Redes de computadoras

Grupo: 7003 Semestre: 2019-1

Paulo Contreras Flores

Virgilio Castro Rendón

Daniel Campuzano Barajas

Transporte

Capa de Transporte

- Provee el servicio de transferencia de datos entre los *host* de manera transparente, confiable y eficiente. [3]

- Operan los protocolos encargados de establecer las comunicaciones de punto a punto (*end to end*). [3]

- Da confiabilidad en la conexión punto a punto, y permite el aprovechamiento de los recursos de red disponibles. [3]

Maximum Transmission Unit

- Existe una medida que limita la cantidad de información que se puede enviar a través de un medio físico o protocolo, se le conoce como Máxima Unidad de Transferencia o MTU (*Maximum Transmission Unit*).

- Debido al MTU es necesario que la información se envíe en cantidades menores que dicho valor, la capa de Transporte se encarga de llevar un control sobre cada parte de información que se envía, tanto de dividirla para el envío, como de rearmarla ordenadamente en el destino.

Maximum Transmission Unit

- De esta forma también se evita la congestión de los *host* de destino, que pudiera presentarse por recibir una cantidad de información mayor a la que puedan manejar.

Checksum

- Esta capa también se encarga de verificar la integridad de los paquetes, esto se hace a través de una suma de verificación o *checksum*. [3]

- Es calculado por el *host* origen, y este valor de *checksum* es concatenado al segmento a enviar. [3]

- El *host* destino aplica el mismo procedimiento que el *host* origen y verifica que su resultado coincida con el valor de *checksum* enviado por el origen. [3]

Checksum

- Si los valores de *checksum* coinciden significa que el mensaje llegó de manera íntegra, sino el mensaje deberá ser reenviado. [3]

Multiplexación

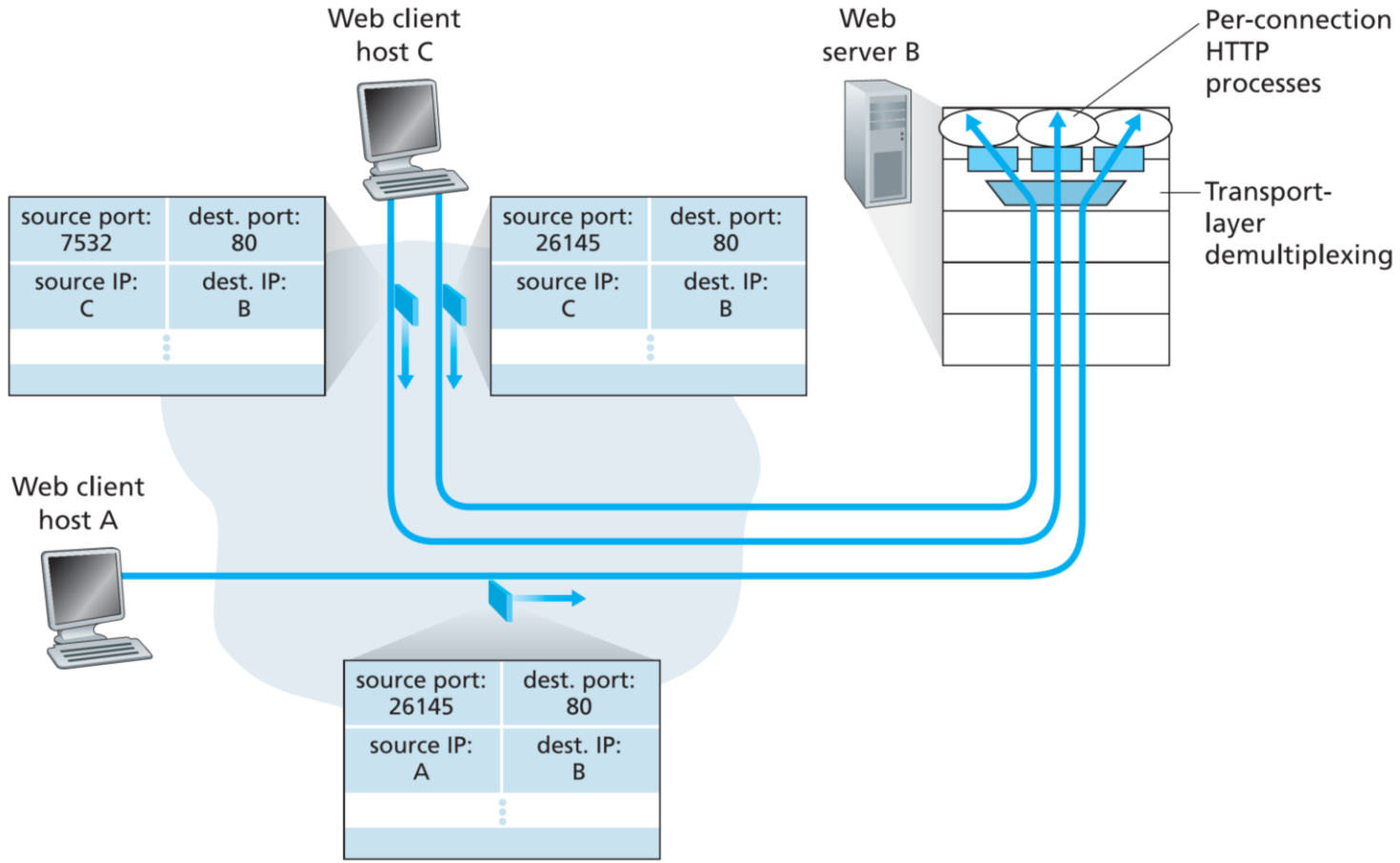
- La multiplexación es el proceso mediante el cual un mismo *host* puede establecer diferentes comunicaciones con otros *host* de manera simultánea sin entorpecer el flujo de información de las conexiones. Esto se logra con el concepto de puertos lógicos. [3]
- Un puerto es un canal de comunicación. Se puede tener una misma dirección IP en un *host*, pero puede estar comunicándose con distintos host a la vez, a través de diferentes puertos. [3]

Capa de transporte - Multiplexación

- Dos diferentes servicios no pueden hacer uso del mismo puerto de forma simultánea. [3]
- Si un servicio se ha iniciado utilizando un puerto específico, este puerto no podrá ser utilizado por otro proceso hasta que el servicio inicial se haya cerrado.
- Los puertos están representados por un número de 16 bits.

$$PORTS = \{port \mid 0 \leq port < 2^{16}, port \in \mathbb{Z}\}$$

Capa de transporte - Multiplexación



Dos clientes usando el mismo puerto destino (80) para comunicarse con el mismo servidor Web. [1]

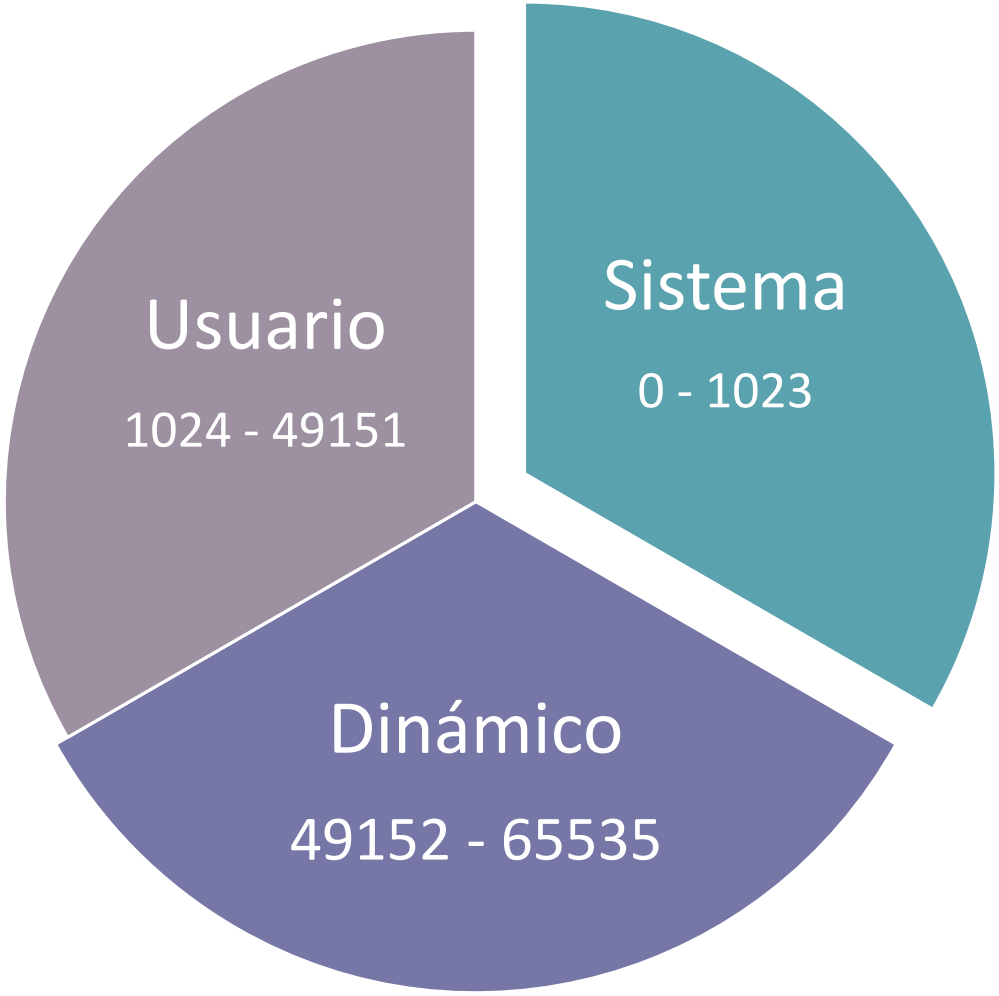
Capa de transporte - Multiplexación

- De acuerdo a la IANA existen tres rangos de puertos.

Rango	Categoría
0 - 1023	Puertos de sistema
1024 - 49151	Puertos de usuario
49152 - 65535	Puertos dinámicos o privados

Multiplexación

Puertos



Multiplexación

- 0 – 1023, Puertos de sistema. También llamados “bien conocidos” (*well-know ports*) son utilizados principalmente por procesos del sistema operativo (SO) y los servicios más comunes de redes. Generalmente el SO exige credenciales administrativas para poder abrir uno de estos puertos. [3]

- 1024 – 49151, Puertos de usuario. También llamados “puertos registrados” (*registered ports*), fueron registrados ante la IANA por fabricantes de software. Un usuario de un SO puede abrir un puerto en este rango sin la necesidad de credenciales administrativas. [3]

Multiplexación

- 49152 – 65535, Puertos dinámicos o privados. También llamados “puertos efímeros” (*ephemeral ports*), son utilizados por el SO que inicia una conexión, llamado cliente, al realizar una conexión hacia un equipo remoto. [3]

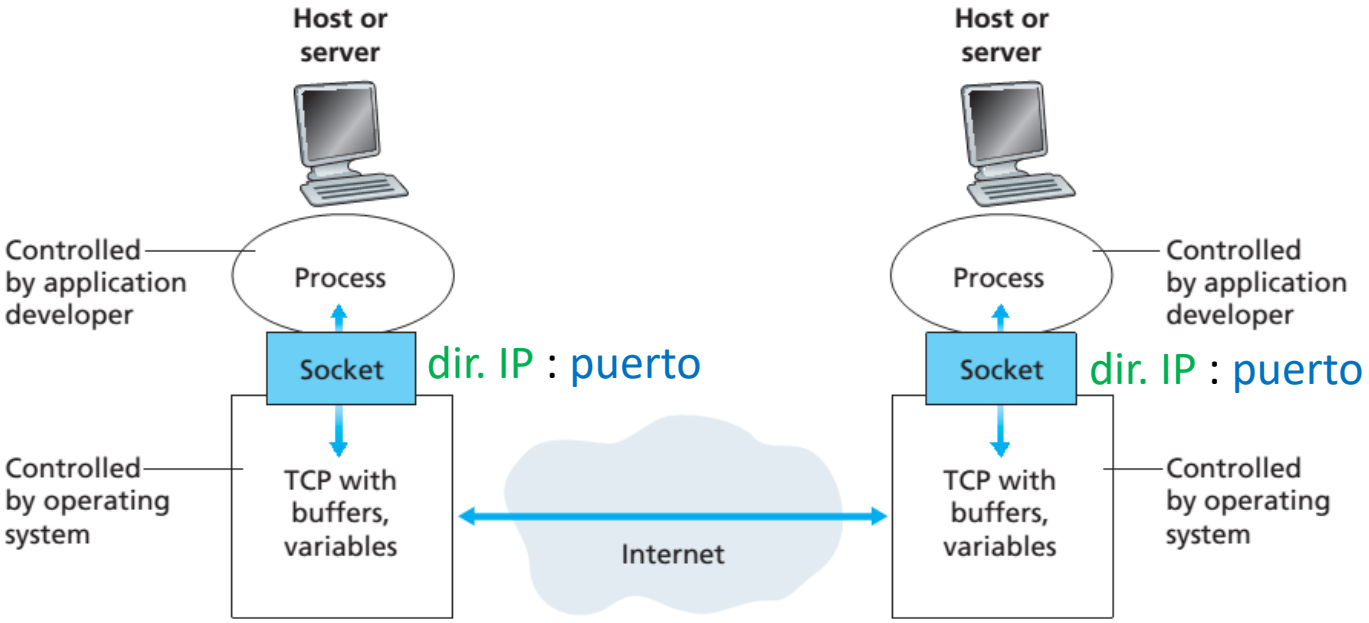
Multiplexación

- Todas las comunicaciones deben poseer un puerto origen y puerto destino. [3]
- Ejemplos de servicios y puertos:

Servicio de la capa de aplicación	Puerto
HTTP (*Hypertext Transfer Protocol*)	80
HTTPS (*Hypertext Transfer Protocol Secure*)	443
SMTP (*Simple Mail Transfer Protocol*)	25
MySQL	3306
Postgresql	5432
SQL Server	1433

Sockets

- Un socket es la forma a través de la cual una aplicación puede mandar y recibir datos.
- Se identifica de forma única por : dirección IP, protocolo de la capa de Transporte y puerto.



Orientado a Conexión

- Establece un camino lógico o virtual entre dos *host*, es confiable, resistente a las pérdidas y duplicación de información, la información es entregada a la capa de Aplicación en el mismo orden en el que se envía.

- El protocolo más común orientado a conexión es TCP (*Transmission Control Protocol*).

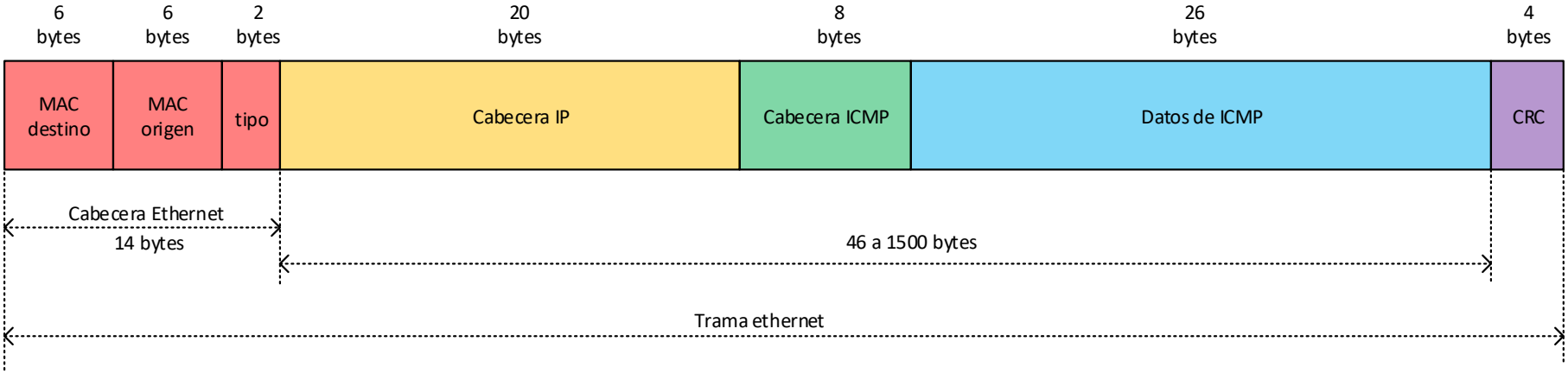
No orientado a conexión

- No es confiable en la entrega de los datos, puede haber pérdidas y duplicación de información, y puede llegar en distinto orden del que se envía.
- La comunicación es más rápida respecto al esquema Orientado a conexión.
- El protocolo más común orientado a conexión es UDP (*User Datagram Protocol*).

- Antes de continuar con la explicación del funcionamiento de los protocolos UDP y TCP se revisarán algunos conceptos de encabezados de los protocolos de Internet, y se relacionarán con los conceptos de la capa de transporte.

MTU y la trama Ethernet (IEEE 802.3)

- Tiene las direcciones MAC fuente y destino.
- El tipo de paquete encapsulado, IP, ARP, RARP, etc.
- Al final un campo de validación por CRC (Cyclic Redundancy Check) de 4 bytes, utilizado para verificar la existencia de tramas dañadas.



