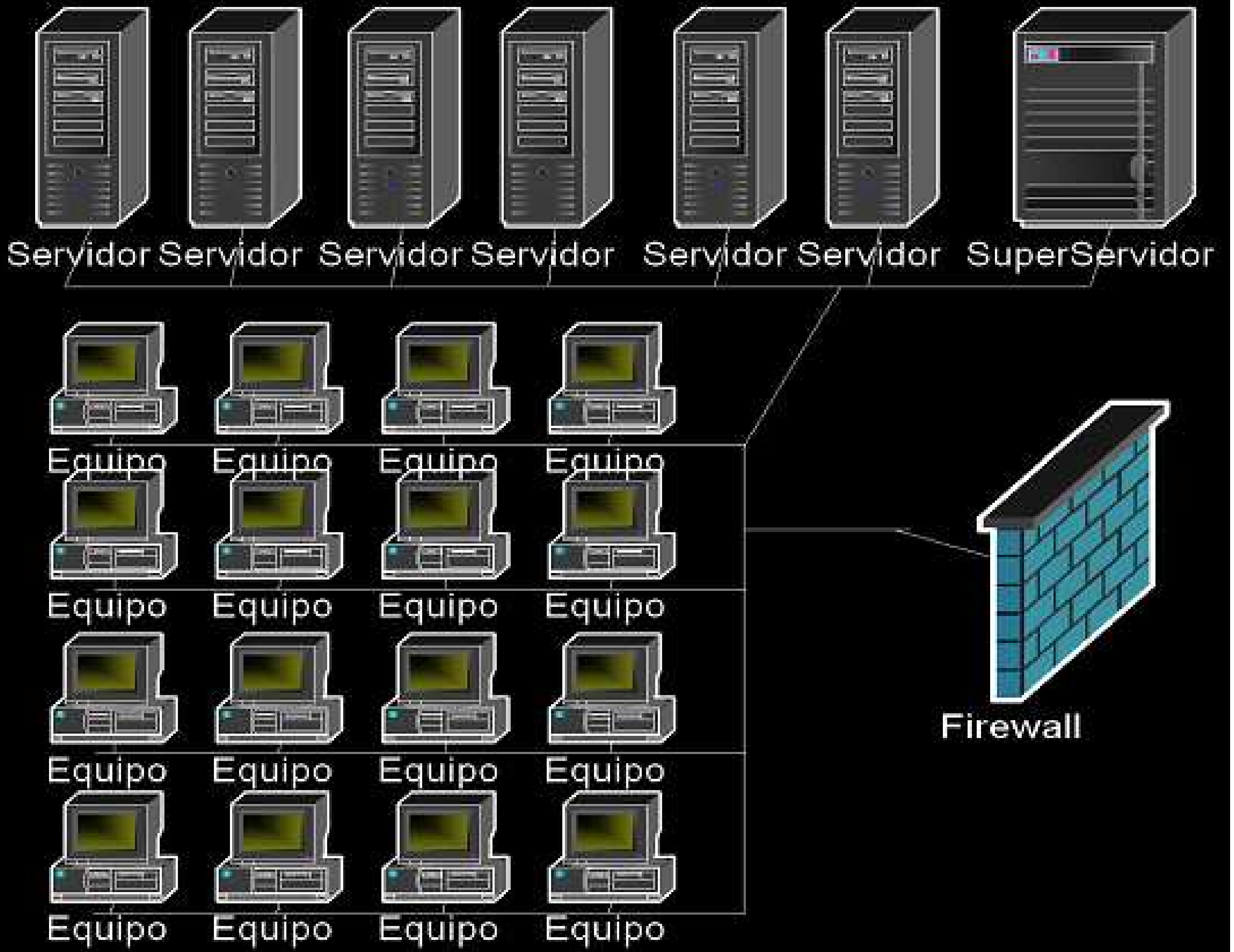
# Parte III

# Implementación

# Implementación

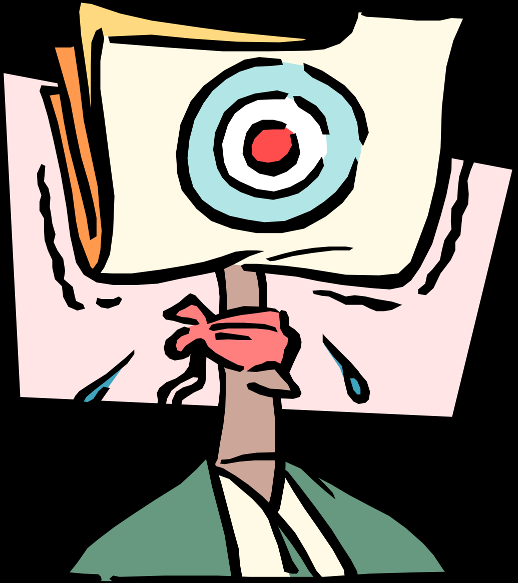
- De qué partimos
  - Buenos conocimientos de sistemas.
  - Sólidos conocimientos de TCP/IP.
  - Teoría de SSOO, redes, y algo de programación.

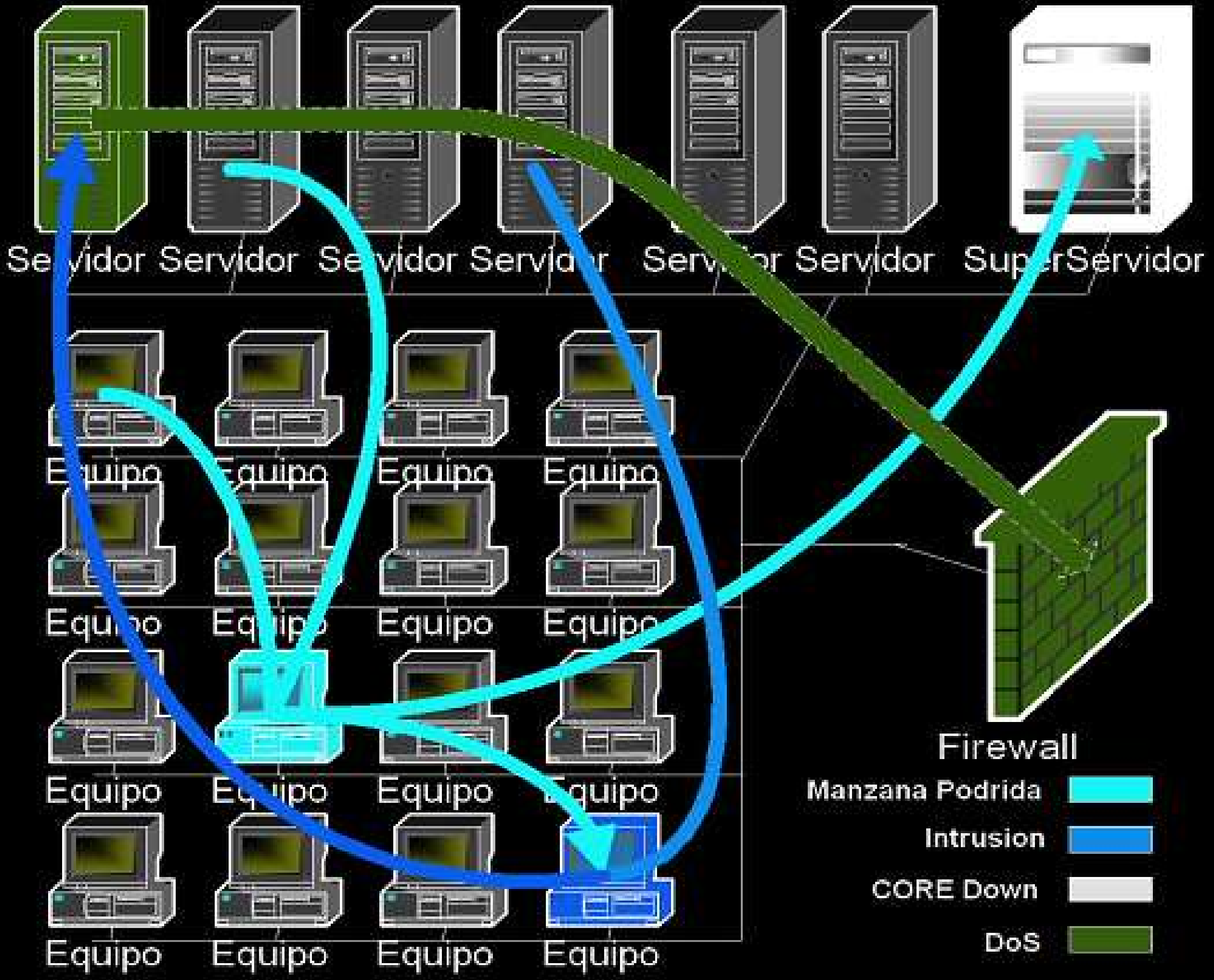
# Implementacion: Compartimentación de redes


Servidor
Servidor
Servidor
Servidor
Servidor
Servidor
SuperServidor
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Firewall

# Compartimentación de redes II.

## Manzana podrida



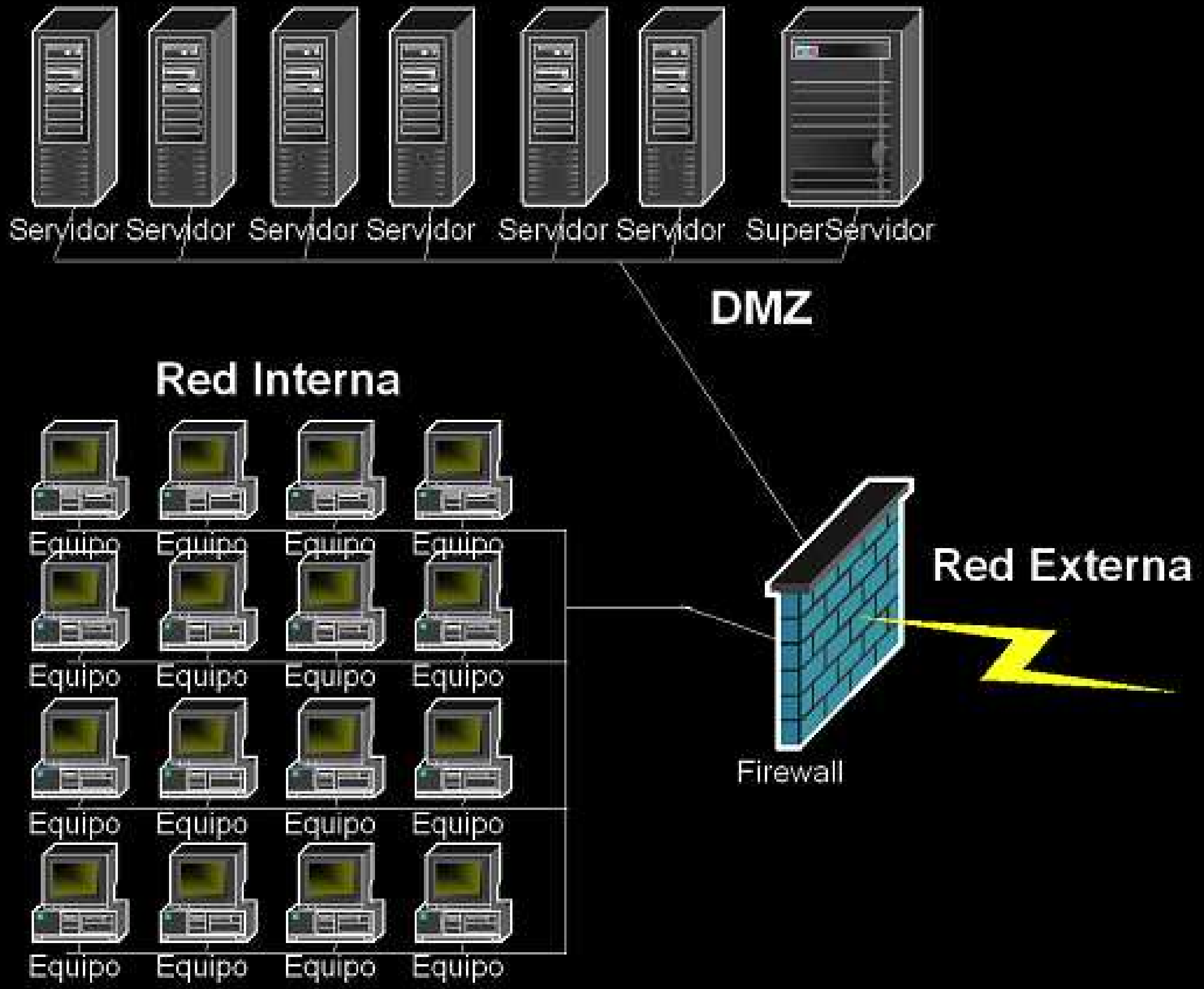

Servidor
Servidor
Servidor
Servidor
Servidor
Servidor
SuperServidor
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Firewall
Manzana Podrida
Intrusion
CORE Down
DoS

# Compartimentacion de redes III

- Diferenciar las redes logicamente por su funcion y su nivel de seguridad.
- Diferenciar las redes fisica y lógicamente.
- Modelos
  - Simple: DMZ
  - Complejo: FrontEnd y BackEnd

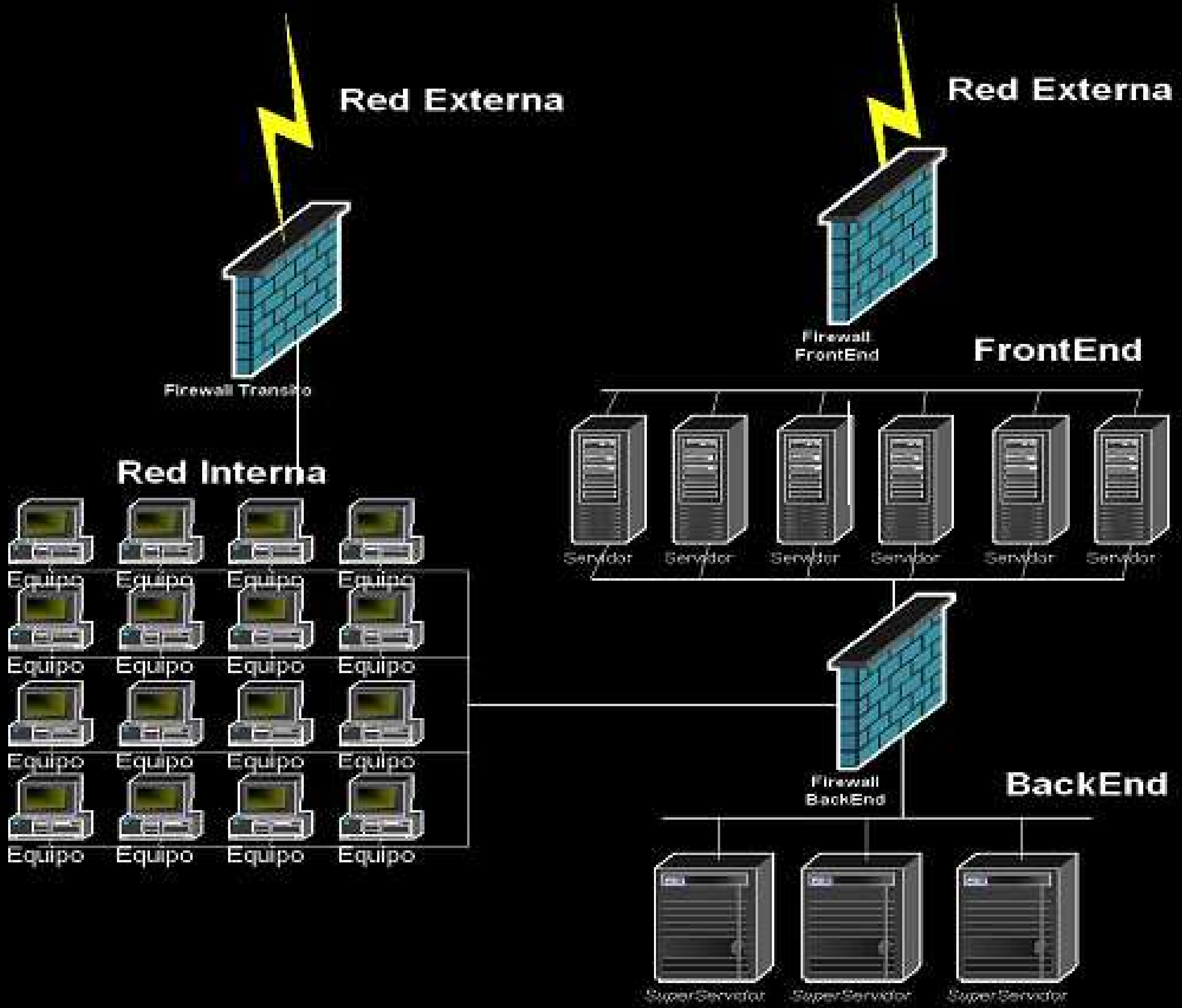
# Compartimentacion de redes IV

- Modelo clásico: DMZ


Servidor
Servidor
Servidor
Servidor
Servidor
Servidor
SuperServidor
DMZ
Red Interna
Equipo
Red Externa
Firewall

