APUNTES, MANUALES, PRESENTACIONES

Universidad de Jaén

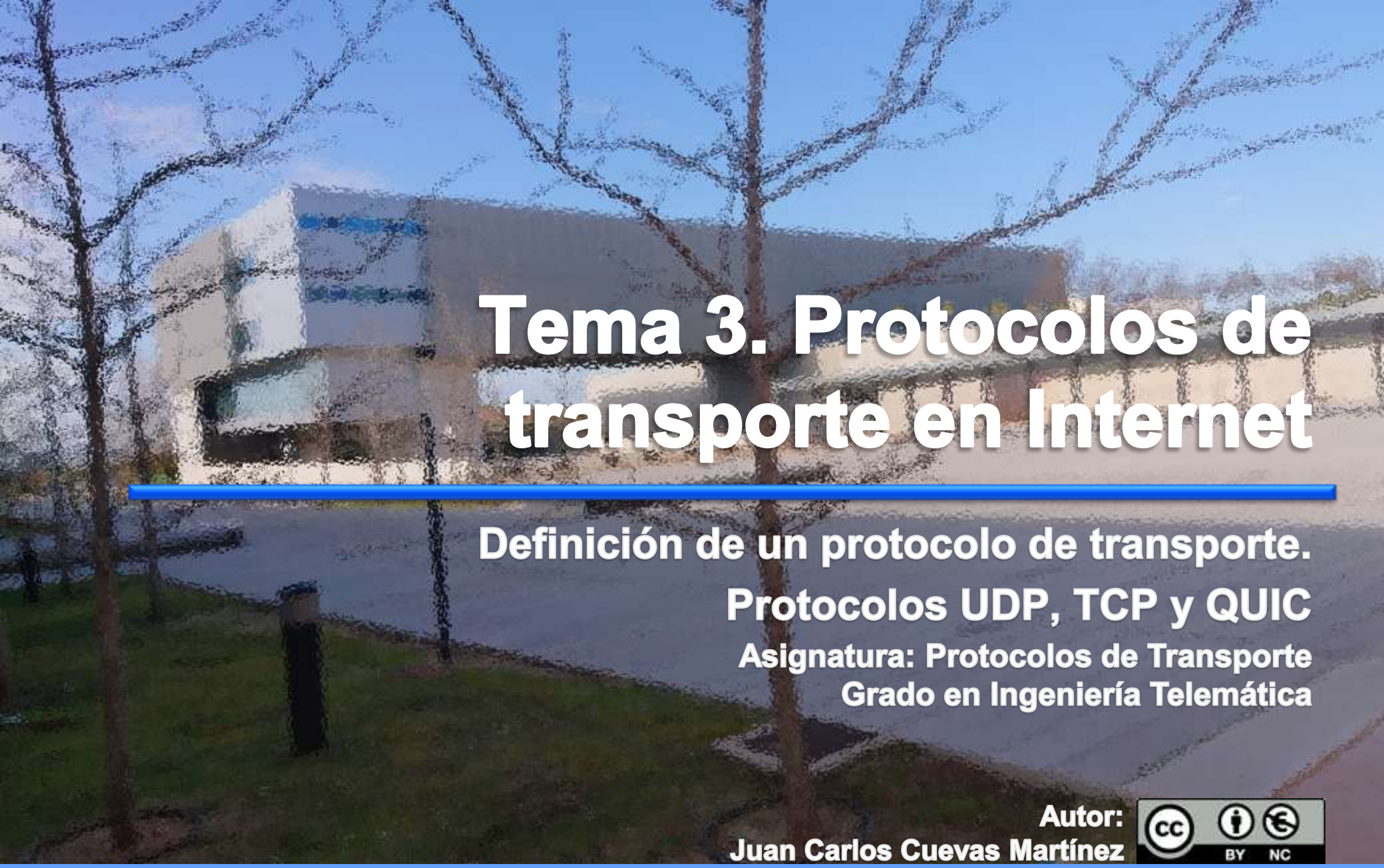
Escuela Politécnica Superior de Linares

Tema 3. Transporte en internet

Autor: Juan Carlos Cuevas Martínez

Fecha: 10/2024

Asignatura: Protocolos de transporte



Tema 3. Protocolos de transporte en Internet

Definición de un protocolo de transporte. Protocolos UDP, TCP y QUIC

Asignatura: Protocolos de Transporte
Grado en Ingeniería Telemática

Autor:
Juan Carlos Cuevas Martínez

Dpto. de Ingeniería de Telecomunicación
Área de Ingeniería Telemática

Universidad de Jaén
Escuela Politécnica Superior de Linares

Versión 6
2.024

Competencias

Código	Descripción
CB2	Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.
CB3	Que los estudiantes tengan la capacidad de reunir e interpretar datos relevantes (normalmente dentro de su área de estudio) para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.
CB4	Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.
CB5	Que los estudiantes hayan desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.
CG4	Capacidad de resolver problemas con iniciativa, toma de decisiones, creatividad, y de comunicar y transmitir conocimientos, habilidades y destrezas, comprendiendo la responsabilidad ética y profesional de la actividad del Ingeniero Técnico de Telecomunicación
CG9	Capacidad de trabajar en un grupo multidisciplinar y en un entorno multilingüe y de comunicar, tanto por escrito como de forma oral, conocimientos, procedimientos, resultados e ideas relacionadas con las telecomunicaciones y la electrónica.

Competencias

Código	Descripción
TEL1	Capacidad de construir, explotar y gestionar las redes, servicios, procesos y aplicaciones de telecomunicaciones, entendidas éstas como sistemas de captación, transporte, representación, procesado, almacenamiento, gestión y presentación de información multimedia, desde el punto de vista de los servicios telemáticos.
TEL4	Capacidad de describir, programar, validar y optimizar protocolos e interfaces de comunicación en los diferentes niveles de una arquitectura de redes
TEL7	Capacidad de programación de servicios y aplicaciones telemáticas, en red y distribuidas

Transporte en Internet

Resultados de aprendizaje

Código	Descripción
3	Se aprenderá a diferenciar, considerando las características de un protocolo de transporte, cuál resulta más conveniente utilizar según los servicios de telecomunicación habituales.
4	El alumno dominará los protocolos más utilizados en la actualidad para resolver la interconexión de redes de diferente naturaleza, estableciendo la diferenciación entre los protocolos vinculados al transporte de información, los relativos al intercambio de información para el encaminamiento y los auxiliares a los dos tipos descritos anteriormente.
26	Adquirir facilidad para el manejo de especificaciones, reglamentos y normas de obligado cumplimiento.
27	Resolver problemas con iniciativa, toma de decisiones y creatividad.
28	Comunicar y transmitir conocimientos, habilidades y destrezas, comprendiendo la responsabilidad ética y profesional de la actividad del Ingeniero Técnico de Telecomunicación.
30	Adquirir facilidad para el manejo de especificaciones, reglamentos y normas de obligado cumplimiento.
34	Trabajar en un grupo multidisciplinar y en un entorno multilingüe.
35	Comunicar, tanto por escrito como de forma oral, conocimientos, procedimientos, resultados e ideas relacionadas con las telecomunicaciones y la electrónica.

Contenido

1. **Fundamentos de la capa de transporte.**
2. **Protocolos de transporte en Internet.**
3. **Protocolo de Datagrama de Usuario (UDP).**
   1. **Características.**
   2. **Formato de PDU.**
4. **Protocolo de Control de Transmisión (TCP).**
   1. **Características.**
   2. **Formato de PDU.**
   3. **Establecimiento de conexión.**
   4. **Máquina de estados.**
   5. **Control de congestión.**
   6. **Ampliaciones.**
5. **Protocolo QUIC.**

Si aparece este símbolo es contenido informativo, no evaluable.

Contenido

Bibliografía básica

- TANENBAUM, Andrew S. Computer Networks, Global Edition. Pearson Universidad, 2021. ISBN 9781292374062.
- COMER, Douglas E. *Internetworking with TCP/IP Volume One: Principles, Protocols, and Architecture: 1*. Financial Times Prentice Hall, 2013. ISBN 978-0136085300.
- FOROUZAN, Behrouz A. *TCP/IP protocol suite*. 4ª ed. Dubuque, IA: McGraw-Hill, 2010. ISBN 9780073376042.
- Reference For Comments (**RFC**). IETF

Contenido

Bibliografía complementaria

- HALL, Eric A. *Internet core protocols: The definitive guide*. Cambridge, Mass: O'Reilly, 2000. ISBN 9781565929531.
- KOZIEROK, Charles. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. No Starch Press, 2005. ISBN 9781593270476.
- KUROSE, James F. y Keith W. ROSS. *Computer Networking. A top-Down Approach*. Pearson, 2022. ISBN 978-1-292-40546-9.
- STALLINGS, William. *Fundamentos de Seguridad En Redes*. Pearson Publications Company, 2004. ISBN 9788420540023.
- STENBERG, Daniel. HTTP/3 Explained. *https://http3-explained.haxx.se/* [en línea]. [sin fecha] [consultado el 23 de mayo de 2022]. Disponible en: https://http3-explained.haxx.se/

Sitios web de visita recomendada

Internet Assigned Numbers Authority (IANA)
www.iana.org

The Internet Engineering Task Force (IETF)
www.ietf.org

Internet Society
www.isoc.org

Internet Corporation for Assigned Names and Numbers (ICANN)
www.icann.org/
www.icann.org/tr/spanish.html

1. Fundamentos de la capa de transporte

Objetivo del nivel de transporte

- Los objetivos fundamentales del nivel de transporte son ofrecer un servicio de transmisión de datos entre procesos en máquinas remotas:
  - Eficiente.
  - Confiable.
  - Económico.
- El software (o hardware) que se encarga de las funciones de la capa de transporte se denomina **Entidad de Transporte.**

1. Fundamentos de la capa de transporte

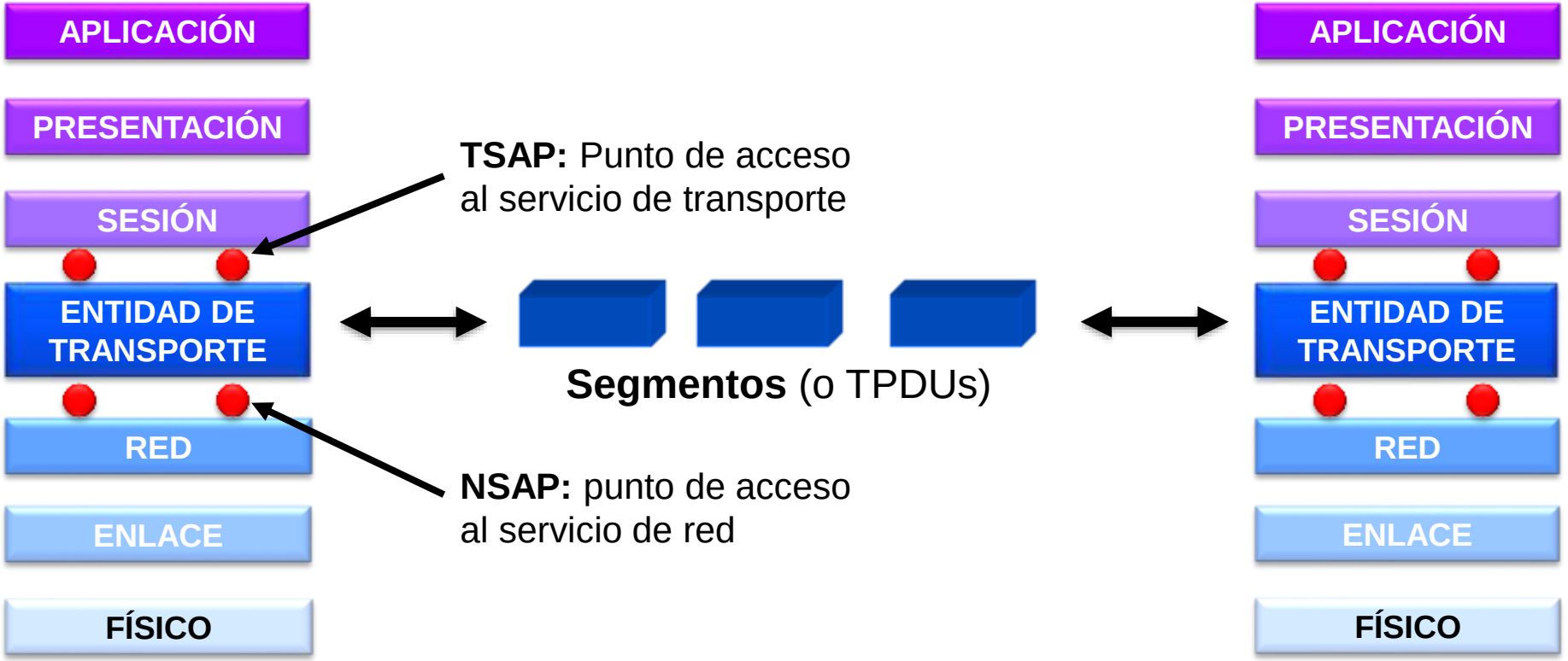
Modelo de referencia OSI[1]

- Es el estándar ISO/IEC 7498-1 de la Organización Internacional para la Estandarización (ISO, del inglés *International Organization for Standardization*)
  - Publicado en 1984 (última revisión consta del año 1994).
  - Elaborado en colaboración con la UIT[2].
    - Recomendación X-200, última versión la X.200 (1994 E).

1: Open Systems Interconnection.
2: La UIT (International Telecommunication Union, ITU), es el organismo especializado en las TICs de la ONU.

