Analitzador gràfic de xarxa en entorn GNU

Francisco Javier Rodríguez Miguel

Enginyeria Tècnica d'Informàtica de Sistemes

Consultora: Maria Isabel March Hermo

15 de gener de 2008

Als meus pares, pel seu esforç durant tota la seva vida,
a la meva germaneta, per aguantar totes les meves tonteries

...al meu amor, per tornar-me a la felicitat

Resum

Un analitzador de xarxa (sniffer) no és més que un programa que intercepta tota la informació que passa per una interfície de xarxa determinada. Una vegada capturada, podrà ser emmagatzemada per al seu anàlisi.

En aquest treball final de carrera es pretén desenvolupar aquest tipus de programari donant-li les funcionalitats necessàries per tal de realitzar una sèrie de filtratges i per realitzar un petit estudi estadístic sobre el tipus de paquets que hi circulen per una determinada interfície de xarxa. Així, la seva finalitat serà la programació d'un analitzador de xarxa senzill que implementi una sèrie de objectius mínims que puguin ser ampliats en futures revisions del mateix. A més, el present treball intentarà posar de manifest tot el que s'ha aprés, en quant a l'elaboració de programari, durant la realització de la carrera de Enginyeria Tècnica de Sistemes.

Paraules clau: Analitzador, xarxa, filtre, protocol, interfície gràfica, programació multifil.

Índex

Índex de figures

1. Introducció

1.1. Justificació: punt de partida i aportació del TFC

El tema de xarxes de computadors és un dels camps on l'evolució de la informàtica ens ha proporcionat resultats més espectaculars en els darrers anys (WAN, LAN, Internet, Wi-fi,...). A més és un dels temes que més desconeixia quan vaig començar a estudiar la present enginyeria tècnica, ara fa uns anys. Es per aquest motiu que s'ha decidit desenvolupar qualque tema dins el camp de les xarxes de computadors. A més, entre els temes proposats inicialment, he cregut interessant desenvolupar l'analitzador de xarxa, ja que crec que així es completaran els conceptes introduïts en altres assignatures de la carrera. A més, un altre tema interessant, i que cal assenyalar, és que l'entorn de programació Java en que aquest analitzador s'implementarà també era bastant desconegut al principi de la carrera. Així i com la creació d'un interfície gràfica.

El resultat final serà un analitzador de xarxa que permeti realitzar captures del tràfic d'un forma intuïtiva, sense que l'usuari final hagi de tenir amplis coneixements sobre el tema de xarxes.

1.2. Objectius

Al començament del present treball s'han marcat una serie de funcionalitats que el present treball final de carrera hauria d'implementar. Aquestes funcionalitats inicials, son uns objectius mínims que podran ser ampliats durant el desenvolupament del projecte. Concretament,

- L'analitzador ha de permetre seleccionar la interfície per la qual es vol realitzar la captura de trames.

- El programari ha de permetre seleccionar el tipus de paquets a capturar. S'hauria de poder triar entre: ARP, TCP, IGMP, ICMP i UDP.

- S'ha de poder seleccionar el port de filtratge o bé seleccionar tots els ports.

- S'ha de mostrar una petita estadística de la captura realitzada.

- La interfície d'usuari ha de ser completament gràfica i ha de complir amb els principis de disseny gràfic.

1.3. Enfocament i mètode seguit

Una vegada fixats els objectius inicials del projecte, s'ha passat a seleccionar el programari i el llenguatge de programació que es farà servir per aconseguir-los. Concretament,

- El llenguatge de programació seleccionat és el JAVA, ja que la programació serà orientada a objectes. No s'ha considerat convenient fer servir altres llenguatges de programació, com podria ser el C++, degut a que es necessita un temps per acostumar-se els seus entorns de programació, temps que degut a les limitacions actuals no es disposa.

- Es farà servir la llibreria JPCAP, que és una llibreria de captura de paquets de comunicacions per aplicacions fetes amb JAVA. Es farà servir la versió 0.4, ja que la versió actual la 0.5 encara conté errors i no és del tot fiable. Aquesta llibreria es troba disponible dins el projecte de sourceforge a la direcció http://jpcap.sourceforge.net . La distribució que es fa servir a més de contenir l'API per desenvolupar aplicacions de captura de paquets amb Java, conté eines de captura i anàlisi de trafic de xarxa en temps real.

- La interfície gràfica de l'aplicació es desenvoluparà mitjançant SWT (Standard Widget Tookit), una interfície gràfica d'usuari desenvolupada dins del projecte Eclipse, que és l'entorn de programació seleccionat. Cal senyalar que aquesta part del projecte pot ser molt interessant ja que és una de les parts que més es deixen de banda durant la carrera, i que menys es coneix de partida.

El desenvolupament del projecte s'ha separat en diferents fases. Concretament, una primera fase on es realitzarà un estudi d'investigació que permetrà la recollida d'informació relacionada amb els punts a desenvolupar. Una segona fase d'anàlisi i disseny, que permetrà descriure amb detall les funcionalitats a implementar així com el disseny del programari a desenvolupar. Finalment, una fase d'implementació i proves, que proporcionarà com a resultat un programari amb les funcionalitats marcades al principi del treball. Cal dir que el mètode seguit en el present treball per realitzar el desenvolupament del programari es basa en el mètode de W, considerant-lo com un cas particular del mètode amb espiral degut a la limitació de temps per dur a terme el projecte (un semestre).

1.4. Planificació

La primera tasca que s'ha desenvolupat en el present treball és la realització d'una planificació temporal del mateix per tal de intentar ajustar els objectius fixats inicialment al temps disponible.

La planificació inicial proposada ha estat la següent:

Recollida d'informació

Termini: fins el 14 d'octubre (dues setmanes)

Punts a desenvolupar:

- Consulta prèvia de treballs anteriors.
- Cerca d'informació relacionada amb els protocols TCP/IP
- Anàlisi de programari que implementi un analitzador de xarxa per poder extreure les possibles funcionalitats.

Anàlisi i disseny

Termini: fins el 4 de novembre (tres setmanes)

Punts a desenvolupar:

- Enumeració de les funcionalitats que ha d'implementar l'analitzador de xarxa.
- Disseny UML de les diferents classes i mètodes ha implementar

Fase d'implementació i proves

Termini: fins el 30 de desembre (vuit setmanes)

Punts a desenvolupar:

- Primera implementació del disseny previ
- Realització de les primeres proves.
- Enumeració de les millores a implementar
- Implementació de les millores
- Proves

Realització de la memòria

Termini: fins el 13 de gener (dues setmanes)

Punts a desenvolupar:

- Realització de la memòria del treball final de carrera

Realització de la presentació de la memòria

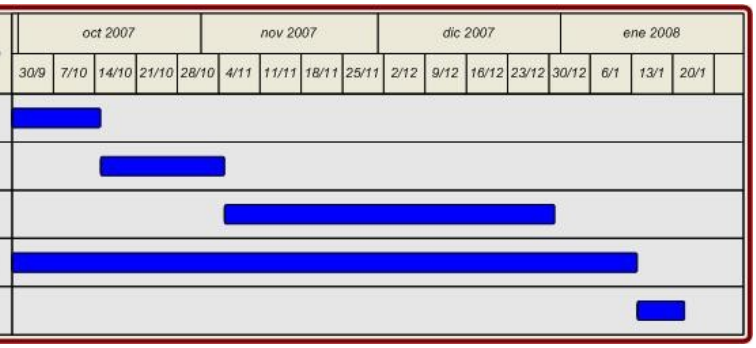
Termini: fins 21 de gener (una setmana)

Punts a desenvolupar:

- Realització d'una presentació virtual del treball desenvolupat

A continuació es mostra un diagrama de temps on es mostren els punts més importants dins la planificació anterior.

Id.	Nom de tasca	Començament	Acabament	Duració
1	Recollida d'informació	30/09/2007	14/10/2007	15d
2	Anàlisi i disseny	15/10/2007	04/11/2007	21d
3	Implementació i proves	05/11/2007	30/12/2007	56d
4	Realització de la memòria	30/09/2007	13/01/2008	106d
5	Realització de la presentació	14/01/2008	21/01/2008	8d



1.5. Modificacions a la planificació

La planificació inicial del projecte ha sofert diferents modificacions. Dels problemes que han aparegut al llarg del projecte podem destacar els següents:

- El primer problema que va aparèixer va ser durant la fase de disseny. Aquest és el segon semestre que es cursa aquest treball final de carrera i possiblement la falta de motivació deguts als escassos resultats obtinguts el semestre anterior va provocar un retràs ja en aquesta fase del projecte. A més, a l'hora de configurar el sistema operatiu amb les llibreries necessàries es va tenir que instal·lar dues distribucions diferents per fer funcionar tot correctament.

- Un altre problema que es va presentar va ser a l'hora de presentar els resultats obtinguts fent servir una representació gràfica. Inicialment, es va pensar en fer servir la llibreria jfreechart que està especialment pensada per fer aquest tipus de representació. Però, degut també a problemes de configuració amb les llibreries gràfiques natives pròpies del sistema operatiu, no es va aconseguir aquesta fita.

- Per la banda positiva, els objectius inicials que es varen marcar han estat ampliats ja que per exemple, el filtratge de paquets s'ha permès tant per ports com per adreces IP; i s'ha aconseguit una interactivitat amb la captura obtinguda no fixada inicialment.

1.6. Productes obtinguts

Una vegada finalitzat el projecte es pot afirmar que s'han obtingut els següents productes:

- Un analitzador de xarxa en entorn GNU/Linux capaç de realitzar captures de les trames que circulen per una interfície de xarxa determinada. Aquesta captura es realitza fent servir una interfície gràfica dissenyada per que l'usuari final la pugui fer servir d'una manera intuïtiva.

  A l'entrega, el programari s'ha dividit en dos directoris: un directori anomenat *codi* on es troba el codi font del programari desenvolupat; i un directori anomenat *executable*, que conté el programa compilat, les llibreries i les imatges que el programari necessita per funcionar i que es pot executar escrivint a la línia de comandes:

  ```
  sudo java -jar sniffer.jar
  ```

- Una memòria explicativa del projecte final de carrera que descriu els diferents passos seguits per aconseguir el desenvolupament del programari anterior.

- Una presentació mitjançant diapositives del treball realitzat, que resumeixen tota la feina feta al treball final de carrera.

1.7. Breu descripció dels capítols següents.

En els següents capítols es descriu els diferents passos seguits per tal d'aconseguir els objectius marcats al principi del present treball. Concretament, el capítol 2 correspon a la fase de recollida d'informació duita a terme i que ens permetrà descriure la base teòrica que forma part del treball. En el capítol 3 es descriu els diferents punts seguits per realitzar el desenvolupament del programari, sent, per tant, un capítol eminentment pràctic. A continuació, en el capítol 4 es realitza una enumeració de les conclusions obtingudes així i com de les possibles millores que es podran dur a terme en futures revisions del treball realitzat. Finalment, en els capítols 5 i 6 es realitza una enumeració dels termes més importants que es poden trobar al llarg del treball i un llistat de la bibliografia consultada, respectivament.

