



Capa de red

- ◆ Asignación de direcciones IP
- ◆ DNAT / SNAT
- ◆ ICMP
- ◆ IPv6

Asignación de direcciones IP

Cada host en una red debe tener una dirección IP compatible con la red a la que pertenece, y a su vez dos hosts no pueden tener la misma dirección. Existen dos esquemas para asignación de direcciones

- ◆ **Direccionamiento estático:** se configura la interface con una dirección IP y máscara de red fija.
- ◆ **Direccionamiento dinámico:**
 - ◆ RARP (RFC 903)
 - ◆ BOOTP (RFCs 951 y 1532)
 - ◆ DHCP (RFCs 2131, 2132, upd: 3396, 3442, 3942)

Asignación de direcciones IP

BOOTP

Es usado por un dispositivo cuando éste se inicia para conocer cuál será su dirección IP. Envía un mensaje broadcast a la red con su número MAC preguntando por su número IP. Un servidor BOOTP le informa cuál es el número de IP y le puede informar también la máscara de red, gateway y otros datos.

BOOTP es reemplazado por DHCP.

DHCP (*Dynamic Host Configuration Protocol*)

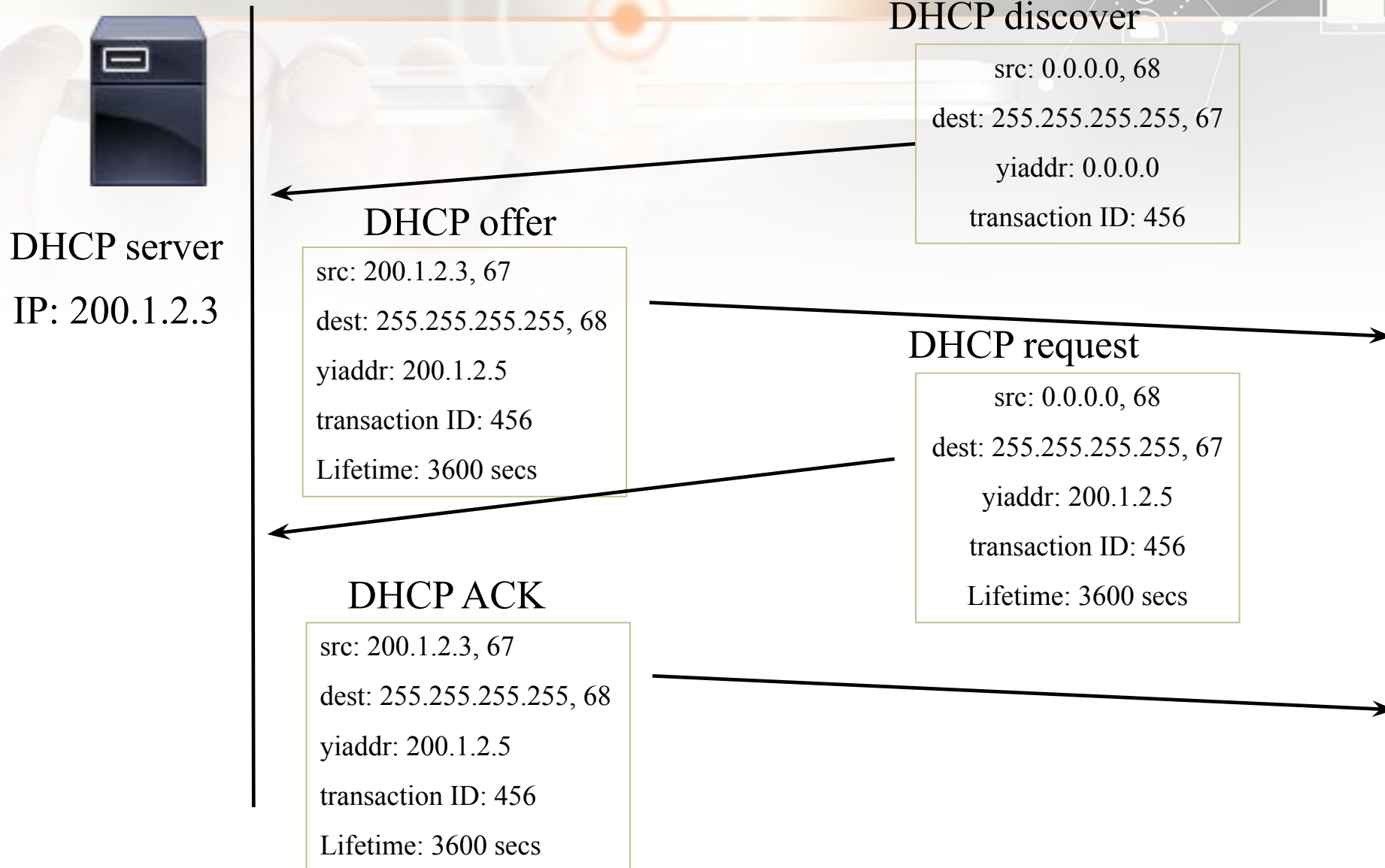
- ◆ Es un sucesor de BOOTP
- ◆ El cliente puede conocer o no la dirección IP del servidor DHCP
- ◆ La asignación de IP es dinámica, aunque permite reservar direcciones IP para direcciones MAC
- ◆ Permite obtener, además del número IP
 - ◆ Máscara de red
 - ◆ Gateways (IP)
 - ◆ DNS servers (nombre e IP)
 - ◆ etc. (ver RFC 2132)

DHCP

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
op (1)										htype (1)										hlen (1)										hops (1)									
										xid (4)																													
secs (2)																				flags (2)																			
										ciaddr (4)																													
										yiaddr (4)																													
										siaddr (4)																													
										giaddr (4)																													
										chaddr (16)																													
										sname (64)																													
										file (128)																													
										options (variable)																													

Ver RFC 2131 item 4.4

DHCP: solitud



DHCP: ejemplo

Discover

```
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
  Source Port: 68
  Destination Port: 67
  Length: 308
  Checksum: 0x61a0 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 0]
Bootstrap Protocol (Discover)
  Message type: Boot Request (1)
  Hardware type: Ethernet (0x01)
  Hardware address length: 6
  Hops: 0
  Transaction ID: 0xfe089c15
  Seconds elapsed: 0
  Bootp flags: 0x0000 (Unicast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 0.0.0.0
  Next server IP address: 0.0.0.0
  Relay agent IP address: 0.0.0.0
  Client MAC address: D-Link_12:47:cb (00:50:ba:12:47:cb)
  Client hardware address padding: 00000000000000000000
  Server host name not given
  Boot file name not given
  Magic cookie: DHCP
  Option: (53) DHCP Message Type (Discover)
  Option: (116) DHCP Auto-Configuration
  Option: (61) Client identifier
  Option: (50) Requested IP Address
  Option: (12) Host Name
  Option: (60) Vendor class identifier
  Option: (55) Parameter Request List
  Option: (255) End
  Padding: 00000000
```




Offer

```

> Internet Protocol Version 4, Src: 10.20.20.4, Dst: 255.255.255.255
- User Datagram Protocol, Src Port: 67, Dst Port: 68
    Source Port: 67
    Destination Port: 68
    Length: 308
    Checksum: 0x8a44 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 2]
- Bootstrap Protocol (Offer)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0xfe089c15
    Seconds elapsed: 0
    Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 10.20.20.20
    Next server IP address: 10.20.20.4
    Relay agent IP address: 0.0.0.0
    Client MAC address: D-Link_12:47:cb (00:50:ba:12:47:cb)
    Client hardware address padding: 0000000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
    Option: (53) DHCP Message Type (Offer)
    Option: (54) DHCP Server Identifier
    Option: (51) IP Address Lease Time
    Option: (1) Subnet Mask
    Option: (15) Domain Name
    Option: (6) Domain Name Server
    Option: (255) End
    Padding: 0000000000000000000000000000000000000000

```


DHCP: ejemplo

Request

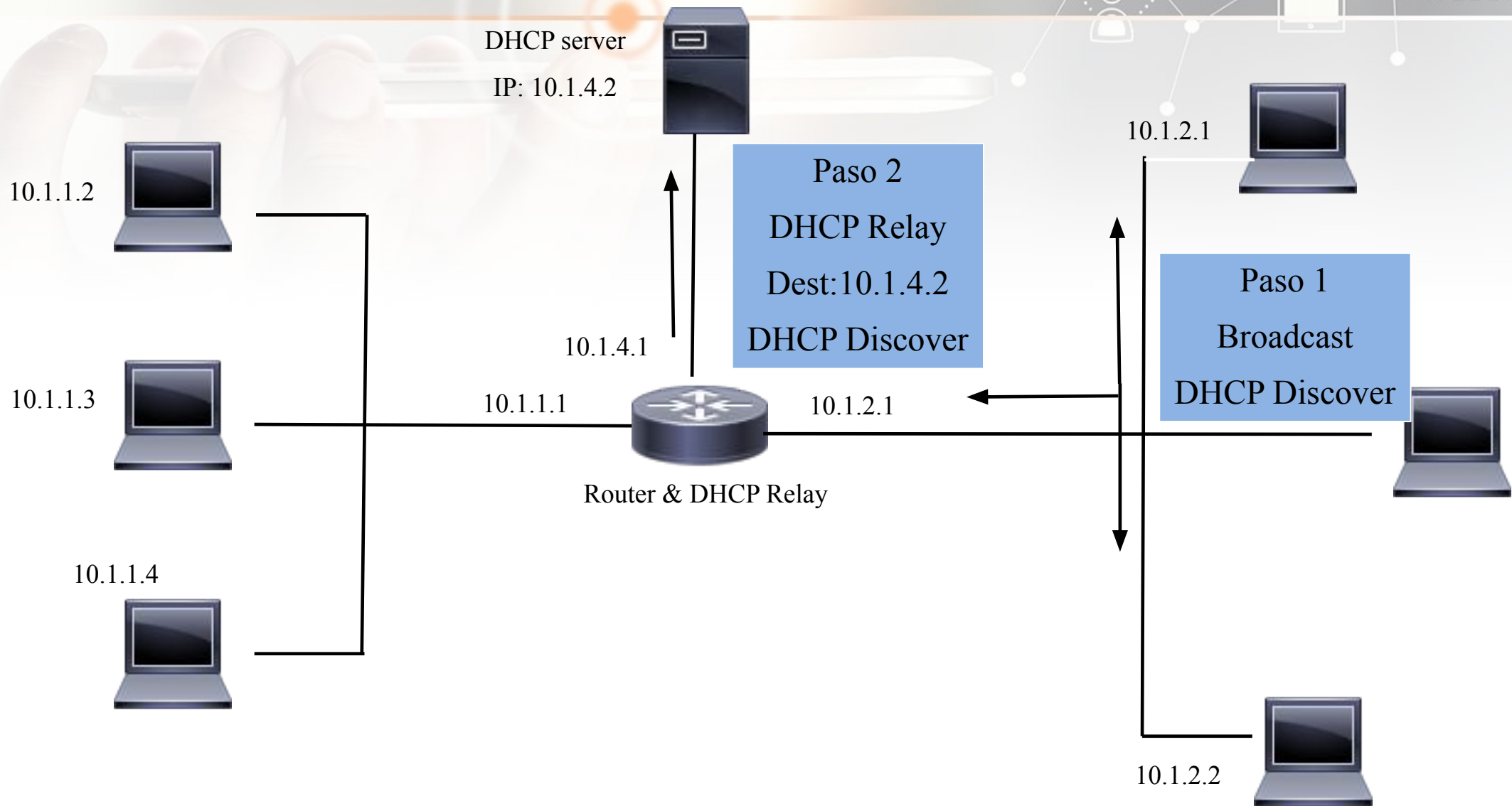
```
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
  Source Port: 68
  Destination Port: 67
  Length: 322
  Checksum: 0xf62b [unverified]
  [Checksum Status: Unverified]
  [Stream index: 0]
Bootstrap Protocol (Request)
  Message type: Boot Request (1)
  Hardware type: Ethernet (0x01)
  Hardware address length: 6
  Hops: 0
  Transaction ID: 0xfe089c15
  Seconds elapsed: 0
  Bootp flags: 0x0000 (Unicast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 0.0.0.0
  Next server IP address: 0.0.0.0
  Relay agent IP address: 0.0.0.0
  Client MAC address: D-Link_12:47:cb (00:50:ba:12:47:cb)
  Client hardware address padding: 00000000000000000000
  Server host name not given
  Boot file name not given
  Magic cookie: DHCP
  Option: (53) DHCP Message Type (Request)
  Option: (61) Client identifier
  Option: (50) Requested IP Address
  Option: (54) DHCP Server Identifier
  Option: (12) Host Name
  Option: (81) Client Fully Qualified Domain Name
  Option: (60) Vendor class identifier
  Option: (55) Parameter Request List
  Option: (255) End
```