1. Fundamentos de la capa de transporte

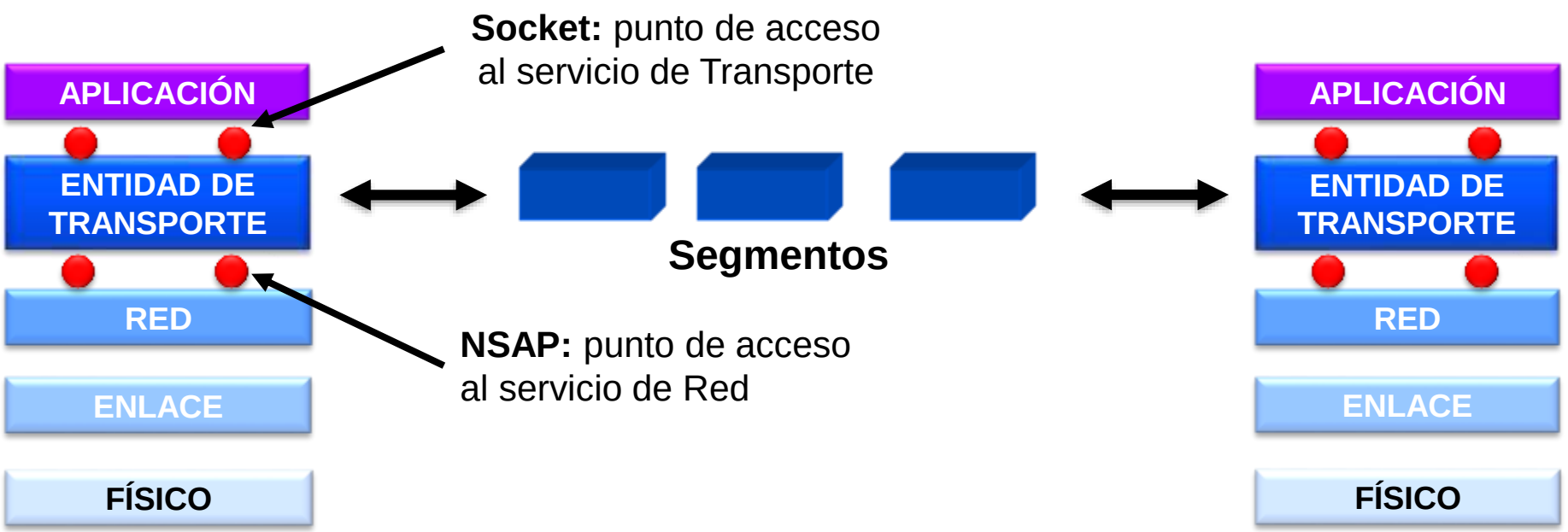
Modelo de referencia OSI



TPDU: Unidad de datos del Servicio de Transporte

1. Fundamentos de la capa de transporte

Modelo de referencia de Internet

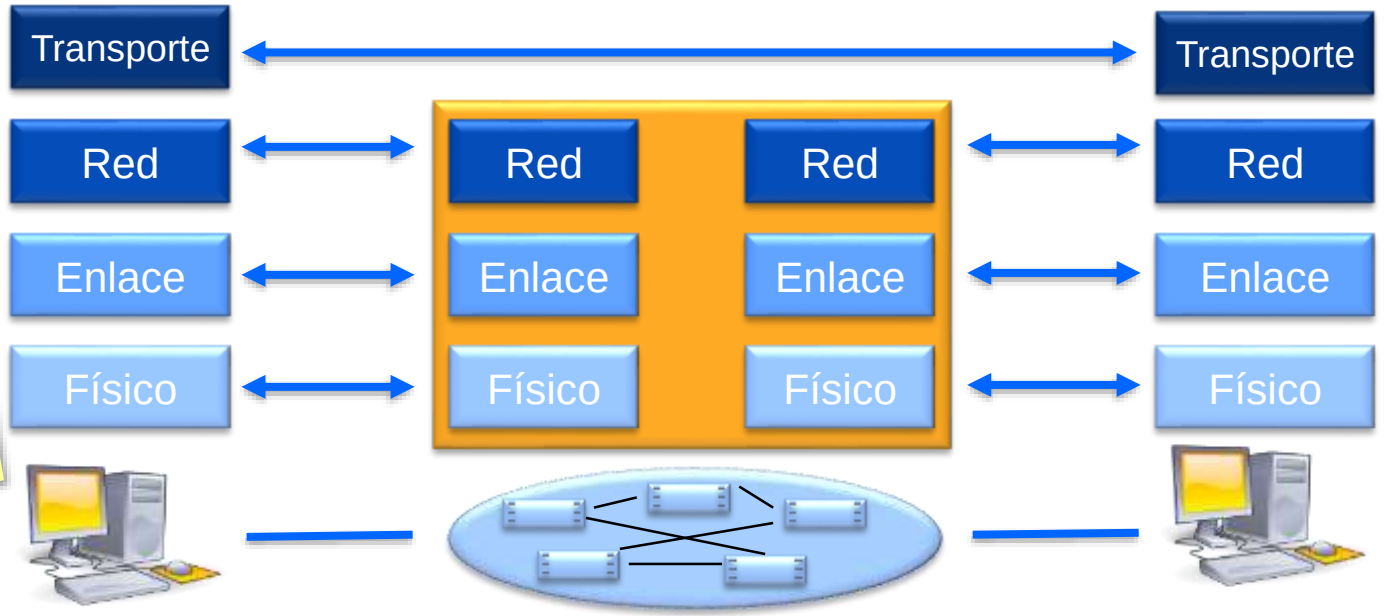


1. Fundamentos de la capa de transporte

Características fundamentales del nivel de transporte

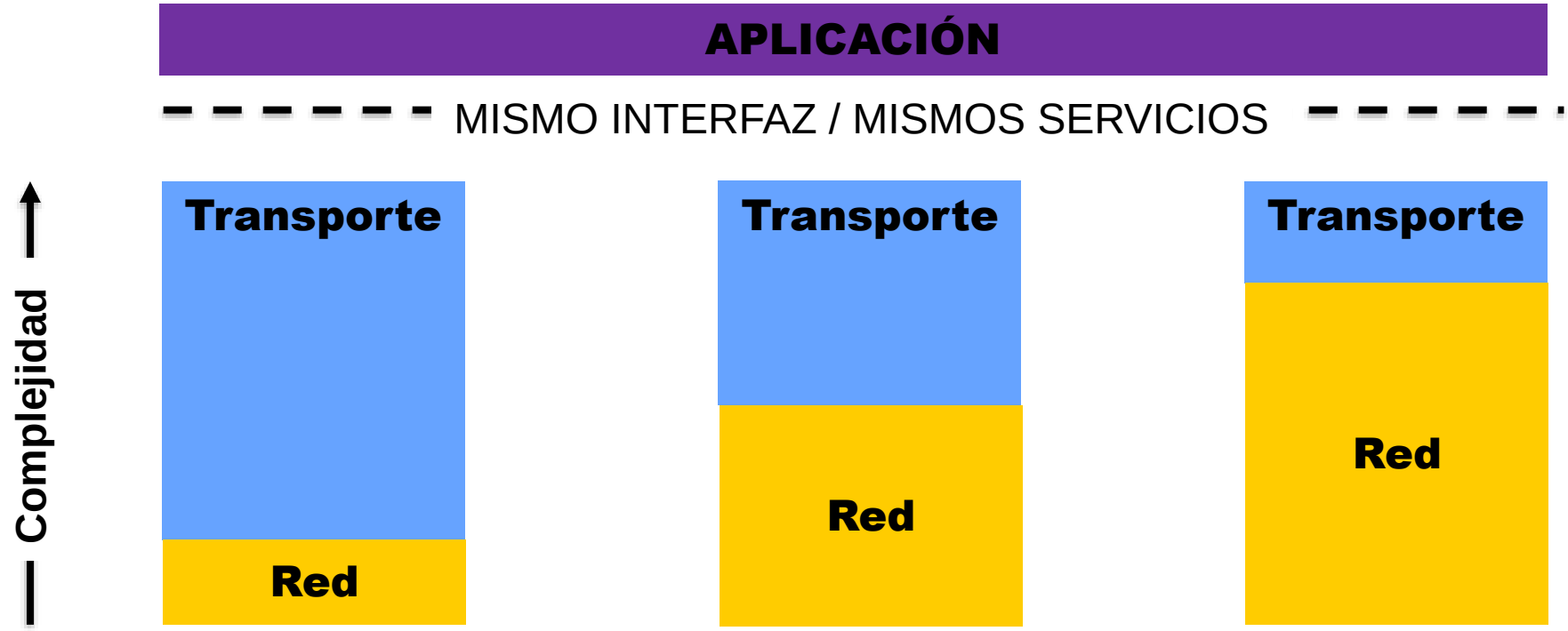
- La entidad de transporte debe ser independiente de la red.
  - Es extremo a extremo.
  - Es parte del kernel (núcleo) del sistema operativo del sistema anfitrión.

La entidad de transporte es un proceso más entre el conjunto de servicios de comunicaciones de red que tiene un sistema con capacidad de comunicación de datos.



1. Fundamentos de la capa de transporte

Complejidad del nivel de transporte



La capa de transporte deberá tener mayor complejidad en redes que proporcionen menos servicios si quiere ofrecer a la aplicación el mismo interfaz de servicios.

1. Fundamentos de la capa de transporte

Servicios

- Servicio fundamental: transporte de datos extremo a extremo.
  - Orientado a conexión: requiere establecimiento, transferencia y liberación de la conexión.
  - Sin conexión: tan solamente aporta los medios mínimos para el intercambio de datos entre los extremos.
- Usuario de los servicios de la capa de transporte:
  - Modelo de referencia de Internet: capa de aplicación.
  - Modelo de referencia OSI: capa de sesión.

1. Fundamentos de la capa de transporte

Servicios

Acceso a los servicio - Primitivas

- Para acceder a los servicios de transporte, el usuario debe emplear un conjunto de funciones de interfaz, denominadas primitivas.
- Cada servicio de transporte implementará una interfaz de primitivas diferentes.
- Servicios típicos:
  - Servicios orientados a conexión.
  - Servicios no orientados a conexión.

1. Fundamentos de la capa de transporte

Acceso a los servicios de transporte - Primitivas

Ejemplo de primitivas

Servicio orientado a conexión

PRIMITIVA	MENSAJE ENVIADO	SIGNIFICADO
ESCUCHAR	Ninguno	Se bloquea hasta que algún proceso intenta conectarse
CONECTAR	PETICIÓN_CONECTAR	Intenta activamente una conexión enviando un mensaje de petición al otro extremo
ENVIAR	DATOS	Envía información al otro extremo
RECIBIR	Ninguno	Se bloquea hasta que se reciben datos
DESCONECTAR	PETICIÓN_DESCONECTAR	Solicita que se libere la desconexión, borrando toda la información dependiente de ésta en ambos extremos.

Ejemplo adaptado de “Redes de Computadoras”, 5 ed. de A. S. Tanenbaum

1. Fundamentos de la capa de transporte

Acceso a los servicios de transporte - Primitivas

Ejemplo de primitivas

Servicio NO orientado a conexión

PRIMITIVA	MENSAJE ENVIADO	SIGNIFICADO
ENVIAR	DATOS	Envía información al otro extremo
RECIBIR	Ninguno	Se bloquea hasta que se reciben datos

1. Fundamentos de la capa de transporte

Acceso a los servicios de transporte - Primitivas

Primitivas Sockets

Comunes - Creación, vinculación y destrucción de sockets

Primitiva	Descripción
`socket`	Crea un descriptor de socket
`close`	Cierra socket
`bind`	Asocia una dirección local con un socket

1. Fundamentos de la capa de transporte

Acceso a los servicios de transporte - Primitivas

Primitivas Sockets

Servicio orientado a conexión

Primitiva	Descripción
`listen`	Crea una cola de espera para almacenar solicitudes de conexión
`accept`	Espera una solicitud de conexión
`connect`	Inicia conexión con un extremo remoto
`shutdown`	Deshabilita al recepción y/o el envío de datos por le socket
`send`	Envía datos
`recv`	Recibe datos

1. Fundamentos de la capa de transporte

Acceso a los servicios de transporte - Primitivas

Primitivas Sockets

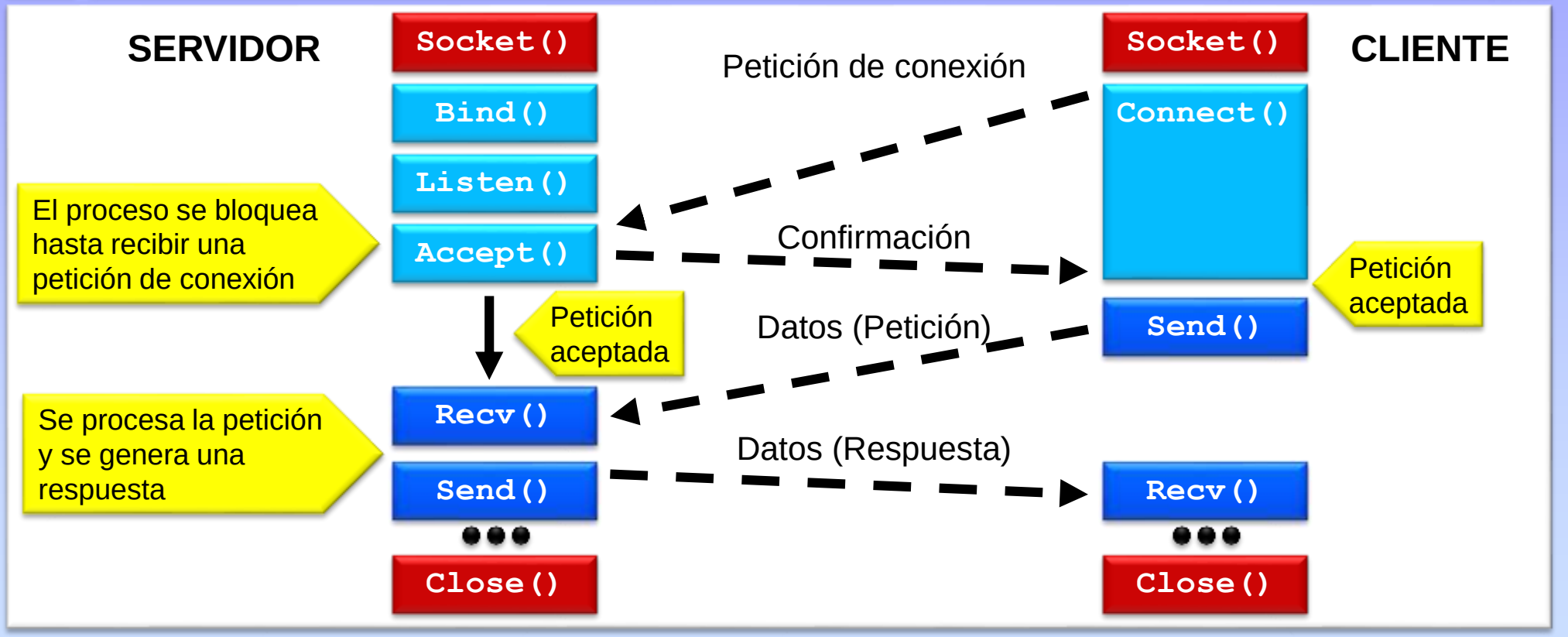
Servicio no orientado a conexión

Primitiva	Descripción
`sendto`	Envía un mensaje
`recvfrom`	Recibe un mensaje

1. Fundamentos de la capa de transporte

Acceso a los servicios de transporte - Primitivas

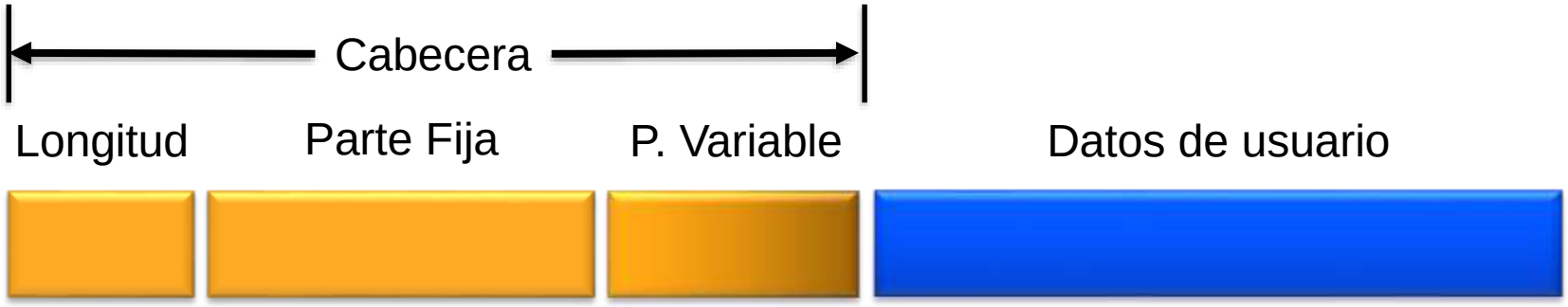
Ejemplo de conexión con primitivas sockets



1. Fundamentos de la capa de transporte

Unidades de Datos del protocolo de Transporte

- Las unidades de datos de transporte se denominan SEGMENTOS (también se las puede referir como TPDU – *Transport Protocol Data Unit*).
  - Hay definiciones de protocolos que cambian esta denominación.
- Formato general.
  - Campos típicos:
    - Indicador de longitud de cabecera.
    - Parte fija de la cabecera.
    - Parte variable de cabecera: normalmente incluye opciones del protocolo.
    - Datos.



1. Fundamentos de la capa de transporte

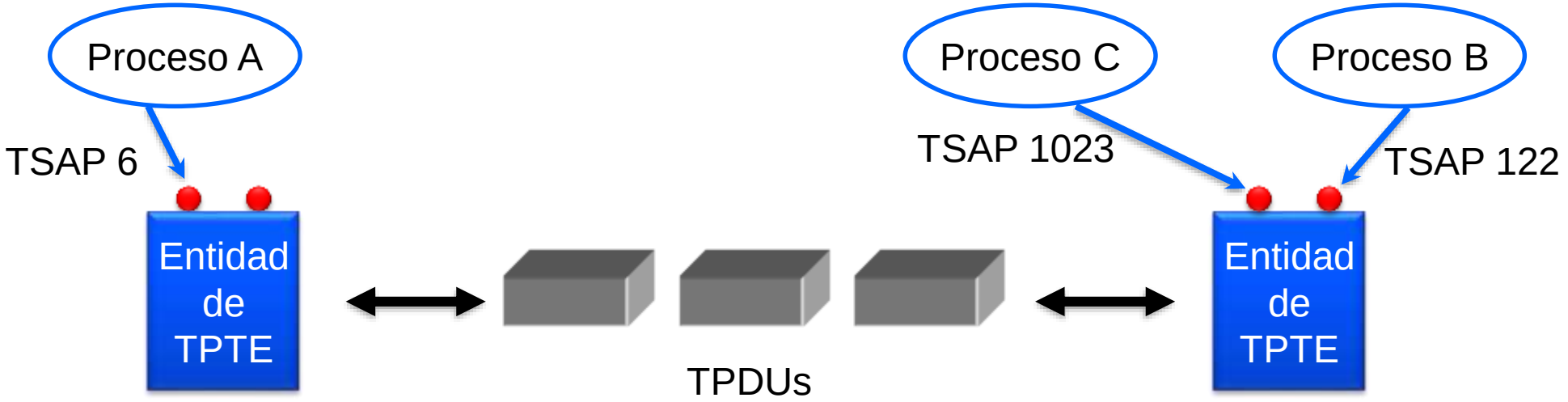
Elementos de la capa de transporte

- La capa de transporte debe superar una serie de problemas que no aparecen en capas inferiores:
  - El nivel de red modela todas las características de la red física, por lo que **la capa de transporte debe ser independiente de la capa de red**.
  - Existen similitudes entre algunas de las características del nivel de transporte y el de enlace (control de flujo, secuencia, control de errores), pero las diferencias son importantes:
    - Las conexiones de transporte necesitan un enrutamiento explícito del destino.
    - En las conexiones de transporte existe la posibilidad de almacenamiento fuera de su control en los niveles inferiores.
    - Se necesitan mayor número de buffers debido al mayor número de conexiones.

1. Fundamentos de la capa de transporte

Direccionamiento

- El método básico que se emplea es definir puntos de acceso al servicio de transporte (*Transport Service Access Point* - TSAP), dotados de dirección, a los que se pueden asociar procesos para establecer conexiones.