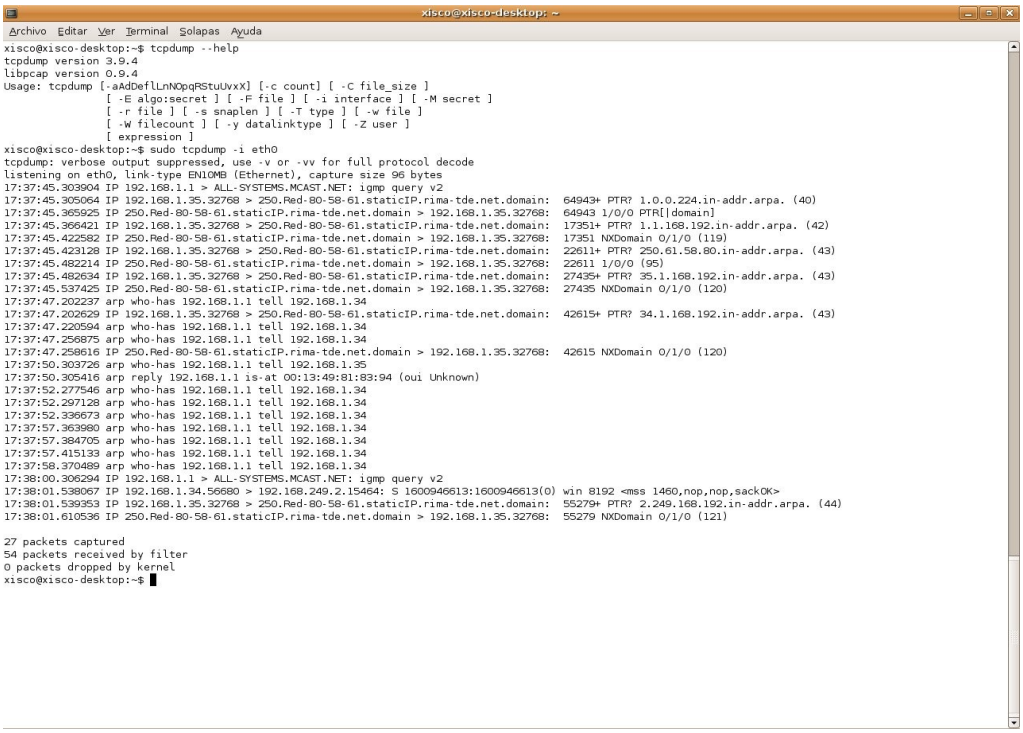
2. Recollida d'informació

2.1. Anàlisi d'analitzadors existents

Una de les tasques que primer s'han realitzat ha estat la cerca i la utilització d'alguns analitzadors de xarxa existents per tal de veure quines eren les funcionalitats que oferien i que podrien ser implementades pel present treball. A continuació es presenten alguns dels analitzadors més interessants:

Tcpdump

El Tcpdump és una analitzador de xarxa basat en línia de comandes amb llicència GPL. Aquest analitzador permet mostrar en temps real les captures obtingudes així com una descripció dels paquets. La següent figura mostra un exemple de captura amb aquest analitzador.

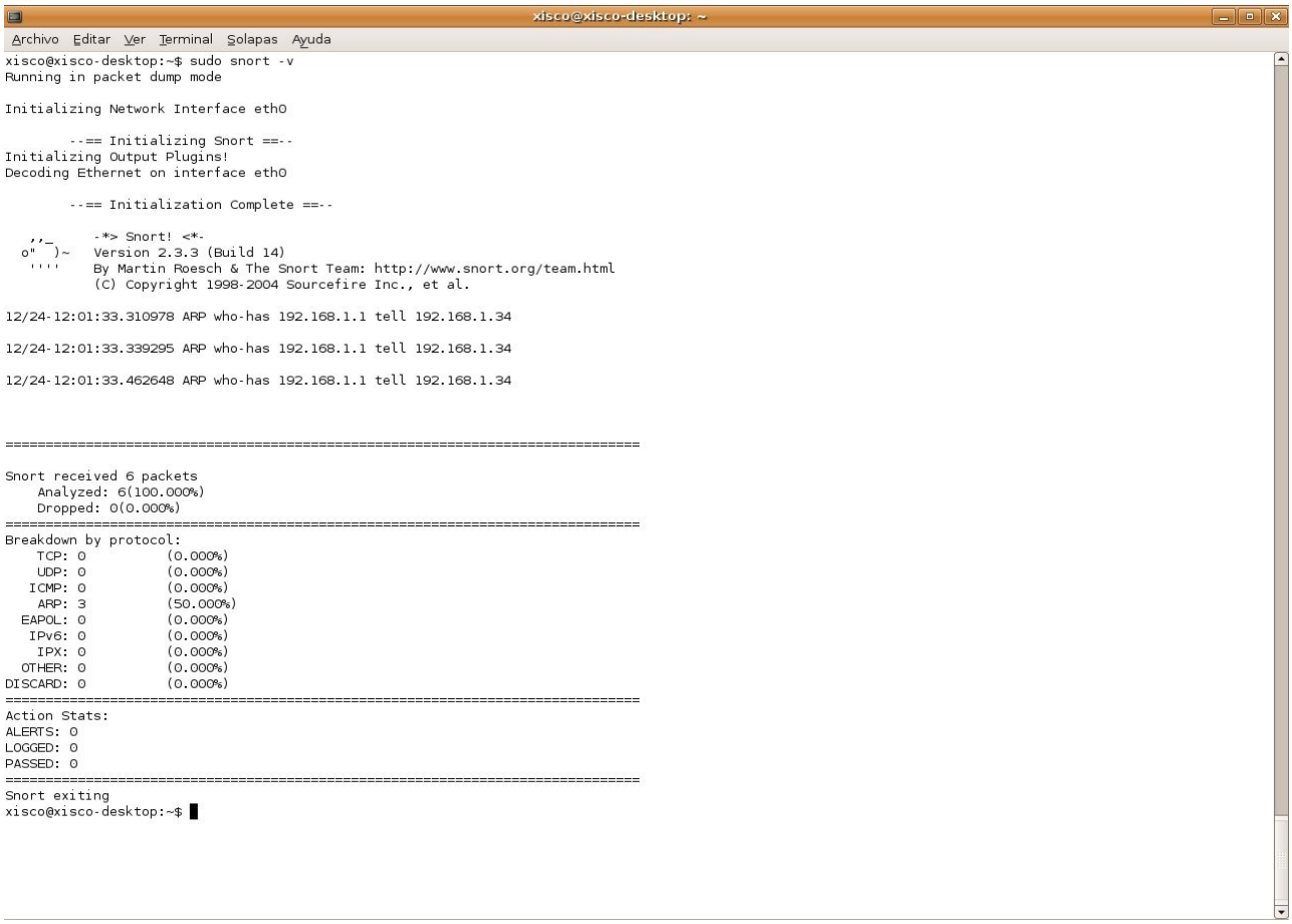


Aquest analitzador de xarxa no disposa de cap entorn gràfic ni per executar-ho ni per poder mostrar els resultats obtinguts amb ell. A més, la seva utilització no és gens fàcil, ja que la utilització de filtres també es realitza mitjançant opcions a la línia de comandes.

Snort

Snort és un altre analitzador de xarxa amb llicència GPL. La seva principal avantatge és que es tracta també d'un sistema de detecció d'intrusions (IDS) molt potent. Permet la detecció d'atacs a una màquina o sistema basant-se en una llista d'atacs coneguts.

Snort es pot fer servir només en mode d'escolta de xarxa tal com mostra la següent figura.



Com es pot veure, quan finalitza la captura mostra una petita estadística dels paquets obtinguts durant aquesta.

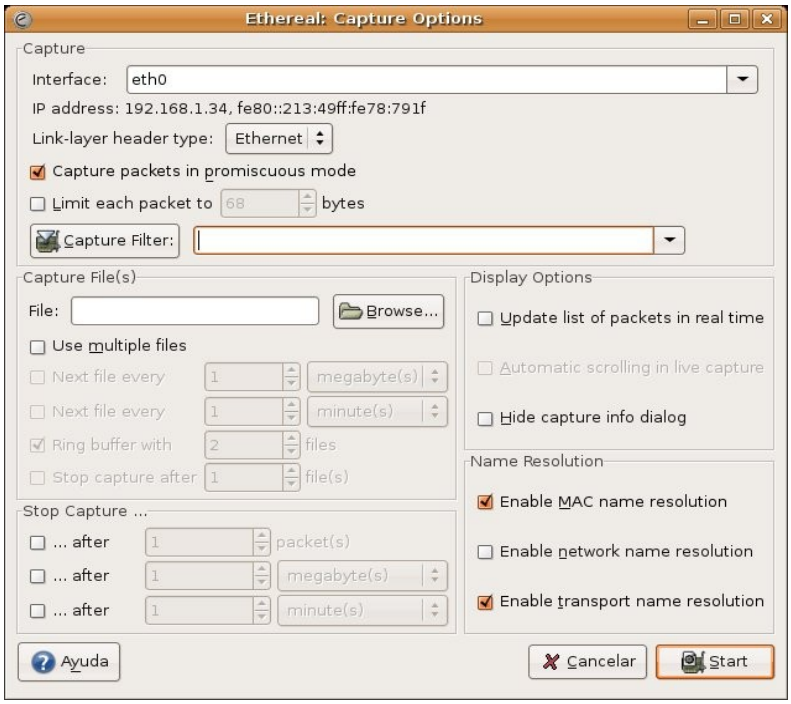
Com en el cas anterior, també es poden introduir una serie de filtres fent servir el mateix format que s'utilitza amb el Tcpdump.

