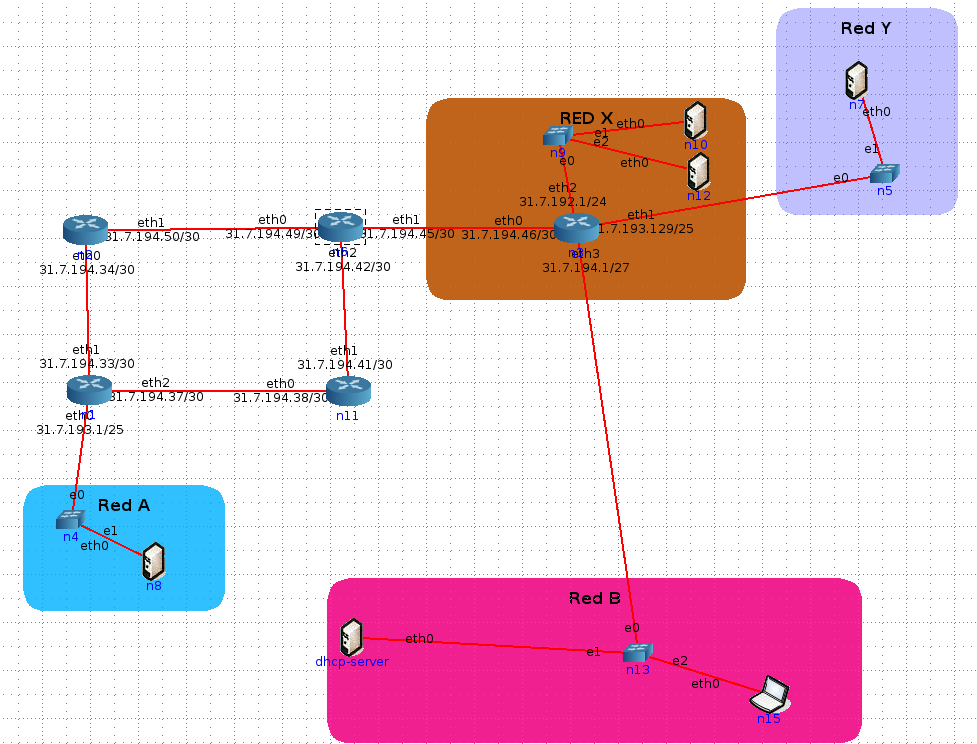
**Trabajo de Laboratorio Grupo U**

*EMANUEL EVARISTO BASTONS*

*SERGIO RAMON DE LUCA*

*JUAN PABLO EFSTATHOPOULOS*



**1. Utilizando topologia-IP.imn y dado el bloque IPv4 31.7.192.0/19 asignado resolver.**

**La red A tiene 70 hosts**

**La red X tiene 150 hosts.**

**La red B cuenta con 20 hosts**

**La red Y tiene 35 hosts y se espera un crecimiento maximo de 30 hosts.**

**Los bloques IP asignados en los enlaces entre routers podran ajustarse a desperdiciar pocas direcciones.**

**Es importante utilizar VLSM?**

**Ordeno las cantidades de mayor a menor.**

**A partir de la ip 31.7.192.0/19**

Para la red **X de 150** hosts, se necesitan 8 bits de la parte de hosts. por lo tanto nos quedaría:

Red

**31.7.192.0 /24**

**Rango**

**31.7.192.1 / 24-------------------------->31.7.192..254 /24**

Para la red **A de 70** + 20 = 90 hosts se necesitan 7 bits.

Nos queda

**31.7.193.0 / 25**

**Rango**

**31.7.193.1-------------------------------->31.7.193.126 / 25**

La red Y tiene 35 hosts, y se espera un crecimiento máximo de 30 hosts.

Por lo que **30+35=65 hosts.**

Para la red Y 65 de hosts se necesitan 7 bits. Nos queda:

**31.7.193.128 /25**

**Rango**

**31.7.193.129 / 25------------------------------>31.7.193.254 / 25**

**Para la red B de 20 hosts se necesitan 5 bits:**

**31.7.194.0 / 27**

**Rango**

**31.7.194.1 / 27---------------------------------->31.7.194.30 / 27**

**Entre los routers tenemos 5 de redes de dos direcciones cada una.**

**Para 2 direcciones se necesitan 2 bits.**

**2 al cuadrado - 2 = 2 direcciones.**

**Por lo tanto las redes entre routers quedan:**

**Red 1**

**31.7.194.32 / 30**

**Rango**

**31.7.194.33 / 30-------------------------------------------------->31.7.194.34 / 30**

**Red 2**

**31.7.194.36 / 30**

**Rango**

**31.7.194.37 / 30--------------------------------------------------->31.7.194.38 / 30**

**Red 3**

**31.7.194.40/ 30**

**Rango**

**31.7.194.41 / 30----------------------------------------------------->31.7.194.42 / 30**

**Red 4**

**31.7.194.44 / 30**

**Rango**

**31.7.194.45 / 30------------------------------------------------------> 31.7.194.46 / 30**

**Red 5**

**31.7.194.48 / 30**

**Rango**

**31.7.194.49 / 30 ------------------------------------------------------->31.7.194.50 / 30**

**Es importante utilizar VLSM?**

Si, por que se desperdician menos direcciones ip para la parte de hosts.

**b) Asigne direcciones IP en los equipos de la topologıa segun el plan anterior.**

**Los equipos quedaron configurados:**

**Hosts de Red A**

**n8 = addr:31.7.193.2 Bcast:31.7.193.127 Mask:255.255.255.128**

**Hosts de Red B**

**dhcp-server = addr:31.7.194.2 Bcast:31.7.194.31 Mask:255.255.255.224**

**n15 = addr:31.7.194.3 Bcast:31.7.194.31 Mask:255.255.255.224**

**Hosts de Red X**

**n10 = addr:31.7.192.2 Bcast:31.7.192.255 Mask:255.255.255.0**

**n12 = addr:31.7.192.3 Bcast:31.7.192.255 Mask:255.255.255.0**

**Hosts de Red Y**

**n7 = inet addr:31.7.193.130 Bcast:31.7.193.255 Mask:255.255.255.128**

**Routers:**

**Router n1**

**eth0 inet addr:31.7.193.1**

**eth1 inet addr:31.7.194.33**

**eth2 inet addr:31.7.194.37**

**Router n2:**

**eth0 inet addr:31.7.194.34**

**eth1 inet addr:31.7.194.50**

**Router n3:**

**eth0 inet addr:31.7.194.46**

**eth1 inet addr:31.7.193.129**

**eth2 inet addr:31.7.192.1**

**eth3 inet addr:31.7.194.1**

**Router n6:**

**eth0 inet addr:31.7.194.49**

**eth1 inet addr:31.7.194.45**

**eth2 inet addr:31.7.194.42**

**Router n11:**

**eth0 inet addr:31.7.194.38**

**eth1 inet addr:31.7.194.41**

**Tablas de Ruta configuradas:**

**Hosts n8 de Red A**

**root@n8:/tmp/pycore.44968/n8.conf# netstat -nr**

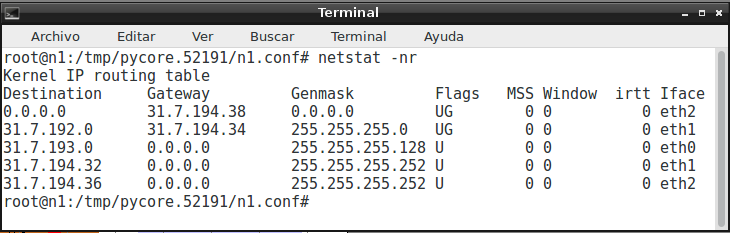
**Kernel IP routing table**

**Destination Gateway Genmask Flags MSS Window irtt Iface**

**0.0.0.0 31.7.193.1 0.0.0.0 UG 0 0 0 eth0**

**31.7.193.0 0.0.0.0 255.255.255.128 U 0 0 0 eth0**

**Router n1:**

****

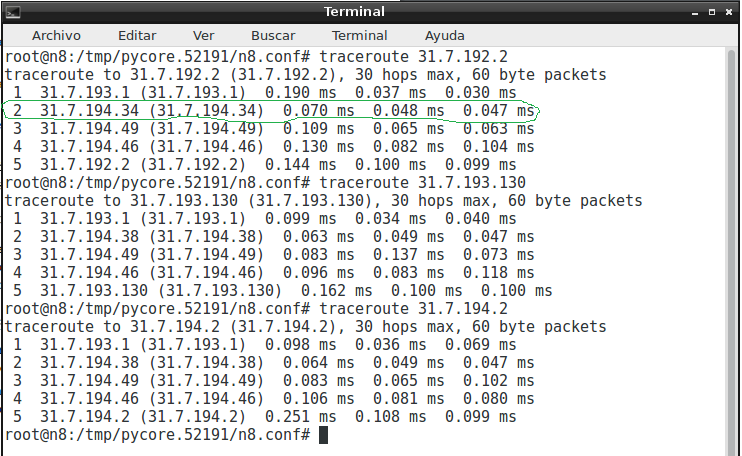
**n1 deberá optar por el enlace verde solamente para rutear el tráfico dirigido a la Red X.**