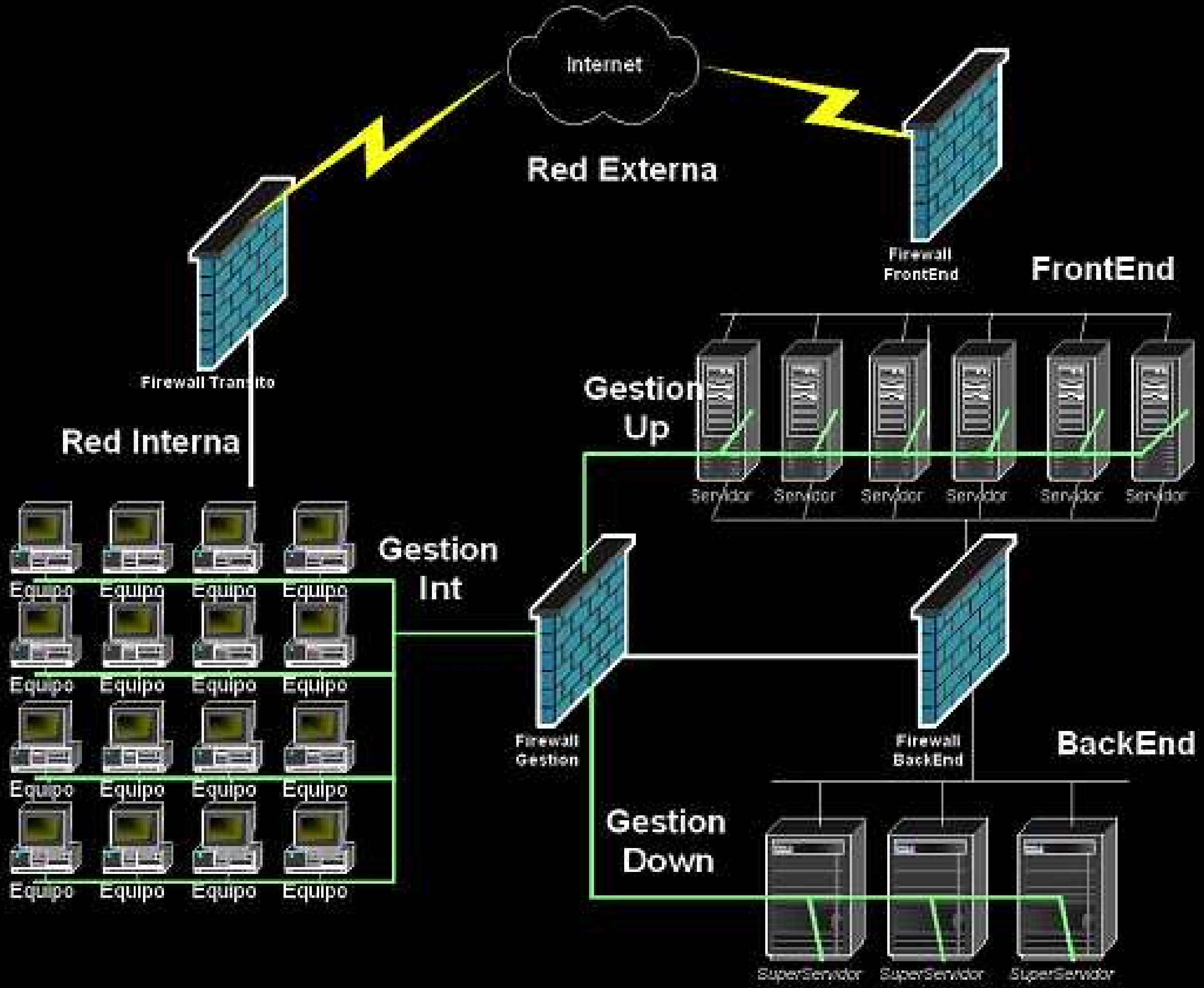
# Compartimentacion de redes V

- Modelo complejo. FE y BE (fase 1)


Red Externa
Firewall Transito
Red Interna
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Red Externa
Firewall FrontEnd
FrontEnd
Servidor
Servidor
Servidor
Servidor
Servidor
Servidor
Firewall BackEnd
BackEnd
SuperServidor
SuperServidor
SuperServidor

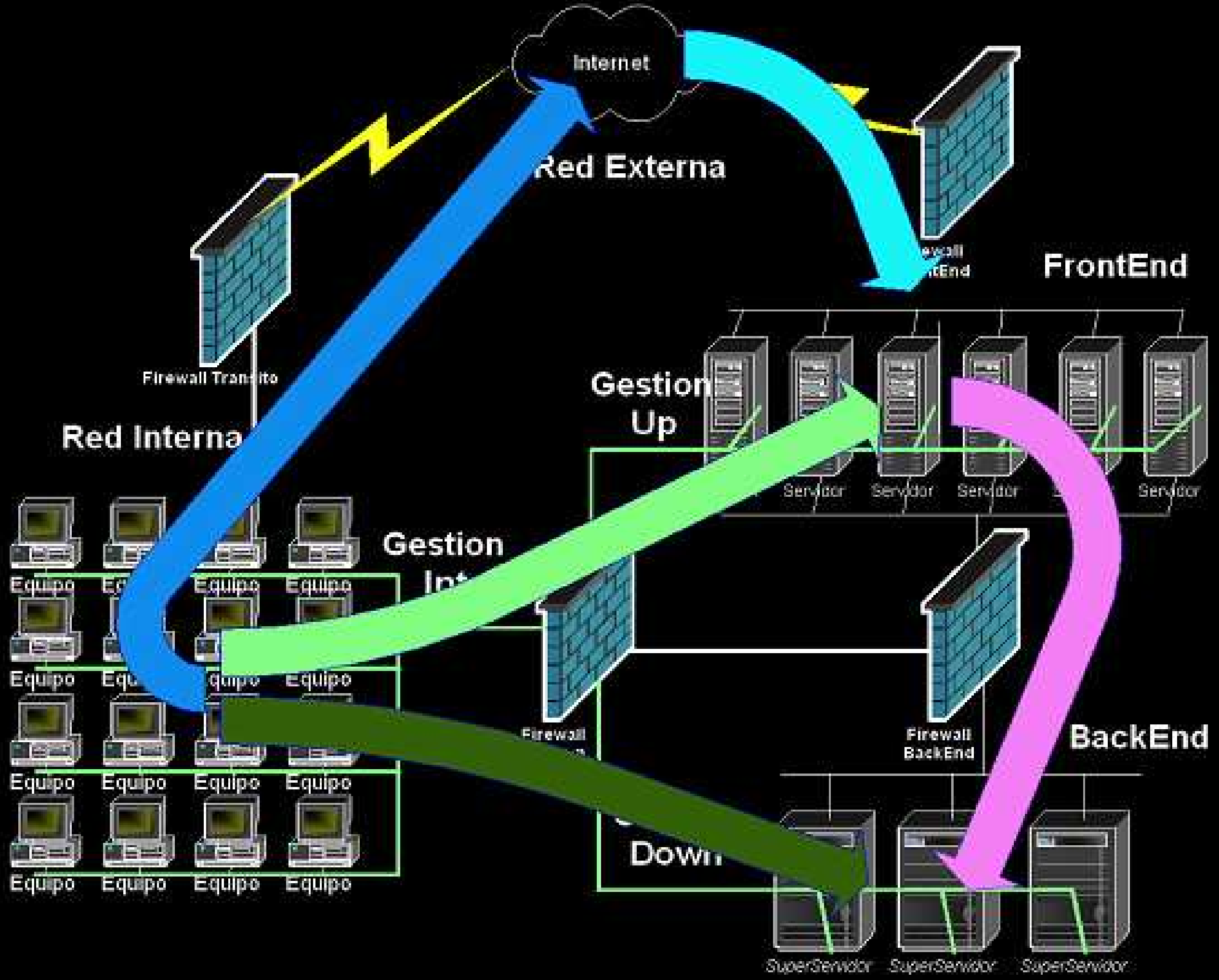
# Compartimentacion de redes VI

- Modelo complejo. FE y BE (fase 2)


Internet
Red Externa
Firewall FrontEnd
FrontEnd
Firewall Transito
Red Interna
Gestion Up
Servidor
Servidor
Servidor
Servidor
Servidor
Servidor
Gestion Int
Equipo
Firewall Gestion
Firewall BackEnd
BackEnd
Gestion Down
SuperServidor
SuperServidor
SuperServidor

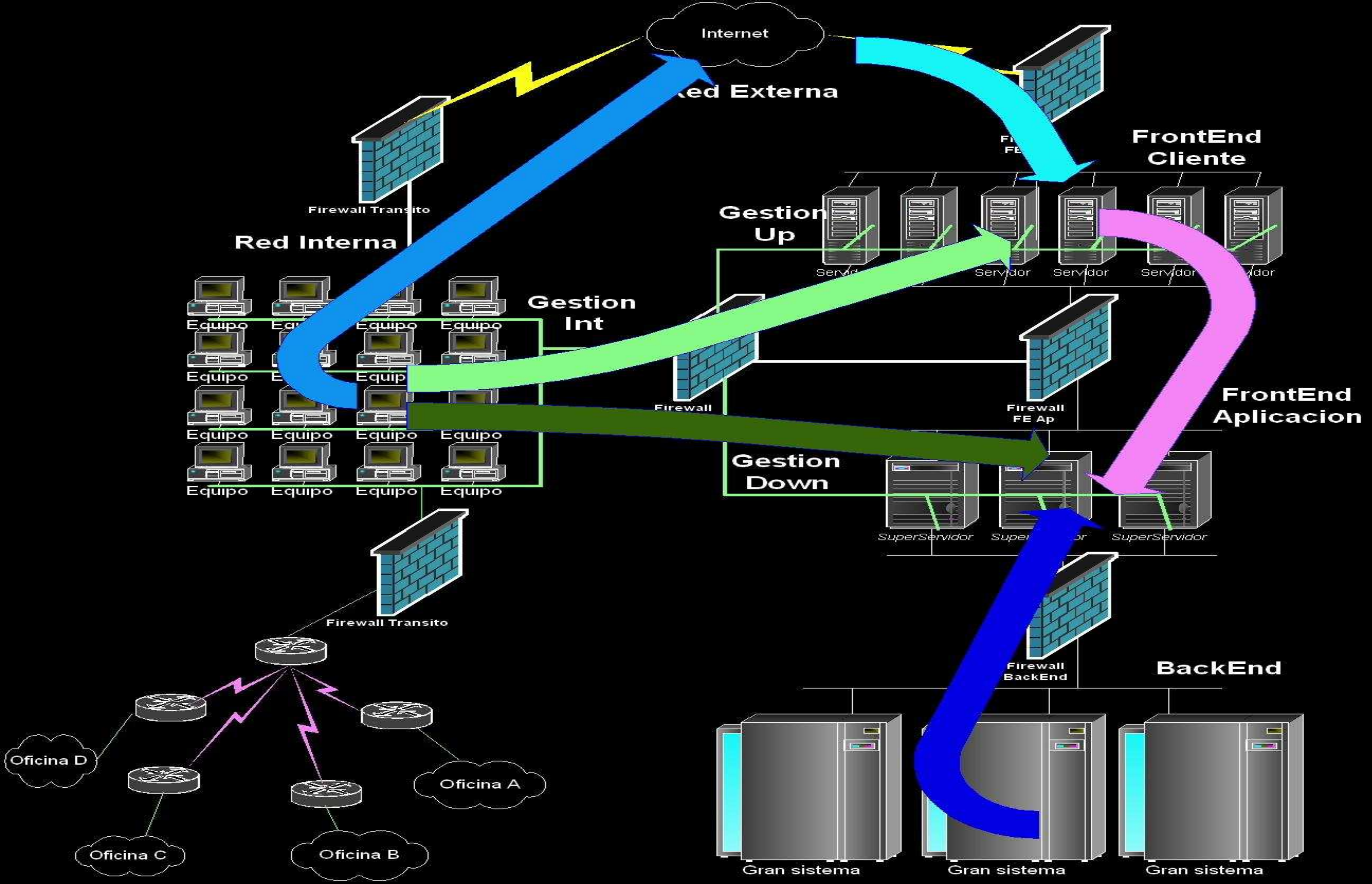
# Compartimentacion de redes VII

- Flujos en el modelo complejo


Internet
Red Externa
FrontEnd
Firewall Transito
Red Interna
Gestion Up
Servidor
Servidor
Servidor
Servidor
Gestion Int
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Equipo
Firewall
Firewall BackEnd
BackEnd
Down
SuperServidor
SuperServidor
SuperServidor

# Compartimentacion de redes VIII

Modelo complejo. FE y BE (fase 3)


Internet
ed Externa
Firewall Transito
Red Interna
Equipo
Gestion Int
Gestion Up
FrontEnd Cliente
Fi
FE
Servidor
Firewall
Firewall FE Ap
FrontEnd Aplicacion
Gestion Down
SuperServidor
Firewall BackEnd
BackEnd
Gran sistema
Firewall Transito
Oficina D
Oficina A
Oficina C
Oficina B

# Implementacion. Netfilter

- Qué es un firewall
- Qué tipos de firewall hay
- Netfilter en Linux
- HA con Netfilter
- Utilidades con Netfilter
- Configurando Netfilter

