyone can use and extend in order to adapt it so it meets any personal requirements.

Paraules clau (entre 4 i 8):

Sniffer
Transit
Protocol
Paquet
GUI

Índex

Llista de figures

1. Introducció

Des de l'aparició d'Internet, la considerada xarxa de xarxes, la popularitat dels sistemes informàtics en general i les aplicacions que requereixen l'ús d'una xarxa en particular ha crescut exponencialment. En l'actualitat, hi ha milers de milions d'usuaris que, sia des de la comoditat de la seva llar o mentre es mouen o fan esport amb les noves tendències d'ubiqüitat, usen aplicacions que es recolzen, en diverses formes i diferents graus, en l'ús i intercanvi d'informació per mitjà d'una xarxa, per regla general, pública.

La seguretat d'aquestes xarxes esdevé, doncs, essencial per al benestar dels usuaris, que volen assegurar-se que la seva informació privada ho segueix sent. En aquest sentit, les xarxes públiques poden ser potencialment hostils i alguns programaris poden posseir codi que provoqui un intercanvi de paquets no lícit, com per exemple, informació de control sobre un computador remot, DDoS i en general tota mena de malware. En aquest context és on eines com els analitzadors de xarxa o sniffers poden ser d'utilitat.

La informació que aquests programes proporcionen permet visualitzar allò que succeeix en el si d'una xarxa, com ho pot ser les connexions que es realitzen amb l'exterior, la seqüència de trànsit intercanviat, el tipus de trànsit i en general tot allò que permeti als usuaris poder disposar de les eines necessàries per a poder prendre decisions.

En efecte, molts individus en informàtica, des d'estudiants que volen aprendre el contingut de paquets d'un determinat protocol fins a experts que tracten de protegir una xarxa de possibles atacs, passant per a aquells individus que volen obtenir informació privada, com contrasenyes, empren aquestes eines amb freqüència.

Per exemple, un expert en seguretat voldrà saber quins paquets s'envien per a detectar possibles atacs provinents de l'exterior, o fins i tot de l'interior de la xarxa. Un estudiant de xarxes, en canvi, podria emprar un analitzador a fi de saber els intercanvis corresponents a un determinat protocol així com el contingut i forma de cada un dels paquets.

En aquest context, s'ha desenvolupat una aplicació que realitza precisament aquesta tasca, la d'analitzar el trànsit present en una xarxa, amb l'objectiu de ser d'utilitat per a qualsevol usuari que ho necessiti, i que el podrien obtenir i estendre sense cap mena de restricció.

1.1 Context i justificació del Treball

Podem dir que existeixen dues principals motivacions per a la realització del present projecte, donades les seves característiques:

1. En primer lloc, el creixement personal mitjançant la creació d'un producte que implica la unió del món de les xarxes, que personalment es considera molt interessant i cabdal en el panorama actual, de la programació, que és quelcom de vital importància en informàtica i de la gestió de projectes, que és rellevant en qualsevol entorn.

   El desenvolupament, a més, implica l'adquisició de nous i rellevants coneixements com, per exemple, el treball amb interfícies gràfiques d'usuari, pel que no es disposa d'experiència prèvia, o aprofundir en les possibles vies d'implementació d'una eina d'anàlisi de xarxa.

2. En segon lloc, la creació d'un programa que pugui ser emprat per a qualsevol persona que requereixi l'ús d'una eina d'aquest tipus. Aquest sistema hauria de ser intuïtiu, de software lliure i de distribució gratuïta, a fi que pugui, no només ser descarregat i usat per a qualsevol sense restriccions sinó també poder fer totes les modificacions que es consideressin oportunes per tal d'adaptar el programa a les necessitats particulars.

Per tant, el present desenvolupament no només representa una gran oportunitat per a créixer i aprendre sinó que també suposa la creació d'una eina de lliure distribució.

1.2 Objectius del Treball

L'objectiu del projecte és el desenvolupament d'un analitzador de xarxa (sniffer) completament funcional, que inclogui les característiques més habituals d'aquest tipus d'eina i presenti en una interfície gràfica d'usuari diferents dades d'interès.

Els objectius del projecte, concretats, es poden veure en la següent llista:

- Crear un Sniffer complet i funcional.
- Capturar el tràfic de xarxa dels protocols TCP, UDP, ICMP i ARP i mostrar informació resumida sobre cada un dels paquets de forma que puguin ser analitzats.
- Deixar a l'elecció de l'usuari la decisió sobre si vol ampliar aquesta informació resumida, cas en en el qual s'hauria de presentar informació completa sobre el paquet, com ho són les capçaleres i els seus continguts, així com el bolcat hexadecimal.
- Desenvolupar un sistema que permeti desar sessions de captura, de forma que l'usuari pugui analitzar el tràfic amb posterioritat.
- En relació a l'anterior punt, desenvolupar un mecanisme que permeti obrir fitxers de captura.
- Crear un sistema que permeti la implantació de filtres de captura, de forma que l'usuari pugui reduir els paquets que el programa captura a aquells que segueixin un determinat criteri plasmat en una expressió. Aquests criteris poden incloure la font del paquet, el protocol, etc.
- Crear un sistema de filtres gràfics, que permeti a l'usuari visualitzar únicament aquells paquets que compleixin un determinat criteri, com ho és l'adreça de font, la de destí o el protocol.
- La creació d'un sistema d'estadístiques que presenti a l'usuari informació relativa a la quantitat de paquets que s'han capturat, així com el nombre de paquets diferenciats per protocol, amb el corresponent percentatge.
- La creació d'una interfície gràfica d'usuari que sigui atractiva a la vegada que fàcil d'emprar i intuïtiva.
- Fer el desenvolupament emprant exclusivament eines de software lliure, que funcionin en sistemes GNU/Linux.

1.3 Enfocament i mètode seguit

Per començar, s'ha considerat important documentar-se sobre quines altres eines hi ha disponibles al mercat per a poder saber quins possibles graus de completesa es pot arribar a assolir. En efecte, hi ha moltes solucions disponibles amb característiques similars al mercat així que el fet d'obtenir informació de primera mà sobre com funcionen altres programes ja existents ajudà a saber què fer exactament.

Després de definir els requisits del projecte, es passà a l'etapa de disseny, en la qual es crearen els casos d'ús, el diagrama estàtic de classes i es van fer tots els passos necessaris per a conceptualitzar la solució a aplicar. Es van identificar 5 mòduls que s'havien d'implementar, concretament, el de la interfície gràfica d'usuari, que també va ser definida en aquesta etapa, el de captura, el de filtres, el de logging i el d'estadístiques.

Un cop es va determinar el què del projecte, és a dir, se'n va definir l'abast, i acabada la fase de disseny es passà a considerar les possibles vies d'implementació del projecte, pel qual es féu necessari l'obtenció d'informació respecte a protocols a capturar, possibles eines d'ajuda i biblioteques, etc. Es féu, per tant, un procés de recopilació d'informació que va conduir a la definició de les eines que s'havien d'emprar per a implementar el projecte.

Quan es va disposar dels principals artefactes del disseny i un cop s'havia entès la forma en la qual es podria desenvolupar, es passà a la fase d'implementació, que va ocupar la major part del temps de desenvolupament. En aquesta fase es va requerir un constant procés d'aprenentatge i millora del codi, que es va prosseguir de forma iterativa. És a dir, en tots els mòduls, s'optà per a implementar primer els aspectes bàsics que el definien i realitzar, amb posterioritat, un procés de refinament sobre el codi.

Paral·lelament a la implementació, es realitzaren un seguit de proves per tal de vetllar que tot el que s'estava desenvolupant funcionava correctament. Cal comentar, però, que tot i que aquesta fase fou transversal a la d'implementació, es féu un procés d'avaluació final després d'implementar tots els components per tal de resoldre possibles errors específics que no s'haguessin considerat en provar cada component o mòdul específic.

1.4 Planificació del Treball

Per a determinar el calendari de la realització del projecte, s'han identificat un seguit de tasques a realitzar i s'han definit unes línies temporals per a cada una d'elles en funció de la previsió del temps disponible i la seva dificultat relativa.

Tasca 1:

Temporització: 5 dies (3 d'octubre – 8 d'octubre).
Descripció: Definició de l'abast del projecte i casos d'ús.

Objectius:
- Definir l'abast del projecte i els casos d'ús.

Passos:
- Determinar les funcionalitats bàsiques del programa, el que determinarà el que caldrà fer.
- Definir els possibles casos d'ús.

Tasca 2:

Temporització: 10 dies (9 d'octubre – 19 d'octubre).
Descripció: Recollida i classificació de tota la informació que pugui ser rellevant per dur a terme el projecte.

Objectius:
- Poder determinar sobre quin entorn gràfic realitzar el projecte.
- Saber quina possible ajuda podem emprar, tant en biblioteques com en IDEs.

Passos:
- Cerca d'informació sobre les diferents biblioteques gràfiques (GTK, QT) disponibles per a l'entorn.
- Cerca d'informació sobre altres sniffers ja creats.
- Cerca d'informació sobre les possibles biblioteques d'ajuda per a la captura pel que fa a la posterior implementació.
- Cerca d'informació sobre els possibles IDEs per al desenvolupament.
- Cercar informació sobre diferents protocols que podrien ser capturats.
- Generar documentació.

Tasca 3:

Temporització: 10 dies (20 d'octubre – 30 d'octubre).
Descripció: Definir i implementar interfície gràfica.

Objectius:
- Tenir la interfície gràfica definida i implementada.

Passos:
- Fer un esbós de la possible GUI del programa, que es fonamenti en la usabilitat.
- Implementar la GUI del programa.
- Generar documentació.

Tasca 4:

Temporització: 2 setmanes (31 d'octubre – 15 de novembre).
Descripció: Implementació mòdul de captura de paquets (iteratiu).

Objectius:
- Tenir el sistema de captura funcional.

Passos:
- Implementar el sistema de captures de paquets.
- Fer les proves adients per a assegurar correcte funcionament del mòdul.
- Generar documentació.

Tasca 5:

Temporització: 10 dies (16 de novembre – 26 de novembre).
Descripció: Creació del sistema de filtratge (iteratiu).

Objectius:
- Tenir el sistema de filtratge funcional.

Passos:
- Implementar sistema de filtre de paquets.
- Fer les proves adients per a assegurar correcte funcionament.
- Generar documentació.

Tasca 6:

Temporització: 10 dies (27 de novembre – 7 de desembre).
Descripció: Creació del sistema de Logging (iteratiu).

Objectius:
- Tenir el sistema de Logging funcional.

Passos:
- Implementar sistema de Logging.
- Fer les proves adients per a assegurar correcte funcionament.
- Generar documentació.

Tasca 7:

Temporització: 1 setmana (8 de desembre – 15 de desembre).
Descripció: Creació del sistema d'estadístiques (iteratiu).

Objectius:
- Tenir el sistema d'estadístiques funcional.

Passos:
- Implementar sistema d'estadístiques.
- Fer les proves adients per a assegurar correcte funcionament.
- Generar documentació.

Tasca 8:

Temporització: 1 setmana (16 de desembre – 23 de desembre).
Descripció: Creació de l'instal·lador i documentació final programa.

Objectius:
- Tenir el programa acabat i amb la documentació adient finalitzada.

Passos:
- Creació de l'instal·lador.
- Generar la documentació final del programa.

Tasca 9:

Temporització: 1 setmana (24 de desembre – 31 de desembre).
Descripció: Possibles desviacions i proves finals.

Objectius:
- Tenir una setmana per a assegurar-se que el programa en el conjunt funciona com tindria i absorbir possibles desviacions que es puguin produir.

Passos:
- Comprovar que tot funciona correctament.

Tasca 10:

Temporització: 1 setmana (1 de gener – 7 de gener).
Descripció: Redacció final de la memòria del projecte i realització de la presentació.

Objectius:
- Tenir tota la documentació final del projecte preparada

Passos:
- Acabar de redactar la memòria del projecte.
- Realitzar la presentació.

La major part de les tasques identificades s'han pogut finalitzar amb força temps restant, pel que hi ha hagut un desplaçament efectiu d'unes dues setmanes, que s'han pogut dedicar a la millora de la robustesa del programa.

1.5 Breu sumari de productes obtinguts

Els principals productes obtinguts en aquest projecte són:

1. Un sniffer gràfic en entorn GNU complet i funcional.
2. Documentació rigorosa sobre el funcionament del sniffer.
3. Una memòria que exposa tot el que és rellevant per a comprendre les decisions preses al llarg del projecte.

1.6 Breu descripció dels altres capítols de la memòria

- Capítol 2, Conceptualització: Es tracta de descriure i definir el concepte d'analitzador de xarxa així com aportar els primers passos en el projecte com ho són els estudis d'analitzadors ja implantats i els protocols que hauran de ser capturats i analitzats.
- Capítol 3, Disseny: Es proporcionen els casos d'ús del projecte corresponents a la fase de disseny del projecte, així com el seu corresponent diagrama i el diagrama estàtic de classes.
- Capítol 4, Implementació: Es descriuen les decisions que es prengueren respecte al llenguatge de programació i les biblioteques de suport emprades, així com es parla dels problemes que van aparèixer i com es va efectuar la seva resolució.
- Capítol 5, Manual d'instal·lació i d'usuari: Aporta el manual d'instal·lació del sniffer així com un exemple del seu funcionament, que pot ser emprat a tall de manual.

2. Conceptualització

2.1 Definició

Un sniffer o analitzador de tràfic és un programa que captura el tràfic de paquets present en una xarxa i permet l'anàlisi d'aquests amb finalitats diverses. Aquest fet sol ser d'utilitat, entre moltes altres coses, per a detectar possibles anomalies en el si d'una xarxa, com per exemple, el fet de ser víctima d'un atac informàtic. Un altre possible ús força estès consisteix en l'obtenció de contrasenyes quan aquestes viatgen per la xarxa en pla, és a dir, sense cap forma de protecció aplicada. Per tant, podem dir que els analitzadors de xarxa tenen usos tant legítims com il·lícits.

Per a realitzar la seva tasca, un sniffer escolta el tràfic provinent d'una determinada interfície de xarxa, la qual, en general, és situada en mode promiscu a fi de poder visualitzar i analitzar tots els paquets rebuts, tinguin la destinació que tinguin.

Per a aconseguir-ho, òbviament, cal que la màquina on s'instal·li el programa tingui accés físic a aquest tràfic. És per a aquest motiu que les xarxes LAN en topologia en bus solen permetre l'anàlisi de tot el tràfic present, ja que el seu funcionament es basa en la inundació.

Com que s'intercepten tots els paquets existents en una xarxa determinada, es pot realitzar una anàlisi del seu contingut i flux que s'estén a tots els nivells de la pila TCP/IP. En efecte, es poden analitzar els continguts de paquets Ethernet, IP, TCP, HTTP, i a més d'obtenir informació estadística del tràfic i aplicar filtres de forma que es redueixi l'abast de la captura. L'abast dels analitzadors de xarxa com a eina és molt variable i dependrà d'allò que pretengui aportar als usuaris.

Per a exemple, un sniffer molt senzill podria simplement capturar i presentar els resultats a l'usuari mitjançant un terminal, mentre que un altre podria aportar un sistema d'alertes en cas que es detecti un tràfic inusual o fins i tot implementar alguna forma d'eina d'anàlisi que permeti seguir el flux corresponent a una determinada transacció que serveixi per a estudiar el comportament i composició d'un sistema distribuït mitjançant enginyeria inversa, etc.

A continuació, s'aporta una descripció del funcionament general d'un Sniffer, suposant el cas en el qual només s'incorpora la funcionalitat bàsica de captura de paquets:

- Captura: En una xarxa, tot paquet no deixa de ser una seqüència binària, és a dir, un conjunt de zeros i uns o bits, que han estat sotmesos a un tractament previ estandarditzat quant a senyalització i codificació. En aquest sentit, el primer pas que tot Sniffer realitza és situar la NIC (Network Interface Controller) objectiu de la màquina on resideix el programa en mode promiscu, de forma que el programa sigui capaç de rebre tots els paquets de la xarxa, en una seqüència binària.

- Conversió: Aquestes seqüències de bits són molt adequades per als computadors però no pas per a agents humans, de forma que el següent pas, un cop es disposa de les dades sobre cada paquet en binari, és convertir correctament aquestes dades, d'una forma que sigui intuïtiva i còmode per a humans.

- Anàlisi: Les dades prèviament capturades i convertides s'han de transformar en informació significativa. Aquesta informació pot incloure camps i valors en els protocols continguts en els paquets, de forma que l'usuari pugui entendre el contingut de cada paquet capturat.

Cal, necessàriament, establir alguna forma de disseminar els paquets, que es basi en el coneixement dels protocols suportats pel programa. Per a exemple, si interceptem trames Ethernet, aquests poden contenir un paquet IP, i al mateix temps aquest pot incloure un segment TCP i aquest últim pot incloure un HTTP i, per tant, cal ser conscients dels continguts i dimensions dels camps de cada un d'aquests protocols per tal de mostrar correctament la informació a l'usuari. Profunditzarem sobre aquest últim punt en l'apartat 2.3.