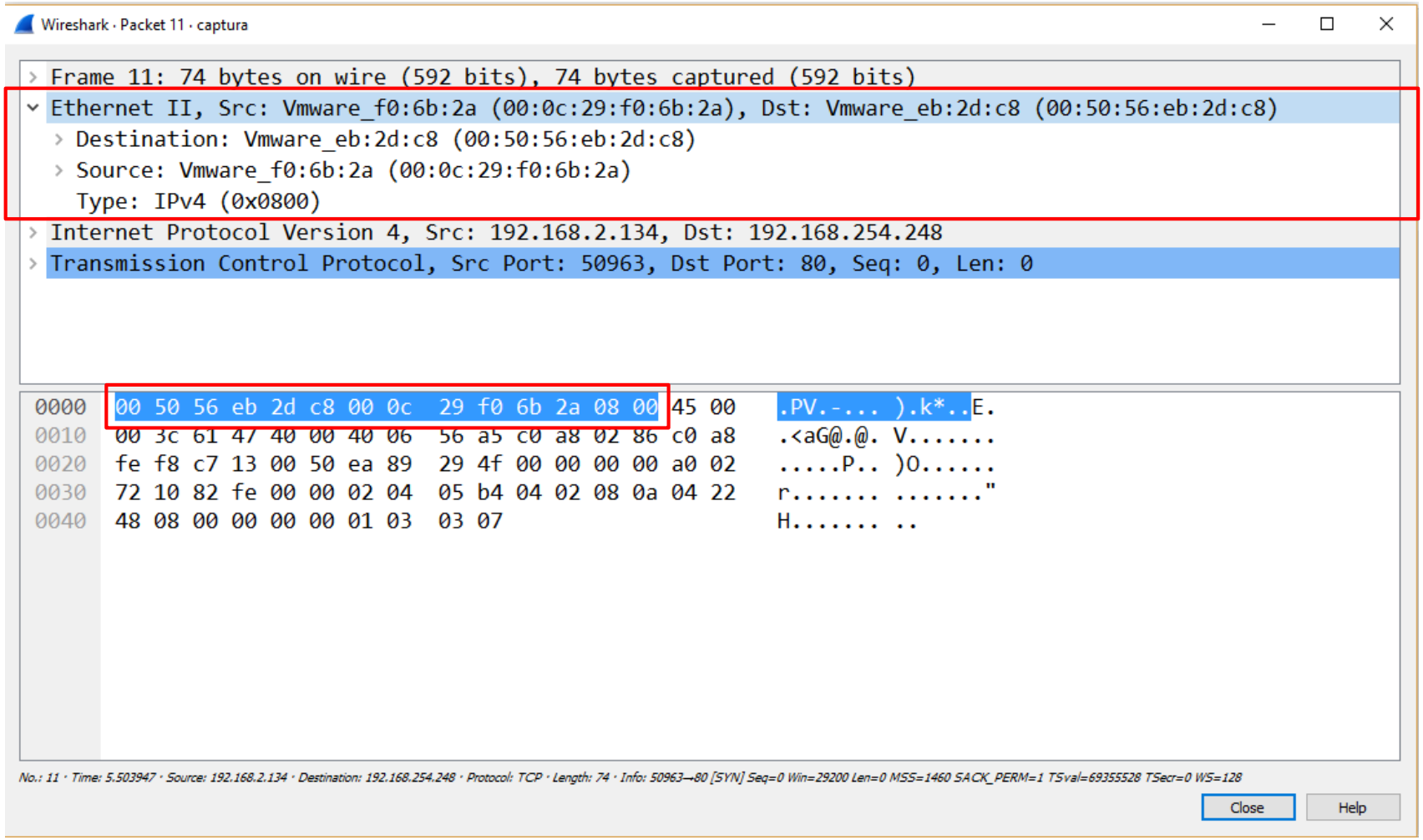
MTU y la trama Ethernet (IEEE 802.3)

- El tamaño de la cabecera de la trama Ethernet es de un total de 18 bytes (14 bytes + 4 bytes de CRC).

- El tamaño máximo para una trama de Ethernet es de 1518 bytes, lo cual deja un máximo de 1500 bytes para el protocolo encapsulado, por ejemplo IP. Este valor se conoce como MTU (*Maximum Transmission Unit*).

- El tamaño mínimo de un trama de Ethernet es de 64 bytes, lo cual deja un tamaño mínimo de paquete encapsulado de 46 bytes.

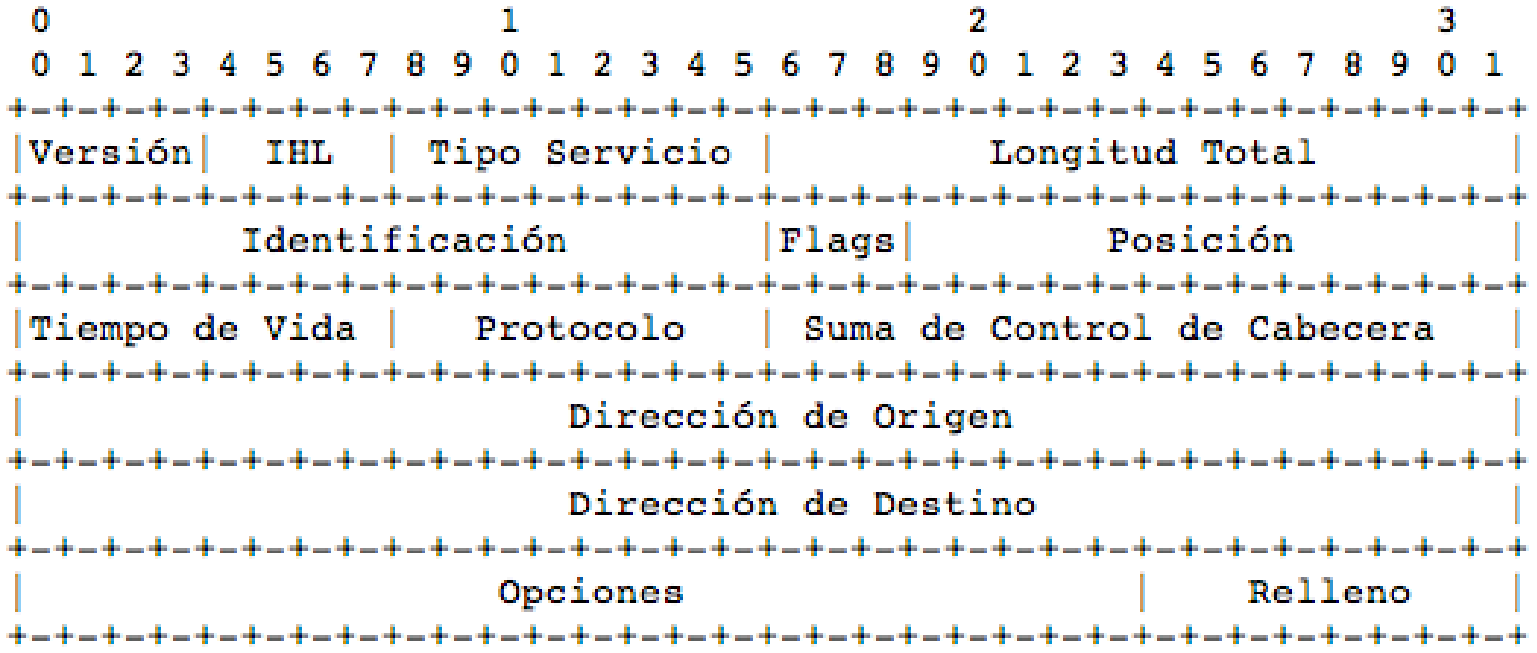
MTU y la trama Ethernet (IEEE 802.3)



Protocolo IP

- Requiere direcciones IP para la comunicación.
- La traducción de nombres de dominio a direcciones IP se efectúa a través del DNS.
- Las direcciones IP las lleva la cabecera IP.
- Los paquetes pueden ser enviados por diferentes rutas.
- Es un protocolo poco confiable, puesto que no garantiza la entrega del paquete. El paquete puede perderse o bloquearse y el protocolo IP no lo sabrá y no le preocupará. Esto lo resuelve TCP en la capa de Transporte.

Encabezado IP

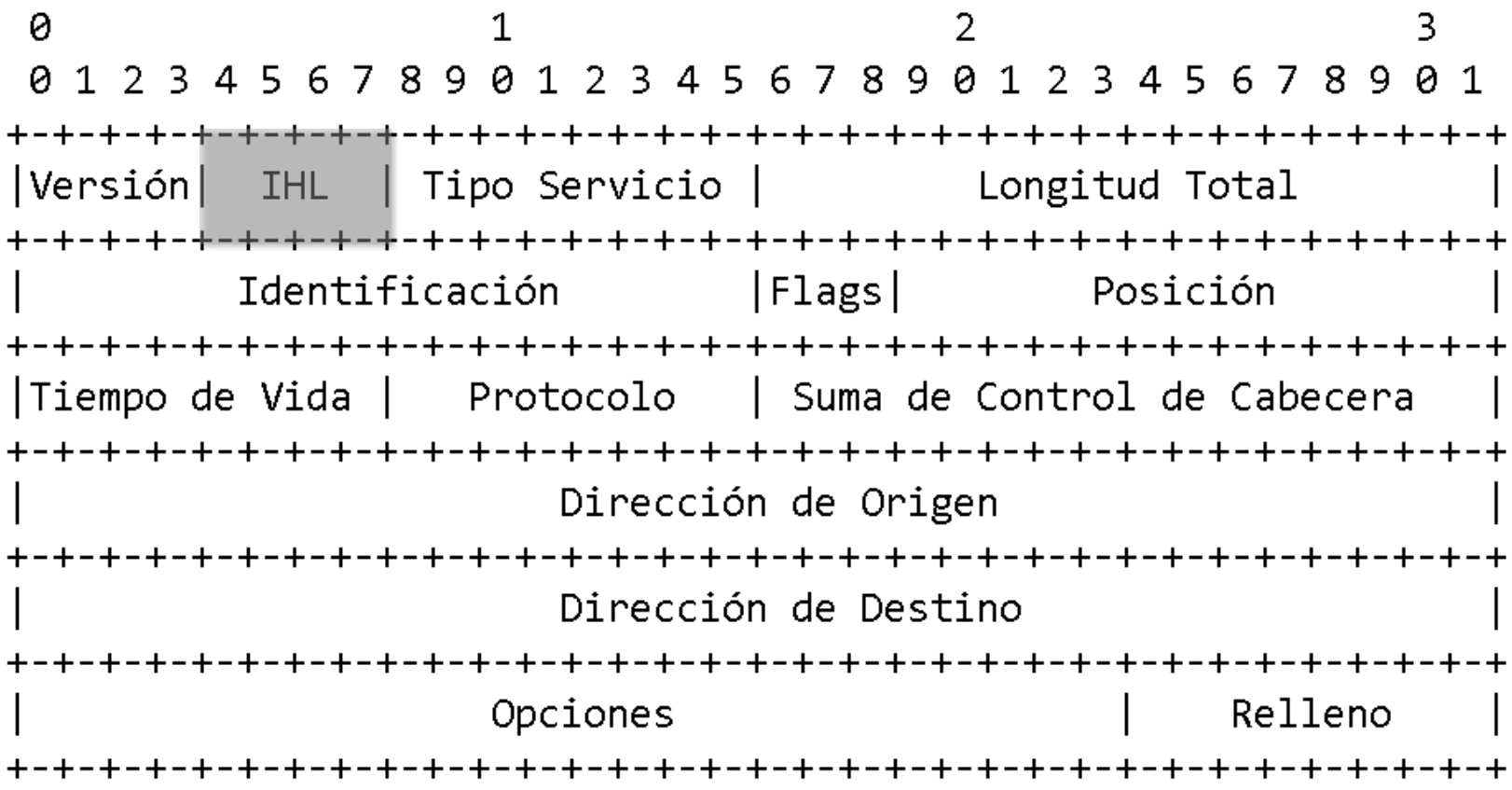


- Cada renglón tiene de 0 a 31 bits, 4 bytes por renglón.
- No todos los campos caen en la separación de bytes, algunos ocupan solo 4 bits de longitud, como el de la versión de IP o el campo de longitud de cabecera de 4 bits (IHL o IP Header Length).

Encabezado IP

- Otros campos son más grandes que un byte como sucede con la longitud total del paquete que es de 16 bits, es decir 2 bytes.
- Se tienen campos menores a 4 bits (nibble), como lo es el campo de banderas (Flags) de 3 bits. Cada una de estas banderas comúnmente ocupa un bit.

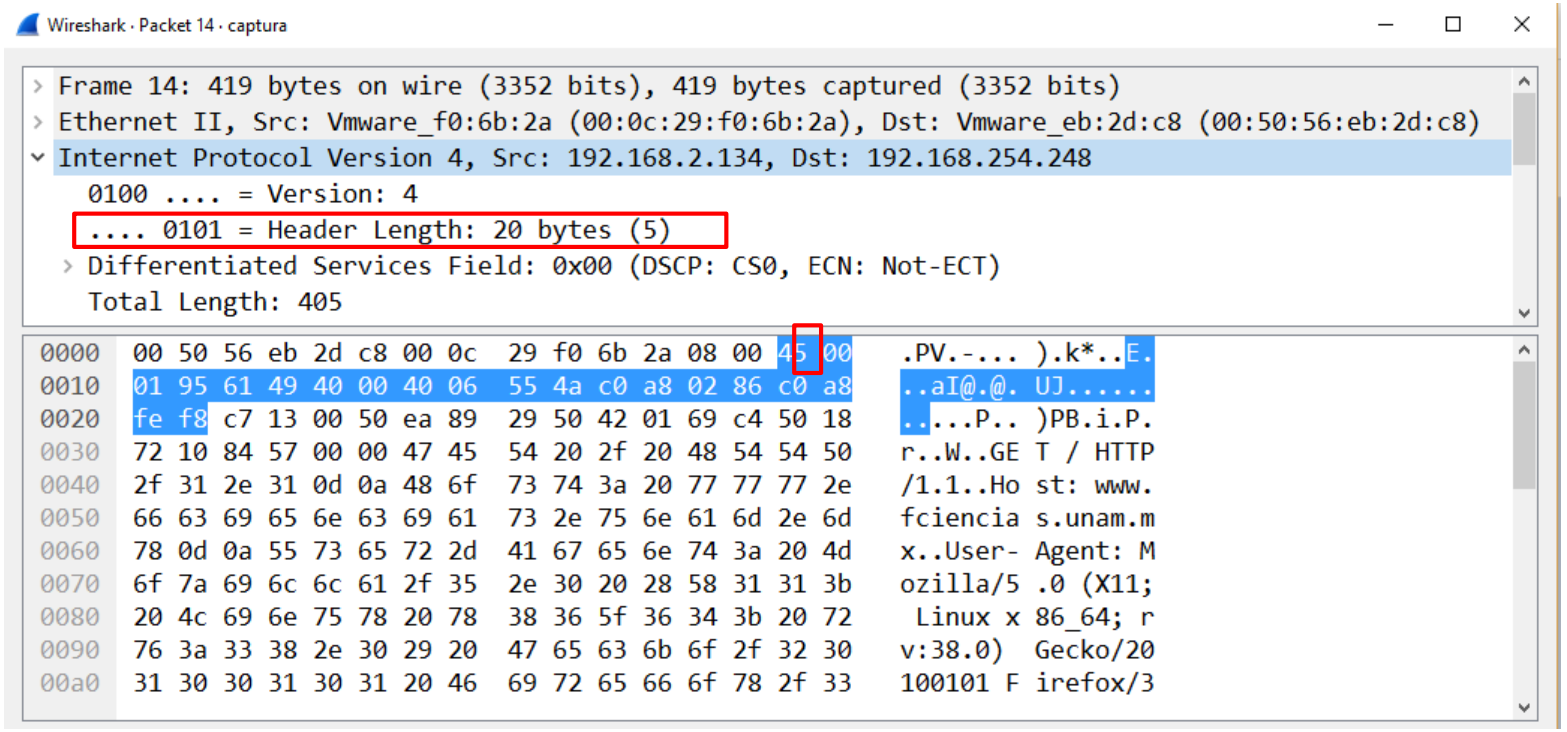
Longitud del encabezado IP



Longitud del encabezado IP

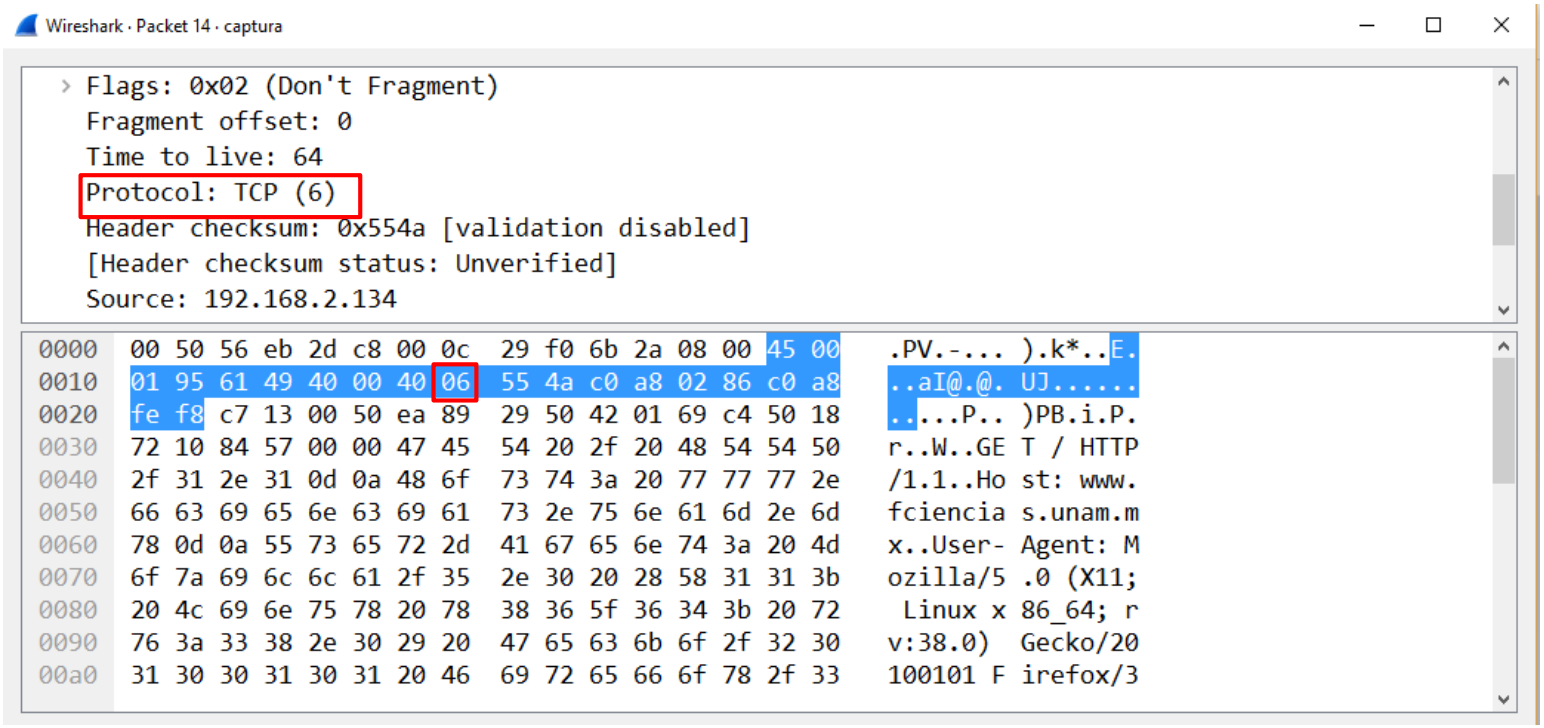
- Es necesario calcular el tamaño del encabezado IP para determinar con exactitud cuándo comienza la información encapsulada en este paquete.
- Para calcular el tamaño del encabezado de IP (IHL), se toma el valor de los 4 bits menos significativos del primer byte de un paquete de IP. De esta forma, se obtiene un número en hexadecimal, que representa el número de palabras (words) que contiene este paquete.

