Captura i filtrat de paquets a xarxes Ethernet.

Jose Luis Perez de Viñaspre Haro
ETIS

Consultor : Maria Isabel March Hermo

1er. Semestre Curs 2007 - 2008

Dedicatòria:

Sempre que llegia les dedicatòries a llibres de text o altres, em semblava, quelcom “cursi”, o amb poc sentit. Però després d'uns quants quadrimestres a la UOC, i ara, al final d'aquest treball de fi d'estudis, compaginant la carrera, amb la feina i la vida familiar, em sembla just reconèixer la paciència i comprensió mostrada pels que he tingut al voltant.

La meva filla gran, també amb la seva carrera recent encetada dient: Papa, que jo també necessito l'ordinador !!!, o el fill mitjà : Papa, que m'has de portar a entrenar o al partit !!! o la petita, Papa, quan acabaràs ? Han estat constants durant aquest temps, però que m'han ensenyat a aprofitar i organitzar el temps per fer-ho tot. Però darrera de tot sempre estava la meva dona, que malgrat això de la informàtica li sembli cosa de màgia, s'ha mostrat en tot moment pacient, encoratjadora i suavitzant tots els problemes i tensions soferts durant aquests anys de carrera.

Evidentment, també voldria recordar a tots els consultors i companys d'aula, que amb la seva participació han aconseguit, a més de l'assimilació dels coneixements de cada assignatura, una visió i enfocament dels problemes i de la informàtica en general que m'ha sigut molt profitós a la vida laboral.

RESUM

Aquest treball es centra en les tècniques de captura i filtratge de paquets a xarxes Ethernet. Com objectiu principal ens hem fixat l'estudi de les estructures implementades pel sistema operatiu i la interfície oferta als programadors per tal de poder utilitzar-la. Com a objectiu secundari pretenem, posar a la practica els coneixements obtinguts i desenvolupar un producte sòlid i robust en la seva funcionalitat bàsica, filtrar i capturar paquets, però que sigui extensible a altres aplicacions orientades a xarxa. Anàlisis de rendiment de la xarxa, per exemple.

INDEX

INDEX DE ILUSTRACIONS

1.- Introducció

1.1 Justificació

Durant la meva carrera professional he fet tasques de desenvolupament i administració de xarxes i sistemes. La proposta d'aquest projecte em permet unir totes dues vessants. Per un costat aprofundir en el coneixement de la pila TCP/IP i de la implementació que fa el sistema operatiu i per un altre el desenvolupament d'una eina útil i robusta que implementi els coneixements adquirits.

Tenir que fer aquest treball sol, ha permès desenvolupar una serie de capacitats, a part les estrictament tècniques o de coneixement. S'ha tingut que prendre decisions de disseny, decidir amb exactitud l'àmbit del treball, decidir que incloure o que no, resoldre problemes i en general tot allò que ha fet possible transformar una idea i una il·lusió, potser quelcom ambigües, en un producte funcional i en una memòria estructurada i coherent.

1.2 Objectius

L'objectiu principal d'aquest treball es el desenvolupament d'un software amb capacitat de escoltar el medi de transmissió, capturar paquets analitzar-los i també ens proposem donar-li capacitat de filtrar els paquets que arriben a modes de firewall. Malgrat existeixen llibreries per incorporar en aplicacions pròpies, s'ha optat pel desenvolupament total de les funcionalitats de captura i filtrat. Som conscients que això, pot restar potencia al producte final, però es prefereix d'aquesta manera, per tal d'assimilar millor les tècniques estudiades.

Com a objectius generals ens fixem els següents :

- Captura de paquets circulants per la xarxa.
- Anàlisi dels paquets capturats.
- Poder acceptar o rebutjar paquets que ens arriben.

1.3 Enfocament i mètode

Tal com hem comentat hem prioritzat el desenvolupament total de la aplicació, sense fer ús de llibreries existents, només partint de les capacitats que ens ofereix el sistema. Per aquest motiu hem fet una fase inicial d'investigació i estudi de les estructures i crides ofertes pel sistema operatiu i les restriccions que imposen, tant en el desenvolupament, com en l'execució. Aquest estudi ens ha servit, per decidir les tècniques que implementarem al producte.

També s'han provat o s'han vist les característiques d'alguns productes que ofereixen funcionalitats similars, com poden ser Ethereal, TCPDump, winpcap, zone alarm, Win XP firewall. Això ens ha permet obtenir idees per el nostre producte.

A partir d'aquí, s'ha optat, per dividir el producte en 2 funcionalitats ben diferenciades. Per un costat la captura i anàlisis de paquets i per l'altre la funcionalitat de filtrat. Aquesta divisió, ha permès tractar totes dues funcionalitats de manera independent, realitzant per a cadascuna d'elles les fases de anàlisis, disseny, implementació i proves.

Encara que es veurà amb més profunditat al capítol dedicat al producte, com a resum, podem dir que, s'ha fet un analista “top-down”. Partint d'una definició general i refinant aquest anàlisi i disseny fins a solucionar el problema. Aquestes fases m'han ajudat també en la tasca, de decidir com es farà la implementació, sobre tot, a la tria dels llenguatges a utilitzar.

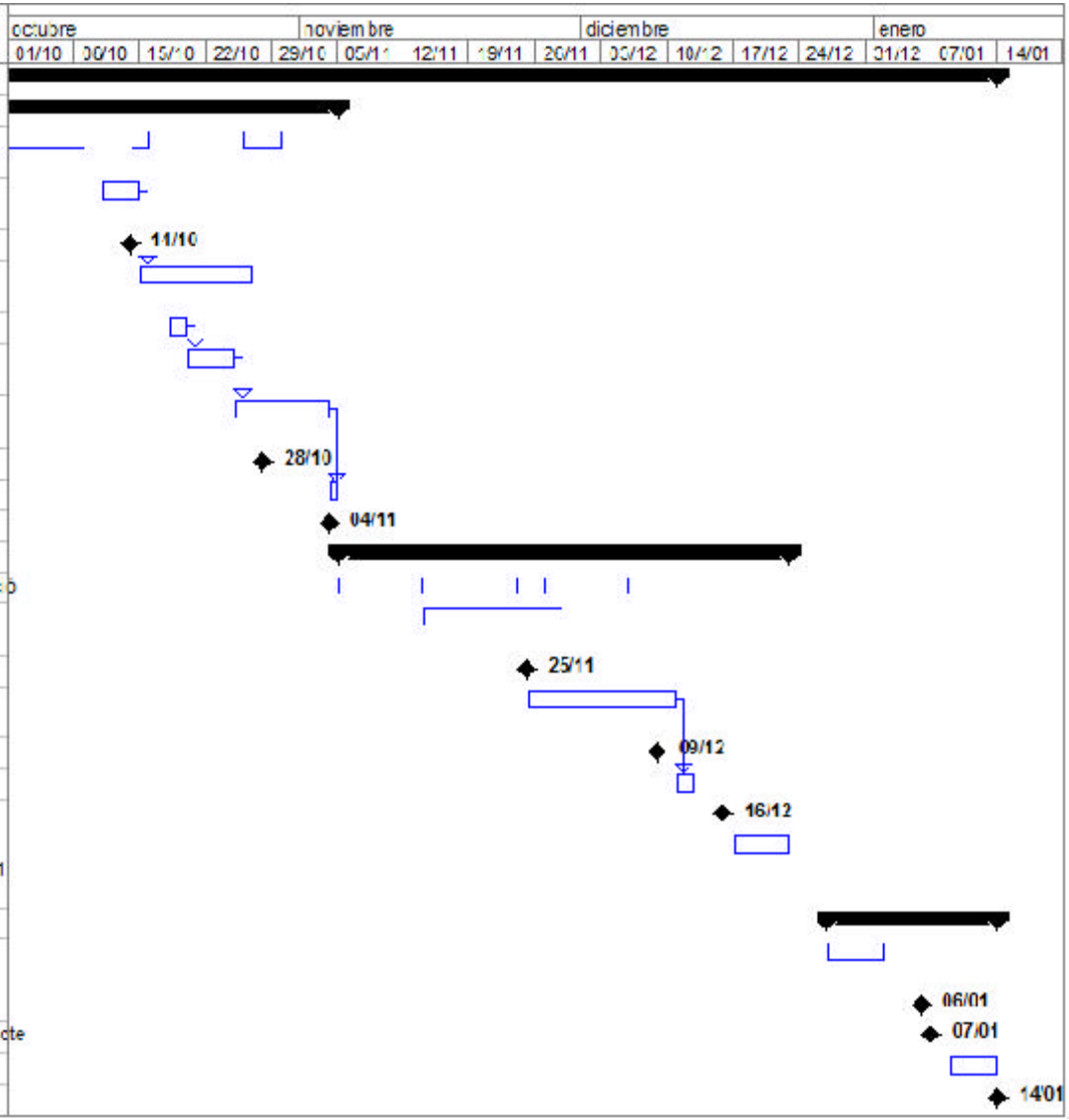
Per la fase de proves, es disposa d'una petita xarxa, composada per el punt de treball utilitzat pels estudis a la UOC i us familiar i personal, un router ADSL amb 4 ports Ethernet i un ordinador portàtil de la feina. Aquesta instal·lació m'ha permès dissenyar els jocs de proves per provar el producte.

1.4 Planificació

El semestre passat vaig intentar realitzar el mateix treball, malauradament, una mala planificació, en la que no es van tindre en compte factors externs (Nous projectes a la feina) em van impedir realitzar aquest TFC. Aquest curs hem partit de part de la feina feta el curs passat. Concretament l'estudi de la captura de paquets estava fet i implementat. Aquest curs s'ha refet totalment la presentació dels paquets capturats, i s'ha afegit suport al protocol igmp per captures multicast. De la part del filtre de paquets l'any passat només es va localitzar informació i es van provar els paquets que hem comentat a apartats anterior.

Dintre del pla de treball presentat a la PAC 1, s'ha realitzat una planificació temporal, aquest cop més acurada, del seu desenvolupament. La planificació es la que es mostra a la figura de la pàgina següent:

Id	Nombre de tarea
1	Captura i Filtrado de paquetes
2	PAC2
3	Busqueda i Analisi Informació
4	Redacció Memoria Capitols Introducció
5	Informe Prog I
6	Redacció Memoria capitols captura de paquets
7	Analisis i Disseny
8	Desenvolupament Interfície Gráfica i Objectes generals
9	Desenvolupament Captura paquets
10	Informe Prog II
11	Revisions i correccions
12	Entrega PAC 2
13	PAC 3
14	Busqueda i Analisi Informació
15	Redacció Memoria capitols Captura
16	Informe Prog III
17	Desenvolupament Filtrat paquets
18	Informe Prog IV
19	Revisions i Correccions
20	Entrega PAC 3
21	Redacció Memoria Conclusions, Manual glosari ...
22	Fase final
23	Correccions generals, maquetació, Tests i proves
24	Informe Progrés V
25	Lliurament Memoria + Producte
26	Creació presentació
27	Lliurament Presentació



1.5 Producte

El producte realitzat, implementa les dues funcionalitats bàsiques d'aquest treball. Per una part l'usuari disposa de les opcions de capturar paquets de xarxa, i d'un analitzador de paquets emmagatzemats en un fitxer.