**La red X es**

**31.7.192.0 /24**

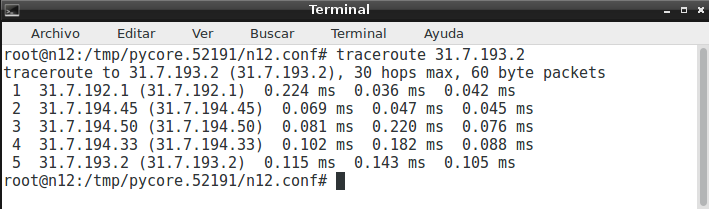
**Para que el trafico sea por el enlace verde:**

Traceroute desde host n1 hasta la red X

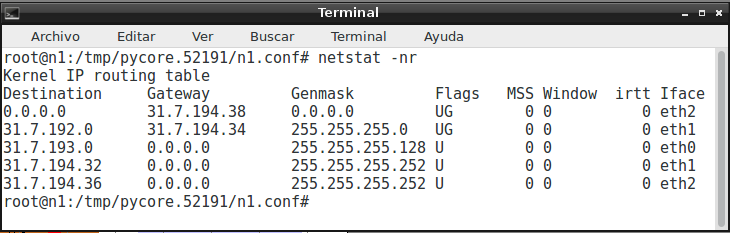
****

También hicimos traceroute hacia las otras redes para ver que no pasen por el enlace verde.

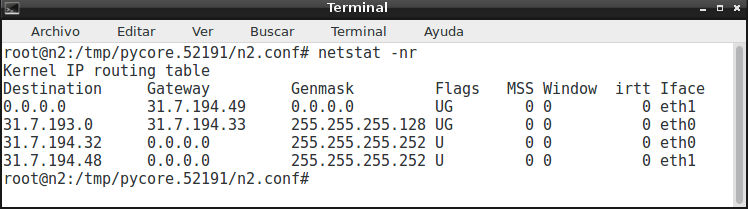
**Traceroute desde el host de la red X a n1**