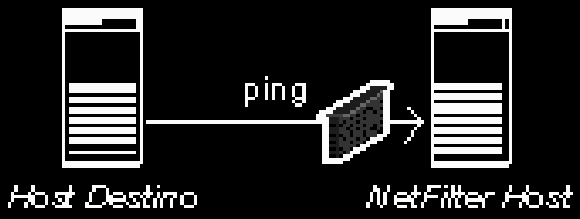
# Implementacion. Netfilter II

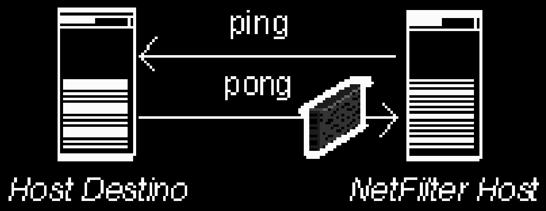
- Qué es un firewall
  - Necesidad de un firewall
  - Funcionamiento de un firewall

- Que tipos de firewall hay
  - Filtrado básico o tonto
  - Filtrado inteligente o de inspección de estados
  - Proxys y Firewalls de aplicación

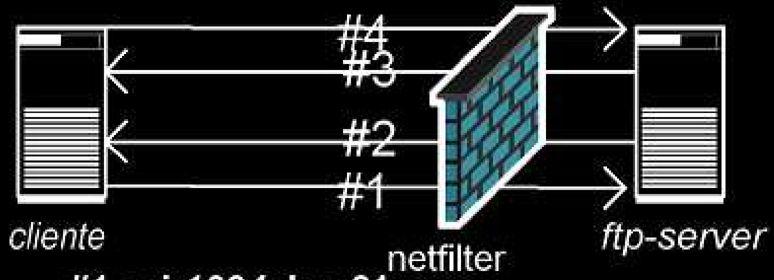
# Implementacion. Netfilter III

- Filtrado básico o tonto


ping
Host Destino
NetFilter Host


ping
pong
Host Destino
NetFilter Host

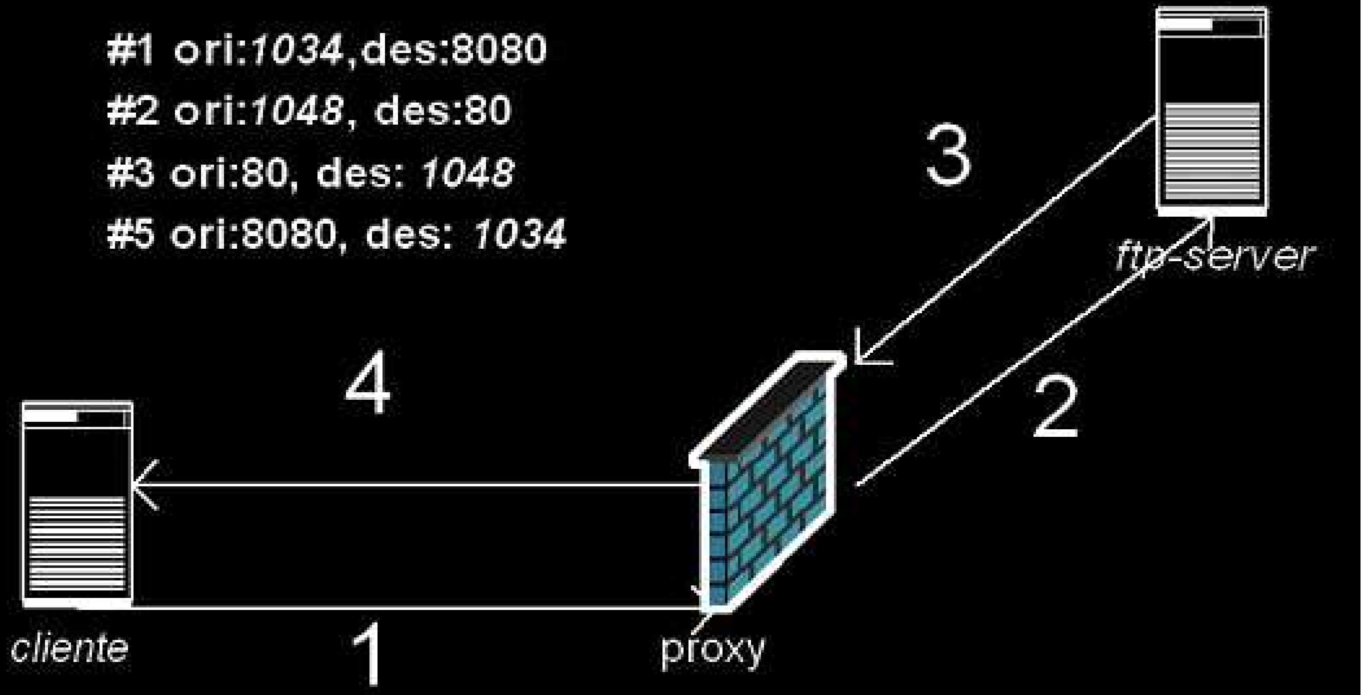
- Filtrado inteligente o de inspección de estados


#4
#3
#2
#1
cliente
netfilter
ftp-server

**#1 ori:*1034*,des:21**
**#2 ori:21, des:*1034***
**[ ... ]**
**#3 ori:*1048*, des:20**
**#2 ori:20, des:*1048***

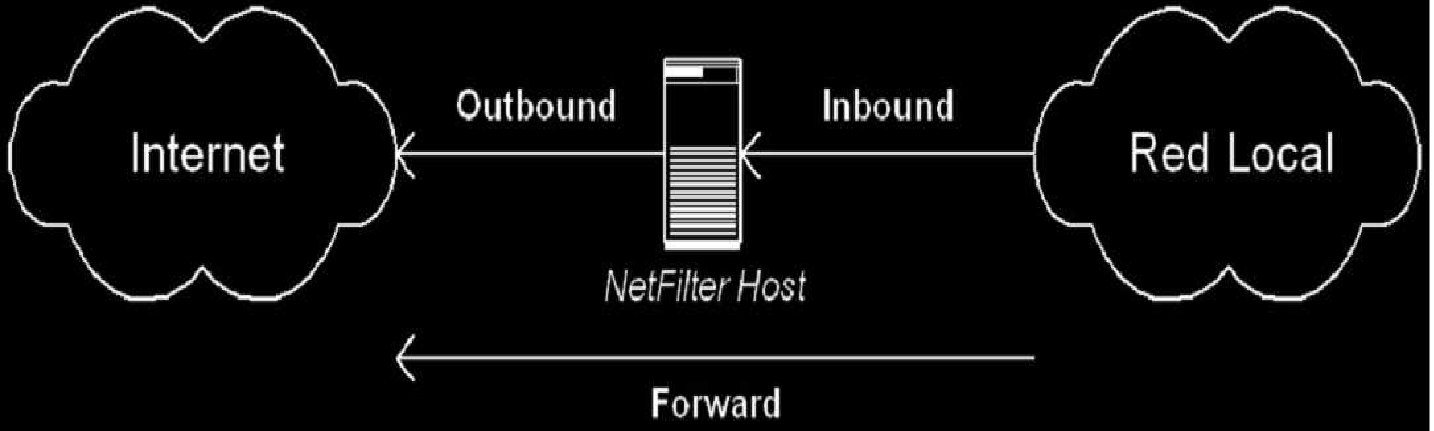
# Implementacion. Proxys

- Proxys y Firewalls de aplicación
  - Squid
  - Proxy Socks


#1 ori:1034,des:8080
#2 ori:1048, des:80
#3 ori:80, des: 1048
#5 ori:8080, des: 1034
3
ftp-server
4
2
cliente
1
proxy

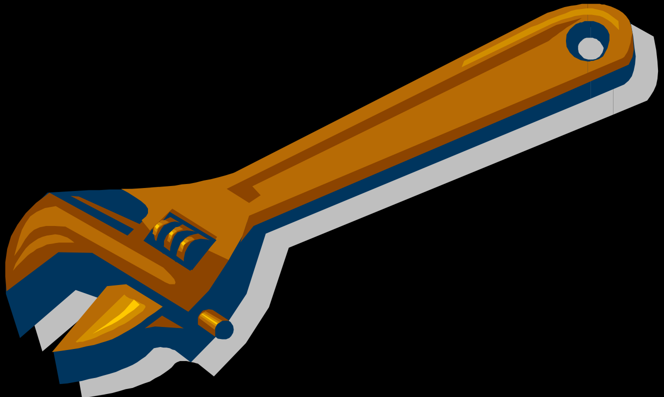
# Implementación. Cadenas

- Cadenas
  - Organizar flujos
  - Orden de ejecucion
  - Jerarquizacion


Internet
Outbound
Inbound
Red Local
NetFilter Host
Forward

# Implementación. Cadenas II

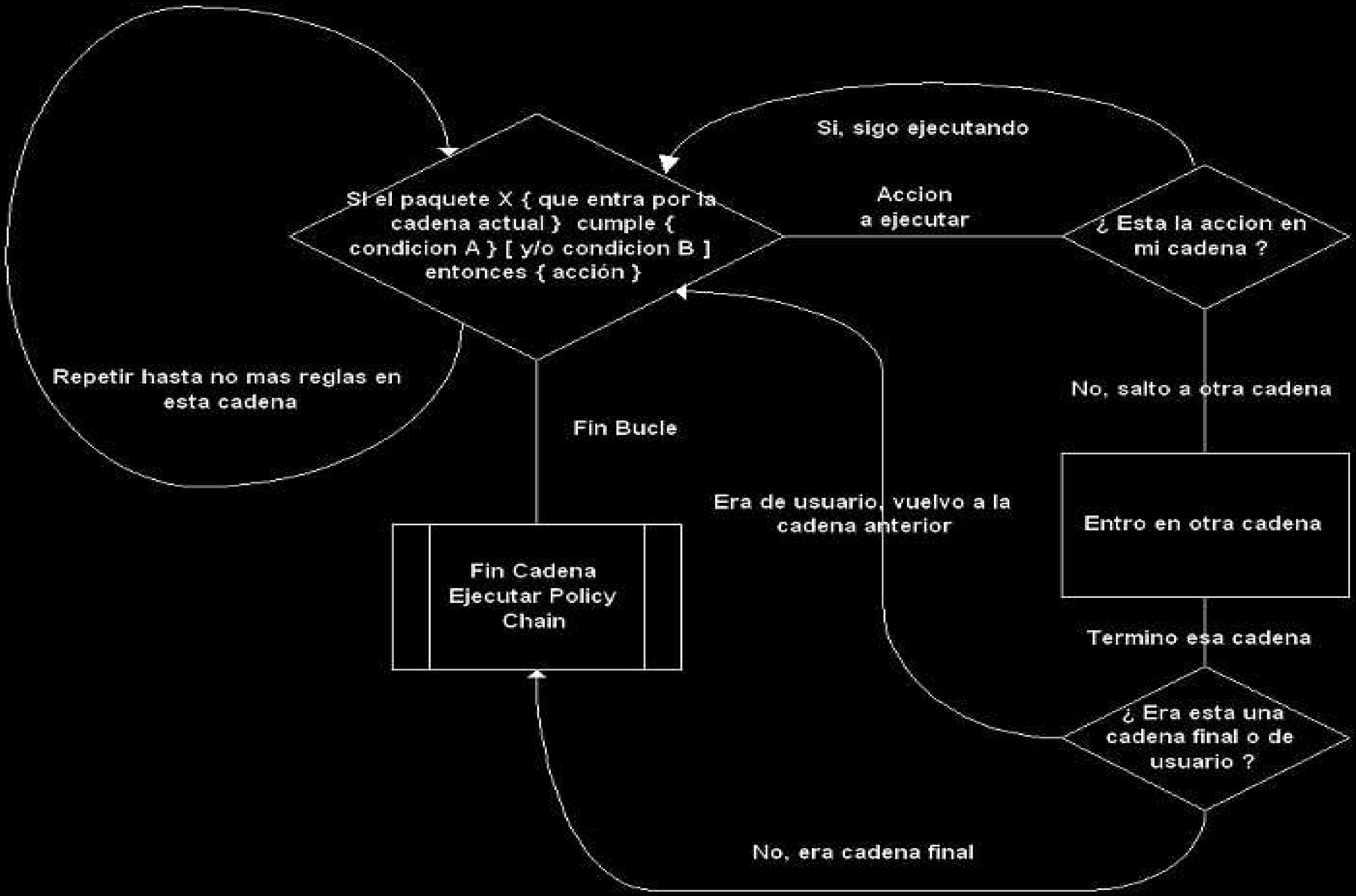
- Cadenas Destino/Acción
  - Accept
  - Drop
  - Reject
  - Log
  - Prerouting
  - PostRouting
  - Masquerade



# Implementación. Cadenas III

- Encadenando otras cadenas
- Cadenas de usuario
- Recursión

# Implementación. Cadenas IV



Si el paquete X { que entra por la cadena actual } cumple { condicion A } [ y/o condicion B ] entonces { acción }
Si, sigo ejecutando
Accion a ejecutar
¿ Esta la accion en mi cadena ?
Repetir hasta no mas reglas en esta cadena
Fin Bucle
No, salto a otra cadena
Entro en otra cadena
Era de usuario, vuelvo a la cadena anterior
Fin Cadena Ejecutar Policy Chain
Termino esa cadena
¿ Era esta una cadena final o de usuario ?
No, era cadena final

# Implementación. Cadenas V

**-N,** --new-chain <nombre_de_cadena>,  crearemos una cadena.

Sintaxis, #iptables

**-X** <nombre_de_cadena>, borramos una cadena.

**-I,** --insert <nombre_de_cadena>  { regla } , insertamos una regla al *principio* de una cadena.

**-A,** --append <nombre_de_cadena> { regla } ,insertamos una regla al *final* de una cadena.

**-D,** --delete <nombre_de_cadena> {# | regla } , borramos una regla de la cadena dada. Podemos especificar la posición ( 1, 2 ..) o la regla en formato convencional.

**-R,** --replace <nombre_de_cadena>  { # regla } ,reemplazamos una regla de la cadena especificada. Hay que especificar el numero de la cadena que queremos reemplazar y luego la regla a introducir.