1. Fundamentos de la capa de transporte

Direccionamiento

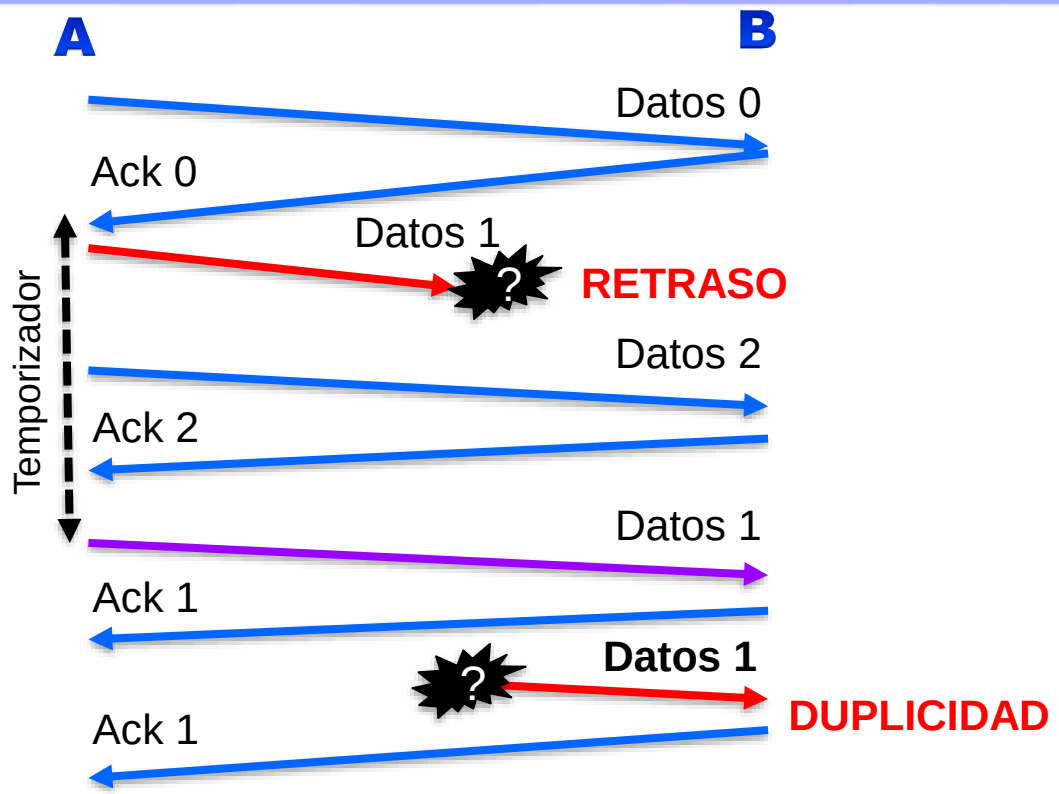
- La asociación de los procesos a los TSAP es realizada por el sistema operativo.
- Pero, ¿Cómo se puede conocer el nº de TSAP destino?
  - No se conocen a priori.
  - No son permanentes (procesos que se ejecutan durante un tiempo).
  - No se puede reservar permanentemente un TSAP a un proceso ya que los recursos de las máquinas son limitados.
- Solución adoptada en Internet, los *Well Known Ports*: cada servicio se vincula a un TSAP preestablecido (en Internet se denomina **puerto**).
  - Este proceso está regulado por el IANA.
  - En los inicios de ARPANET (precursora de Internet) había un protocolo de conexión inicial.

1. Fundamentos de la capa de transporte

Administración de conexiones de transporte

Paquetes duplicados y retardados

- En redes **inseguras** pueden existir:
  - Pérdidas de segmentos.
  - Retrasos variables.
  - Duplicidad de segmentos.
- El caso peor son los **DUPLICADOS RETARDADOS.**



1. Fundamentos de la capa de transporte

Administración de conexiones de transporte

Paquetes duplicados y retardados

Posibles soluciones

- Direcciones TSAP desechables:
  - No permite servidor de procesos o direcciones.
  - No permite la existencia de servicios en puertos conocidos.
  - Agotamiento de los recursos en servicios con alta demanda.
- Identificador de conexión:
  - Número de secuencia para cada conexión que se incrementa con cada nueva conexión.
  - Cuando finaliza la conexión se registra en una lista de conexiones obsoletas.
  - Problemas:
    - Necesita información histórica (más recursos).
    - Si un host cae puede perder toda esa información.

1. Fundamentos de la capa de transporte

Administración de conexiones de transporte

Paquetes duplicados y retardados

Posibles soluciones

- Eliminar segmentos de la red duplicados:
  - Marcas de tiempo.
    - Problemas se sincronía.
  - Tiempo de vida de mensajes en la red.
    - Normalmente se emplea un conteo de saltos.
  - Combinaciones de identificador y tiempo de vida.
- Administración de conexiones basadas en temporizador.
  - Se establece un algoritmo para la generación del número de secuencia inicial a partir del reloj (normalmente los msb de la hora).
  - También se establecen zonas prohibidas.
  - Empleado también en TCP (RFC 6528)
- Protocolo Ida – Vuelta – Ida.

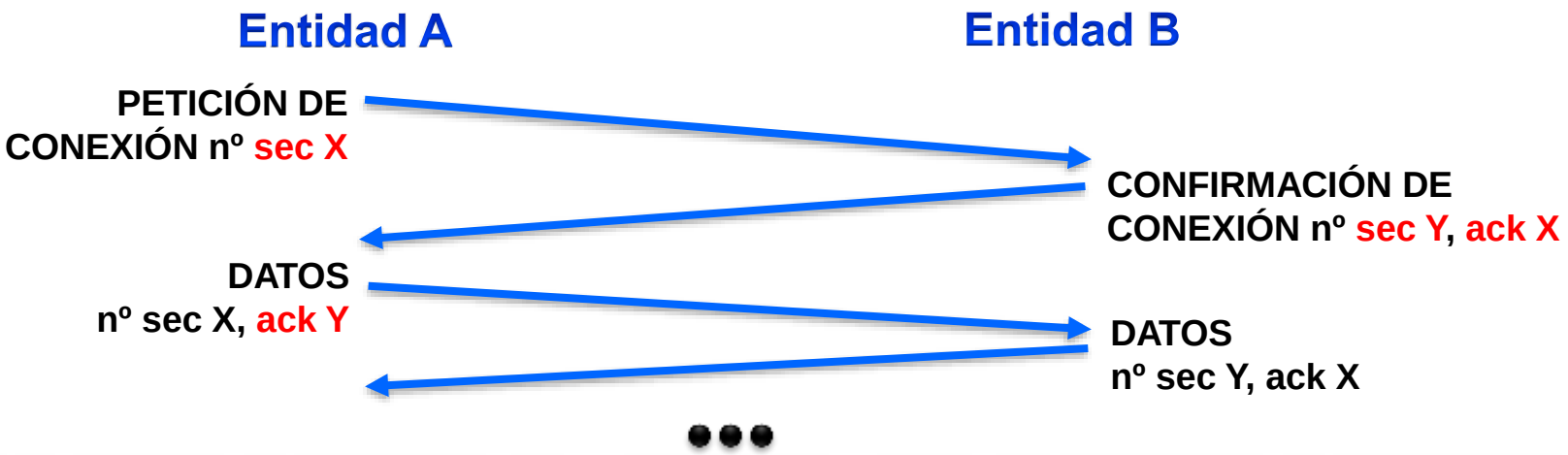
1. Fundamentos de la capa de transporte

Administración de conexiones de transporte

Paquetes duplicados y retardados

Protocolo IDA-VUELTA-IDA

- Es difícil que las dos entidades se pongan de acuerdo para imponer el número de conexión además de los problemas de las conexiones basadas en temporizador.
- El protocolo IDA-VUELTA-IDA o a TRES VÍAS (*Three way handshake*, Tomlinson 1975) propone una numeración independiente para cada entidad.

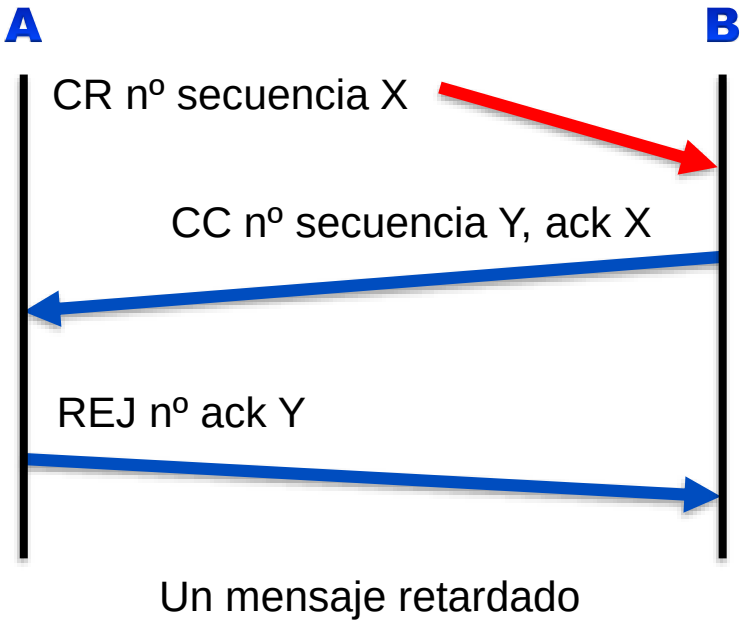


1. Fundamentos de la capa de transporte

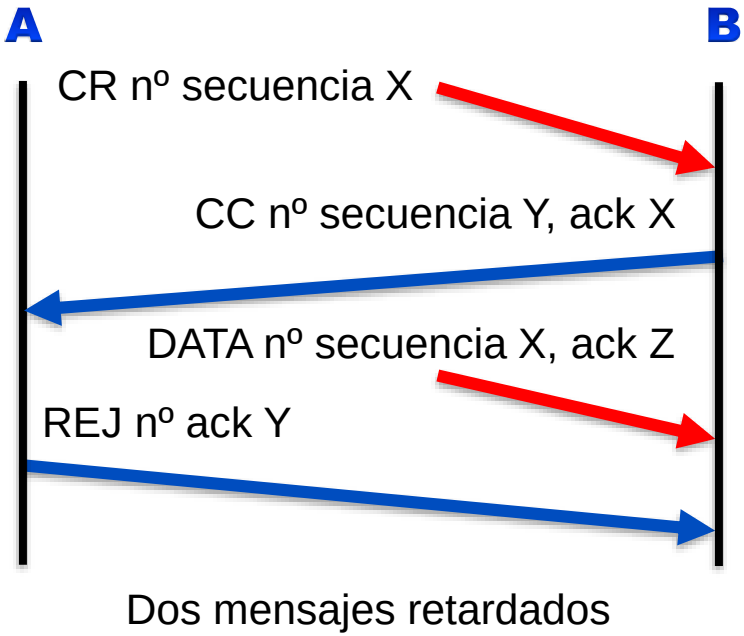
Administración de conexiones de transporte

Paquetes duplicados y retardados

Protocolo IDA-VUELTA-IDA. Ejemplo de duplicados



Un mensaje retardado



Dos mensajes retardados

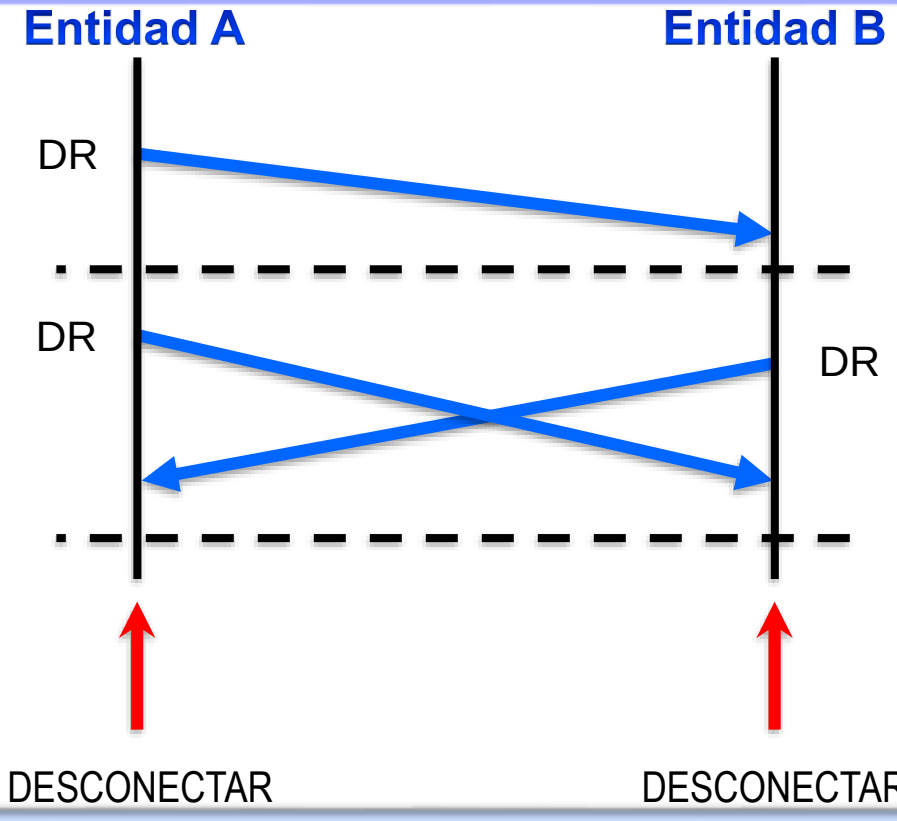
CR: Connection Request, CC: Connection Confirm, REJ: Reject

1. Fundamentos de la capa de transporte

Administración de conexiones de transporte

Liberación de conexión

- Pueden ser:
  - Iniciada por un usuario.
  - Iniciada por los dos usuarios.
  - Iniciada por la entidad de transporte.



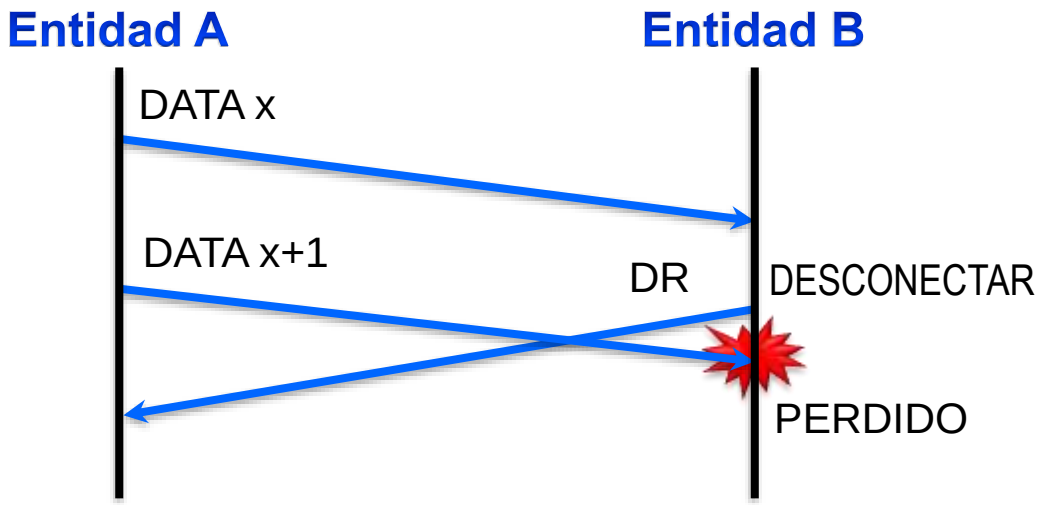
DR: Disconnect Request

1. Fundamentos de la capa de transporte

Administración de conexiones de transporte

Liberación de conexión

- Las liberaciones abruptas pueden provocar perdidas de datos.
- Las liberaciones deben realizarse cuando las dos entidades se encuentren de acuerdo, y **NO EXISTAN DATOS PENDIENTES**, pero es difícil de implementar.

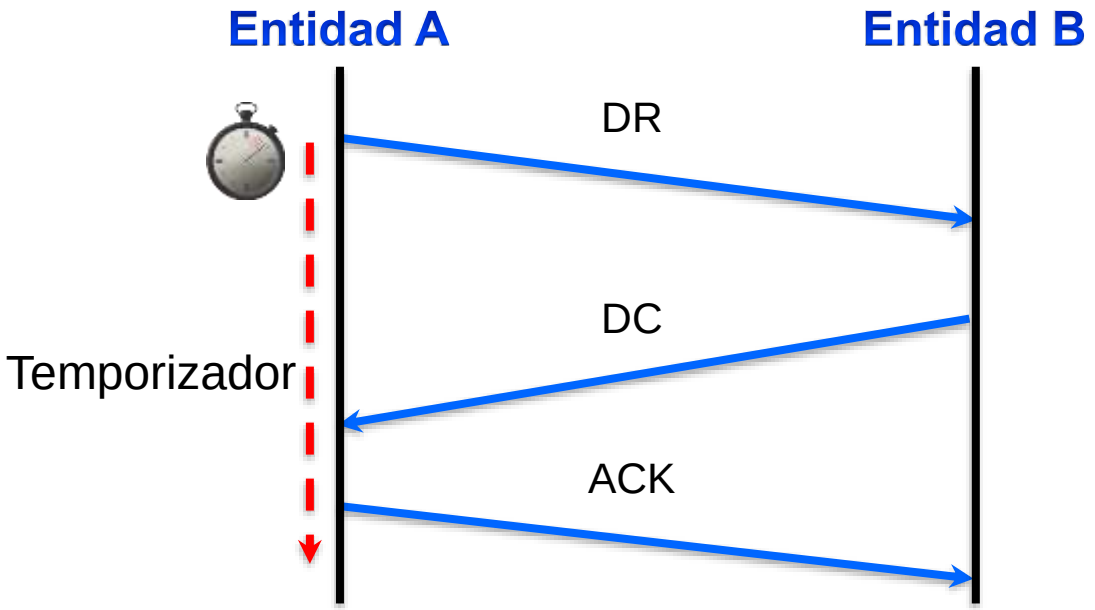


1. Fundamentos de la capa de transporte

Administración de conexiones de transporte

Liberación de conexión

- La solución más común está basada en la utilización de:
  - Asentimiento de liberación.
  - Temporizadores.