Ethereal

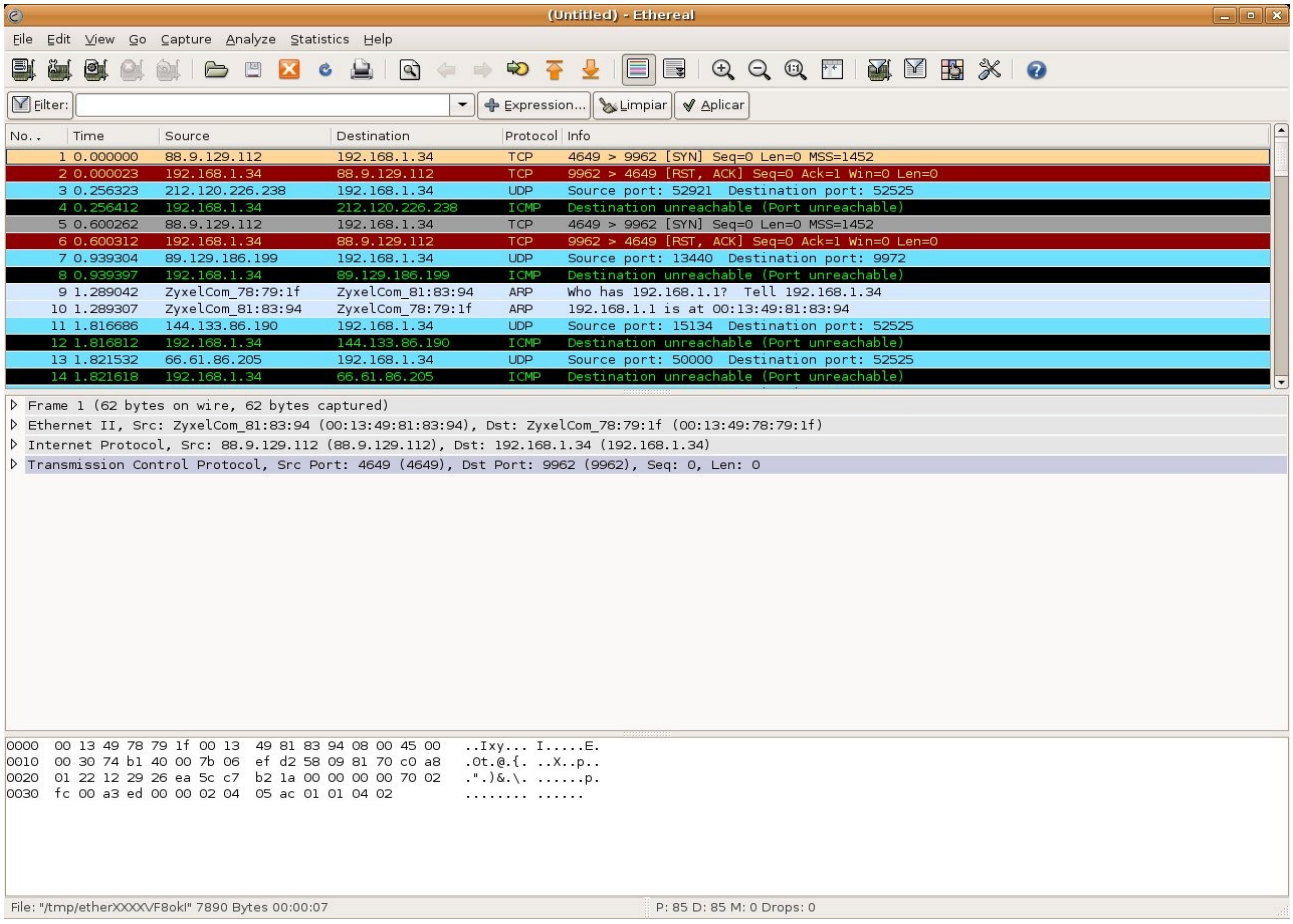
És un dels analitzadors de xarxa més utilitzat i un dels més potents, permetent capturar molts tipus diferents de protocols. Aquest analitzador fa servir una entorn gràfic, permetent, a més, la utilització de diferents filtres en aquestes captures. Actualment, es troba baix llicència GPL i ha estat batiat amb el nom de **Wireshark**.

El procés comença seleccionant la interfície que es vol escoltar, mostrant totes les possibilitats:

A més es poden introduir una serie de criteris de captura:



I una vegada acabada la captura es poden analitzar les trames obtingudes:



Aquest analitzador ofereix una gran varietat de funcionalitats, totes elles amb un entorn gràfic que simplifica la utilització del mateix. Cal senyalar però, que algunes de les seves opcions són bastant complicades de fer-les servir per a un usuari no expert.

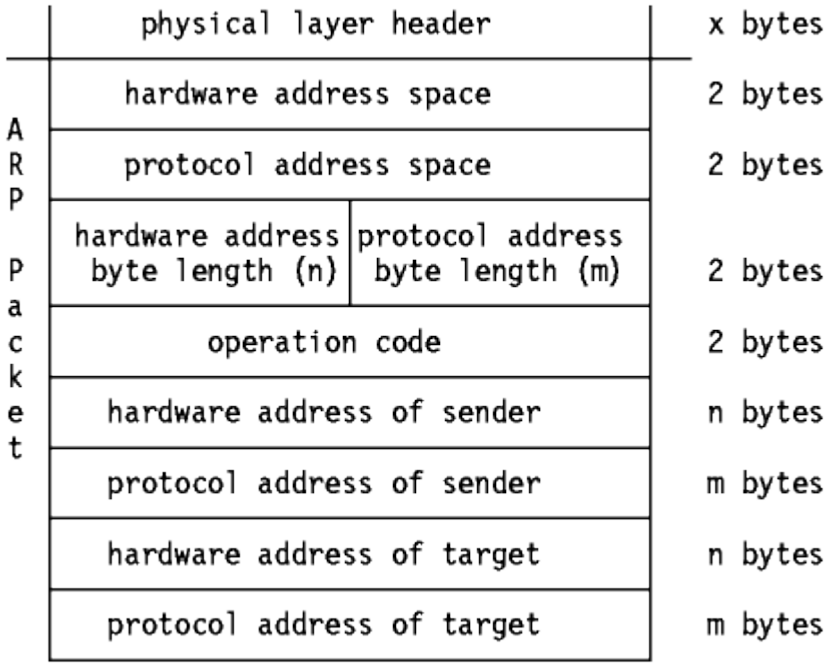
2.2. Tipus de protocols

Un protocol és una serie de codis i formats que es fan servir perquè els ordinadors puguin intercanviar informació. En funció del tipus de connexió que es pugui establir entre les màquines i els missatges que aquestes s'intercanviïn es faran servir diferents tipus de protocols. Per exemple, internet es fa servir el conjunt dos protocols independents: el protocol TCP i el protocol IP. A [IÑI03] es fa una descripció més detallada sobre les xarxes i els diferents protocols que poden fer servir.

Un dels objectius del present treball final de carrera és el d'analitzar els paquets capturats i descobrir quin tipus de protocol s'ha fet servir per codificar-los, extraient alguns camps importants d'aquests. Es per això que a continuació es passa a descriure breument els diferents tipus de protocols que la nostra aplicació hauria de reconèixer així com els camps més importants que es faran servir.

Paquets ARP

El protocol ARP (Address Resolution Protocol) és l'encarregat de realitzar la connexió entre les direccions físiques de xarxa (adreces MAC) i les direccions IP. Aquesta assignació es manté fent servir una taula de direccions anomenada *cau ARP*. Quan l'IP de la màquina desitjada no es troba dins aquesta cau, s'envia un missatge a totes les màquines de la xarxa demanant per la màquina amb direcció IP determinada, aquest missatge és un paquet ARP i es coneix amb el nom de petició ARP. Si una de les màquines té l'adreça IP desitjada contestarà amb un altre paquet ARP, anomenat *resposta ARP*, informant de la seva adreça física. A [RFC826] es pot trobar una descripció detallada sobre aquest protocol.

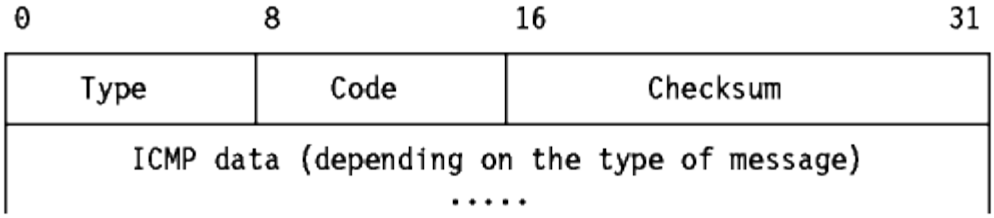


Inicialment, dels paquets ARP capturats s'analitzaran el següents camps:

- La direcció física de la màquina origen del paquet.
- La direcció física de la màquina destí del paquet.

Paquets ICMP

El protocol ICMP (Internet Control Message Protocol) és l'encarregat de gestionar les diferents incidències que puguin produir-se en una xarxa basada en l'intercanvi de trames IP. Aquest missatges es troben encapsulats dins una trama IP i, per tant, han de ser tractats com una subclasse de paquets IP. Aquests tipus de paquets es generen fent servir programes tipus *ping* o *traceroute*. A [RFC792] es pot trobar una descripció detallada sobre aquest protocol.

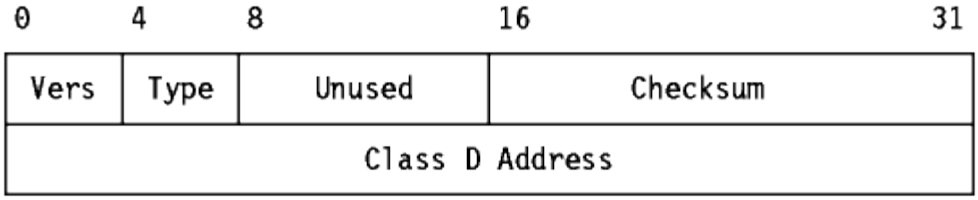


Inicialment, dels paquets ICMP capturats s'analitzaran els següents camps:

- La direcció IP de la màquina origen del paquet.
- La direcció IP de la màquina destí del paquet.

Paquets IGMP

El protocol IGMP (Internet Group Management Protocol) s'encarrega de l'administració de grups IP multicast. Aquest protocol és una subclasse dels paquets IP que permet enviar un mateix paquet a diversos servidors d'una mateixa xarxa o bé a totes les màquines d'una subxarxa. A més, es fa servir per intercanviar informació entre les màquines que formen una mateixa xarxa.

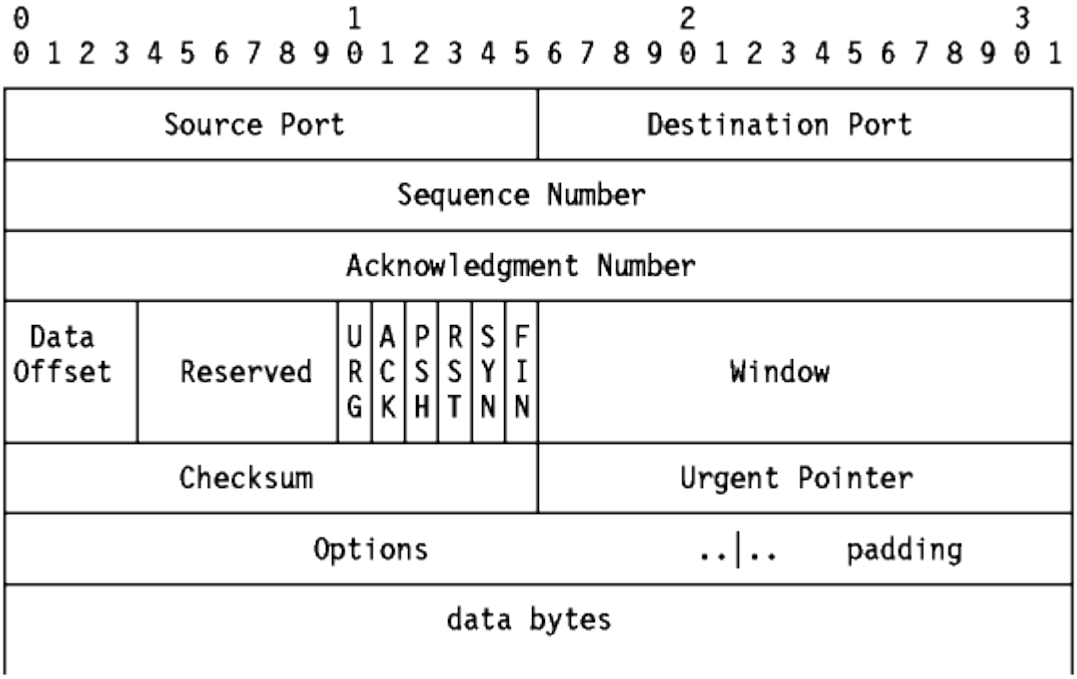


Inicialment, dels paquets IGMP capturats s'analitzaran els següents camps:

- La direcció IP de la màquina origen del paquet.
- La direcció IP de la màquina destí del paquet.