Longitud del encabezado IP



- Longitud de encabezado IP: 5 palabras
- $\frac{32\ bits}{1\ palabra} \cdot \frac{1\ byte}{8\ bits} \cdot 5\ palabras = 20\ bytes$

Protocolo encapsulado



- El número de protocolo de IP indica el protocolo que se incluye después de la cabecera de IP.
- Se encuentra en el 9o byte de la cabecera de IP.

Protocolo encapsulado

- Algunos protocolos
  - 0x01 ICMP
  - 0x02 IGMP
  - 0x06 TCP
  - 0x11 UDP

- La lista de protocolos se encuentra en http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

Checksum en IP

- La validación de IP se encuentra en los bits 10 y 11 del encabezado de IP.
- Esta validación de IP cubre todos los campos en el encabezado de IP únicamente.
- Las validaciones de protocolo embebidas, tales como UDP, TCP, o ICMP son validadas por el equipo de destino solamente. La validación de IP es hecha por cada router por donde pasa el paquete desde el origen hasta el destino, además de que es validado finalmente por el equipo destino.

Checksum en IP

- Si el valor calculado de la validación no concuerda con el que se encontró en el paquete, el paquete es silenciosamente descartado.
- No se intenta informar de ninguna forma a la fuente del problema. La idea es que los protocolos de más alto nivel (como TCP) o las aplicaciones lo detectarán y lo solucionarán.

Algoritmo para cálculo del checksum de IP

- Separar el encabezado IP en campos de 16 bits.
- Una vez que todos los campos son separados, se toma el complemento a uno de cada uno de ellos (se obtiene al cambiar cada dígito binario por su complementario).
- Sumar todos los valores.

4				5				0				0				Hexadecimal
0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	Binario
1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1	Complemento a uno

Algoritmo para cálculo del checksum de IP

- Después, se obtiene la suma de cada 16 bits con el resultado de la suma anterior.
- Como el campo de checksum siempre es de 16 bits, se acarrea el valor de 1 si así se obtiene de la suma.
- Este 1 se suma al siguiente bloque de 16 bits, pero en la posición menos significativa.
- El resultado final es el valor del campo checksum.

Algoritmo para cálculo del checksum de IP

- Por ejemplo para el siguiente paquete IP, se describe el cálculo del checksum

0x0000: **4500 003c 6e56 4000 4006 2b52 ac10 f384**

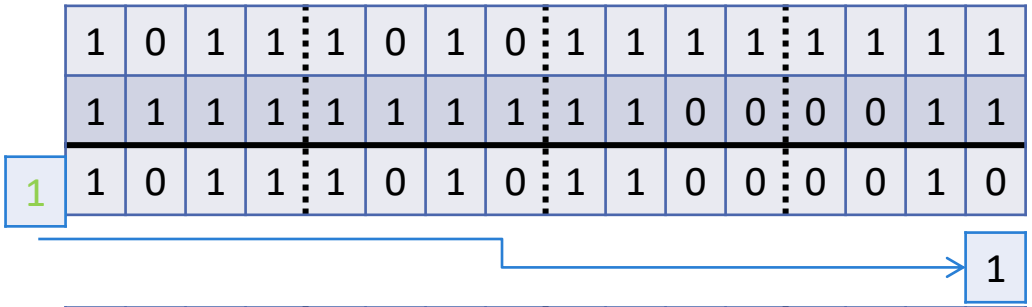
0x0010: **84f8 7c86** e2bb 0050 d42a 03ec 0000 0000

0x0020: a002 16d0 6736 0000 0204 05b4 0402 080a

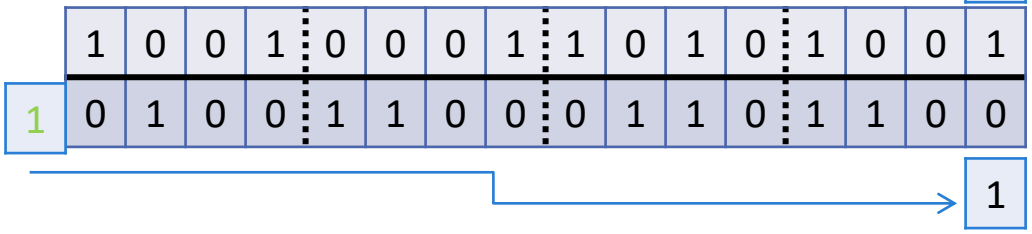
0x0030: 000a 6dba 0000 0000 0103 0306

Algoritmo para cálculo del checksum de IP

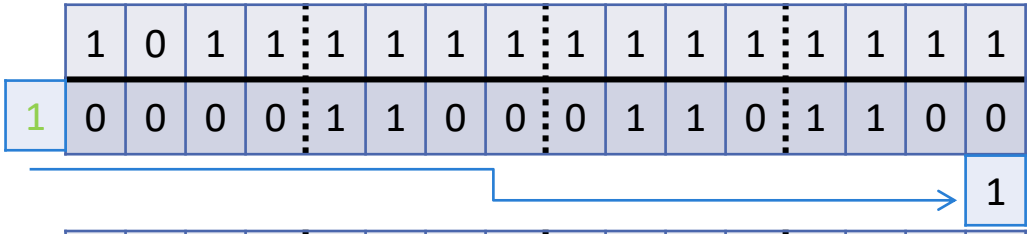
4500	0100 0101 0000 0000
003c	0000 0000 0011 1100



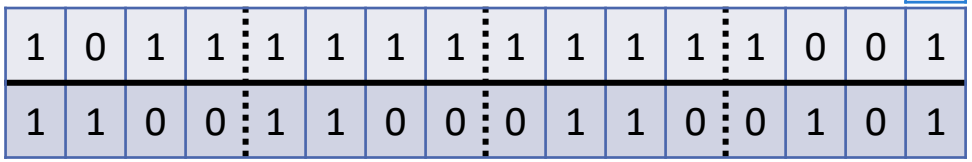
6e56	0110 1110 0101 0110



4000	0100 0000 0000 0000

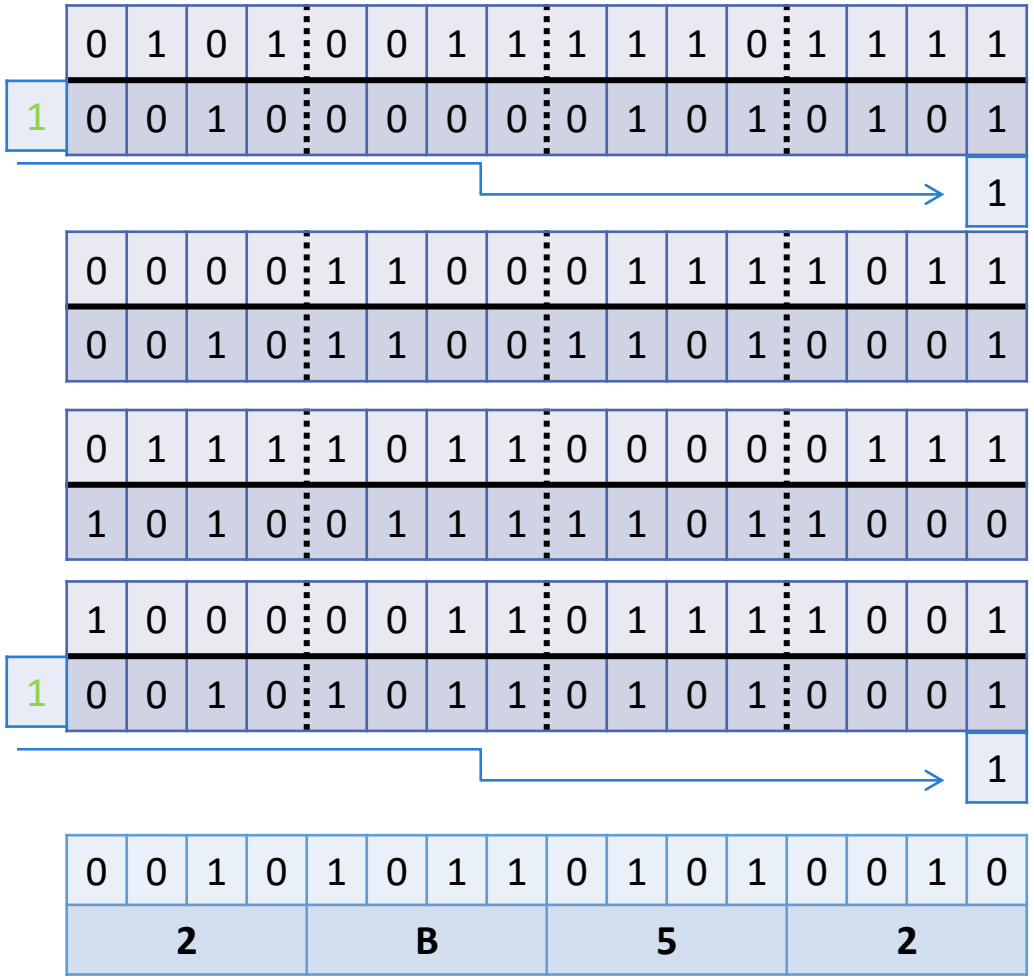


4006	0100 0000 0000 0110



Algoritmo para cálculo del checksum de IP

ac10	1010 1100 0001 0000
f384	1111 0011 1000 0100
84f8	1000 0100 1111 1000
7c86	0111 1100 1000 0110



Algoritmo para cálculo del checksum de IP

- Si el checksum de IP de un paquete es inválida, el paquete será descartado.
- Cada router examina el checksum de IP.
- Si es válido, decrementa el valor del TTL y recalcula la validación de IP.

Cabecera IP corrupta no detectada

- Si los campos de 16 bits son intercambiados, la validación de IP permanece igual.

4500	0100 0101 0000 0000
003c	0000 0000 0011 1100

1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1
1	0	1	1	1	0	1	0	1	1	0	0	0	0	1	0

003c	0000 0000 0011 1100
4500	0100 0101 0000 0000

1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1
1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1
1	0	1	1	1	0	1	0	1	1	0	0	0	0	1	0

Protocolo UDP

- Es “ligero” porque es simple y no garantiza la confiabilidad de la entrega de los datos.

- A diferencia de TCP, que soporta únicamente comunicaciones de un *host* o a otro (unicast), UDP puede entregar tráfico a uno o más *host*.

- No hay certeza de entrega en la capa de UDP. Sin embargo, una aplicación puede ser escrita para que verifique la entrega de estos paquetes.

Protocolo UDP [1]

- Se dice que hace “su mejor esfuerzo”, porque los segmentos pueden perderse en el tránsito, o pueden ser entregados en desorden a la capa de aplicación.

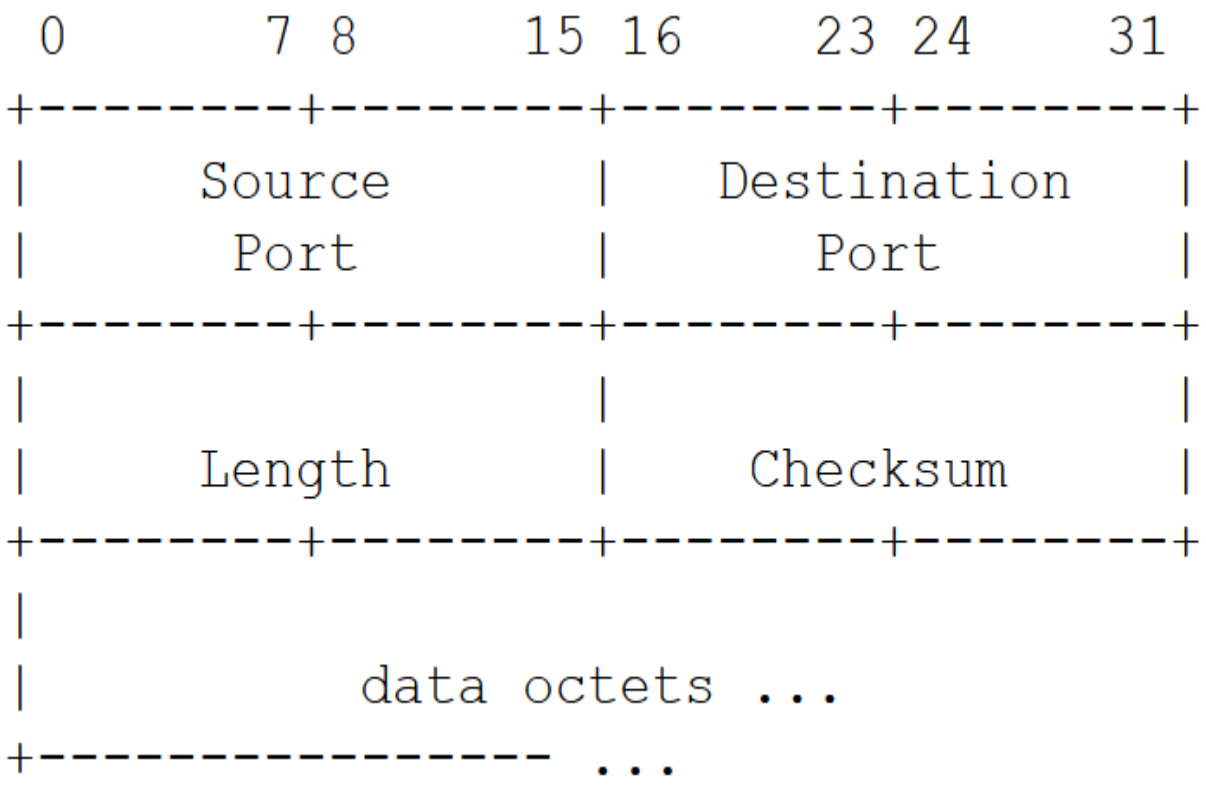
- Es no orientado a conexión porque no un hay establecimiento previo a mandar los datos de la capa de aplicación entre el cliente y el servidor.

- Cada segmento es tratado de forma independiente a los demás.