DR: Disconnect Request
DC: Disconnect Confirm
ACK: Acknowledgement

1. Fundamentos de la capa de transporte

Administración de conexiones de transporte

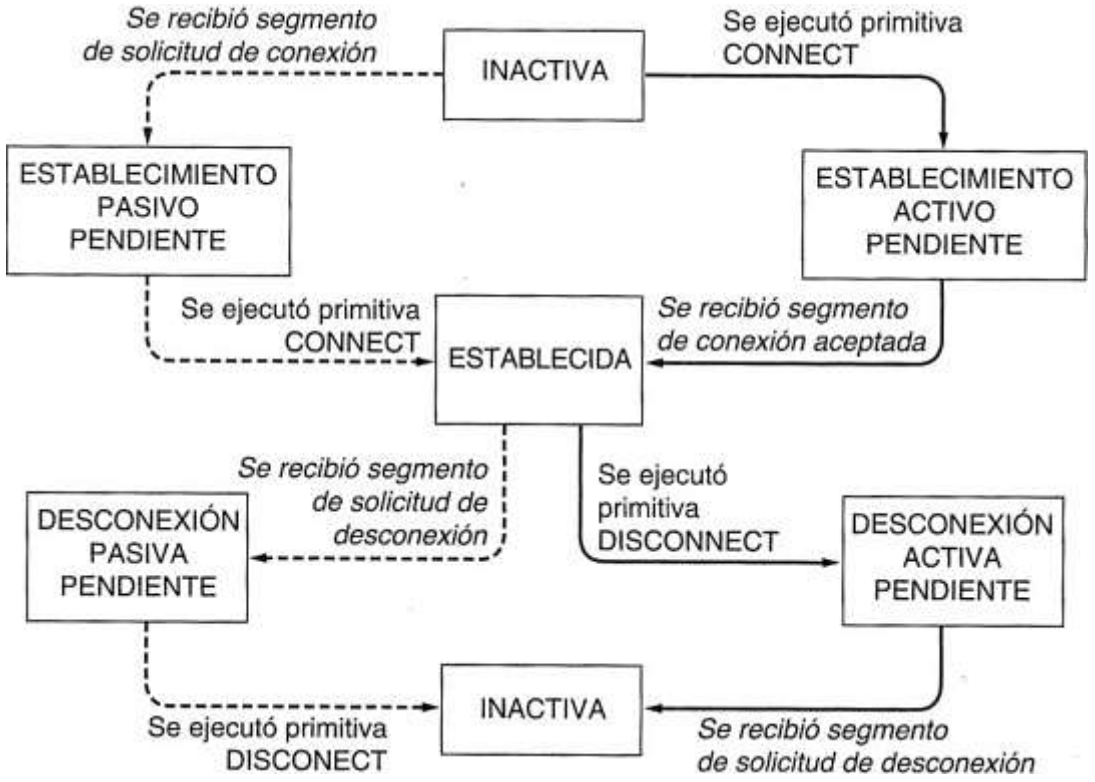
Administración de conexiones basadas en temporizador

- Fundamento: **Liberar conexión si no existe actividad.**
- Cuando se desea enviar un *flujo consecutivo* de segmentos se crea un registro de conexión:
  - Número de secuencia de segmentos enviados.
  - Asentimientos recibidos.
  - Temporizador de actividad.
- Si el temporizador vence, se elimina la conexión y el registro.

1. Fundamentos de la capa de transporte

Administración de conexiones de transporte

Máquina de estados simple de un protocolo de transporte



Fuente: "Redes de Computadoras", 5 ed. de A. S. Tanenbaum

1. Fundamentos de la capa de transporte

Control de flujo

Finalidad y características

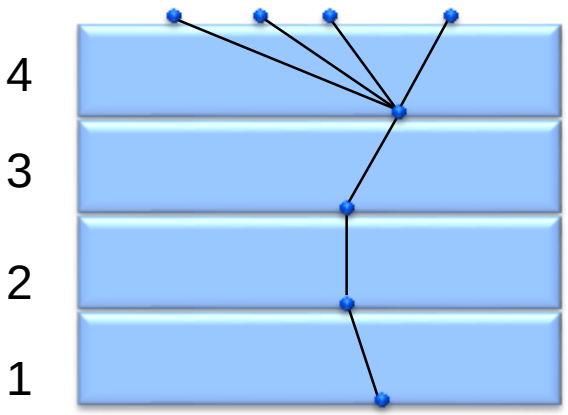
- El propósito del control de flujo es **evitar que los emisores rápidos saturen a receptores lentos**.
- Hay que tener en cuenta la cantidad de conexiones que pueden establecerse, (diferencia con respecto a enlace), y que los recursos de almacenamiento son limitados.
  - Se deben tener estrategias adecuadas para el almacenamiento de los datos en memoria, tales como asignación de registros o memoria dinámica.
- Los segmentos enviados deben almacenarse para posibles retransmisiones por errores en la red y control de flujo.
  - En redes inseguras hay que almacenar los segmentos enviados.
  - En redes seguras, en principio no habría que almacenar, pero la capacidad del receptor puede ser pequeña, por lo que también hay que almacenar los segmentos.

1. Fundamentos de la capa de transporte

Multiplexación

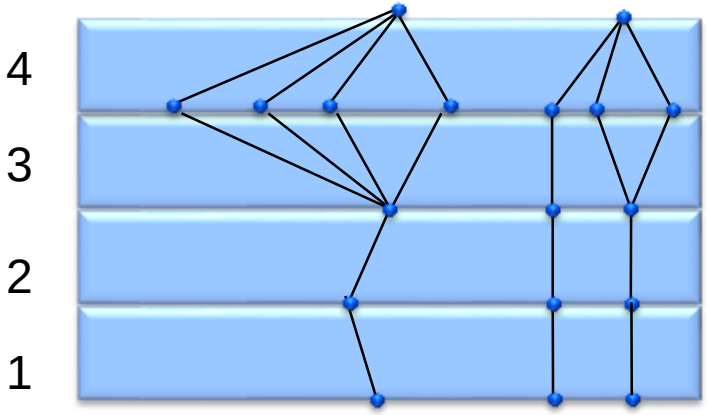
- Son posibles dos tipos de multiplexación de transporte:

Multiplexación Ascendente



- Varias conexiones de transporte comparten la misma de red.
- Ancho banda y rendimiento pequeño.

Multiplexación Descendente



- 1 conexión de transporte utiliza varias conexiones de red.
- Mayor ancho de banda (Limitado por niveles inferiores).

1. Fundamentos de la capa de transporte

Llamadas a procedimiento remoto (RPC[1])

- Permiten **ejecutar procesos en una máquina remota** a través de una interfaz como la de un lenguaje de programación.
- Surgieron de la idea de interpretar las peticiones y respuestas de los protocolos como una llamada a una función (petición) y su resultado (respuesta del servidor). [Birrel y Nelson, «Implementing Remote Procedure Calls» 1984]

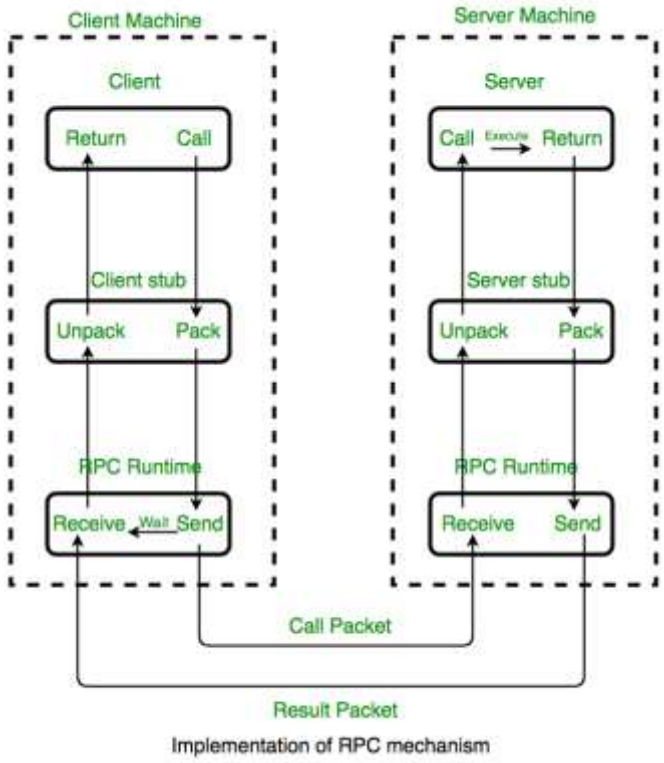
1: RPC, del inglés, *Remote Procedure Call*.

1. Fundamentos de la capa de transporte

Llamadas a procedimiento remoto (RPC)

Características

- El intercambio de mensajes entre el proceso cliente y el servidor es transparente.
- Su objetivo es que un RPC sea lo más parecido a una llamada a procedimiento local.
- Ejemplos: SOAP, Network File System (NFS emplea RPC) o gRPC.



Implementation of RPC mechanism

Fuente: https://www.geeksforgeeks.org/operating-system-remote-procedure-call-rpc/

1. Fundamentos de la capa de transporte

Llamadas a procedimiento remoto (RPC)

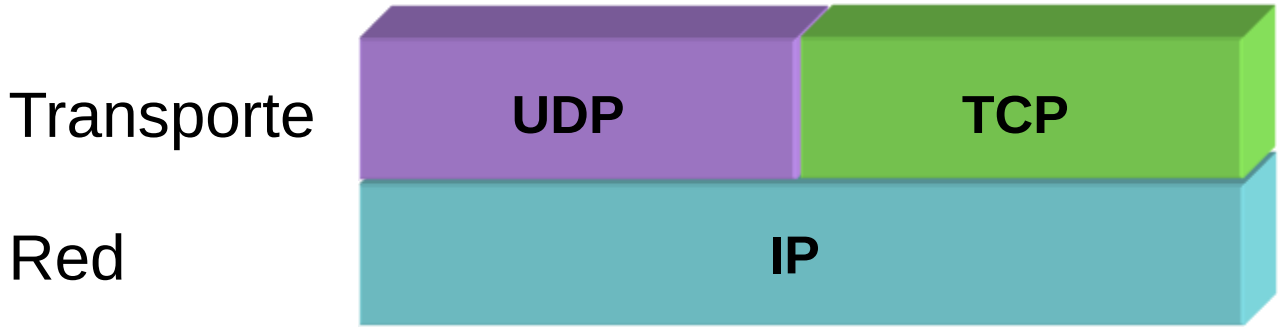
Desventajas

- Manejo de punteros (referencias a direcciones de memoria locales).
- Definición de tipos flexibles: ejemplo arrays en C.
- Variables globales.
- Definición de parámetros variable: ejemplo printf() de C.

2. Protocolos de transporte en Internet

Introducción

- En Internet los dos protocolos de transporte más usados son:
  - **UDP (User Datagram Protocol):** protocolo de transporte sin conexión.
  - **TCP (Transmission Control Protocol):** protocolo orientado a conexión.

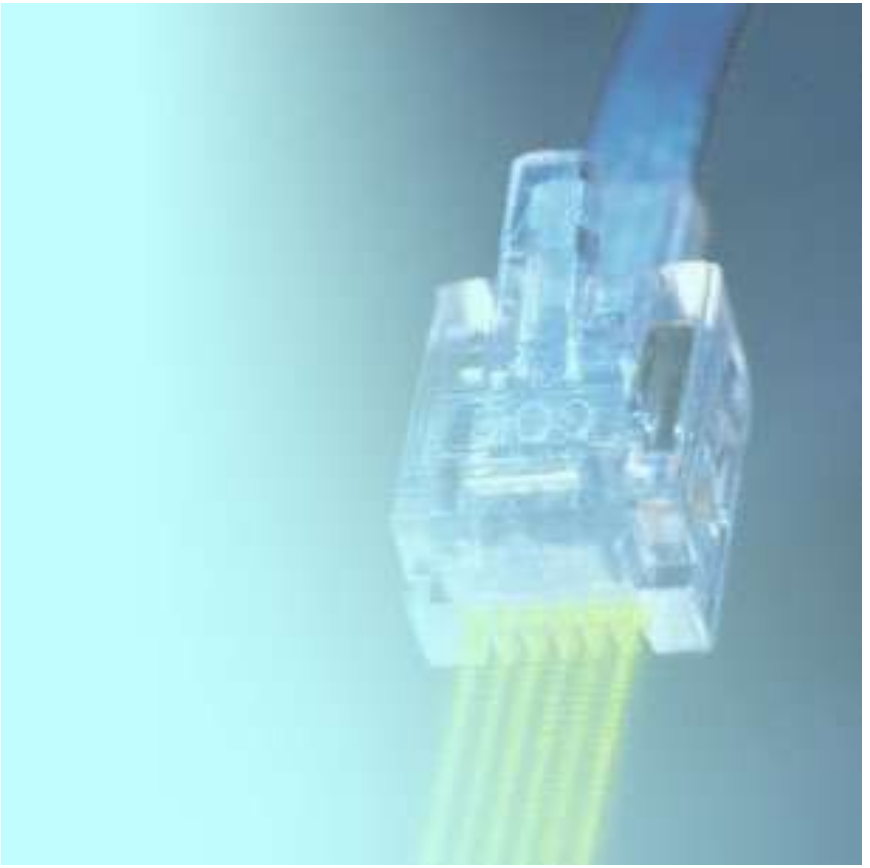


  - Otros protocolos de transporte: RTP (Real Time Protocol), SCTP (Stream Control Transport Protocol) y QUIC.

3. Protocolo de Datagrama de Usuario (UDP)

Introducción

- UDP proporciona un servicio con baja sobrecarga de cabecera para protocolos de aplicación (y de transporte) que no necesiten, o no puedan usar, los servicios orientados a conexión de TCP.
- UDP es frecuentemente usado en aplicaciones que hacen un uso intensivo de difusión (*broadcasts*) y multidifusión (*multicast*), así como aplicaciones que necesitan tiempos cortos para el envío de una petición y la obtención de información.
- **`Identificador de protocolo: 17`**
- **`Estándar: 6 (RFC 768)`**



3. Protocolo de Datagrama de Usuario (UDP)

Características

- Permite que los usuarios puedan transmitir mensajes sin establecimiento de conexión y con una sobrecarga de cabecera muy baja, pero sin garantía o confirmación de entrega y sin secuenciado.
- Transporta unidades de datos entre PUERTOS de sistemas.
- UDP es el estándar 6 y se describe en la RFC 768 (28 de agosto de 1.980)
- Otra RFC relevante es la 1122 (Host Network Requirements) la cual es necesaria para poder implementar UDP correctamente.
- Es apropiado para aplicaciones que implementan sus propios controles de errores, porque estos sean muy específicos o por la forma en la que tratan la información (DNS, streaming multimedia, multicast, etc.)

3. Protocolo de Datagrama de Usuario (UDP)

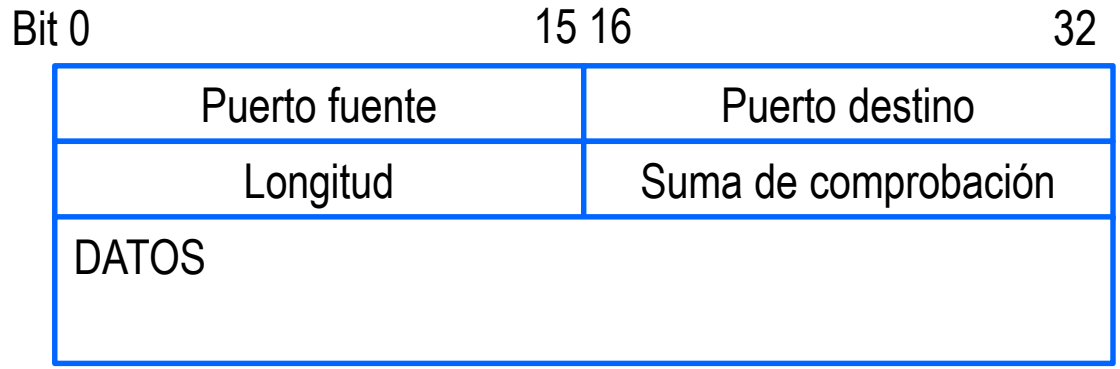
Características

Encapsulado

- UDP agrupa todos los datos que se le solicita enviar en un solo datagrama, ya sea 8 bytes (sólo su cabecera) o 64Kbytes, sin importarle este tamaño o la posible fragmentación en IP.
  - *De esta forma, si un datagrama UDP es leído por su destinatario, éste puede estar seguro de que el datagrama está completo, ya que si por haberse fragmentado no hubiese llegado algún trozo, se hubieran descartado todos los demás y el receptor no se apercibiría de este hecho.*
- Dada la implantación del protocolo UDP y de las primitivas Socket, UDP se emplea como base para desarrollar otros protocolos de transporte, como por ejemplo RTP o QUIC.
  - El coste y el impacto de crear un nuevo protocolo de transporte hoy en día hace a UDP un candidato ideal para este tipo de protocolos que lo usan como medio para enviar sus PDUs.