DHCP: ejemplo

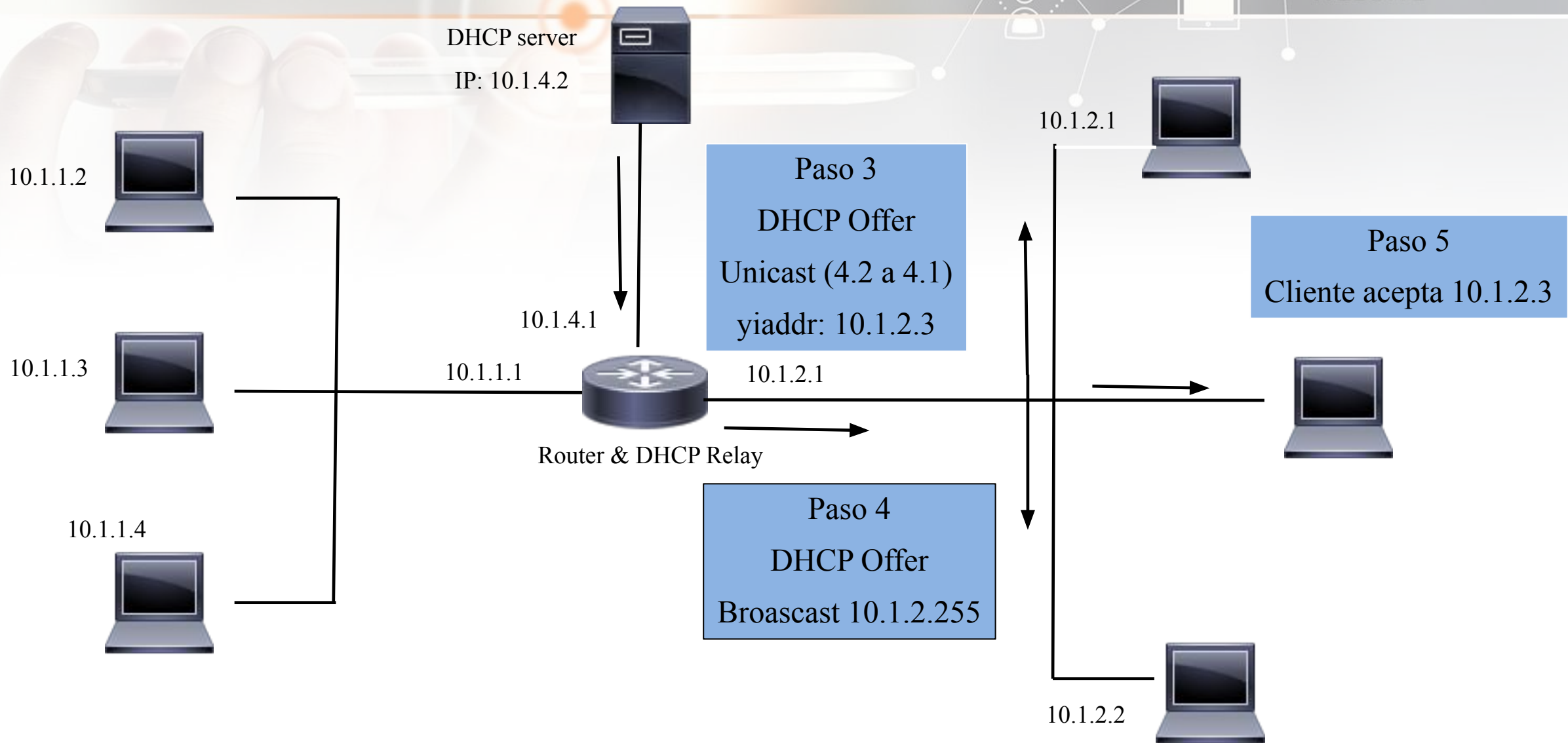
ACK

```
Internet Protocol Version 4, Src: 10.20.20.4, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 67, Dst Port: 68
  Source Port: 67
  Destination Port: 68
  Length: 317
  Checksum: 0xb7a8 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 2]
Bootstrap Protocol (ACK)
  Message type: Boot Reply (2)
  Hardware type: Ethernet (0x01)
  Hardware address length: 6
  Hops: 0
  Transaction ID: 0xfe089c15
  Seconds elapsed: 0
  Bootp flags: 0x0000 (Unicast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 10.20.20.20
  Next server IP address: 10.20.20.4
  Relay agent IP address: 0.0.0.0
  Client MAC address: D-Link_12:47:cb (00:50:ba:12:47:cb)
  Client hardware address padding: 00000000000000000000
  Server host name not given
  Boot file name not given
  Magic cookie: DHCP
  Option: (53) DHCP Message Type (ACK)
  Option: (54) DHCP Server Identifier
  Option: (51) IP Address Lease Time
  Option: (81) Client Fully Qualified Domain Name
  Option: (1) Subnet Mask
  Option: (15) Domain Name
  Option: (6) Domain Name Server
  Option: (255) End
```

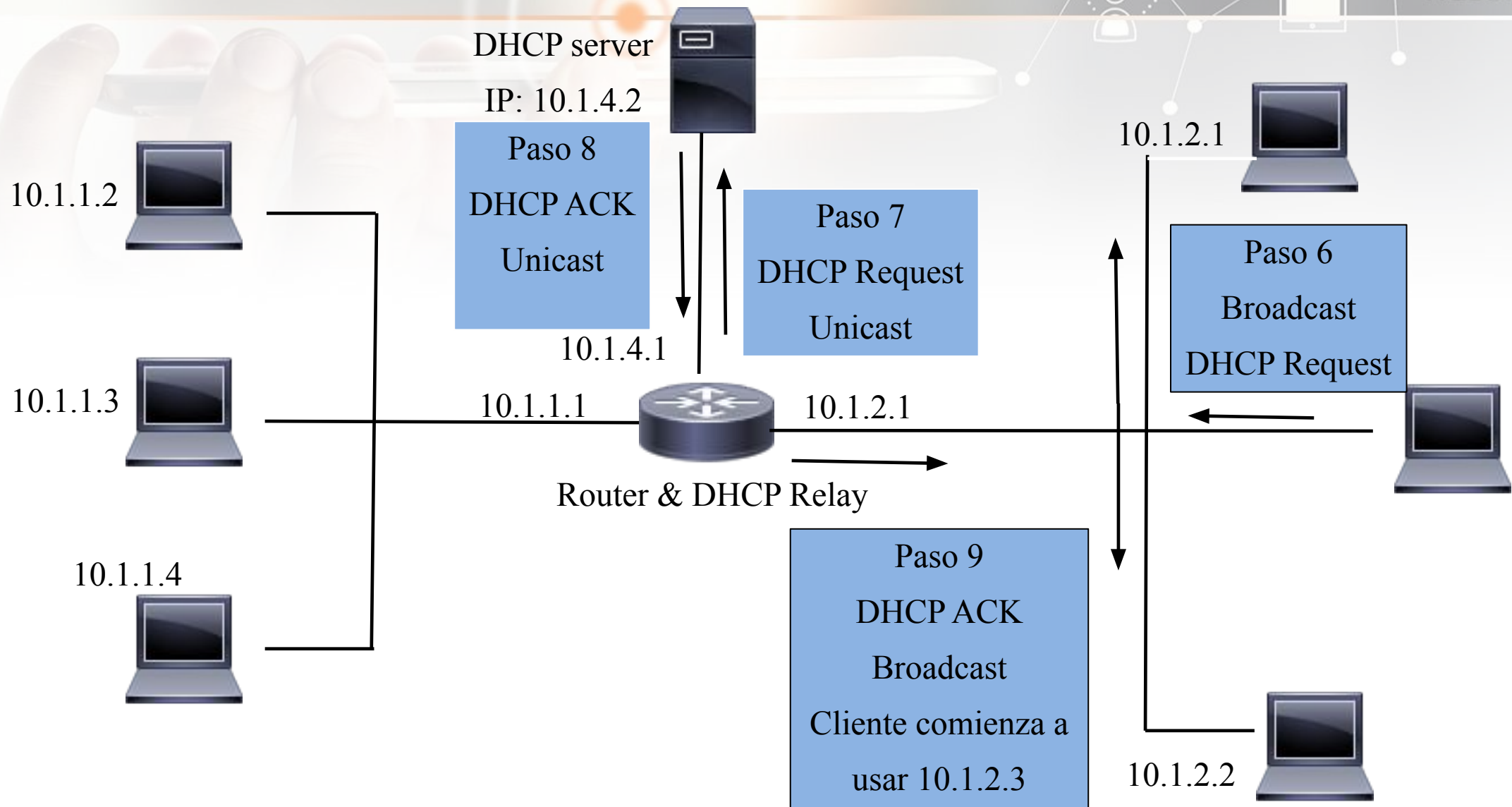
DHCP: relay agent



DHCP: relay agent



DHCP: relay agent



DHCP: renew request

```

> Internet Protocol Version 4, Src: 192.168.1.101, Dst: 192.168.1.1
> User Datagram Protocol, Src Port: 68, Dst Port: 67
  Bootstrap Protocol (Request)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x0dd1b124
    Seconds elapsed: 0
    Bootp flags: 0x0000 (Unicast)
    Client IP address: 192.168.1.101
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: AsustekC_22:77:a0 (40:16:7e:22:77:a0)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
    Option: (53) DHCP Message Type (Request)
  Option: (61) Client identifier
    Length: 7
    Hardware type: Ethernet (0x01)
    Client MAC address: AsustekC_22:77:a0 (40:16:7e:22:77:a0)

```

```

  Option: (12) Host Name
  Option: (81) Client Fully Qualified Domain Name
  Option: (60) Vendor class identifier
  Option: (55) Parameter Request List
    Length: 12
    Parameter Request List Item: (1) Subnet Mask
    Parameter Request List Item: (15) Domain Name
    Parameter Request List Item: (3) Router
    Parameter Request List Item: (6) Domain Name Server
    Parameter Request List Item: (44) NetBIOS over TCP/IP Name Server
    Parameter Request List Item: (46) NetBIOS over TCP/IP Node Type
    Parameter Request List Item: (47) NetBIOS over TCP/IP Scope
    Parameter Request List Item: (31) Perform Router Discover
    Parameter Request List Item: (33) Static Route
    Parameter Request List Item: (121) Classless Static Route
    Parameter Request List Item: (249) Private/Classless Static Route (Microsoft)
    Parameter Request List Item: (43) Vendor-Specific Information
  Option: (255) End