**-L,** --list <nombre_de_cadena>, lista todas las reglas pertenecientes a la cadena dada.

**-F,** --flush , elimina todas las reglas, es lo mismo que eliminar todas las reglas una por una.

# Implementación. Iptables

- Sintaxis general
  - Opciones (limit burst)
  - Modularidad

```
Iptables -I|-A <cadena> -I <ifaz> -p <tcp|udp>
         --sport|--dport [!] <puerto>|<rango-rango>
         -d [!] <ip> -s [!] <ip> -m <modulo>
         --from <IP_SNAT> --to <IP_DNAT>
         -j <destino>
```

# Implementación. Filtrado I

- Filtrado
  - Por direcciones
  - Por puertos (TCP, UDP, ICMP)
  - Por interfaz
  - Por rafagas
  - Por bits de cabecera

# Implementación. Filtrado II

- ¿Dónde filtramos?
  - Input, o hacia el firewall
  - Output, o desde el firewall
  - Forward, o lo que pasa por el firewall.

```
# iptables -I|-A INPUT|OUTPUT|FORWARD
```

# Implementación. Filtrado III

- Como filtrar, algunos ejemplos:

```
iptables -A FORWARD -d POLLUX -p tcp --dport 22   -j ACCEPT # SSH
iptables -A FORWARD -d POLLUX -p tcp --dport 21   -j ACCEPT # FTP
iptables -A FORWARD -s POLLUX -i eth1 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -P FORWARD DROP

iptables -A INPUT -s PRIVATE -p tcp --dport 22    -j ACCEPT # SSH
iptables -A OUTPUT -d PRIVATE -p tcp --sport 22 -J ACCEPT # SSH out
iptables -P INPUT DROP
iptables -P OUTPUT DROP
```

# Implementacion. NAT

- Network Address Translation
  - Necesidad
    - Escasez IP’s
    - Ocultación direccionamiento
    - Port Multiplexing
    - Flexibilidad

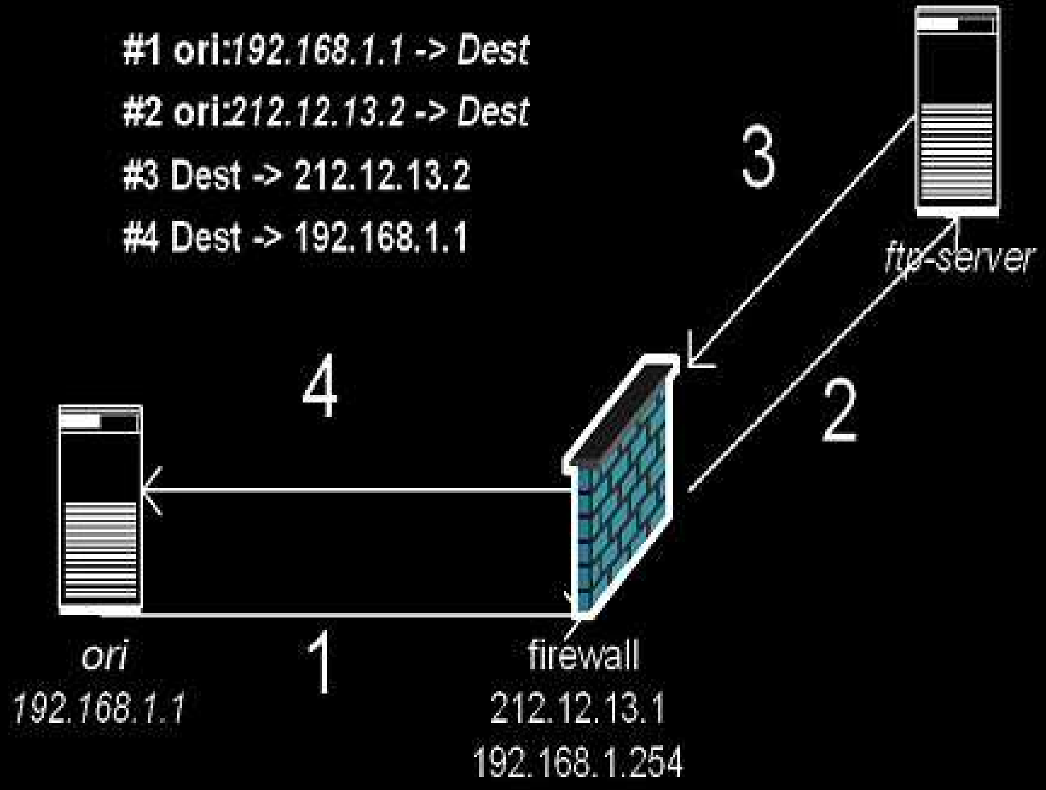
  - Tipos
    - NAT de IP
    - NAT de Puertos (PAT)

# Implementacion. NAT II

- Network Address Translation
  - Tipos de NAT.
    - Descripcion sobre variedades
      - Port Forwarding, Dynamic Nat, Smart NAT, Intelligent NAT, Dinamic NAT, Hide NAT, Static NAT, Masquerading y otras disparidades.
    - Source NAT
    - Dynamic NAT/Masquerading
    - Destination NAT

# Implementacion. SNAT

- Source NAT



#1 ori:*192.168.1.1 -> Dest*

#2 ori:*212.12.13.2 -> Dest*

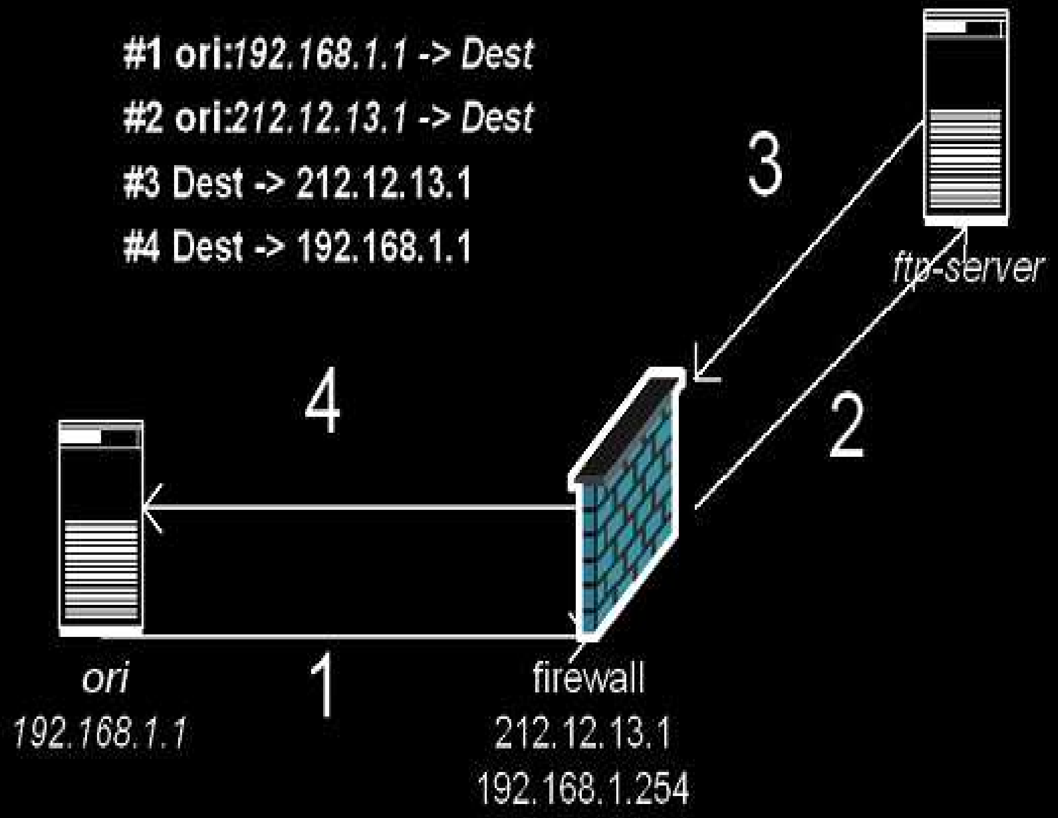
#3 Dest -> 212.12.13.2

#4 Dest -> 192.168.1.1

3
ftp-server
4
2
1
*ori*
*192.168.1.1*
firewall
212.12.13.1
192.168.1.254

# Implementacion. SNAT II

- Masquerading, Dynamic SNAT

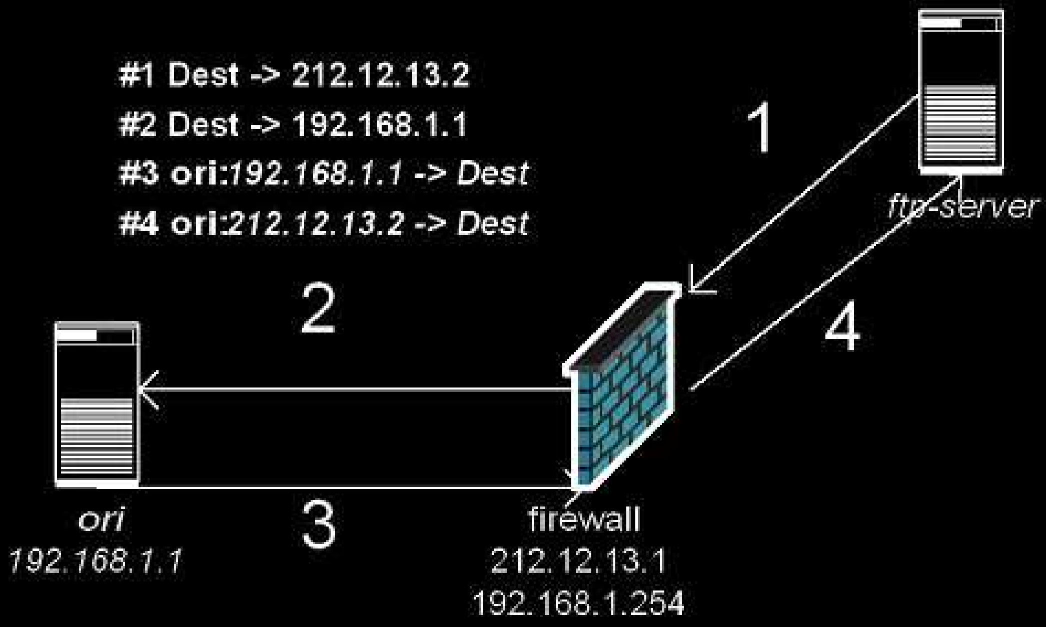

#1 ori:192.168.1.1 -> Dest
#2 ori:212.12.13.1 -> Dest
#3 Dest -> 212.12.13.1
#4 Dest -> 192.168.1.1
3
ftp-server
4
2
1
ori
192.168.1.1
firewall
212.12.13.1
192.168.1.254

# Implementacion. DNAT

- Destination NAT (DNAT)
  - Necesidad e importancia
  - Funcionamiento básico
  - Formas de implementarlo
  - Algoritmo de entrega ethernet/ip
  - Routing
  - Proxy ARP

# Implementacion. DNAT II

- DNAT. La idea
  - ¿De donde sacamos esa IP?
  - IP Virtual en Firewall



#1 Dest -> 212.12.13.2

#2 Dest -> 192.168.1.1

#3 ori:*192.168.1.1 -> Dest*

#4 ori:*212.12.13.2 -> Dest*

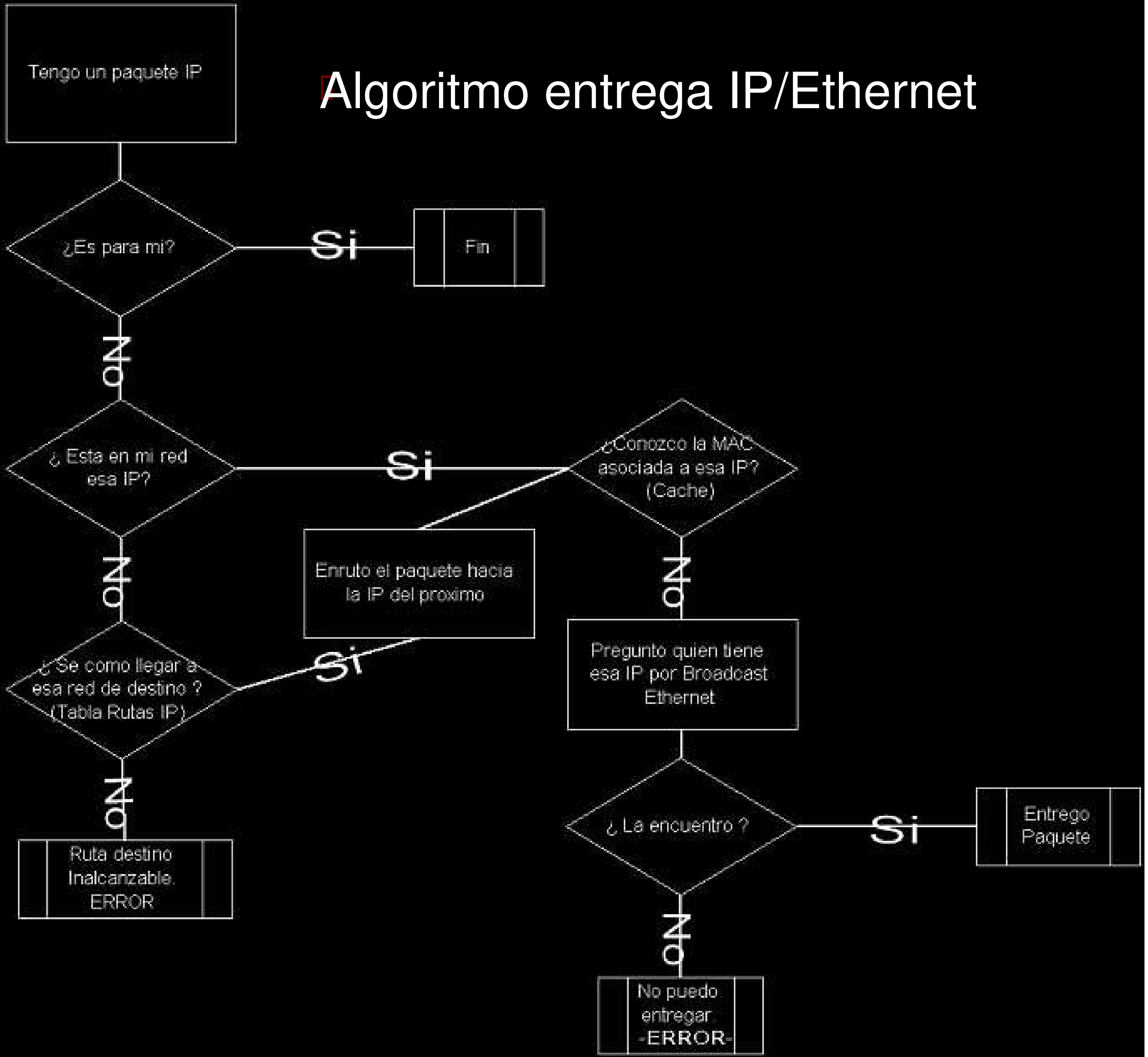
1
2
3
4
ftp-server
ori
192.168.1.1
firewall
212.12.13.1
192.168.1.254

# Implementacion. DNAT III

- Algoritmo de entrega Ethernet/IP
  - ¿Esta la IP en mi ruta?
  - Entrega ETHERNET
    - Capa L2
  - Broadcast IP
  - Broacast Ethernet
    - Ejemplos broadcast otros protocolos (ATM)