2.2 Anàlisi Sniffers existents

Wireshark (abans Ethereal): Es tracta d'un sniffer molt complet i estès, escrit en C i C++, que usa la biblioteca pcap (Packet Capture) i que es distribueix en forma de software obert i lliure, a través de la llicència GNU GPL, en la majoria de sistemes operatius actuals. Com ja s'ha comentat, es tracta d'una eina molt versàtil, molt completa i que ofereix un ventall molt ampli de protocols i opcions per a personalitzar la presentació de la informació. Ofereix, a més, una interfície gràfica altament personalitzable sobre QT, en les versions actuals:

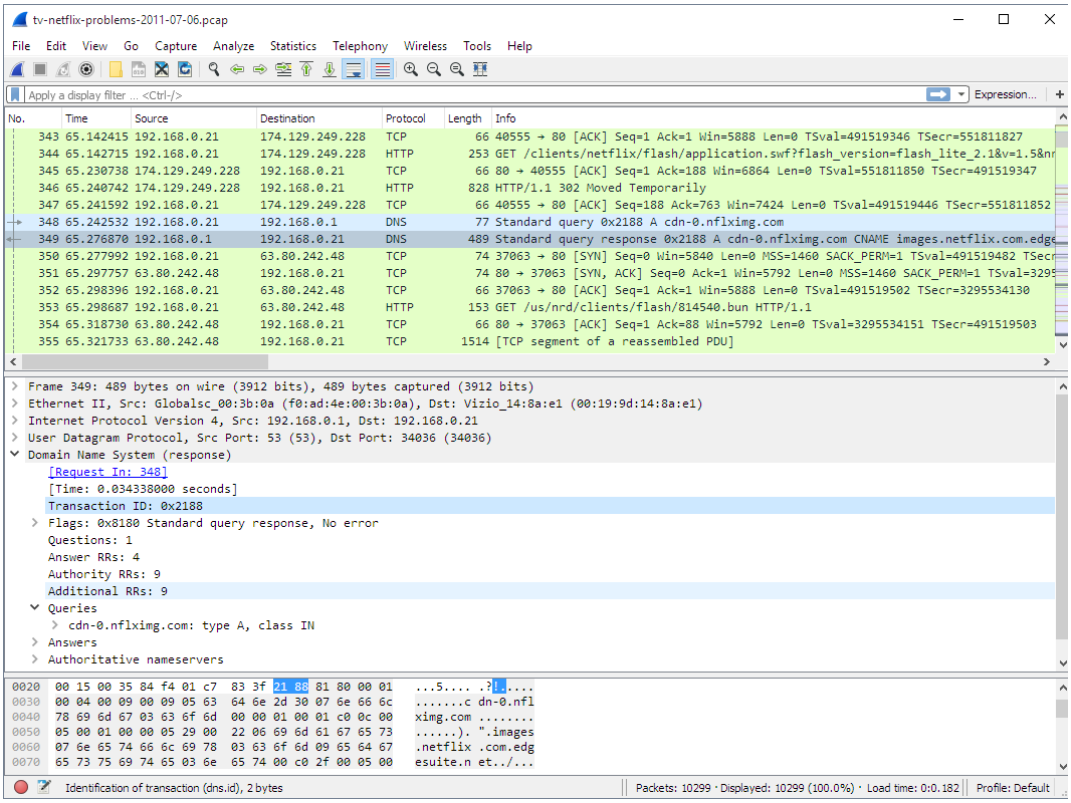


Figura 1: Pantalla principal Wireshark

A continuació, vegem amb una mica més de detall algunes de les opcions que aquest sistema ofereix, que es consideren especialment interessants:

Sistema de filtres de pantalla (display filters) que permet la incorporació d'expressions de filtratge, de forma que per a la pantalla principal només es recullin els paquets que compleixen amb els criteris seleccionats:

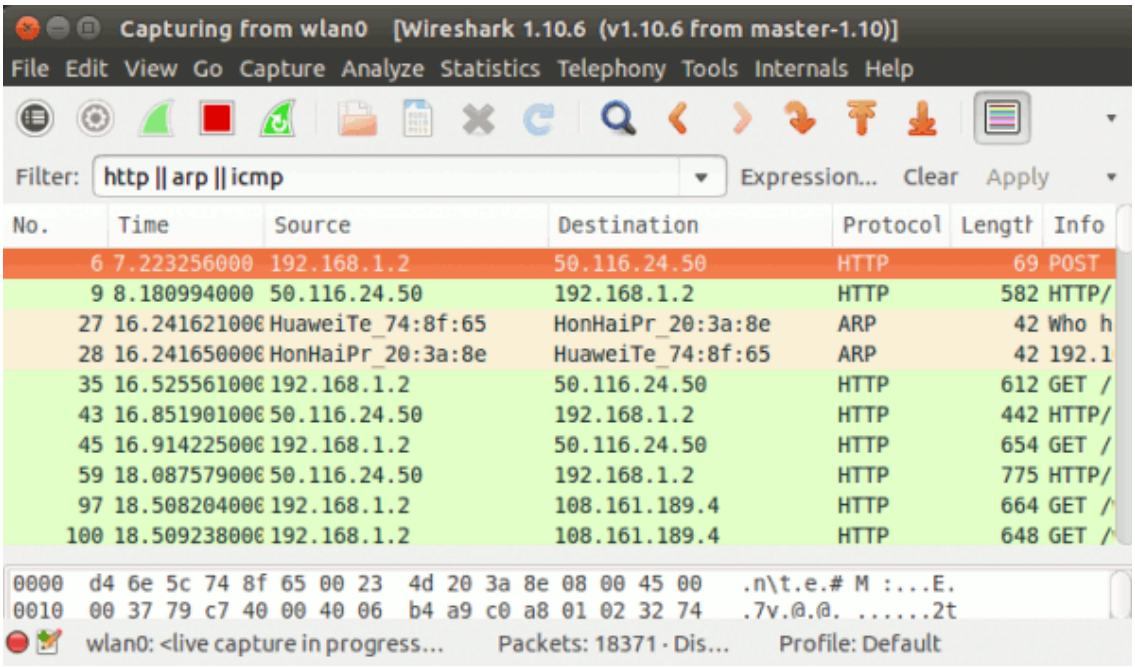
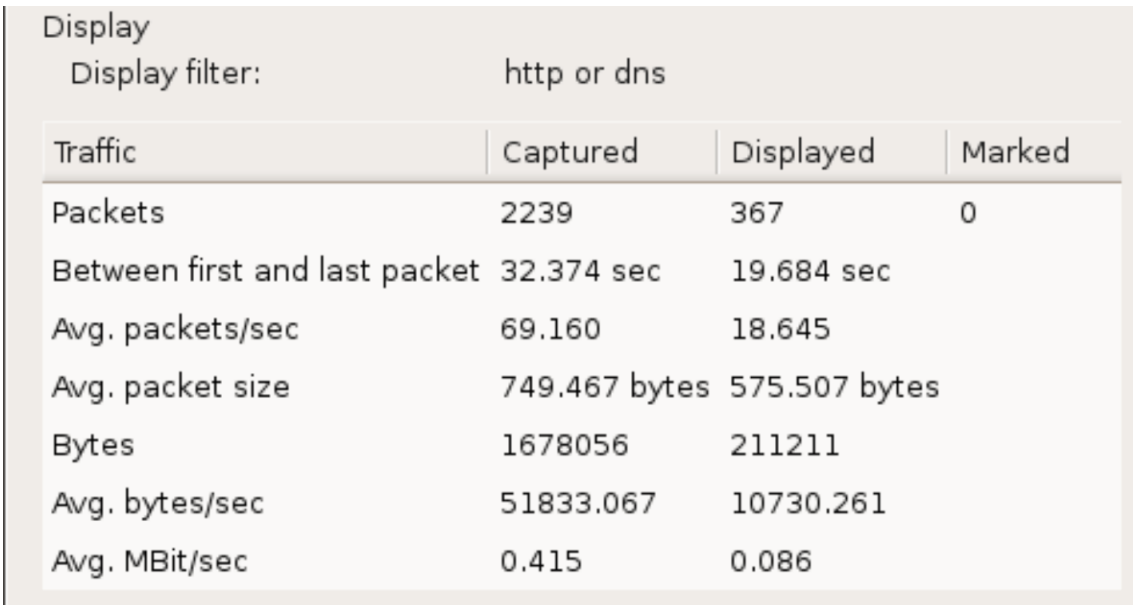


Figura 2: Filtres de pantalla Wireshark

Com veiem, l'usuari ha entrat l'expressió http || arp || icmp, que bàsicament implica que es mostrin els paquets HTTP, ARP, o ICMP. Cal comentar, que els filtres de pantalla en Wireshark poden ser aplicats i trets en qualsevol moment simplement esborrant l'expressió del camp corresponent.

Sistema rigorós d'estadístiques, que ofereix una quantitat molt gran d'estadístiques de tota mena, des de la disseminació de la quantitat de paquets capturats segons el protocol, passant per a les conversacions específiques, transaccions HTTP, flux TCP, tot amb un nivell variable de granularitat:



Display

Display filter: http or dns

Traffic	Captured	Displayed	Marked
Packets	2239	367	0
Between first and last packet	32.374 sec	19.684 sec	
Avg. packets/sec	69.160	18.645	
Avg. packet size	749.467 bytes	575.507 bytes	
Bytes	1678056	211211	
Avg. bytes/sec	51833.067	10730.261	
Avg. MBit/sec	0.415	0.086	

Figura 3: Informació estadística resum

L'anterior captura mostra la pantalla de resum d'estadístiques, i com veiem, considera els paquets que s'han capturat, els que s'estan mostrant actualment, el temps transcorregut, la mitja de paquets que es reben per unitat de temps, etc.

Sistema de logging, que permet desar, obrir, i crear noves sessions de captura. Aquesta opció és emprada per a poder bolcar els continguts capturats en una determinada sessió de captura a un fitxer de tipus PCAP, per a poder ser analitzats amb posterioritat, o fins i tot, ser compartits amb finalitats diverses, com per exemple per a permetre a estudiants poder entendre els continguts de determinats paquets o protocols, o per a permetre l'anàlisi externa de la seguretat d'una determinada xarxa. També permet exportar la sessió a diferents tipus de fitxer, com en text pla, i imprimir les captures:

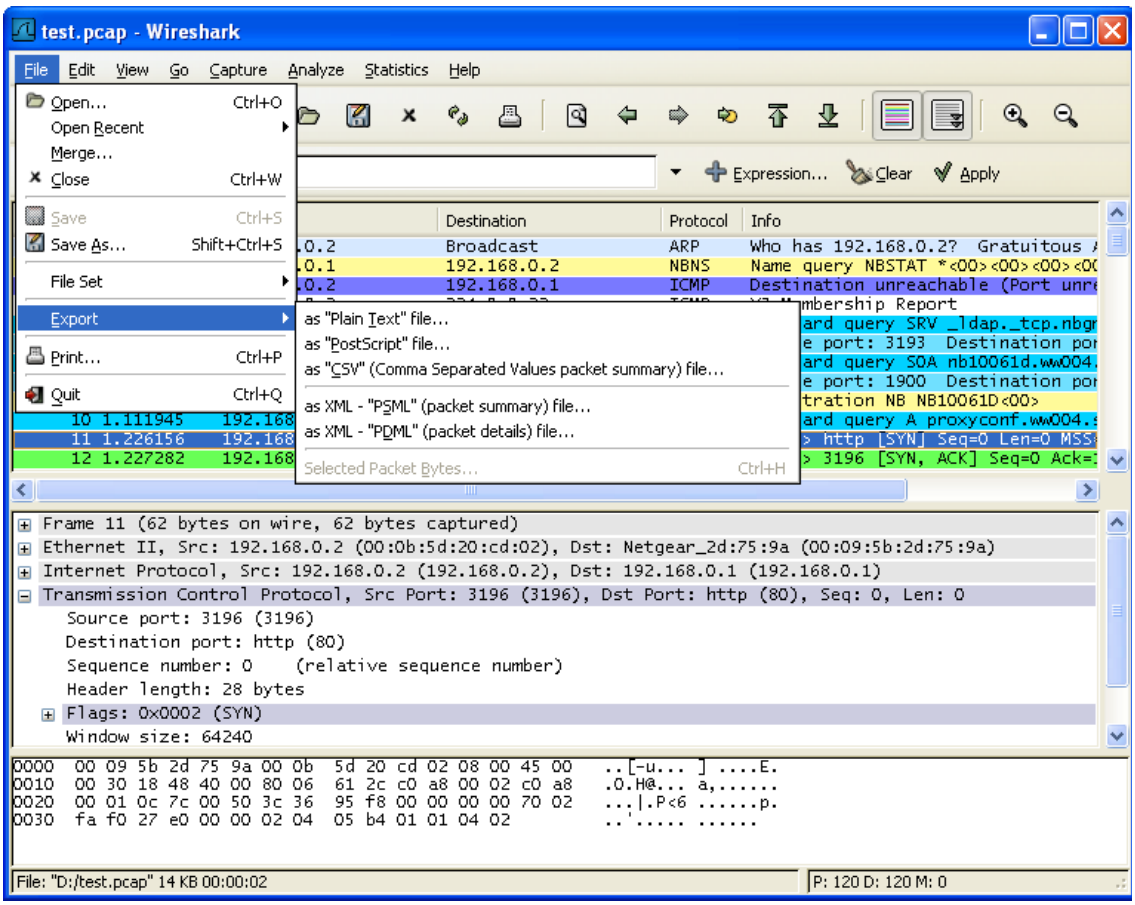


Figura 4: Menú File a Wireshark

Veiem que dins les opcions que s'ofereixen hi ha les que hem comentat.

D'altra banda, sobre la interfície gràfica del programa, convé comentar que aquesta no se sobrecarrega innecessàriament però que es poden afegir multitud d'opcions respecte a la presentació.

Vegem, a continuació, una captura del menú de configuració de la interfície gràfica d'usuari que implementa Wireshark:

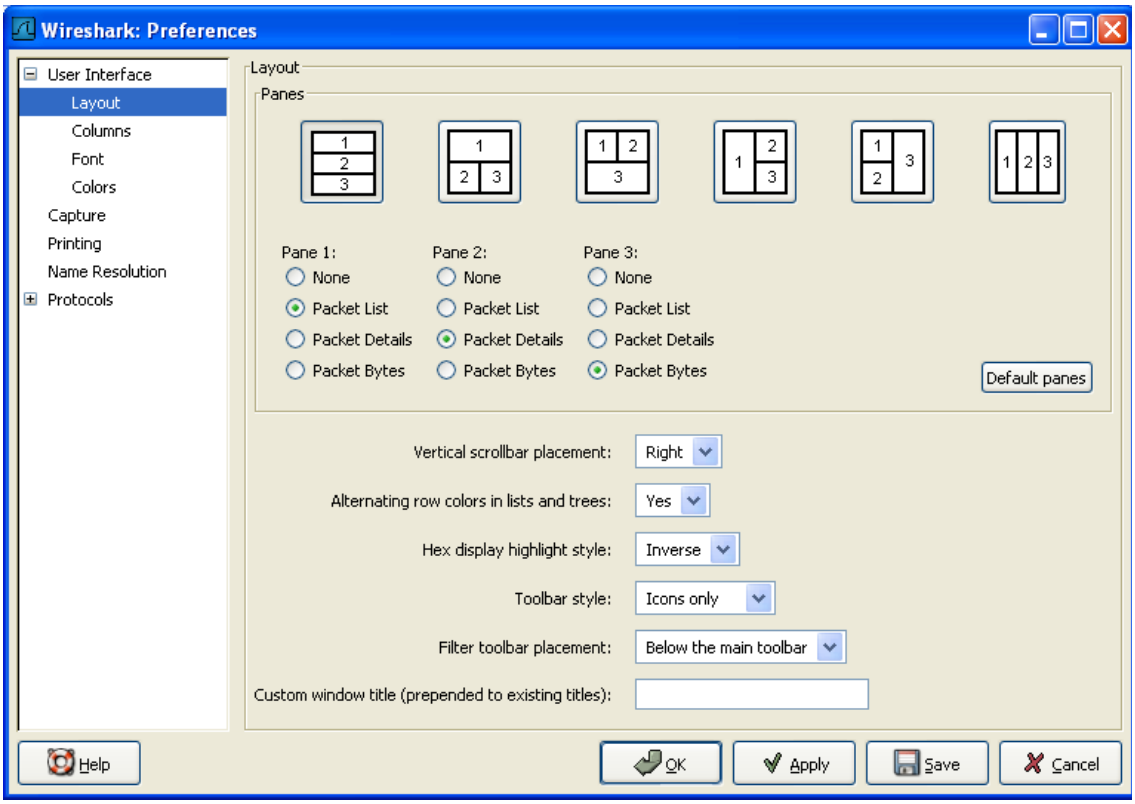


Figura 5: Opcions gràfiques Wireshark

Veiem, doncs, que permet definir la disposició de les pantalles, així sobre quina informació es mostra en cada una d'elles. També té opcions per colors, fonts, contingut de les columnes, a més de moltes altres coses.

Colasoft Capsa: Es tracta d'un sniffer molt complet, que també disposa de suport per a un nombre molt elevat de protocols, disponible actualment per al sistema operatiu Windows seguint un sistema de software propietari que contempla tres formes diferents, des del gratuït que segueix el model freeware fins al comercial destinat a empreses amb un cost de 995$.