****

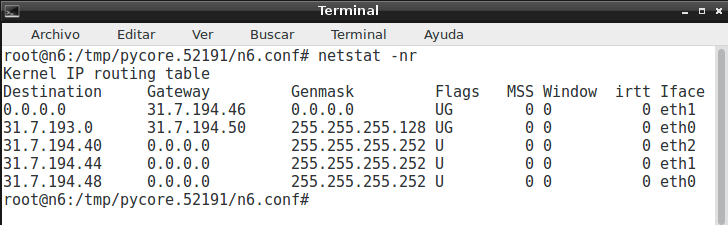
**Tabla Router 1**

****

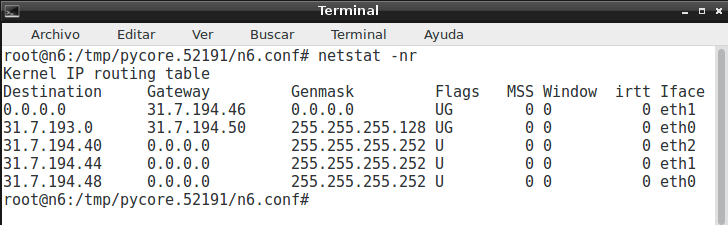
**Tabla Router n2:**

****

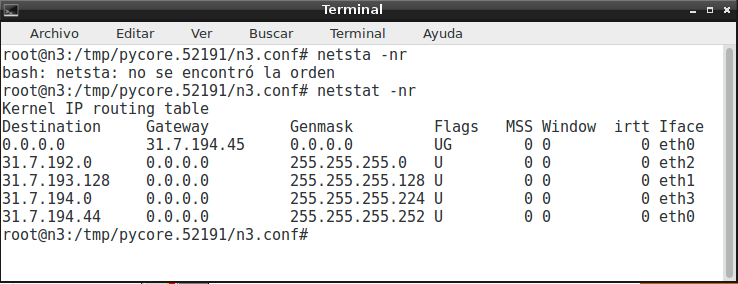
**Tabla Router n3**

****

**Tabla Router n6**

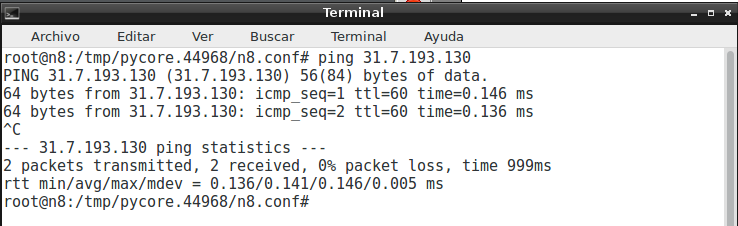
****

**Tabla Router n11**

****

**Utilizando la herramienta ping(8), verifique conectividad entre los hosts pertenecientes a las diferentes redes de usuarios.**

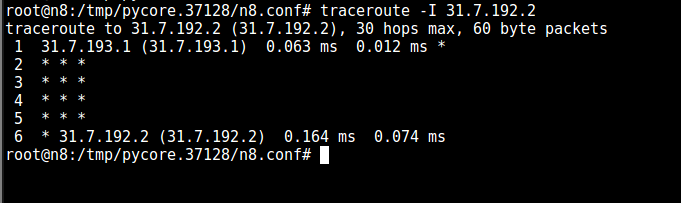
**Ping desde n8 (Red A) a n7 (Red Y)**

****

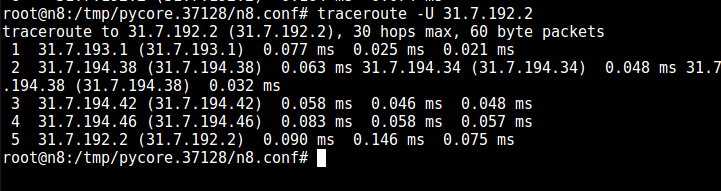
**2. TTL Adjunte capturas de tráfico para cada uno de los incisos:**

**a) Utilizando el comando traceroute(1)/mtr(8), realice una traza entre el host n8 y n10, tanto utilizando UDP como ICMP. Que diferencias tiene cada metodo y en que casos utilizarıa cada uno?**

Comando **traceroute** utilizando ICMP

****

Comando **traceroute** utilizando UDP



Con UDP pueden verse las direcciones IP de los routers intermedios, a diferencia de

ICMP.

**b) Realice un ping entre n8 y n5 y determine el valor inicial del campo TTL capturando trafico en la interfaz eth0 del host n8**

*\*Aclaración, este ejercicio fue realizado entre n8 y n7 ya que n5 es un Switch y no tiene dirección ip asignada.\**

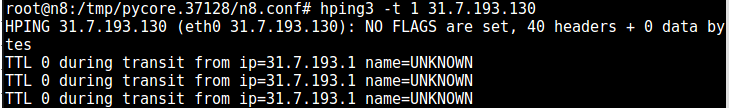


El TTL Inicial es igual a 64.

**c) A través de la capturas de tráfico, determine en que momento el router decrementa el valor del TTL.**

El router le decrementa el valor TTL cuando la dirección de destino no está dentro del router.

**d) Utilizando la herramienta para enviar mensajes ICMP con la opcion -t desde n8 envıe un datagrama a n7 con TTL=1. Que mensaje recibe? Por que?**



El mensaje que recibe es el mostrado en la captura. el campo de nombre es solo la dirección numérica resuelta a un nombre (una solicitud PTR de dns) o DESCONOCIDA si la resolución falla. Por lo tanto nos dice que la resolución de la misma falló.

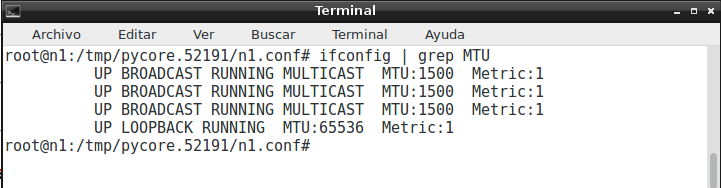
**3. Fragmentación:**

**a) Cambiando los MTU y enviando tráfico desde un host de la parte “A" a la “B" ver de forzar que los routers fragmenten.**