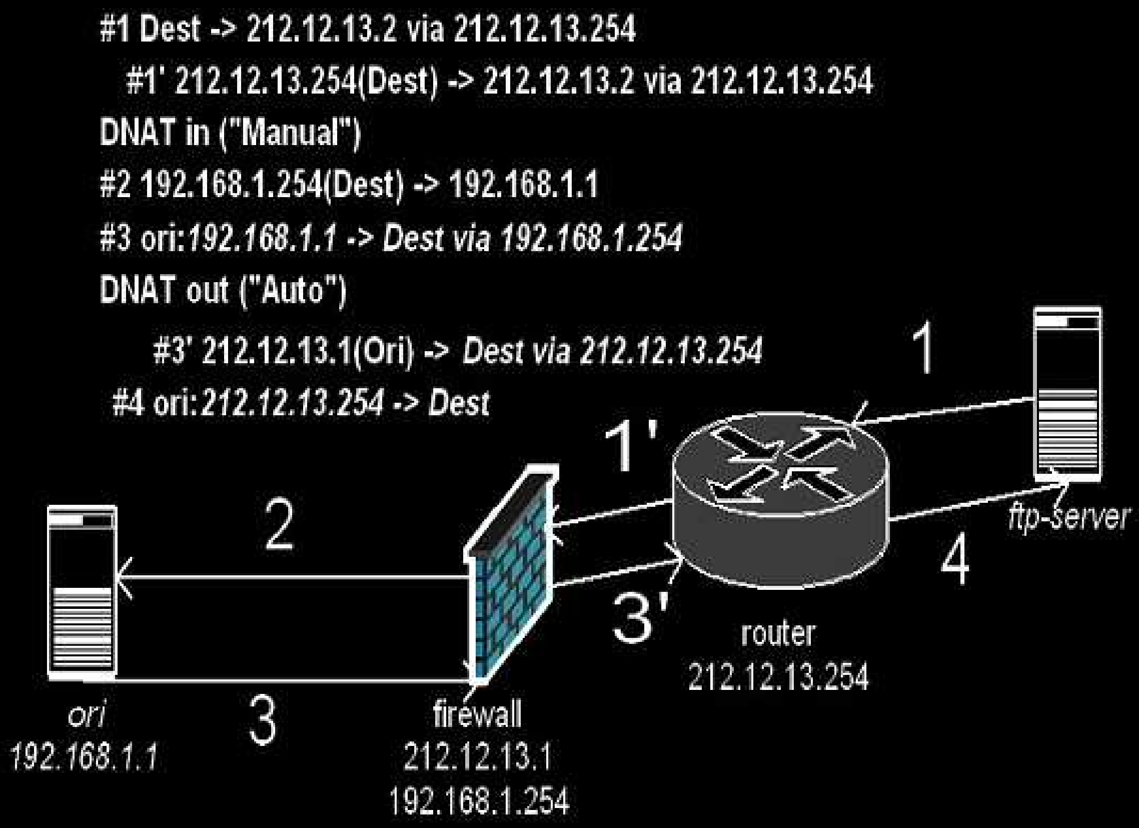
# Implementacion. DNAT IV

- Algoritmo entrega IP/Ethernet


Tengo un paquete IP
¿Es para mi?
Si
Fin
No
¿ Esta en mi red esa IP?
Si
¿Conozco la MAC asociada a esa IP? (Cache)
No
Enruto el paquete hacia la IP del proximo
No
¿ Se como llegar a esa red de destino ? (Tabla Rutas IP)
Si
Pregunto quien tiene esa IP por Broadcast Ethernet
No
Ruta destino Inalcanzable. ERROR
¿ La encuentro ?
Si
Entrego Paquete
No
No puedo entregar -ERROR-

# Implementacion. DNAT V

- DNAT con Enrutamiento L3



#1 Dest -> 212.12.13.2 via 212.12.13.254

#1' 212.12.13.254(Dest) -> 212.12.13.2 via 212.12.13.254

DNAT in ("Manual")

#2 192.168.1.254(Dest) -> 192.168.1.1

#3 ori:*192.168.1.1 -> Dest via 192.168.1.254*

DNAT out ("Auto")

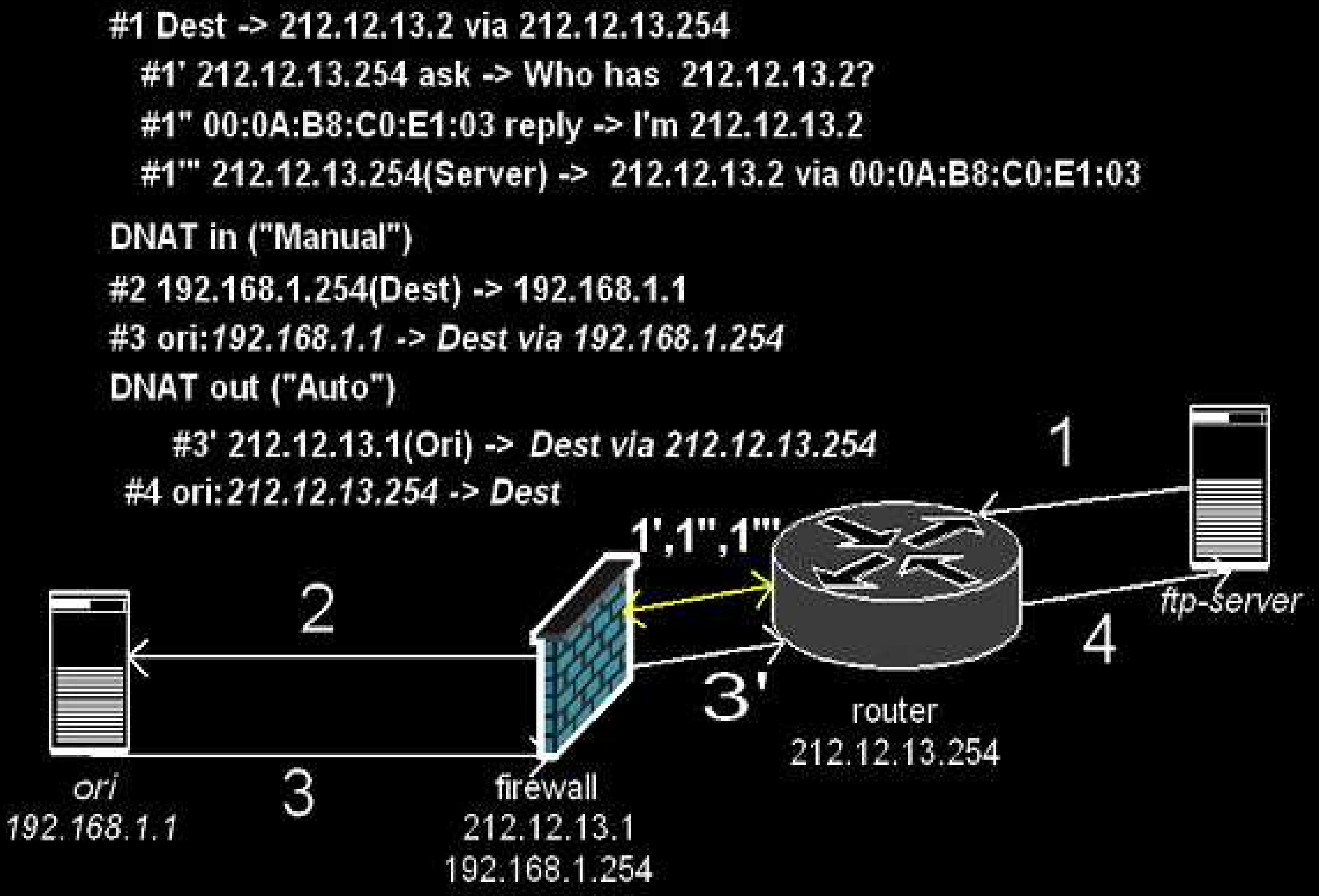
#3' 212.12.13.1(Ori) -> *Dest via 212.12.13.254*

#4 ori:*212.12.13.254 -> Dest*

1
1'
2
3
3'
4
ftp-server
router
212.12.13.254
firewall
212.12.13.1
192.168.1.254
ori
192.168.1.1

# Implementacion. DNAT VI

- DNAT con Enrutamiento L2. Proxy ARP



#1 Dest -> 212.12.13.2 via 212.12.13.254

#1' 212.12.13.254 ask -> Who has 212.12.13.2?

#1" 00:0A:B8:C0:E1:03 reply -> I'm 212.12.13.2

#1''' 212.12.13.254(Server) -> 212.12.13.2 via 00:0A:B8:C0:E1:03

DNAT in ("Manual")

#2 192.168.1.254(Dest) -> 192.168.1.1

#3 ori:*192.168.1.1 -> Dest via 192.168.1.254*

DNAT out ("Auto")

#3' 212.12.13.1(Ori) -> *Dest via 212.12.13.254*

#4 ori:*212.12.13.254 -> Dest*

1
1',1",1'''
2
3'
4
3
*ftp-server*
router
212.12.13.254
*ori*
*192.168.1.1*
firewall
212.12.13.1
192.168.1.254

# Implementacion. DNAT VII

- DNAT con Enrutamiento L3, implementacion con linux.
  - Hay que hacerlo en el punto intermedio (router)
  - A veces no tenemos acceso

```
# route add 212.12.13.2 gw 212.12.13.1
```

# Implementacion. DNAT VIII

- DNAT con Enrutamiento L2. Proxy ARP, implementacion con linux
  - Determinar la MAC externa

```
# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:04:4C:03:E4:CD
          inet addr:212.12.13.1    Bcast:217.126.145.192
eth1      Link encap:Ethernet  HWaddr 00:A0:C9:4C:F9:09
          inet addr:192.168.1.254  Bcast:192.168.1.255
```

  - Meter ruta estatica ARP

```
#arp -s 212.12.13.2 00:04:4C:03:E4:CD pub
```

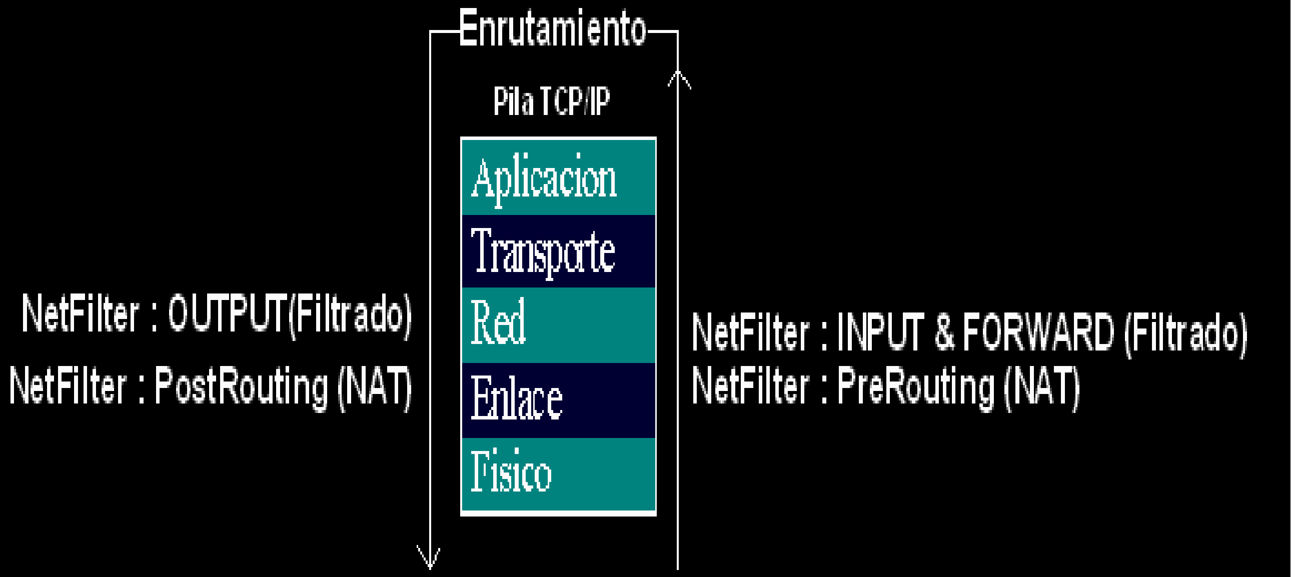
# Implementacion. NAT IX

## Reglas de NAT, ejemplos.

```
# DNAT de ip real (VIRTUAL) hacia Vulcano, con el puerto 21
iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p tcp --dport 21 --
    to $VULCANO    # FTP

# SNAT de Caligula con la ip VIRTUAL
iptables -t nat -A POSTROUTING -o eth0 -s $CALIGULA     -j SNAT --to
    $VIRTUAL

#HTTP Proxy transparente a traves de SQUID en VULCANO
iptables -t nat -A PREROUTING -d ! $LOCALNET -s $PRIVATE -p tcp --dport
    80 -j DNAT --to $VULCANO:8080
```

# Implementación. NAT X

- ¿Cuál es el orden de aplicación del filtrado y el NAT ?

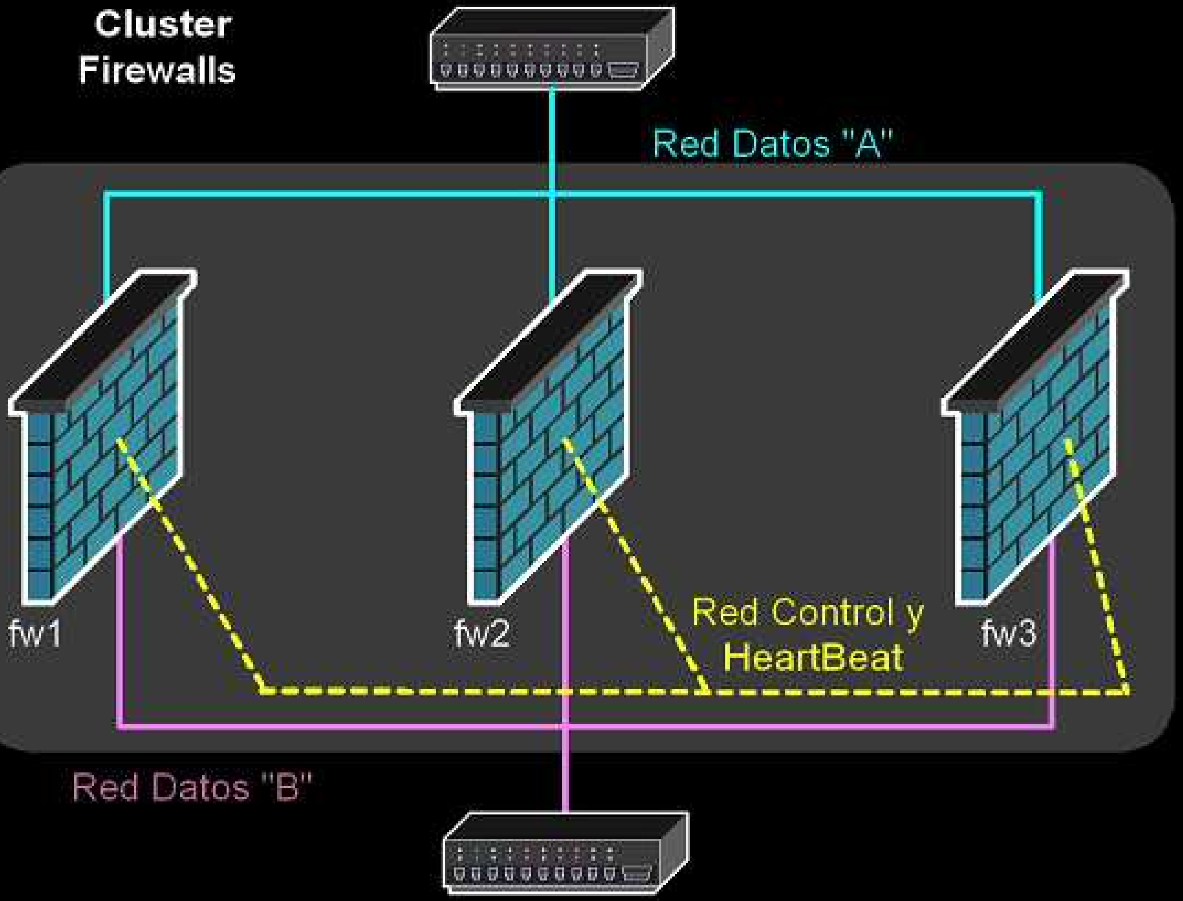

Enrutamiento
Pila TCP/IP
Aplicacion
Transporte
Red
Enlace
Fisico
NetFilter : OUTPUT(Filtrado)
NetFilter : PostRouting (NAT)
NetFilter : INPUT & FORWARD (Filtrado)
NetFilter : PreRouting (NAT)

# HA en Firewalls

- Conceptos de HA
- Tipos de HA
  - Clustering
  - HA Hot Standby (pasivo/activo)
  - HA Activo/Activo
  - HA A/A con Load Sharing.