Ofereix un gran nombre de característiques i opcions diferents, entre elles l'anàlisi de seguretat de xarxa i generació d'alertes, detecció de diversos problemes de xarxa a més de la recopilació de diverses estadístiques i la seva presentació en un format gràfic, i una interfície molt completa:

Figura 6: Pantalla principal Capsa

Com veiem, la interfície del programa proporciona una gran quantitat de gràfics i opcions.

Vegem, a continuació, alguns dels components d'aquest programa que, almenys en l'àmbit personal, es consideren especialment interessants:

Sistema d'estadístiques molt complet i amb una gran quantitat de gràfics per a ajudar a veure diverses coses, com per exemple en la següent captura, els paquets de sincronització TCP:

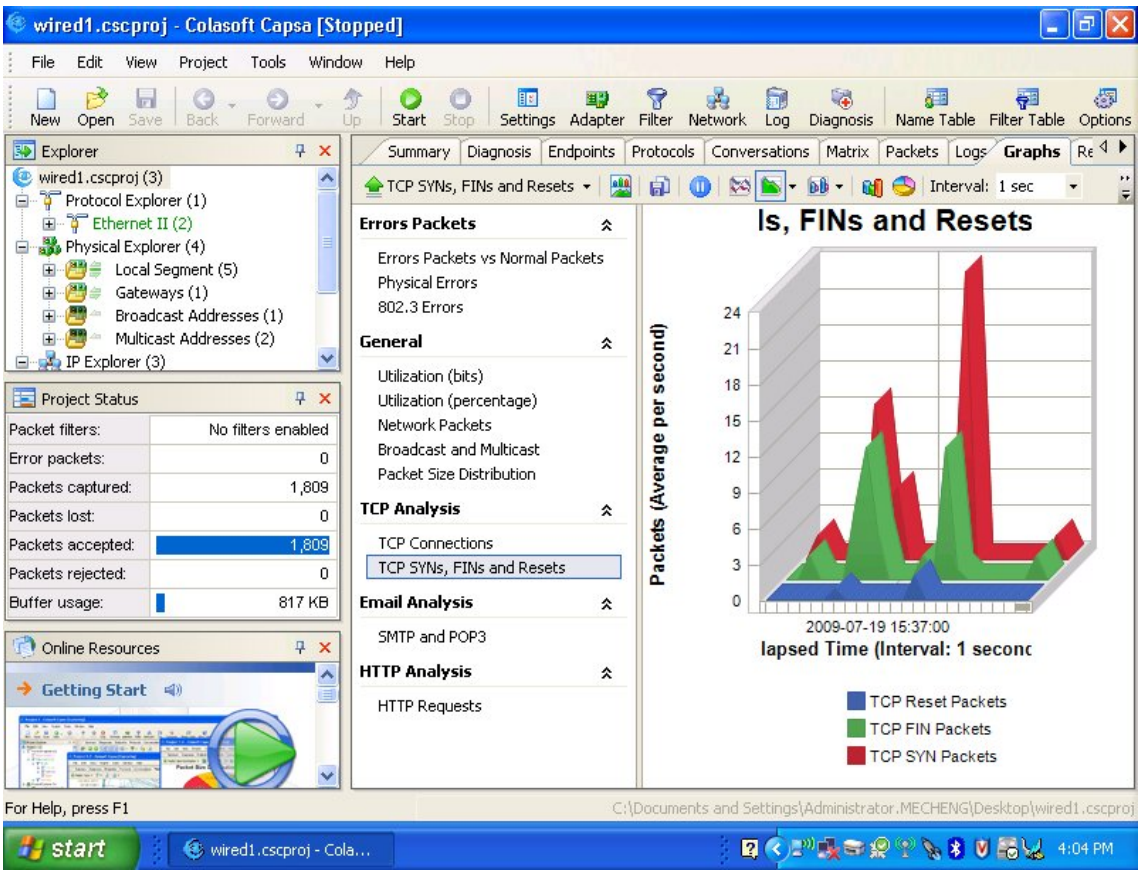


Figura 7: Capsa, alguna estadística

Destaca per la multitud de gràfics que ofereix, però també permet l'obtenció de moltes estadístiques resumides, tal com mostra la següent captura:

Dashboard | Summary | Diagnosis | Protocol | Physical Endpoint | IP Endpoint | Physical Conversation

Statistics Item					Current Value
⊞ **Alarm**					Trigger Count
⊞ **Diagnosis Statistics**					Count
⊟ **Traffic**	Bytes	Packets	Utilization	Bits Per Second	Packets Per Second
Total	37.621 MB	104,117	2.854%	28.541 Mbps	11,195
Broadcast	666 B	5	0.000%	0 bps	0
Multicast	1.688 KB	22	0.000%	0 bps	0
Average Size					378.889 Bytes
⊟ **Packet Size Distribution**	Bytes	Packets	Utilization	Bits Per Second	Packets Per Second
<=64	1.289 MB	21,113	0.126%	1.256 Mbps	2,453
65-127	3.203 MB	37,085	0.299%	2.985 Mbps	4,236
128-255	617.928 KB	3,269	0.067%	666.712 Kbps	419
256-511	6.450 MB	18,582	0.561%	5.609 Mbps	1,919
512-1023	6.927 MB	9,906	0.672%	6.716 Mbps	1,170
1024-1517	18.820 MB	13,934	1.130%	11.296 Mbps	997
>=1518	337.992 KB	228	0.001%	12.144 Kbps	1
⊞ **Address**					Count
⊞ **Protocol**					Count
⊞ **Flow**					Count
⊞ **TCP**					Count
⊞ **DNS Analysis**					Count
⊞ **Email Analysis**					Count
⊞ **FTP Analysis**					Count
⊞ **HTTP Analysis**					Count

Figura 8: Pestanya resum Capsa

Veiem que ofereix moltes dades, i moltes més que poden ser expandides fent clic als botons corresponents.

Amb tot, el sistema d'estadístiques no és l'únic pel qual aquest sniffer destaca sobre la resta. En efecte, disposa d'un sistema de captura molt complet i que contempla diferents tipus de conversació, amb diferent informació que es presenta i captura:

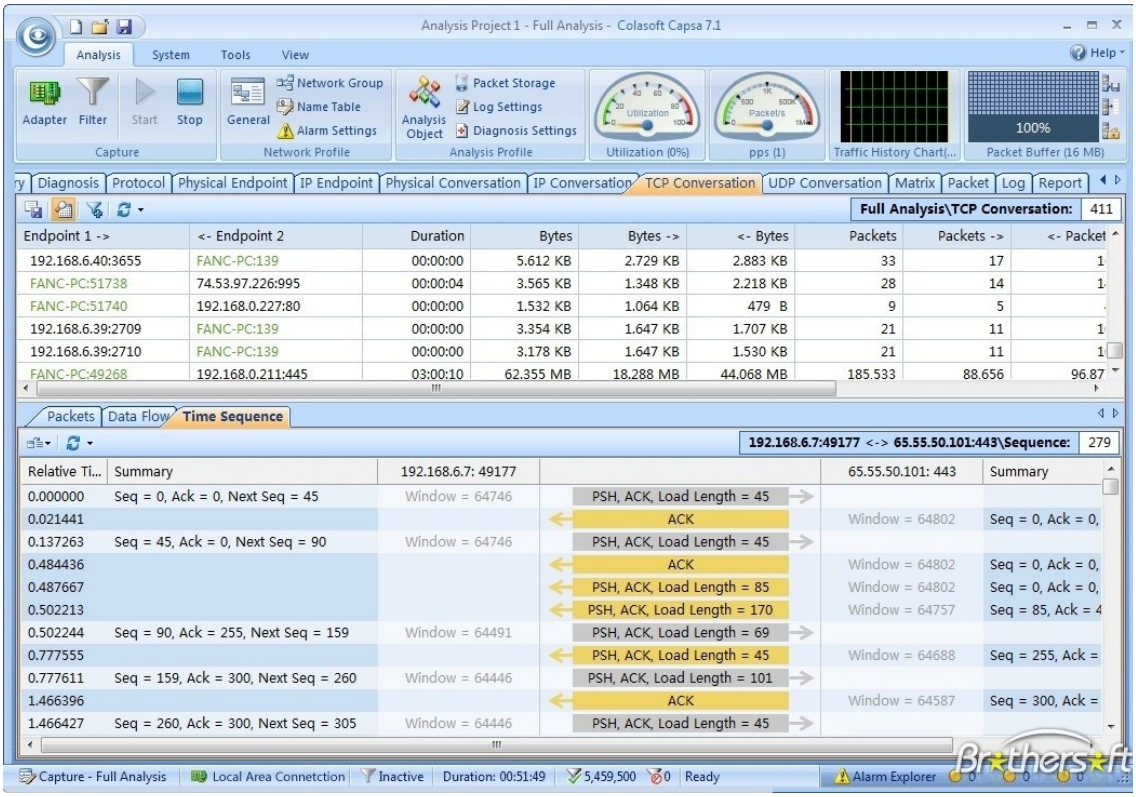


Figura 9: Captura Capsa

Veiem que en aquest cas s'estan analitzant les conversacions TCP, que s'agrupen per font, destinació i es mostra informació relativa a cada seqüència com a un tot i específicament. A tall d'exemple, a la captura podem veure que, de cada conversació, podem veure que ha tingut una determinada duració, s'han intercanviat un determinat nombre de bytes, sortints i entrants, agrupats en un nombre de paquets, on també podem veure els sortints i entrants. També és possible veure la seqüència de paquets intercanviats i informació relativa a cada un d'ells.

El programa també permet funcionar com un sniffer normal, seleccionant a tal efecte la pestanya Packet del programa:

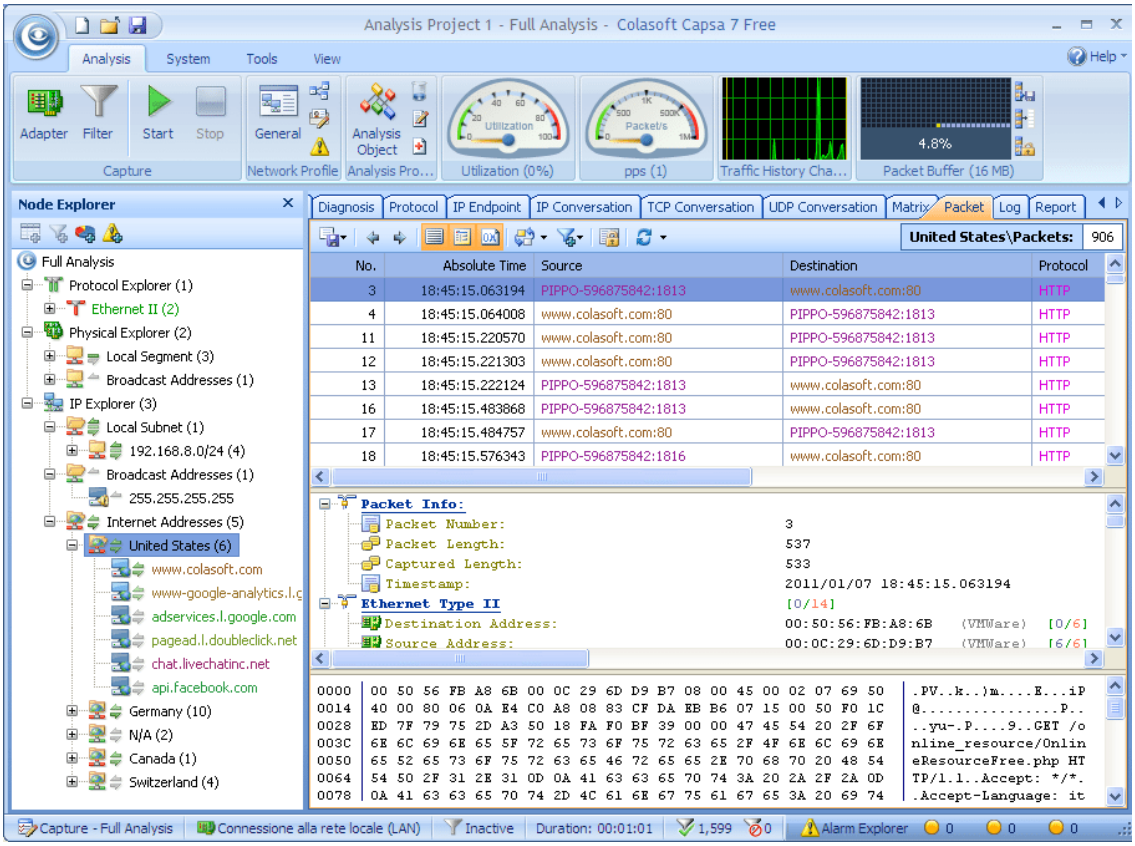


Figura 10: Capsa com a sniffer convencional

Podem dir que els dos analitzadors provats fins al moment, tots dos molt complets, adopten aproximacions diferents respecte a la interfície d'usuari. En efecte, mentre que Wireshark opta per a oferir una interfície atractiva i molt neta, i deixar opcions a l'usuari per a ampliar-la, afegir i usar totes les funcionalitats que el programa ofereix, que normalment romanen ocultes en contraposició de les opcions més importants i generals que són perfectament visibles per defecte, Capsa proposa oferir totes les funcionalitats diferents en la mateixa interfície i no ocultar-la de la mateixa forma que Wireshark.

Tcpdump: Un sniffer, també força popular i estès, creat en C i distribuït sota el model de free software, llicència BSD, que funciona en la majoria de sistemes basats en UNIX actuals. En principi, es tracta d'una eina que funciona primordialment en la línia d'ordres, pel qual no disposa, en principi, de cap interfície gràfica, tal com mostra la captura següent:

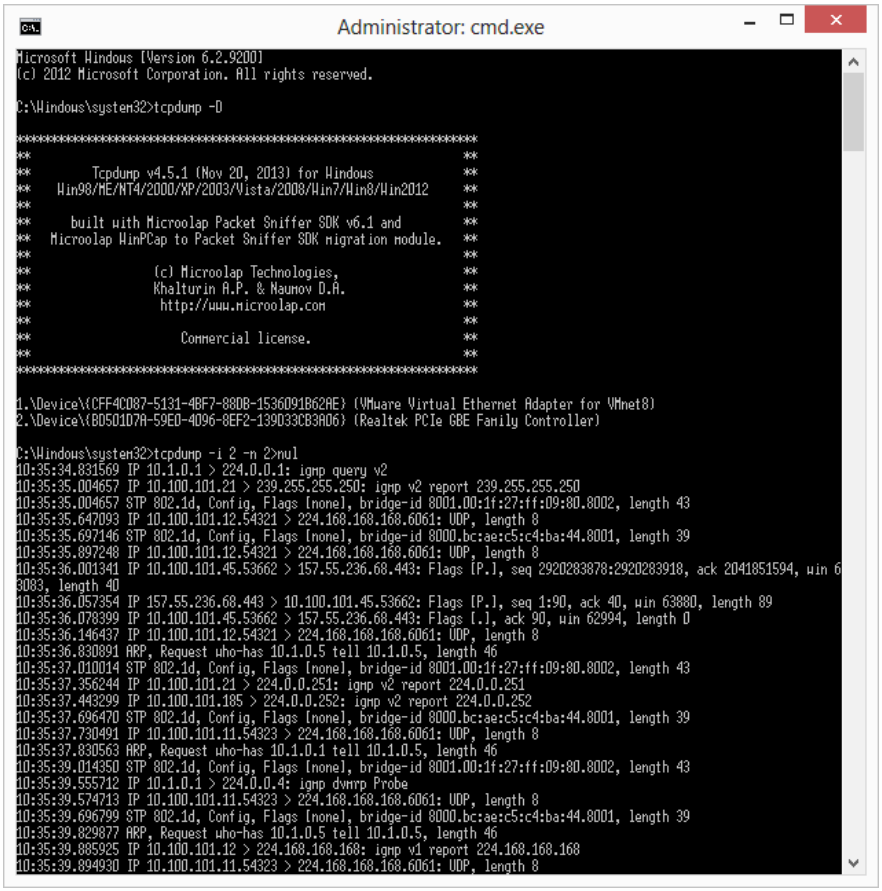


Figura 11: Pantalla principal cmd Windows executant tcpdump

La principal bondat d'aquest sniffer és la seva completesa i bon funcionament que feu que l'equip que el va crear va decidir extreure un conjunt de funcionalitats d'utilitat de la comanda i plasmar-les a una biblioteca, PCAP (Packet Capture), perquè aplicacions de xarxa poguessin emprar aquestes funcions per a facilitar-los la feina.

Podem dir, doncs, que aquesta comanda serveix, en cert grau, de base per a una gran quantitat d'aplicacions de xarxa, més enllà de programari analitzador de xarxa.

Justniffer: Es tracta d'un analitzador de xarxa distribuït sota la llicència GNU GPL, que es troba disponible per als principals entorns UNIX. Va néixer amb el propòsit de resoldre problemes amb serveis de TCP, pel que es considera un analitzador de xarxa TCP. La principal bondat d'aquest sniffer, que funciona en línia d'ordres, és que permet personalitzar el format de la sortida, capturar els continguts d'alguns protocols, com HTTP, o calcular el temps de resposta, que pot ser d'utilitat per a definir el rendiment d'un determinat servei.