Paquets TCP

El protocol TCP (Transmission Control Protocol) és un protocol de transport que proporciona fiabilitat a l'intercanvi d'informació ja que garanteix el lliurament de tota la informació en el mateix ordre en que ha estat generada. Per poder garantir aquesta fiabilitat aquest protocol proporciona un servei orientat a la connexió amb un control de flux i d'errors. Aquest servei orientat a la connexió significa que s'ha d'establir una connexió abans de que es comenci a fer l'intercanvi d'informació. A [RFC793] es pot trobar una descripció detallada sobre aquest protocol.

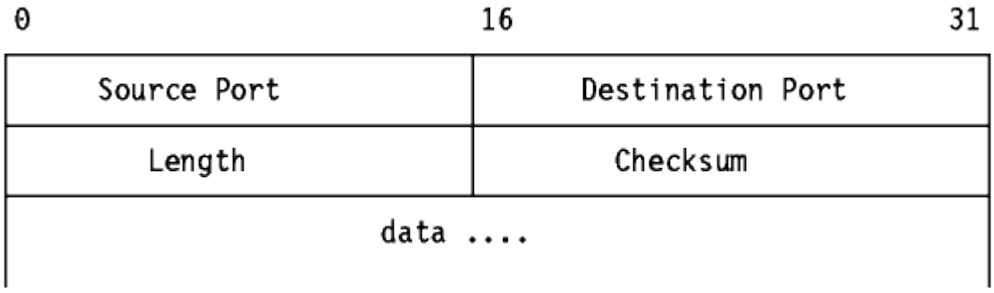


Inicialment, dels paquets TCP capturats (coneguts amb el nom de segments TCP) s'analitzaran els següents camps:

- La direcció IP de la màquina origen del paquet.
- El port de sortida del paquet capturat.
- La direcció IP de la màquina destí del paquet.
- El port de destí del paquet capturat.

Paquets UDP

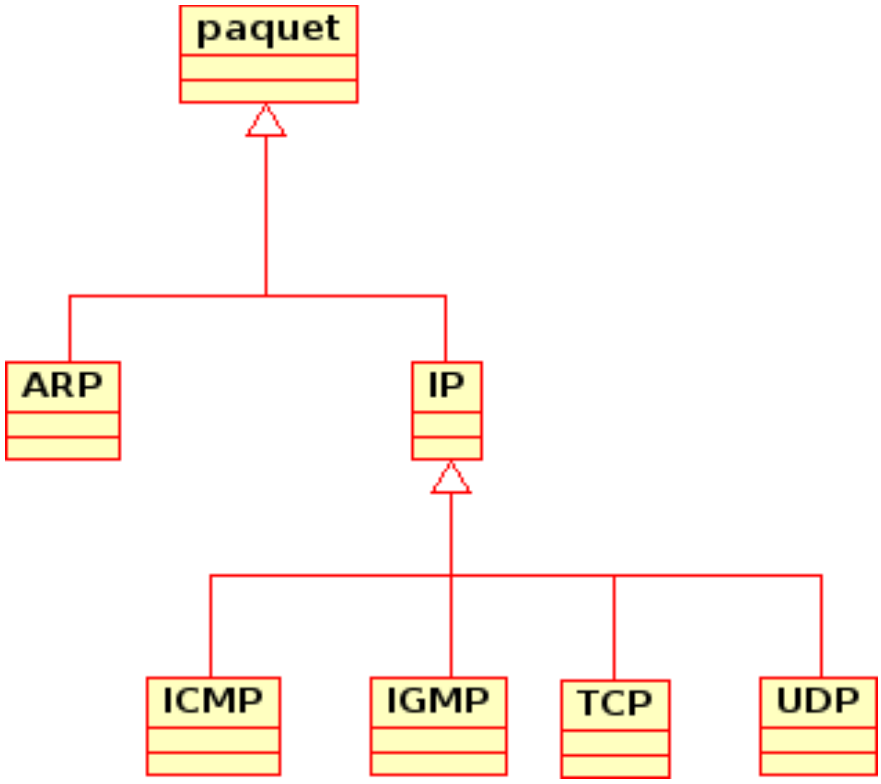
El protocol UDP (User Datagrama Protocol) és un protocol de transport que es caracteritza perquè no és orientat a la connexió, al contrari que els segments TCP. A més, no té cap mecanisme de control d'errors, en cas de detectar-se algun, el paquet UDP (conegut amb el nom de datagrama UDP) serà descartat. Aquests paquets es troben encapsulats dins un paquet IP. A [RFC768] es pot trobar una descripció detallada sobre aquest protocol.



Inicialment, dels datagrames UDP capturats s'analitzaran els següents camps:

- La direcció IP de la màquina origen del paquet.
- El port de sortida del paquet capturat.
- La direcció IP de la màquina destí del paquet.
- El port de destí del paquet capturat.

La següent figura ens mostra la relació d'herència entre els diferents tipus de paquets que es consideraran en aquest treball.

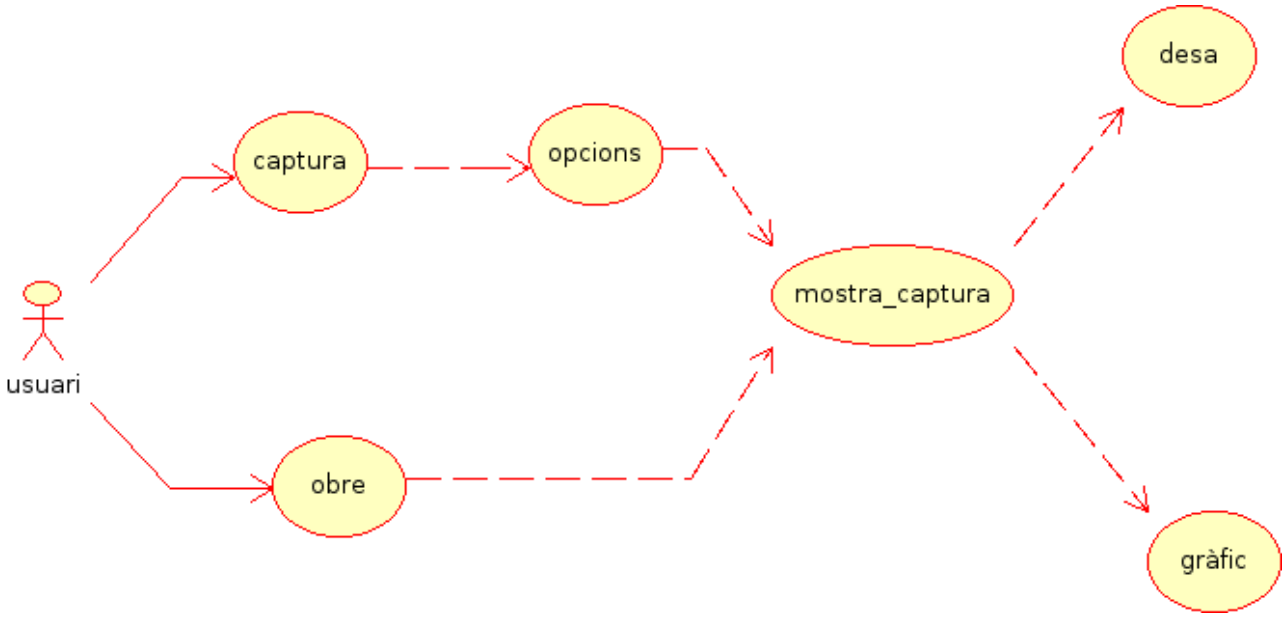


3. Desenvolupament del programari

Una vegada recollida tota la informació teòrica necessària per al desenvolupament del projecte passam a la fase de desenvolupament del programari. Tal, i com es va indicar a l'apartat 1.3 de la present memòria, el mètode seguit en el present treball per realitzar aquest desenvolupament es basa en el mètode de W, considerant-lo com un cas particular del mètode amb espiral, degut a les limitacions de temps imposades pel fet de que el present projecte s'ha de desenvolupar durant el primer semestre del curs 2007-2008.

3.1. Casos d'us

El primer que s'ha fet per tal de portar a terme el desenvolupament del programari ha estat realitzar una breu descripció dels diferents casos d'us en que es podrà trobar el nostre programari. Aquests s'han basat amb els objectius marcats a principi del projecte i descrits a l'apartat 1.2. La descripció realitzada es farà mitjançant el vocabulari visual per descriure la informació de que es disposa.



Així doncs, l'usuari haurà de poder triar entre realitzar una captura nova o bé obrir una captura anterior. En cas de voler realitzar una captura nova, haurà d'introduir una serie de opcions de captura: filtres, nombre de paquets, etc. A continuació, el sistema passarà a mostrar les dades més importants dels paquets capturats. En aquest punt el sistema ha de permetre a l'usuari desar la captura realitzada o bé mostrar una estadística dels paquets capturats mitjançant una representació gràfica.

3.2. Utilització de la llibreria jpcap

Tal i com s'ha indicat a les decisions inicials del present treball, la llibreria bàsica per poder implementar l'analitzador de xarxa és la jpcap. Aquesta llibreria ens permetrà la captura de paquets de comunicacions mitjançant el llenguatge de programació JAVA. Així doncs, passam a descriure breument el procediment per poder realitzar la captura.

Procediment de captura mitjançant jpcap

Primer de tot, hem de crear una instància de la classe `PacketCapture` que ens permetrà la captura de paquets de xarxa.

```
PacketCapture pCapture = new PacketCapture();
```

A més, es crearà un objecte Vector on s'aniran col·locant els paquets que siguin capturats per al seu tractament posterior:

```
public Vector paquets = new Vector();
```

Per tal de poder capturar, primer s'ha de indicar la interfície de xarxa per la que volem

realitzar la captura. El llistat d'interfícies disponibles s'obté amb el mètode lookupDevices() de la classe PacketCapture anterior:

```
PacketCapture.lookupDevices();
```

Una vegada seleccionada la interfície s'ha d'obrir per tal de permetre la captura dels diferents paquets fent:

```
pCapture.open(interficie,true);
```

on `interficie` és la interfície de xarxa seleccionada (ja s'explicarà el procés de selecció més endavant quan s'expliqui la interfície gràfica) i **`true`** ens indica que la captura es realitzarà en mode promiscuo. Aquest mode ens permet la captura de tot el tràfic que viatgi per la interfície seleccionada.

Ara per poder capturar els paquets només hem d'afegir un `PacketListener` a l'objecte `pCapture` fent:

```
pCapture.addPacketListener(this);
```

Així quan un paquet sigui capturat es cridarà al mètode `packetArrived(Packet data)` de la classe `PacketCapture`, al nostre cas aquest mètode únicament s'introdueix el nou paquet al vector paquets fent:

```
public void packetArrived(Packet data)
{
     paquets.add(data);
}
```

Finalment, es pot indicar el nombre de paquets a capturar amb el mètode:

```
pCapture.capture(nombrepaquets);
```

on `nombrepaquets` és el nombre de paquets, en cas de ser un nombre negatiu s'entendrà que es vol capturar paquets fins que l'usuari vulgui aturar la captura.