Per altre banda, l'usuari disposa d'un petit firewall que filtrarà els paquets d'acord a les regles que hagi definit. Aquest firewall permetrà al usuari emmagatzemar conjunts de regles per tal de carregar-les més endavant.

Per tal facilitar la feina de l'usuari, s'ha desenvolupat una interfície gràfica que permetrà realitzar les diferents tasques implementades més còmodament.

El producte s'acompanya amb un fitxer d'ajuda accessible en tot moment per tal d'aclarir els termes utilitzats i el funcionament de d'interfície d'usuari.

Per últim i no menys important, trobem aquest document que esteu llegint, la memòria, en ella, es troben des de la planificació del treball fins a la descripció del programari que s'ha realitzat, passant pels conceptes i tècniques que s'han implementat.

1.6 Estructura de la memòria

La resta d'aquesta memòria s'ha dividit en les següents parts:

Al capítol 2 s'introduiran els conceptes bàsics de comunicacions, i funcionament de les xarxes Ethernet, que ens serviran per entendre millor els processos de captura i filtrat de paquets.

El capítol 3 el dedicarem a l'eina de programació “per excel·lència” en aplicacions de xarxa. Els sockets., historia, funcionament, crides i implantacions a Windows i Linux.

El capítol 4 es dedicarà a la captura de paquets, veurem les tècniques per poder escoltar el medi i capturar paquets pel seu anàlisi posterior. Compararem las possibilitats que ens ofereixen el sistemes operatius per aquesta tasca.

El capítol 5 està dedicat al filtratge de paquets i segueix la mateixa estructura que el dedicat a la captura.

El capítol 6 Descriurem el producte realitzat. Començarem enumerant les funcionalitat que tindrà, descriurem la arquitectura que segueix, Tècniques i decisions de disseny i d'implementació realitzada. Comentarem els requisits i el procés d'instal·lació i ús del programari,

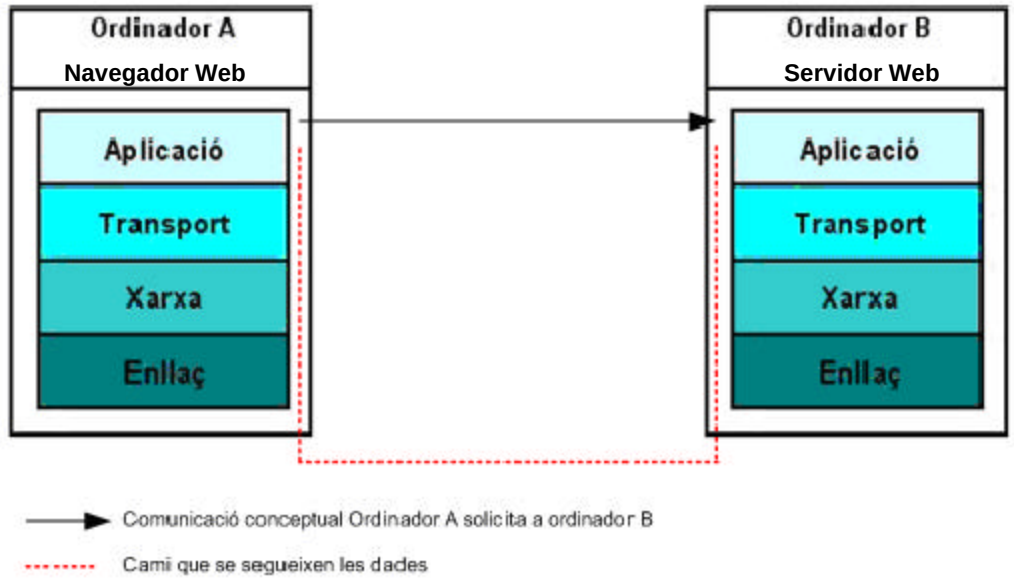
Finalment, acabarem aquesta memòria amb les conclusions, un glossari de termes i els annexos.

2.- Conceptes bàsics comunicacions

En aquest, capítol introduirem els conceptes bàsics de comunicacions, necessaris per entendre el procés de captura, anàlisis i filtrat de paquets. No pretenem explicar tots els detalls, sinó, fixar-nos en els conceptes elementals necessaris que ens permetrà descriure les tècniques de captura i filtrat. Començarem descrivint la pila de protocols TCP/IP, el seu funcionament, i finalment veurem el funcionament de les xarxes Ethernet.

2.1 El procés de comunicació TCP/IP.

Malgrat l'estàndard OSI de comunicacions divideix el procés en 7 nivells o capes, TCP/IP es basa en un model de 4 capes. La següent figura mostra aquestes capes i el procés que segueixen les dades a la comunicació entre 2 ordinadors.



En aquest model quan s'estableix una comunicació, les capes superiors envien les dades a les capes inferiors fins que arriben al medi físic de transport, que les fa arribar al ordinador de destí. A l'ordinador destí les dades pugen de les capes inferiors fins a la superior que s'encarrega de tractar la sol·licitud. En definitiva, cadascun dels protocols que intervenen construeixen el seu propi paquet inserint el paquet construït, pel protocol superior. Al destí es segueix el procés invers. Les funcions de cadascuna de les capes son las següents:

Aplicació : conté els protocols de nivell més alt. HTTP, FTP, Telnet, ssh …

Transport : Es el responsable d'establir la connexió, dividir la informació en paquets i garantir que aquests paquets son entregats correctament. Aquí trobem els protocols TCP i UDP.

Xarxa : Aquesta capa s'encarrega de afegir la identificació de l'ordinador de destí, de decidir si el paquet s'ha de processar localment o ha de ser transmès i de establir el mecanisme per verificar si el paquet es correcte (cheksum). A aquesta capa els paquets s'anomenen datagrames i els protocols que hi actuen son IP, ICMP, ARP.

Enllaç : Finalment, només ens resta adequar el paquet, a la estructura de xarxa utilitzada. Aquest paquet (frame o trama a aquesta capa) serà posat al medi físic de transport pel hardware adient (tarja de xarxa).

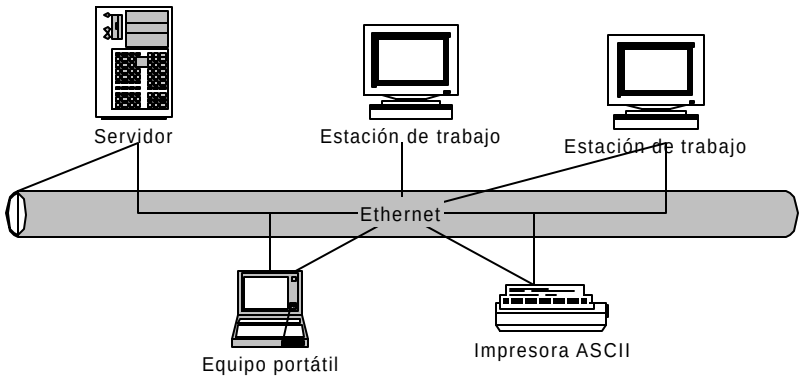
TCP/IP identifica als dispositius amb un mecanisme d'adreces, (adreça IP), al igual que passava amb les adreces físiques, només seran tractats aquells paquets, que vagin a la nostra adreça IP. Un aspecte important es que a cada capa el protocol que intervé afegeix informació al paquet. Aquesta informació dona suport per complir les funcionalitats de cadascun del protocols. Per exemple prenem el cas d'un navegador web que fa una petició a un servidor demanant la plana home.htm la transformació que pateix el paquet es la següent. (S'ha simplificat i no s'ha afegit tota la informació)

Capa	*Protocol*	*Paquet*
Aplicació	***HTTP***	***Get http1.1 \home.htm***
Transport	***TCP***	Port origen, Port Destí, informació control, dades (**Get http1.1 www.miempresa.com\home.htm)**
Xarxa	***IP***	Adreça Origen, Adreça Destí, informació control Dades **(Port origen, Port Destí, informació control, dades (Get http1.1 \home.htm)**
Enllaç	***Ethernet***	Adreça Origen, Adreça Destí, informació control Dades **(Port origen, Port Destí, informació control, dades Get http1.1 \home.htm)**

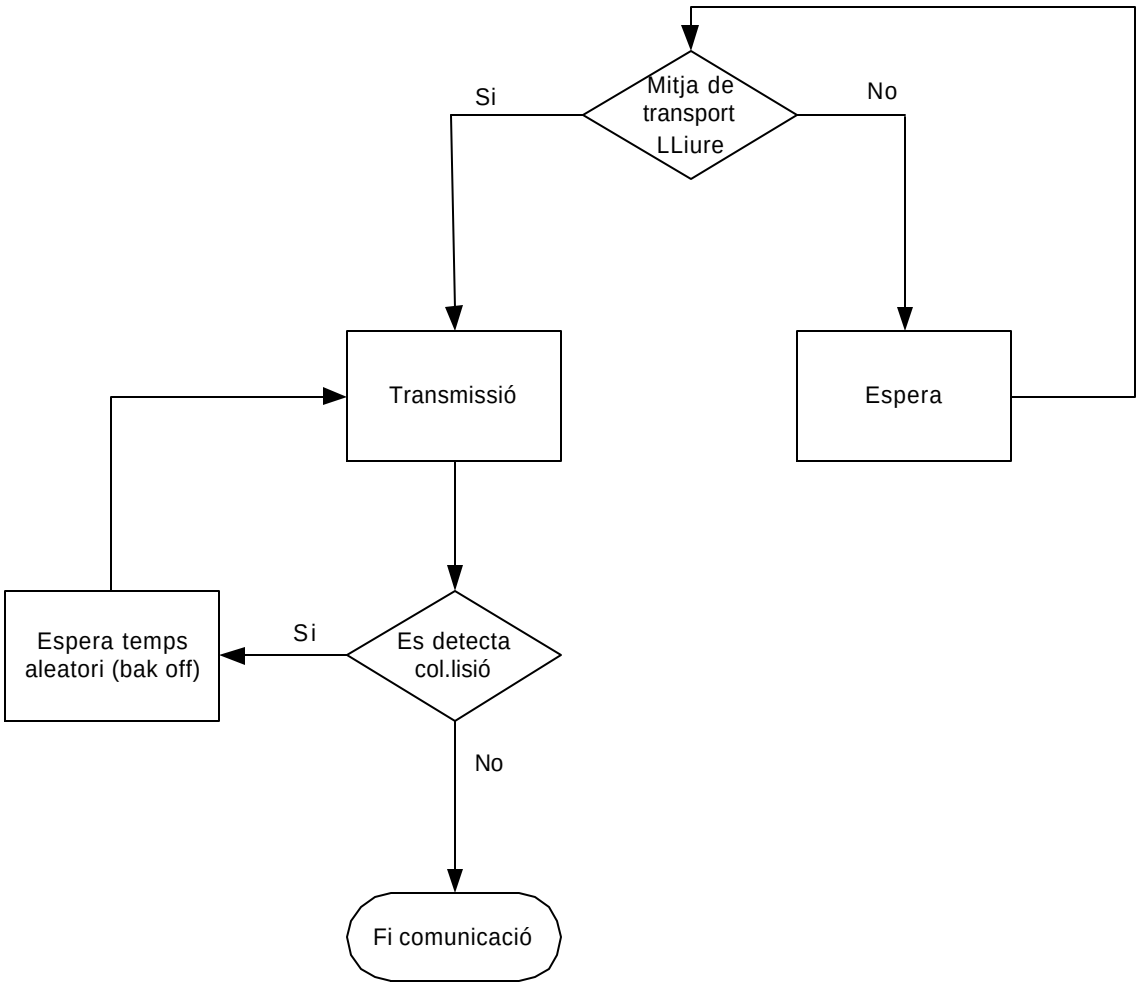
Com podem veure en l'exemple, el navegador fa una petició HTTP, A la capa de transport s'afegeixen el port d'origen i el port de destí així com informació de control,. Els ports son valors numèrics dintre del rang 0-65355. Aquest valor identifica al sistema operatiu la aplicació a la que va destinada la informació. Al nivell de xarxa s'afegeixen les adreces a on s'ha iniciat la petició i l'adreça d'on va destinada. Finalment el protocol de enllaç, s'encarrega de afegir les adreces físiques del origen i destí. Aquest procés es complica en la realitat, per exemple, es possible que tinguem que conèixer primer l'adreça ip del host de destí o l'adreça física, MAC del destí, o si no la podem conèixer-la de la nostre porta d'enllaç, però ens serà útil per adonar-nos de com podrem filtrar paquets (quins camps podem verificar).