Vegem una captura de la sortida del programa:

Figura 12: Sortida Justniffer

Tots els programes vistos fins al moment són considerats de propòsit general, és a dir, la seva tasca és la d'oferir un gran nombre d'opcions i suport per a molts protocols diferents. Tanmateix, existeixen diversos analitzadors que tracten de ser molt més especialitzats que els anteriors, per exemple centrant-se en un determinat protocol o conjunt reduït de protocols.

A continuació, vegem dos d'aquests sniffers:

dhcpdump: Es tracta d'una comanda de terminal centrada a mostrar els paquets DHCP en un format força accessible, de forma que se simplifiqui el procés de monitoratge o debugging. Internament, aquesta comanda confia en tcpdump.

Vegem la sortida de la comanda:

```
  TIME: 15:45:02.084272
    IP: 0.0.0.0.68 (0:c0:4f:82:ac:7f) > 255.255.255.255.67 (ff:ff:ff:ff:ff:ff
    OP: 1 (BOOTPREQUEST)
 HTYPE: 1 (Ethernet)
  HLEN: 6
  HOPS: 0
   XID: 28f61b03
  SECS: 0
 FLAGS: 0
CIADDR: 0.0.0.0
YIADDR: 0.0.0.0
SIADDR: 0.0.0.0
GIADDR: 0.0.0.0
CHADDR: 00:c0:4f:82:ac:7f:00:00:00:00:00:00:00:00:00:00
 SNAME: .
 FNAME: .
OPTION:  53 (  1) DHCP message type         3 (DHCPREQUEST)
OPTION:  54 (  4) Server identifier         130.139.64.101
OPTION:  50 (  4) Request IP address        130.139.64.143
OPTION:  55 (  7) Parameter Request List      1 (Subnet mask)
                                              3 (Routers)
                                             58 (T1)
                                             59 (T2)
```

Figura 13: Sortida dhcpdump

Podem veure, per tant, que únicament se'ns presenta informació sobre els paquets DCHP amb un format més llegible i comprensible.

httpry: Es tracta d'una eina que mostra i desa informació relativa a paquets HTTP, com ho és les pàgines que els usuaris visiten, cerca de patrons, detectar arxius potencialment perillosos o diverses estadístiques.

Vegem un exemple de l'execució de la comanda:

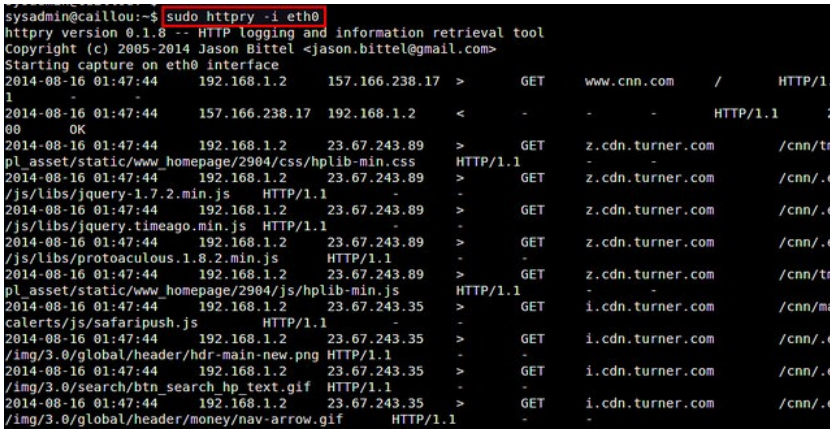


Figura 14: Sortida httpry

Com veiem, en aquest cas únicament es capturen paquets HTTP, en aquest cas entrants o sortints de la interfície eth0.

Una part per general positiva dels sistemes analitzadors de propòsit específic és que, o són molt simples i fan la seva feina molt eficientment, o permeten alguna anàlisi o funcionalitats que els de propòsit general no es poden permetre pel fet de ser molt menys focalitzats.

Podem veure una taula que resumeix els aspectes més rellevants de cada un dels programes analitzats:

	Wireshark	Capsa	Tcpdump	Justsniffer	dhcpdump	httpry
Avantatges	Molt complet, interfície neta, sistema de filtres gràfics, alta personalització	Molt complet, sistema d'estadístiques molt profund i atractiu	Molt eficient i complet, serveix com a base de bona part dels sniffers actuals, a més d'altres aplicacions de xarxa	Permet personalitzar el format de sortida i explorar aspectes més propis dels sistemes més específics	Major nivell de focalització vers a DHCP i format de sortida més agradable	Major nivell de focalització vers a HTTP, el que permet anàlisi molt específic del protocol
Desavantatges	-	Interfície gràfica poc intuïtiva en estar molt carregada	No disposa d'interfície gràfica	No disposa d'interfície gràfica	No disposa d'interfície gràfica	No disposa d'interfície gràfica

S'ha vist que els sniffers més emprats incorporen sistemes de filtratge, del que prendrem com a model a Wireshark, que incorpora filtres de captura i filtres de pantalla, sistemes d'estadístiques per a facilitar l'anàlisi, sistemes de logging per a desar i obrir sessions de captura i per descomptat els sistemes de captura amb diverses opcions disponibles, com ho és l'activació del mode promiscu o la quantitat de paquets que s'ha de capturar. Totes aquestes opcions i característiques interessaran pel nostre projecte.

2.3 Tipus de tràfic capturat

En primer lloc, esdevé necessari definir a què ens hem estat referint al llarg del present document amb "trànsit". En una xarxa de computadors, en la qual els nodes es comuniquen únicament mitjançant l'intercanvi de missatges, el tràfic o trànsit s'entén com el conjunt de missatges emesos/rebuts per a un determinat nombre de computadors.

Aquests missatges segueixen unes normes estrictes i, possiblement, estandarditzades quant a contingut i format segons un determinat protocol. De protocols n'hi ha de molts diferents i que s'empren per a unes tasques o unes altres, però en el si d'una xarxa la millor classificació d'aquests protocols és prenent una eina conceptual, la pila de protocols, parant, nosaltres especial atenció a la pila TCP/IP:

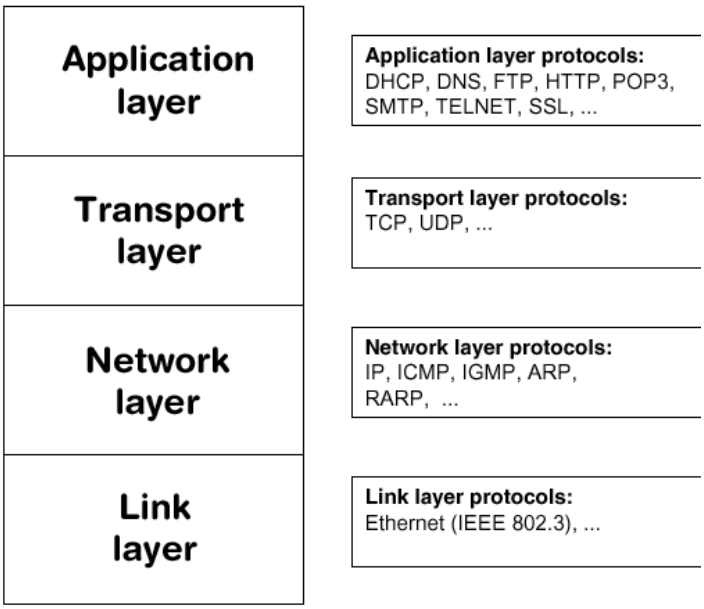


Figura 15: Pila TCP/IP

Podem veure que aquesta pila disposa de quatre capes principals:

- **Capa d'aplicació:** Conté les dades de rellevància pròpies d'una aplicació en concret, com ho poden ser el tipus d'intercanvi, com per exemple un HTTP GET per a obtenir un document Web, formularis d'accés en FTP, i, ja en general, tot allò que pugui servir per a definir, segons el que s'estableix en el protocol, els passos que s'han de seguir, és a dir, per a decidir que cal fer i com fer-ho a nivell de programa o aplicació. Exemples de protocols a nivell d'aplicació són HTTP, FTP o DNS.

- **Capa de transport:** Manté informació com el port d'origen i destinació, i pot incloure tota mena d'informació de control que serveixi per a fer un seguiment de cada paquet amb l'objectiu de retransmetre'l o controlar-ne el flux. Aquesta informació sobre els ports és necessària perquè, actualment, sobre una sola màquina poden haver-hi diversos serveis d'aplicació funcionant simultàniament

i cal establir una forma de decidir on enviar cada paquet que entra per a una determinada interfície del computador. En són exemples TCP o UDP.

- **Capa de xarxa:** En aquesta capa s'hi troba informació relativa a com un paquet ha d'arribar del host origen al de destinació. A tal efecte, s'inclou informació sobre l'adreçament lògic global així com l'encaminament. En són exemples IPv4, IPv6 o ICMP.

- **Capa d'enllaç de dades:** Capa que s'encarrega d'independitzar les capes superiors dels detalls específics de cada tecnologia. Tracte de disposar cada trama en el medi així com d'extreure aquestes trames del medi. Pot incloure informació sobre la integritat de les trames i prendre decisions al respecte. N'és un exemple Ethernet.

Els protocols d'Internet s'articulen en capes perquè, com en qualsevol projecte informàtic de mida considerable, ajuda a modularitzar el desenvolupament, el que el simplifica i en millora la possible mantenibilitat, ja que s'independitzen, en cert grau, una capa de les altres.

Totes les capes són necessàries per a assegurar que les dades que es volen comunicar són intercanviades amb garanties en una xarxa com Internet. En aquest sentit, convé comentar el procés d'encapsulament/desencapsulament que es duu a terme per a comunicar dos nodes en la xarxa. A mode il·lustratiu, a continuació es pot veure una imatge que resumeix el procés que se segueix:

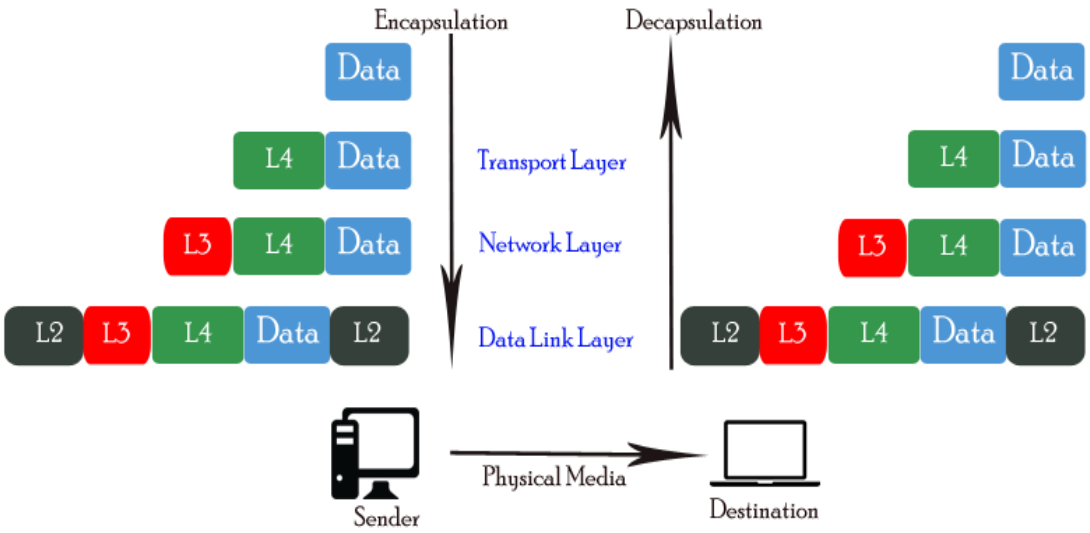


Figura 16: Encapsulació/Desencapsulació

El primer que es fa és, com ja s'ha comentat, és considerar que es disposa d'unes dades que són regides per a un determinat protocol a nivell d'aplicació, que cal comunicar a un altre node a fi que es proporcioni algun determinat servei, com ho pot ser un document HTML si ens trobem en el context de HTTP i el Web.

Per tal que aquestes dades arribin correctament al seu destí i, d'entre els serveis disponibles en la màquina destí, arribin a l'aplicació corresponent, cal afegir informació de transport en forma de capçalera, que, a més, pot proporcionar certa confiabilitat en la comunicació. S'ha produït doncs, la primera encapsulació, consistent en afegir una capçalera a les dades formant una PDU (Protocol Data Unit) de transport.

A aquest segment cal afegir informació perquè es pugui dirigir el paquet des de l'origen al destí a través d'una xarxa global com ho és Internet. Aquesta informació és necessària per als diferents mecanismes d'encaminament existents que tracten cada paquet per a poder decidir on enviar-lo perquè, en última instància, arribi a la xarxa de destinació. Aquesta és la funció dels protocols de xarxa com IP i aquests encapsulen les PDU de transport formant una nova PDU.

En una xarxa local, interessa independitzar els protocols superiors dels detalls específics de cada tecnologia i medi físic, alhora que es controla l'accés a aquest medi i potser es resolen errors. Aquesta és la funció principal de la capa més baixa de la pila TCP/IP, la d'enllaç, que encapsula el paquet resultant de la capa de xarxa en una nova unitat anomenada trama.

Un cop es disposa de la trama, el node emissor pot enviar el paquet al seu encaminador, que desencapsularà el trama, fixant-se en el paquet IP i prenent decisions sobre el tractament que se n'ha de fer, per exemple enviar-lo, formant primer un nou trama, a un encaminador veí segons un determinat protocol d'encaminament.

Finalment, al destí es rebrà el paquet i es produirà el procés complet contrari a l'encapsulatge, és a dir, que en última instància s'acabarà obtenint, sempre i que tot sigui correcte, les dades a nivell d'aplicació que contenien el missatge de l'emissor, que es pot passar a l'aplicació corresponent, per exemple un servidor Web, perquè realitzi el tractament del missatge.

Com ja s'ha comentat, existeixen molts protocols que tracten de donar resposta a multitud de requisits diferents. Donada la complexitat de capturar informació relativa a tots els protocols existents, es considerà imprescindible reduir el nombre de protocols a capturar. Concretament, es va prendre la decisió de capturar informació sobre els protocols següents:

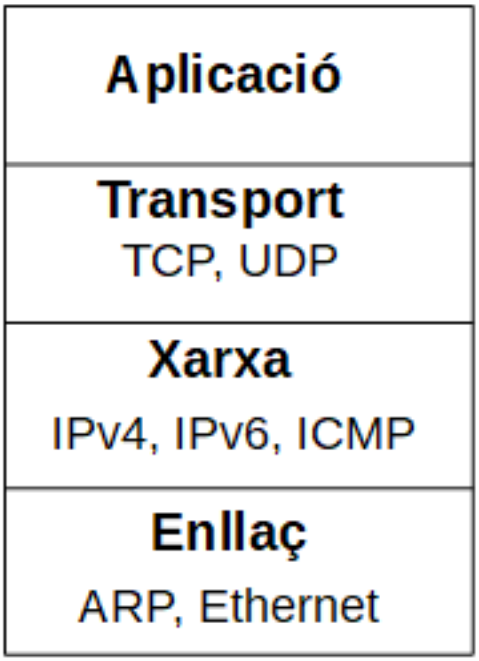


Figura 17: Esquema protocols seleccionats

TCP

Un protocol de transport que conté informació sobre l'estat de la connexió i que permet retransmetre els paquets en cas que sigui necessari. Per a aquest motiu, diem que el protocol ofereix confiabilitat. A més, hi ha altres dades que permeten establir el flux de l'intercanvi de forma que s'equilibrin emissor i receptor i es redueixi la sobrecàrrega de la xarxa en cas de pèrdua de paquets.

UDP

Un protocol de transport que, a diferència de TCP, no manté informació sobre l'estat, no és orientat a connexió i, per tant, no és confiable. Això implica que, per a tràfic que requereix integritat, aquest protocol no sigui recomanable perquè no assegura que els paquets arribin a la seva destinació ni que aquests siguin reordenats en cas que arribin fora d'ordre.

La motivació principal del protocol rau en la velocitat, no s'empra informació sobre l'estat i no es limita el flux, el que el fa idoni per a aplicacions multimèdia que tolerin possible pèrdua de paquets, com ho és el streaming de vídeos.

IP

És el protocol de nivell de xarxa principal a Internet. S'encarrega de definir l'adreçament i tota aquella informació que permeti que un paquet arribi a la seva destinació, en el si d'una xarxa de commutació de paquets i de màxim esforç. El protocol IP fa la seva feina eficientment, pel qual tampoc manté informació sobre l'estat o correcció d'errors sinó que delega aquestes funcions a altres capes.

IPv6

La versió de protocol de IP més estesa és la versió 4. Amb tot, la dimensió en bits de l'adreça font i de destí és de 32 bits, el que permet un màxim de 2^{32} adreces diferents. Aquest nombre, en el moment de la creació del protocol semblà suficient, però amb l'aparició de tendències com la IoT l'exhauriment d'adreces va esdevenir un problema. Per a resoldre aquest problema i d'altres, es creà la versió 6 d'IP.

ICMP

És un protocol de suport emprat molt habitualment per a l'enviament de missatges d'error en situacions com la incapacitat d'arribar a un determinat host o el TTL ha expirat, per exemple.

ARP

En Internet, tot dispositiu requereix dues adreces força diferents, una física, pròpia d'una determinada interfície i que permet diferenciar uns nodes d'altres en el si d'una xarxa en local, i una adreça lògica, que permeti crear i definir rutes a un nivell més global.