Protocolo UDP [1]

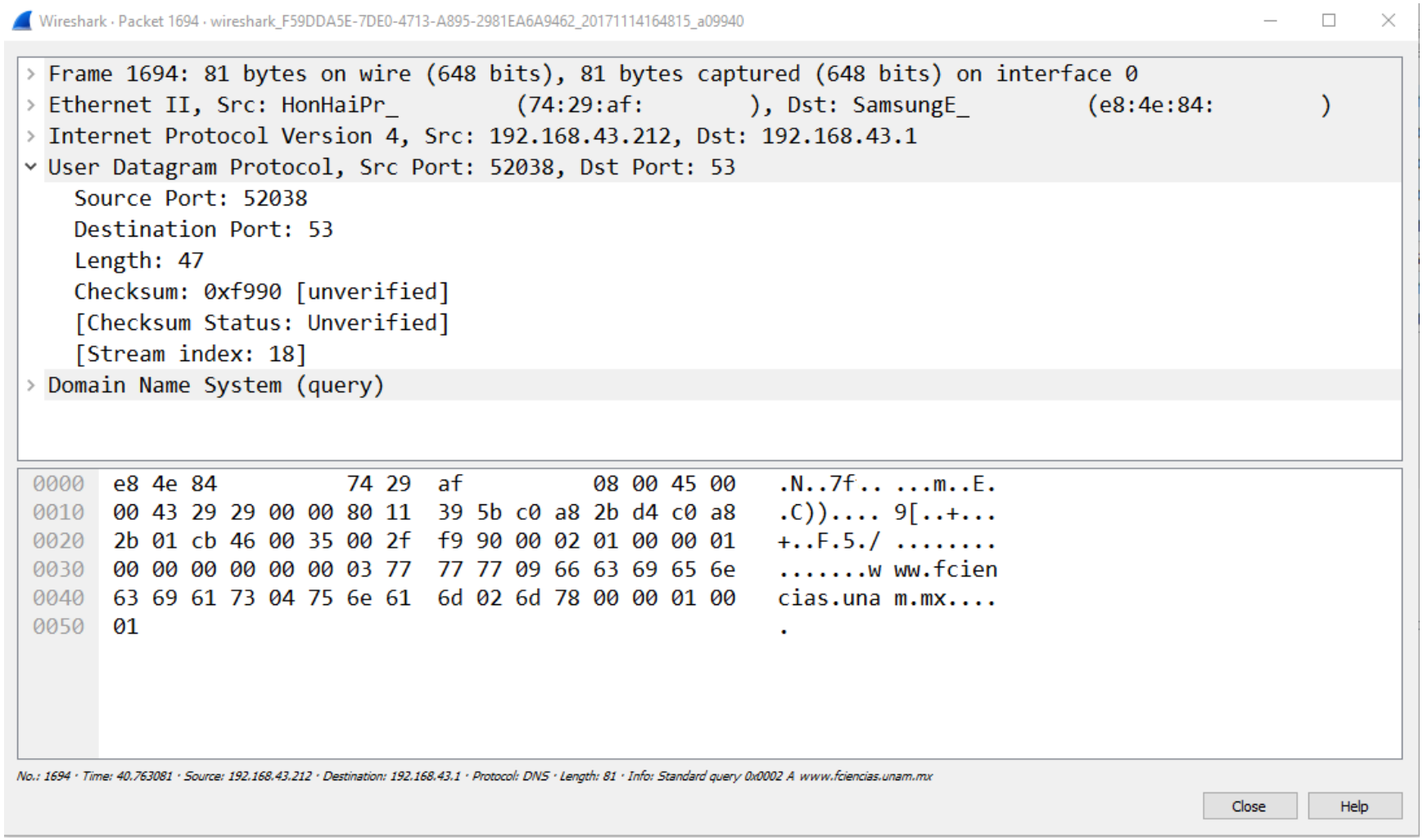
- Es utilizado en aplicaciones:
  - Streaming multimedia
  - DNS
  - SNMP
- Permite identificar algunos errores de corrupción del mensaje a través de un *checksum*, que es validada por el equipo receptor.

Cabecera UDP



Cabecera UDP (RFC 768)

Segmento UDP

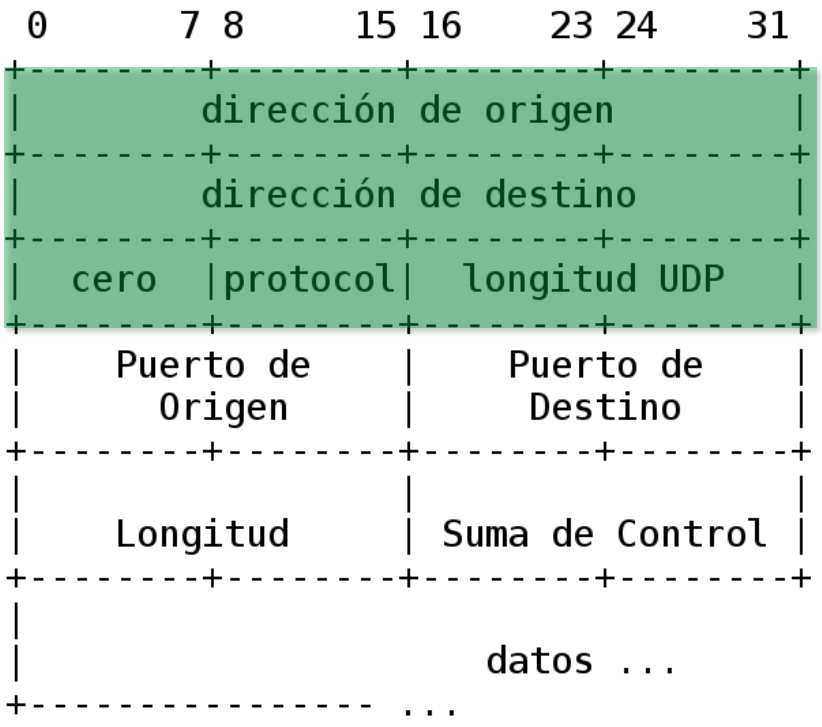


Longitud UDP

- Del 5º al 6º byte de la cabecera UDP
- Tamaño: 16 bits
- Tamaño total de la cabecera y datos de UDP
- Su longitud mínima es de 8 bytes
- Longitud máxima 65535 bytes, aunque normalmente es menor que el tamaño máximo debido a las limitaciones del MTU y de las aplicaciones UDP.

Checksum en UDP

- El host origen agrega al segmento UDP un pseudoencabezado.



- Pseudoencabezado

Checksum en UDP

- El pseudoencabezado consiste de 12 bytes de datos adicionales al segmento UDP:
  - direcciones IP de origen y destino,
  - 8 bits con el tipo de protocolo embebido localizado en la cabecera del protocolo IP
  - copia del campo de longitud de UDP (cabecera + datos).
  - el campo de ceros es utilizado para rellenar el campo de protocolo de 8 a 16 bits, ya que los *checksums* son calculados por bloques de 16 bits de datos.

Checksum en UDP

- Para el cálculo del checksum se usa el mismo algoritmo que para el cálculo del checksum en el protocolo IP,
  - divide el segmento UDP (incluyendo la pseudocabecera), en subconjuntos de 16 bits, se obtiene el complemento a uno de cada subconjunto y se suman entre sí.
  - Si como resultado de la suma se acarrea un bit, éste se suma al resultado pero en la posición menos significativa.

- Se siguen las mismas reglas en TCP para este cálculo.

Protocolo UDP

- En resumen,
  - No tiene confiabilidad en la capa de transporte.
  - Soporta direcciones IP unicast, broadcast y multicast.
  - No cuenta con ordenamiento de mensajes.
  - No existen las sesiones.
  - Tiene verificación de integridad del mensaje a través de un checksum.

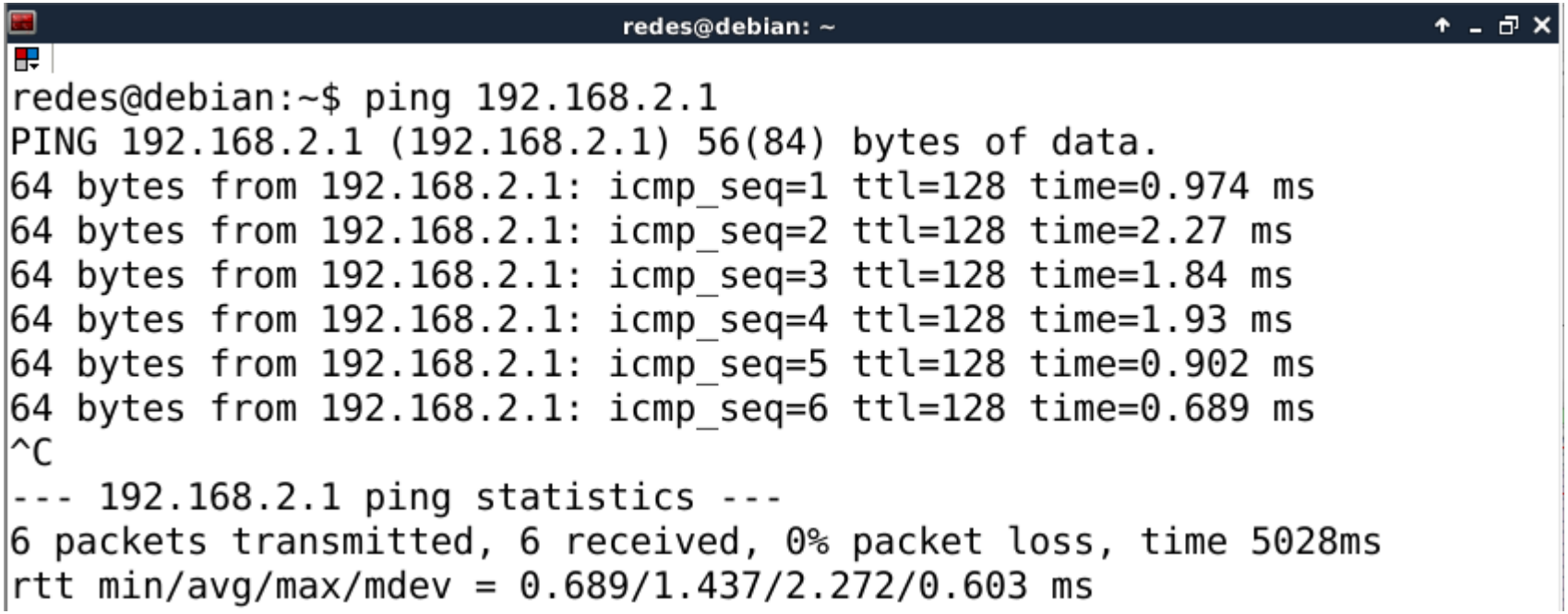
Protocolo ICMP

- ICMP, *Internet Control Message Protocol*
- Es un protocolo de la capa de red. Aunque está encapsulado en la cabecera IP como los protocolos de la capa de transporte (TCP y UDP), no se le considera un protocolo de transporte.
- Encapsulado en la cabecera IP:
  - Número de protocolo 1
- ICMP es “ligero” y fue creado originalmente para la localización de fallas de red.

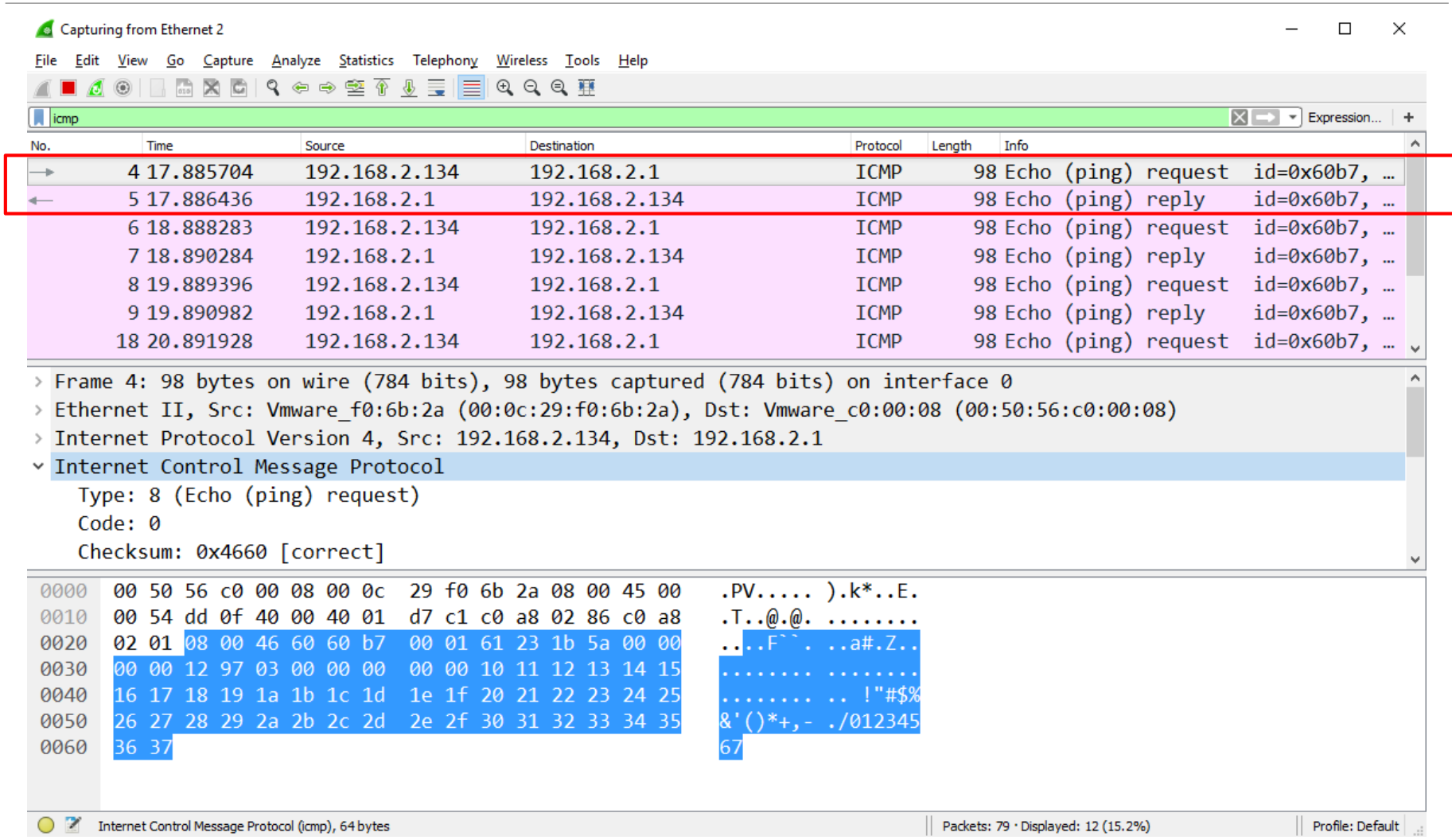
Protocolo ICMP

- ICMP, como IP, no es un protocolo confiable, ya que los paquetes ICMP pueden perderse o dañarse sin que el protocolo lo note.
- Los mensajes ICMP también se emplean para un intercambio simple de información.
- La aplicación del protocolo ICMP más conocida, es la solicitud de respuesta (echo request/echo reply) o ping.
- Ping se usa para conocer si algún equipo en una red es alcanzable o no.

Protocolo ICMP - Ping



Protocolo ICMP - Ping



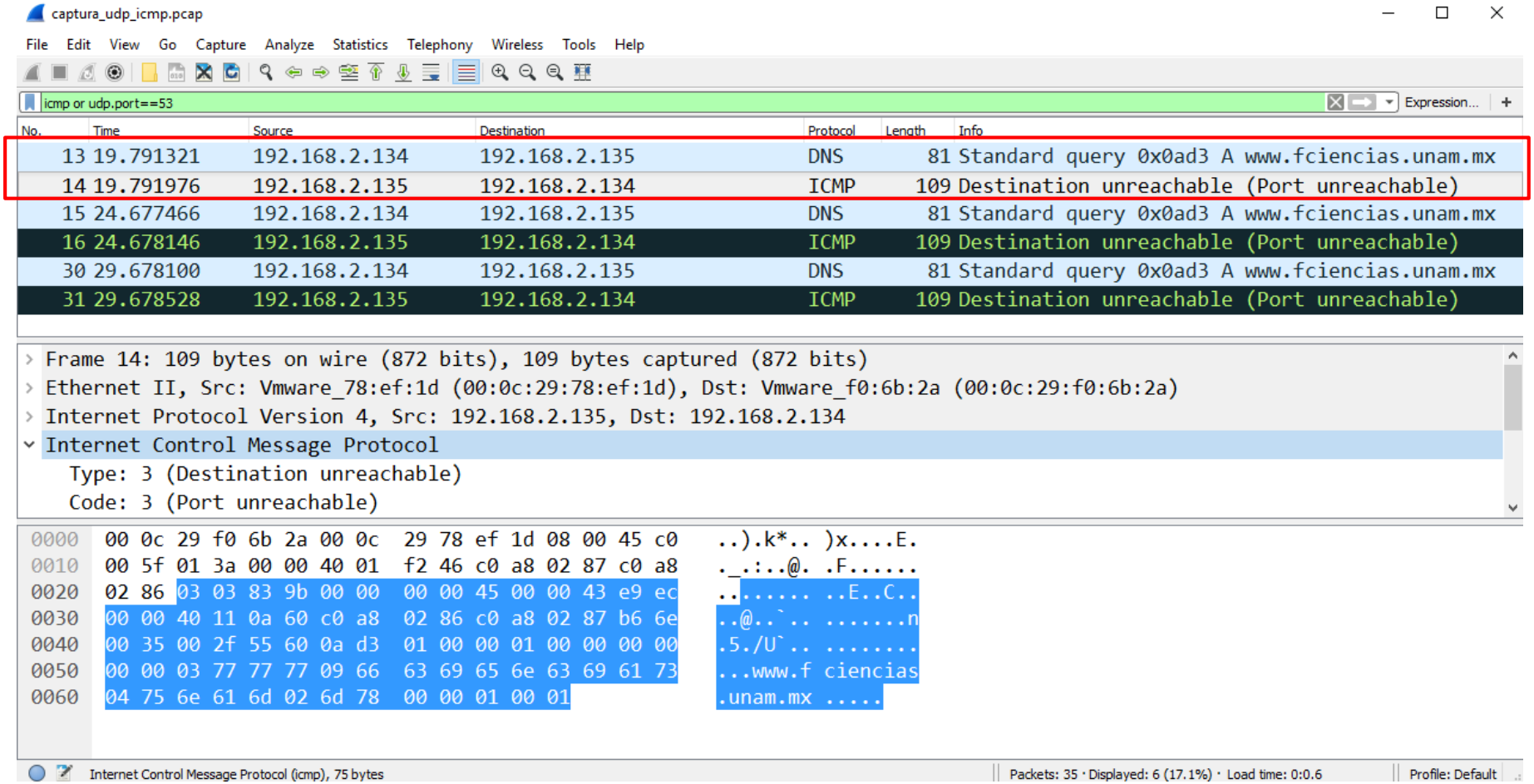
Protocolo ICMP

- Debido a que UDP no posee envío de mensajes de error, los host se valen de ICMP para informar de posibles errores.
- Un ejemplo de un problema permanente en la capa de transporte, podría ser el envío de un paquete de UDP a un puerto que no se encuentre escuchando en el equipo remoto.
- Este problema no será resuelto, aún cuando sean enviados varios paquetes UDP. ICMP es utilizado para informar al equipo origen del paquete que el puerto UDP en el equipo remoto es inalcanzable.