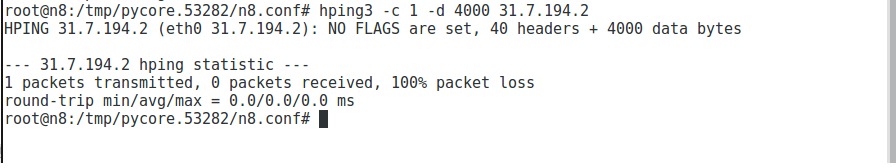
**b) Donde se fragmenta y donde se re-ensambla ? Qué campos son cambiados durante la fragmentación, Como se identifican los fragmentos de un mismo segmento ?**

**c) Considera que la fragmentación es necesaria o puede evitarse? Justifique.**

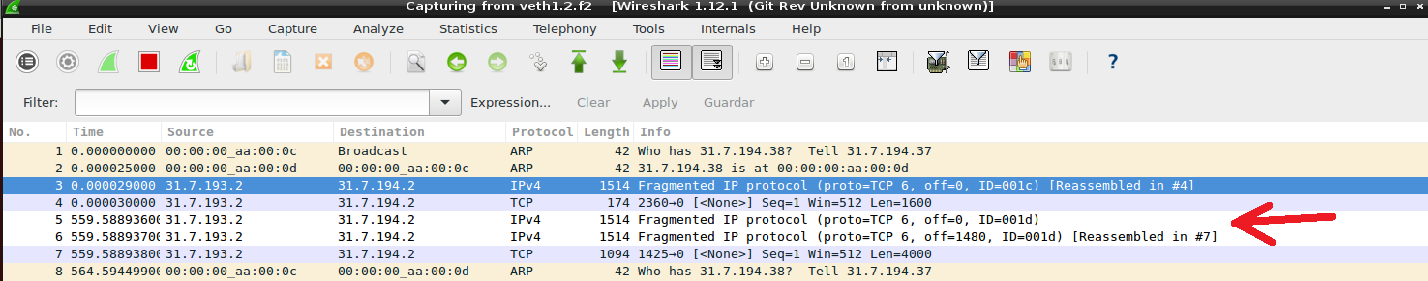
Los MTU de los routers de la simulación están por defecto en 1500.



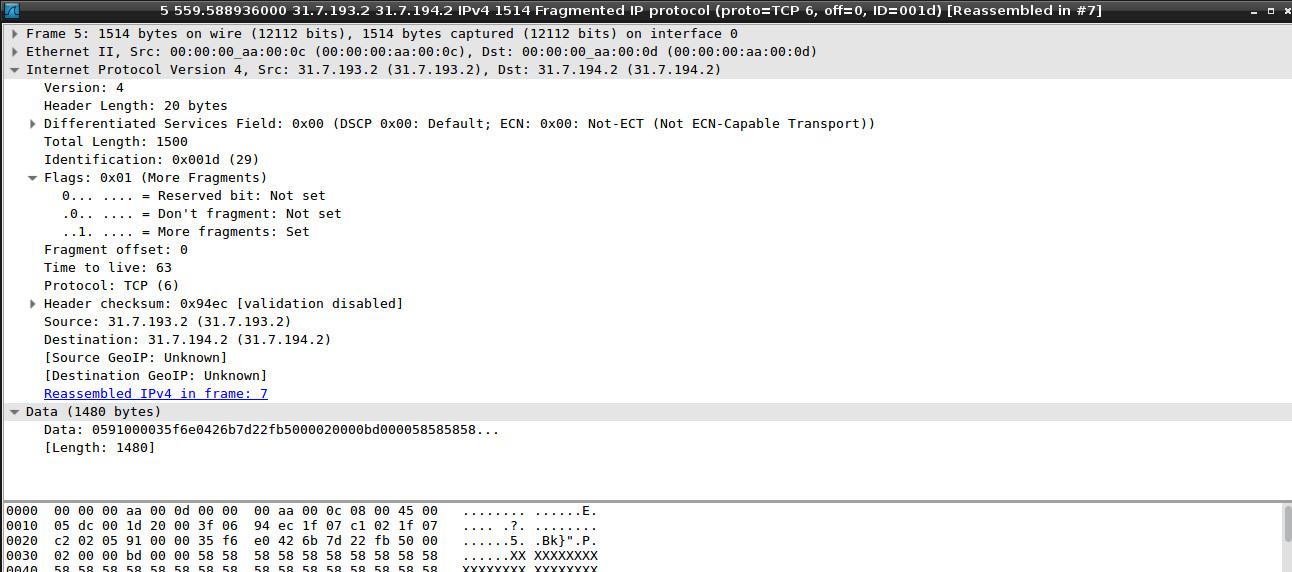
**En el hosts de la red A enviamos un datagrama superior al MTU, en este caso 4000 bytes.**