En aquest sentit, el protocol ARP s'encarrega d'establir una traducció entre adreces lògiques i físiques. Normalment, es parteix sobre la base del coneixement de l'adreça lògica, així que el que interessa obtenir és l'adreça física. Essent una mica més específics, per exemple, partim del coneixement d'una determinada adreça IP, diguem la 192.168.0.1 i volem saber la direcció MAC corresponent a aquest host, que després dels missatges corresponents podem saber que és, per exemple, 11-22-33-44-55-66.

Ethernet

Un protocol de nivell d'enllaç de dades emprat per a la creació de xarxes d'accés local (LAN) cablejades. Inclou informació sobre l'adreçament físic, mitjançant les conegudes adreces MAC, així com informació relativa al protocol contingut (entre altres i per exemple, IP).

Els protocols seleccionats són molt freqüents i representen una bona tria que equilibra utilitat i complexitat. Es pot veure el contingut exacte dels protocols i les seves capçaleres a l'apartat Annexos de la present memòria, concretament a l'annex 1.

3. Disseny

En la fase de disseny de tot projecte de desenvolupament de software s'hi engloben totes aquelles activitats orientades a definir i descriure tots els components de la solució del sistema, la seva interacció i plasmar tots els aspectes rellevants, sense entrar en detalls tècnics, que es concretaran posteriorment en la fase d'implementació, en un seguit d'artefactes com diagrames, models i documents.

Aquests documents inclouen el diagrama de casos d'ús, la descripció textual d'aquests casos d'ús, que els detallen, així com el diagrama estàtic de classes, que tracta de definir totes les entitats importants pel domini que s'està considerant:

- Els casos d'ús ajuden a la conceptualització del problema que cal resoldre i la forma en la qual aquest es resol, detallant els passos seguits per a realitzar alguna tasca concreta. Els diagrames de casos d'ús serveixen per a descriure, de forma gràfica, el comportament del sistema en relació a les interaccions amb altres sistemes o actors.

- Els diagrames de classes serveixen per a definir els components del sistema i la relació entre ells. Normalment aquests diagrames poden realitzar-se seguint una determinada especificació o llenguatge, com ho és UML. Són d'utilitat per a conceptualitzar millor el problema i la seva solució.

Ambdós artefactes han estat creats per al present projecte i en aquest capítol és on es presenten aquests documents previs a la implementació. També s'aporta la definició de la interfície gràfica, que es considera un procés previ a la implementació de la resta de mòduls del sistema.

3.1 Diagrama estàtic de classes

Podem veure el diagrama estàtic de classes tot seguit. Aquest descriu a alt nivell els components o mòduls del sistema i la relació existent entre ells:

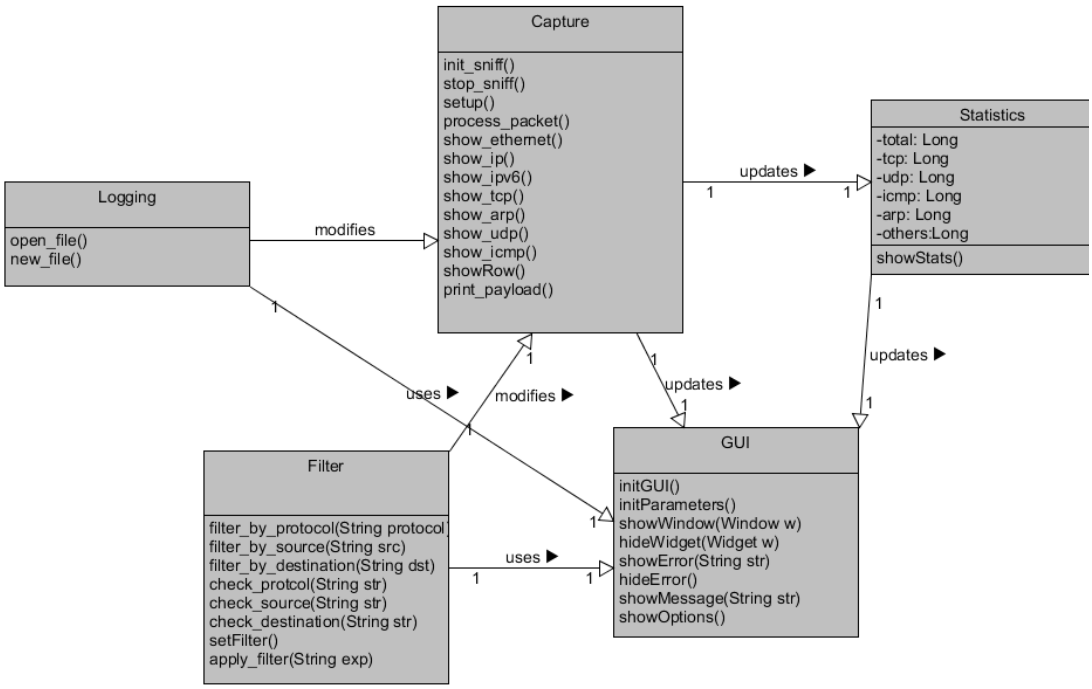


Figura 18: Diagrama de classes

S'han identificat 5 mòduls corresponents al GUI, al sistema de filtratge, al de Logging, al d'estadístiques i, per descomptat, al de captura. El sistema de captura inclou tot el que és necessari per a capturar paquets i mostrar-los mitjançant l'actualització del GUI. Actualitza, a més, el sistema d'estadístiques, el qual manté i mostra, actualitzant també la GUI, modificant els termes sobre el nombre de paquets i el comptatge per tipus de protocol.

El sistema de Logging pot modificar el sistema de captura, per exemple en obrir un fitxer, i empra la GUI per a poder mostrar els diàlegs de selecció corresponents. També confia en el sistema de GUI el mòdul de filtres, que necessita obrir un diàleg perquè l'usuari entri l'expressió de filtratge desitjada i, a més, pot modificar el sistema de captura fent que només es capturin un determinat tipus de paquets. Els filtres gràfics únicament modificaran la GUI.

3.2 Diagrama de casos d'ús

El diagrama de casos d'ús té com a principal objectiu mostrar gràficament la relació entre actors i els casos d'ús que es detallaran textualment:

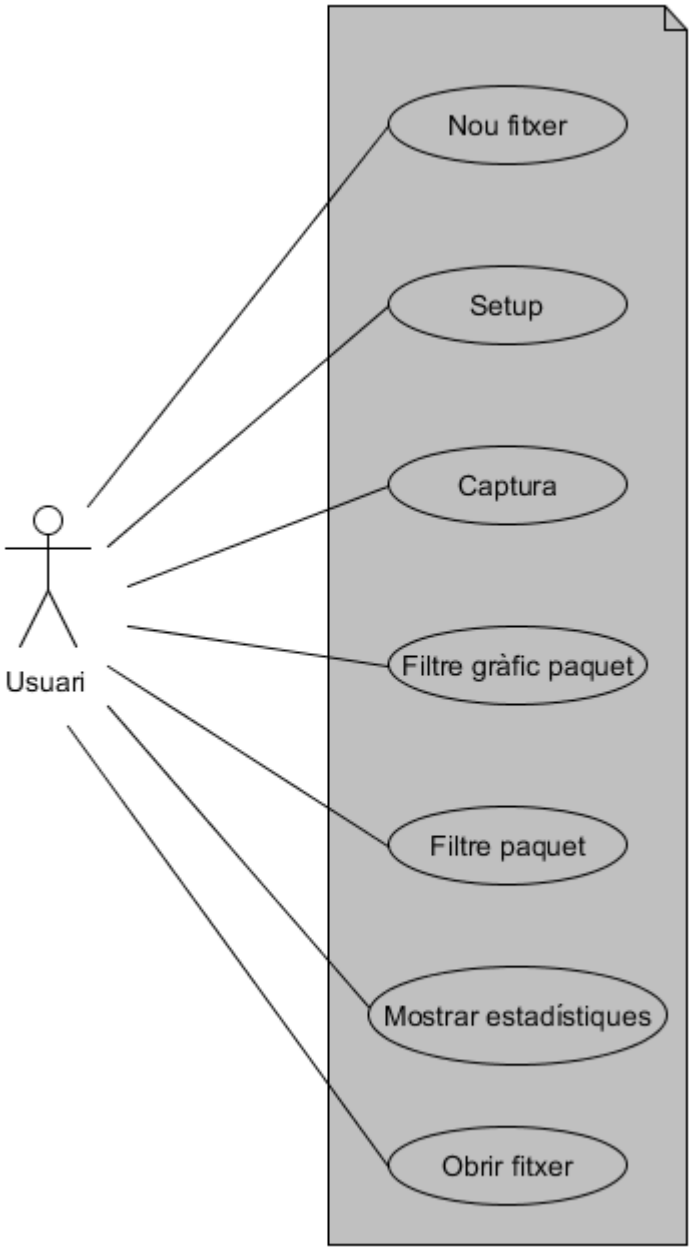


Figura 19: Diagrama de casos d'ús

3.3 Descripció textual dels casos d'ús

Cas d'ús 1: "Posada en marxa (setup)"
Resum: L'usuari selecciona primer un dispositiu, si vol que funcioni en mode promiscu, si vol limitar la quantitat de paquets a capturar i si vol definir un nom de fitxer per a disposar el resultat de la captura en un fitxer.

Flux d'esdeveniments principal:
1. L'usuari selecciona, d'entre els botons disponibles, el de Setup.
2. El sistema mostra una pantalla amb un selector d'interfície, una opció per a activar el mode promiscu, un camp per a poder limitar la quantitat de paquets a capturar, així com un camp per a definir un nom de fitxer.
3. L'usuari selecciona, d'entre la llista d'interfícies disponibles, una amb suport Ethernet, i les opcions respecte el mode promiscu, el límit i el nom de fitxer.
4. L'usuari prem el botó per a acceptar la configuració.
5. El sistema activa l'opció per a començar la captura de paquets.
6. El cas d'ús finalitza.

Flux d'esdeveniments alternatiu:
3.a L'usuari selecciona una interfície no Ethernet i prem acceptar.
3.b El sistema mostra un error.
3.c El cas d'ús finalitza.

Cas d'ús 2: "Captura"
Resum: L'usuari selecciona l'opció per iniciar la captura i el sistema mostra informació diversa sobre cada paquet capturat.

Flux d'esdeveniments principal:
1. L'usuari prem el botó per a iniciar la captura de paquets.
2. El sistema comença a capturar paquets i mostra un resum del contingut de cada un d'ells per a la pantalla principal.
3. El sistema mostra la informació detallada del primer paquet (capçaleres i payload) per a les pantalles secundàries.
4. El sistema segueix capturant paquets fins que l'usuari acaba l'execució del programa o premi l'opció de parar captura.
5. El cas d'ús finalitza.

Flux d'esdeveniments alternatiu:
4.a L'usuari selecciona una de les files.
4.b El sistema mostra informació detallada sobre aquell paquet per a les pantalles secundàries.
4.c Torna al pas 4.

Cas d'ús 3: "Filtre paquet"
Resum: L'usuari selecciona l'opció per a afegir filtres de captura, i el sistema els aplica, de forma que només es capturin els paquets que compleixen les condicions establertes.

Flux d'esdeveniments principal:
1. L'usuari selecciona l'opció per a aplicar filtres.
2. El sistema mostra una pantalla amb un camp per a inserir l'expressió de filtratge.
3. L'usuari entra l'expressió desitjada i prem el botó per a aplicar-la.
4. El sistema registra l'expressió i, de ser vàlida, aplica el filtre de forma que els paquets capturats responguin al criteri introduït.
5. El cas d'ús finalitza.

Cas d'ús 4: "Filtre gràfic paquet"
Resum: L'usuari selecciona l'opció per a aplicar filtres gràfics i entra una expressió de filtre gràfic al camp corresponent i prem el botó per a aplicar-ho. El sistema mostra per pantalla els paquets que responen al criteri introduït i, a més, aplica un filtre de captura.

Flux d'esdeveniments principal:
1. L'usuari selecciona l'opció per a aplicar filtres gràfics.
2. El sistema mostra una pantalla amb un camp per a inserir l'expressió de filtratge.
3. L'usuari entra l'expressió desitjada i prem el botó per a aplicar-la.
4. El sistema registra l'expressió i, de ser vàlida, aplica el filtre de forma que els paquets que es mostren responguin al criteri introduït.
5. El sistema aplica, a més, un filtre de captura perquè els pròxims paquets que es capturin segueixin responent al criteri establert.
6. El cas d'ús finalitza.

Cas d'ús 5: "Obrir fitxer"
Resum: L'usuari selecciona, en un menú, l'opció per a obrir un fitxer de bolcat. Es mostra la informació continguda per a ser analitzada.

Flux d'esdeveniments principal:
1. L'usuari selecciona l'opció per a obrir un fitxer.
2. El sistema mostra una pantalla amb un diàleg per a entrar el nom i la ubicació del fitxer o seleccionar-lo d'entre els disponibles visualment.
3. L'usuari selecciona un fitxer i prem l'opció per a obrir.
4. El sistema obre el fitxer en qüestió i restaura la sessió de captura, de forma que l'usuari pugui analitzar el que es va capturar amb anterioritat.
5. El cas d'ús finalitza.

Flux d'esdeveniments alternatiu:
3.a L'usuari cancel·la l'obertura del fitxer.
3.b El sistema tanca el diàleg i no obre cap fitxer.
3.c El cas d'ús finalitza.
4.a El fitxer no és del tipus que hauria que ser.
4.b El sistema mostra un error.
4.c El cas d'ús finalitza.

Cas d'ús 6: "Nou fitxer"
Resum: Es neteja tot el contingut anterior per a iniciar una nova sessió de captura.

Flux d'esdeveniments principal:
1. L'usuari selecciona l'opció per a crear una nova sessió.
2. El sistema mostra una pantalla de confirmació.
3. L'usuari confirma que vol iniciar una nova sessió.
4. El sistema neteja totes les pantalles i fa les operacions pertinents per a eliminar tot rastre de l'anterior sessió.
5. El cas d'ús finalitza.

Flux d'esdeveniments alternatiu:
3.a L'usuari cancel·la l'operació
3.b El cas d'ús finalitza.

Cas d'ús 7: "Mostrar estadístiques"
Resum: L'usuari mostra la seva voluntat per a obtenir diferents estadístiques i el sistema les proporciona.

Flux d'esdeveniments principal:
1. L'usuari selecciona l'opció per a mostrar estadístiques.
2. El sistema presenta una pantalla amb diferents estadístiques rellevants.
3. El cas d'ús finalitza.

3.3 Creació de la interfície gràfica

Pel que fa a l'etapa de disseny de la interfície gràfica, es va partir de la premissa de la completesa i a la vegada senzillesa. En aquest sentit, se seguí una aproximació semblant a programaris com l'Office, mantenint molt visibles les opcions més importants i populars, mentre que es permetin altres opcions, en un menú, i per tant no són visibles directament.

Podem veure, a continuació, una captura que mostra la interfície gràfica principal del programa:

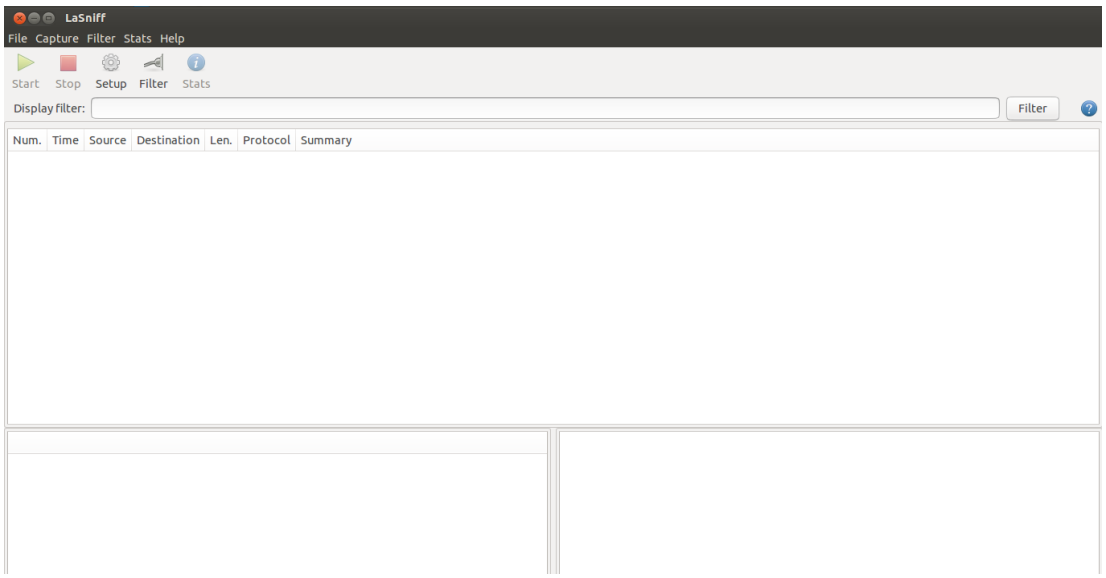


Figura 20: Interfície LaSniff

Com es pot veure, s'han creat tres pantalles:

- Pantalla principal: Es mostren tots els paquets capturats, presentant a l'usuari informació bàsica sobre el contingut de cada un d'ells. Aquesta informació bàsica inclou un nombre, corresponent al comptatge del paquet, una marca de temps, l'origen del paquet, la seva destinació, la seva longitud, el protocol corresponent i una breu descripció del contingut del paquet.