```


DHCP: renew ACK

```

> Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.101
> User Datagram Protocol, Src Port: 67, Dst Port: 68
< Bootstrap Protocol (ACK)
  Message type: Boot Reply (2)
  Hardware type: Ethernet (0x01)
  Hardware address length: 6
  Hops: 0
  Transaction ID: 0x0dd1b124
  Seconds elapsed: 0
  > Bootp flags: 0x0000 (Unicast)
    Client IP address: 192.168.1.101
    Your (client) IP address: 192.168.1.101
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: AsustekC_22:77:a0 (40:16:7e:22:77:a0)
    Client hardware address padding: 000000000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP

```

[illegible]

DNS dinámico

Integración con DHCP

Cliente
DHCP



DHCP discover

DHCP offer

DHCP request

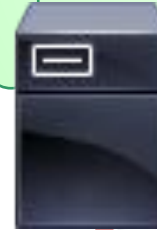
DHCP ACK

DHCP code 81 +
FQDN cliente DHCP

dhcp.conf

```
ddns-updates on;  
ddns-update-style interim;  
ddns-domainname "midominio.com";  
ddns-rev-domainname "in-addr.arpa";  
option domain-name "midominio.com";
```

Servidor
DHCP




Registro de
actualización

Servidor
DNS



DDNS: ejemplo



Security

View and change router settings

Firewall DMZ Apps and Gaming

DDNS | Single Port Forwarding | Port Range Forwarding | Port Range Triggering

Select a provider: NO-IP.com ▾

User Name: mgarbe@itba.edu.ar

No-IP Password: [REDACTED]

Host name: [REDACTED]

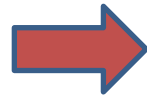
Internet IP address: [REDACTED]

Status: Success

Update

DHCP

DHCP es atendido por un servidor DHCP, no por un router



+



+

DNS Caching-only Server

+

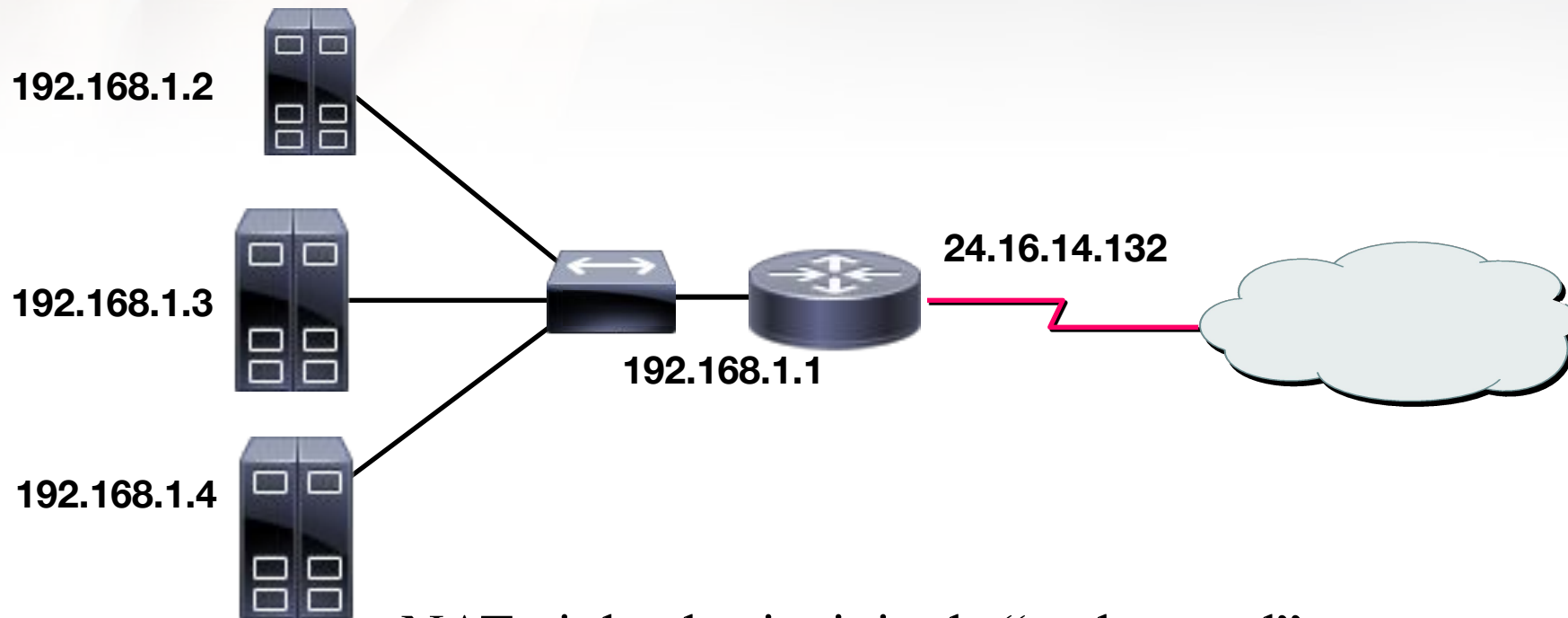
DHCP Server

+

Firewall

NAT (SNAT)

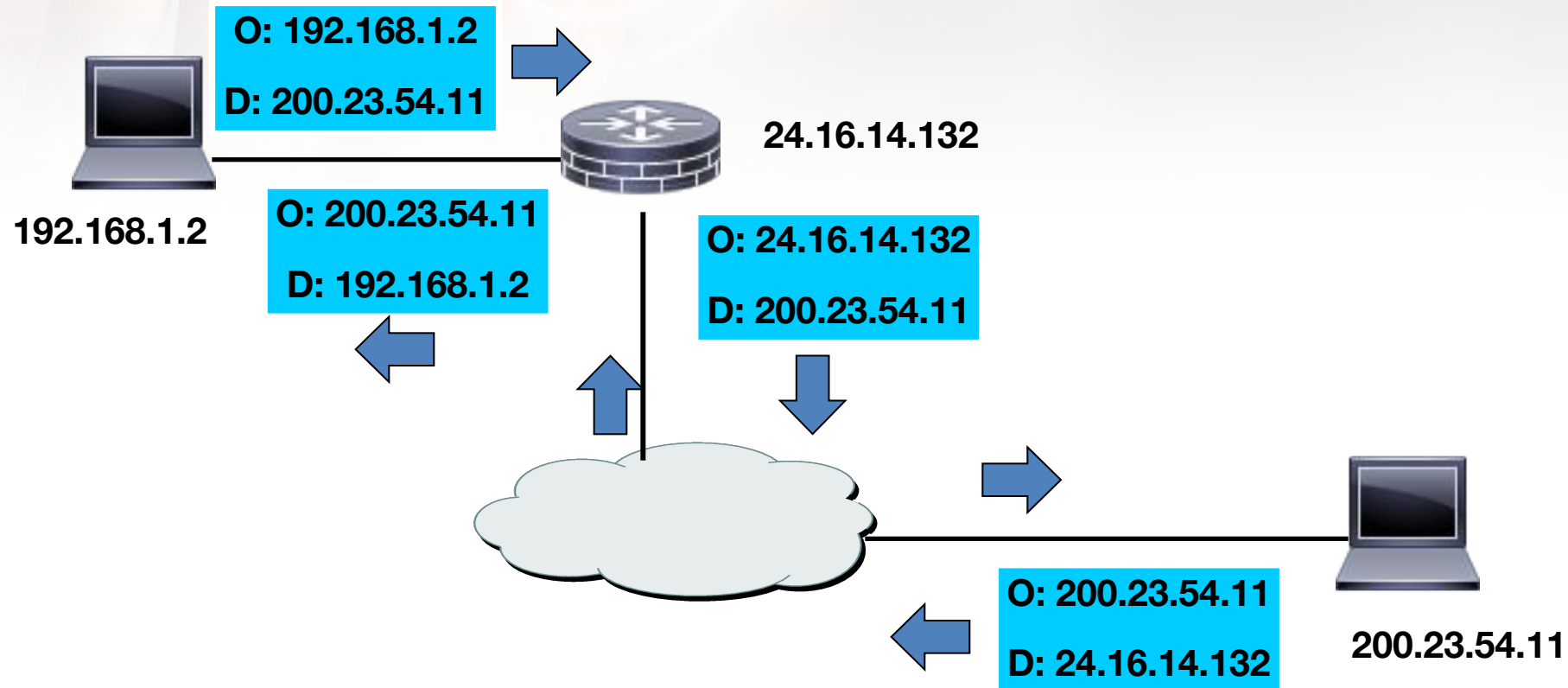
Una medida para aliviar la necesidad de IPs públicas fue **NAT**
(1994, RFCs 1631 y 3022)



NAT viola el principio de “end-to-end”.
Debe ser tenido en cuenta por aplicaciones P2P

SNAT

Ejemplo

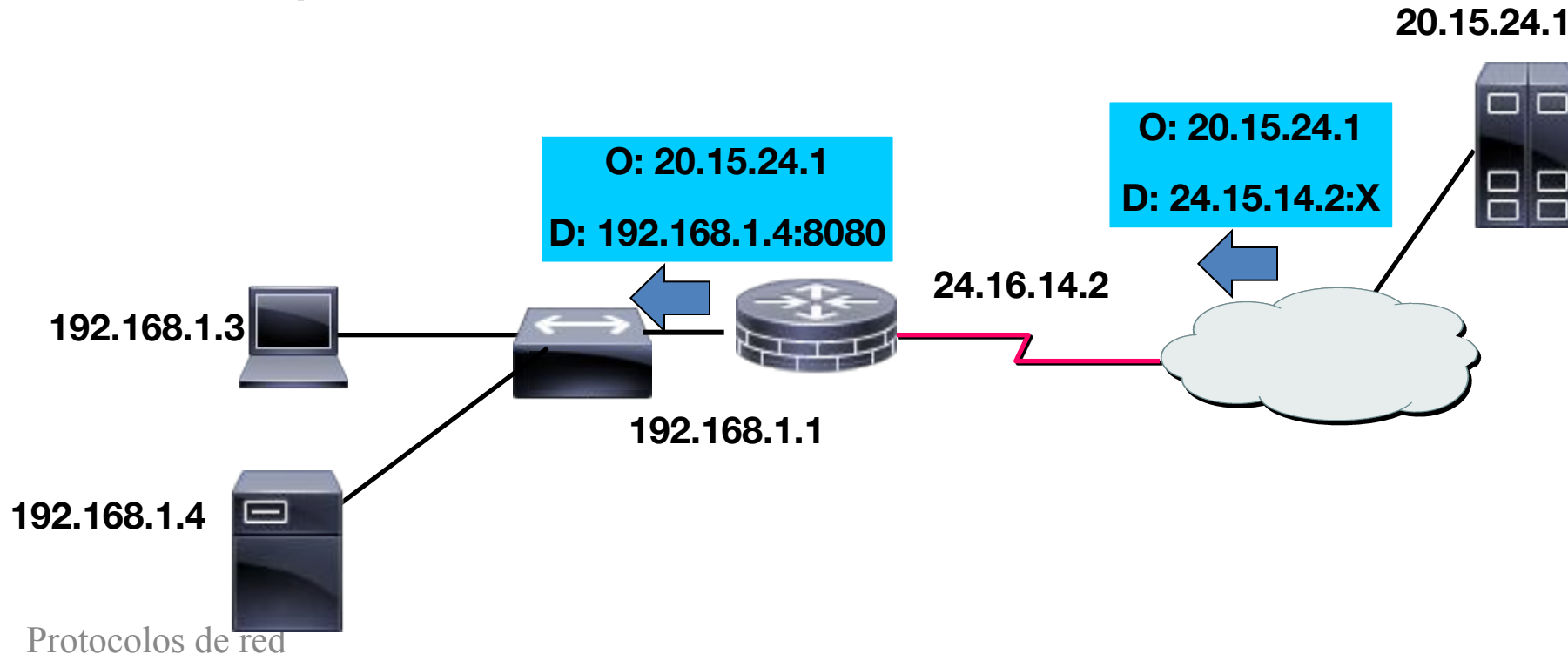


¿Qué campos se modifican al aplicar NAT a un paquete IP?