3. Protocolo de Datagrama de Usuario (UDP)

Formato de la PDU de UDP

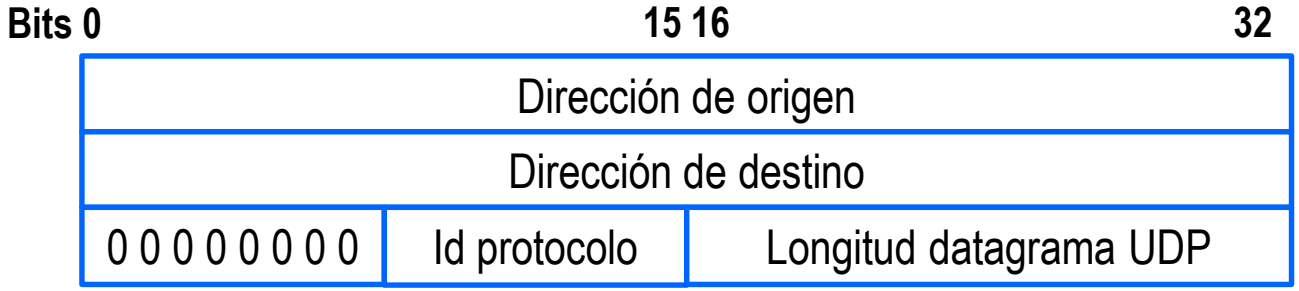


- Puerto fuente [opcional]: Puerto del proceso que envía el datagrama. Cero si no se usa.
- Puerto destino: Puerto destino donde espera el proceso receptor.
- Longitud del datagrama: tamaño del datagrama en bytes incluyendo la cabecera y los datos del usuario.
  - Tamaño mínimo 8 bytes
  - Tamaño máximo 65.635 bytes.

3. Protocolo de Datagrama de Usuario (UDP)

Formato de la PDU de UDP

- Suma de comprobación (*checksum*) [opcional]: se calcula sumando en complemento a 1 la cabecera UDP, los datos de usuario y la pseudo cabecera.
  - Para el cálculo, los bytes resultantes se alinean a un múltiplo 16 bits (se rellena con ceros).
  - Si se recibe un datagrama con un *checksum* distinto de cero se debe realizar su comprobación. Si no es igual, el datagrama se descarta sin que se notifique a ninguna instancia o entidad de UDP.
  - En IPv6 esta suma de comprobación se realiza con las direcciones de 128 bits, la longitud del paquete pasa a 32 bits y se incorpora a una cabecera especial en el protocolo de red. Además no está permitido un *checksum* a cero en origen (Sección 8.1 RFC 8200).



Pseudo cabecera para la suma de comprobación

3. Protocolo de Datagrama de Usuario (UDP)

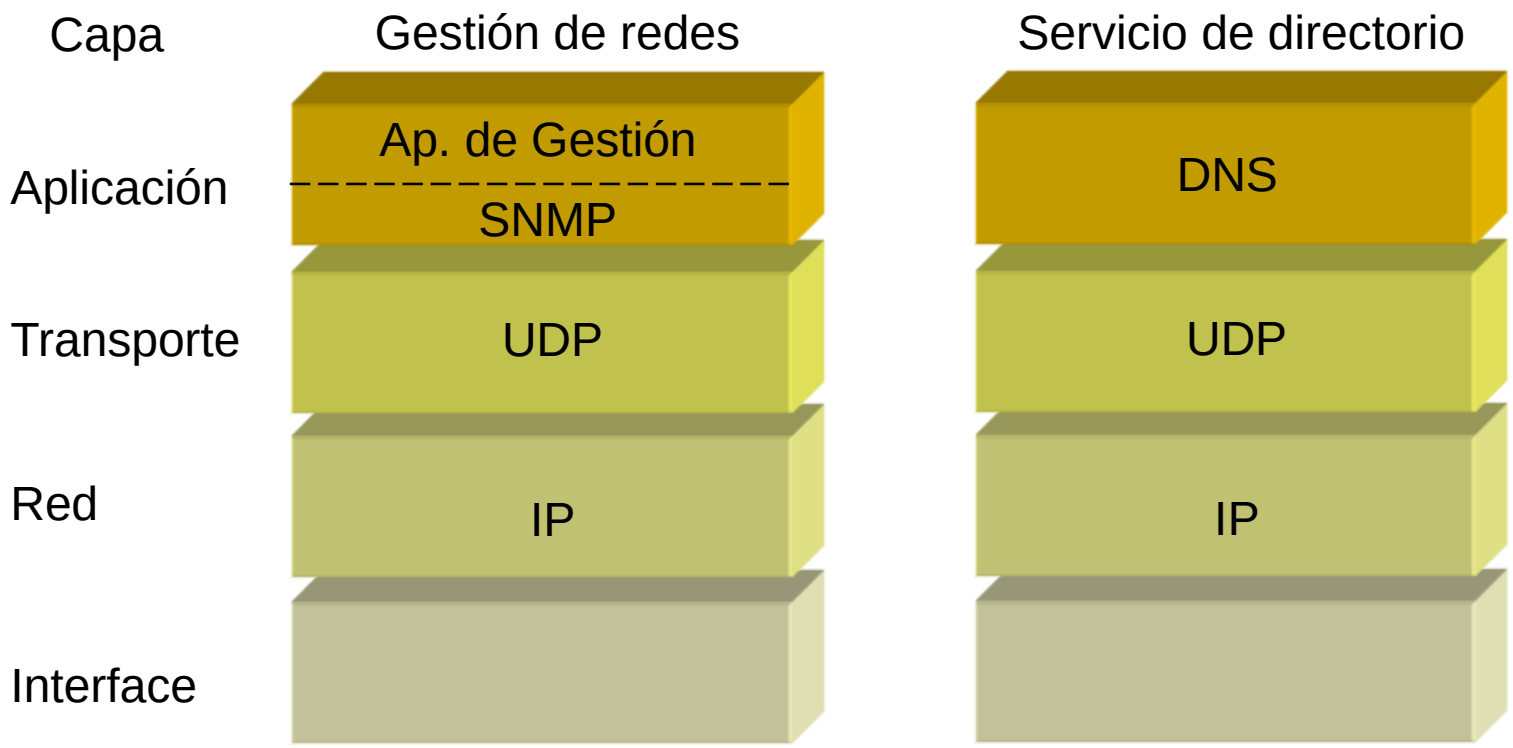
Protocolos que usan UDP

Puerto	Referencia	Protocolo	Comentarios
53	domain	Domain Name Server (DNS)	Usa un sencillo sistema de preguntas y respuestas para la mayoría de intercambios, aunque también usa TCP para las consultas más largas
67 y 68	bootps/ bootpc	Bootstrap Protocol (BOOTP) Y Dynamic Host Configuration Protocol (DHCP)	Protocolos de configuración de host consistentes en peticiones y respuestas cortas
69	tftp	Trivial File Transfer Protocol (TFTP)	TFTP está diseñado para el envío rápido y sencillo de ficheros pequeños. Para evitar errores en los ficheros, TFTP implementa algunas versiones simplificadas de características de TCP como los asentimientos
137	NetBios	NetBIOS Extended User Interface (NetBEUI)	Netbios no es un protocolo, si no un interfaz diseñado por IBM para acceder a PCs conectados en una red
161 **162**	SNMP SNMP trap	Simple Network Management Protocol	Un protocolo de gestión de red que usa mensajes relativamente cortos

3. Protocolo de Datagrama de Usuario (UDP)

Protocolos que usan UDP

Ejemplos



3. Protocolo de Datagrama de Usuario (UDP)

Desventajas del uso de UDP

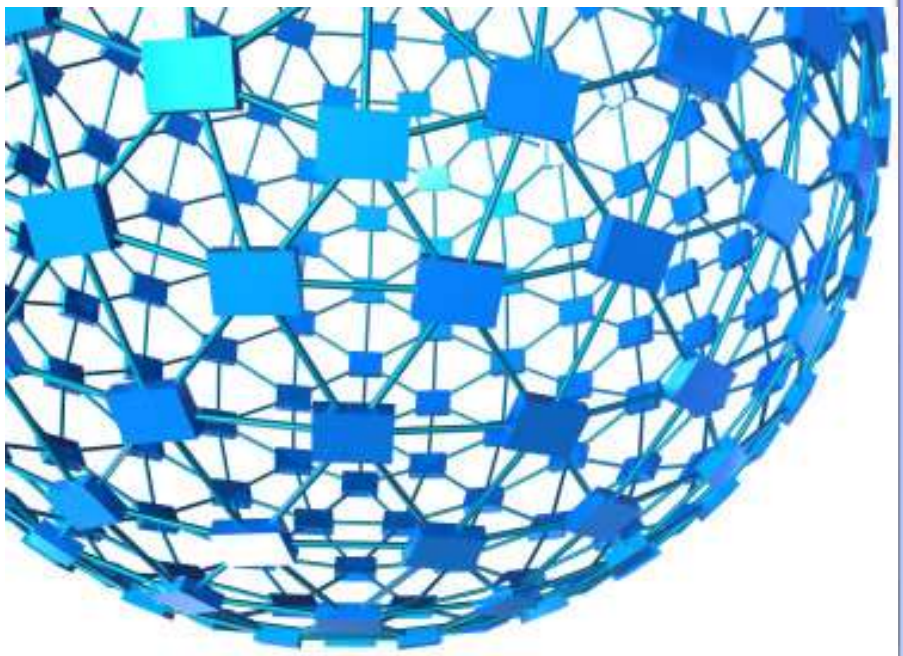
- Si un datagrama UDP es fragmentado por IP, y uno de esos fragmentos se pierde, todo el datagrama será descartado y ni la aplicación receptora tendrá notificación de este suceso, ni la entidad emisora.
- Normalmente, para depurar efectivamente los problemas con la transmisión de paquetes UDP, es necesario delegar en ICMP.
- Si se desea implementar corrección de errores, ésta debe relegarse a la aplicación.
- En entornos de red con altas tasas de pérdidas de paquetes, las aplicaciones que usen UDP comenzarán a notar problemas antes que aquellas que usen protocolos como TCP (lo veremos en el siguiente apartado del tema).
- El tráfico UDP (salvo el del puerto 53 de DNS) suele estar bloqueado en muchas redes, como por ejemplo con políticas en los firewall.

4. Protocolo de Control de Transmisión (TCP)

Introducción

- **Transmission Control Protocol o TCP** proporciona un **servicio confiable, orientado a conexión** para aplicaciones fundamentalmente dedicadas a intercambiar información.
- TCP es usado prácticamente por todos los protocolos de aplicación que existen hoy en día en Internet, los cuales requieren las capacidades de TCP para asegurar que la información no se pierde y que no está corrupta.

- **`Identificador de protocolo: 6`**
- **`Estándar: 7`**
- **`Definido en la RFC 9293 agosto 2022 (anterior: RFC 793 septiembre 1981)`**



4. Protocolo de Control de Transmisión (TCP)

RFCs vinculadas con TCP

¿Por qué una nueva RFC de TCP?

- Durante décadas TCP ha sido ampliamente implementado y empleado como protocolo de transporte para numerosas aplicaciones en Internet.
- En este tiempo la RFC 793, junto con otros documentos, ha servido de referencia para su implementación.
- Además, las erratas, modificaciones, las deficiencias encontradas y resueltas en seguridad, rendimiento, etc., han hecho que el número de documentos crezca.

El objetivo de la RFC 9293 es unificar estos documentos en una especificación actualizada del protocolo.

4. Protocolo de Control de Transmisión (TCP)

RFCs vinculadas con TCP

Nueva revisión de TCP

- La RFC 9293 es una revisión/compendio de los siguientes documentos:
  - RFC 793. TRANSMISSION CONTROL PROTOCOL (09/1981).
  - RFC 879. The TCP Maximum Segment Size and Related Topics (12/1983).
  - RFC 2873. TCP Processing of the IPv4 Precedence Field (06/2000).
  - RFC 6093. On the Implementation of the TCP Urgent Mechanism (1/2011).
  - RFC 6429. TCP Sender Clarification for Persist Condition (12/2011).
  - RFC 6528. Defending against Sequence Number Attacks (2/2012).
  - RFC 6691. TCP Options and Maximum Segment Size (MSS) (07/2012).
- *No se incluyen, por ejemplo, aquellas que definen el control de congestión o la detección de pérdidas, que podrán ser objeto de futuros documentos de revisión*.

4. Protocolo de Control de Transmisión (TCP)

RFCs vinculadas con TCP

Nueva revisión de TCP

- La RFC 9293 también actualiza los siguientes documentos:
  - RFC 1011. OFFICIAL INTERNET PROTOCOLS (05/1987).
  - RFC 1122. Host Network Requirements (10/1989).
  - RFC 5961. Improving TCP's Robustness to Blind In-Window Attacks (08/2010).

4. Protocolo de Control de Transmisión (TCP)

Características básicas

- Proporciona a las aplicaciones un servicio confiable basado en un flujo de bytes.
  - Utiliza temporizadores y retransmisiones.
  - Numeración de bytes por offset.
- Permite establecer conexiones de transporte entre puertos de diferentes sistemas.
  - Establecimiento de conexión mediante protocolo de “Ida - vuelta - ida” o “a tres vías”.
- Control de flujo mediante ventana deslizante.
- Control de congestión.
- Encapsula datos de capas superiores, fragmentándolos en unidades, de 64 KBytes como máximo, denominadas **segmentos**.

4. Protocolo de Control de Transmisión (TCP)

Servicios proporcionados por TCP

¿Por qué usar TCP?

- Los protocolos de aplicación pueden proporcionar sus propias características de transporte de datos seguros y confiables, además de control de flujo, pero:
  - No es práctico.
  - Complejidad añadida.
  - Ciclos de desarrollo más largos.
  - Peor interoperabilidad.
- Solución mejor: usar una capa de transporte apropiada.

4. Protocolo de Control de Transmisión (TCP)

Servicios proporcionados por TCP

- Circuitos virtuales.
- Gestión de la entrada salida de las aplicaciones.
- Gestión de la entrada salida hacia la red.
- Control de flujo.
- Control de congestión (se verá más adelante).
- Confiabilidad (*reliability*).

Todos estos servicios hacen a TCP un protocolo de transporte muy robusto.

4. Protocolo de Control de Transmisión (TCP)

Servicios proporcionados por TCP

Circuitos virtuales

- Cuando dos aplicaciones necesitan comunicarse, TCP establece un circuito virtual entre ellas.
- Los circuitos virtuales son el corazón del diseño de TCP y proporcionan:
  - Confiabilidad.
  - Control de flujo.
  - Características de control de entrada y salida.
- Lo diferencian de UDP.

4. Protocolo de Control de Transmisión (TCP)

Servicios proporcionados por TCP

Gestión de la entrada/salida de las aplicaciones

- Las aplicaciones se comunican entre sí enviado datos a la entidad local de TCP.
  - Los datos se transmiten a través del circuito virtual al otro extremo, siendo eventualmente recogidos por la aplicación destinataria.
- TCP proporciona buffers de entrada y salida para ser usados por las aplicaciones:
  - Permitiendo enviar y recibir datos como flujos continuos.
  - Es TCP quien fragmenta los datos adecuadamente en segmentos monitorizados, que se envían sobre IP.
  - Los datos empaquetados en un segmento TCP no tienen por qué corresponder con un envío (o escritura en un socket) desde la aplicación.

4. Protocolo de Control de Transmisión (TCP)

Servicios proporcionados por TCP

Gestión de la entrada/salida hacia la red

- Cuando TCP necesita enviar datos a la red usa IP.
  - Proporcionan un servicio de gestión de entrada y salida para IP basado en:
    - Envío a la red de paquetes de tamaños apropiados.
    - Ensamblado de los diferentes paquetes recibidos en un flujo continuo de datos.

4. Protocolo de Control de Transmisión (TCP)

Servicios proporcionados por TCP

Control de flujo

- TCP se encarga de controlar las diferentes necesidades y capacidades de los equipos (procesos) puestos en comunicación a través de un circuito virtual, mantenido por él de manera transparente a las aplicaciones.