- Pantalla header: En aquesta pantalla, situada a la part de sota a l'esquerra, es mostra les capçaleres del paquet seleccionat, en forma d'arbre. Per defecte, el primer paquet que es mostra per la pantalla principal, i fins que l'usuari seleccioni una fila, és del que es mostrarà inicialment les capçaleres en aquesta pantalla.

- Pantalla data: En aquesta pantalla, situada a l'extrem inferior dret de la finestra principal, es mostra el paquet seleccionat en hexadecimal i en un format més llegible. Per defecte, tal com passa amb la pantalla header, es mostra el primer paquet capturat.

Convé comentar que també per simplificat i per a fer el programa més intuïtiu i homogeni, s'optà per a replicar algunes funcionalitats, de forma que a més de ser-hi a la barra d'eines en considerar-se opcions importants, també hi són presents al menú.

També en relació a guiar l'usuari, s'han afegit algunes seccions d'ajuda, principalment per a l'aplicació de filtres, tant gràfics com de captura. A més, s'han inhabilitat alguns botons en certs moments a fi que l'usuari tingui una idea del que pot i ha de fer en cada moment.

Cal comentar, a més, que s'ha dissenyat la interfície pensant en ser emprada en mode pantalla completa a fi que l'usuari tingui una major visibilitat de la sortida del programa.

4. Implementació

4.1 Llenguatge de programació

Una de les primeres consideracions quant a la implementació del programa fou el llenguatge de programació en la qual aquesta s'havia de realitzar. Aquesta decisió limitaria força el rang de possibles solucions aplicables.

Hi ha multitud de llenguatges de programació, amb una API rica i que per a determinats projectes simplifiquen molt la tasca de desenvolupament, sia pel paradigma que segueixen o per la quantitat de software ja creat i plasmat en biblioteques.

En aquest sentit, en un primer moment es valoraren dues opcions, motivades principalment per l'experiència prèvia que es posseïa en ells, força diferents:

Opció 1	Opció 2
ANSI C	Java
Programació estructurada	Programació orientada a objectes
Implementació compilada molt sòlida	Implementació interpretada
Portabilitat no gaire elevada	Gran portabilitat
Gestió de memòria per programador	Gestió de memòria intèrpret
Permet codi molt eficient	El codi pot no ser tan eficient
Programació potencialment complexa	Programació potencialment més fàcil

Finalment, i motivat primordialment per a la qüestió de l'eficiència i el fet que diversos productes importants han estat creats en C, s'optà per al llenguatge C. Es tracta, com ja s'ha comentat, d'un llenguatge molt potent i, donada la seva popularitat i el fet que és la base de molts altres llenguatges, la documentació disponible per a la plataforma, així com biblioteques de suport que facilitin el desenvolupament serà rica i completa. Aquest punt és força important, ja que, de seleccionar un llenguatge no massa popular, a més del procés d'aprenentatge requerit i la necessitat de crear molt més codi, seria més complex trobar la solució a qualsevol problema o dubte que pugui aparèixer en el transcurs normal del procés d'implementació.

4.2 Biblioteques de suport emprades

Hi ha diverses vies per a implementar un sniffer funcional en el llenguatge C. Per a exemple, es pot fer simplement amb sockets RAW i cridar a tots aquells mètodes (com ho són les crides de sistema) que ens ofereixi GNU/Linux i C per a la detecció d'interfícies, la disposició d'aquestes interfícies en mode promiscu, la generació de fitxers, filtratge, etc.

Amb tot, s'optà per a simplificar lleugerament aquestes tasques mitjançant la utilització de la biblioteca PCAP (Packet Capture). Aquesta va ser desenvolupada per l'equip de Tcpdump, i ajuda a aplicacions de xarxa, no només sniffers, a realitzar la tasca de captura de paquets, la de logging i la de filtratge.

D'aquesta biblioteca, que és molt popular i emprada, tal com ja s'ha vist, per a multitud de sistemes existents, ens interessa especialment la part que permet la introducció i processament d'expressions de filtratge, segons la seva pròpia sintaxi, el qual faria força més fàcil el procés d'implementació dels filtres de captura. A més, també ens interessa la que permet l'obertura de fitxers de captura en el seu format, el que simplificaria molt el procés de logging, ja que no caldria definir un format propi.

A més de PCAP, s'han emprat algunes biblioteques més, per a simplificar, per exemple la utilització de capçaleres com IP, TCP, ICMP, UDP o Ethernet. En aquest sentit, ha resultat especialment important emprar diverses definicions contingudes en la biblioteca netinet, que defineixen estructures corresponents a les capçaleres ja comentades amb la major part dels seus camps. D'aquesta forma ens estalviàvem redefinir aquestes estructures que d'haver optat per una definició pròpia els resultats haurien estat força semblants.

Finalment, per a la realització de la interfície gràfica d'usuari (GUI) s'optà per a emprar la biblioteca GTK, en conjunció amb el programari Glade.

En entorns GNU/Linux i pel llenguatge de programació seleccionat, les dues opcions més prominents quant a biblioteques gràfiques són el GTK+ i el QT. D'entre les dues solucions, GTK+ es troba molt més estesa en C i en GNU en general, fet pel qual s'optà per a aquesta biblioteca, ja que, entre altres motius, la seva popularitat faria que el procés d'obtenció d'ajuda i documentació resultés bastant més senzill.

A més a més, durant la fase de familiarització i selecció de possibles solucions s'empraren eines d'ajuda en els respectius entorns, i va poder ser constatat una major facilitat, almenys en l'àmbit personal, en la creació de la interfície amb l'ajuda del programari Glade per a la definició i posicionament d'objectes gràfics.

4.3 Decisions i problemes

S'han desenvolupat els 5 mòduls principalment, plasmats en els corresponents fitxers amb extensió .h i .c. És a dir, un mòdul per a la GUI, un per a les captures, un pels filtres gràfics, un per a obrir i crear noves sessions de captura (logging) i el d'estadístiques. Per a la correcta comunicació i funcionament amb GTK, s'ha optat per a definir dues estructures principals de cabdal magnitud:

1. La primera, capture, aglutina totes les variables necessàries, tant gràfiques com d'estat, per a la correcta implementació del sistema. Això inclou tant botons, pantalles, buffers, i en general tota mena de recursos GTK, com variables com tcp, str_filter o src, orientades a controlar i contenir diversos paràmetres, com ho són la quantitat de paquets TCP capturats, l'expressió de filtratge o la font del paquet.

2. La segona, pck, manté tota la informació necessària per a emmagatzemar una mena de base de dades molt simple que conté els paquets i el seu contingut. L'objectiu principal d'aquesta estructura és el de poder ampliar i analitzar els paquets, més enllà de mostrar-los. Per exemple, necessitem accedir a aquesta estructura quan s'aplica un filtre gràfic o quan el clica una fila per a mostrar més informació sobre el paquet corresponent.

D'altra banda, el desenvolupament del projecte ha estat un constant procés d'aprenentatge. En ell, la principal font de problemes ha estat el mecanisme de filtres gràfics, que provocava errors de segmentació per la forma en la qual es gestionaven, especialment pel protocol ARP. Finalment, s'ha optat per a fer que en afegir ARP com a filtre de protocol, el procés de captura s'aturi fins que l'usuari acabi esborrant el filtre, i faci clic a Start.

De fet, per qüestions d'homogeneïtat, d'una banda, i el fet que van aparèixer problemes amb la conjunció de filtres gràfics i filtres de captura, de l'altre, es va repensar la funció dels mateixos filtres gràfics. Es passà a considerar que aquests són una forma de filtrar allò que es mostra per a qüestions d'anàlisi així que, de fet, el procés de captura s'hauria d'aturar per a poder realitzar l'anàlisi i assegurar que només es mostren aquells paquets que compleixen amb les condicions establertes. El procés es podria reprendre quan s'esborrés el filtre.

D'altra banda, inicialment van aparèixer problemes de segmentació atesos a condicions de cursa creades pel desconeixement de la presència de diversos threads, que van ser resolts mitjançant la utilització d'una funció implementada per gtk pel tractament de threads i concurrència en la funció process_packet.

Val a dir que, per a la implementació de la interfície gràfica, ha estat necessari emprar Glade únicament en algunes parts, com per exemple per a la definició i disposició dels components gràfics, mentre que altres parts, com la connexió de senyals per a la gestió i interacció amb l'usuari, donaven problemes en alguns casos.

5. Manual d'instal·lació i d'usuari

5.1 Manual d'instal·lació

Podem veure un exemple pràctic que il·lustra el funcionament de l'analitzador desenvolupat:

El primer que cal fer és òbviament la instal·lació del programa. A tal efecte cal assegurar-se que es disposa d'una instal·lació de PCAP i GTK. Generalment la majoria de distribucions de GNU/Linux ja disposen d'una instal·lació de GTK. Si no fos el cas, es pot trobar informació sobre el procés d'instal·lació en el lloc web del projecte:

https://www.gtk.org/

En tot cas, el que sí que es necessita per a instal·lar el programa són les eines de desenvolupament GTK, instal·lables a través de la següent comanda:

sudo apt-get install libgtk-3-dev

D'altra banda, si no es disposa de PCAP, cal instal·lar-lo entrant la següent comanda en un terminal:

sudo apt-get install libpcap-dev

Tot seguit, un cop es compleixen tots els requisits, es pot instal·lar el programa entrant la comanda:

sh INSTALL.sh

I per a executar-lo, l'usuari ha d'entrar (suposant que es troba al directori d'instal·lació):

./LaSniff

Convé comentar, que per a un correcte funcionament del programa, cal que l'usuari disposi de permisos de superusuari. A tal efecte, la comanda que cal entrar per a la correcta execució del programa és *sudo ./LaSniff* si bé es recomana entrar la comanda *sudo su* per a mantenir l'estat de superusuari per tota la sessió de treball.

5.2 Manual d'usuari

El primer que es presenta a l'usuari és la pantalla inicial:

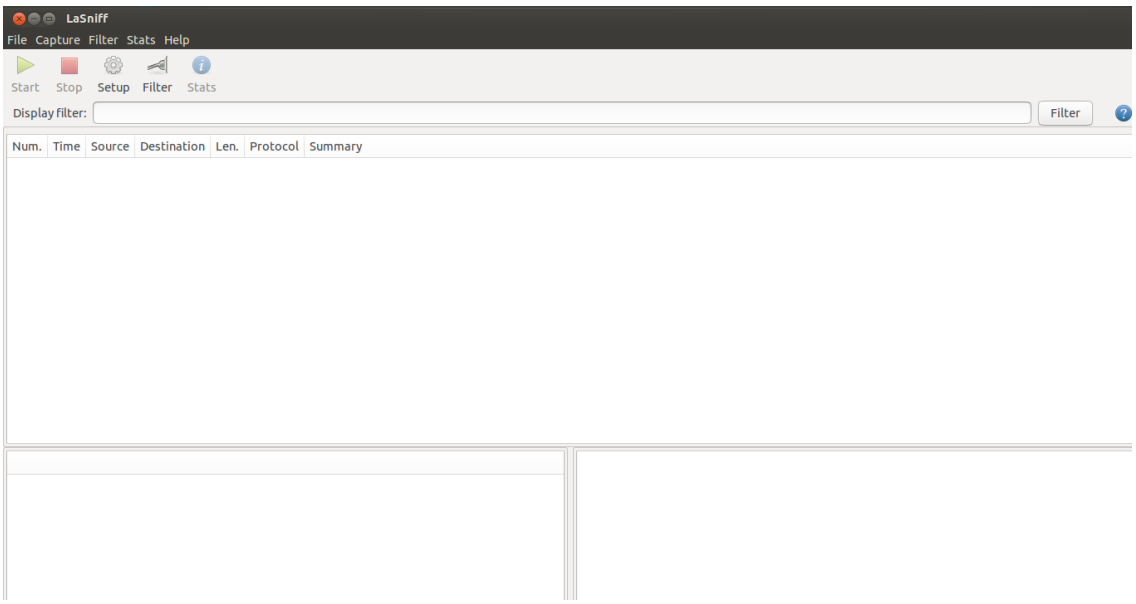


Figura 21: Interfície general

En aquest punt, l'usuari pot obrir una captura ja existent, anant al menú File → Open File i seleccionant d'entre els disponibles un amb el format correcte. Convé comentar que, en tot cas, el programa gestiona que l'usuari seleccioni un fitxer amb una extensió i format incorrectes.

Alternativament, l'usuari pot emprar l'opció Setup des del botó corresponent o mitjançant el menú Capture → Setup per a configurar una nova sessió de captura.

En aquest cas, s'obre la següent pantalla:

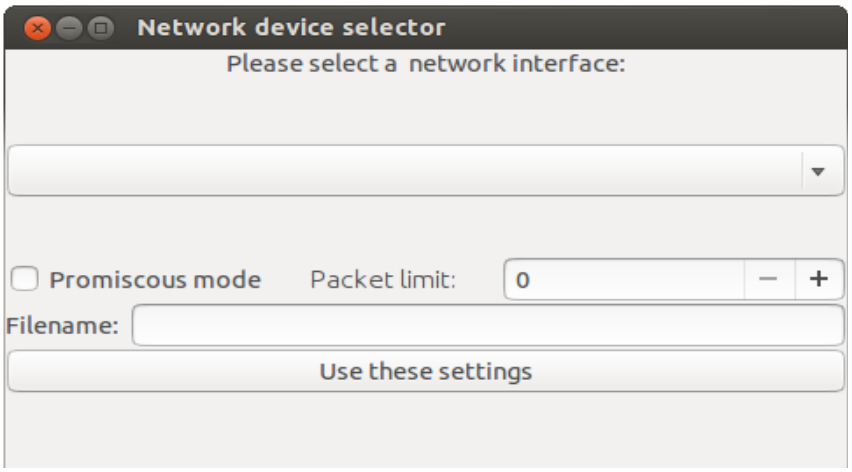


Figura 22: Pantalla Setup inicialment

- El primer desplegable mostra a l'usuari una llista de les interfícies de xarxa disponibles al sistema. Aquesta ha de ser necessàriament de tipus Ethernet o equivalent. Si l'usuari no selecciona una interfície correcta, el programa mostra un missatge d'error.

- L'opció Promiscous mode serveix per a activar, o no, el mode promiscu per a la interfície.

- L'usuari pot, a més, disposar un límit en la quantitat de paquets capturats de forma que quan s'arriba al nombre de paquets capturats, simplement s'aturi la captura.

- Finalment, el camp de text a la part inferior serveix perquè l'usuari entri un nom de fitxer per si vol desar la present sessió de captura en un fitxer per a una anàlisi posterior.

Un cop seleccionades totes les opcions desitjades, l'usuari pot prémer el botó Use these settings per a ocultar la present pantalla i aplicar la configuració:

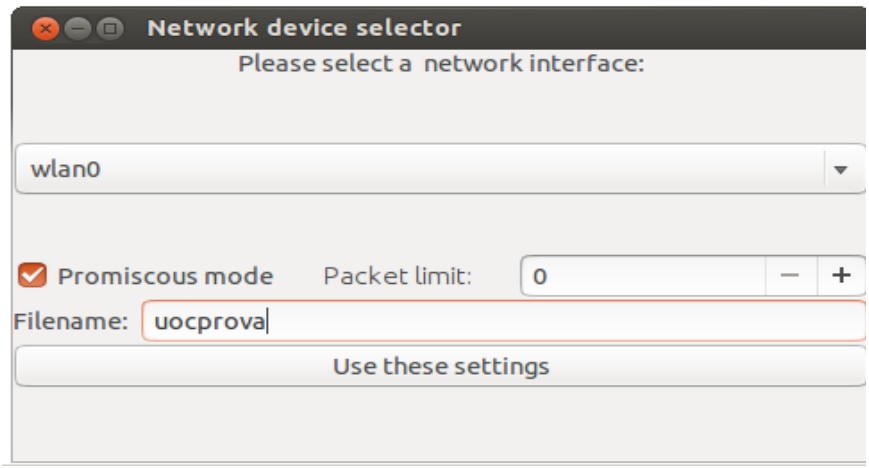


Figura 23: Pantalla Setup valors establerts

L'usuari podrà, a partir d'aquest moment, iniciar el procés de captura fent clic al botó Start, que ara es troba en estat sensible:

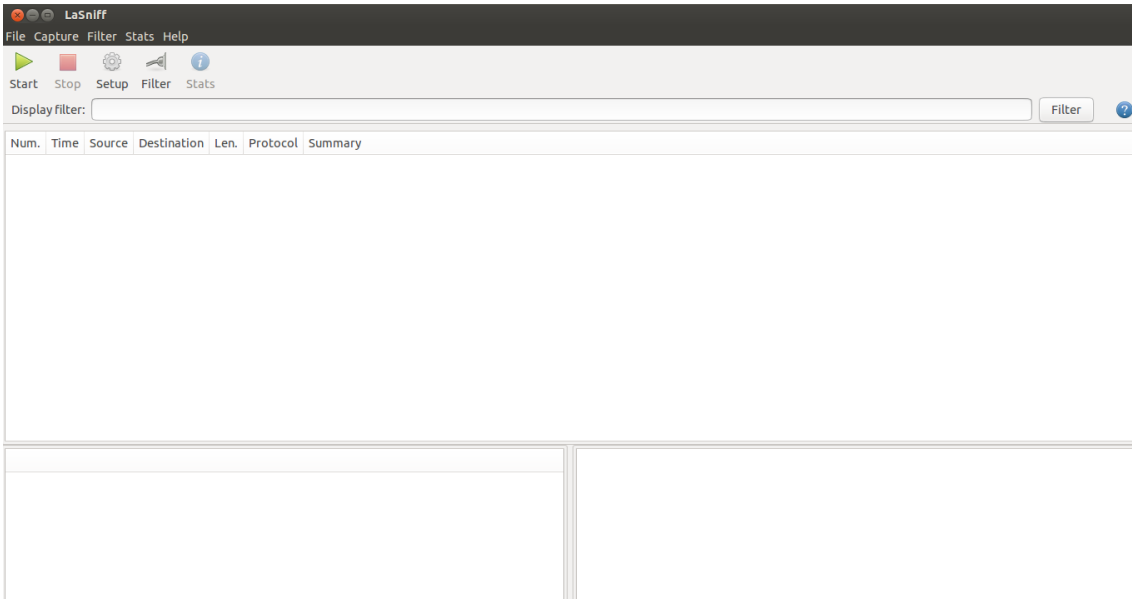


Figura 24: Botó Start disponible

L'usuari pot, a més, afegir expressions PCAP com a filtres de captura, de forma que el programa únicament capturi aquells que compleixen amb els criteris seleccionats per l'usuari:

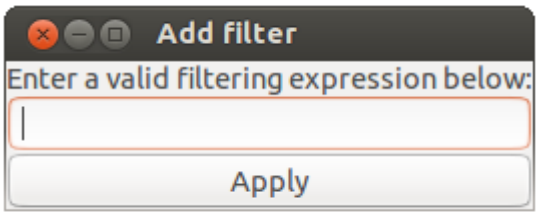


Figura 25: Pantalla filtres captura

Cal entrar una expressió pcap vàlida (el programa inclou un enllaç a una pàgina amb informació al respecte) i prémer la tecla Enter o fer clic a Apply perquè el programa enregistri l'expressió.