Una vegada capturat el nombre de paquets desitjat es tanca el procés fent una crida al mètode:

```
pCapture.close();
```

3.3. Filtratge de paquets

Una de les opcions que ha de realitzar el nostre programari és el de poder controlar el tipus de paquets que s'han de capturar. Aquest control es realitza amb la introducció del filtre de captura. Aquest filtre és una expressió basada en una serie de camps que ens permetran seleccionar els paquets desitjats. Una vegada introduïda l'expressió simplement l'afegim al procés de captura anterior fent:

```
pCapture.setFilter(filter, true);
```

Degut a la importància dels filtres dins el projecte, passam a realitzar una breu descripció sobre el format que han d'implementar els filtres en general. L'expressió que es fa servir per definir un filtre serà avaluada i en cas de ser certa farà que el paquet es capturi; en cas de que no hi hagui cap expressió a avaluar tots els paquets seran capturats. El document [ILI07] realitza un estudi detallat sobre les diferents opcions disponibles. A la següent explicació únicament ens centrarem als camps més interessants per assolir els objectius que ens hem fixat a principi del treball, si bé aquest punt podria ser millorat en següents revisions del programari..

Les tres opcions més importants per al nostre treball son les següents:

- tipus. Poden ser **host**, **net** o **port**. i ens indiquen respectivament una direcció IP d'una màquina, per exemple, *host 192.168.1.1*; una xarxa, per exemple, *net 192.168* ; o bé un port determinat, per exemple *port 22*. Per defecte, el tipus és host.

- dir. Poden ser **src** o **dst** i ens indica l'origen o el destí de les dades. Aquests modificadors poden anar combinats amb el operadors lògics **and** i **or**. Per exemple, *src 192.168.1.2 and dst 10.10.10.2* ens permetrà filtrar els paquets amb origen 192.168.1.2 i destí 10.10.10.2. Per defecte, si no hi ha operador se suposarà l'operador or.

- proto. Es tracta del protocol que volem capturar. Aquests pot ser **tcp, udp, ip, icmp, ether, arp, ...** Igual que en el cas anterior, es pot combinar amb la utilització dels operadors lògics: and i or.

A més d'aquests modificadors es poden fer servir combinacions fent servir parèntesis afegint a més els operadors lògics **and** i **or**.

Per tal d'aclarir les diferents opcions de filtratge de que disposam anam a posar alguns exemples sobre la utilització dels filtres:

- Capturar els paquets amb origen la màquina amb adreça 192.168.249.2

```
src 192.168.249.2
```

- Capturar els paquets amb origen la màquina amb adreça 192.168.249.2 i destí 192.224.66.89

```
src 192.168.249.2 and dst 192.244.66.89
```

- Capturar els paquets tcp amb destí el port 80

```
tcp dst port 80
```

- Capturar els paquets tcp o udp amb origen 192.168.249.2 i destí 192.224.66.89

```
tcp or udp src 192.168.249.2 and dst 192.224.66.89
```

3.4. Anàlisi del camps

Una vegada implementat el procés de captura i codificats el filtres disposam d'un vector on es troben tots els paquets obtinguts durant la sessió de captura. Posteriorment, s'han de analitzar els diferents camps d'aquest paquets per tal de mostrar a la interfície gràfica els seus valors. Aquest anàlisi s'ha realitzat fent servir la llibreria jpcap. Aquesta llibreria defineix una classe anomenada Packet (o bé EthernetPacket) i diferents classes que hereten d'aquesta en funció del tipus de paquet de que es disposi: ARPPacket, IPPacket, ICMPPacket, etc.; així com diferents mètodes en funció del tipus de classe. És per això que es fa servir la programació orientada a objectes per tal de fer l'anàlisi dels camps del paquets obtinguts. Concretament, del tipus de paquets considerats es faran servir els següents mètodes:

De tots el paquets :

- `public java.lang.String` **`getSourceHwAddress`**`()`
  Ens proporciona l'adreça de la màquina origen del paquet capturat.
- `public java.lang.String` **`getDestinationHwAddress`**`()`
  Ens proporciona l'adreça de la màquina destí del paquet capturat.
- `public int` **`getProtocol`**`()`
  Ens proporciona el protocol que implementa el paquet capturat.
- `public Timeval` **`getTimeval`**`()`
  Ens proporciona la data de captura del paquet.

Dels paquets ARP:

- `public java.lang.String` **`getSourceHwAddress`**`()`
  Ens proporciona l'adreça MAC de la màquina origen.
- `public java.lang.String` **`getDestinationHwAddress`**`()`
  Ens proporciona l'adreça MAC de la màquina destí.
- `public java.lang.String` **`getSourceProtoAddress`**`()`
  Ens proporciona l'adreça IP de la màquina origen.

- `public java.lang.String` **`getDestinationProtoAddress`**`()`

  Ens proporciona l'adreça destí de la màquina origen.

- `public int` **`getOperation`**`()`

  Ens proporciona el codi de l'operació. Aquest codi ens servirà per classificar el paquet com **de petició** (valor 1) o **de resposta** (valor 2).

Dels paquets IP:

- `public java.lang.String` **`getSourceAddress`**`()`

  Ens proporciona l'adreça IP de la màquina origen.

- `public java.lang.String` **`getDestinationAddress`**`()`

  Ens proporciona l'adreça IP de la màquina destí.

Dels paquets ICMP:

- `public int` **`getMessageMajorCode`**`()`

  Ens proporciona el codi dels tipus de missatge ICMP. Únicament s'han considerat els següents: **Resposta Echo** (valor 0), **desti inabastable** (valor 3), **petició echo** (valor 8) i **temps excedit** (valor 11) .

- `public int` **`getMessageMinorCode`**`()`

  Ens proporciona el codi de l'operació.

Dels paquets IGMP:

- `public int` **`getMaxResponseTime`**`()`

  Ens proporciona els temps màxim de resposta.

- `public java.lang.String` **`getGroupAddress`**`()`

  Ens proporciona l'adreça de grup multicast

Dels paquets UDP:

- `public int` **`getSourcePort`**`()`

Ens proporciona el port origen del paquet.

- `public int` **`getDestinationPort`**`()`

  Ens proporciona el port destí del paquet.

- `public int` **`getLength`**`()`

  Ens proporciona la longitud total en bytes del paquet UDP.

Dels paquets TCP:

- `public int` **`getSourcePort`**`()`

  Ens proporciona el port origen del paquet.

- `public int` **`getDestinationPort`**`()`

  Ens proporciona el port destí del paquet.

- `public long` **`getSequenceNumber`**`()`

  Ens proporciona el nombre de seqüència del paquet TCP.

- `public long` **`getAcknowledgmentNumber`**`()`

  Ens proporciona el nombre ACK del paquet TCP.

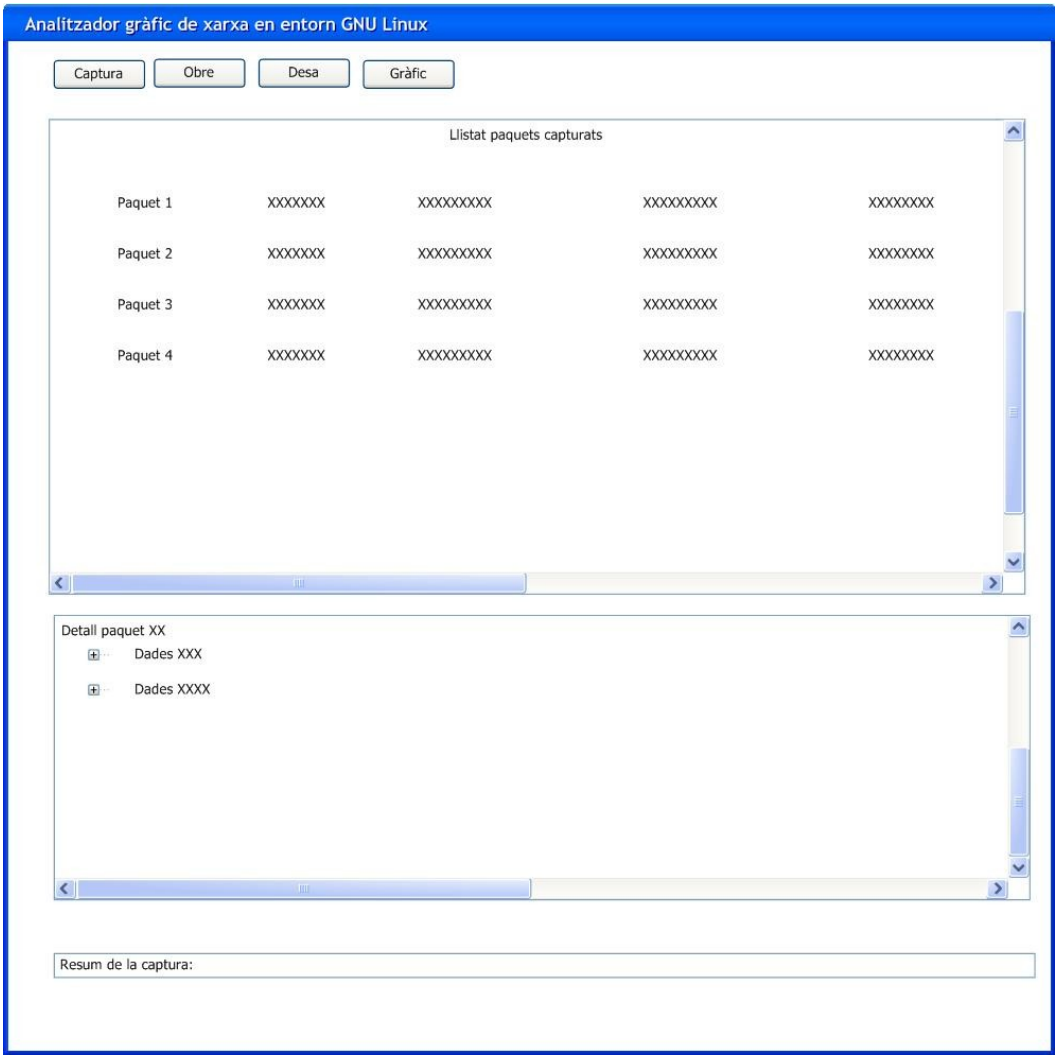
- `public int` **`getWindowSize`**`()`

  Ens proporciona el tamany de finestra que es fa servir.

3.5. Interfície gràfica d'usuari

La utilització del programari de captura per part de l'usuari es farà mitjançant una serie de pantalles gràfiques que implementaran els principis de disseny gràfic. A [GAY] es pot trobar una descripció detallada d'aquest principis.

La següent imatge mostra un prototipus de pantalla inicial dissenyada a partir de les observacions obtingudes del funcionament del programa Wireshark comentat anteriorment en aquesta memòria.



A la part superior de la pantalla es trobarien les opcions de menú disponibles. Just davall d'aquestes un llistat dels paquets obtinguts obtinguts (tal com ho fa el wireshark) i quan es pitgi sobre un d'aquests paquets es mostraria un detall dels camps analitzats fent servir una estructura d'arbre. Finalment, a la part inferior s'ha cregut convenient introduir un resum de la captura obtinguda per tal de tenir una idea global de la mateixa.