4. Protocolo de Control de Transmisión (TCP)

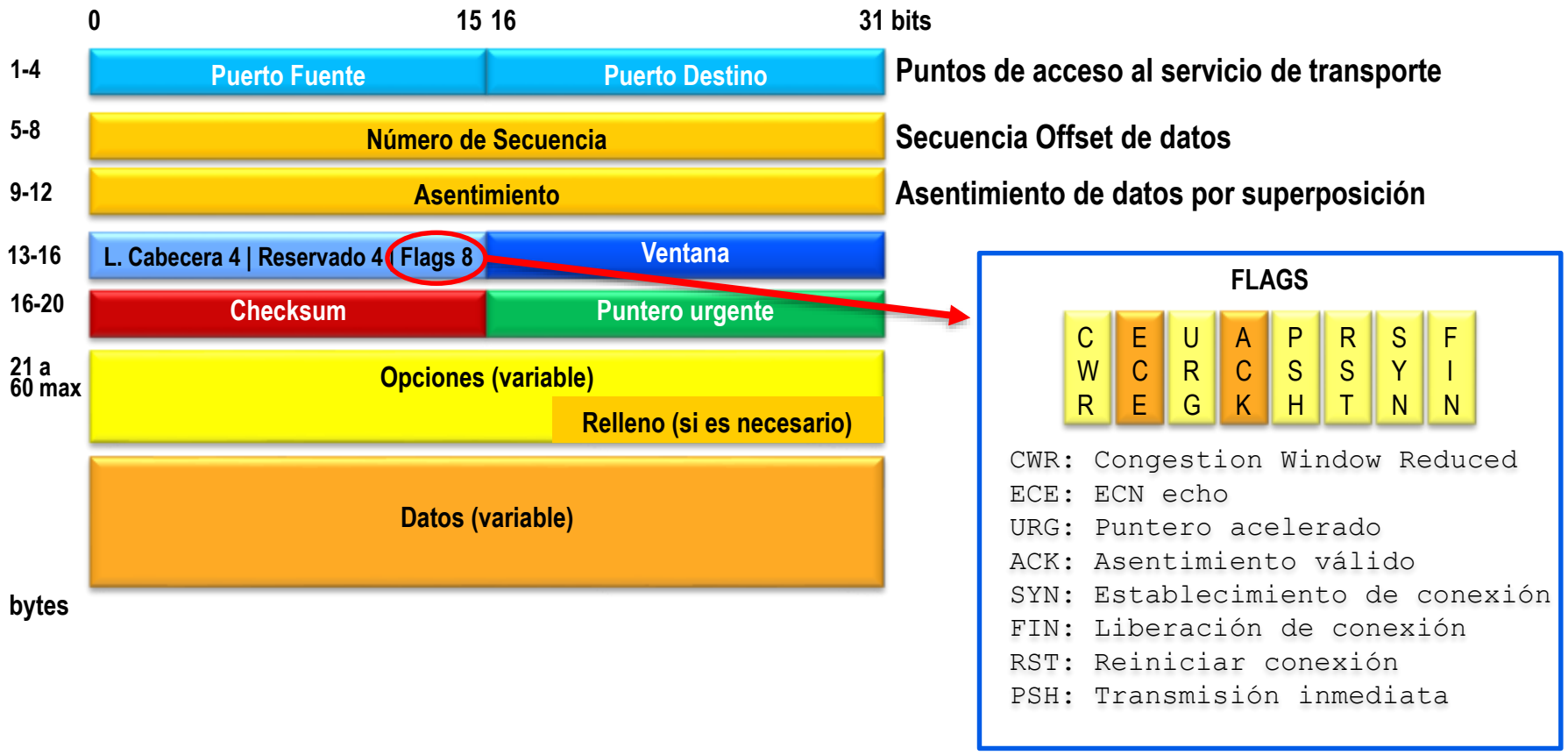
Servicios proporcionados por TCP

Confiabilidad (reliability)

- La confiabilidad se trata de detectar:
  - la pérdida de paquetes a través de los números de secuencia (asentimientos).
  - los errores a través del *checksum* en cada segmento.
- y corregirlos con la retransmisión de los segmentos afectados.

4. Protocolo de Control de Transmisión (TCP)

Formato de los segmentos de TCP



4. Protocolo de Control de Transmisión (TCP)

Formato de los segmentos de TCP

Campos de cabecera

- **Puerto fuente y puerto destino (16 bits):** Identifican los puntos terminales de la comunicación. Cada host establece sus puertos libremente dentro del rango disponible.
  - *Existen los denominados "well-known ports" que suelen estar reservados por aplicaciones específicas o por el sistema operativo, como el puerto 80 para HTTP.*
  - *La Internet Assigned Numbers Authority (IANA), tradicionalmente ha reservado para el rango de 16 bits de los puertos los siguientes usos [1]:*
    - Well Known Ports, del 0 hasta 1023
    - Puertos Registrados, del 1024 hasta 49151
    - Puertos Dinámicos o Privados o efímeros, del 49152 hasta 65535

[1] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011,

4. Protocolo de Control de Transmisión (TCP)

Formato de los segmentos de TCP

Campos de cabecera

- **Número de secuencia y número de acuse de recibo (32 bits):** Cada byte de datos está numerado en una comunicación TCP.
  - El número de secuencia indica el índice del primer byte enviado en el paquete. Esto es así salvo en los mensajes SYN, donde el número de secuencia es el ***Initial Sequence Number*** **(ISN)**, siendo el primer byte a enviar el ISN+1.
  - En un segmento que incluya un acuse de recibo, en el campo de acuse de recibo, se pone el índice del siguiente byte recibido en secuencia (sin huecos), es decir, el número de secuencia del primer byte esperado en un subsiguiente mensaje.

4. Protocolo de Control de Transmisión (TCP)

Formato de los segmentos de TCP

Campos de cabecera

Número de secuencia inicial (ISN)

- La elección del ISN no es sencilla porque debe evitar confundir segmentos de conexiones anteriores con segmentos nuevos, además de los ataques de falsas peticiones de conexión.
  - Se pretende que los nuevos números de secuencia vayan más allá del *Maximum Segment Lifetime* (MSL).
  - Una implementación de TCP debe emplear la fórmula siguiente:

```
ISN = M + F(localip, localport, remoteip, remoteport, secretkey)
```

- M: Temporizador que incrementa en uno el ISM cada 4 microsegundos.
- F(): Función seudoaleatoria
- Secretkey: al menos de 128 bits.

4. Protocolo de Control de Transmisión (TCP)

Formato de los segmentos de TCP

Campos de cabecera

- **Longitud de cabecera (*data offset*. 4 bits):** Número de palabras de 32 bits que contiene la cabecera, indicando por tanto dónde comienzan los datos.
  - Cabecera sin extensiones: 20 bytes, por lo tanto la longitud es 5.
  - Máxima longitud de la cabecera: valor del campo=15, que da un total de 60 bytes.
- **Bits de reserva (4 bits):** No se usan, deben ser 0 en los segmentos que se generen e ignorados en la recepción. Algunas RFC experimentales los usan, pero aún no son estándar.
- **Banderas (8 bits):** en inglés *flags*. Están activas cuando su valor es "1".

4. Protocolo de Control de Transmisión (TCP)

Formato de los segmentos de TCP

Campos de cabecera

Banderas

- **CWR (Congestion Window Reduced - 1 bit [RFC 3168]):** Puesto a 1 para negociar el soporte a la notificación de congestión explícita de IP (ECN) y para informar de la reducción de la ventana por parte del emisor.
- **ECE (ECN echo - 1 bit [RFC 3168]):** El igual que CWR para negociar el soporte a ECN y avisar de la recepción de un paquete con los bits ECN de IP a 11 (*Congestion Experienced* o CE) que se asimila a haber perdido un segmento.
  - *El funcionamiento que implica estos flags se verá al estudiar los mecanismos de control de la congestión de TCP*

4. Protocolo de Control de Transmisión (TCP)

Formato de los segmentos de TCP

Campos de cabecera

Banderas

- **URG (URGENT - 1 bit):** Envío de segmento con datos urgentes. Si está activo el campo de PUNTERO URGENTE es válido e indica el final de los datos urgentes. Mecanismo rudimentario de señalización de interrupción. Raramente usado. En los sockets Berkeley se activa con el parámetro MSG_OOB.
  - La RFC 6093 (y la RFC 9293 lo menciona en los mismos términos) desaconseja su uso futuro, ya que ha habido mucha ambigüedad sobre el tema y según esta RFC 1120 el valor del campo de puntero urgente debería apuntar al número de secuencia del byte siguiente del último byte urgente, y esto no ha sido tenido muy en cuenta en algunas implementaciones de TCP.
- **ACK (ACKNOWLEDGMENT -1 bit):** A “1” el segmento porta un acuse de recibo positivo, haciendo válido el campo asentimiento en la cabecera. Si es “0” no contiene acuse de recibo.

4. Protocolo de Control de Transmisión (TCP)

Formato de los segmentos de TCP

Campos de cabecera

Banderas

- **PSH (PUSH - 1 bit):** Cuando está activo en un segmento, toda la información pendiente de enviar en TCP debe ser transmitida, y además, en la entidad receptora, la información se pasará directamente a la aplicación destino aunque aún quede espacio en el buffer de dicha aplicación.
  - No se puede emplear como un sistema para delimitar un segmento, ya que no se garantiza que solamente vayan datos de una escritura desde la aplicación.
- **RST (RESET - 1 bit):** Se utiliza para abortar un circuito virtual cuando no puede ser finalizado de una manera ordenada o para rechazar un segmento o petición de conexión que proviene de un socket inválido.

4. Protocolo de Control de Transmisión (TCP)

Formato de los segmentos de TCP

Campos de cabecera

Banderas

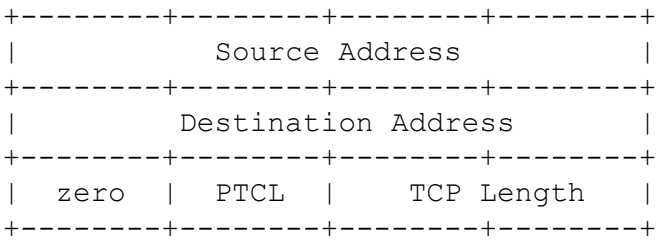
- **SYN (SYNCHRONIZE - 1 bit):** Convierte el segmento en una petición de conexión si SYN = 1 o en una confirmación de conexión si SYN = 1 y ACK = 1.
- **FIN (FINISH - 1 bit):** Liberación de la conexión por el transmisor, pero aún puede seguir recibiendo datos indefinidamente.
  - *Tanto los segmentos con SYN o FIN tienen número de secuencia por lo que serán tratados en orden.*

4. Protocolo de Control de Transmisión (TCP)

Formato de los segmentos de TCP

Campos de cabecera

- **Ventana (16 bits):** Cantidad de bytes que pueden enviarse a partir del último byte del que se ha recibido acuse de recibo.
  - Si este campo es cero, indica que se confirman los datos anteriores pero el receptor no admite más datos por el momento.
  - Cuando la entidad puede procesar nuevos datos, se vuelve a enviar un mensaje con el mismo acuse de recibo, pero con el campo ventana mayor que 0.
- **Suma de comprobación (CHECKSUM - 16 bits):** Igual que en UDP pero con el nº de protocolo correspondiente a TCP (6).



- **Apuntador urgente (16 bits):** Puntero al byte final de los datos urgentes si el bit URG está a “1”, si URG no está a uno, este valor no es válido.

4. Protocolo de Control de Transmisión (TCP)

Formato de los segmentos de TCP

Campos de cabecera

- **Opciones (tamaño variable):** Características adicionales no cubiertas por la cabecera normal.
  - Cada opción está identificada por:
    - Un solo byte de tipo de opción (option-kind).
    - o, al menos, con 3 bytes: tipo (1 byte), longitud (1 byte que incluye también a los bytes de tipo y longitud, además de la longitud del campo valor) y valor (variable).
  - Como máximo se pueden usar 40 bytes de opciones.
  - Ejemplos: MSS (2), Window Scale(3), SACK permited (4) SACK data(5).

- **Relleno (tamaño variable):** El tamaño de la cabecera de TCP debe ser un múltiplo de 32 bits, por lo que si hay campo de opciones es posible que sea necesario este relleno para mantener esa condición.

4. Protocolo de Control de Transmisión (TCP)

Opciones de TCP

- Las opciones en TCP son generalmente negociadas al inicio de una conexión usando el campo de opciones de los segmentos SYN y SYN ACK para su información y configuración.
- Una entidad TCP debe ignorar sin error cualquier opción que no implemente.
- Las opciones disponibles de TCP con diversos estatus se pueden encontrar en :

  https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xhtml

  (última actualización: 2024-07-11)
- En la especificación de TCP tan sólo se detallan tres opciones y son obligatorias para todas las implementaciones de TCP:
  - End of Options List (EOL), No Operation (NOP) y Maximun Segment Size(MSS).

4. Protocolo de Control de Transmisión (TCP)

Opciones de TCP

Maximun Segment Size (MSS)

- Definida en la RFC 9293 (estándar).
- Esta es la opción más importante y más comúnmente usada.
- Con esta opción TCP indica el máximo de ***carga útil o Maximum Segment Size*** que está dispuesto a aceptar.
- El valor de MSS son los datos que podrá llevar un segmento TCP sin contar la cabecera.
- Debería ser enviada como respuesta a un segmento SYN recibido con un valor de MSS diferente a los valores por defecto.
  - Los valores por defecto son a 536 (576-40) en IPv4 y 1220 (1280-60) en IPv6.

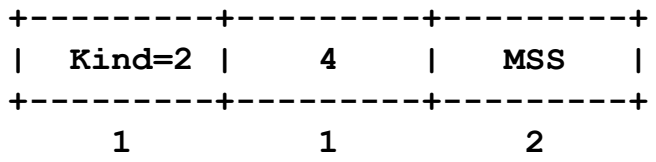
4. Protocolo de Control de Transmisión (TCP)

Opciones de TCP

Maximun Segment Size (MSS)

- Para IPv4, se exige que los host sean capaces de manejar segmentos de 576 bytes (valor mínimo de MTU o *Maximum Transmission Unit*) , 556 de datos más los 20 de cabecera, por lo tanto se toma por defecto 536 para los datos de TCP (20 cabecera IP + 20 cabecera TCP + 536 Datos).
  - Valor típico para IPv4: 1460 bytes, 40 bytes menos, que coinciden con las cabeceras de IP y TCP para que el tamaño coincida con la trama típica de Ethernet (1500 bytes).
- El valor mínimo de MTU para IPv6 es 1280 bytes (RFC 8200), por lo que se aumenta el límite anterior.

- Tipo=2 longitud=4 valor: 2 bytes.



4. Protocolo de Control de Transmisión (TCP)

Opciones de TCP

Escalado de ventana (WSopt)

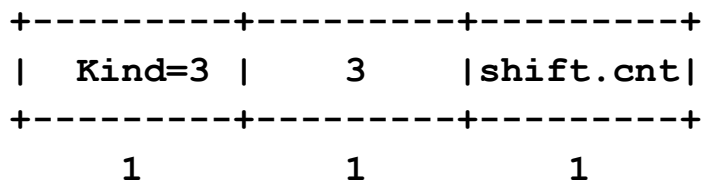
- Definida en la RFC 7323 (Deja obsoleta la RFC 1323).
- Permite mejorar el rendimiento en redes de alto ancho de banda, alto retardo o ambas, denominadas LFNs (Long Fat Networks).
- Esta opción tan solo se puede activar al comienzo de una conexión, enviando la opción *Window Scale* en un paquete SYN.
- Ambos extremos pueden establecer un valor diferente, pero solamente si se ha recibido la opción.

4. Protocolo de Control de Transmisión (TCP)

Opciones de TCP

Escalado de ventana (WSopt)

- La ventana se escala a un tamaño variable con un máximo de 30 bits (2^{30}= 1 Gbyte) desplazando el valor 14 bits a la izquierda.
  - TCP Window Scale Option (WSopt): Tipo(kind): 3 Longitud: 3 bytes



  - ***shift.cnt*** indica el número de bits a desplazar a la izquierda, como máximo puede ser 14 (16 del campo tamaño de ventana + 14 shift.cnt = 30) y está permitido que sea 0 (factor de escala 1).
  - ***shift.cnt*** tiene que ser 14 para que se puedan identificar en qué ventana se está en función del número de secuencia y lo cerca que estén a los punteros que controlan la ventana deslizante.

4. Protocolo de Control de Transmisión (TCP)

Opciones de TCP

Marcas de tiempo (TSopt)

- Definida en la RFC 7323 (Deja obsoleta la RFC 1323).
- Permite mejorar una medición más precisa del tiempo de ida y vuelta (Round Trip Time – RTT).
- Problema:
  - La respuesta a una medición de tiempo se hace sólo en los ACKs: debido a que TCP puede demorar los ACKs por eficiencia, la medición del RTT con los asentimientos puede no ser fiable.

4. Protocolo de Control de Transmisión (TCP)

Opciones de TCP

Marcas de tiempo (TSopt)

Funcionamiento

1. La entidad que envía pone una marca de tiempo en un segmento enviado (TSval).
2. La entidad receptora, en el siguiente ACK envía la nueva marca de tiempo incrementada (se recomienda incrementarla en 1 por cada segundo pasado) en TSval y el valor recibido antes en TSecr.
   Si no es un mensaje de ACK TSecr debe ser 0.

– TCP Timestamp option (TSopt): Tipo(kind): 8 Longitud: 10 bytes

```
+--------+--------+---------------------+---------------------+
| Kind=8 |  10    | TS Value (TSval)    |TS Echo Reply (TSecr)|
+--------+--------+---------------------+---------------------+
     1        1              4                     4
```

4. Protocolo de Control de Transmisión (TCP)

Opciones de TCP

Selective Acknowledge (TCP SACK)

- Definida en las *RFCs 2018 y 2883*.
- Aporta instrumentos para una retransmisión selectiva de segmentos perdidos.
  - Mejor comportamiento ante pérdidas múltiples.
  - Limitada al uso de la capacidad de ampliación de la cabecera de TCP, como máximo, cuatro huecos de datos sin asentir (si no se emplean otras extensiones).
  - No diferencia entre pérdidas o errores producidos por la congestión de la red o por la corrupción de una TPDU.
  - Una entidad sabe que la otra entidad es capaz de usar SACK si en la conexión (Segmento SYN o SYN+ACK) se envía la opción SACK-Permitted.

4. Protocolo de Control de Transmisión (TCP)

Opciones de TCP

Selective Acknowledge (TCP SACK)

```
TCP Sack-Permitted
Option: Kind: 4

+---------+---------+
| Kind=4  | Length=2|
+---------+---------+
```

```
TCP SACK Option:
Kind: 5, Length: Variable
+--------+--------+
| Kind=5 | Length |
+--------+--------+--------+--------+
|      Left Edge of 1st Block       |
+--------+--------+--------+--------+
|     Right Edge of 1st Block       |
+--------+--------+--------+--------+
|                                   |
/                . . .              /
|                                   |
+--------+--------+--------+--------+
|      Left Edge of nth Block       |
+--------+--------+--------+--------+
|      Right Edge of nth Block      |
+--------+--------+--------+--------+
```

4. Protocolo de Control de Transmisión (TCP)

Opciones de TCP

Selective Acknowledge (TCP SACK)

Extensión de SACK o D-SACK

- Existe una extensión al funcionamiento de SACK (RFC 2883) que permite comunicar bloques duplicados.
- Ayuda al emisor a dejar de transmitir bloques cuyos ACKs se han perdido.
- El bloque duplicado es el primero que se comunica de los bloques SACK. Se identifican porque los márgenes de los bloques son anteriores al número de secuencia del asentimiento del segmento que los lleva.
- No interfiere con SACK y no necesita negociación aparte.