Convé comentar que els filtres de captura poden ser entrats en qualsevol moment, fins i tot quan una nova sessió de captura ja ha estat configurada i iniciada. Tot el que cal fer és aturar la captura amb el botó Stop i entrar l'expressió desitjada. Quan es premi de nou Start, només es capturaran els paquets que compleixin amb les restriccions especificades.

En prémer el botó Start es dóna inici a la captura de paquets:

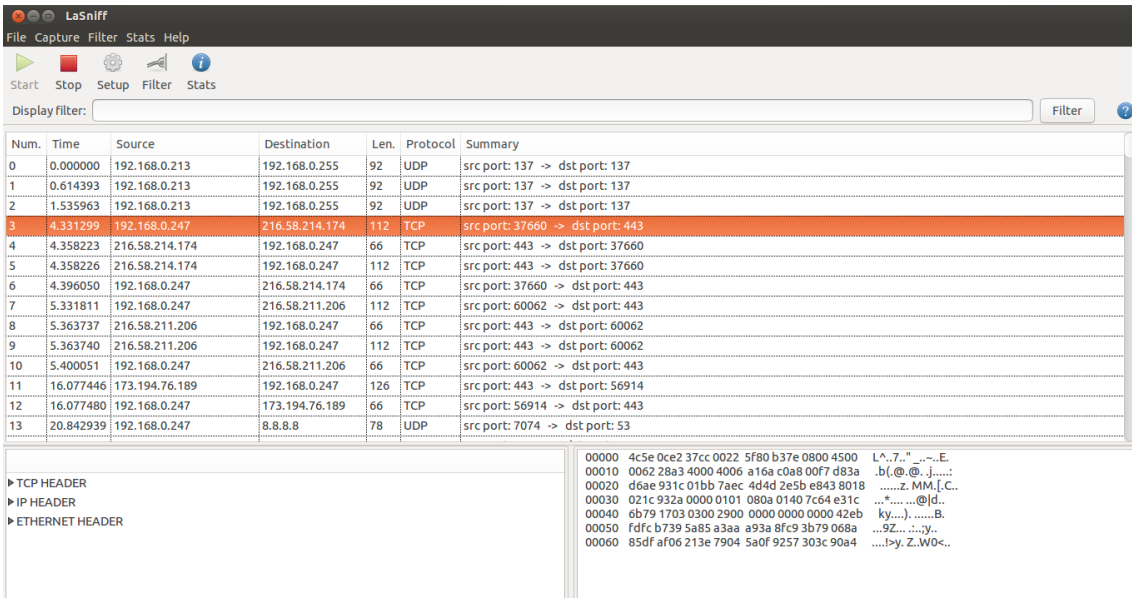


Figura 26: Interfície amb captura iniciada

Veiem que les pantalles principals, de capçaleres i de dades s'han omplert amb els continguts disseminats dels paquets, presentant per a la pantalla principal un resum del paquet amb les parts més significatives. Pel cas de l'exemple, s'ha seleccionat el paquet numero 3, que ha estat capturat en un temps de 4,331299 respecte el primer paquet capturat, amb font 192.168.0.247, destí 216.54.214.174, amb mida 112, protocol TCP i una breu descripció del contingut, en aquest cas que el port origen és 37660 mentre que el de destí és el 443 (HTTPS).

Per a la pantalla de dades es mostra la transcripció completa del paquet en format hexadecimal, a més de la transformació d'aquests valors a caràcters. Concretament, a la part esquerra de la pantalla apareix el nombre de línia en hexadecimal, al mig el valor del paquet en hexadecimal i a la dreta la conversió a caràcters.

Per a la pantalla de capçalera, s'ha optat per a un model basat en arbre inicialment ocult on es presenten, sota demanda, els continguts de la major part de les capçaleres i camps. Per exemple, pel cas del paquet 3 podem consultar els continguts de la capçalera TCP, de la IP i d'Ethernet:

Figura 27: Exemple extensió pantalla capçaleres

En qualsevol moment, l'usuari pot entrar una expressió de filtratge gràfic en el camp de text sobre la pantalla principal. Les expressions acceptades són source font, on font pot ser una adreça IPv4, IPv6 o ARP; destination destí, on destí pot ser una adreça IPv4, IPv6 o ARP; finalment l'usuari pot entrar protocol protocol, que admet TCP, UDP, ICMP o ARP:

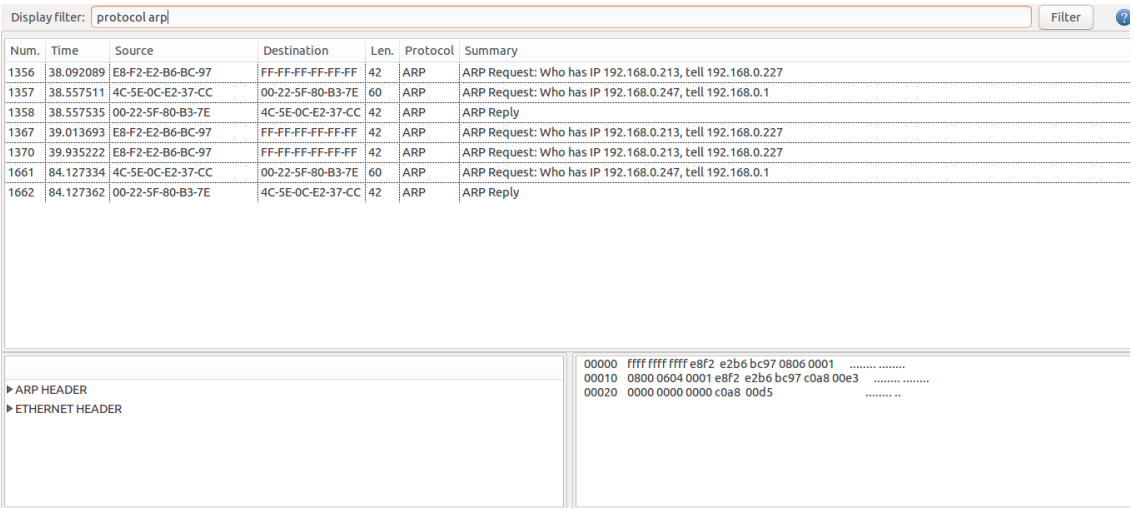


Figura 28: Filtre gràfic aplicat

Aquest tipus de filtratge és indicat com a forma d'anàlisi, mostrant únicament els paquets capturats fins al moment que compleixen amb les condicions establertes.
Per a desfer els filtres aplicats, de forma que es pugui reprendre el procés de captura si així es desitja, o simplement per a tornar a veure tots els paquets capturats, cal esborrar el text corresponent a l'expressió introduïda i prémer enter o el botó Filter.

També en qualsevol moment, l'usuari pot crear noves sessions de captura o obrir fitxers emprant l'opció abans esmentada:

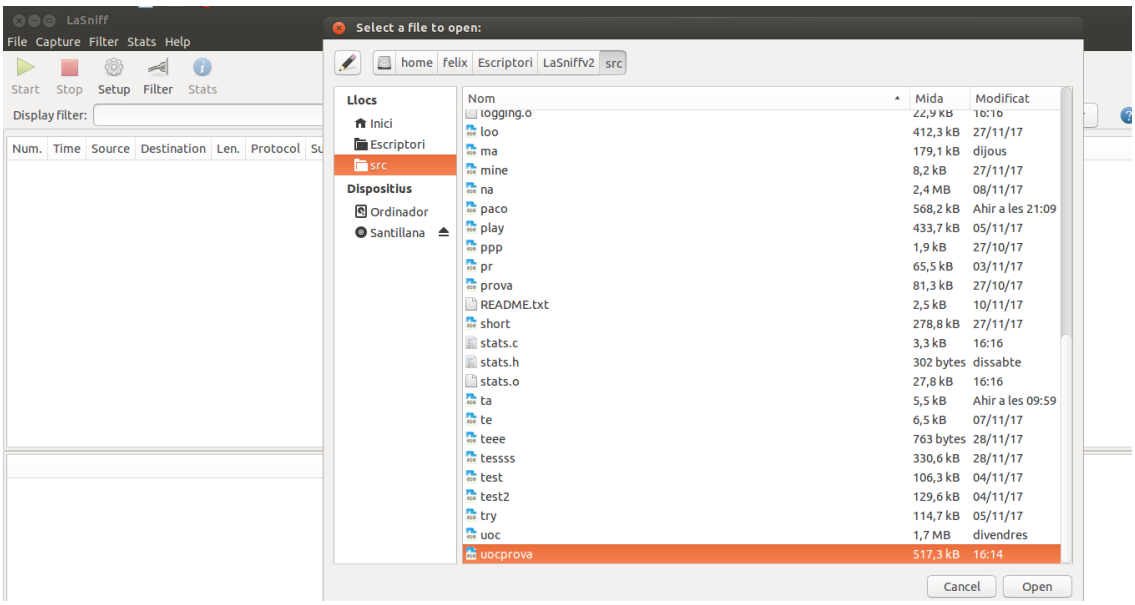


Figura 29: Pantalla obrir fitxer

Per a crear noves sessions de captura, opció que serveix per a netejar les pantalles i restaurar totes les variables en l'estat inicial, cal que l'usuari vagi al menú File → New.

Tant si es prové d'una captura prèvia oberta com si es troba en una de nova, l'usuari pot consultar estadístiques sobre la sessió de captura. Concretament pot veure el nombre total de paquets capturats així com el nombre de paquets per protocol. Es presenta, a més, un gràfic:

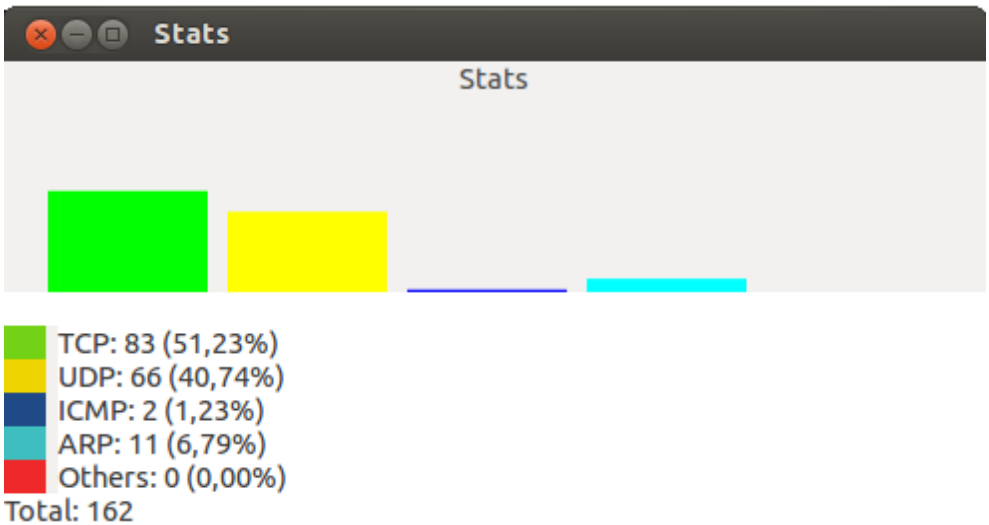


Figura 30: Pantalla estadístiques

6. Conclusions

El treball de final de grau constitueix un exercici en el qual es tracta d'aplicar els coneixements adquirits al llarg de la titulació a un projecte amb una potencial utilitat real. En aquest sentit, el present projecte ha implicat l'aplicació dels coneixements de programació adquirits, principalment en el llenguatge C, així com els coneixements tant en xarxes de computadors com en gestió de projectes, a la vegada que ha permès aprendre nous conceptes relacionats amb tots aquests mons.

En efecte, el desconeixement sobre alguns conceptes sobre els quals no s'havia treballat anteriorment, com ho són el desenvolupament d'una interfície gràfica d'usuari, les diferents formes de capturar paquets en un sniffer o la forma de crear gràfics per les estadístiques, ha implicat un procés d'aprenentatge previ i constant al llarg de tot el desenvolupament.

El producte final de projecte, el sniffer, d'altra banda, ha assolit totes les fites marcades inicialment. Pel que fa a la seva planificació, cal comentar que algunes parts, especialment els filtres gràfics, han suposat un major esforç respecte al que inicialment s'havia previst, tot i que la velocitat en la implementació d'altres components ha permès absorbir el major temps que aquesta part ha comportat.

En un futur, LaSniff podria ser estès amb altres protocols, per exemple de nivell d'aplicació, i característiques, com per a exemple, possibilitar els plugins, afegir més estadístiques, pintar línies segons protocols o filtres, facilitar el seguiment de fluxos TCP, ampliar els filtres gràfics o una major capacitat de personalització que, per motius de temps disponible i quantitat de treball, no hauria estat possible desenvolupar satisfactòriament complint els requisits temporals establerts.

7. Glossari

- **Protocol:** Conjunt de definicions, possiblement estandarditzades, que poden ser seguides per a realitzar alguna tasca.

- **Paquet:** Unitat d'informació que, en una xarxa com Internet, constitueix les dades que s'han d'enviar entre dos computadors distants més la informació que cal afegir perquè aquestes dades arribin al destí.

- **Tràfic:** Flux de paquets en circulació per a una determinada xarxa. Això inclou els que entren, els que surten i els locals.

- **GUI:** Acrònim de Graphical User Interface, o interfície gràfica d'usuari. Conjunt de recursos, com ara senyals i icones, amb què un usuari interactua per a dur a terme una determinada tasca en el si d'un sistema electrònic.

- **NIC:** Acrònim de Network Interface Controller, nom amb el qual es coneix a qualsevol targeta de xarxa que conté un computador. Serveixen per a connectar el computador a una xarxa.

- **Logging:** Procés pel qual s'enregistra quelcom, en aquest cas les captures realitzades en un fitxer.

- **PCAP:** Biblioteca de suport per a multitud d'aplicacions de xarxa per a desar i obrir fitxers de captura o aplicar filtres, entre altres coses.