Vers	Hlen	ToS	Longitud total	
Identificación			Flags	Desplazam. de fragmento
TTL	Protocolo		Header checksum	
Dirección origen				
Dirección destino				
Opciones				Relleno
Datos				

DNAT: port forwarding

- ◆ Quiero conectarme a un servidor en 192.168.1.4:8080
- ◆ Solución 1: configurar NAT para reenviar conexiones entrantes de puerto X a 192.168.1.4:8080



DNAT: Ejemplo



Security

View and change router settings

Firewall

DMZ

Apps and Gaming

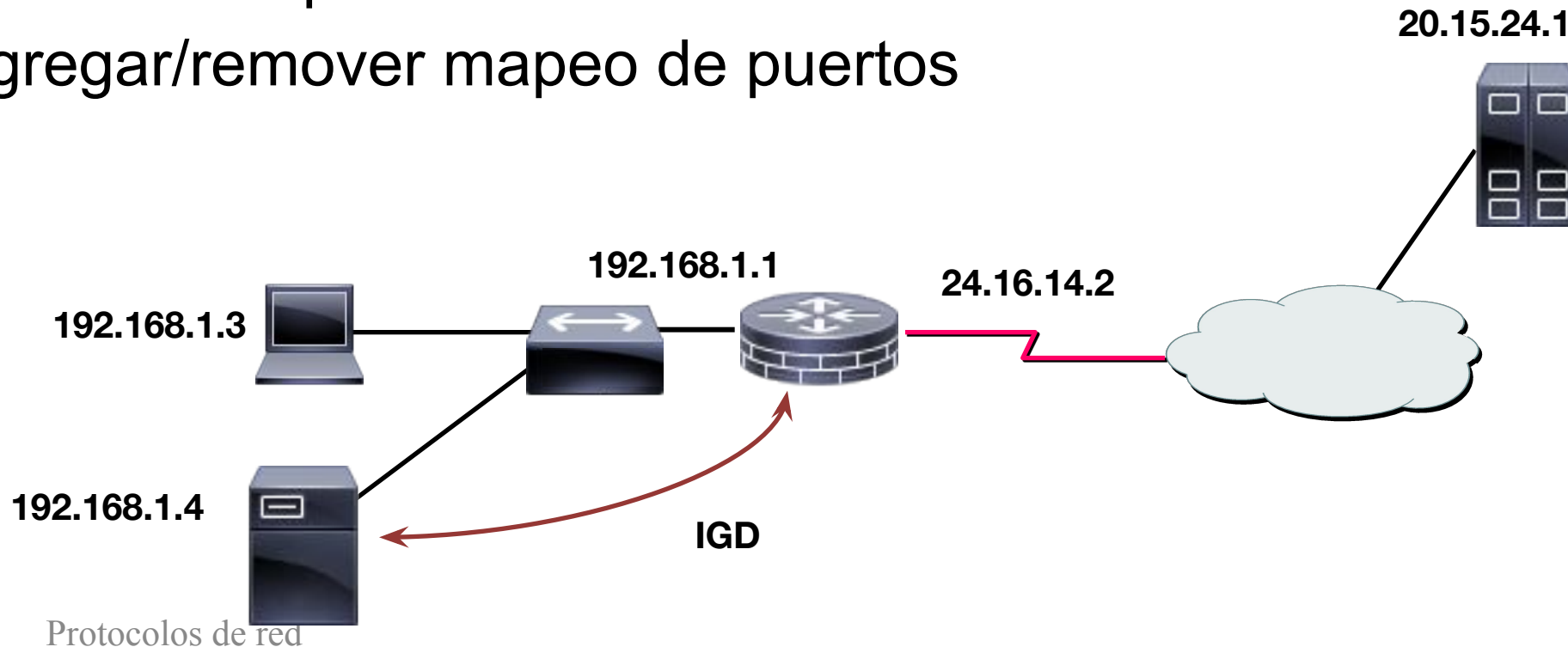
DDNS | [Single Port Forwarding](#) | [Port Range Forwarding](#) | [Port Range Triggering](#)

Application name	External Port	Internal Port	Protocol	Device IP#	Enabled	
Apache	9090	8080	TCP	192.168.1.142	True	Edit/ Delete
Remoto Marce	4489	3389	Both	192.168.1.127	True	Edit/ Delete
Apache Crea	9093	8080	Both	192.168.1.149	True	Edit/ Delete
Front	9100	8100	Both	192.168.1.149	True	Edit/ Delete
Back	9105	8105	Both	192.168.1.149	True	Edit/ Delete

Add a new Single Port Forwarding

DNAT: port forwarding

- ◆ Solución 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Permite al host “nateado”
 - ◆ Aprender IP pública
 - ◆ Agregar/remover mapeo de puertos



UPnP

Note: Make sure the nat is **enable** if you want the UPnP configuration take effect

Current UPnP Status:

Enabled

Disable

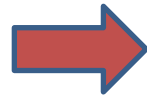
Current UPnP Settings List

ID	App Description	External Port	Protocol	Internal Port	IP Address	Status
1	uTorrent (TCP)	21103	TCP	21103	192.168.1.106	Enabled
2	uTorrent (UDP)	21103	UDP	21103	192.168.1.106	Enabled

Refresh

NAT

Tanto DNAT como SNAT son funciones de un firewall, no de un router



+



+

DNS Caching-only Server

+

DHCP Server

+



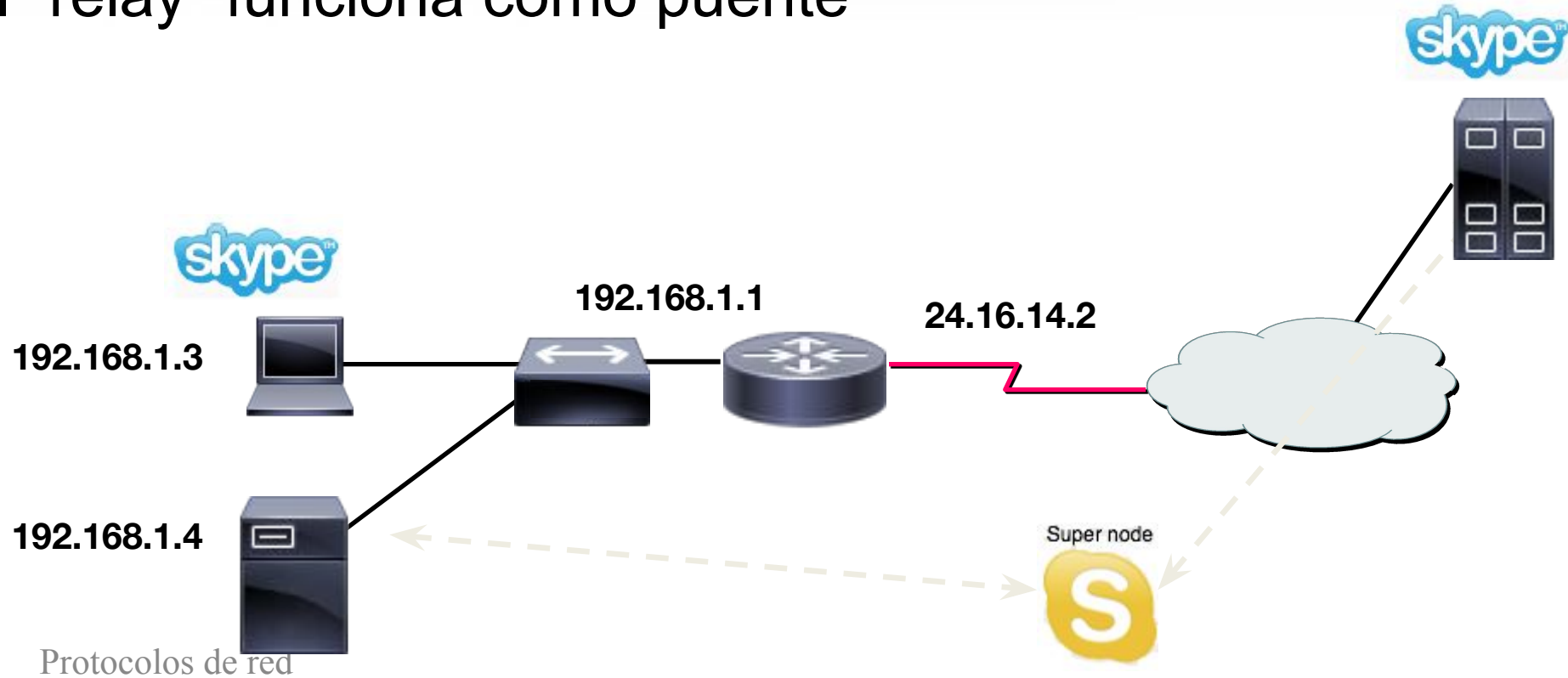
+

Webapp Server

NAT

◆ Solución 3: relaying (p.e. Skype)

- ◆ Host “nateado” conecta con “relay”
- ◆ Host externo conecta con “relay”
- ◆ El “relay” funciona como puente



ICMP

**IP no maneja errores ni retransmisiones.
En caso de querer avisar de un error se debe usar
mensajes ICMP**

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Type										Code										Checksum																			
Data (Depende de Código y Dato)																																							

Los valores para los campos Tipo y Código se definen en RFC1700

ICMP: Tipo y Código más comunes

Tipo	Código	Descripción
3	0	Net unreachable
3	1	Host unreachable
3	2	Protocol unreachable
0	0	Echo reply
8	0	Echo request
5	0	Redirect datagrams for the Network
5	1	Redirect datagrams for the Host
11	0	TTL exceeded
13/14	0	Timestamp request/reply

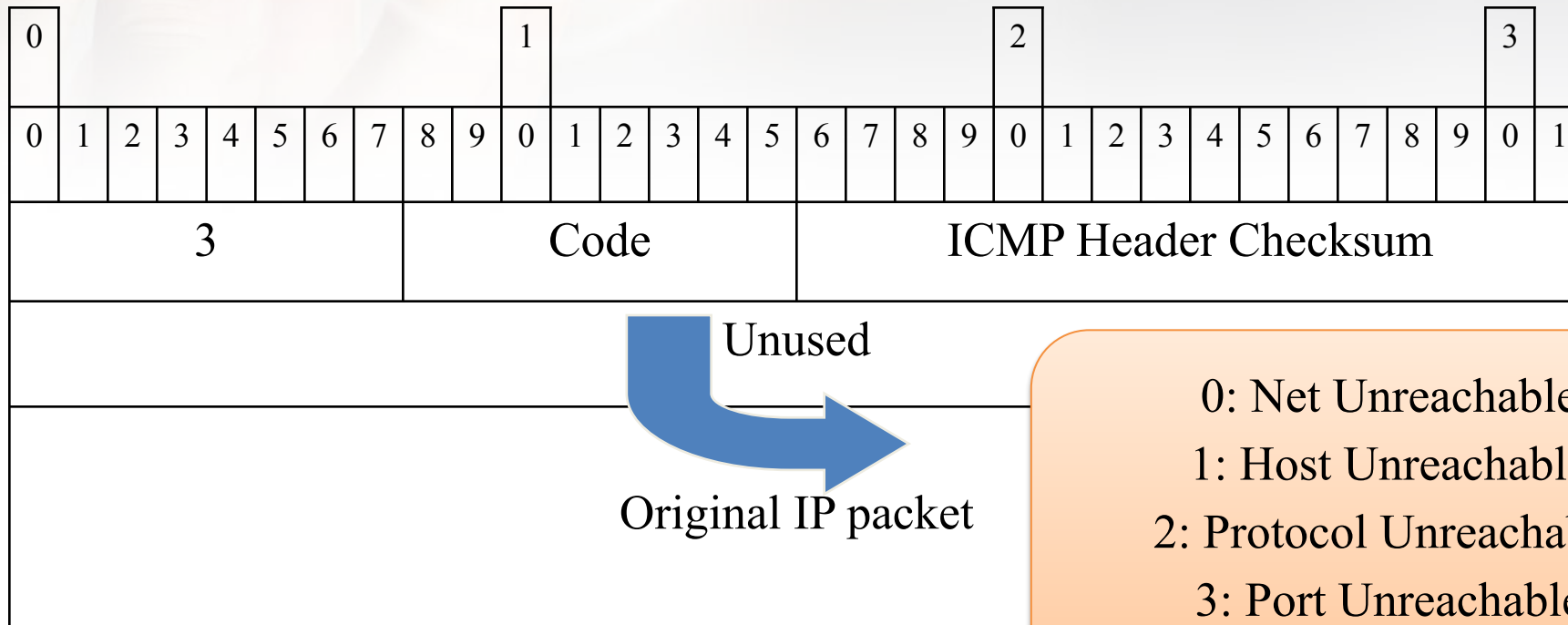
ICMP: ejemplos

Timestamp request / reply

0																1																2																3															
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1																						
13/14										0										ICMP Header Checksum																																											
Identifier																				Sequence number																																											
Originate timestamp																																																															
Receive timestamp																																																															
Transit timestamp																																																															

ICMP: ejemplos

Destino inalcanzable



0: Net Unreachable
1: Host Unreachable
2: Protocol Unreachable
3: Port Unreachable
4: Fragmentation needed
...