****

**Capturamos el tráfico en la interface eth2 del Router n1 y se ve como se fragmenta:**

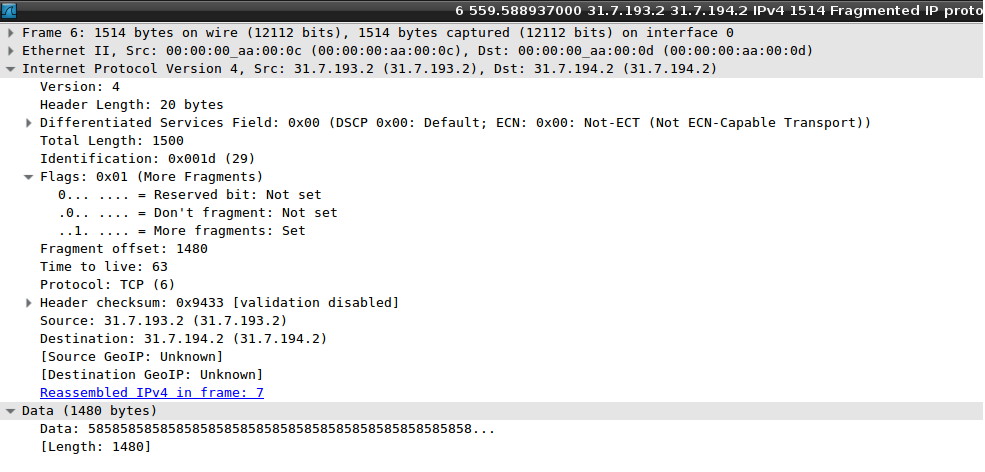
****

**Como enviamos un datagrama ip de 4000 bytes, el router lo envio en 2 fragmentos:**

****

**Se modifican los flags (More Fragments)**

**Fragmento 2:**

****

Los fragmentos se identifican por el campo Fragment offset.

Cuando la capa IP obtiene un datagrama para enviar, si el tamaño del datagrama es más grande que la [MTU](https://es.wikipedia.org/wiki/Unidad_m%C3%A1xima_de_transferencia) por esta capa, la capa IP divide el datagrama disponible en varios datagramas más pequeños. Este proceso es denotado como *fragmentación*. La fragmentación puede tener lugar en el emisor inicial o en los routers que están entre el emisor y el receptor. Si un datagrama es fragmentado, no será ensamblado(desfragmentado) de nuevo hasta llegar al receptor. (Excepción: Un *reassembly* de [Cortafuegos](https://es.wikipedia.org/wiki/Cortafuegos_(inform%C3%A1tica)) intercalados antes de transmitir los datos) Si es necesario, un paquete ya fragmentado puede ser fragmentado otra vez (por ejemplo durante un cambio de método de transmisión).

Cada fragmento del datagrama original obtiene en vez del *datagram header* (cabecera de datagrama) del paquete original un denominado *fragment header* (cabecera de fragmento) que contiene entre otras cosas el [offset](https://es.wikipedia.org/wiki/Offset_(inform%C3%A1tica)) que indica la porción de datos enviado en este paquete en relación al paquete original. El *fragment offset* (13 bit en el *IP header*) está indicado en bloques de 64 bits. Todos los fragmentos menos el último tienen el *more fragments flag* con valor "1". El campo de longitud en el *IP header* contiene la longitud del fragmento, y se calcula la [suma de verificación](https://es.wikipedia.org/wiki/Suma_de_verificaci%C3%B3n) para cada fragmento apretadamente, mientras que el resto del header corresponde al header original.

El receptor es el responsable de reensamblar todos los fragmentos en el orden correcto para obtener el datagrama original y entregarlos al protocolo de nivel superior. El reensamblaje se espera que ocurra en el equipo receptor, pero en la práctica puede ocurrir también en routers intermedios, por ejemplo, [NAT](https://es.wikipedia.org/wiki/Traducci%C3%B3n_de_direcciones_de_red) puede necesitar reensamblar fragmentos para traducir flujos de datos, e.g. el protocolo de control de FTP, como se describe en el RFP 2993