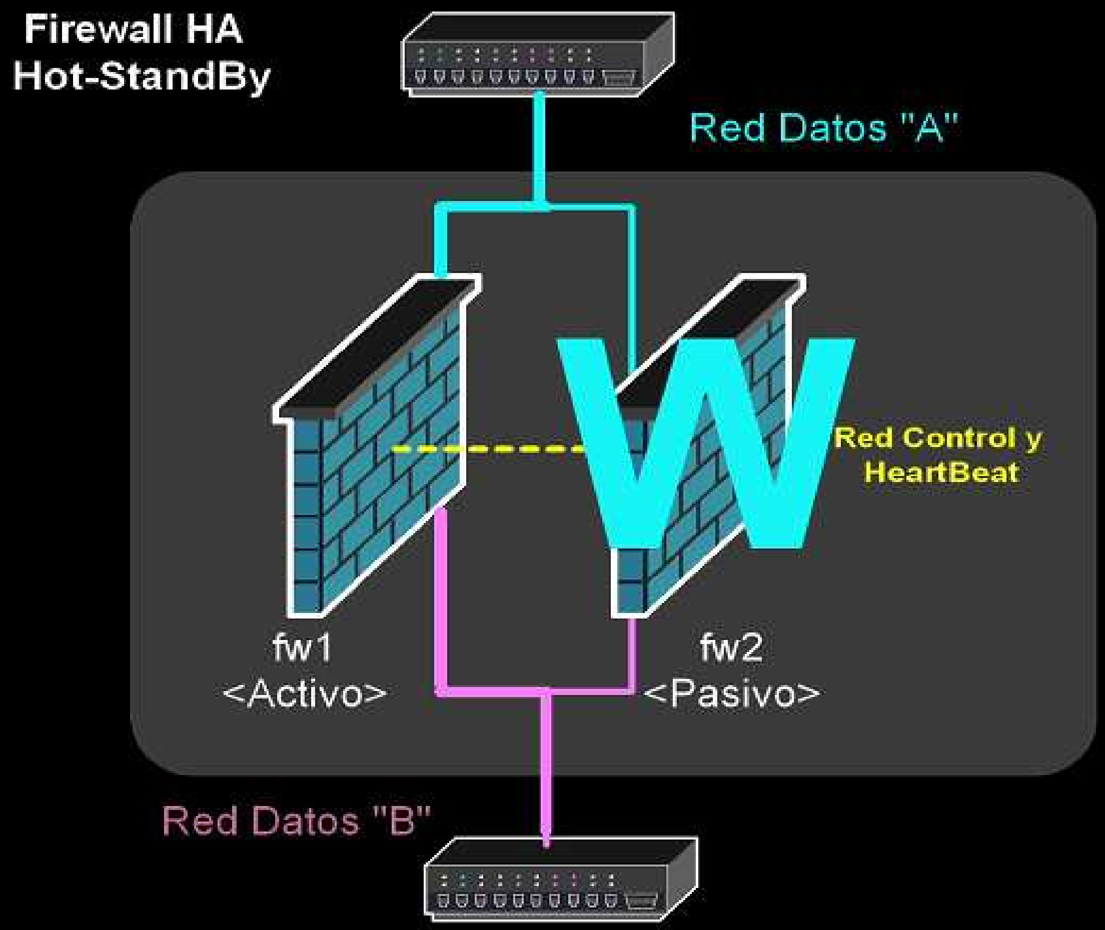
# HA en Firewalls I

- Clustering


Cluster
Firewalls
Red Datos "A"
fw1
fw2
fw3
Red Control y
HeartBeat
Red Datos "B"

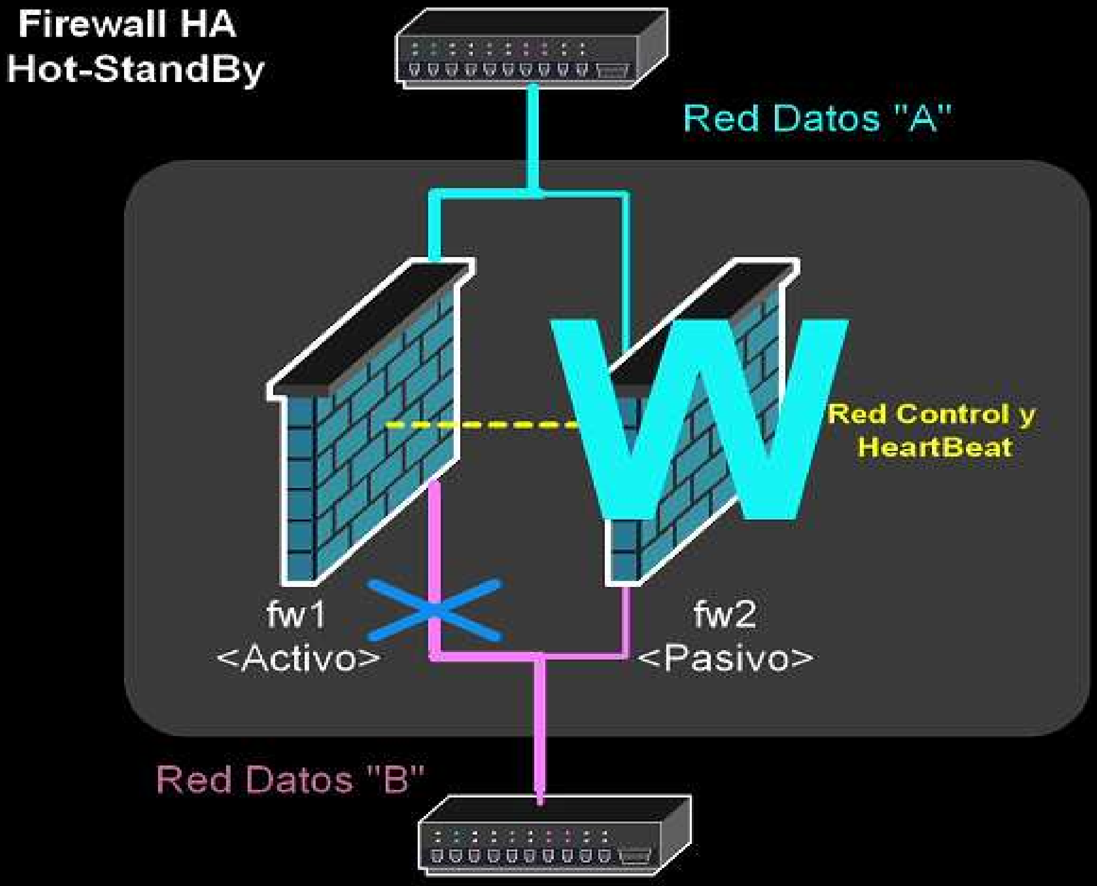
# HA en Firewalls II

- HA Hot StandBy Activo/Pasivo


Firewall HA
Hot-StandBy
Red Datos "A"
W
Red Control y
HeartBeat
fw1
<Activo>
fw2
<Pasivo>
Red Datos "B"

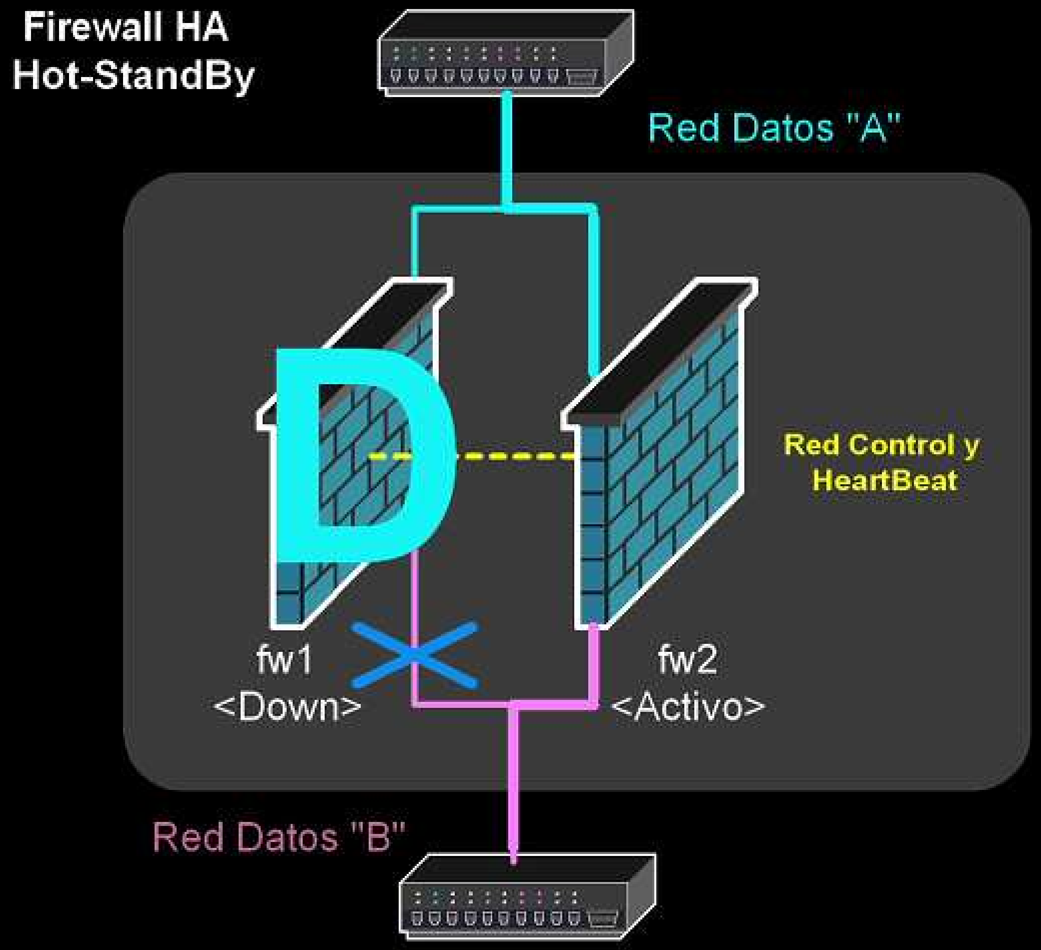
# HA en Firewalls III

- Hot StandBy, fallo


Firewall HA
Hot-StandBy
Red Datos "A"
Red Control y
HeartBeat
fw1
<Activo>
fw2
<Pasivo>
Red Datos "B"

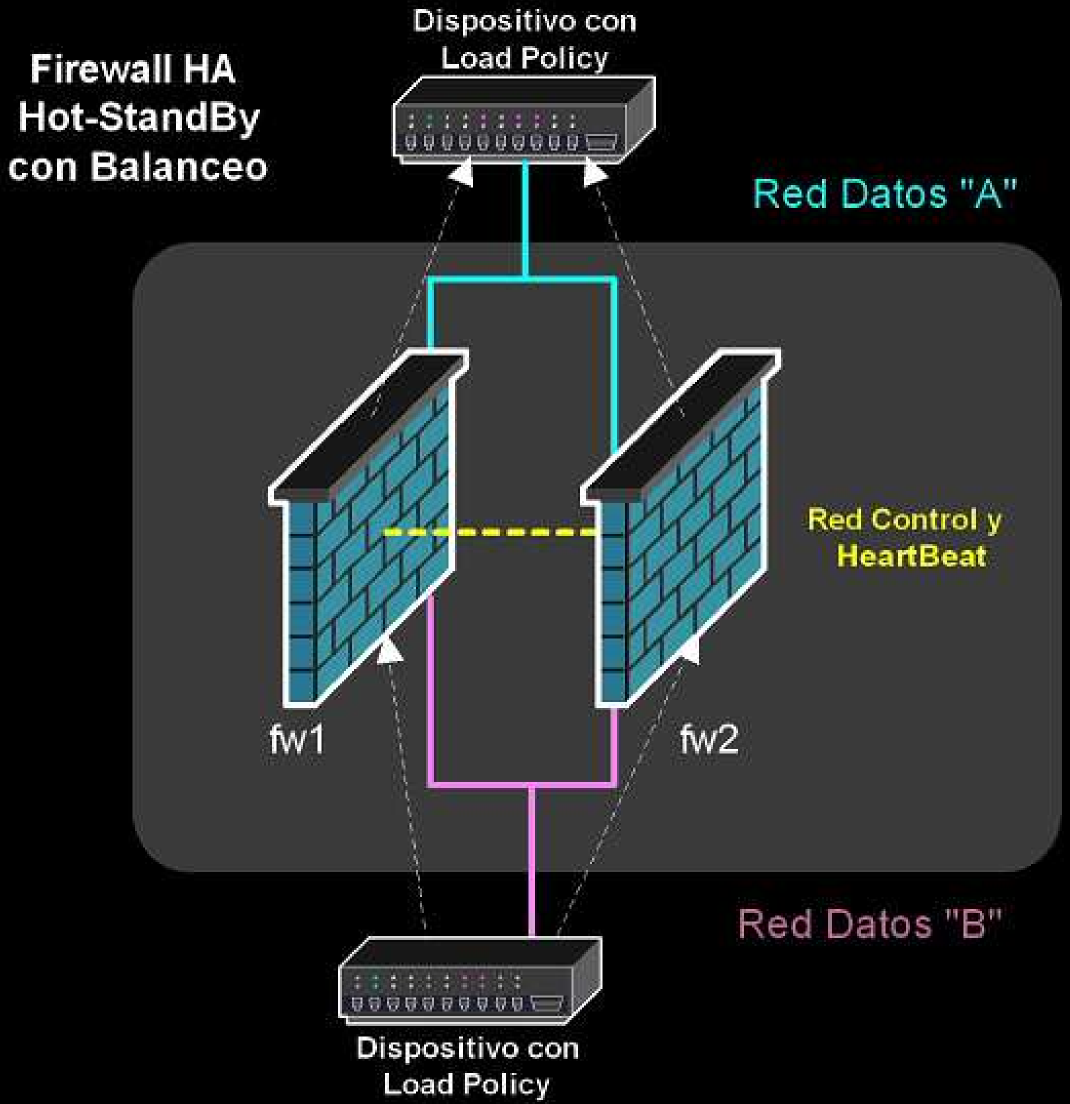
# HA en Firewalls IV

- Hot StandBy, switch over


Firewall HA
Hot-StandBy
Red Datos "A"
D
Red Control y
HeartBeat
fw1
<Down>
fw2
<Activo>
Red Datos "B"