Utilidades básicas de networking

ping *hostDestino*: Utiliza mensajes ICMP tipo 8 y 0

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
8 ó 0										0										ICMP Header Checksum																			
Identifier															Sequence number																								
Optional data																																							

Si *hostDestino* es alcanzable, mostrará un resumen de paquetes enviados, recibidos, perdidos y tiempo medio de transferencia.

Utilidades básicas de networking

```
user@server:~$ ping 200.49.213.50
```

```
PING 200.49.213.50 (200.49.213.50) 56(84) bytes of data.
```

```
64 bytes from 200.49.213.50: icmp_seq=1 ttl=55 time=21.4 ms
```

```
64 bytes from 200.49.213.50: icmp_seq=2 ttl=55 time=37.2 ms
```

```
64 bytes from 200.49.213.50: icmp_seq=3 ttl=55 time=35.7 ms
```

```
64 bytes from 200.49.213.50: icmp_seq=4 ttl=55 time=25.2 ms
```

```
64 bytes from 200.49.213.50: icmp_seq=5 ttl=55 time=23.2 ms
```

```
--- 200.49.213.50 ping statistics ---
```

```
5 packets transmitted, 5 received, 0% packet loss, time 4036ms
```

```
rtt min/avg/max/mdev = 21.424/28.601/37.273/6.594 ms
```

Utilidades básicas de networking

```
user@server:~$ ping -c1 200.49.213.50 -r
PING 200.49.213.50 (200.49.213.50) 56(124) bytes of data.
64 bytes from 200.49.213.50: icmp_seq=1 ttl=56 time=2286 ms
RR:      192.168.2.10      209.13.133.250
        200.26.75.69      209.13.133.57
        200.51.241.170     200.51.240.82
        200.51.241.161     200.51.241.57
        172.30.248.46

--- 200.49.213.50 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2286.637/2286.637/2286.637/0.000 ms
```

Utilidades básicas de networking

En caso de tener problemas de conectividad, el comando **ping** nos puede ayudar a localizar dónde se encuentra el problema.

1. `ping 127.0.0.1`
2. `ping direccion_IP_local`
3. `ping host_en_mi_segmento`
4. `ping host_en_otro_segmento`
5. `ping host_fuera_de_la_red`

Tener en cuenta que muchos routers o hosts no responden solicitudes de eco.

Utilidades básicas de networking

tracert

Al igual que **ping**, trata de alcanzar un host destino, pero informa cada router por el que va pasando hasta llegar a destino.

Lo que hace en realidad es enviar mensajes **ICMP *echo request*** (igual que ping) pero primero con un tiempo de vida de 1 salto, luego de 2 saltos, 3 saltos, etc, con un máximo por defecto de 30 saltos.

Especialmente útil en dos casos:

- ❑ Detectar dónde “muere” un paquete
- ❑ Detectar dónde un paquete se “empantana”

Traceroute: demoras reales

```
user@server:~$ traceroute campus.itba.edu.ar
```

```
Tracing route to campus.itba.edu.ar [192.230.225.58]
```

1	<1 ms	<1 ms	<1 ms	192.168.1.1
2	*	*	*	Request timed out.
...				
6	15 ms	15 ms	15 ms	129-161-89-200.fibertel.com.ar [200.89.161.129]
7	13 ms	15 ms	16 ms	130-165-89-200.fibertel.com.ar [200.89.165.130]
8	13 ms	14 ms	11 ms	222-165-89-200.fibertel.com.ar [200.89.165.222]
9	*	*	*	Request timed out.
10	139 ms	140 ms	138 ms	ael-300G.ar5.MIA1.gblx.net [67.17.94.249]
11	*	*	*	Request timed out.
12	*	*	*	Request timed out.
13	160 ms	163 ms	159 ms	BLACKBOARD.ear2.Washington1.Level3.net [4.59.144.218]
14	165 ms	159 ms	164 ms	eth4-1-bb1-dc2.blackboard.com [69.196.255.133]
15	159 ms	169 ms	160 ms	eth4-3-bb1-va2.mhint [69.196.255.145]
16	166 ms	174 ms	175 ms	eth3-3-gw1-va2.mhint [69.196.255.158]
17	158 ms	159 ms	159 ms	itba.blackboard.com [192.230.225.58]

IPv6

- ◆ Direcciones de 128 bits
- ◆ Estructura de direcciones jerárquica
- ◆ Número de MAC como parte de la dirección IP
- ◆ Incorpora seguridad y multicasting
- ◆ Encabezado de tamaño fijo, pensado para hacer más eficiente su procesamiento por los routers
- ◆ Facilita QoS
- ◆ No permite fragmentación
- ◆ El protocolo IPv6 se define en RFC 2460 (Dic. 1998)

Estadísticas de uso

- <https://www.internetsociety.org/deploy360/ipv6/statistics/>
 - [IPv6 – Google](#)
 - [IPv6 Measurement Maps](#)

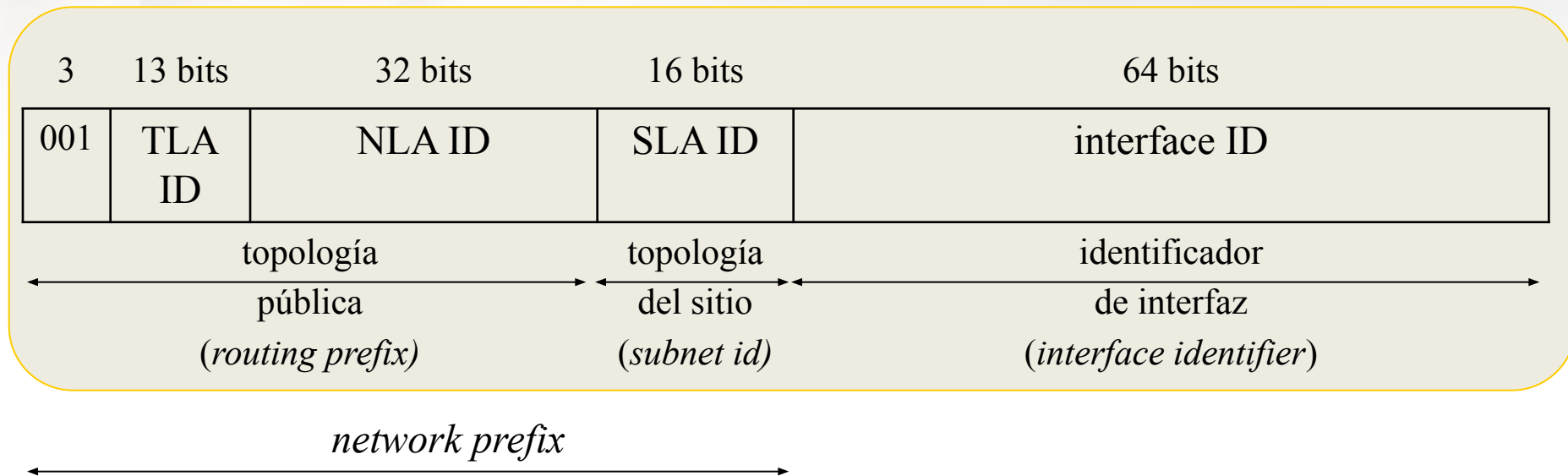
Prefijos IPv6

TIPO DE DIRECCIÓN	BITS DE PREFIJO	PREFIJO
Link-Local	1111 1110 10	FE80::/10
Unspecified	0000 ... 0 (128 bits)	::/128
Loopback	0000 ... 01 (128 bits)	::1/128
Multicast	1111 1111	FF00::/8
IPv4-Mapped	000 ... 0111111111111111 (96 bits)	::FFFF/96
ULA	1111 110	FC00::/7
Global Unicast	001	2000::/3
Anycast		

IPv6

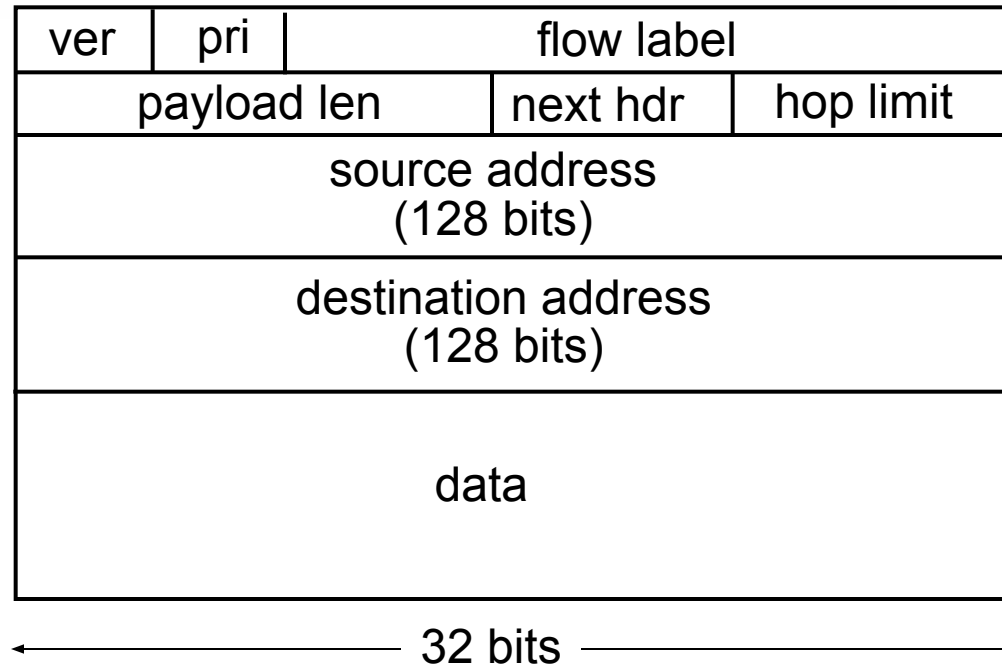
- ◆ IPv4-compatible:
 - ◆ 0:0:0:0:0:FFFF:192.168.30.1
 - ◆ = ::FFFF:192.168.30.1
 - ◆ = ::FFFF:C0A8:1E01
- ◆ Como URL, se la encierra entre corchetes
 - ◆ [http://\[2001:1:4F3A::206:AE14\]:8080/index.html](http://[2001:1:4F3A::206:AE14]:8080/index.html)
- ◆ No existe IP broadcast

Direcciones Global Unicast (RFC 3587)