2.2 Xarxes Ethernet

Les xarxes Ethernet, es la tecnologia de xarxa més estesa en entorns d'àrea local. A aquesta tecnologia de xarxa. el dispositius comparteixen el medi de transport. Cada dispositiu s'identifica a través d' una adreça física de 6 bytes, anomenada “MAC address”. Aquesta adreça es única e identifica físicament al dispositiu. L'esquema següent mostra una xarxa Ethernet amb topologia de bus.



Ja que el medi de transport es compartit per tots els dispositius de la xarxa, es necessari, disposar de mecanismes que permetin que 2 ordinadors no realitzin una transmissió simultàniament, això es garanteix utilitzant CSMA/CD. Aquesta tècnica segueix el següent diagrama de luxe:



La informació que s'ha de transmetre es formatada en un paquet anomenat trama, aquest paquet a més de la informació a transmetre e informació de control conte 2 dades importants, l'adreça física del dispositiu que inicia la transmissió i l'adreça física del dispositiu destinatari de la transmissió, quan la trama es posada al medi transmissió, es visualitzada per tots el dispositius connectats a la xarxa. Tots aquells dispositius en que la seva adreça física no coincideixi amb la que porta el paquet, rebutjaran la trama. El dispositiu, amb la mateixa adreça física, llegirà la trama i la tractarà.

3.- Sockets

Una vegada vista la teoria bàsica de comunicacions, veurem tot seguit, els sockets, que seran el principal element en els desenvolupaments d'aplicacions de xarxa. Comentarem el seu concepte i la seva historia per sobre, quins tipus existeixen, el seu funcionament i la seva utilitat. Aquest capítol es molt important per tal que, els sockets, seran la estructura principal que farem servir per implementar el nostre producte.

3.1 Sockets. Concepte i Història

Segons el Oxford Dictionnary of the internet in computing un socket es :

"*An entry or exit point for data in a computer which is connected to a TCP-IP network. If a programmer wants to develop a program which reads data from a network, then he or she programmatically sets up a socket on the computer which is going to receive the data. This socket is then referenced in any program which carries out the data transfer. In Java a socket is created by specifying the PORT number through which data transfer is to occur. A socket is a logical concept not a hardware concept.*"[(1)].

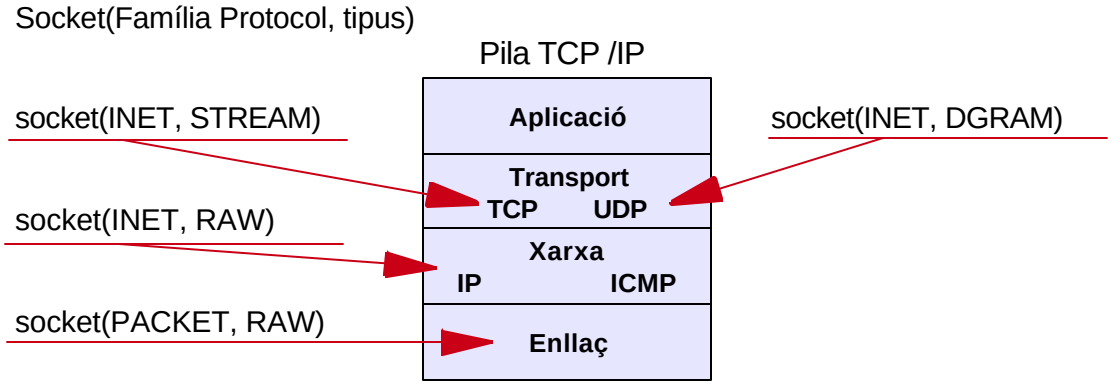
Des de el punt de vista del programador els sockets son estructures que creem a les aplicacions i des de els que enviem i rebem informació. Els sockets uneixen l'adreça del host i un número de port, número que identifica la aplicació, això "avisa" al sistema operatiu que tots el paquets dirigits a la adreça i ports indicats seran entregats a la aplicació que ha creat el socket. A més de enviar i rebre informació, els podem classificar en actius i passius. Els sockets passius esperen a rebre informació (normalment peticions) per enviar una resposta, en aquest cas diem que el socket "escolta" . Els actius, pel contrari, prenen la iniciativa en el procés de comunicació.

Els sockets disposen de un ampli ventall de característiques, que ens permetran, des de comunicar 2 processos al mateix host, comunicar, processos a diferents hosts, fins accedir als protocols subjacents. Aquesta última característica es la que hem triat pel producte que acompanya aquesta memòria.

(1) "socket" A Dictionary of the Internet. Darrel Ince. Oxford University Press, 2001. Oxford Reference Online. Oxford University Press. <http://www.oxfordreference.com/views/ENTRY.html?subview=Main&entry=t12.e2986>"

3.2 Sockets. Tipus i funcionament.

Per crear un socket hem de especificar, entre d'altres 2 paràmetres, un es la família de protocol i l'altre el tipus de socket. Aquests paràmetres configuraran la funcionalitat i el nivell al que actuarà el socket. La següent imatge mostre aquest concepte:



A la figura amb representat 2 famílies de protocols, i han més, INET que fa referència a la pila de protocols TCP/IP i PACKET, que fa referència als paquets, que passen pel nivell d'enllaç, es dir els protocols subjacents.

Com a tipus de socket s'ha mostrat 3 tipus, DATASTREAM, DATAGRAM i RAW. El primer orientat a connexions fiables, el segon per connexions no fiables y el tercer que ens permetrà obtenir o enviar els paquets tal com son llegits pel driver de la tarja o tal com els generi el nostre programa. Una vegada el socket està creat el tindrem que “lligar “amb el dispositiu que utilitzarà per la comunicació, i especificar si volem que connectar amb un altre (connect) o esperar que es connectin al nostre (listen). Finalment el socket, es podrà utilitzar com un fitxer, llegint de ell per rebre dades, o escrivint per enviar-les.

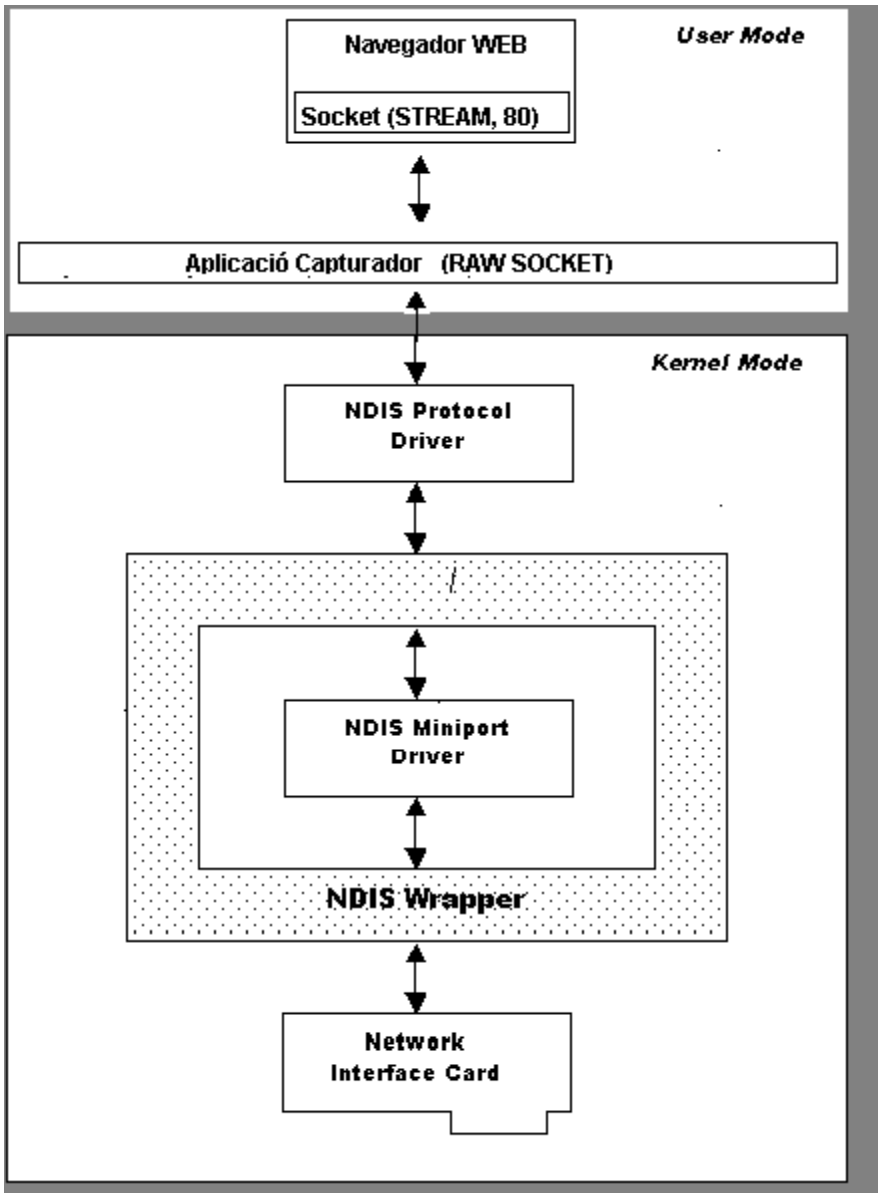
3.3 Sockets. Protocols subjacents.

En el cas concret d'un capturador de paquets, aquest últims passos no els necessitarem. Posar el socket a la espera (listen) o connectar amb un altre socket (connect) només es necessari quan volem enviar o rebre dades des de un origen o cap a un destí. En el nostre cas, volem capturar i analitzar tots els paquets que circulin per la xarxa, per tant, necessitem treballar al nivell més baix possible, per tal de no perdre paquets. No volem establir comunicació entre dos host.

Per tal d'aconseguir-ho, disposem del RAW sockets i la família genèrica de protocols PACKET. Aquesta combinació de tipus i protocol, crearà el socket al nivell més baix, a la capa d'enllaç. Els paquets que arribin a la tarja passen pel socket directament, sense cap modificació. Això obligarà a la nostre aplicació a afegir codi per tractar, el paquet i poder descodificar-lo. Diferenciar capçaleres de dades, detectar e identificar els protocols inserits al paquet, etc.