Protocolo ICMP – Port unrechable

- Consulta DNS al host con dirección IP 192.168.2.135, que permite conexiones pero no ofrece el servicio de DNS, es decir, no tiene el puerto 53 de UDP abierto.

Protocolo ICMP – Port unrechable



Transporte confiable

- La capa de red ofrece un servicio del mejor esfuerzo (*best-effort*) a las capas superiores en el modelo de referencia OSI.
- En muchas ocasiones, el mejor esfuerzo de la capa de red no es suficiente pues los paquetes transmitidos pueden:
  - Ser alterado su contenido
  - Pueden perderse
  - Pudieran llegar paquetes duplicados

Envío confiable de datos

- Para que la capa de transporte del host que envió un paquete esté segura que el paquete llegó correctamente al destino, es necesario:
  - que la capa de transporte del host destino envíe una **confirmación positiva** (ACK)
  - que se tenga un **número de secuencia**, para poder ordenar los paquetes en el host destino
- En el caso del protocolo UDP estas condiciones no son necesarias, ya que es un protocolo no orientado a la conexión.

Envío confiable de datos

- Es importante notar que el mismo encabezado de IP cuenta con un campo de número de secuencia de cada paquete que se transmite.
- Este número de secuencia a nivel de la capa de red puede ser suficiente para aplicaciones basadas en UDP, pero no es suficiente para protocolos orientados a conexión como TCP, pues no es lo mismo intentar que entregar correctamente.

Paquetes alterados

- Cuando un paquete es alterado en el trayecto es necesario que la capa de transporte del host destino detecte que uno o varios bits del paquete entrante fueron alterados.
- Al igual que el protocolo IP, la capa de transporte del host origen añade el campo de *checksum* en el encabezado del paquete de tal forma que la capa de transporte del host destino pueda detectar la presencia de errores.
- Este *checksum* forma parte tanto del encabezado UDP como del encabezado TCP. Y es un **método de detección de errores**.

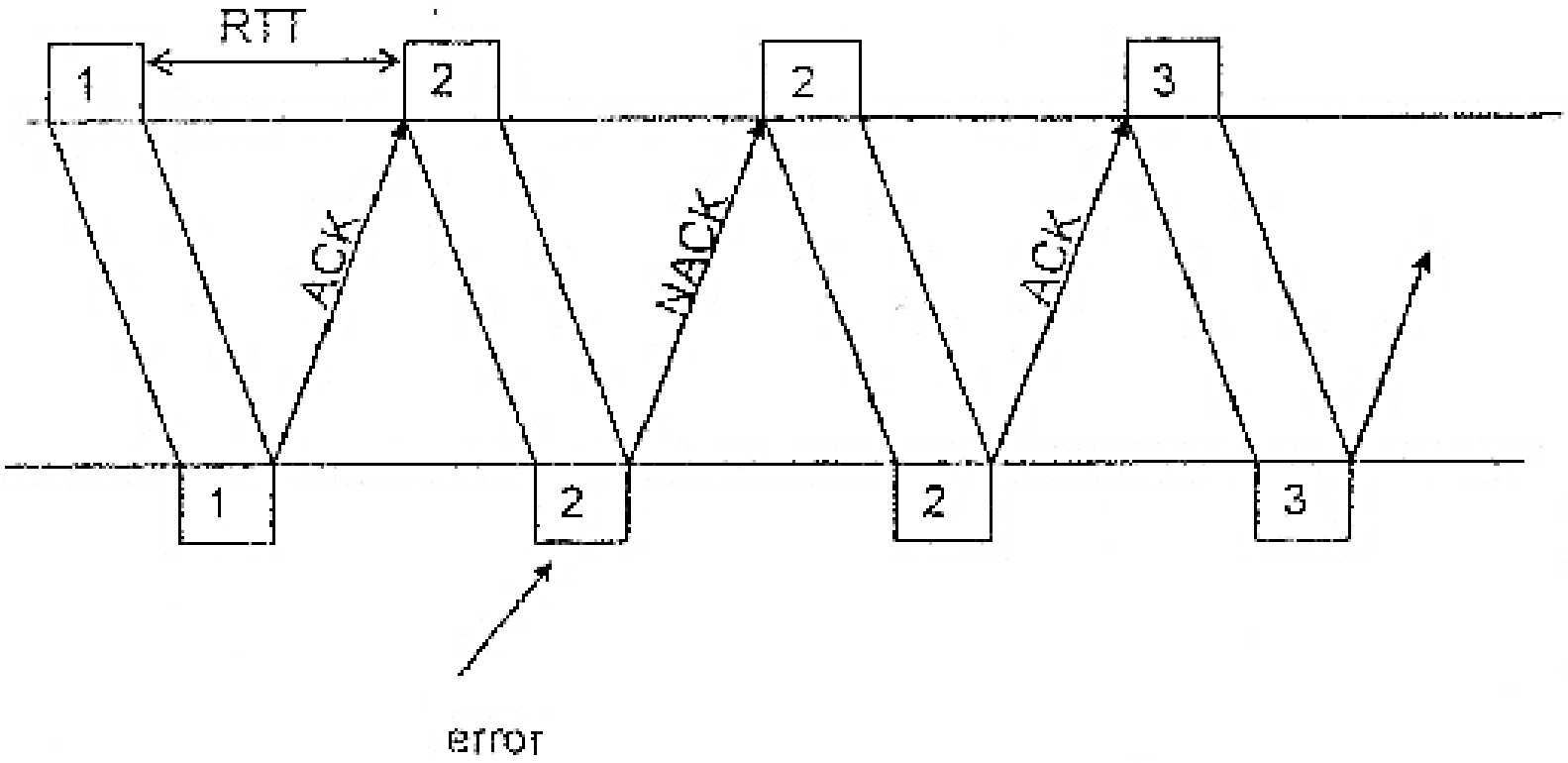
Paquetes alterados

- En caso de detectarse errores en el segmento, es necesario informar a la capa de transporte del host origen con una **confirmación negativa** (NACK) para que ésta retransmita el paquete correspondiente.

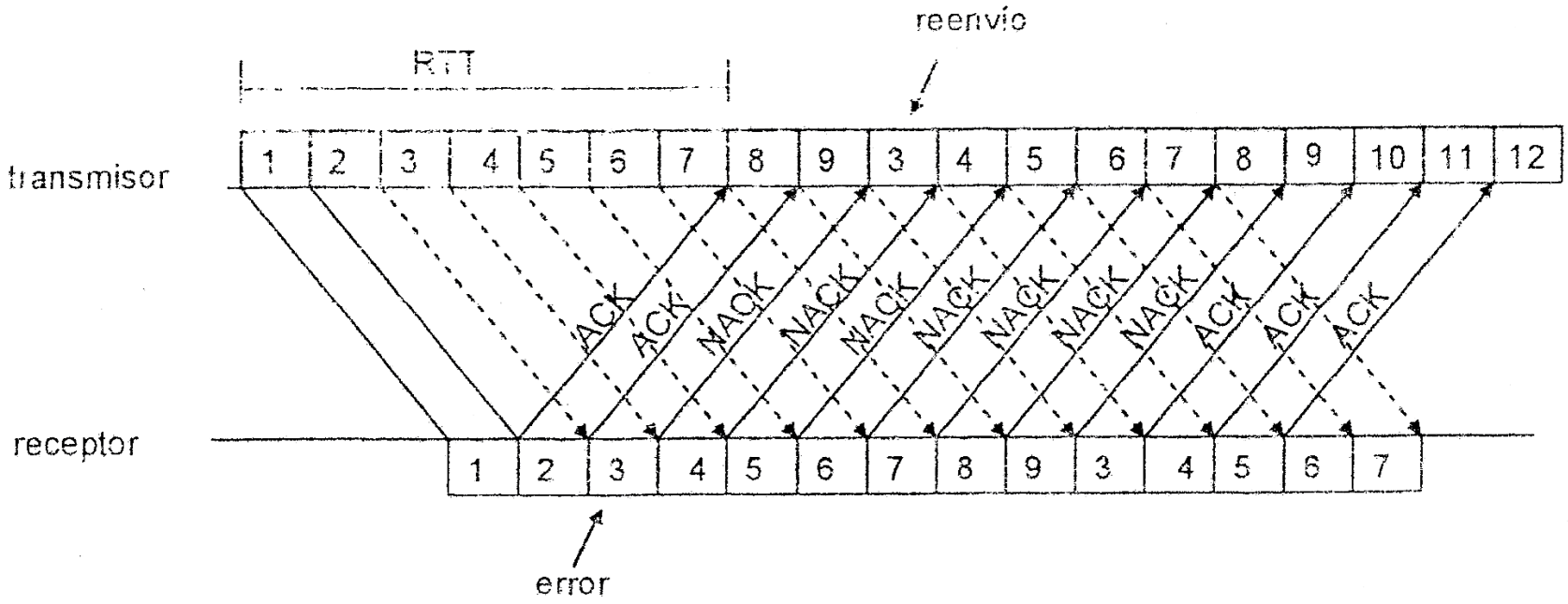
Técnicas de control de flujo

- Las siguientes tres técnicas se proponen para un tener un transporte confiable de datos, enviados de un host origen a un destino,
  - Stop & wait
  - Go-back-n
  - Selective repeat

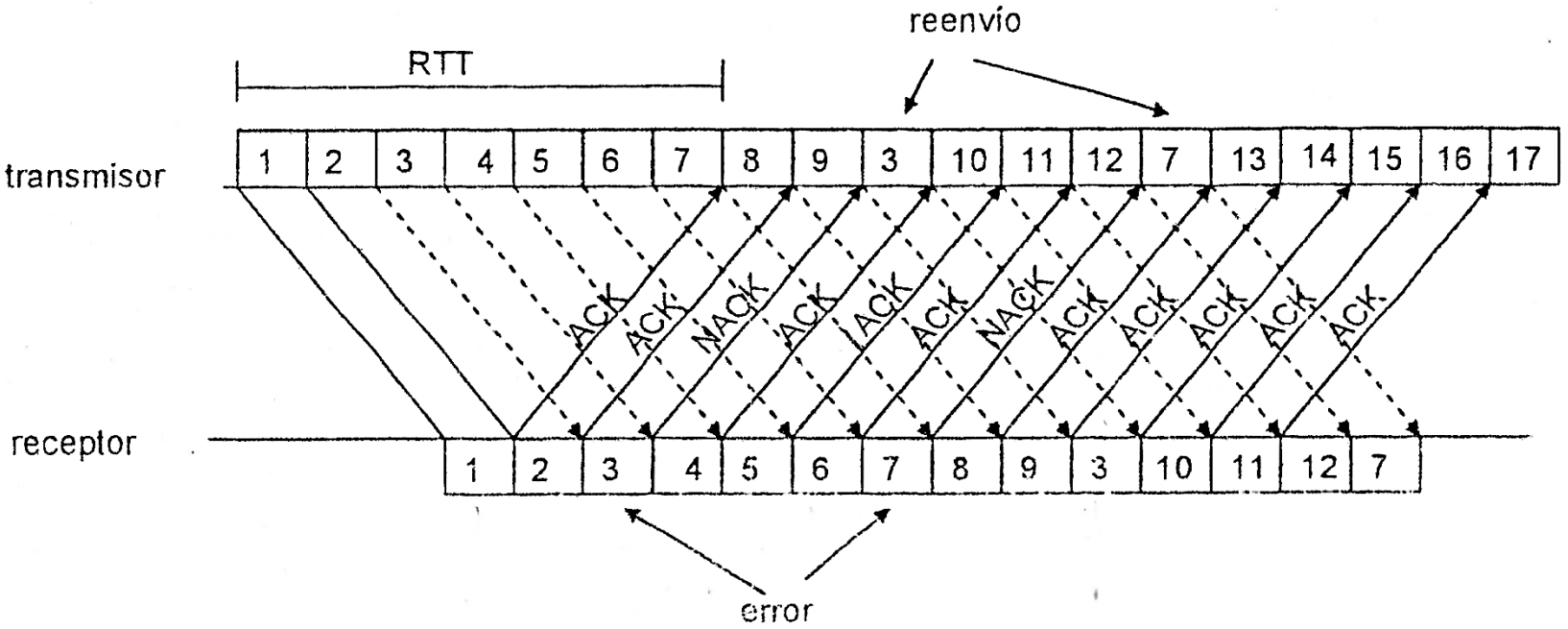
Stop & wait



Go-back-n


reenvío
RTT
transmisor
receptor
1 2 3 4 5 6 7 8 9 3 4 5 6 7 8 9 10 11 12
ACK
ACK
NACK
NACK
NACK
NACK
NACK
NACK
ACK
ACK
ACK
1 2 3 4 5 6 7 8 9 3 4 5 6 7
error

Selective repeat



Transporte confiable[1]

- En resumen para contar con un transporte confiable existen los siguientes mecanismos:

Checksum	Se utiliza para detectar errores a nivel de bits en el paquete transmitido.
Timer	Se usa para finalizar o retransmitir un paquete, posiblemente porque el paquete (o su ACK) se perdió durante la transmisión. Un timeout puede presentarse debido al retraso de un paquete, pero no se ha perdido y puede llegar al destino después de que el tiempo de espera ha finalizado (timeout prematuri); o se presenta cuando el paquete a sido recibido por el receptor, pero el paquete con ACK enviado por el receptor se perdió, por lo tanto el receptor puede recibir copias duplicadas del paquete.

Transporte confiable[1]

Sequence number	Se usa para llevar la numeración secuencial de los paquetes enviados por el emisor al receptor. La falta de un número de secuencia permite que el receptor identifique un paquete perdido. Los paquetes con números de secuencia duplicados permiten que el receptor detecte copias duplicadas de un paquete.
Acknowledgment	Es utilizado por el receptor para informar al remitente que un paquete o conjunto de paquetes se ha recibido correctamente. Los acuses de recibo generalmente llevan el número de secuencia del paquete o paquetes confirmados (que han sido recibido correctamente). Los paquetes de ACK pueden ser individuales o acumulativos, según el protocolo.

Transporte confiable[1]

Negative acknowledgment	Utilizado por el receptor para informar al remitente que un paquete no se ha recibido correctamente. Las confirmaciones negativas (o NACK) generalmente llevarán el número de secuencia del paquete que no se recibió correctamente.
Window, pipelining	El tamaño de ventana puede ser establecido en función de: • la capacidad del receptor de recibir y almacenar mensajes en su buffer (que serán procesados), • nivel de congestión de la red, • o ambos.