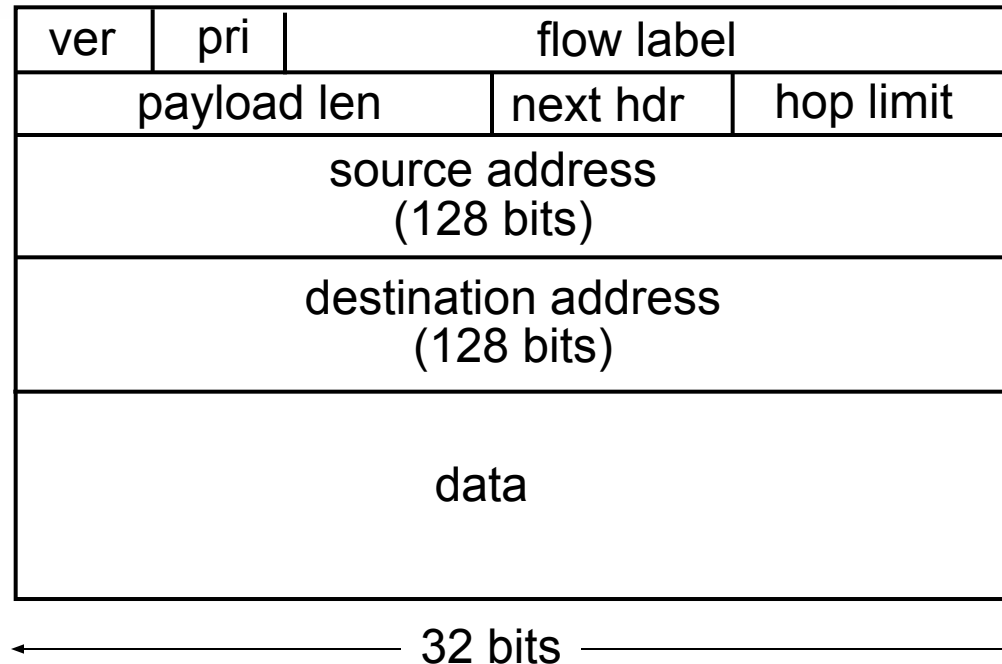
IPv6: formato

- ◆ Priority: prioridad del paquete dentro del flujo
- ◆ Flow label: identifica paquetes dentro de un mismo “flujo”
- ◆ Checksum: no está más

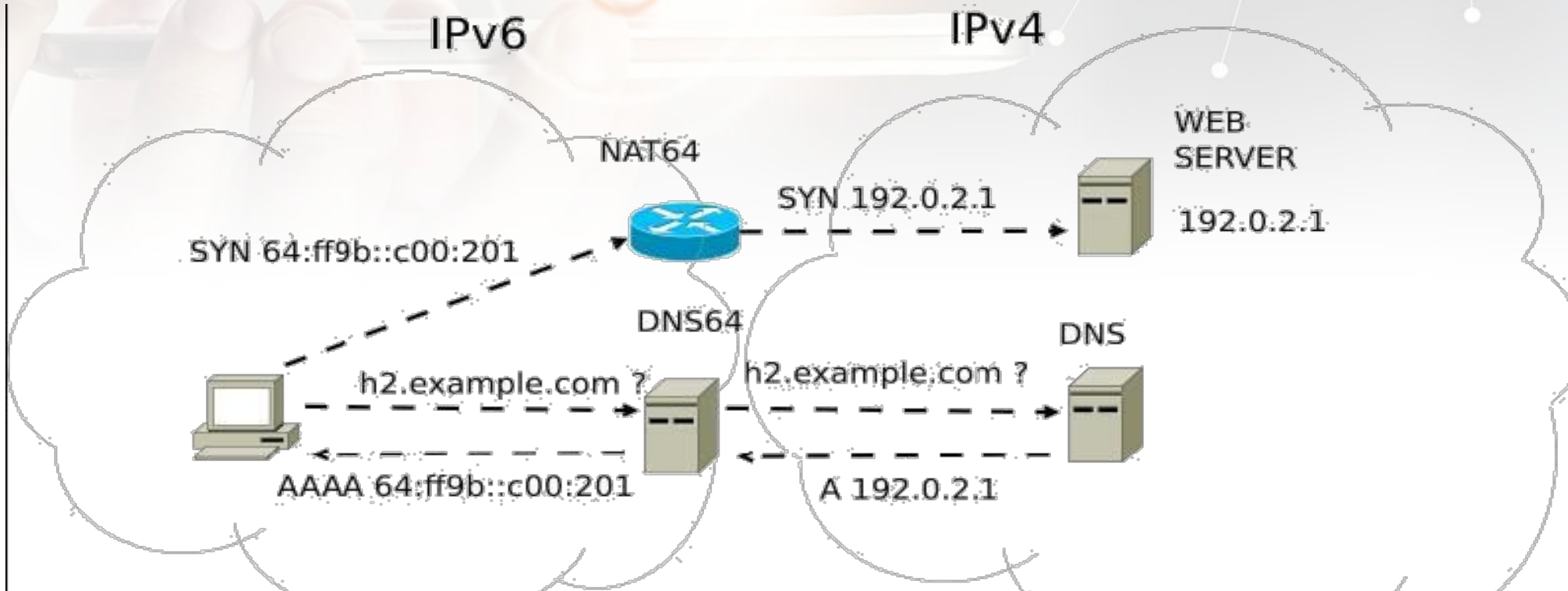


IPv6: formato

- ♦ Payload len: longitud en octetos a continuación del header
- ♦ Next header: similar a tipo de protocolo en IPv4
- ♦ Hop limit: similar a TTL en IPv4

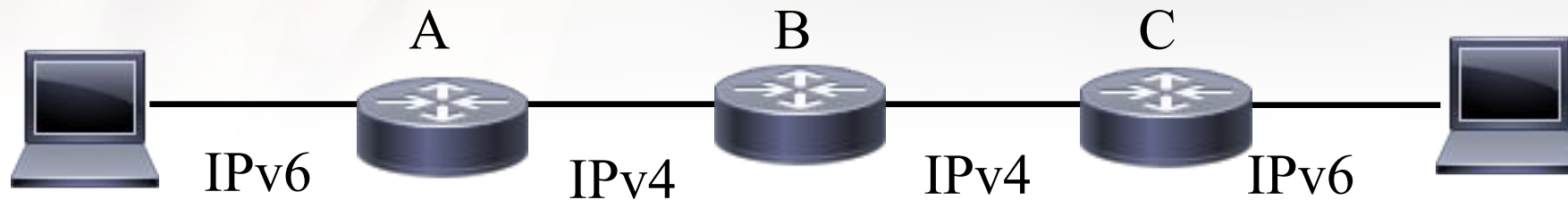


NAT64: conectando IPv6 con IPv4



NAT64 se encarga de la traducción de las direcciones entre hosts que sólo tienen conectividad IPv6 con hosts que sólo tienen conectividad IPv4. Tras la resolución del nombre por el DNS64, se pide la dirección web a través de NAT64.

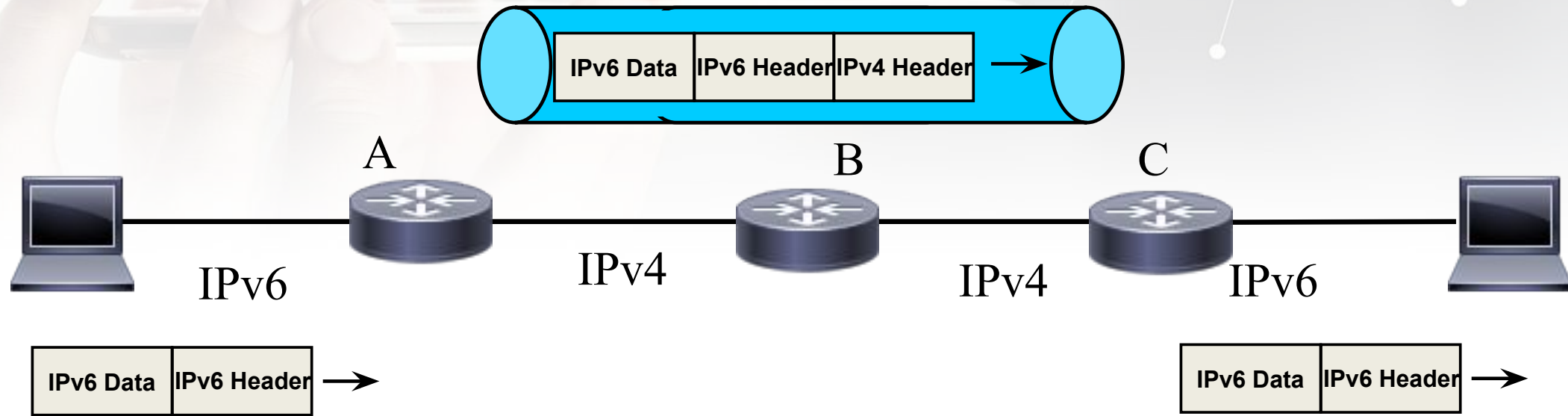
¿Qué sucede si una red intermedia sólo entiende IPv4?



El Router A debe "encapsular" el paquete Ipv6 dentro de un paquete IPv4

Tunneling: consiste en encapsular datos de un protocolo en los datos de otro protocolo de igual o mayor nivel.

IPv6 over IPv4



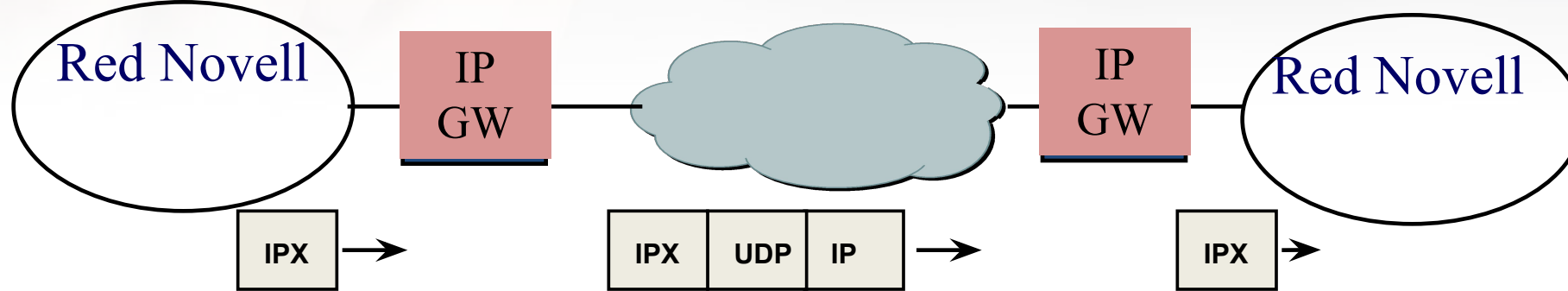
Túneles

Supongamos que existen dos LAN Novell distantes que queremos conectar a través de Internet.