Sally Floyd (Charlottesville, 20 de mayo de 1950 - Berkeley, 25 de agosto de 2019) fue una ingeniera estadounidense. Es conocida fundamentalmente por su trabajo en el control de congestión de Internet, y es una de las diez investigadoras más citadas en informática. Floyd es una de las autoras de los estándares **Selective Acknowledgement** (SACK), Explicit Congestion Notification (ECN), el Protocolo de Control de Congestión de Datagrama (DCCP) y TCP Friendly Rate Control (TFRC).
Fuente: Wikipedia

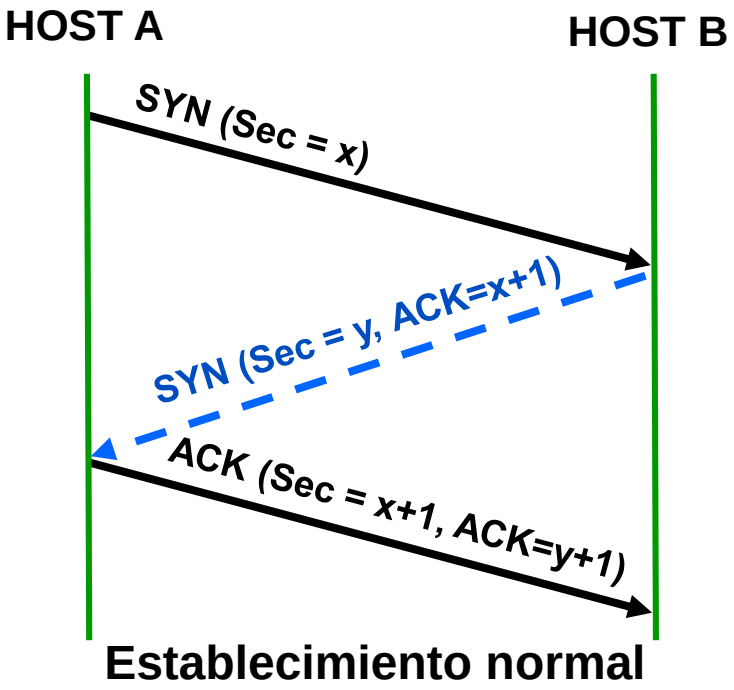
4. Protocolo de Control de Transmisión (TCP)

Establecimiento de conexión en TCP

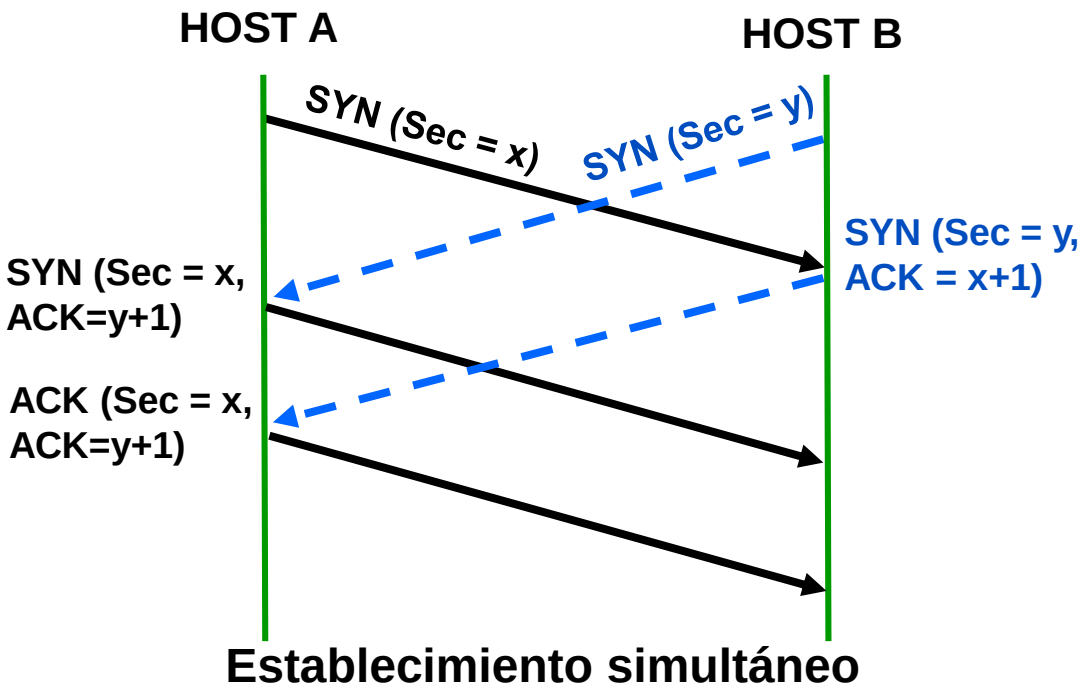
- TCP emplea el mecanismo de conexión a tres vías o ida-vuelta-ida.
- La conexión se inicia cuando un extremo (normalmente un cliente) ejecuta la primitiva CONNECT (lado activo) con destino a otro extremo (lado pasivo) que haya ejecutado la primitiva LISTEN (o CONNECT).
  - El inicio de una conexión en el lado activo, y la recepción de la solicitud en el lado pasivo crea un registro de la conexión con las variables que se necesitan denominado ***Transmission Control Block*** **(TCB)**.
- Es un mecanismo muy fiable, que permite la conexión entre dos extremos tanto por la solicitud de uno de ellos, como cuando ambos quieren conectarse entre sí simultáneamente (poco común).
- Las PDUs involucradas en el proceso de conexión son SYN, SYN+ACK, ACK y RST.

4. Protocolo de Control de Transmisión (TCP)

Establecimiento de conexión en TCP



Establecimiento normal



Establecimiento simultáneo

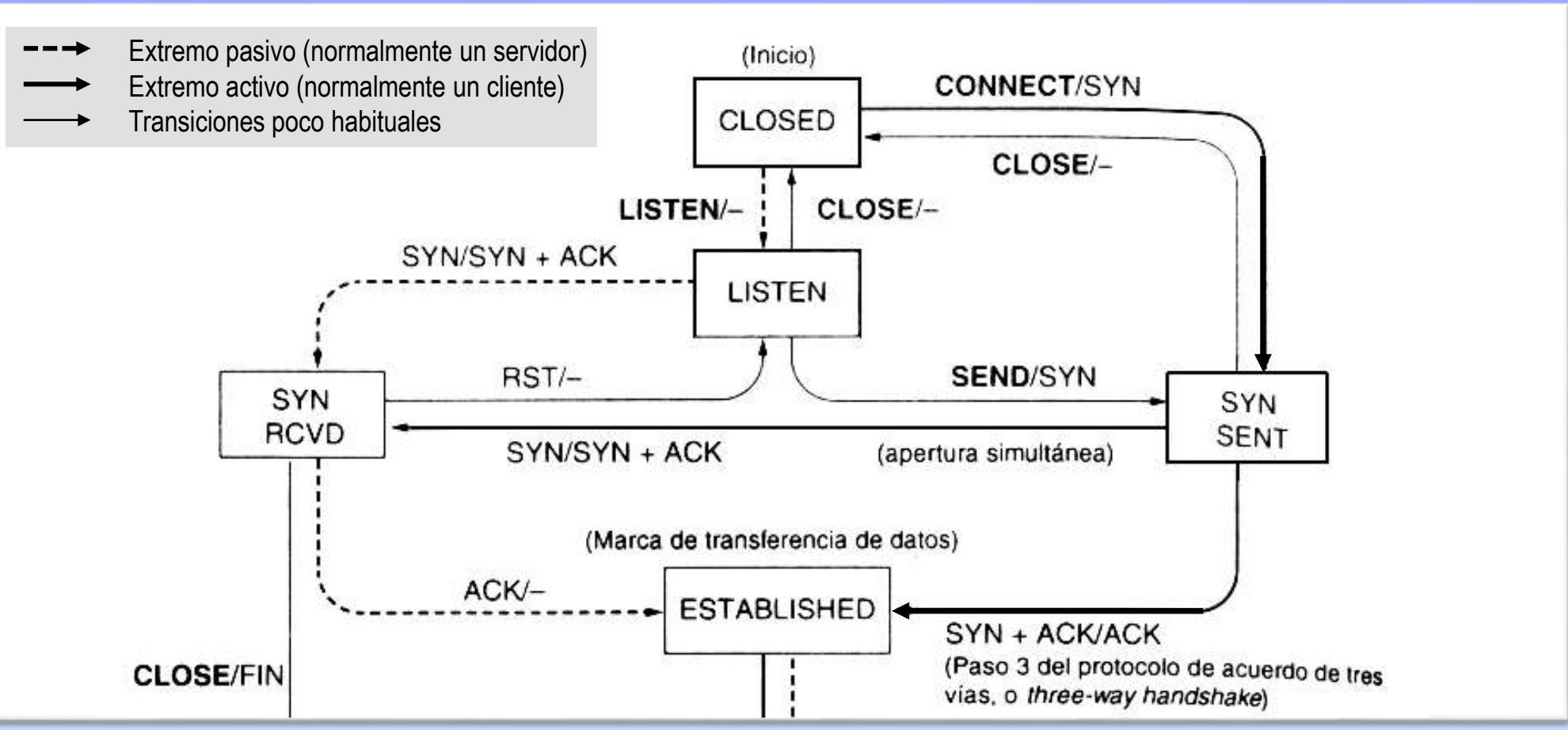
4. Protocolo de Control de Transmisión (TCP)

Establecimiento de conexión en TCP

- Según TCP, existe la posibilidad de enviar datos en el ACK que asiente el SYN+ACK
  - Sin embargo, los sockets nunca han implementado esta opción, por lo que no se usa.
- Una entidad de TCP reintenta una conexión un número determinado de veces tras un intento fallido.
  - Este número es configurable en el sistema operativo (normalmente 4 o 5).
  - El espaciado entre reintentos comienza con 3 segundos y se va duplicando cada vez. Así, la siguiente vez se reintenta a los 6 segundos, luego a los 12, y continua hasta conseguir conectar o agotar los reintentos.
  - El incremento del temporizador de esta manera se debe al algoritmo de Karn, que se verá más adelante.

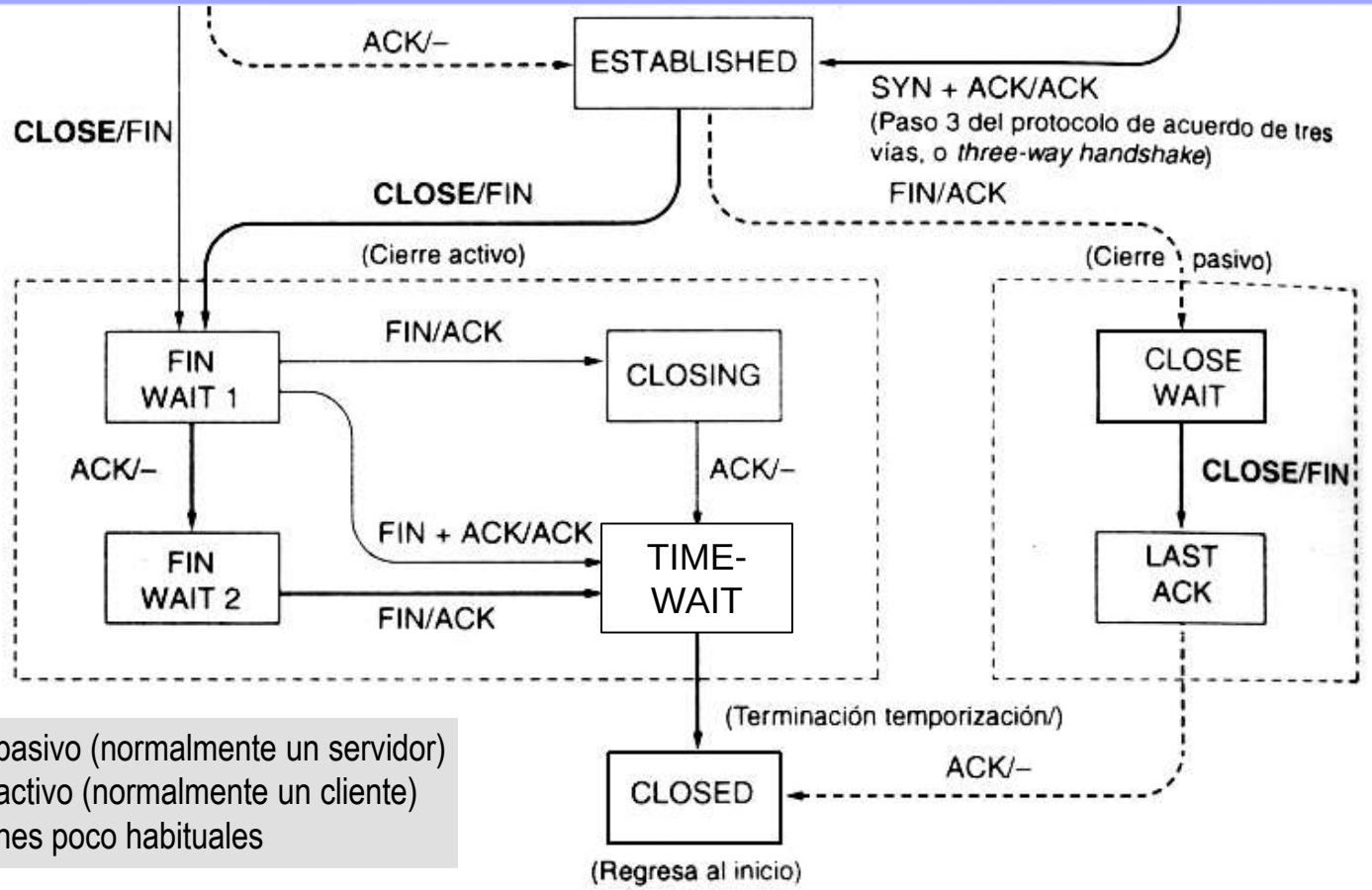
4. Protocolo de Control de Transmisión (TCP)

Máquina de estados de TCP (parte conexión)



4. Protocolo de Control de Transmisión (TCP)

Máquina de estados de TCP (parte desconexión)



Un RST puede ser enviado desde cualquier estado, lo que correspondería con una transición al estado TIME-WAIT.

4. Protocolo de Control de Transmisión (TCP)

Máquina de estados de TCP

Resumen de estados

ESTADO	DESCRIPCIÓN
CLOSED	No hay conexión activa ni pendiente.
LISTEN	Se espera una petición de conexión una vez se haya enviado una petición de conexión.
SYS-RECEIVED	Llegó solicitud de conexión; espera ACK.
SYN-SENT	La aplicación comenzó a abrir una conexión. Se ha enviado una petición de conexión y se espera recibir una respuesta adecuada.
ESTABLISHED	Estado normal de transferencia de datos en una conexión.

4. Protocolo de Control de Transmisión (TCP)

Máquina de estados de TCP

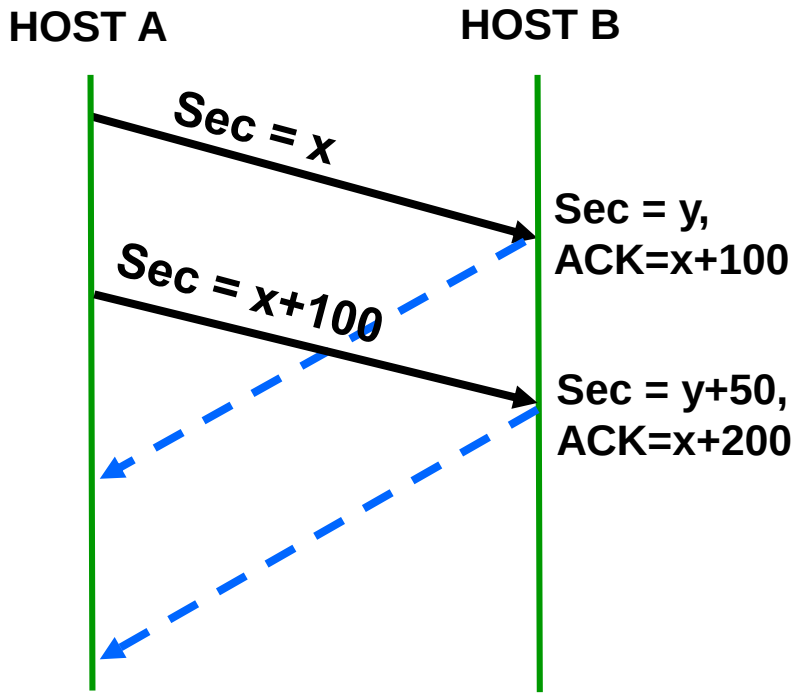
Resumen de estados

ESTADO	DESCRIPCIÓN
FIN-WAIT-1	Esperar a una petición de terminación de la conexión del otro extremo o al asentimiento de una petición de cierre previa.
FIN-WAIT-2	Esperar a una petición de cierre del otro extremo.
TIME-WAIT	Espera que todos los paquetes mueran. El tiempo se define como 2×MSL (*Maximun Segment Lifetime*, 120 segundos).
CLOSING	Ambos lados intentaron cerrar simultáneamente.
CLOSE-WAIT	El otro lado inició una liberación. Se espera a finalizar la conexión local.
LAST-ACK	Representa la espera del asentimiento de la solicitud de conexión previamente enviada.