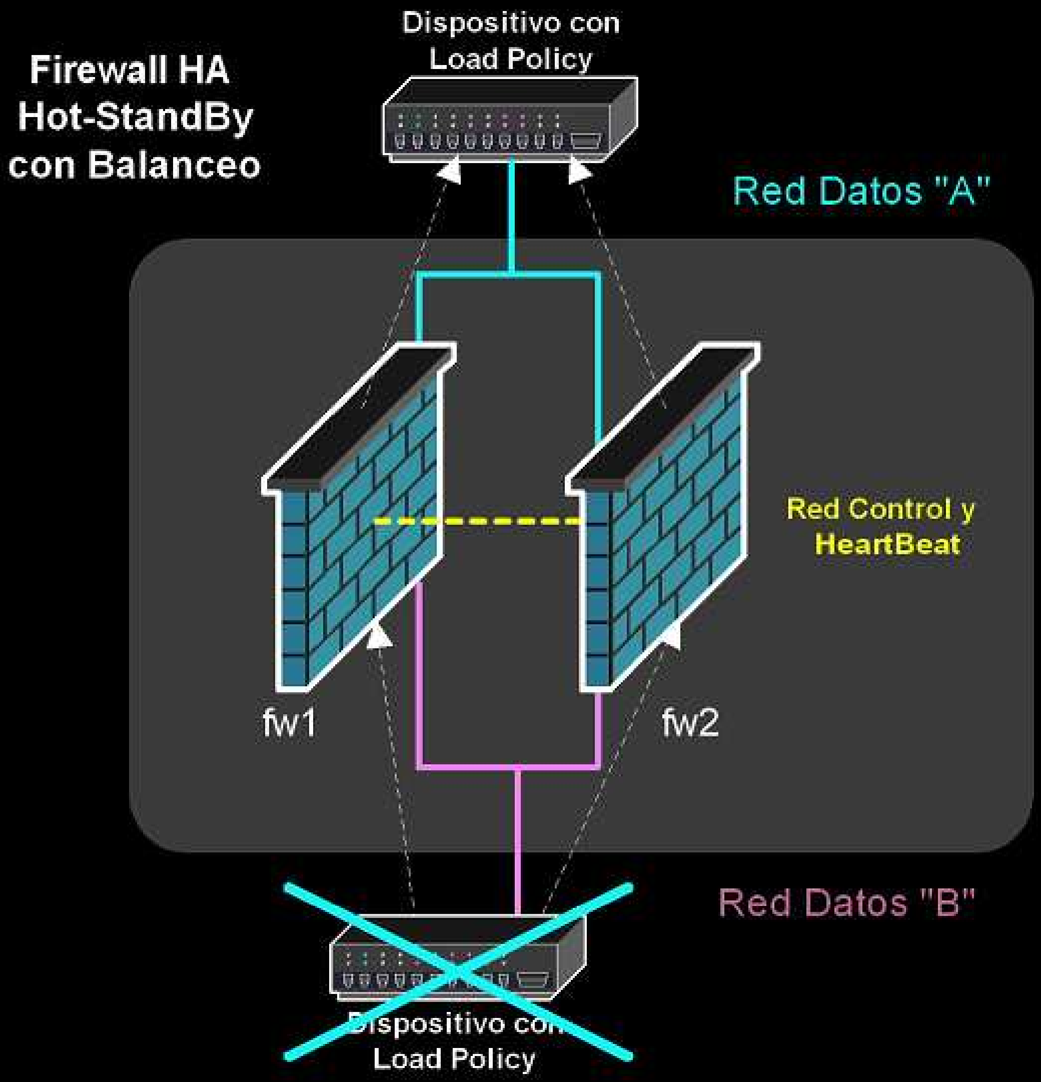
# HA en Firewalls V

## HA Activo/Activo: Load Sharing


Firewall HA
Hot-StandBy
con Balanceo
Dispositivo con
Load Policy
Red Datos "A"
Red Control y
HeartBeat
fw1
fw2
Red Datos "B"
Dispositivo con
Load Policy

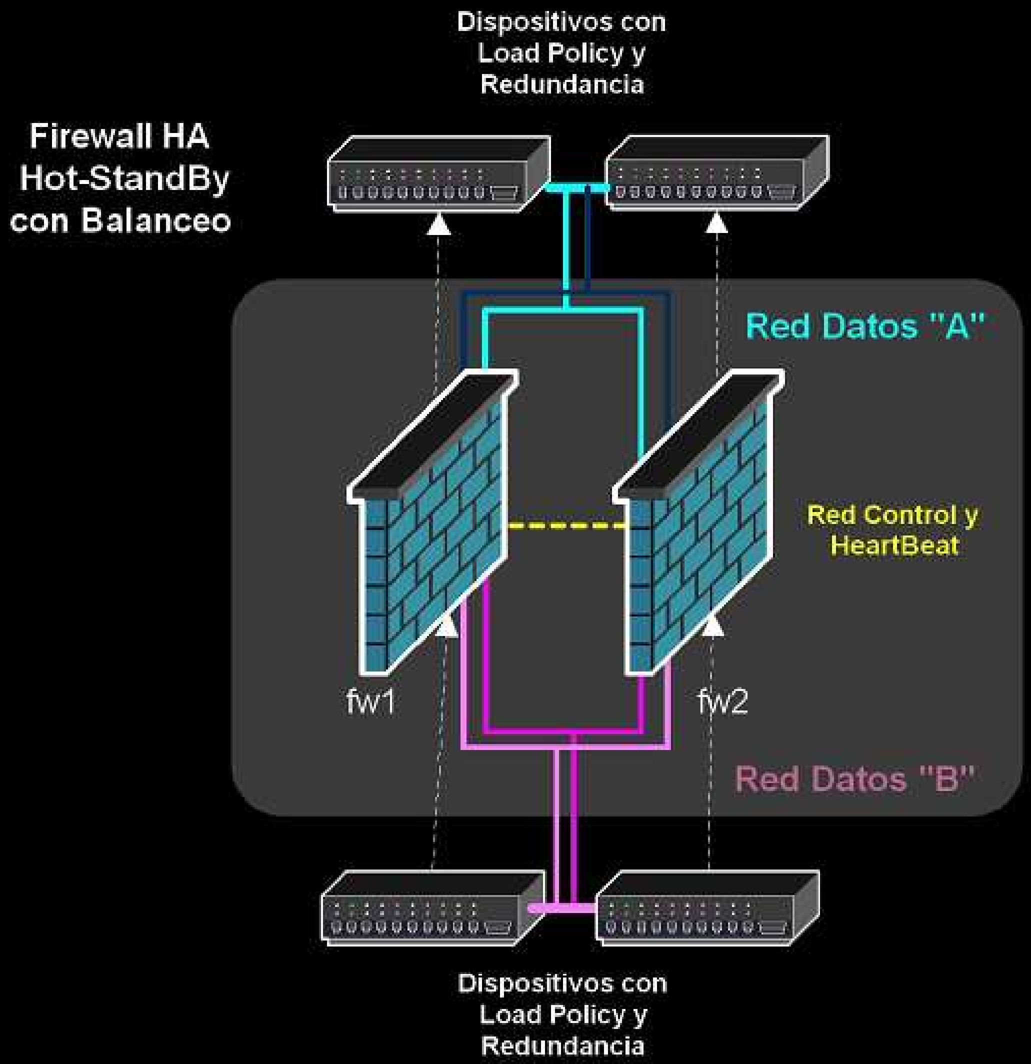
# HA en Firewalls VI

- ¿Es esto perfecto?.


Firewall HA
Hot-StandBy
con Balanceo
Dispositivo con
Load Policy
Red Datos "A"
Red Control y
HeartBeat
fw1
fw2
Red Datos "B"
Dispositivo con
Load Policy

# HA en Firewalls VII

## HA Integral y Load Sharing

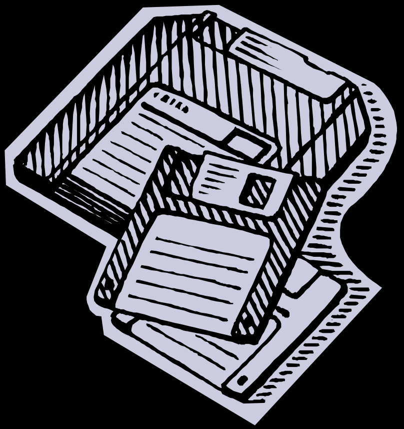


Dispositivos con Load Policy y Redundancia
Firewall HA Hot-StandBy con Balanceo
Red Datos "A"
Red Control y HeartBeat
fw1
fw2
Red Datos "B"
Dispositivos con Load Policy y Redundancia

# Implementando HA

- VRRP. Rfc 2338
  - Cisco HSRP
  - Autenticación

- Standard y fabricantes.
  - Firewalls
  - Electronica de red
  - SSOO



# Implementando VRRP I

- Arranque

```
#!/bin/bash
#
# VRRP Daemon Start, 01/03/02
# Sancho Lerena, slerena@gnusec.com

LISTA_PROCESOS=`ps -A | grep "vrrpd" | tr -s " " | cut -d " " -f 2`
if [ -n "$LISTA_PROCESOS" ]
then
                echo " There are present VRRP daemons running. Aborting."
else
                echo "Arrancando demonio VRRP en eth1/192.168.5/24"
                nohup vrrpd -v 105 -p 100 -i eth1 192.168.5.100 > /dev/null &
                echo "Arrancando demonio VRRP en eth0/192.168.6/24"
                nohup vrrpd -v 106 -p 100 -i eth0 192.168.6.100 > /dev/null &
                echo "Waiting for VRRP Daemons"
                sleep 10
                echo "Restoring IP routing"
                route add default gw 192.168.5.1; fi;
```

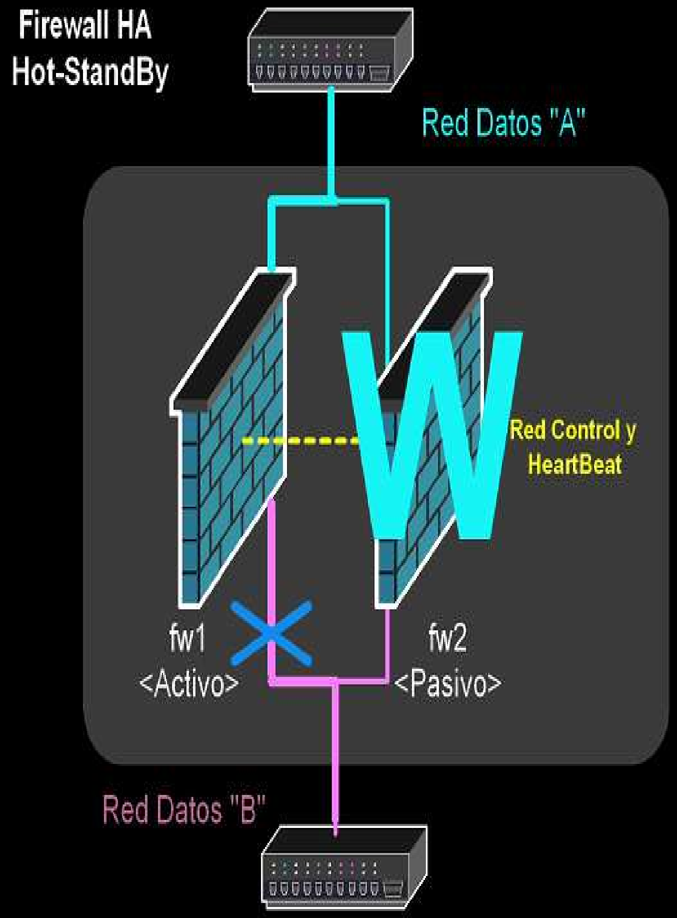
# Implementando VRRP II

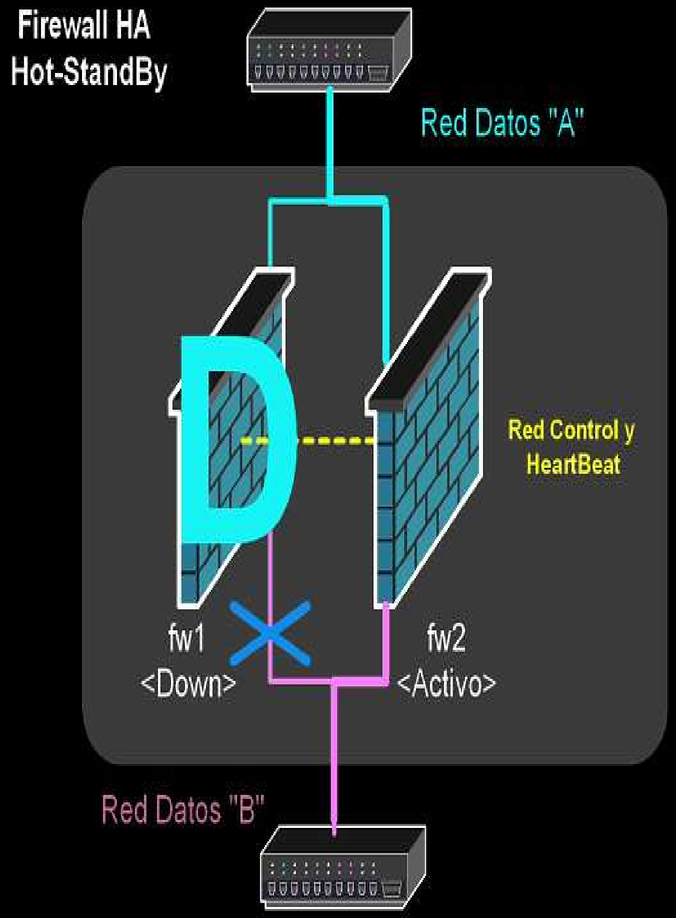
- Parada

```
#!/bin/bash
# VRRP Daemon Stop, 01/03/02
# Sancho Lerena, slerena@gnusec.com

if [ "$1" == "-?" ]
then
  echo "Syntax:"
  echo "        vrrp-stop [-f] "
  echo " "
  echo " -f : force kill with signal 9 "
  exit
fi;

# Obtener el PID de los procesos en memoria
LISTA_PROCESOS=`ps -A | grep "vrrpd" | tr -s " " | cut -d " " -f 2`
if [ -z "$LISTA_PROCESOS" ]
then
  echo " No VRRP Daemons present. Aborting."
else
  echo " Killing all VRRP daemons"
  if [ "$1" == "-f" ]
  then
     kill -s 9 $LISTA_PROCESOS
  else
     kill $LISTA_PROCESOS
  fi;
fi;
```

# Implementando VRRP III

- Monitorización.
  - Problema del agujero negro


Firewall HA
Hot-StandBy
Red Datos "A"
W
Red Control y
HeartBeat
fw1
<Activo>
fw2
<Pasivo>
Red Datos "B"


Firewall HA
Hot-StandBy
Red Datos "A"
D
Red Control y
HeartBeat
fw1
<Down>
fw2
<Activo>
Red Datos "B"

# Implementando VRRP IV

## Monitorización

```
#!/bin/bash
# Checking conectivity with ICMP Ping, VRRPD Companion Script
VER="11/03/2002 - v1.0"
SLEEP_TIME=$2             # Tiempo de parada entre checks, en segundos
if [ -z $2 ]
then
    SLEEP_TIME=5          # Si no se especifica, el check es cada 5 segundos
fi;
# Obtener el PID de los procesos de VRRPD en memoria
LISTA_PROCESOS=`ps -A | grep "vrrpd" | tr -s " " | cut -d " " -f 2`
if [ -z "$LISTA_PROCESOS" ]
then
    echo " No VRRP Daemon running, aborting. "
    exit
fi;
IP_DESTINO=$1             # IP de comprobacion, pasada como 1# parametro
COMANDO="`ping -c 1 "$IP_DESTINO" | grep '100% packet loss'`"
RES=0
while [ "$RES" -eq 0 ];do
    if [ ! -z "$COMANDO" ] ;then
        echo " Ping fail "
        echo " Shutting down VRRP daemons "
        kill -s 9 $LISTA_PROCESOS
        RES=1;else
        echo " Debug: Ping ok"
        sleep $SLEEP_TIME
    fi;done;
```