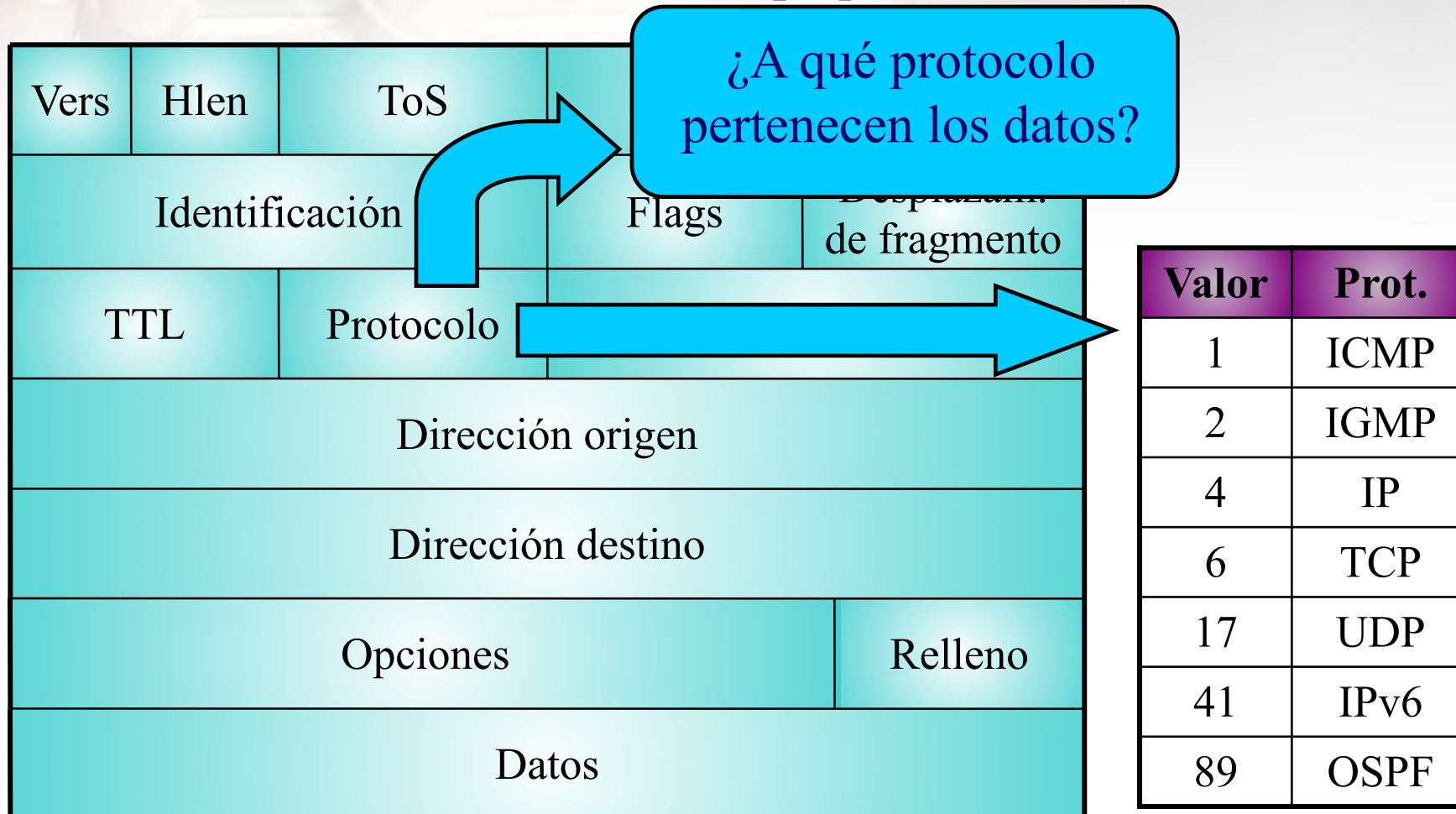
Túneles

Solución: Encapsular los paquetes IPX en paquetes UDP



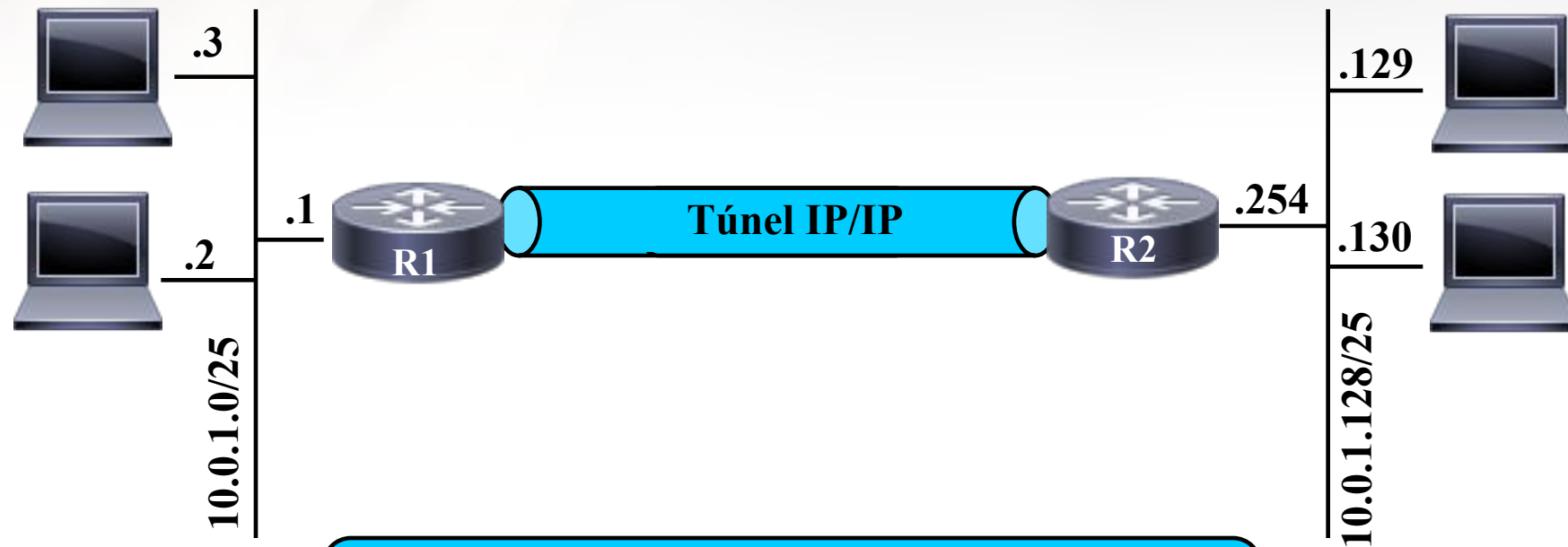
Túneles IP-en-IP

Recordemos el paquete IP



Túneles IP-en-IP

Supongamos que tenemos dos redes IP y queremos enviar tráfico desde una hacia otra como si estuvieran conectadas directamente.



¿Podría resolverlo utilizando NAT?

Túneles IP-en-IP

Cómo configurar el túnel en Linux

En R1 (IP privada: 10.0.1.1, IP pública: 24.10.11.12)

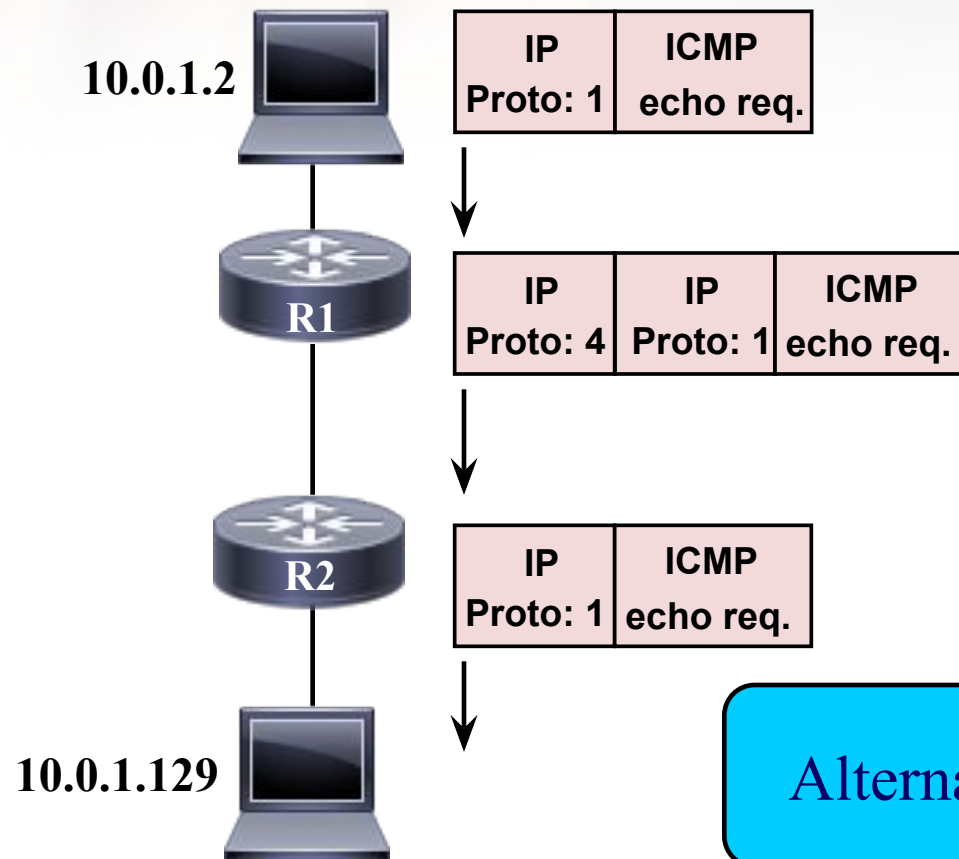
```
$ ifconfig tunl0 10.0.1.1 pointopoint 24.13.14.15  
$ route add -net 10.0.1.128 netmask 255.255.255.128 dev tunl0
```

En R2 (IP privada: 10.0.1.254, IP pública: 24.13.14.15)

```
$ ifconfig tunl0 10.0.1.254 pointopoint 24.10.11.12  
$ route add -net 10.0.1.0 netmask 255.255.255.128 dev tunl0
```


Túneles IP-en-IP

¿Cuál sería el flujo de paquetes si el host 10.0.1.2 envía un ping al 10.0.1.129?



¿Seguridad?

Alternativa: protocolo PPP o IPSec

Material de lectura

The background features a conceptual illustration. At the top, a hand holds a magnifying glass, focusing on a central point. Behind this, a network of nodes and lines is visible, with labels such as 'SEARCH', 'CONTENT', 'RESOURCE', and 'WEBSITE'. The overall theme is digital research or information management.

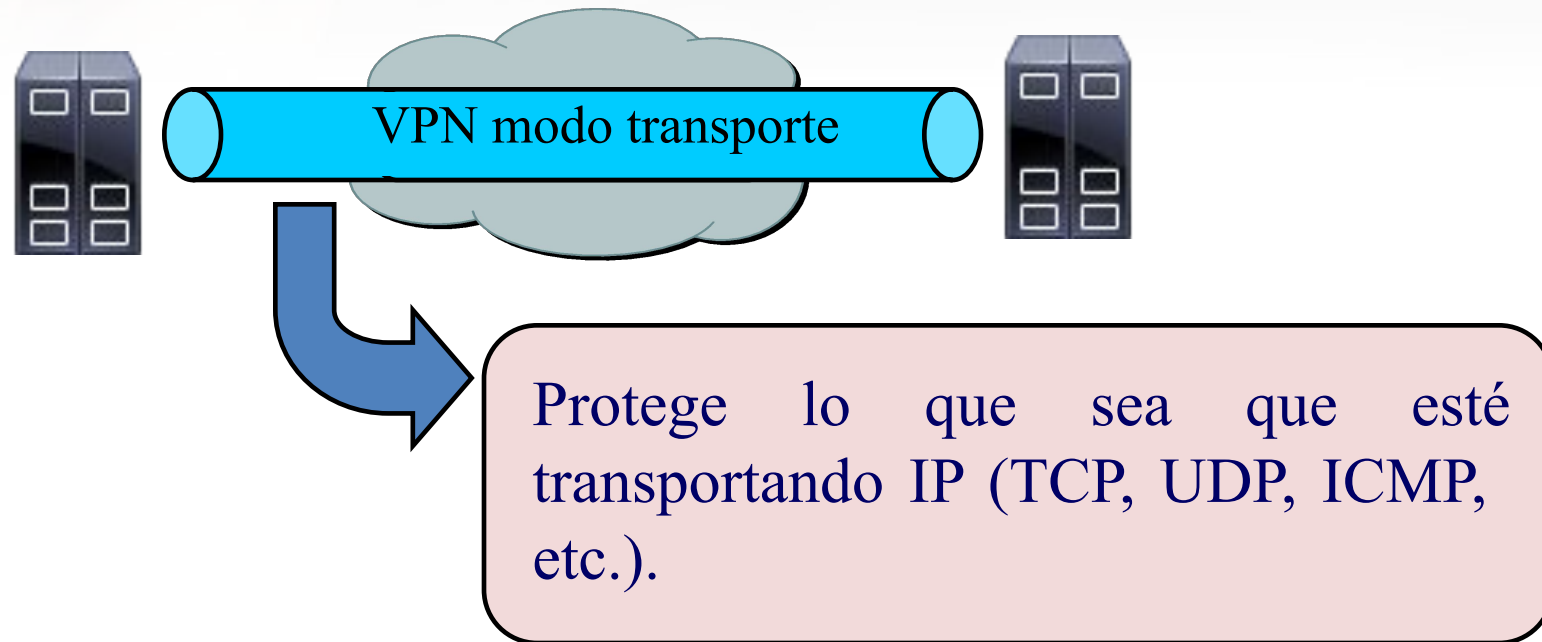
Capítulos 4.1 a 4.4 inclusive de la bibliografía

El objetivo de IPSec es proveer servicios de seguridad para los paquetes IP en el nivel de red.