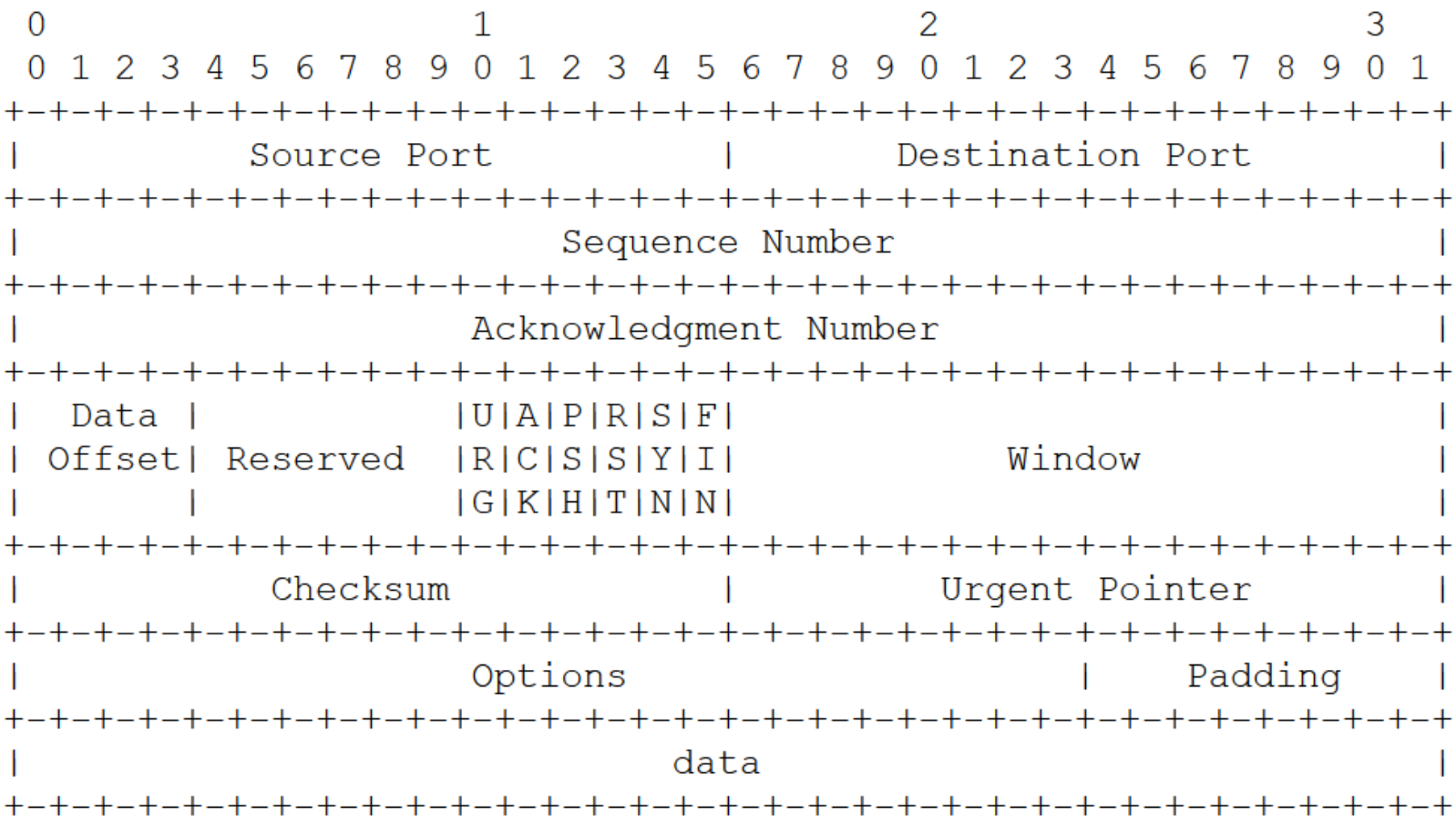
Protocolo TCP

- Es un protocolo confiable, porque provee mecanismos para asegurar la entrega de los datos.
- Utiliza un modelo de conexión basado en direcciones unicast, un equipo origen habla con un equipo destino
- Controla las sesiones para optimizar intercambio de datos.
- Utiliza números de secuencia para ordenar los paquetes. Puesto que los segmentos TCP son enviados en paquetes IP y pueden tomar diferentes rutas para llegar al equipo destino, es posible que los datos lleguen en un orden diferente al que se generaron.

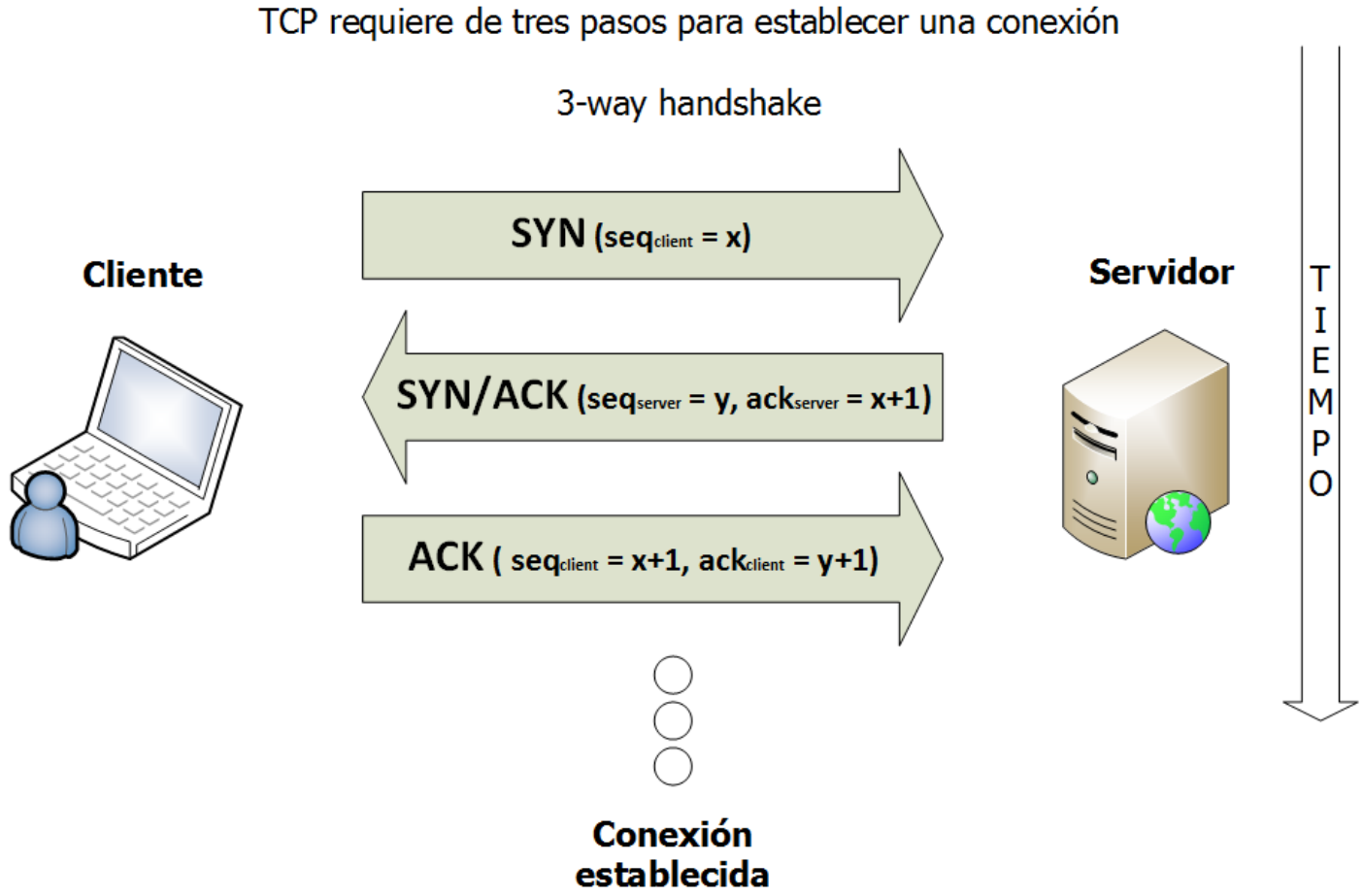
Protocolo TCP

- Por ello, TCP necesita de los números de secuencia para ordenar los datos en el equipo destino.
- Utiliza banderas para indicar el estado de la conexión.
- Encapsulado en la cabecera IP:
  - Número de protocolo 6.

Protocolo TCP



Estableciendo una conexión



Números de secuencia

- La transmisión de información en el protocolo TCP se realiza a través de varios paquetes.
- Cada paquete es identificado por un número de secuencia.
- Número de 32 bits.
- Utilizado para ordenar otros paquetes enviados previamente.

Números de secuencia

- Ocupa del 4º al 7º byte de la cabecera TCP.
- El número de 32 bits identifica únicamente el byte inicial de segmento de datos.
- Puede cambiar para cada nuevo (no reintento) segmento TCP enviado.
- Números de secuencia inicial (ISN) representan el primer número de secuencia en el intercambio TCP.

Números de confirmación (Acknowledgement)

- El equipo origen sabe que el equipo destino ha obtenido los datos por un acuerdo (acknowledgement).
- Su tamaño es de 32 bits.
- La confirmación que envía el equipo destino identifica la recepción de datos enviados.
- Los números de reconocimiento son el último número de secuencia recibido más uno, en el establecimiento de la sesión TCP.
- Indica el próximo número de secuencia esperado por el equipo destino.

Números de confirmación (Acknowledgement)

- Una conexión inicial SYN utilizará un número de secuencia único.
- La confirmación deberá ser el próximo número de secuencia esperado.
- Cero es un número de confirmación inicial imposible (a menos que el número de secuencia inicial sea 0xffffffff).

Ejemplo números de secuencia y confirmación

Universidad Nacional Autónoma de México
Facultad de Ciencias

Redes de computadoras

Números de secuencia en TCP

Paulo Contreras Flores
paulo.contreras.flores@ciencias.unam.mx

Cada capa ya sea del modelo OSI o del modelo TCP/IP, está conformada por datos que tienen un significado para esa capa es decir, que con esos datos la capa lleva a cabo su función. Cada conjunto de estos datos tiene un nombre dependiendo de la capa en la que se encuentre, en la siguiente figura 1 se muestran dichos nombres.

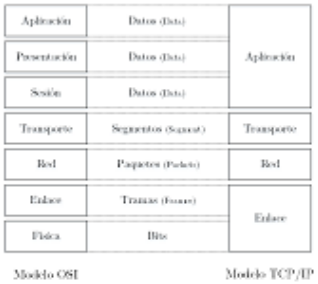


Figura 1: Nombre para cada tipo de datos en cada capa del modelo OSI y TCP/IP

Ray Tomlinson introdujo en 1975 el concepto del three-way handshake, el cual es un protocolo de establecimiento de comunicación para la transmisión de datos. Este proceso se realiza generalmente antes de que cualquier dato sea enviado entre los equipos. Lo que se representa, es un cliente o equipo origen iniciando una conexión al servidor o equipo destino. Debido a que es una conexión TCP (capa de transporte), un puerto o servicio debe identificar al servicio al cual se desea conectar.

Cada equipo de la sesión seleccionará un número de secuencia inicial (ISN por sus siglas en inglés) como el primer número de secuencia. Esto significa que el cliente y el servidor seleccionarán diferentes números de secuencia individuales. ¿Cómo son generados estos números de secuencia?

1

Tiempo de retransmisión

11:48:21.428547 IP riu.unam.mx.33105 > fciencias.unam.mx.ssh: Flags [S], seq 3482595538, win 5840, options [mss 1460,sackOK,TS val 9714011 ecr 0,nop,wscale 6], length 0

11:48:24.426368 IP riu.unam.mx.33105 > fciencias.unam.mx.ssh: Flags [S], seq 3482595538, win 5840, options [mss 1460,sackOK,TS val 9714761 ecr 0,nop,wscale 6], length 0

11:48:30.426908 IP riu.unam.mx.33105 > fciencias.unam.mx.ssh: Flags [S], seq 3482595538, win 5840, options [mss 1460,sackOK,TS val 9716261 ecr 0,nop,wscale 6], length 0

Reconocimientos duplicados

fciencias.unam.mx.ftp > riu.unam.mx.31054: P 1:81(80) ack 1 win 32120

riu.unam.mx.31054 > fciencias.unam.mx.ftp: . ack 81 win 8660 (DF)

fciencias.unam.mx.ftp > riu.unam.mx.31054: P 81:116(35) ack 1 win 32120 (DF)

bug.seguridad.unam.mx.ftp > riu.unam.mx.31054 P 116.144(28) ack 1 win 32120 (DF)

riu.unam.mx.31054 > fciencias.unam.mx.ftp: . ack 81 win 8680 (DF)

fciencias.unam.mx.ftp > riu.unam.mx.31054: P 144:173 (29) ack 47 win 32120 (DF)

riu.unam.mx.31054 > fciencias.unam.mx.ftp: . ack 81 win 8680 (DF)

fciencias.unam.mx.ftp > riu.unam.mx.31054: P 81:116(35) ack 1 win 32120 (DF)

Puertos TCP

- Puerto origen: del 0 al 1er byte
- Puerto destino: del 2º al 3er byte
- Cada campo es de 16 bits.
- Número de puertos: 1 al 65,535 (216).
- Las conexiones se realizan de puertos dinámicos a puertos reservados (servicios).

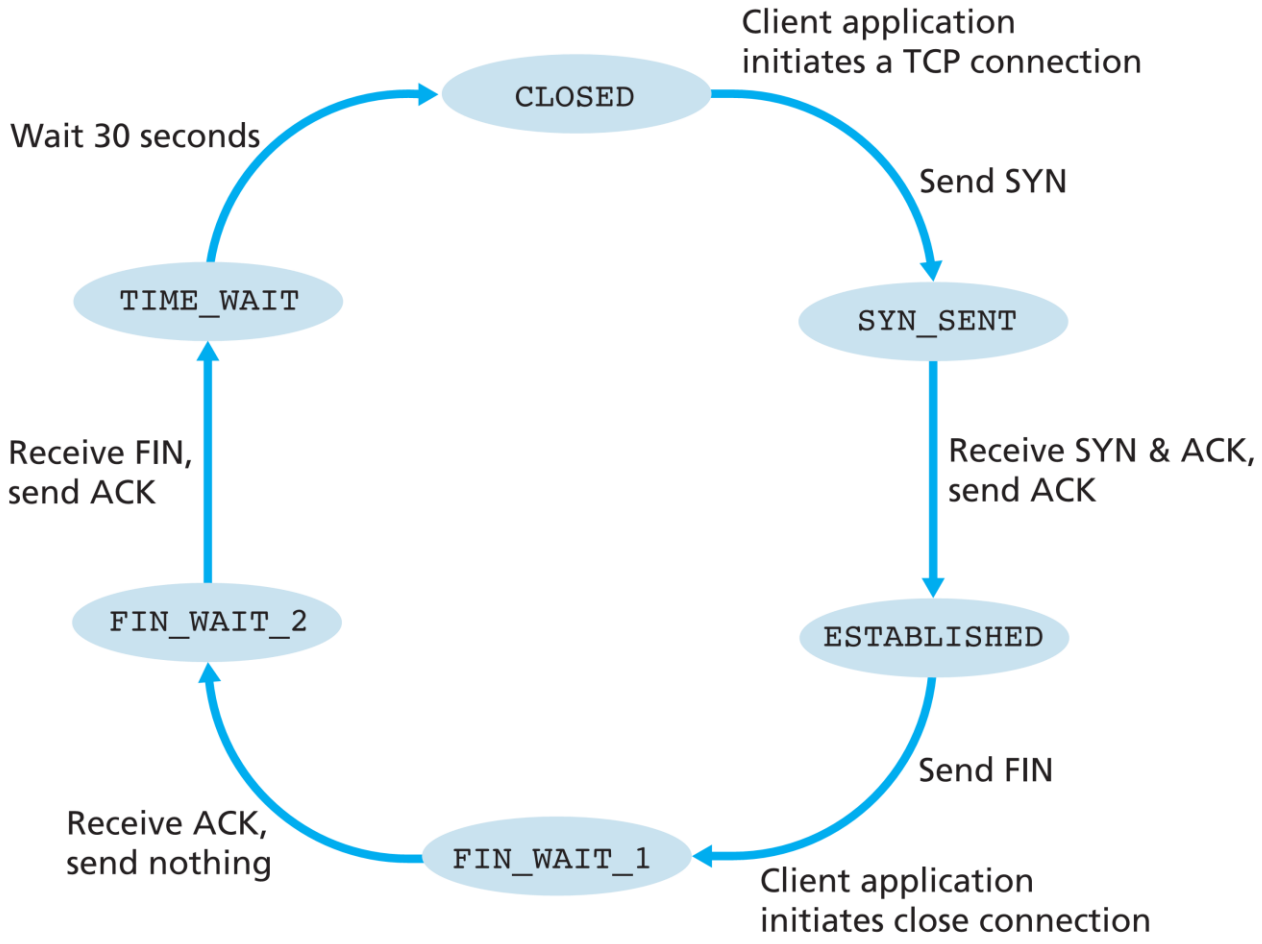
Puertos servidor y cliente

- El puerto en un servidor no cambia.
- Antes de que el cliente inicie la conexión, escoge un puerto efímero aleatorio.
- El cliente y servidor se comunican usando los puertos establecidos.