8. Bibliografia

https://en.wikipedia.org/wiki/Comparison_of_packet_analyzers, visitada per primer cop el 8/10/2017

https://developer.gnome.org/gtk3/stable/, visitada per primer cop el 10/10/2017

https://en.wikipedia.org/wiki/Address_Resolution_Protocol, visitada per primer cop l'11/10/2017

https://en.wikipedia.org/wiki/User_Datagram_Protocol, visitada per primer cop l'11/10/2017

https://en.wikipedia.org/wiki/Transmission_Control_Protocol, visitada per primer cop l'11/10/2017

https://wiki.mikrotik.com/wiki/Testwiki/Introduction_to_internetworking, visitada per primer cop el 11/10/2017

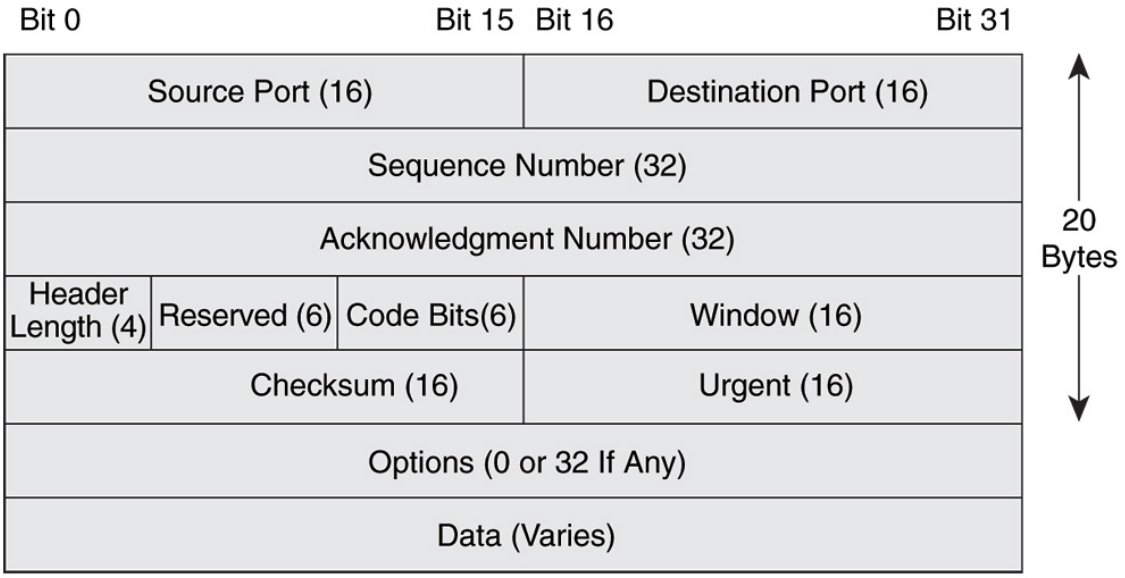
https://technet.microsoft.com/en-us/library/cc958821.aspx?f=255&MSPPError=-2147217396, visitada per primer cop el 11/10/2017

https://www.tcpdump.org/manpages/pcap.3pcap.html, visitada per primer cop el 15/10/2017

9. Annexos

Annex 1. Capçaleres protocols

TCP



Capçalera TCP

Els camps que hi podem trobar són:

- Source Port (16 bits): Port d'origen, que serveix per a diferenciar un servei d'un altre i poder mantenir l'intercanvi de dades correctament.
- Destination Port (16 bits): Port de destinació.
- Sequence Number (32 bits): Indica el byte concret del flux de dades que s'envia.
- Acknowledgment Number (32 bits): Si el flag d'ACK és a 1, aquest camp contindrà el pròxim Sequence Number que s'espera. Serveix per a confirmar la recepció de tots els bytes anteriors.
- Header Length (4 bits): Serveix per a indicar la longitud de la capçalera.
- Reserved (3 bits): Es reserva per a usos futurs.

- Flags (9 bits): Conté 9 flags, concretament:
    - FIN: Que indica si present paquet és l'últim que envia l'emissor.
    - SYN: Per a sincronitzar els nombres de seqüència en el primer paquet que s'envia.
    - RST: Per a reiniciar la connexió
    - PSH: Emprat per a indicar que s'ha de portar la informació a l'aplicació directament.
    - ACK: Que indica que el camp d'Acknowledgment Number és significatiu.
    - URG: Que indica que el camp Urgent és significatiu.
    - ECE: Que serveix per a indicar la capacitat de tractament ECN i notificar congestió.
    - CWR: Que indica que el receptor ha pres mesures de control de congestió.
    - NS o ECN-nonce: protecció en amagament.

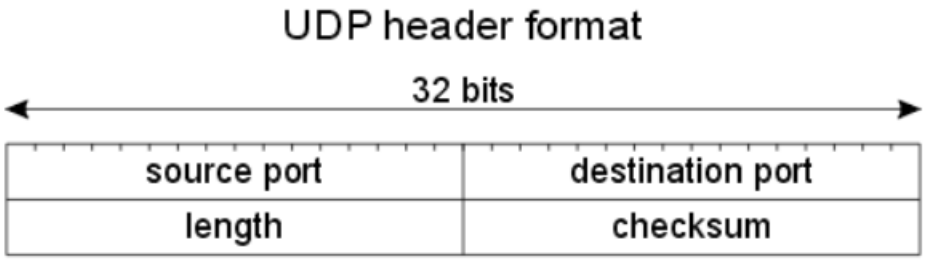
- Window (16 bits): Especifica la mida de la finestra de recepció que s'emprarà. Això indica la quantitat de dades que es podran processar a una banda i enviar a l'altre.

- Checksum (16 bits): Conté informació d'ajuda a la detecció d'errors tant en la capçalera, com en les dades.

- Urgent (16 bits): Si el flag URG és significatiu, indica un valor que cal sumar al nombre de seqüència, indicant l'últim byte urgent de dades.

- Options (variable, si hi és): Camp opcional, de mida variable i divisible per 32, que conté diverses opcions.

UDP

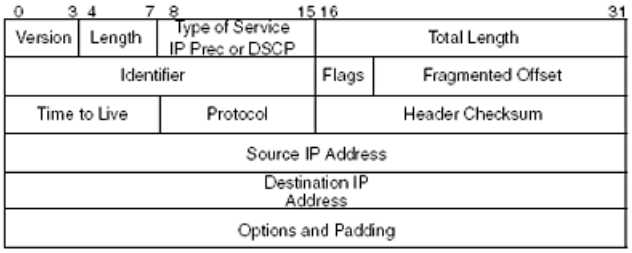


Capçalera UDP

Les seves parts són:

- Source Port (16 bits): Port d'origen, que serveix per a diferenciar un servei d'un altre i poder mantenir l'intercanvi de dades correctament.
- Destination Port (16 bits): Port de destinació.
- Length (16 bits): Conté la longitud de la capçalera més les dades.
- Checksum (16 bits): Camp que pot incloure informació relativa al control d'errors.

IPv4



Capçalera IP

Els seus camps són:

- Version (4 bits): Indica la versió del protocol a emprar. Per exemple, 4 per a IPv4.

- Length (4 bits): Conté la longitud de la capçalera IP en paraules de 32 bits. La seva longitud mínima és de 20 bytes.

- TOS (8 bits): Actualment, s'ha redefinit com a DSCP en 6 bits i els dos restants s'empren per ECN, inclou informació de rellevància per a ajudar a gestionar millor els paquets amb restriccions temporals (com en el cas del streaming) i la notificació de congestió, respectivament.

- Total Length (16 bits): Conté la longitud total del paquet, expressada en bytes, incloent-hi dades i capçalera. La mida mínima és de 20 bytes (sense dades) i la màxima 65536.

- Identification (16 bits): Camp que serveix per a identificar un conjunt de fragments de forma unívoca.

- Flags (3 bits): Tres bits, dels quals un d'ells no és usat en l'actualitat, emprats per al control o identificació de fragments. El bit MF (More Fragments) indica que el fragment actual no és l'últim, mentre que el bit DF (Don't Fragment) indica que no es permet la fragmentació.

- Fragmented Offset (13 bits): Especifica el offset d'un determinat fragment respecte al començament del datagrama original sense fragmentar.

- TTL (8 bits): Limita el cicle de vida dels datagrames a Internet, evitant que aquests hi persisteixin indefinidament. Cada cop que el datagrama travessa un router, el seu valor es redueix en una unitat.

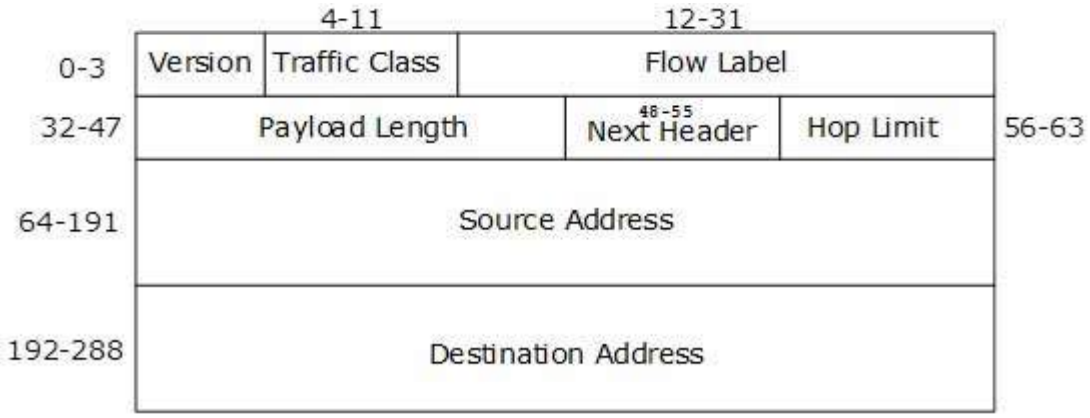
- Protocol (8 bits): Defineix el protocol encapsulat en la porció de dades. Per a exemple, el numero 6 per a TCP.

- Header Checksum (16 bits): Emprat per a comprovar errors en la capçalera.

- Source IP address (32 bits): Conté l'adreça lògica d'origen.

- Destination IP address (32 bits): Conté l'adreça lògica de destí.
- Options (variable, si hi és): Camp opcional, poc emprat en l'actualitat, que pot contenir informació de seguretat, debugging o proves.

IPv6

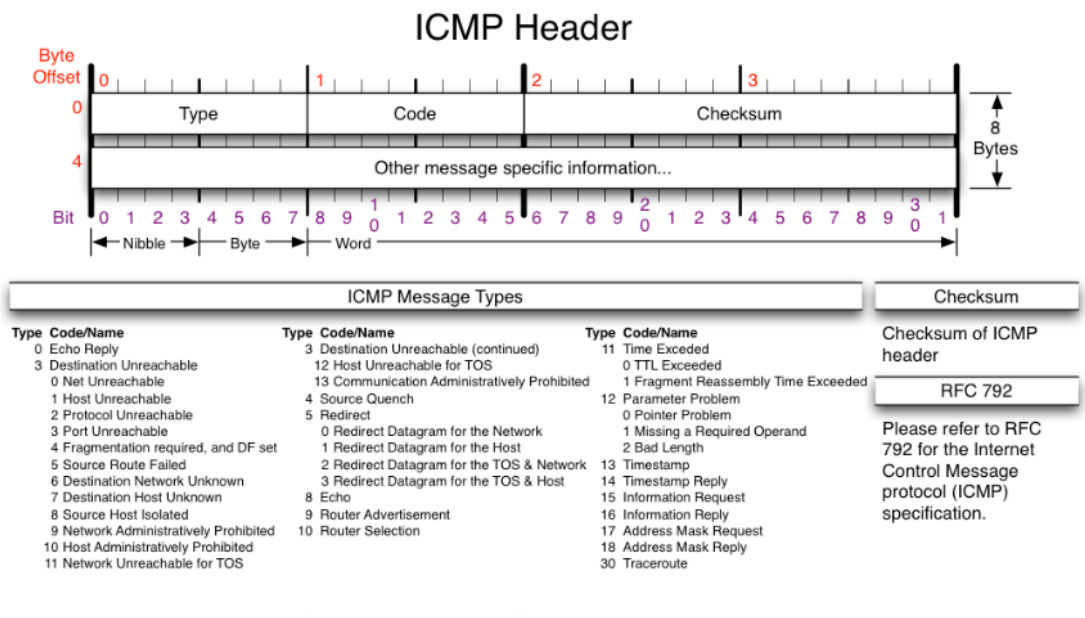


Capçalera IPv6

Els seus camps són:

- Version (4 bits): La versió del protocol emprada, en aquest cas val 6.
- Traffic Class (8 bits): El 6 primers bits serveixen per a indicar als routers quins serveis haurien d'oferir, mentre que els 2 restants s'empren com a ECN.
- Flow Label (20 bits): Serveix per a mantenir el flux seqüencial dels paquets corresponents a una determinada comunicació mitjançant el marcatge de cada paquet per a veure que aquest forma part del flux. Normalment s'empra per a aplicacions en temps real.
- Payload Length (16 bits): Indica la mida del payload del paquet. Aquesta és composta per a paquets de nivell superior i extensions de capçaleres.
- Next Header (8 bits): Si hi ha extensions de capçalera, aquest camp conté el tipus d'extensió. Si no, conté les PDU de nivell superior.
- Hop Limit (8 bits): La seva funció és la mateixa que el camp TTL d'IP.
- Source Address (128 bits): Conté l'adreça font del paquet.
- Destination Address (128 bits): Conté l'adreça de destí del paquet.

ICMP

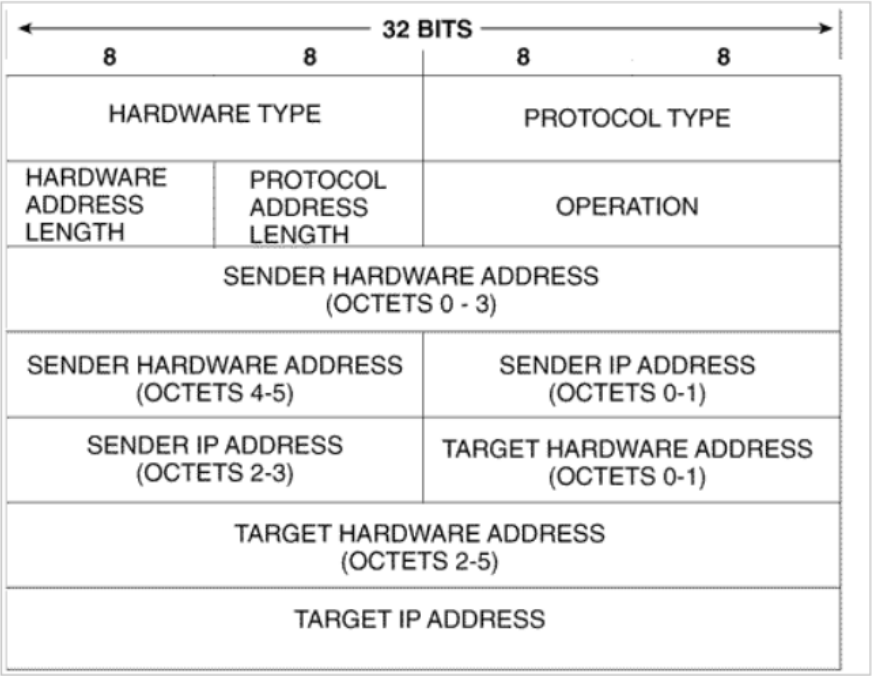


Capçalera ICMP

Veiem que té principalment quatre parts:

- Type (8 bits): Especifica el tipus ICMP.
- Code (8 bits): Especifica el subtipus ICMP.
- Checksum (16 bits): S'empra per a la comprovació d'errors.
- Other message specific information (32 bits): Que varia en funció del tipus i subtipus de missatge ICMP.

ARP

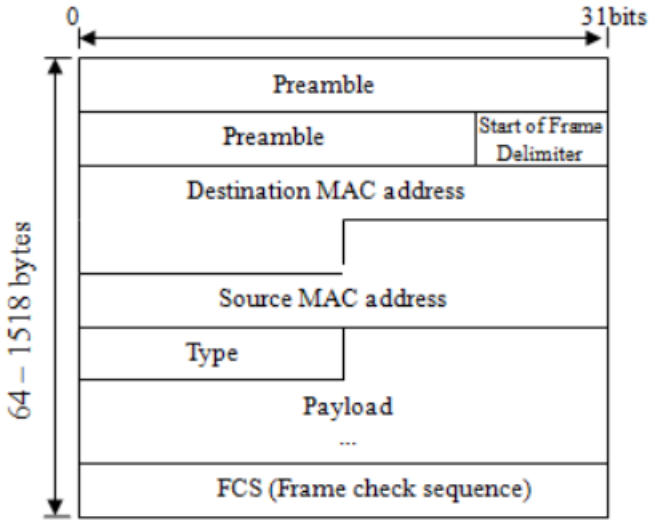


Capçalera ARP

Les seves parts són:

- Hardware type (16 bits): Un nombre que identifica el protocol de capa d'enllaç.
- Protocol Type (16 bits): Un nombre que identifica el protocol de xarxa.
- Hardware address length (8 bits): Longitud de l'adreça física.
- Protocol address length (8 bits): Longitud de l'adreça lògica.
- Operation (16 bits): El codi d'operació. En l'actualitat s'admeten 4 valors d'operació. L'1 per a petició ARP; El 2 per a resposta ARP; El 3 per a petició RARP (reverse ARP) i el 4 per a resposta RARP.
- Sender Hardware Address (48 bits): Inclou la informació sobre l'adreça física d'origen.
- Sender IP Address (32 bits): Inclou informació sobre l'adreça lògica origen.
- Target Hardware Address (48 bits): Inclou la informació sobre l'adreça física de destinació.
- Target IP Address (32 bits): Inclou informació sobre l'adreça lògica destinació.

Ethernet



Capçalera Ethernet II

Les parts són:

- Preamble (56 bits): Es tracta d'una cadena de 7 bytes que segueix el mateix patró i que serveix per a iniciar la sincronització en els dispositius implicats.
- SFD (8 bits): Indica el començament d'una nova trama.
- Destination MAC address (48 bits): Marca la direcció física del host destí.
- Source MAC address (48 bits): Marca la direcció física del host origen.
- Type (16 bits): Inclou informació per a determinar quin és el protocol contingut.
- Payload (variable): Les dades en si, que inclouen els paquets a nivell de xarxa, a més de les seves capçaleres.
- FCS (32 bits): Es tracta d'un CRC que serveix per a la comprovació d'errors.