- ❑ Estos servicios incluyen:
 - ❑ Control de acceso
 - ❑ Integridad de datos
 - ❑ Autenticación
 - ❑ Confidencialidad de datos
- ❑ IPSec trabaja en dos modos:
 - ❑ Túnel
 - ❑ Transporte

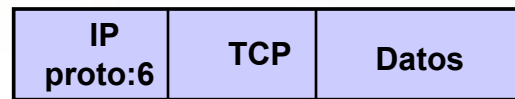
IPSec: modo transporte

Es usado entre dos hosts en forma directa. No puede ser usado para conectar dos redes o un host a una red.

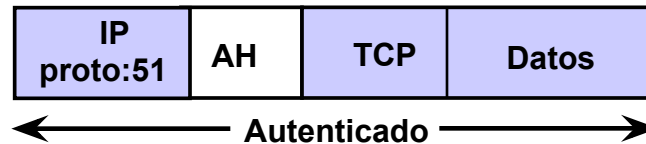


IPSec: modo transporte

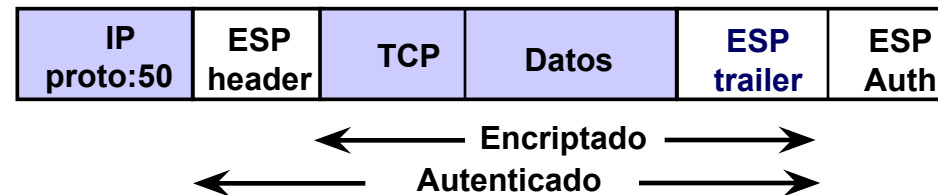
En este modo se agrega un header AH (*Authentication Header*) o ESP (*Encapsulating Security Payload*)



Paquete original



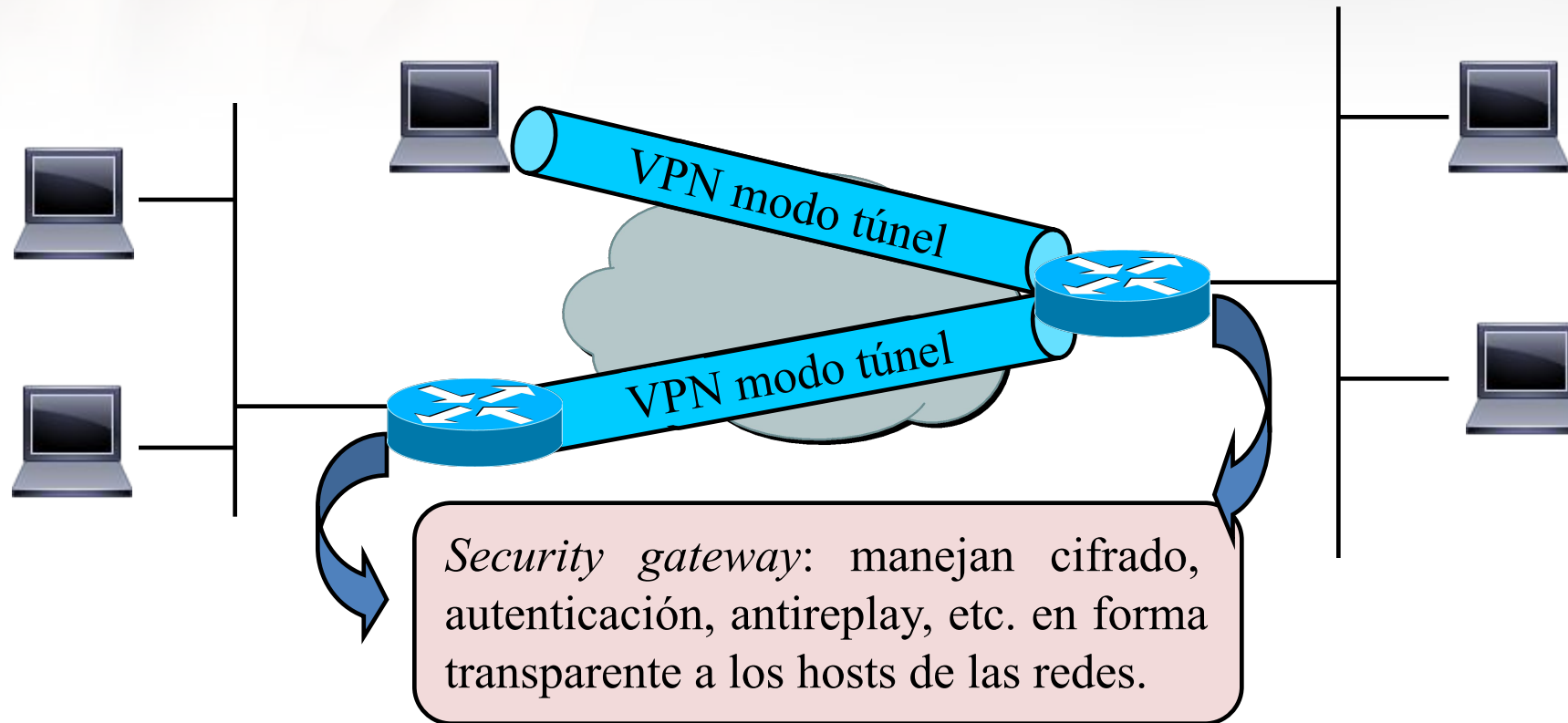
Autenticado



Autenticado y
encriptado

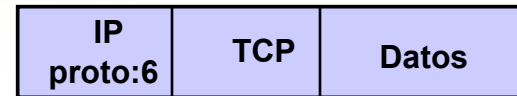
IPSec: modo túnel

En modo túnel es posible formar una VPN con dos redes o un host y una red.

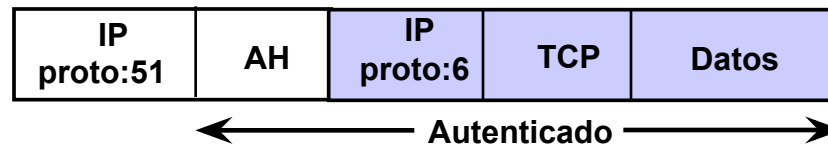


IPSec: modo túnel

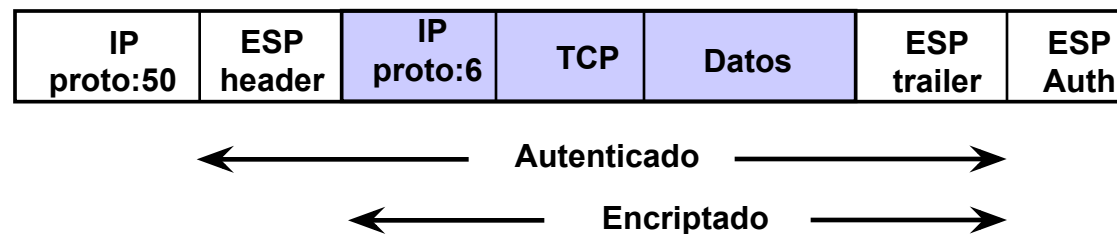
El paquete IP original es encapsulado en otro datagrama IP, insertando un header (AH o ESP)



Paquete original



Autenticado



Autenticado y
encriptado

IpTables e Ip6Tables



- Filtrado de paquetes y NAT
- Configurar, mantener e inspeccionar las tablas usadas por el kernel para filtrar paquetes
- Un paquete va pasando por distintas tablas, las cuales a su vez contienen una o más listas

IPTables

Hay 5 tablas independientes que utiliza ipTables:

- ❑ **raw**: filtra los paquetes antes que otras tablas. Se utiliza principalmente para configurar exenciones de seguimiento de conexiones. Dos cadenas:
 - ❑ PREROUTING:
 - ❑ OUTPUT
- ❑ **filter**: es la usada por defecto. Cadenas:
 - ❑ INPUT: tráfico entrante
 - ❑ OUTPUT: tráfico saliente generado localmente
 - ❑ FORWARD: tráfico enrutado

```
~$ iptables -A INPUT -p tcp --dport 17500 -j REJECT  
--reject-with icmp-port-unreachable
```


IPTables

Hay 5 tablas independientes que utiliza ipTables:

- ❑ **nat:** es consultada cuando un paquete crea una nueva conexión. Se usa para traducir direcciones de red y puertos. No se debe usar para reglas de filtrado. Cadenas:
 - ❑ PREROUTING: cambia paquetes antes de ser “ruteados”
 - ❑ POSTROUTING: cambia paquetes luego de ser “ruteados”
 - ❑ OUTPUT: aplica NAT a paquetes generados localmente
- ❑ **Acciones:**
 - ❑ DNAT
 - ❑ SNAT
 - ❑ MASQUERADE
 - ❑ REDIRECT

Tabla nat

Utilizada para traducir el IP origen o destino de una secuencia de paquetes.

Posibles *targets*:

- ◆ DNAT
- ◆ SNAT
- ◆ MASQUERADE

```
~$ iptables -t nat -A PREROUTING -p tcp --dport 80  
-i eth1 -j DNAT --to 10.0.1.80:8080
```

IpTables: Ejemplos

Listar reglas y ver tabla NAT actual

```
~$ iptables -t nat -L  
~$ netstat-nat -n
```

Aplicar SNAT (eth1: interfaz WAN)

```
~$ iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

```
~$ iptables -t nat -A POSTROUTING -s 192.168.1.0/24  
-o eth1 -j SNAT --to 20.28.101.14
```

```
~$ iptables -t nat -A POSTROUTING -p tcp  
-o eth1 -j SNAT --to 20.28.101.14:1-1023
```

Ejemplo

Sea el siguiente escenario



Ejemplo

```
# Reenviar todos los paquetes de eth1 (red interna)
# a eth0 (Internet).
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT

# Habilitar SNAT para IP pública fija:
iptables -t nat -A POSTROUTING -o eth0 -j SNAT
    -to-source 24.232.138.143

# No aceptar IP Spoofing
iptables -A INPUT -i eth0 -s 24.232.138.143/32 -j DROP
iptables -A INPUT -i eth0 -s 192.168.0.0/24 -j DROP
iptables -A INPUT -i eth0 -s loopback/8 -j DROP
```


Ejemplo

Aceptamos lo que venga de localhost

```
iptables -A INPUT -i lo -j ACCEPT
```

Implementamos Web Server en un host local

```
iptables -t nat -A PREROUTING -d 24.232.138.143  
-p tcp -dport 80 -j DNAT  
--to-destination 192.168.2.3
```