Al crear un socket RAW, el sistema passarà els bytes rebuts per la tarja al socket abans de enviar-ho a la aplicació de destí. No en tenim el control, i no podem aprofitar el socket per eliminar paquets no desitjats.

A la següent figure podem veure de manera esquemàtica el concepte que acabem de comentar. La figura mostra el flux de paquets en un host (windows) que està executant un navegador web, i una aplicació que crea un RAW SOCKET.



Com podem veure tot el tràfic que rebi el host, el sistema entregarà el missatge al RAW SOCKET i a la aplicació. Això significa que totes dues rebran el paquet intacte d'acord al nivell que estan treballant, En el cas de l'aplicació WEB rebrà les dades que conté el paquet TCP (la resposta del servidor) i en el cas del raw socket, un paquet ip si es un socket de Windows o un paquet Ethernet si es un socket de Linux, especificat amb la família de protocols PACKET. Aquest paràmetre ens permetrà seleccionar el nivell al que volem treballar enllaç o xarxa.

Hem volgut fer aquest esquema perquè com veurem al capítol següent existeixen diferencies en la implementació que fan els diferents fabricants dels sockets. Aquestes diferencies, condicionaran de manera molt significativa el desenvolupament de un analitzador de protocols.

4- Captura de paquets

Un cop vistos els conceptes previs, enfocats totalment al objectiu d'aquest treball. Entrarem directament en el procés de captura de paquets. Farem una petita introducció, replantejant el problema, descriurem el procés de escolta del medi i captura, prestant atenció a les diferencies entre Windows i Linux. Finalitzarem el capítol amb un resum de les crides a modes d'esquema de un programa de captura.

4.1 Introducció

El procés de captura de paquets ens permetrà de veure el tràfic que circula per la xarxa, tal com hem dit abans, la informació circula per un medi compartit. Per tant poder capturar-la ens permetrà realitzar feines de diagnosi de problemes, anàlisis de protocols, o simplement escoltar el medi de transmissió per prendre mesures de l'ocupació i rendiment de la xarxa, malauradament, també s'ha utilitzat per fins no tan lloables i fins i tot il·lícits (descobriment de contrasenyes, espionatge de correus etc.).

Per poder capturar paquets tenim que resoldre 2 problemes, per un costat hem vist que el dispositius de xarxa rebutgen els paquets que no tenen la seva adreça física, per altre, costat, el protocol IP també rebutja tots aquells paquets que no son per la seva adreça IP.

4.2 Captura de paquets

Per capturar paquets, tenim que permetre que la tarja de xarxa, accepti tot el tràfic que circula pel medi de transmissió i no només aquell que va dirigit a la seva adreça física, això es pot aconseguir, desactivant un flag de la tarja, anomenat “promiscu flag”. Amb això aconseguirem que la tarja no rebutgi cap paquet i els passi tots cap al sistema operatiu. Per solucionar el problema del paquets dirigits a altres host (adreces IP diferents), ho podrem resoldre amb els RAW sockets, que com hem vist treballant a nivell de enllaç o xarxa.

Amb tot el que hem comentat fins ara ja estem en disposició de descriure com podem fer el procés de captura de paquets.

El primer que necessitem fer es deixar que la tarja no filtri les direccions físiques, ja hem explicat abans que això s'ha aconsegueix desactivant un bit d'estat de la tarja, per fer-ho, utilitzarem la crida IOCTL per desactivar el bit corresponent.

Per tal de acceptar tot els paquets circulants, ho farem creant un RAW Socket amb família de protocol PACKET. Com hem vist al capítol anterior, aquest tipus sockets donen accés a tots el paquets circulants per la capa d'enllaç, per tal estem treballant per sota la capa de xarxa que es on treballa el protocol IP.

Això que en teoria sembla tan senzill, quan ho posem a la practica, toparem amb les primeres dificultats. Aquestes venen donades per la diferencia entre las implementacions del sockets i les pròpies característiques de cadascun dels sistemes operatius.

El primer problema el trobarem al intentar desactivar el filtre d'adreces físiques, a Linux qualsevol programa que se executi amb permisos de root, no tindrà problemes, en canvi a Windows, per defecte, no està permès l'accés al hardware des de programes d'usuari, per tant si volem accedir a la tarja, tindrem que crear un driver. Aquesta restricció, la podrem evitar, la crida SETSOCKETOPTION ens permetrà obtenir la mateixa funcionalitat, i avisarà al sistema operatiu per que “avisi” al driver i no filtri les adreces MAC .

Amb el segon problema que trobarem, no tindrem tanta sort. El problema ve donat per que Windows no implementa la família de protocols PACKET. Això implica que no podem crear un socket a Windows al nivell d'enllaç.

Aquesta afirmació te una implicació important i és que, des de un programa d'usuari, sense el suport de drivers (entenen per driver qualsevol aplicació que se executa al nivell de kernel), només podrem capturar paquets a nivell de xarxa. Això suposa que els paquets de protocols com ARP, RARP o NetBeui per exemple, no els podrem capturar. Per baixar a més baix nivell tindrem que crear un driver. Dintre del àmbit d'execució del driver podrem fer servir l'estàndard NDIS. Aquest estàndard ens ofereix un API per accedir a la pila de protocols implementada pel sistema operatiu.

3.3 Estructures i crides de programació

No pretenem donar aquí una llista de totes les crides i tots els paràmetres oferts pels sistemes operatius, ja que és molt extensa. Ens centrarem en les crides i estructures bàsiques que ens deixaran capturar paquets i els passos per realitzar-la a cadascun dels sistemes operatius.

Windows

Per tal de capturar paquets amb aplicacions Windows tindrem que seguir els següents passos: Utilitzant un llenguatge com C# o C++.

- Creació del socket:

```
MiSocket =  Socket(AddressFamily.InterNetwork,  SocketType.Raw, ProtocolType.IP )
```

  Aquesta crida ens retornarà un socket. El primer paràmetre indica la família de protocols a utilitzar, el segon, indica el tipus de socket, i per últim el protocol, sobre el que treballarem. Pel nostre objectiu, i a la plataforma que ens trobem, Especifiquem, esquema TCP/IP Versió 4, socket raw i protocol IP;

- Unió del socket amb la interfase de xarxa a la que volem capturar:

```
MiSocket.Bind( <DirIP>);
```

  Aquesta associa el socket amb la adreça IP des de la que volem escoltar el medi. Ens permetrà identificar la tarja des de la que capturarem.

- Forcem opció del socket per capturar tot el tràfic IP.

```
MiSocket.SetSocketOption(SocketOptionLevel.IP,
                                        SocketOptionName.HeaderIncluded, 0);

int SIO_RCVALL          = unchecked((int)0x98000001);
int iErc = MiSocket .IOControl(SIO_RCVALL, aDatIn, aDatOut);
```

- Per últim només ens queda especificar la funció que tractarà els paquets rebuts i assignar-li al socket...

```
AsyncCallback lFunc = new AsyncCallback( <funció per tractar paquets>);
IAsyncResult arParams = MiSocket.BeginReceive(aBuffer, 0, iBuffLength,
SocketFlags.None,  lFunc, this);
```

Com podem veure, el procés es bastant senzill, en funció del llenguatge i l'entorn, les crides poden variar lleugerament,però la idea es la mateixa. Tal com hem dit, amb el que hem vist fins ara només podrem capturar paquets IP i en consecució tots el protocols que porti inserits. A windows per baixar a més baix nivell, tindrem que crear un driver.

Linux

L'esquema que hem de seguir a Linux es el mateix. A continuació mostrem els passos que hem de seguir.

- Creació del socket:

```
MiSocket= socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
```

  Aquesta crida ens retornarà un socket. El primer paràmetre indica la família de protocols a utilitzar, el segon, indica el tipus de socket, i per últim el protocol concret sobre el que treballarem. A Linux, si que disposem de la família de protocols PACKET i com a protocol concret especifiquem ETH_P_ALL, valor que significa qualsevol dels protocols de la capa d'enllaç a xarxes Ethernet.

- Unió del socket amb la interfase de xarxa a la que volem capturar:

```
strncpy(disp_req.ifr_name, "eth0", 4);
ioctl(MiSocket, SIOCGIFINDEX, &disp_req);

bind(MiSocket, (struct sockaddr *) &addr_ll, size_ll);
```

  Aquest codi, simplificat, associa el socket amb la tarja de xarxa des de la que escoltarem i capturarem paquets. A diferencia de Windows, primerament especifiquem el dispositiu pel seu nom, eth0 en aquest cas, i recuperem l'índex del dispositiu amb la crida ioctl. Per fi amb la crida bind unim el dispositiu al socket.

- Forcem opció del socket per capturar tot el tràfic IP.

```
ioctl(MiSocket, SIOCGIFFLAGS, &request);
request.ifr_flags |= IFF_PROMISC;

ioctl(MiSocket, SIOCSIFFLAGS, &request);
```

Aquest codi ens permet posar la tarja de xarxa en mode promiscu. Això desactivarà el filtratge de paquets per IP o mac Address. Primer recuperem el valor dels flags, activem el flag de mode promiscu i assignem el nou valor.

- Rebem paquets.

```
recvfrom(MiSocket, buffer, 4096, 0, (struct sockaddr *) &addr_ll, &size_ll);
```

Aquesta crida ens permetrà rebre les dades rebudes.

Com podem veure, l'esquema es molt semblant a Windows, amb les diferencies a les diferents crides degudes a la implementació realitzada pel sistema operatiu. S'ha de comentar que la informació rebuda a Linux es, el frame Ethernet, això ens obligarà a afegir codi per identificar i tractar el protocol que porta el paquet. Amb les crides que hem vist a Windows només rebem paquets IP, a Linux rebrem tot el tràfic circulant per la xarxa. Per exemple el protocol ARP no el veurem a Windows però si a Linux. La figura següent mostra la estructura d'un frame Ethernet.

Trama de Ethernet

Preámbulo	SOF	Destino	Origen	Tipo	Datos	FCS
7 bytes	1 byte	6 bytes	6bytes	2 bytes	46 a 1500 bytes	4 bytes

El camp “Tipo” ens permetrà identificar el protocol subjacent. Els valors possibles d'aquest camp son mantinguts per un organisme internacional de estandardització el IEEE EtherType Field Registration Authority. Per exemple el valor =0x800 correspon al protocol IP V4, 0x806 a ARP, 0x8137 a IPX etc.

5.- Filtrat de paquets

A aquest capítol veurem el filtratge de paquets, seguirem un esquema similar al que hem vist al capítol anterior, començarem definint el problema i veurem per sobre els conceptes i la història de les eines de filtratge, continuarem amb les tècniques de filtratge de paquets i acabarem amb les crides i estructures ofertes pels sistemes operatius que estem estudiant.

5.1 Introducció

Al filtratge de paquets, ens plantegem just el problema contrari, al que hem vist a la captura, amb la captura volíem veure tot el tràfic circulant, ara el que volem es rebutjar o descartar, tràfic que va dirigit a la nostra màquina.

Segons el Oxford Dictionnary of the internet in computing podem definir un firewall com :

"firewall A system designed to control the passage of information from one network into a second network. Typically a firewall will be used as a means of reducing the risk of unwanted access to sensitive systems, where one carefully regulated network contains the sensitive systems and is connected to a larger less-regulated network. A firewall can be effective if access to the firewall itself is carefully regulated."