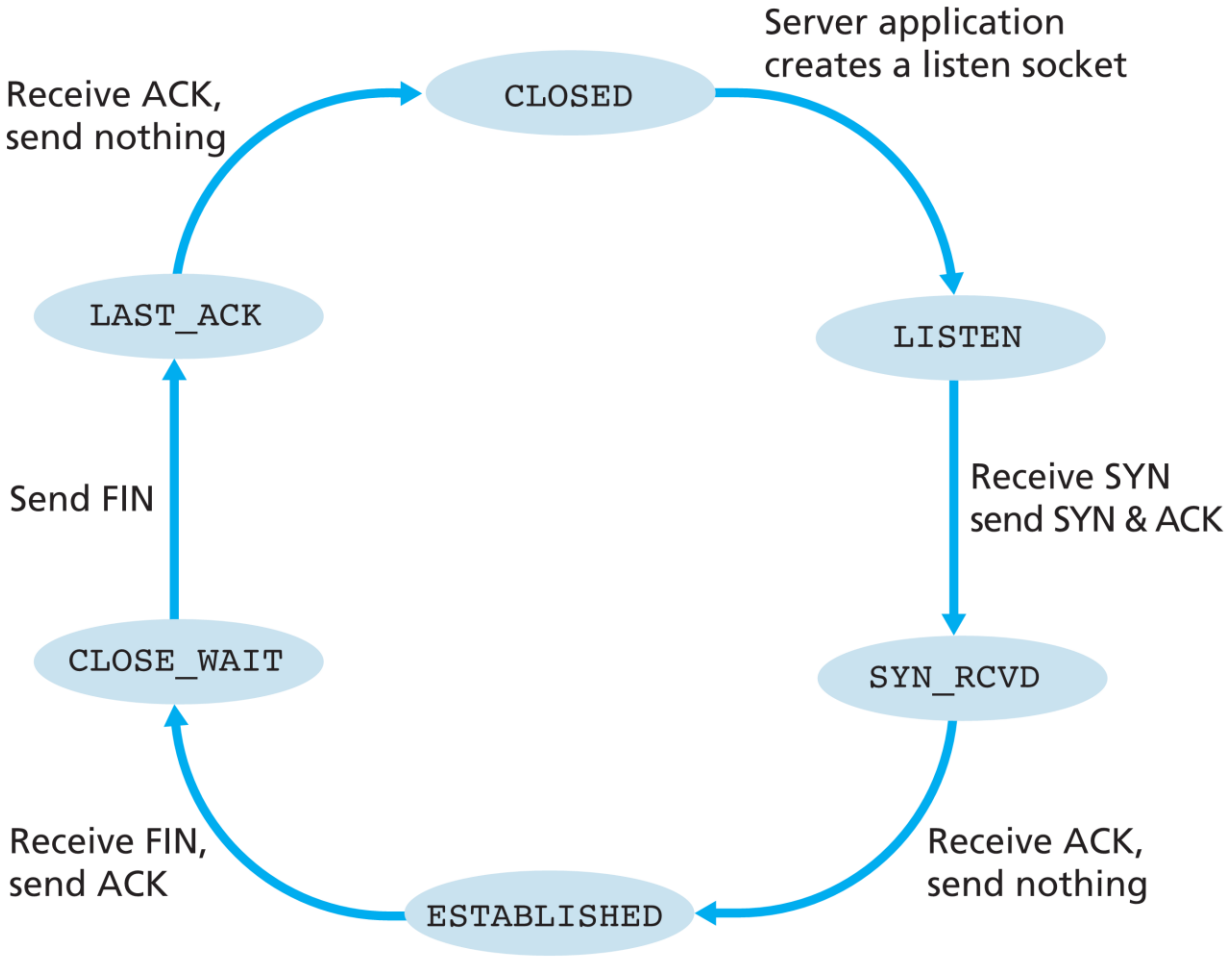
Banderas TCP

- FIN: Cierre correcto de una conexión TCP.
- SYN: Inicio de una conexión TCP.
- RESET: Aborta una conexión TCP de forma repentina.
- PUSH: Indica transmisión de datos.
- ACK: Confirmación de envío de información o de una conexión TCP.
- URG: Indica el envío de un paquete urgente (envío de datos con alta prioridad).
- ECN-related: Indica la notificación explícita de la congestión.

Estados de TCP en el cliente



Estados de TCP en el servidor



Tamaño de ventana

- El campo de 16 bits del equipo receptor es el tamaño del buffer para la conexión TCP.
- Actúa como control de flujo:
- El tamaño de ventana cambia de forma dinámica conforme los datos son recibidos.
- El tamaño de ventana 0 dice al equipo origen que deje de enviar datos temporalmente.
- Cuando un equipo puede recibir datos adicionales, el tamaño de ventana aumenta a más de 0.

Tamaño de ventana dinámica

```
12:59:21.302481 192.168.0.25. 54643 > 192.168.0.31.21: .ack 1 win 8760 (DF)
12:59:21.313179 192.168.0.31.21 > 192.168.0.25. 54643 : P 1:81(80) ack 1 win 32120 (DF)
12:59:21.421038 192.168.0.25. 54643 > 192.168.0.31.21: .ack 81 win 8680 (DF)
12:59:23.885149 192.168.0.25. 54643 > 192.168.0.31.21: P 1:14(13) ack 81 win 8680 (DF)
12:59:23.885458 192.168.0.31.21 > 192.168.0.25. 54643 : .ack 14 win 32120 (DF)
12:59:23.886251 192.168.0.31.21 > 192.168.0.25. 54643 : P 81:116(35) ack 14 win 32120 (DF)
12:59:24.022024 192.168.0.25. 54643 > 192.168.0.31.21: .ack 116 win 8645 (DF)
12:59:25.684119 192.168.0.25. 54643 > 192.168.0.31.21: P 14:29(15) ack 116 win 8645 (DF)
12:59:25.695559 192.168.0.31.21 > 192.168.0.25. 54643 : P 116:114(28) ack 29 win 32120 (DF)
[tos 0x10]
12:59:25.821992 192.168.0.25. 54643 > 192.168.0.31.21: .ack 144 win 8617 (DF)
12:59:26.729110 192.168.0.25. 54643 > 192.168.0.31.21: P 29:35(6) ack 144 win 8617 (DF)
```

Maximum Segment Size - MSS

- El máximo tamaño de segmento (Maximum Segment Size) puede ser enviado solamente junto a un SYN y representa el tamaño más grande que los datos de la capa de aplicación pueden tener en la comunicación.
- Cabe recordar que el tamaño total del paquete IP será al menos de 40 bytes más que este valor, debido a las cabeceras IP y TCP.
- Este es establecido solamente una vez y no es modificado como el tamaño de la ventana.

Maximum Segment Size - MSS

- El valor óptimo para el MSS es cercano al valor del MTU menos 40.
- Si el MSS no es establecido, el equipo que recibe el paquete considerará un valor de 546 bytes como predeterminado.
- Este parámetro opcional para el encabezado de TCP se ubica después de los 20 bytes de la cabecera de TCP y son conocidos como opciones de TCP.

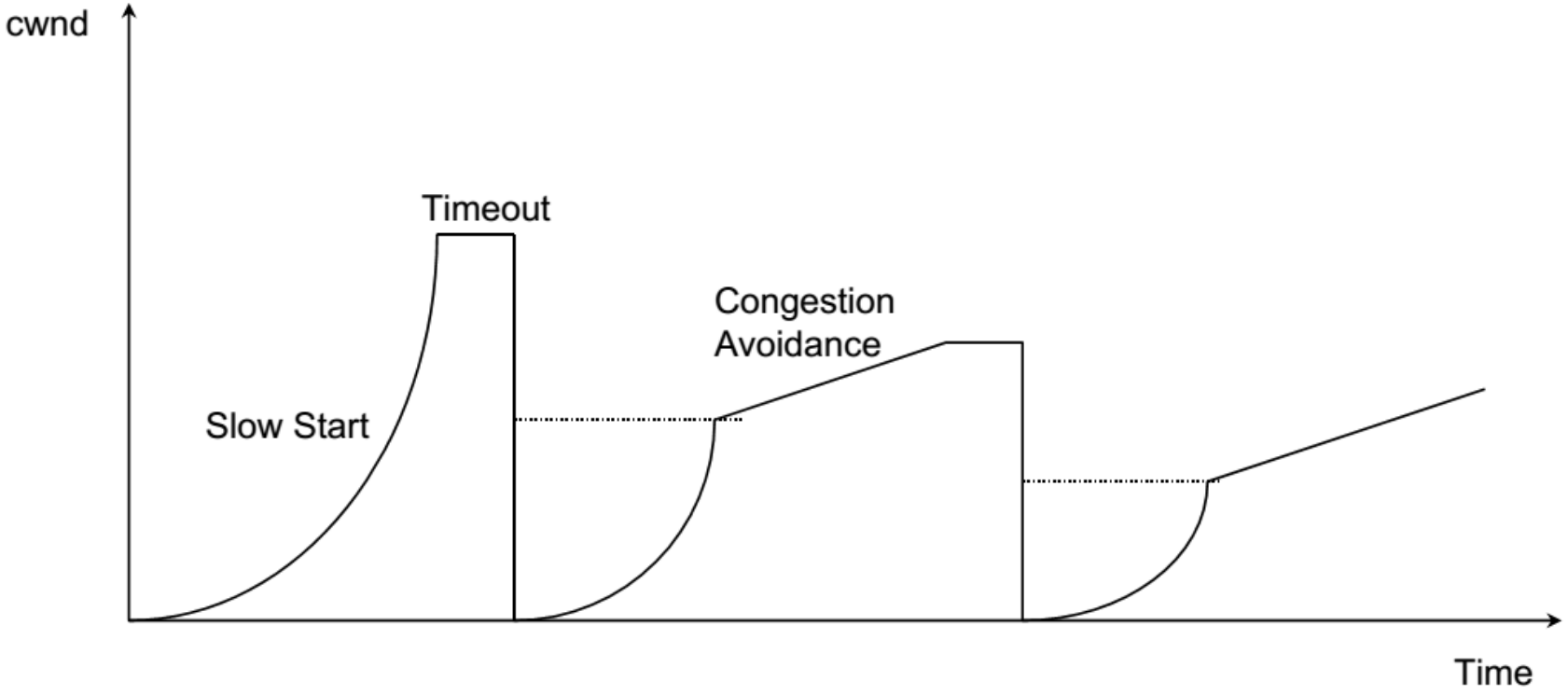
Control de Congestión en TCP

- TCP necesita un control de congestión en la comunicación entre dos *host* (*end-to-end*).
- El enfoque adoptado por TCP es que cada emisor limite la velocidad a la que envía tráfico a su conexión como una función de la congestión de la red.
- Si quien envía datos en la comunicación percibe que hay poco congestión entre el destino y él, entonces incrementa su tasa de envío de datos.
- El control de congestión TCP utiliza una variable llamada venta de congestión o simplemente ventana (cwnd) para realizar el seguimiento.

Control de Congestión en TCP

- El algoritmo de congestión de TCP tiene tres modos:
  - slow start
  - congestion avoidance
  - fast recovery
- De acuerdo al RFC 5681, slow start y congestion avoidance son componentes obligatorios.
- La diferencia principal entre ellos es la forma de incrementar cwnd, slow start incrementa el tamaño de ventana "más rápido" que congestion avoidance.

Slow start – congestion avoidance



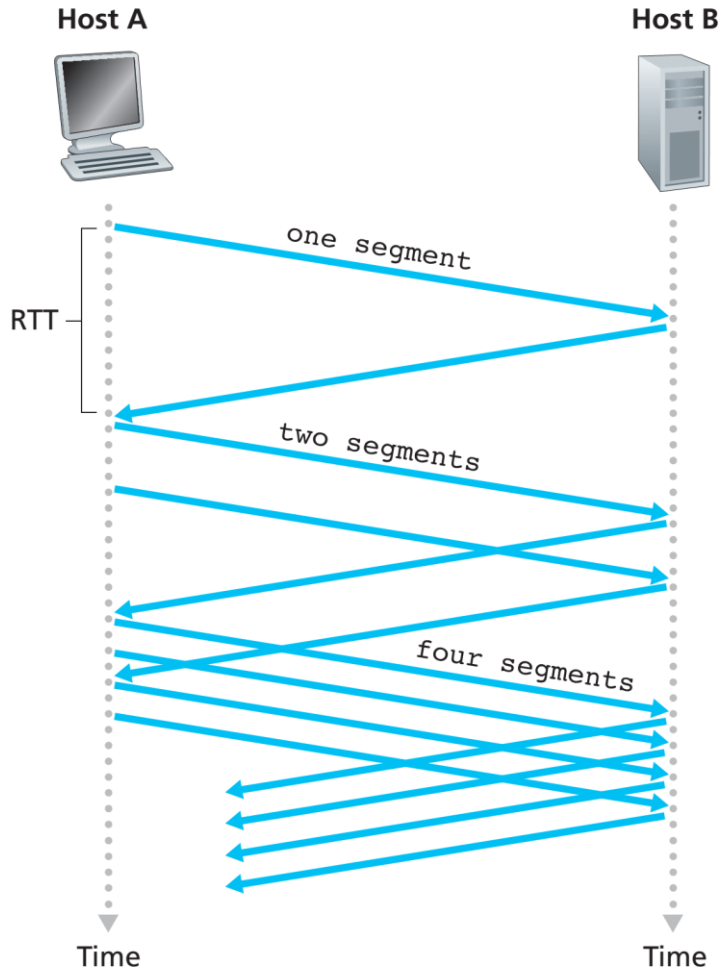
Pseudocódigo del control de congestión en TCP

```
%Inicia la conexión TCP
cwnd =  1;              % congestion window
ssthresh =  infinito;   % slow start threshold
Por cada ACK recibido
        if (cwnd < ssthresh)
                % Slow Start
                cwnd =  cwnd + 1;  % exponencial
        else
                %Congestion Avoidance
                cwnd =  cwnd + 1 / cwnd;
Por cada timeout
        % Multiplicative decrease
        ssthresh = 0.5*cwnd_antes_ocurrir_problema;
        cwnd =  1;
```

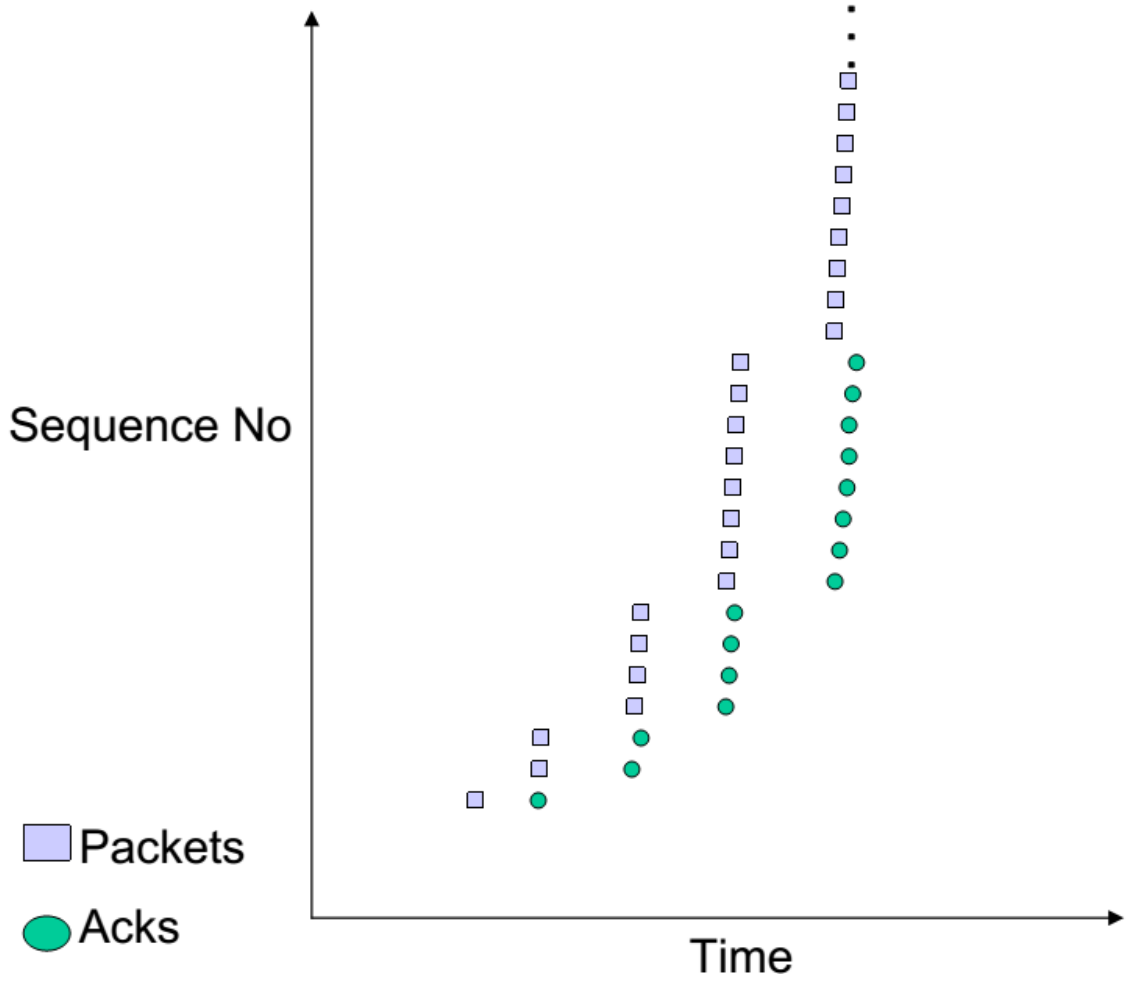
Slow start

- Cada vez que inicie el tráfico en una nueva conexión, o cuando se reinicie la transmisión después de una congestión, el valor de cwnd es de 1 MSS.
  - cwnd = 1
- Cada vez que un segmento es confirmado con el envío del ACK, se incremente cwnd en 1 MSS (cwnd++), es decir, se incrementa en un 1 MSS por cada ack recibido.
- Entonces el incremento de cwnd es exponencial.
- La tasa de envío en TCP comienza lento pero crece exponencialmente durante la fase de slow start.

Slow start


Host A
Host B
one segment
RTT
two segments
four segments
Time
Time

Slow start



Slow start

- Existen tres casos para detener slow start:
- Primero,
  - Se presentó un timeout, por una pérdida de segmentos, por ejemplo por una congestión en el medio de transmisión.
  - Establece el valor de ssthresh a la mitad del valor cwnd cuando la congestión fue detectada.
  - El que envía los datos de TCP establece el valor de cwnd en 1 MSS, y comienza el proceso de slow start de nuevo.
- Segundo,
  - Cuando cwnd es igual o mayor al valor de ssthresh, se entra al modo de congestion avoidance.