Una vegada dissenyat un prototipus de pantalla inicial, passam a implementar-la. Com ja es va indicar a les decisions inicials, la interfície gràfica de l'aplicació es desenvoluparà mitjançant SWT (Standard Widget Tookit), que ha estat creat per IBM per reemplaçar AWT i Swing. SWT és un conjunt de *widgets* per aplicacions Java que ofereix un API portable i una integració lligada amb la interfície gràfica d'usuari nativa al sistema operatiu de cada plataforma.

SWT es composa de tres elements:

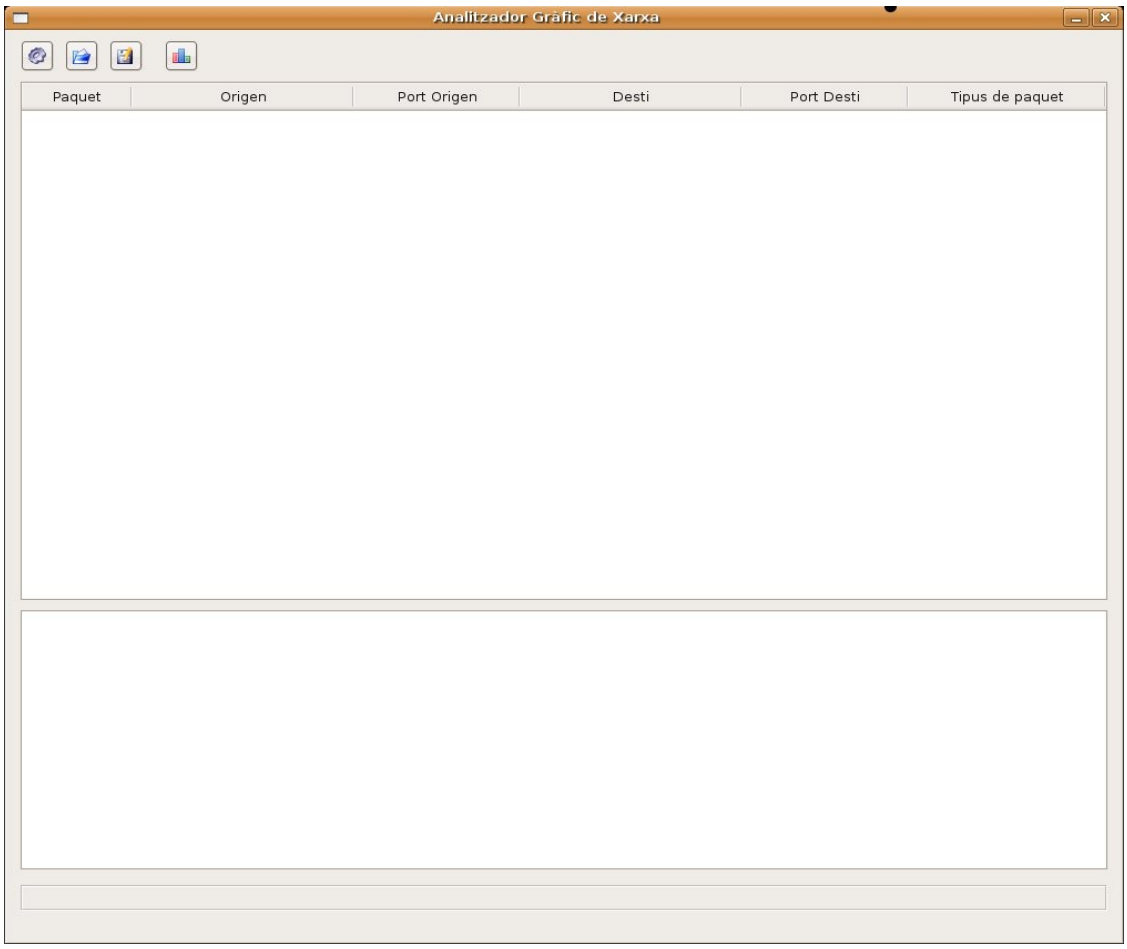
1. Al nivell inferior es troba la llibreria nativa que interacciona directament amb el sistema operatiu, anomenada JNI (Java Native Interface).

2. Per damunt de l'anterior capa es troba la classe `Display`, que és la interfície amb la que SWT es comunica amb la interfície gràfica.

3. El nivell superior esta format per la classe `Shell`, que representa tot tipus de finestres i es on es van col·locant tots els altres components.

Per a una explicació més detallada sobre la implementació de la interfície gràfica mitjançant SWT es pot consultar [NW04].

La següent figura mostra la pantalla inicial del programari fent servir SWT.



En aquesta es pot veure quatre zones ben diferenciades, concretament:

- La part superior on es troben les quatre accions que un usuari pot realitzar: realitzar una captura, obrir una captura desada a disc, desar una captura o bé fer una representació gràfica de la captura.

- A la part central superior es troba una taula on ens proporcionen una serie de dades generals sobre els paquets capturats. Concretament, ens informa sobre l'ordre dels paquets, l'adreça i el port d'origen, l'adreça i el port de destinació i sobre el tipus de paquet de que es tracta.

- A la part central inferior es troba un requadre on s'informa sobre els camps més importants d'un paquet determinat. Aquesta part s'actualitzarà quan es pitgi sobre un paquet determinat al requadre anterior. La informació que es proporciona en aquesta part del programari s'ha descompost seguint una estructura en forma d'arbre. Veurem alguns exemples més endavant.

- Finalment, a la part inferior es troba un requadre on s'informa sobre el nombre de paquets capturats i els nombre de paquets de cada tipus.

Per començar una captura nova s'ha pitjar sobre el botó  . Ens apareixerà una finestra on es troben totes les possibles opcions que podem imposar a l'hora de fer la nova captura. A la següent figura es mostra la finestra emergent.

Les opcions disponibles son les següents:

- Triar la interfície de xarxa per la que es vol fer la captura. Totes les interfícies disponibles es troben enumerades fent servir una llista desplegable, tal com es pot veure a la figura.

- Es pot triar el nombre de paquets a capturar. Aquí tenim diferents opcions: si es vol capturar un nombre determinat de paquets introduirem aquest nombre a la casella corresponent; pel contrari, si volem capturar un nombre indefinit de paquets o bé fins que l'usuari vulgui capturar, s'haurà d'introduir un nombre negatiu en aquesta casella. Més endavant, s'explicarà com fer perquè el procés de captura finalitzi quan l'usuari així ho desitgi.

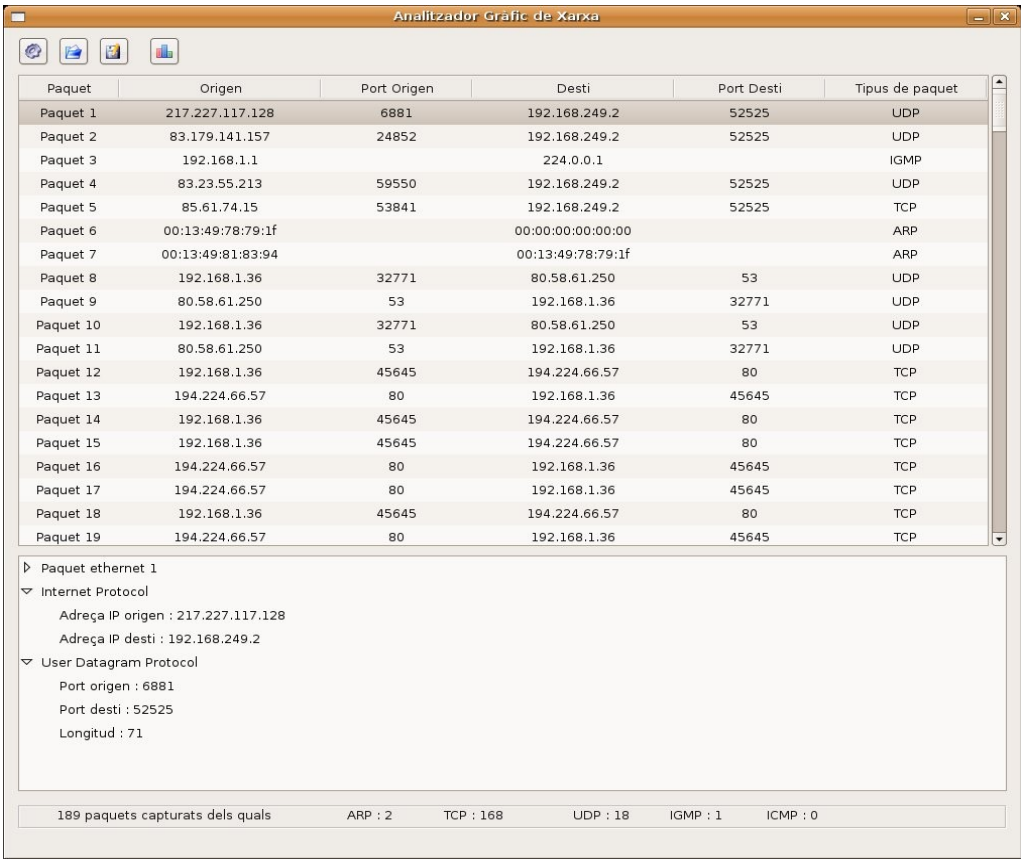
- A continuació hi ha una de les opcions del filtre implementat. Com es pot veure està agrupat a una zona que rep el nom de *paquets a capturar*. Per defecte, es capturaran tots els protocols disponibles, si l'usuari desitja realitzar un filtratge sobre el tipus de protocols capturats, només haurà d'eliminar la selecció dels protocols que no desitja capturar.

- A la part inferior, hi ha dos grups, anomenats *origen* i *destí*, on es poden introduir un altre filtratge tant sobre l'adreça IP com sobre el port, tant de la màquina origen com de la

màquina destí. Per defecte, no hi ha cap selecció, indicant que no es vol aplicar cap filtratge en aquests camps.

- Finalment, hi ha dos botons: Un botó anomenat *Captura* que començaria el procés de captura amb les opcions de filtratge indicades en aquesta pantalla; i un altre botó anomenat *Atura* que té una doble funcionalitat. Per una banda, si el procés de captura encara no ha començat i es pitja el botó *Atura*, el control del programa tornaria a la pantalla inicial i el procés de captura no es duria endavant. Per una altra banda, si el procés de captura ja ha començat i es pitja el botó *Atura* el procés de captura s'aturaria a voluntat de l'usuari amb el nombre de paquets capturats fins en aquest moment.

Una vegada finalitzat el procés de captura se'ns mostra a la pantalla principal els paquets obtinguts durant la mateixa. La següent figura mostra una captura amb 189 paquets capturats.