Com podem veure un firewall es un element de seguretat, permetrà el pas dels paquets o els rebutjarà en funció de la configuració realitzada.

La capacitat de filtrar paquets es anterior a Linux que a Windows. Linux disposa de mòduls de filtrat dese la versió 1.1 en que es va afegir una adaptació de l'eina de BDS ipfw, cap a finals de 1994 va ser millorada i s'afegí la interfície d'usuari per configurar-la, la eina ipfadm. Al 1998 es reescriu totalment el mòdul de filtratge i neix netfilter, el qual es configurava amb ipchains. Finalment al 1999 es realitzen algunes millores al mòdul i es substitueix la eina de configuració ipchains per la actual iptables. Sobre aquesta s'han realitzat una serie de millores i s'ha afegit el suport per NAT o emmascarant IP per exemple. Windows, en canvi, no va disposar de capacitat de filtratge de paquets fins a la versió Windows 2000 en que es van afegir aquestes funcionalitats al propi sistema operatiu, No va ser fins a la sortida de Windows 2003 o del service pack 2 per Windows XP que els usuaris van disposar d'una eina oferira pel propi sistema per configurar el filtrat de paquets.

5.2 Tècniques de filtrat de paquets.

A diferencia de la captura de paquets, trobem diferencies molt importants a l´hora d'implementar aquesta funcionalitat. A windows disposem d'un interfície gràfic que actua directament amb el mòdul de filtratge, les possibilitats son bastant pobres, basicament podem configurar ports o aplicacions, s´ha de tenir en compte que està mes pensat per un usuari domèstic, no es adequat com un firewall per entorns més grans. A Linux, per defecte, no disposem de cap eina gràfica, la configuració es guarda en un fitxer de script. La capacitat de configuració, a Linux es molt més gran.

A windows disposem de la llibreria IPHLPAPI.dll. Es la eina que ens permetrà configurar el modul de filtratger. Aquesta llibreria ens permetrà afegir les regles que decidiran quins paquets son acceptats o rebutjats i a quina tarja de xarxa actuaran.

El modes de funcionament es molt simple, es fixa la interfície de xarxa a la que es vol filtrar paquets i es defineix un comportament per defecte, llavors las regles actuen com patrons que neguen el comportament per defecte seleccionat. Per exemple, Si definim que per defecte es rebutgi tots els paquets, només seran acceptats tots aquells paquets que compleixin alguna de les regles establertes, i al contrari, si definim un comportament per defecte de acceptar tots els paquets, aquells que compleixin alguna de les regles seran rebutjats.

A part del funcionament comentat, les possibilitats de filtratge ofertes ens permetran utilitzar els camps adreça, port, tan d'origen com de destinació, protocol i si volem aplicar la regla sobre un paquet d'entrada, de sortida o tots 2 sentits.

En el cas que necessitem mes funcionalitats o estem desenvolupant un firewall amb més prestacions tindrem que recorre al seu desenvolupament. De la mateixa manera que amb la captura de paquets, ens veurem obligats a desenvolupar un driver per tal d'accedir al tractament que fa el sistema del entorn de xarxa. La idea subjacent d'aquest driver es en teoria senzilla i consta del desenvolupament de les funcions de filtrat i de la seva instal·lació com ha funcions de “callback” del driver IP del propi. Aquestes funcions seran cridades per cada paquet rebut, el valor retornat per la funció avisarà al sistema la acció a realitzar sobre el paquet. DROP per eliminar el paquet, FORWARD per acceptar-ho. Evidentment al tractar-se d'un driver s'executa a nivell de kernel. Aquesta tècnica es coneix amb el nom “hook” i es pot aplicar a altres aspectes.

Per crear el nostre firewall a Linux partint de les eines del propi sistema, només ens caldrà realitzar una aplicació “maquilladora” que a partir de les entrades realitzades per l'usuari crei, modifiqui e invoqui el fitxer de script amb les regles de configuració del firewall.

Es important recordar en aquest punt que estem treballant sobre un escenari, en el que ens fixat realitzar una aplicació de filtrat partint de les eines pròpies de cada sistema. Evidentment, en un altre escenari en el que les funcionalitats que s'ofereixen son insuficients, a tots 2 sistemes hem de desenvolupar les funcionalitats requerides. A Windows com un driver, o a Linux com una extensió o mòdul del Kernel.

5.3 Estructures de programació

A continuació veurem les crides i estructures principals que ens permetran crear el nostre firewall. Per Linux no inclourem cap. Com hem comentat anteriorment, per tal de configurar el filtrat de paquets, només hem de crear un fitxer de script amb les regles. Per fer aquesta tasca, només ens caldran les funcions estàndards d'obrir, llegir i escriure a fitxers ofertes per qualsevol llenguatge de programació.

Windows

Totes les funcions que veurem tot seguit, resideixen a la dll IPHLPAPI.

Per crear el filtrat de paquets tindrem que seguir els següents passos:

- Creem la “interface” i la unim a la adreça IP de la tarja a la qual volem filtrar.

```
lErc  =  PfCreateInterface(0,  PF_ACTION_FORWARD,  PF_ACTION_FORWARD,
   FALSE, TRUE, hInterface);

lErc = PfBindInterfaceToIPAddress(*hInterface, PF_IPV4, (PBYTE)&lIp);
```

La primera crida crea la interface. Com a paràmetres passem 0 per indicar que creem un interface nou, a continuació venen les accions per defecte pels paquets d'entrada i sortida, en aquest cas es permet tots els paquets. El valor false indica que no utilitzarem la capacitat de LOG. A continuació especificarem si es tracta de un interface únic o compartit per altres processos per últim indiquem un paràmetre de rebuda que en cas d'èxit de la funció s'omplirà amb el handle al interface creat.

La segona crida uneix la interface creat amb la adreça IP, a la que volem filtrar. Els paràmetres són clars, la interface creat a la crida anterior, el identificador del protocol a utilitzar i l'adreça a lligar.

- Afegim les regles.

lErc = PfAddFiltersToInterface(*(hInterface), 1, &ipFlt, 0, NULL, NULL);

En aquest cas estem afegint una regle que s'aplicarà als paquets d'entrada. EL primer paràmetre indica el handle al interface, el segon paràmetre indica la quantitat de filtres que porta la estructura. Aquesta estructura es la que trobem al 3 paràmetre, la veurem amb més profunditat tot seguit. El 4 i cinquè paràmetres son idèntics al 2 i 3 però en contes d'aplicar-se als paquets de entrada son els que s'aplicaran als paquets de sortida. L'últim paràmetre es un paràmetre de rebuda que s'omple amb un array amb tots els handles als filtres establerts.

La estructura de ipFlt es del tipus PF_FILTER_DESCRIPTOR i la seva definició es la següent

```
typedef struct _PF_FILTER_DESCRIPTOR {
  dwFilterFlags;  --> Aquest camp només soporta la constant
                      FD_FLAGS_NOSYN;
  dwRule;         --> Contador de la regla al filtre
  pfatType;       --> Especifica el tipus de adreça IPV4 o IPV6
  SrcAddr;        --> Adreça d'origen dels paquets a filtrar.
  SrcMask;        --> Màscara de l'adreça d'origen els paquets a filtrar.
  DstAddr;        --> Adreça de destí dels paquets a filtrar.
  DstMask;        --> Màscara de l'adreça de destí els paquets a filtrar.
  DwProtocol;     --> Enumerat que identifica el protocol a filtrar.
  WsrcPort;       --> Port d'origen que volem filtrar.
  WdstPort;       --> Port de destí a filtrar.
  SrcPortHighRange; --> Limit del rang de ports d'origen a filtrar. Es
            filtraran tots els ports entre WsrcPort i  SrcPortHighRange.
  WdstPortHighRange; ---> Limit del rang de ports de destí a filtrar. Es
            filtraran tots els ports entre WdstPort i  WdstPortHighRange.
}
```

Com es pot deduir, aquesta estructura es la que conté les dades de la regla.

- Per desactivar el filtrat de paquets.

lErc = PfUnBindInterface(*hInterface);
lErc = PfDeleteInterface(*hInterface);

Aquestes crides tenen com a únic paràmetre el handle al interface creat.

6 Producte realitzat

6.1 Introducció

Per il·lustrar tot el que s'ha explicat fins ara, s'ha desenvolupat un producte, que implementa alguna de les tècniques descrites en aquest treball. El producte realitza les 2 funcionalitats principals, captura i filtre paquets.

El producte ha estat realitzat per Windows sota framework .Net versió 1.1. Per triar les tècniques de captura i filtrat de paquets ja hem comentat, que hem donat prioritat a desenvolupar totalment alguna de les tècniques descrites al treball, en contes d'utilitzar alguna llibreria ja desenvolupada, encara que això suposi, perdre potencia o capacitat. No podem obviar que altres factors com poden ser el coneixement de les eines de desenvolupament i el temps en que s'ha de realitzar, han influït també a la decisió.

5.2 Capacitats i funcionalitats

Les capacitats que volem que tingui el nostre producte son :

Captura i Anàlisis.

- Capturar paquets circulants per la xarxa i emmagatzemar-los a un fitxer.
- Identificar els protocols i la informació que ofereixen tan a la capçalera com a les dades que porten

Filtrat de Paquets.

- Engegar o parar el filtrat de paquets..
- Definir un comportament per defecte
- Definir i crear regles.
- Aplicar conjunt de regles prèviament definides.

5.3 disseny e implementació realitzada

Tal com s'ha comentat, la separació existent entre les dues funcionalitats a implementar ha permès pensar en el disseny per separat per cadascuna d'elles. A continuació explicarem el disseny realitzat en tres passos, captura de paquets, filtrat de paquets i interfície d'usuari. A cadascuna de les tres parts aprofitarem per explicar les decisions d'implementació.

Generalitats.

Tal com s'ha comentat, s'ha realitzat un procés d'investigació sobre les funcionalitats. Durant aquest procés s'han realitzat també, algunes proves de codificació que han sigut aprofitades per completar aquesta memòria, com es el cas del codi que hem posat com exemple a la captura de paquets en Linux. Aquestes proves també han servit per verificar la complexitat de alguna de les tècniques explicades i valorar la seva implementació total i poder afinar la planificació.

Inicialment aquestes proves també van servir per decidir el llenguatge d'implementació. La decisió va ser realitzar-ho en c# .Net per plataforma Windows. La necessitat d'acotar temporalment el projecte, el temps planificat que se'l podia dedicar, coneixements previs de .Net i nuls coneixements programació de interfícies gràfiques a Linux van ser factors decisius per tal de prendre aquesta decisió.

Captura de paquets.

El nucli de la captura es realitzat per una classe, des de la que es realitza tot el procés. Aquesta classe te el suport d'altres que estan orientades a funcions no directament relacionades amb la captura de paquets.