# Detectores de intrusión

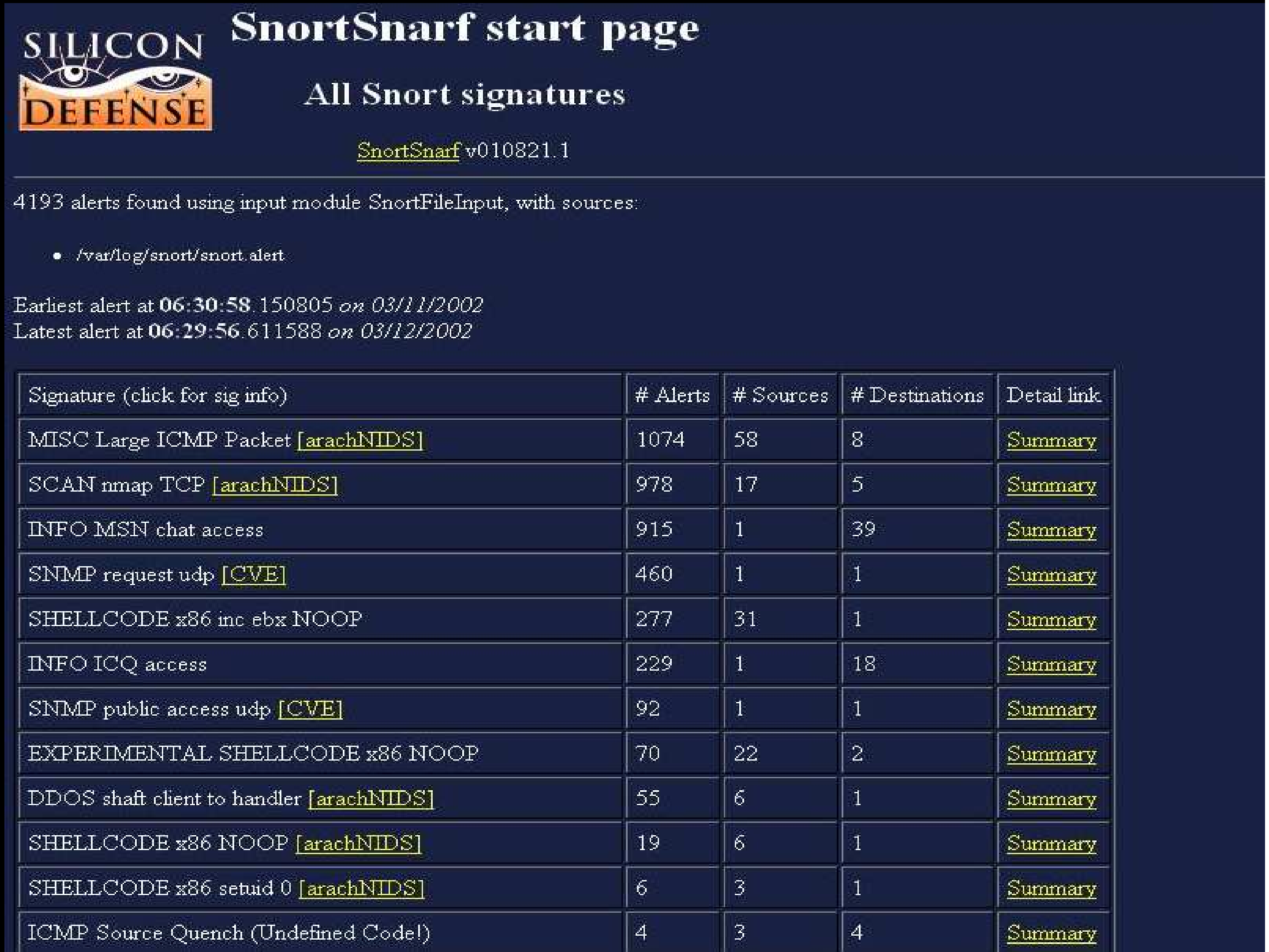
- Fundamentos de un IDS
- Tipos
  - Host / Servidor / FS
  - Red
- Arquitecturas distribuidas
- Importancia de las actualizaciones

# IDS. Snort

- Que es SNORT
- Plataformas y rendimiento
- Arquitectura
  - Configuracion
  - Filtros
  - Reglas
    - Actualización
- Respuestas activas
- Manejo de incidencias
  - SnortSnarf
  - SQL

# Implementando Snort I

## Informes automatizados



SnortSnarf start page

All Snort signatures

SnortSnarf v010821.1

4193 alerts found using input module SnortFileInput, with sources:

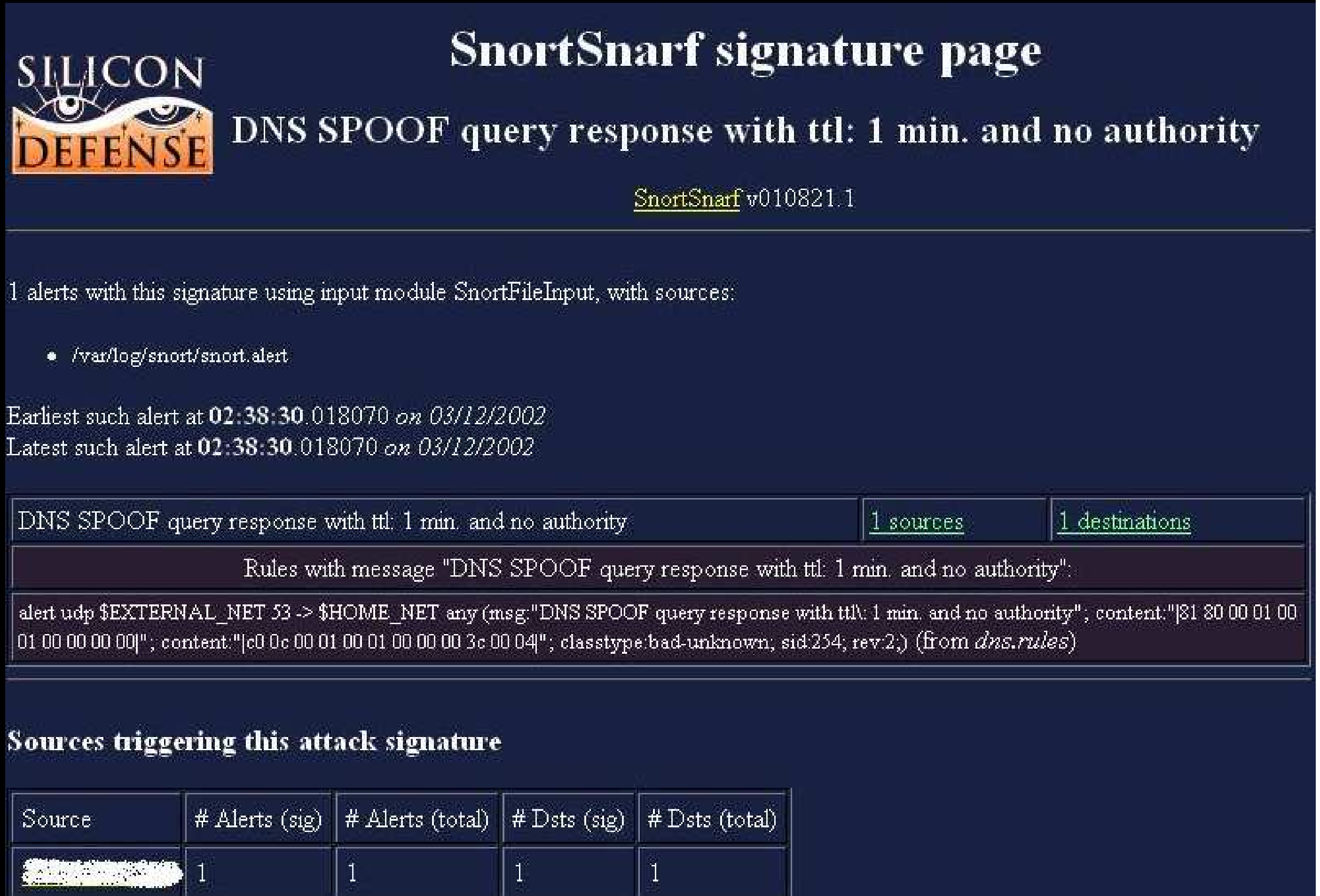
- /var/log/snort/snort.alert

Earliest alert at **06:30:58**.150805 *on 03/11/2002*
Latest alert at **06:29:56**.611588 *on 03/12/2002*

| Signature (click for sig info) | # Alerts | # Sources | # Destinations | Detail link |
|---|---|---|---|---|
| MISC Large ICMP Packet [arachNIDS] | 1074 | 58 | 8 | Summary |
| SCAN nmap TCP [arachNIDS] | 978 | 17 | 5 | Summary |
| INFO MSN chat access | 915 | 1 | 39 | Summary |
| SNMP request udp [CVE] | 460 | 1 | 1 | Summary |
| SHELLCODE x86 inc ebx NOOP | 277 | 31 | 1 | Summary |
| INFO ICQ access | 229 | 1 | 18 | Summary |
| SNMP public access udp [CVE] | 92 | 1 | 1 | Summary |
| EXPERIMENTAL SHELLCODE x86 NOOP | 70 | 22 | 2 | Summary |
| DDOS shaft client to handler [arachNIDS] | 55 | 6 | 1 | Summary |
| SHELLCODE x86 NOOP [arachNIDS] | 19 | 6 | 1 | Summary |
| SHELLCODE x86 setuid 0 [arachNIDS] | 6 | 3 | 1 | Summary |
| ICMP Source Quench (Undefined Code!) | 4 | 3 | 4 | Summary |

# Implementando Snort II

Investigando ataques



## SnortSnarf signature page

### DNS SPOOF query response with ttl: 1 min. and no authority

SnortSnarf v010821.1

1 alerts with this signature using input module SnortFileInput, with sources:

- /var/log/snort/snort.alert

Earliest such alert at **02:38:30**.018070 *on 03/12/2002*
Latest such alert at **02:38:30**.018070 *on 03/12/2002*

| DNS SPOOF query response with ttl: 1 min. and no authority | 1 sources | 1 destinations |
|---|---|---|
| Rules with message "DNS SPOOF query response with ttl: 1 min. and no authority": | | |

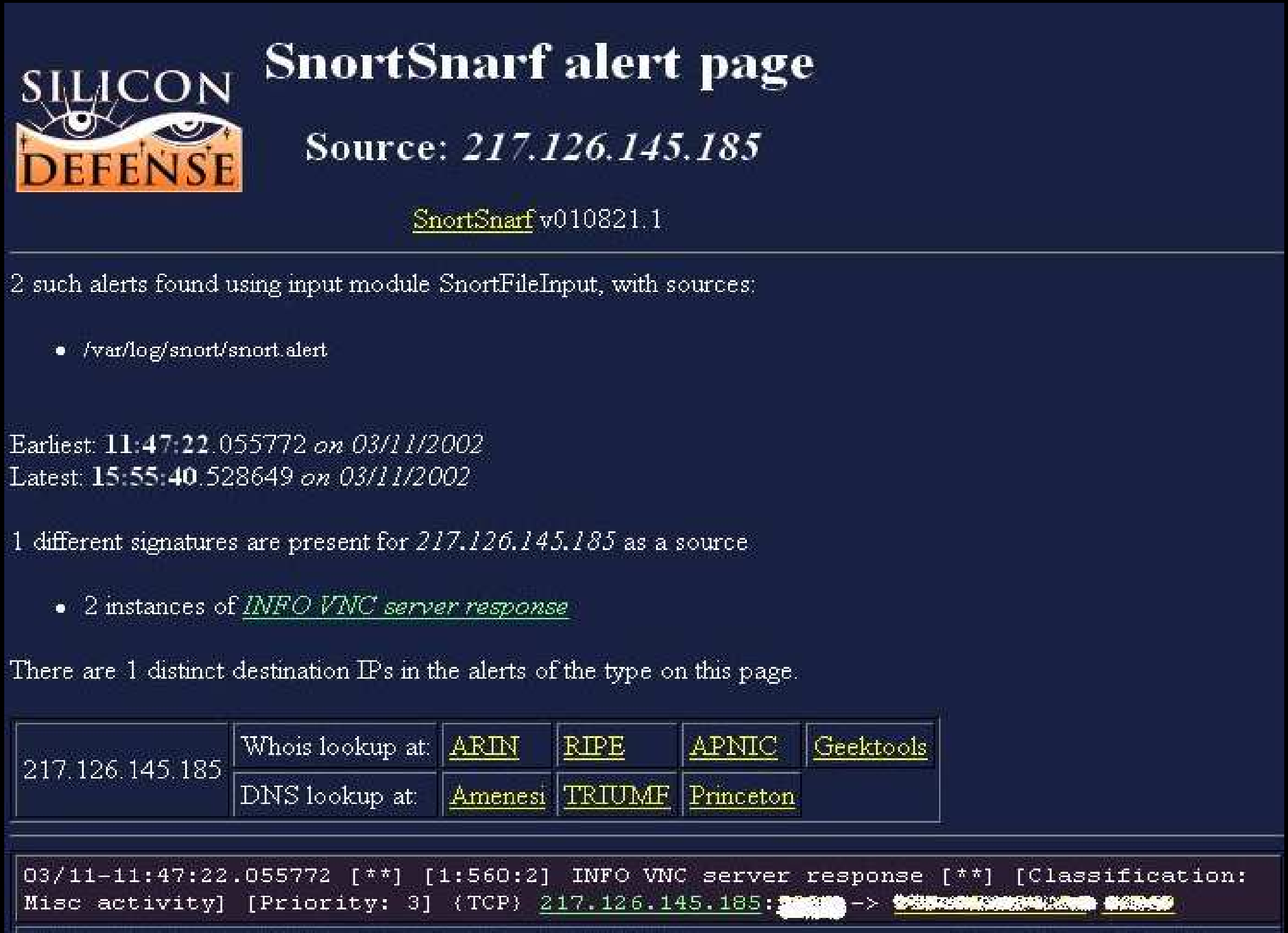
```
alert udp $EXTERNAL_NET 53 -> $HOME_NET any (msg:"DNS SPOOF query response with ttl\: 1 min. and no authority"; content:"|81 80 00 01 00 01 00 00 00 00|"; content:"|c0 0c 00 01 00 01 00 00 00 3c 00 04|"; classtype:bad-unknown; sid:254; rev:2;) (from dns.rules)
```

**Sources triggering this attack signature**

| Source | # Alerts (sig) | # Alerts (total) | # Dsts (sig) | # Dsts (total) |
|---|---|---|---|---|
| [redacted] | 1 | 1 | 1 | 1 |

# Implementando Snort III

## Investigando atacantes



# SnortSnarf alert page

## Source: *217.126.145.185*

SnortSnarf v010821.1

2 such alerts found using input module SnortFileInput, with sources:

- /var/log/snort/snort.alert

Earliest: **11:47:22**.055772 *on 03/11/2002*
Latest: **15:55:40**.528649 *on 03/11/2002*

1 different signatures are present for *217.126.145.185* as a source

- 2 instances of *INFO VNC server response*

There are 1 distinct destination IPs in the alerts of the type on this page.

| | | | | | |
|---|---|---|---|---|---|
| 217.126.145.185 | Whois lookup at: | ARIN | RIPE | APNIC | Geektools |
| | DNS lookup at: | Amenesi | TRIUMF | Princeton | |

```
03/11-11:47:22.055772 [**] [1:560:2] INFO VNC server response [**] [Classification:
Misc activity] [Priority: 3] {TCP} 217.126.145.185:[redacted] -> [redacted] [redacted]
```