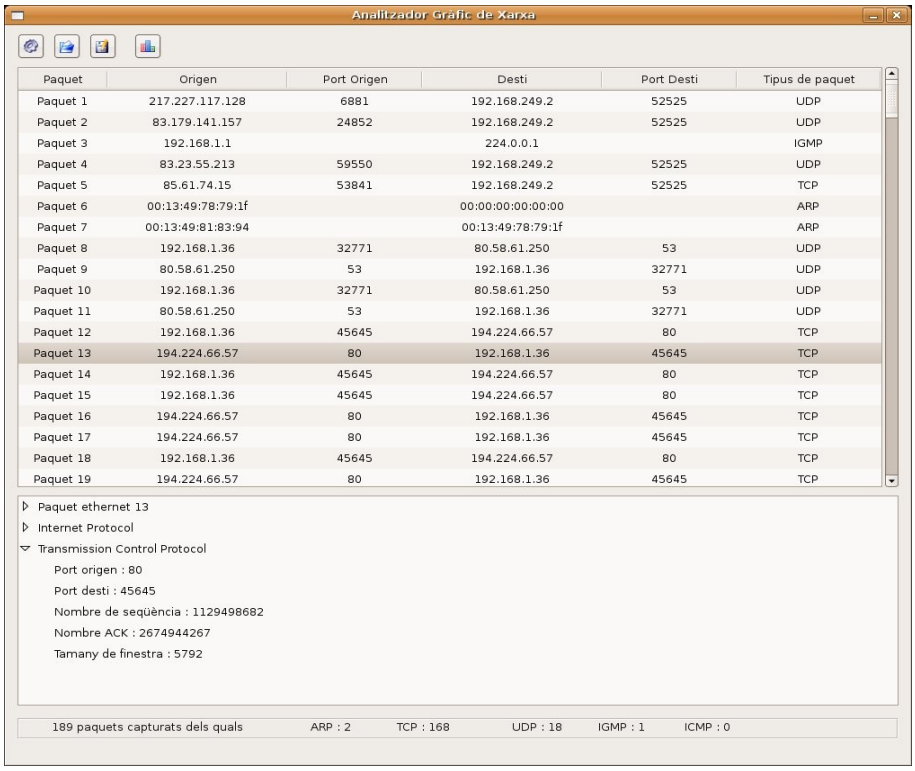


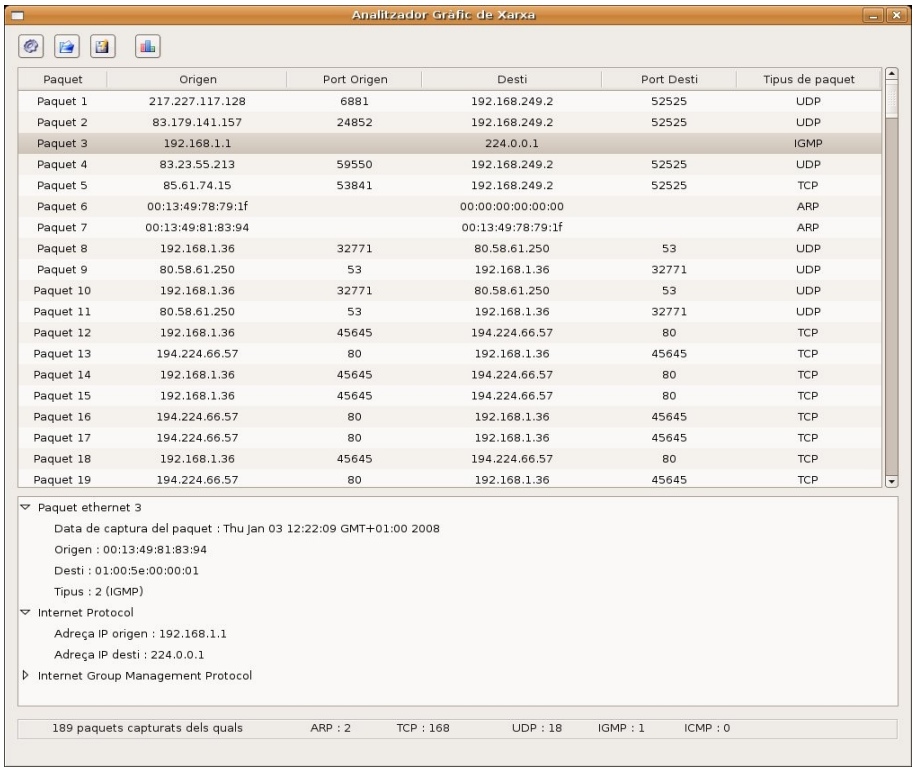
Paquet	Origen	Port Origen	Desti	Port Desti	Tipus de paquet
Paquet 1	217.227.117.128	6881	192.168.249.2	52525	UDP
Paquet 2	83.179.141.157	24852	192.168.249.2	52525	UDP
Paquet 3	192.168.1.1		224.0.0.1		IGMP
Paquet 4	83.23.55.213	59550	192.168.249.2	52525	UDP
Paquet 5	85.61.74.15	53841	192.168.249.2	52525	TCP
Paquet 6	00:13:49:78:79:1f		00:00:00:00:00:00		ARP
Paquet 7	00:13:49:81:83:94		00:13:49:78:79:1f		ARP
Paquet 8	192.168.1.36	32771	80.58.61.250	53	UDP
Paquet 9	80.58.61.250	53	192.168.1.36	32771	UDP
Paquet 10	192.168.1.36	32771	80.58.61.250	53	UDP
Paquet 11	80.58.61.250	53	192.168.1.36	32771	UDP
Paquet 12	192.168.1.36	45645	194.224.66.57	80	TCP
Paquet 13	194.224.66.57	80	192.168.1.36	45645	TCP
Paquet 14	192.168.1.36	45645	194.224.66.57	80	TCP
Paquet 15	192.168.1.36	45645	194.224.66.57	80	TCP
Paquet 16	194.224.66.57	80	192.168.1.36	45645	TCP
Paquet 17	194.224.66.57	80	192.168.1.36	45645	TCP
Paquet 18	192.168.1.36	45645	194.224.66.57	80	TCP
Paquet 19	194.224.66.57	80	192.168.1.36	45645	TCP

En aquesta figura es poden veure les quatre zones que varem descriure al començament amb informació referent a la captura realitzada. A l'exemple, s'ha pitjat sobre el paquet 1 de la taula de la

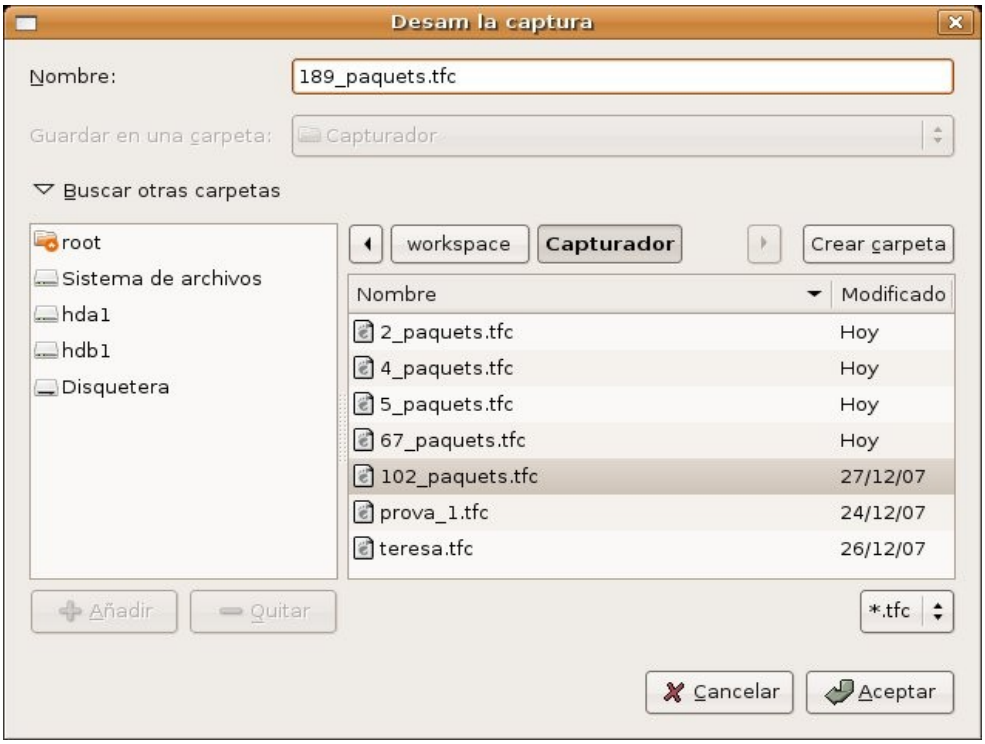
part central superior, i a la part inferior se'ns obri un arbre amb la informació disponible sobre aquest paquet determinat (a l'exemple un paquet amb protocol UDP).

Les dues següents figures mostren el resultat de pitjar sobre un paquet amb protocol TCP i IGMP, respectivament.



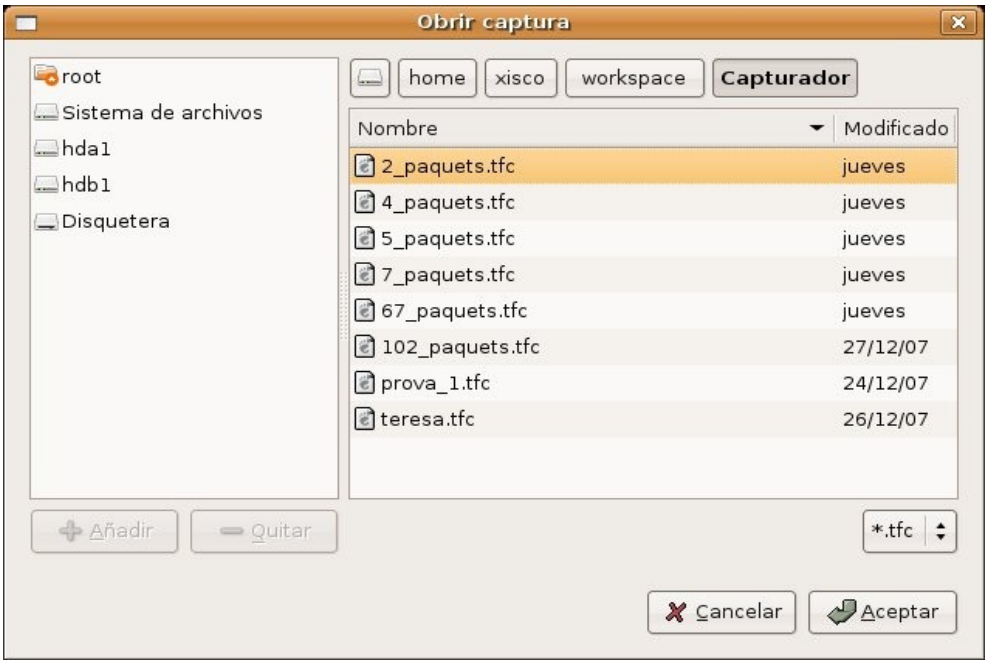


Per tal de desar aquesta captura a disc, s'ha de seleccionar l'opció de desar, pitjant sobre el botó . Ens apareixerà la següent finestra per poder desar la nostra captura:



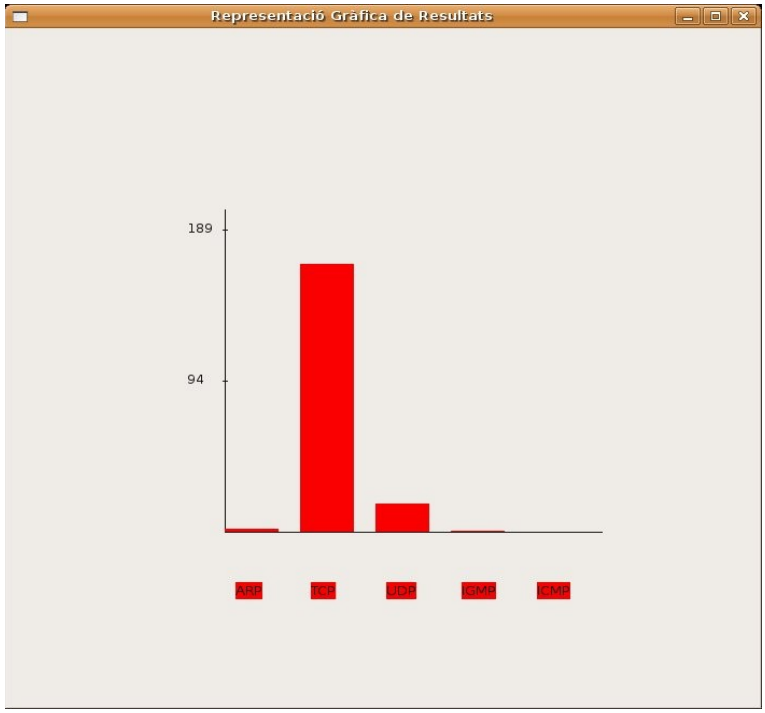
on només haurem d'anomenar la nostra captura amb extensió *.tfc*, per tal de poder diferenciar-lo d'altres fitxers.