Per tal de no consumir molts recursos de màquina, el procés de captura i anàlisis de trafic s'ha dividit en 2 fases

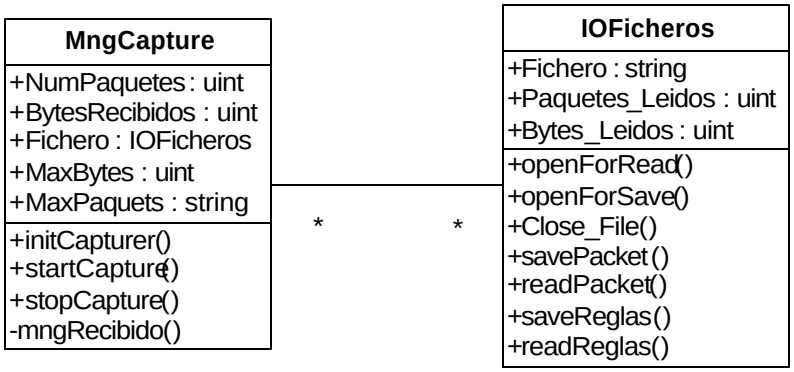
- La captura de paquets
- El anàlisis de paquets

Això s'ha realitzat així per tal de no descodificar els paquets durant el procés de captura.

Captura de Paquets

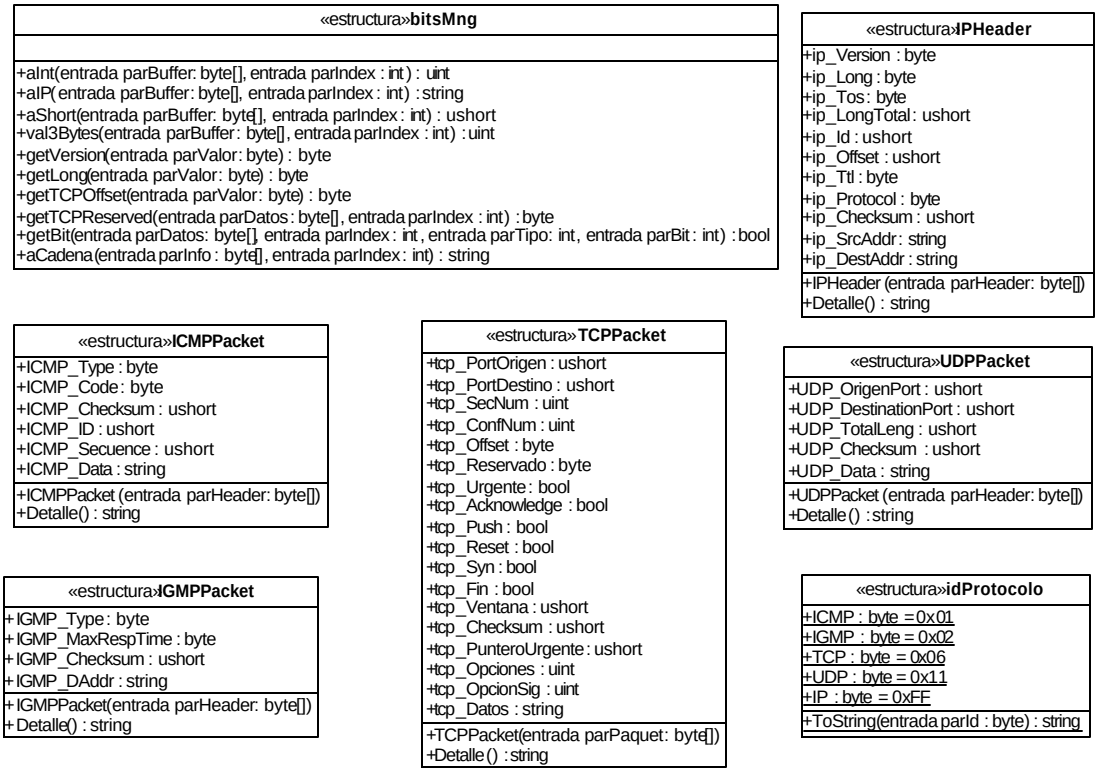
La idea de funcionament implementada es molt senzilla, quan l'usuari inicia el procés de captura, crida al mètode startCapture, que inicia el procés. Els paquets son rebuts com arrays de bytes. Aquests son emmagatzemats en el fitxer configurat per l'usuari, tal com han arribat.

El diagrama següent mostre les classes que intervenen en aquest procés



Anàlisis de Paquets

Aquest procés es realitza a partir del fitxer capturat. Es seleccionat des de la interfície gràfica. Ja que només som capaços de capturar paquets IP, la estructura del fitxer es molt simple. Primer llegim els 20 primers bytes corresponents a la capçalera IP rebuda. A partir de la descodificació d'aquesta capçalera es dedueix la mida de les dades, que correspon al protocol inserit. Aquest bytes son llegits d'un cop, aquest procés es va repetint fins a arribar al final del fitxer. Per tal de facilitar la descodificació del protocol s'han creat unes estructures que es mostren a la figura de la plana següent.



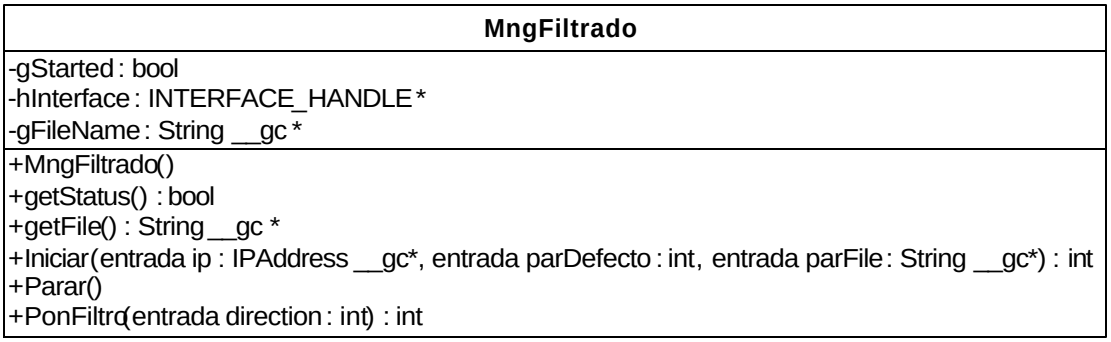
A part de les estructures de suport a la descodificació dels protocols, ja esmentades, caldria fer notar dues classe no comentades, idProtocolo, aquesta classe la fem servir per identificar el protocol que estem tractant i bitsMng, que la utilitzem per realitzar operacions de bits en la descodificació de les capçaleres, identificar els flags de TCP per exemple.

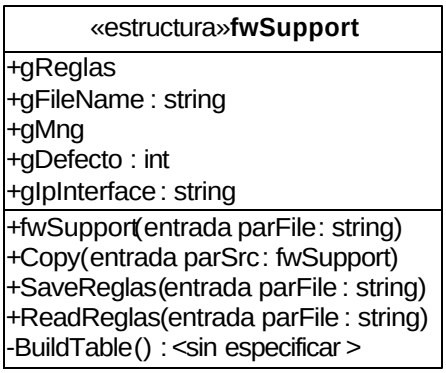
Una vegada llegit el fitxer la informació es presentada a l'usuari.

Filtrat de paquets.

El nucli del filtrat també s'ha implementat a una classe diferenciada. Aquesta classe consisteix en un “Wrapper” de la dll que implementa l'accés a les capacitats de filtratge de Windows. Val a dir, que durant les proves prèvies realitzades, es van detectar problemes a l'hora d'accedir a les funcions de la dll. Aquestes problemes, feien referència a tipus de dades que 's'han de passar a les funcions i accés a algunes funcions com “malloc” per assignar memòria. S'ha de recordar que a C# no existeixen els punters des de “managed code”. Aquests problemes van fer decidir que aquesta classe s'implementés en C++, sempre en un entorn de “managed code” propi de .Net.

A més a més, s'ha creat una classe de suport, per tal d'emmagatzemar les regles. La figura següent mostra les classe implementades. També s'ha de comentar que per raons de espai i claredat al mètode ponFiltro no apareixen tots els paràmetres. Els valors que falten son els corresponents a l'estructura PF_FILTER_DESCRIPTOR explicada al capítol 5.

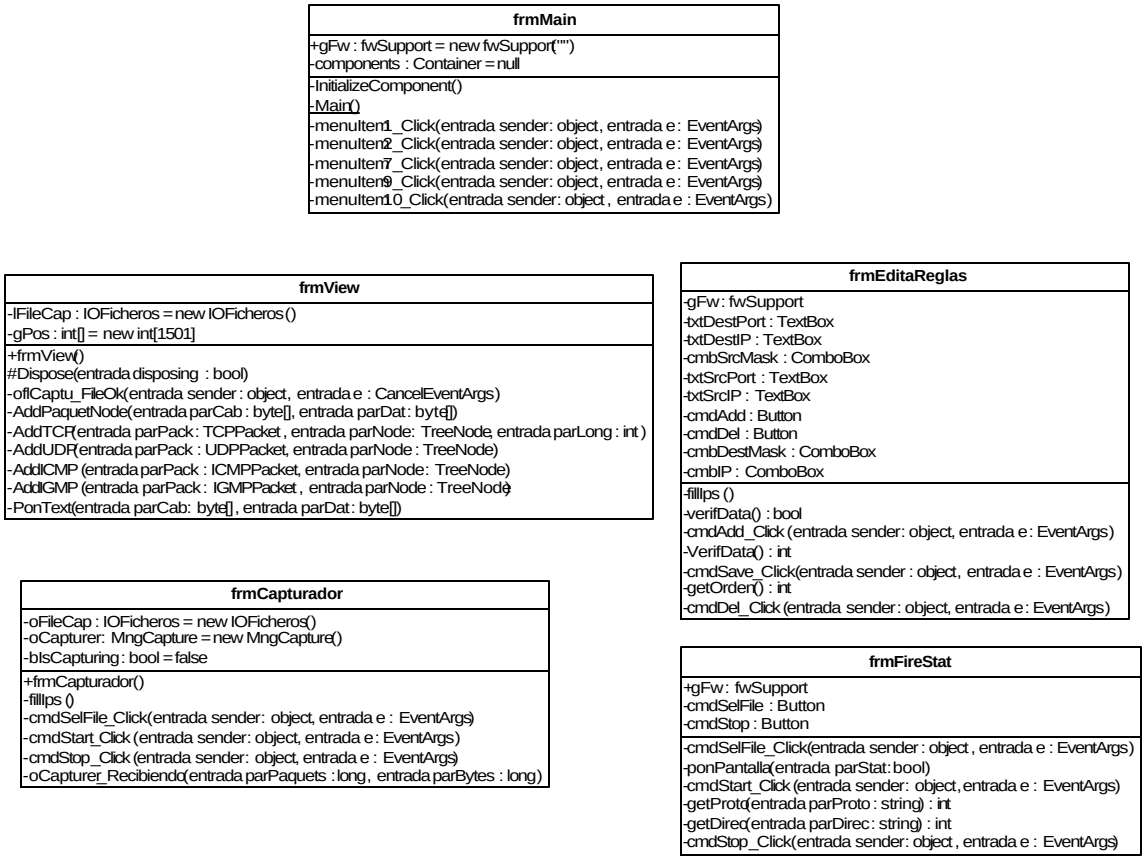




Interfície gràfica.

La interfície gràfica, en principi actuarà com una màscara per utilitzar les classes que implementen les funcionalitats, per recollir les dades necessaris i verificar la seva coherència. En aquest punt s'ha afegit una classe que s'ha propagat a tota la resta per tal de controlar els error propis generats a l'aplicació. Aquesta classe hereta de exception.