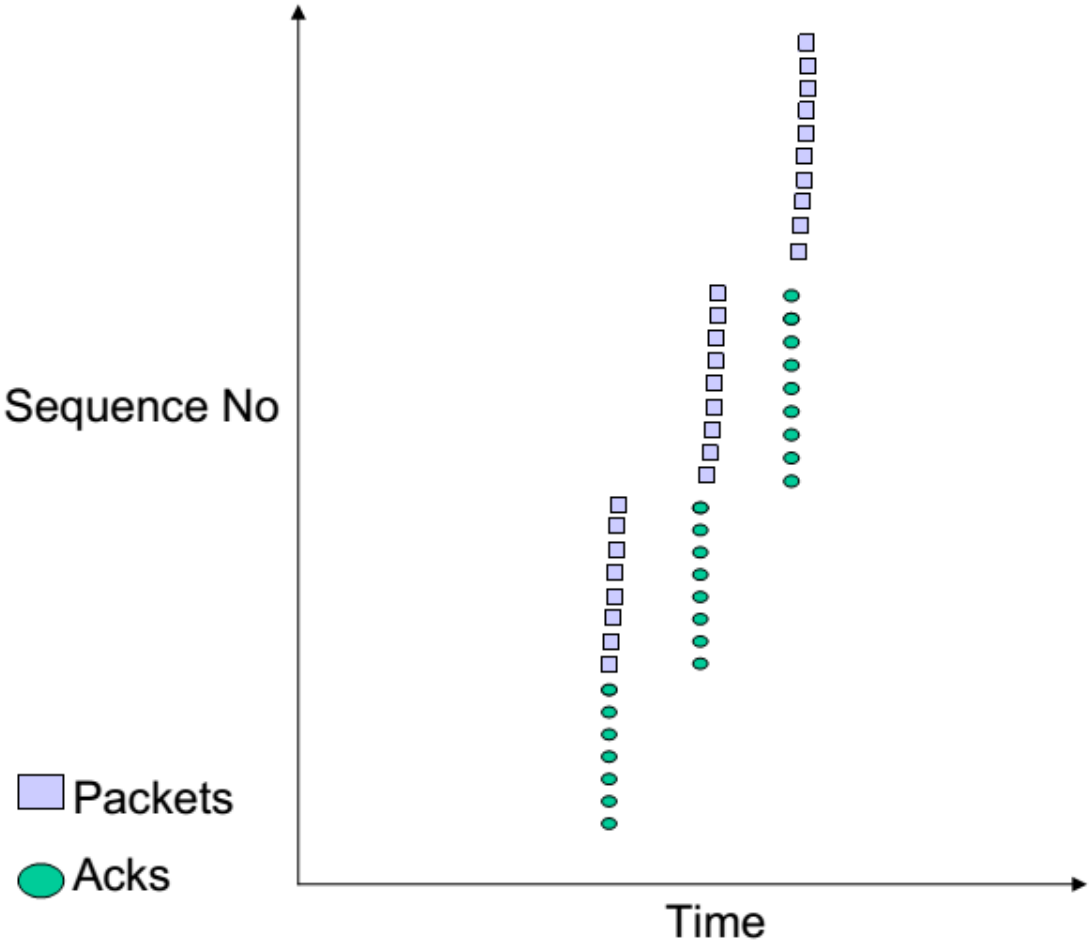
Slow start

- Tercero,
  - Si TCP detecta 3 segmentos con la bandera de ACK activada y cada uno con el mismo número de acknowkledgement, entonces TCP lleva a cabo la técnica de Fast retransmit, y entra al modo de Fast recovery.

Congestion avoidance

- Cuando se entra al modo de Congestion avoidance, el valor de cwnd es igual o mayor al ssthresh
- TCP adopta un comportamiento más “conservador” e incrementa el valor de cwnd tan solo 1 MSS por cada RTT.
- Tras la recepción de ACK
  - Incrementa cwnd = cwnd + 1 / cwnd
- El incremento de cwnd es lineal.

Congestion avoidance



Congestion avoidance

- Casos para detener congestion avoidance:
- Primero
  - Cuando un timeout ocurre presenta el mismo comportamiento que slow start.
  - Establece el valor de ssthresh a la mitad del valor cwnd cuando la congestión fue detectada.
  - El que envía los datos de TCP establece el valor de cwnd en 1 MSS, y comienza el proceso de slow start de nuevo.
  - Cuando cwnd es igual o mayor al valor de ssthresh, se entra al modo de congestion avoidance de nuevo.

Congestion avoidance

- Segundo,
  - Si TCP detecta 3 segmentos con la bandera de ACK activada y cada uno con el mismo número de acknowkledgement.
  - Asigna el valor de ssthresh para que sea la mitad del valor de cwnd cuando se recibieron los ACK duplicados.
  - Inicia el modo de Fast recovery.

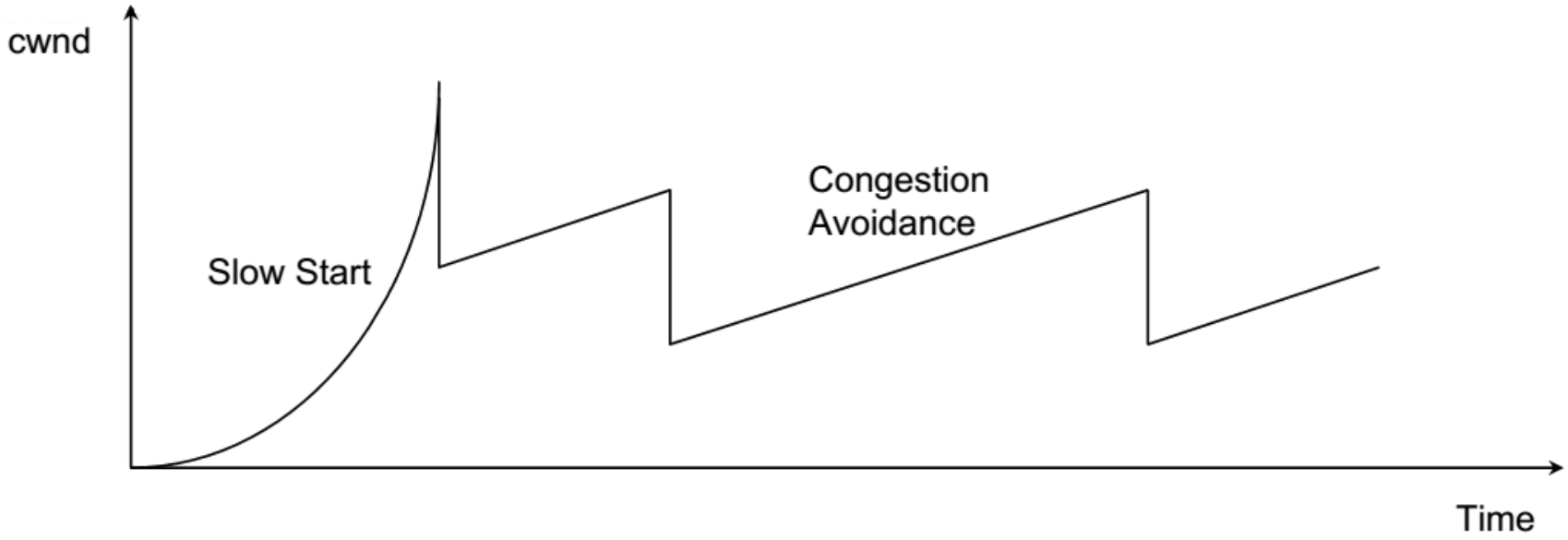
Fast recovery

- Ocurre cuando se repiten ACK para un mismo número de secuencia (dupacks).
- Ocurre por:
  - Pérdidas
  - Reordenamiento de los segmentos
  - Actualización del tamaño de la ventana
- Este tipo de pérdidas es poco frecuente
  - Utiliza la recepción de 3 o más ACK duplicados como indicador de la pérdida de segmentos.
  - No esperará por un timeout para retransmitir el paquete.

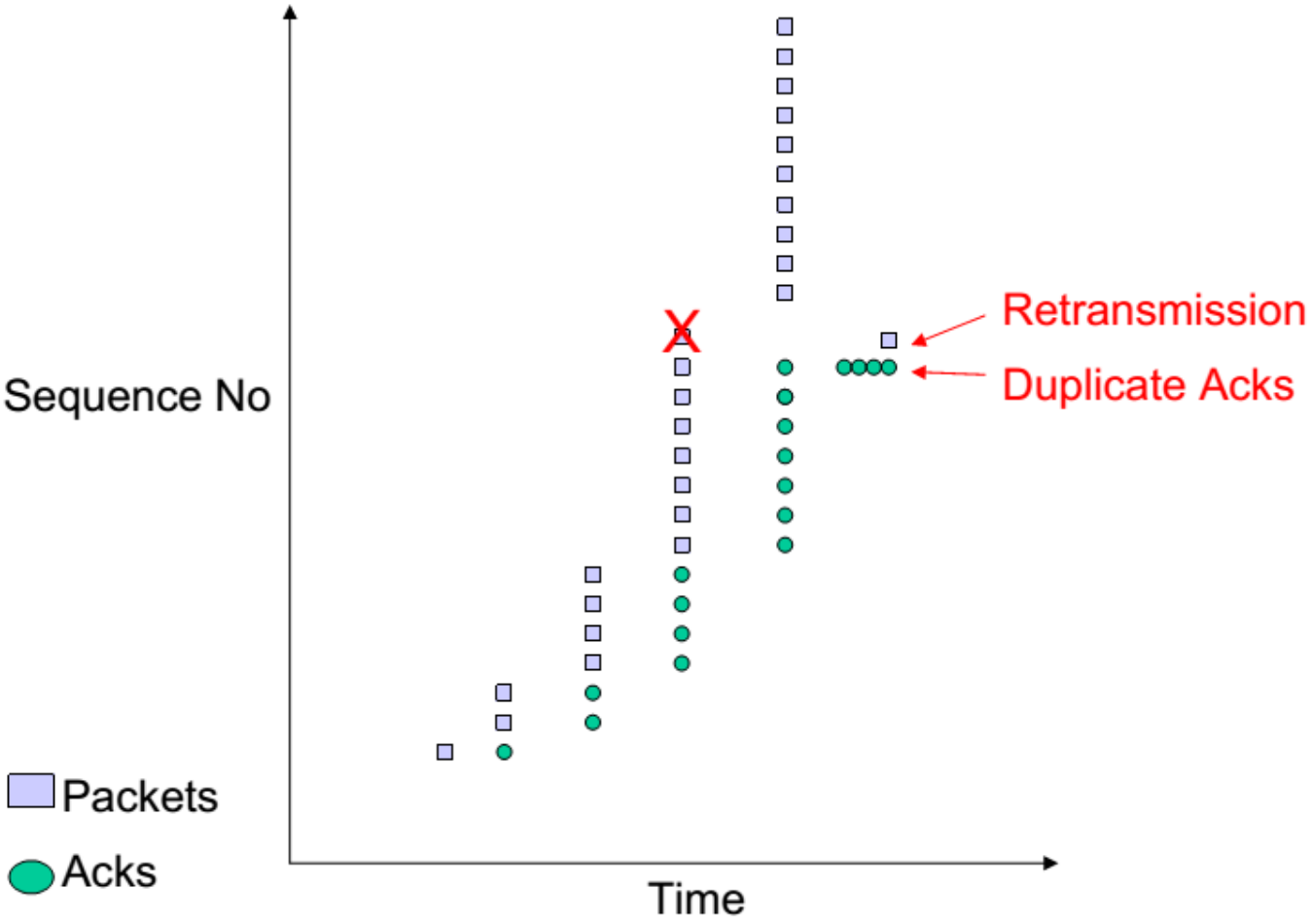
Fast recovery

- El valor de cwnd se incrementa en 1 MSS por cada ACK duplicado recibido para el segmento perdido que causó que TCP ingresará en el modo de Fast recover.
- Eventualmente, cuando llega un ACK para el segmento perdido, TCP entra en el estado de congestion avoidance después de decrementar el cwnd.
- Si se produce un timeout, fast recovery cambia a slow start después de realizar las mismas acciones que en slow start y congestion avoidance: el valor de cwnd se establece en 1 MSS y el valor de ssthresh se establece en la mitad del valor de cwnd cuando el timeout.

Fast retransmit and fast recovery



Fast retransmit



Fast recovery

- De acuerdo al RFC 5681, Fast recovery es recomendado pero no es requerido.

Referencias

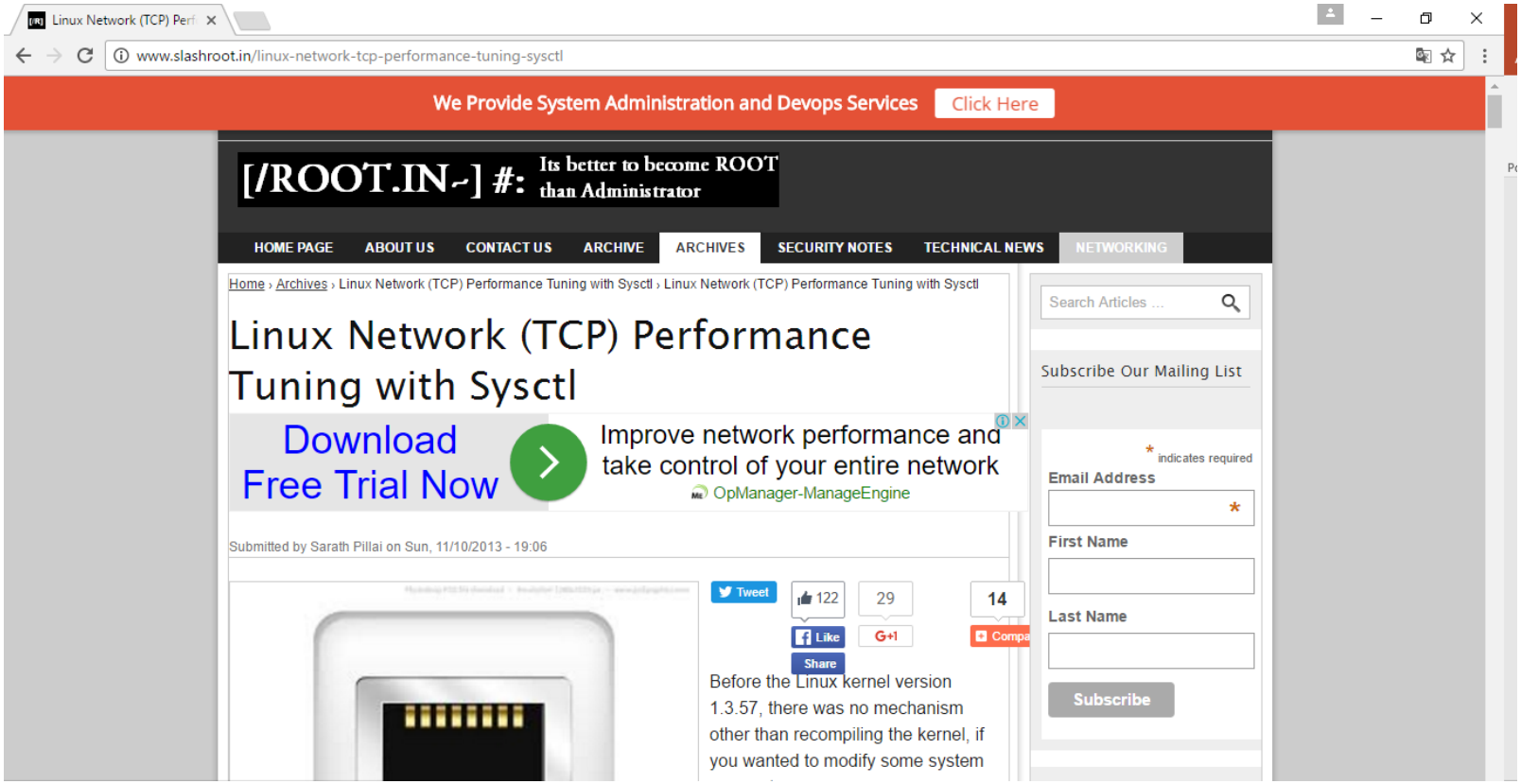
- [1] J. Kurose and K. Ross, *Computer networking*. Boston: Pearson, 2013.
- [2] A. Tanenbaum and D. Wetherall, *Redes de computadoras*. México: Pearson Educación, 2012.
- [3] M. Katz, *Redes y seguridad*. Argentina: Alfaomega, 2013.
- [4] *Request For Comments 791, 793*. IETF.

Referencias

- [5] Javier Gómez Castellanos. Apuntes redes de datos. PCIC, UNAM. 2015-I
- [6] Hui Zhang. Congestion Control, 15-441 Computer Networks. School of Computer Science. Carnegie Melon University.


Thread
iamkirkbater and jkjustjoshing
iamkirkbater Aug 23rd, 2017 at 9:37 AM
in #www
Do you want to hear a joke about TCP/IP?
7
7 replies
jkjustjoshing 5 months ago
Yes, I'd like to hear a joke about TCP/IP
iamkirkbater 5 months ago
Are you ready to hear the joke about TCP/IP?
jkjustjoshing 5 months ago
I am ready to hear the joke about TCP/IP
iamkirkbater 5 months ago
Here is a joke about TCP/IP.
iamkirkbater 5 months ago
Did you receive the joke about TCP/IP?
jkjustjoshing 5 months ago
I have received the joke about TCP/IP.
iamkirkbater 5 months ago
Excellent. You have received the joke about TCP/IP. Goodbye.

Linux Network (TCP) Performance Tuning with Sysctl



http://www.slashroot.in/linux-network-tcp-performance-tuning-sysctl