Una vegada desada la captura en disc, la podrem obrir en qualsevol altre moment, per això s'ha de pitjar sobre el botó , se'ns obrirà una finestra com la de la figura següent, on es podrà seleccionar l'arxiu que contengui la captura a carregar. Per defecte, sortiran tots els arxius amb extensió *.tfc*



Finalment, i com un dels objectius del present treball era mostrar uns resultats de forma gràfica, s'ha dissenyat una funcionalitat que permet mostrar un gràfic que mostra una estadística sobre el tipus de paquets obtinguts. Tal com s'ha comentat en les modificacions de la planificació inicial, es va intentar fer servir la llibreria jfreechart, que conté mètodes ja implementats per fer aquest tipus de representacions; però degut a problemes de configuracions amb les llibreries pròpies del sistema operatiu aquesta utilització no va ser possible.

La decisió que es va adoptar va ser la de realitzar el dibuix fent servir les eines disponibles amb la distribució de java. Aplicant un reescalat sobre els resultats obtinguts a cada captura per tal d'ajustar-los a les dimensions de la gràfica disponible, que es va triar de 300x300 píxels.. Per tal de veure aquesta gràfica, s'ha de pitjar sobre el botó  . Se'ns obrirà una finestra com la següent, on fent servir un diagrama de barres podem veure un petit resum gràfic sobre la captura realitzada.



3.6. Programació multifil.

Una altra de les millores obtingudes respecte a la planificació inicial ha estat la utilització de fils (Threads) en el nostre analitzador. Una vegada implementat el programari es va apreciar que el programa quedava aturat fins que es capturaven el paquets que s'havia passat a la funció `pCapture.capture(nombrepaquets)` seguint un flux de control seqüencial i que l'usuari no podia aturar el procés de captura en qualsevol moment. Els fils permeten que el programa segueixi interaccionant amb l'usuari mentre es realitza la tasca de captura, fent una bifurcació en l'execució del programa i permetent que diferents processos es puguin executar en paral·lel.

Per poder crear un fil amb java simplement hem de heretar de la classe Thread i definir un mètode run() que serà la tasca que s'haurà d'executar mentre el fil estigui viu. En el nostre cas, la herència s'ha aconseguit fent que la classe que s'encarrega de la captura estengui Thread:

```
public class Capturador extends Thread implements PacketListener
```

i el mètode run( ):

```
public void run()
{
     while(this.capturant)
     {
          try{
               pCapture.capture(1);

               if(this.paquets.size()==nombre_max_paquets)
                    this.capturant=false;
          }catch (Exception e) {
               System.out.println("No es pot capturar :"
                         +e.getMessage());
          }
     }

}
```

Una vegada codificat el fil, s'ha d'instanciar i posar-lo en funcionament fent servir el mètode start().

4. Conclusions i millores...

El present treball ha consistit en la implementació d'un analitzador de xarxa senzill fent servir el que s'ha après a diferents assignatures de la carrera d'enginyeria tècnica en informàtica de sistemes. A aquest analitzador se li han donat unes funcionalitats que permeten diferents opcions tant de filtratge com de processat en les dades obtingudes. Per aconseguir els objectius marcats s'ha desenvolupat el programari seguint totes les seves fases de realització: recollida d'informació, anàlisis, disseny , implementació i proves. Els objectius aconseguits son els següents:

- S'ha aconseguit un programari capaç de realitzar la captura de les diferents trames que circulen per una interfície determinada. Aquesta interfície pot ser canviada en el moment de realitzar la captura de forma interactiva pel usuari.

- La interfície gràfica ha estat desenvolupada fent servir SWT. Cal assenyalar que aquesta part del projecte ha estat especialment satisfactòria ja que era la que menys coneixements es tenia al començament. A més, el resultat final ha estat superior al dissenyat inicialment. La utilització d'aquesta llibreria s'ha demostrat eficient i robust; inclòs per aplicacions i projectes més importants que es puguin presentar al futur.

- S'ha dotat d'una interactivitat que permet mostrar la informació de diferents camps d'una forma ordenada i que ha permès que es comprengui millor alguns conceptes de xarxes introduïts anteriorment per altres assignatures durant la carrera.

- S'ha implementat una serie de opcions de filtratge: tipus de protocol, adreces IP i ports utilitzats. Aquestes opcions permeten realitzar un control sobre els paquets capturats sense que l'usuari final hagi de tenir grans coneixements sobre xarxes.

- S'ha fet servir la programació de fils per tal de permetre l'execució de processos diferents en paral·lel i que el programari desenvolupat no s'aturi quan es realitza la captura dels paquets.

- S'ha generat una representació gràfica de la captura realitzada.

Si bé han estat molts els objectius assolits, alguns nous objectius han aparegut com a conseqüència de les proves realitzades i del perfeccionament del programari. Aquests nous objectius no s'han pogut implementar degut a la naturalesa pròpia del treball final de carrera i del temps disponible per desenvolupar-lo; si bé poden ser fites de futures revisions o millores del programari. Entre aquests cal destacar els següents:

- Un problema que no s'ha pogut solucionar ha estat que el procés de captura d'un nombre determinat de paquets tanqui automàticament la finestra de captura quan aquest hagi acabat. En el programari desenvolupat es l'usuari qui ha de tancar la finestra de captura, pitjant sobre *Atura*. Aquest problema no apareix quan el nombre de paquets es indeterminat.

- Es podria subministrar més informació sobre els paquets capturats. Tal com es va senyalar, dels paquets s'han analitzat alguns dels seus camps. Al present treball s'ha preferit doncs analitzar els camps més importants dels diferents protocols enlloc que un protocol en detall. Si bé aquest nivell de detall no implicaria massa dificultat d'implementació, si que es necessita un temps no disponible durant el projecte per dur-la a la pràctica.

- Es podria introduir una millora en el filtre utilitzat, introduint una consola d'ordres que permetin la utilització de filtres més complicats per part d'usuaris especialitzats.

- També es podria millorar la informació d'ajuda subministrada per part del programari a l'usuari final. Fent aparèixer els missatges d'error en forma de finestres gràfiques i no en forma de text com ha estat el cas.

Com a conclusió final indicar que, si bé existeixen millores possibles al producte desenvolupat, estic satisfet amb la feina feta i crec que he aconseguit uns bons resultats tant amb el programari obtingut com amb la memòria descriptiva del mateix. A més, m'he pogut donar una idea de la feina real que implica la realització d'un projecte des del seu començament fins a l'entrega del producte final.

5. Glossari

ARP (*Address Resolution Protocol*), Protocol per al mapatge automàtic d'adreces MAC amb adreces IP.

Adreça MAC Identificador hexadecimal de 48 bits que correspon de forma única a una targeta de xarxa.

Adreça IP Nombre que identifica de forma lògica una interfície de xarxa. En la seva versió 4, aquest nombre es representa mitjançant un nombre binari de 32 bits.

ICMP (*Internet Control Message Protocol*), Protocol de nivell de xarxa emprat només per a tasques de gestió del nivell de xarxa.

IGMP (*Internet Group Management Protocol*), Protocol que es utilitzat exclusivament pels membres d'una xarxa multicast per tal de propagar informació de redireccionament.

IP (*Internet Protocol*) Protocol de nivell de xarxa utilitzat a Internet.

Java Llenguatge de programació orientat a objectes desenvolupat per Sun Microsystems i que permeten crear programes multiplataforma.

jpcap Llibreria implementada en Java que proporciona un interfície de programació de comunicacions.

MAC (*Medium Access Control*) Protocol de control d'accés al medi utilitzat per les estacions d'una mateixa xarxa d'àrea local.

Protocol Conjunt de codis i formats que es fan servir perquè els ordinadors puguin intercanviar informació.

RFC (*Request For Comment*), Cadascun dels documents que expliquen amb tot detall els diferents protocols de la xarxa Internet.

SWT (*Standard Widget Tookit*), Conjunt de widgets per aplicacions Java que ofereix una integració lligada amb la interfície d'usuari nativa al sistema operatiu de cada plataforma desenvolupat per IBM dins el seu projecte Eclipse.

TCP (*Transmission Control Protocol*), Protocol a nivell de transport que permet garantir que el intercanvi d'informació s'ha realitzat correctament.

Thread Cadascun dels processos que s'executen en paral·lel a una màquina i que permeten la utilització del processador amb un temps compartit.

UDP (*User Datagram Protocol*), Protocol a nivell de transport utilitzat quan no es necessari la confirmació d'un intercanvi d'informació.

6. Bibliografia

[GAY] GAYET J.M. *Interacció humana amb els ordinadors*. Fundació per a la Universitat Oberta de Catalunya.

[ILI07] ILIOFOTOUS, MARIOS. *Tcpdump filters* [en línia]. Disponible en: <http://www.cs.ucr.edu/~marios> [Consulta: 27 de desembre de 2007].

[IÑI03] IÑIGO, JORDI et. al. *Xarxes. Aplicacions i protocols d'internet.* Fundació per a la Universitat Oberta de Catalunya, 2003. ISBN: 84-8429-582-2003

[NW04] NORTHOVER, STEVE; WILSON, MIKE. *SWT:The Standard Widget Toolkit,* Addison-Wesley Professional, 2004. ISBN: 978-0321256638

[RFC768] *User datagram Protocol.* IETF, 1980. Disponible en: <http://www.ietf.org/rfc/rfc768.txt> [Consulta: 15 de desembre de 2007]

[RFC792] *Internet Control Message Protocol.* IETF, 1981. Disponible en: <http://tools.ietf.org/html/rfc792> [Consulta: 15 de desembre de 2007]

[RFC793] *Transmission Control Protocol.* IETF, 1981. Disponible en: <http://tools.ietf.org/html/rfc793> [Consulta: 15 de desembre de 2007]

[RFC826] *An Ethernet Address Resolution Protocol.* IETF, 1982. Disponible en: <http://tools.ietf.org/html/rfc826> [Consulta: 15 de desembre de 2007]