Per la interfície gràfica s'ha optat per un model MDI. Aquest model permet una millor organització del treball des de el punt de vista de l'usuari. A la imatge següent es mostra totes les interfícies creades.



També en aquest cas, no s'han mostrat tots els atributs, per clarificar el diagrama.

5.4 Proves i Test

Una vegada finalitzat el desenvolupament, s'ha verificat el producte. S'han dissenyat jocs de proves per totes dues funcionalitats. Els resultats i les proves realitzades son les següents:

- Captura de paquets

Prova	Resultat
Captura execució comanda ping (ICMP)	OK
Captura acció “envia / Rebre” des de client correu Outlook (SMTP i POP)	OK
Captura asignació IP (DHCP)	OK
Captura navegació per Internet (HTTP)	OK
Captura accés recurs compartit de xarxa	OK

Totes aquestes accións han estat realitzades amb el producte i a la carpeta PVHXarxaEx es troben els fitxers amb les captures realitzades. La nomenclatura seguida ha sigut <nom_protocol>.cab

- Filtrat de paquets

Pel filtrat de paquets també s'han realitzat, tot un seguit de tests, per verificar la correcció del producte. Per fer aquestes proves, s'han utilitzat els 2 ordinadors disponibles, tots dos executen el producte. Un realitzaba el filtrat i l'altre capturaba el tráfic, verificant la aplicació de les regles de filtrat. Les proves realizades han sigut:

Prova	Resultat
Bloqueig protocol ICMP	OK
Bloqueig de paquets entrants desde el port 80	OK
Bloqueig paquets amb destinació al port 53	OK
Bloqueig paquets amb destinació al port 25 (SMTP)	OK

5.5 instal·lació i ús

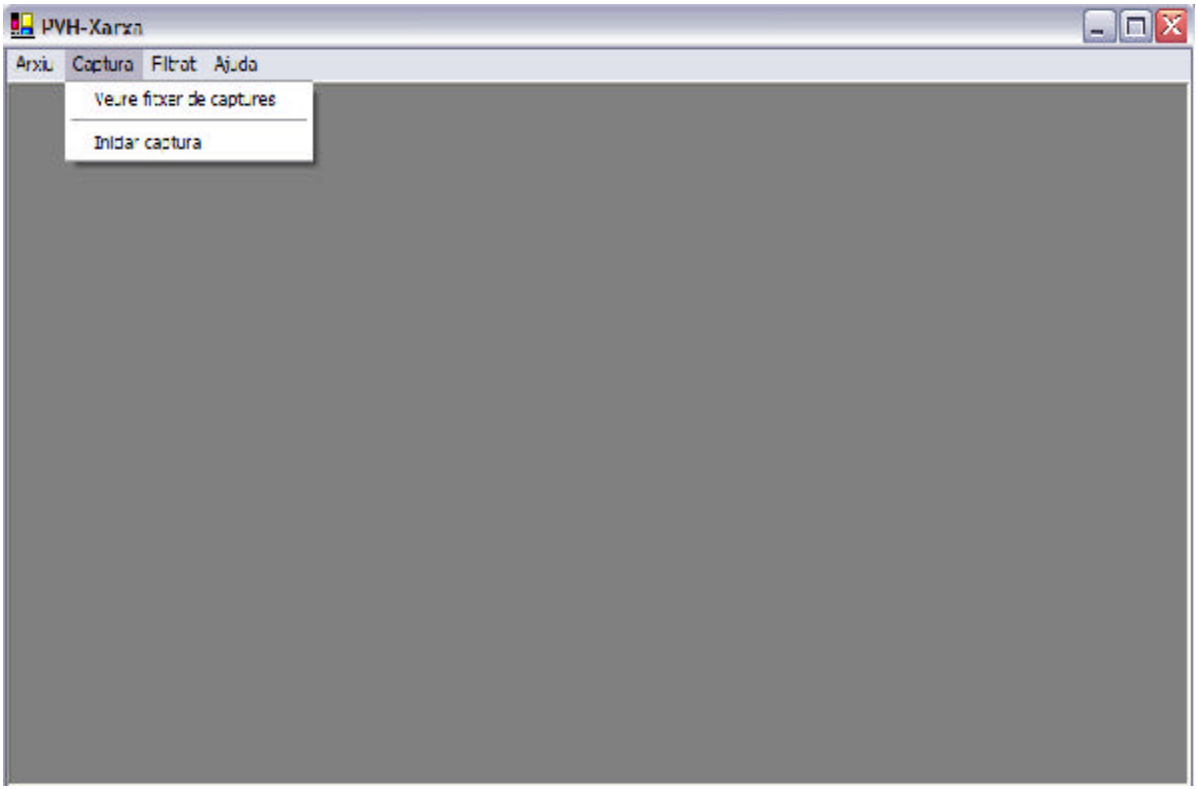
Havans de instal·lar i executar el producte necessitem el .Net Framework versió 1.1. Aquest paquet es pot baixar de manera gratuïta a la següent adreça de la pàgina de Microsoft:

http://www.microsoft.com/downloads/details.aspx?displaylang=es&FamilyID=262d25e3-f589-4842-8157-034d1e7cf3a3

Un cop baixat executarem el fitxer i seguirem el passos del assistent per completar la seva instal·lació.

La instal·lació del producte es més senzilla. Només caldrà extreure la carpeta anomenada pvhXarxaExec del fitxer zip de la entrega i guarda-la al disc dur del nostre ordinador.

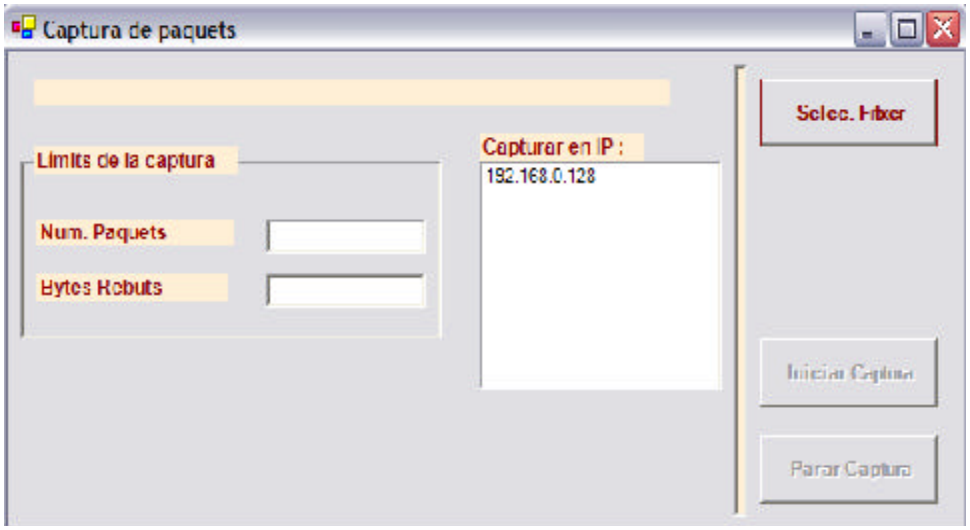
Per executar el producte farem doble clic sobre el fitxer pvhXarxa.exe, amb aquesta acció es desplegarà la finestra principal de la aplicació. Aquesta pantalla disposa d'un menú des de el que podrem accedir a totes les funcionalitats. La pantalla següent mostra una imatge d'aquest formulari.



La aplicació es MDI, per tal, totes les opcions s'executaran dintre el marc de la finestra principal. Tot seguit explicarem l'us de cadascuna de les opcions:

Iniciar Captura:

Aquesta opció ens permetrà capturar paquets circulants per la xarxa i emmagatzemar-los a un fitxer pel seu posterior anàlisis. L'us d'aquesta opció es el següent:



Primer seleccionarem el fitxer on volem guardar la captura, per això farem clic al botó “Selec Fitxer”. Aquest, obrirà el quadre de diàleg “Obrir Fitxer” des de el que seleccionarem el path i el nom del fitxer a crear.

A continuació especificarem un limit per la captura. O be un número de paquets o be una quantitat de bytes rebuda.

Després seleccionarem l'adreça IP des de la que volem capturar. Per fer-ho farem clic sobre l'adreça IP de la llista.

Per últim només ens quedarà fer clic al botó “iniciar captura” per executar el procés.

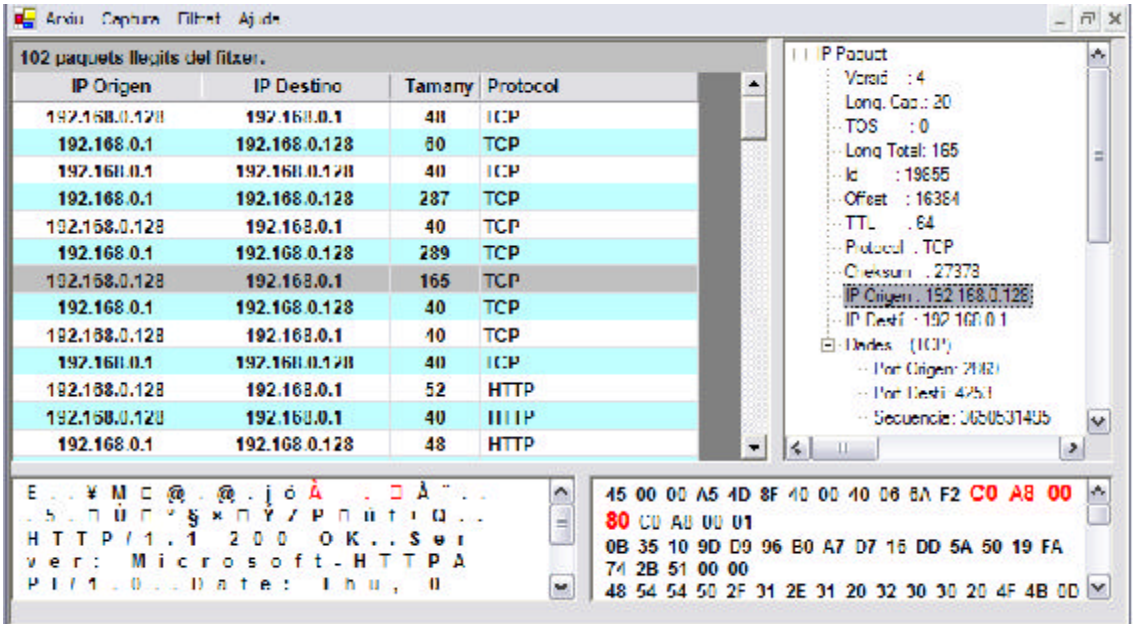
Durant la captura apareixerà una barra de progrés indicant els paquets rebuts. Aquesta finalitzarà automàticament al superar el limit establert L'usuari pot parar el procés en tot moment fent clic al boto Parar Captura.

Veure fitxer Captura:

Aquesta opció obrirà un fitxer de captures i mostrarà el seu contingut. El sistema mostrarà una taula amb les adreces d'origen i destí, la mida del paquet i el protocol que porta el paquet inserit. Clican sobre aquesta taula obtindrem tota la informació del paquet triat.

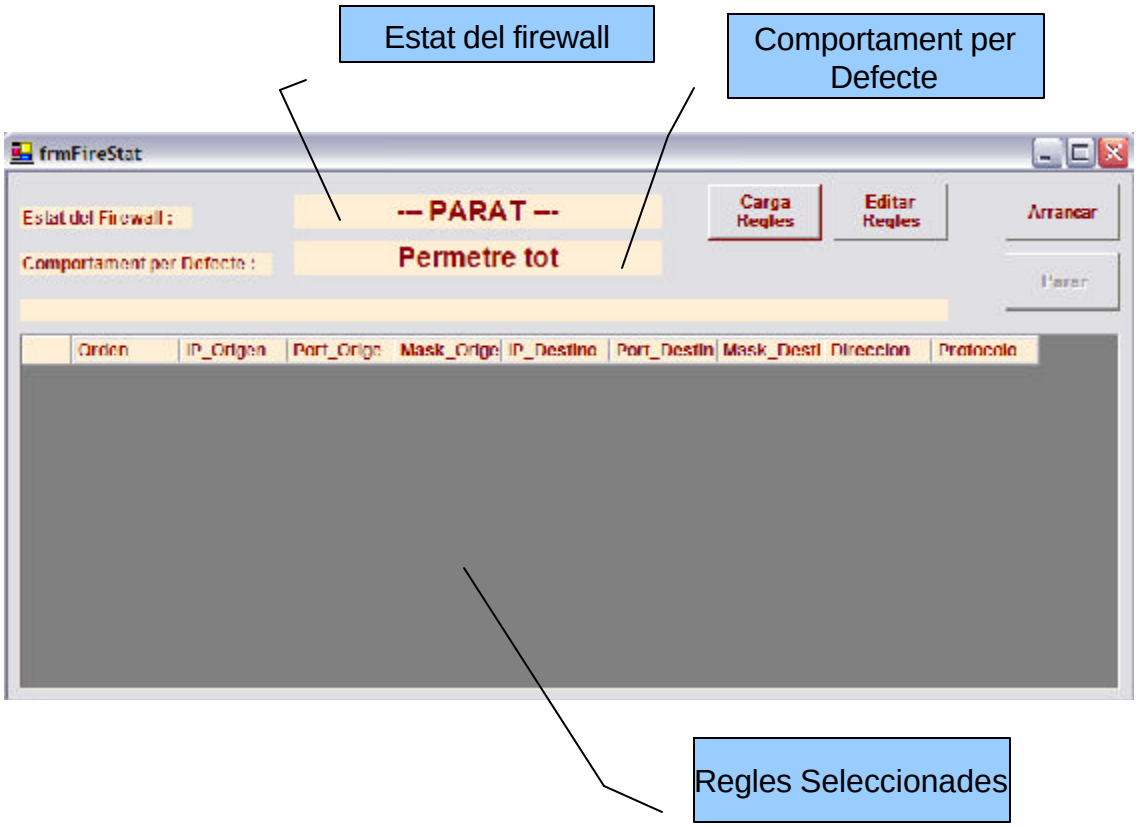
El detall del paquet es carrega en un arbre a la dreta de la taula, amb els camps que composen la capçalera. Hi han dues àrees de text, a la primera podrem veure el paquet complert en format llegible, (els caràcters que no es poden visualitzar s'han substituir per un punt. A la dreta trobarem la seqüència de bytes que hem rebut a la captura.

A seleccionar en un camp del arbre automàticament es ressalten en una font més gran i de color vermell el byte o bytes que porten la informació seleccionada.



Filtrat de Paquets:

Aquesta opció implementa el firewall. Al accedir-hi, entrem a la finestra d'estat del firewall, des de ella podrem realitzar totes les funcions de filtrat. La imatge següent ens mostra aquesta finestra i el seu ús:

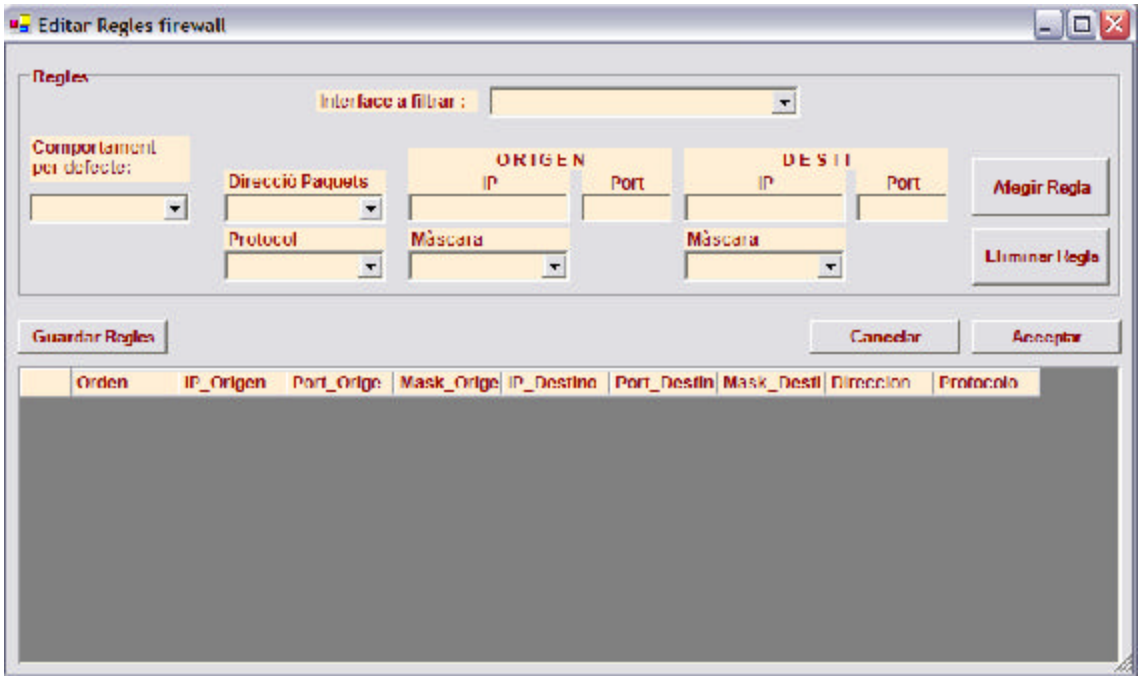


Des de el botó “carga Regles” accedirem al quadre de diàleg que ens permetrà seleccionar un fitxer de regles prèviament emmagatzemat

El botó “Arrancar” engegarà el firewall aplicant les regles seleccionades. Be des de un fitxer, o des de el editor de regles.

EL botó “Parar” detendrà el funcionament del firewall i deixarà de filtrar-se els paquets que ens arribin.

Des de “Editar Regles” accedirem al quadre de diàleg de editar regles, el seu funcionament i l'aspecte que te es el següent:



Des de l'agrupament, “Regles” introduirem les dades dels paquets a filtrar. El botó “Afegir Regla” l'afegirà a l'àrea de regles seleccionades. Abans d'afegir una regla es verifica que les dades introduïdes siguin correctes, no introduint la regla i avisant a l'usuari en cas de error.

El botó “Eliminar Regla” Esborra la regla seleccionada a la llista de regles creades.

Des de “Guardar regles” accedirem al quadre de diàleg que ens permetrà guardar en un fitxer el conjunt de regles creat.

“Acceptar” Tanca el quadre de diàleg conservant les modificacions realitzades que quedaran reflectides a la finestra d'estat del firewall.

“Cancel·lar” Tanca el quadre de diàleg descartant els canvis realitzats.

6 Conclusions

Des de un punt de vista tècnic, sorprèn que per les tasques a realitzar disposem de més capacitat i facilitats de desenvolupament a Linux que a Windows. Des de una senzilla aplicació que s'executa a nivell d'usuari es pot arribar a un accés mes complert i profund a Linux que el que podem realitzar a Windows.

També ha sigut emocionant anar descobrint detalls de funcionament i d'implementació que fan els sistemes operatius de la pila de protocols o d'altres aspectes de desenvolupament.

Les tècniques descrites també poden ser aprofitades per altres tasques, com per exemple anàlisi de rendiment, construcció total de paquets de xarxa que poden ser enviats al medi.

Ara, al final d'aquest projecte, es pot dir que malgrat, les dificultats, ha sigut un projecte per passar-ho be, a nivell personal, els objectius fixats s'han aconseguit àmpliament i ha sigut gratificant veure funcionant les implementacions realitzades.

7 Millores de producte

El producte implementa una base sòlida de captura de paquets si s'orienta a la pila de protocols TCP/IP. Les millores aniran orientades a ampliar les funcionalitats o potencia del producte. Aquestes poden ser:

- Afegir un filtre per tal poder capturar aquell tràfic que ens inste resi i no tot com es fa actualment.
- Es pot millorar l'anàlisi de protocols, buscar seqüències de paquets o afegir més protocols.
- Afegir opcions per tal d'arrancar el programa i engegar el filtrat de paquet.
- Implementar un driver per capturar més paquets de la capa de enllaç.
- Portar el producte a Linux.

8. Glossari.

Aquest glossari recull tots aquells termes utilitzats a la memòria i que per no ser directament objecte de l'estudi no s'han explicat.

Byte

Unitat mínima d'emmagatzemament d'informació a un ordinador, normalment composada de 8 bits.

Driver

Programa informàtic que permet al sistema operatiu comunicar-se amb un dispositiu.

Hook

Nom en general que rep la tècnica d'enganxar al sistema operatiu funcions pròpies per ampliar o modificar las funcionalitats oferides. Les tècniques particulars s'anomenen d'acord al àrea a tractar, per Ex filter-hook, per enganxar-se al filtre, Message-hook per capturar els missatges emesos per Windows.

Handle

Un handle es un valor numèric que serveix per identificar i accedir objectes creats a l'execució d'un programa.

Managed Code

Aquest terme se aplica a qualsevol programa d'ordinador que es executat per una màquina virtual. En l'àmbit de la memòria se utilitza pels programes realitzats en .Net framework executats per la CLR.

MDI

Tècnica de desenvolupament d'interfícies gràfiques en la que totes les finestres s'ubiquen dintre d'una finestra pare.

NDIS

Sigles que corresponen a **Network Driver Interface Specification.** Es un API de programació de tarja de xarxa, usat principalment en la implementació de controladors de dispositius de xarxa.

9. Bibliografia.

La bibliografia i fonts d'informació consultades son las següents

Jhon Gatrell, Jhon Karas i altres, *TCP/IP tutorial and techinal overview*, IBM Redbooks

IEEE Registration Authority
http://standards.ieee.org/regauth/ethertype/index.shtml

El projecte NETFILTER
http://www.netfilter.org

Desenvolupament de Firewalls a Windows 2000/XP
http://www.codeproject.com/KB/IP/drvfltip.aspx

El projectte Winpcap
http://www.winpcap.org

Web de Ethereal
http://www.ethereal.com

Gianluca Insolvibile, Inside the Linux Packet filter. Linux Journal article Març 2002.
http://www.linuxjournal.com/article/5617

Inumerables articles MSDN microsoft.
http://msdn2.microsoft.com/es-es/default.aspx

Firewalls Complete, 1997 The McGraw-Hill Companies, Inc.

Tony Field, Uli Harder & Peter Harrison, Analisys of network traffic in switched Ethernet systems.
6 Novembre 2001