4. Protocolo de Control de Transmisión (TCP)

Asentimiento de la información

- Los asentimientos de información es parte de la capacidad de TCP de aportar confiabilidad a la conexión.
- TCP asiente los segmentos recibidos mediante el envío al emisor de otros segmentos con el flag ACK a “1” y con el campo asentimiento conteniendo el valor de secuencia del siguiente byte con respecto último byte recibido que fuera correcto y en secuencia.



Asentimiento de la información

4. Protocolo de Control de Transmisión (TCP)

Asentimiento de la información

- TCP puede acumular asentimientos de manera que no enviará un ACK por cada segmento recibido, sino cuando haya pasado cierto tiempo.
  - Según la RFC 1122 este tiempo debería ser menor de 500 ms, y normalmente se establece en 200 ms.
  - Cuando una entidad TCP reciba un segmento nuevo, pero aún tenga un hueco en la secuencia, automáticamente enviará un ACK asintiendo el último byte recibido sin huecos.
    - Este tipo de ACK se denominan ACK puros (segmentos con ACK="1" y que no llevan datos de usuario), y no son tratados por TCP de manera confiable (es decir, no se retransmiten ni se controla su pérdida).

4. Protocolo de Control de Transmisión (TCP)

Control de flujo

- Cuando una aplicación necesita enviar datos a otra sobre TCP, escribe los datos en la entidad local de TCP, la cual encola los datos y periódicamente los empaqueta y se los pasa a IP para que los envíe al sistema destino.
- Uno de los elementos clave de este proceso es el control de flujo: son las acciones que lleva a cabo una entidad emisora para adaptar su velocidad de envío a la capacidad de recibir datos por el destinatario.
  - Esta capacidad puede variar debido a múltiples parámetros, ya sea capacidad del receptor, carga de trabajo, estado de la red, etc.
- TCP lleva a cabo esta tarea con dos mecanismos fundamentalmente:
  - Control del tamaño de la ventana de recepción.
  - Ventanas de recepción deslizantes.

4. Protocolo de Control de Transmisión (TCP)

Control de flujo

Control del tamaño de la ventana de recepción

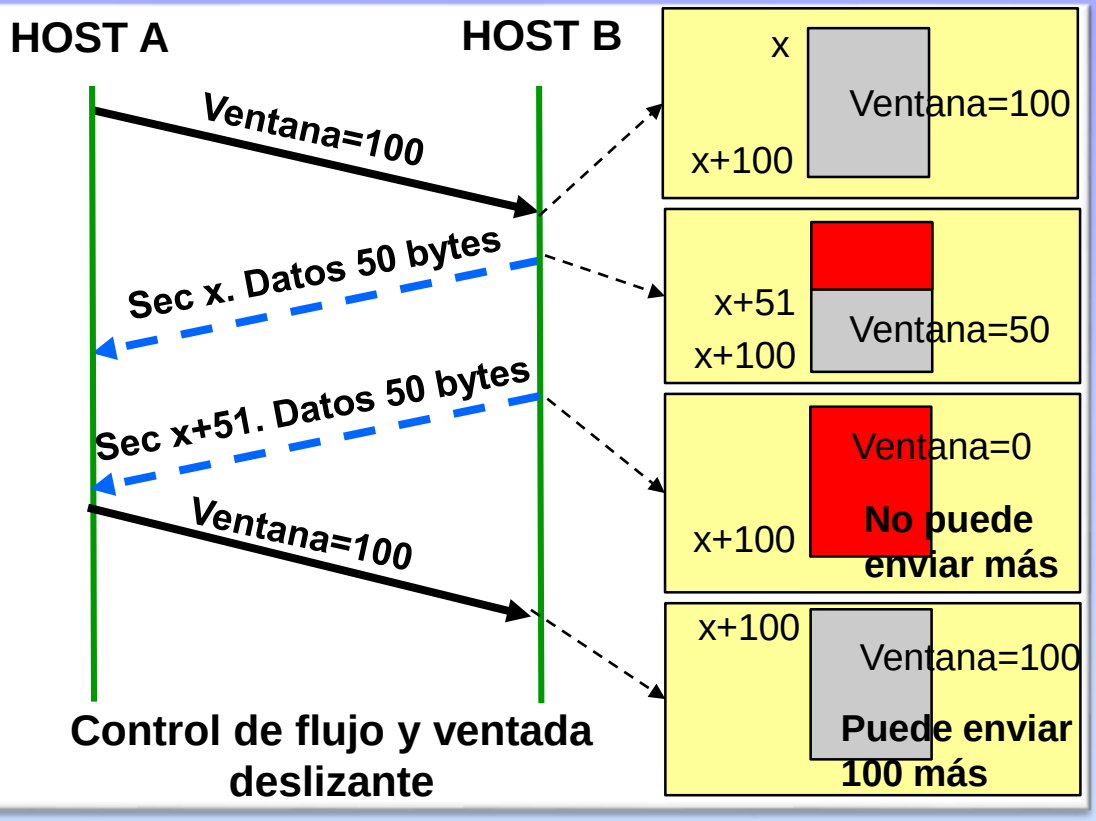
- Una entidad emisora TCP puede enviar solamente la cantidad de bytes que el receptor le permite en cada momento.
  - Una entidad TCP informa de los bytes permitidos a otra entidad TCP empleando el campo de ventana de los segmentos en envía, y viceversa.
- Esto depende de:
  - Espacio libre en la entidad receptora.
  - La frecuencia con la que los buffers se vacían.
  - El caudal de la red.

4. Protocolo de Control de Transmisión (TCP)

Control de flujo

Ventanas de recepción deslizantes

- Conjuntamente con el control de la ventana de recepción, las entidades TCP usan la ventana deslizante.
  - Permite enviar datos aun quedando algunos pendientes, esperando recibir los correspondientes asentimientos más tarde.
  - Algunos problemas iniciales de este esquema fueron el ***síndrome de la ventana tonta***.



Control de flujo y ventada deslizante

4. Protocolo de Control de Transmisión (TCP)

Control de flujo

Ventanas de recepción deslizantes

Síndrome de la ventana tonta

- Este problema se ocasiona cuanto una entidad TCP está saturada:
  - La entidad emisora recibirá segmentos de manera alternada con la ventana a cero, o con pocos bytes cada vez que la otra entidad hubiera procesado unos pocos datos.
- Esto fue corregido en la RFC 1122:
  - Tan solamente se permite que una aplicación vuelva a poner un tamaño de ventana mayor que cero, después de un periodo de inactividad, cuando:
    - Tiene tantos datos como un MSS.
    - o la cantidad de datos es la mitad de la ventana normal.

4. Protocolo de Control de Transmisión (TCP)

Control de flujo

Algoritmo de Nagle

- **Definido en la RFC 896 y recomendado su uso, no obligatorio, en la RFC 1122.**
  - Es comúnmente implementado en las implementaciones actuales de TCP, con ligeras variaciones.
- **Problema:**
  - Baja eficiencia de protocolo con IP/TCP al enviar pocos datos.
    - IPv4 cabecera 20 bytes, más cabecera TCP (20 bytes mínimo): 40 bytes.
    - IPv6 cabecera 40 bytes mínimo, más cabecera TCP: 60 bytes mínimo.
  - Así, mensajes por debajo de estos valores, desaprovechan altamente la capacidad de la red.

4. Protocolo de Control de Transmisión (TCP)

Control de flujo

Algoritmo de Nagle

Funcionamiento

- Así el algoritmo de Nagle, hace que los segmentos sean agrupados y retardados mientras:
  - Haya datos pendientes de asentir y:
    - Haya menos datos que un MSS.
    - Haya menos datos que una facción de la ventana del receptor (el factor normalmente es 1/2).
    - No sea un envío con PUSH.
  - De esta forma, un segmento pequeño tan solamente será enviado si no hay datos pendientes de asentir o se utiliza el mecanismo PUSH.

4. Protocolo de Control de Transmisión (TCP)

Control de flujo

Algoritmo de Nagle

Aplicabilidad

- El algoritmo de Nagle no es 100% aplicable, y puede hacer que aplicaciones interactivas visuales no funcionen correctamente.
- Sin embargo, el beneficio ante aplicaciones mal diseñadas, con múltiples paquetes pequeños puede ser significativo, aunque siempre está ligado a la propia funcionalidad de la aplicación.
- Por eso, debe existir un mecanismo para ser desactivado.

4. Protocolo de Control de Transmisión (TCP)

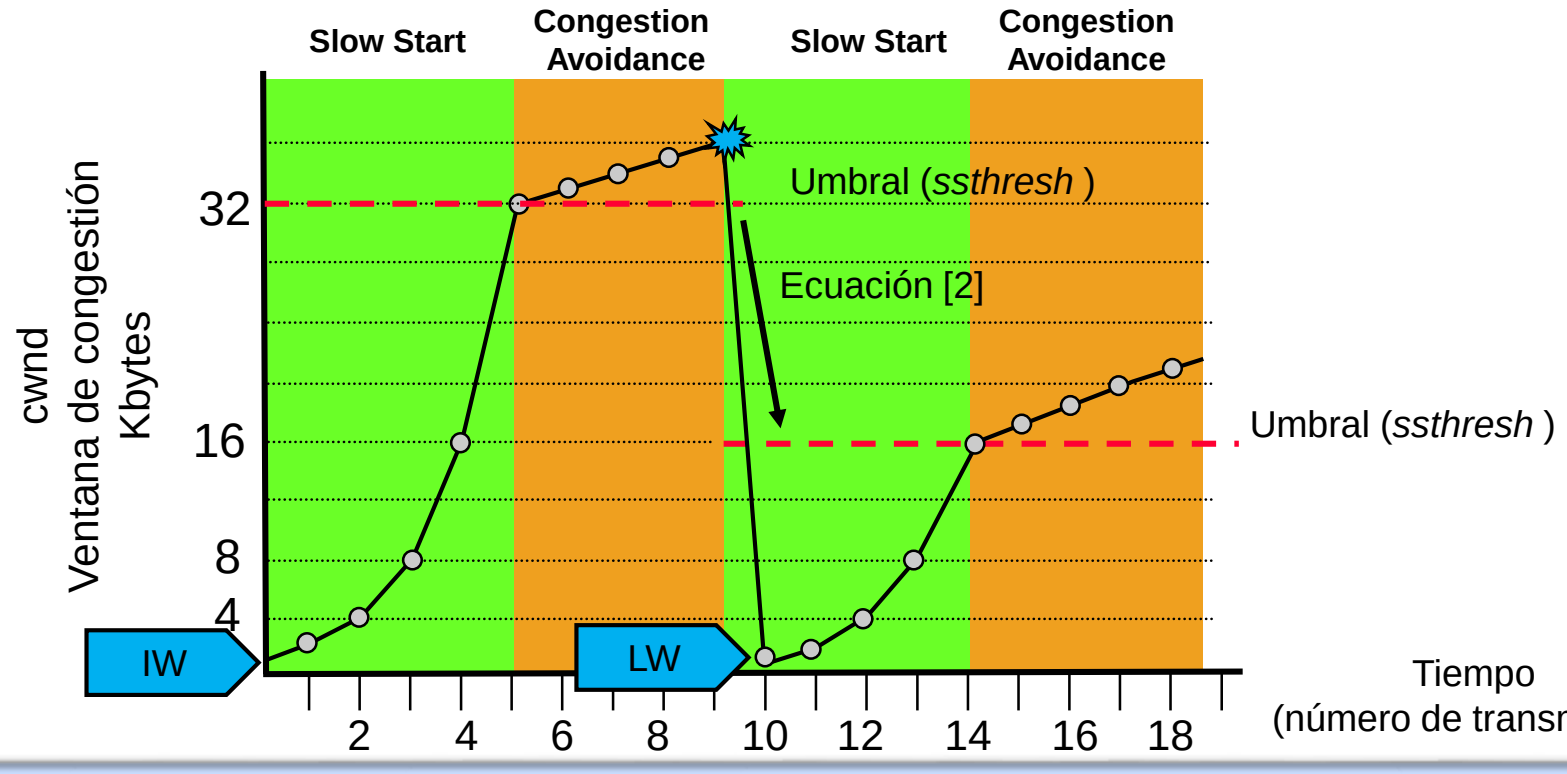
Control de congestión (TCP Reno)

- El control de congestión básico de TCP está definido en la RFC 5681 y complementado con la RFC 6298 Computing TCP's retransmission Timer y es de obligatoria implementación.
  - Versiones anteriores: 2581 y 2001.
- A la versión de TCP que incorporó esta modificación se la denominó TCP Reno.
- El control de congestión se modela por dos comportamientos o fases:
  - Arranque lento (Slow-start).
  - Prevención de la congestión (Congestión Advoidance).

4. Protocolo de Control de Transmisión (TCP)

Control de congestión (TCP Reno)

Ejemplo evolución del control de congestión



4. Protocolo de Control de Transmisión (TCP)

Control de congestión (TCP Reno)

Parámetros

- ***cwnd (Congestion Window):*** *ventana de congestión. La ventana de congestión limita la cantidad de información que puede enviar la entidad de transporte junto con la ventana del control de flujo advertida por el otro extremo.*
- ***ssthresh (Slow Start Threshold):*** Umbral de arranque lento.
- ***SMSS:*** Sender Maximum Segment Size. Es el tamaño del segmento más largo que el emisor puede transmitir, el cual se puede basar en la MTU de la red o en algoritmos como ***Path MTU Discovery algorithm*** [RFC1191] [RFC4821].
- ***flightsize:*** Cantidad de datos que se encuentran en la red, es decir, que han sido enviados y aún no han sido asentidos.
- ***IW (Initial Window):*** Ventana inicial en arranque lento.
- ***LW(Loss Window):*** Ventana de pérdidas a emplear cuando se produzca una pérdida de temporizador.

4. Protocolo de Control de Transmisión (TCP)

Control de congestión (TCP Reno)

Parámetros

Qué es un ACK duplicado.

- **Para que un ACK se considere duplicado**:
  a) El receptor del ACK aún tiene datos pendientes.
  b) El ACK recibido no tiene datos.
  c) Los bits SYN y FIN están desactivados.
  d) El número de asentimiento es igual al mayor número ya recibido en un asentimiento.
  e) El tamaño de la ventana anunciada en el segmento de ACK es igual que la del último ACK.

4. Protocolo de Control de Transmisión (TCP)

Control de congestión (TCP Reno)

Fase de Arranque Lento (Slow Start)

- La Ventana de Inicio (IW) del arranque lento *cwnd* debe ser como máximo:

```
If SMSS > 2190 bytes:
   IW = 2 * SMSS bytes and MUST NOT be more than 2 segments
If (SMSS > 1095 bytes) and (SMSS <= 2190 bytes):
   IW = 3 * SMSS bytes and MUST NOT be more than 3 segments
If SMSS <= 1095 bytes:
      IW = 4 * SMSS bytes and MUST NOT be more than 4 segments
```

4. Protocolo de Control de Transmisión (TCP)

Control de congestión (TCP Reno)

Fase de Arranque Lento (Slow Start)

Comportamiento

Cada vez que se recibe un ACK ➡ $$cwnd = cwnd + \min(N, SMSS)$$ **(1)**

N es el número de bytes asentidos en el ACK entrante

Pérdida de un segmento
Se aplica la fórmula si el segmento perdido no ha sido reenviado aún, si ya se ha enviado *ssthresh* se mantiene constante.

➡ $$ssthresh = \max\left(\frac{flightsize}{2}, 2 \times SMSS\right)$$ **(2)**

Además, cuando se produce una pérdida *cwnd* se establece al valor de *LW* (*Loss Window*) que es de un segmento completo.

4. Protocolo de Control de Transmisión (TCP)

Control de congestión (TCP Reno)

Fase de Prevención de Congestión (Congestion Avoidance)

- Se está en esta fase cuando se cumple la condición:

$$cwnd \geq ssthresh$$

Cada vez que se recibe un ACK ⇒

$$cwnd = cwnd + \left(SMSS \times \frac{SMSS}{cwnd} \right) \quad \textbf{(3)}$$

La RFC especifica que no se debe aumentar *cwnd* más de un *SMSS* por cada *RTT (Round Trip Time)*.

4. Protocolo de Control de Transmisión (TCP)

Control de congestión (TCP Reno)

Fast Restransmit y Fast Recovery

- Estos dos algoritmos, inicialmente descritos en la RFC 2581 (ahora RFC 5681) fueron diseñados con el objetivo de que TCP se comportara mejor ante la pérdida de un segmento, evitando esperas o retransmisiones innecesarias.
  - Cuando estos algoritmos detectan que un segmento enviado ha podido perderse, lo reenvían, intentando que no expire el temporizador de retransmisión.
- Esta mejora se denomina TCP RENO.

4. Protocolo de Control de Transmisión (TCP)

Control de congestión (TCP Reno)

Fast Restransmit y Fast Recovery

- Estos algoritmos se basan en una serie de requisitos que se deben cumplir por parte de una entidad TCP:
  - La entidad receptora debería enviar inmediatamente un ACK duplicado si recibe un segmento fuera de orden.
  - La entidad TCP receptora debería enviar un ACK inmediatamente después de recibir un segmento que rellenara un hueco en el espacio de secuencias de asentimientos.

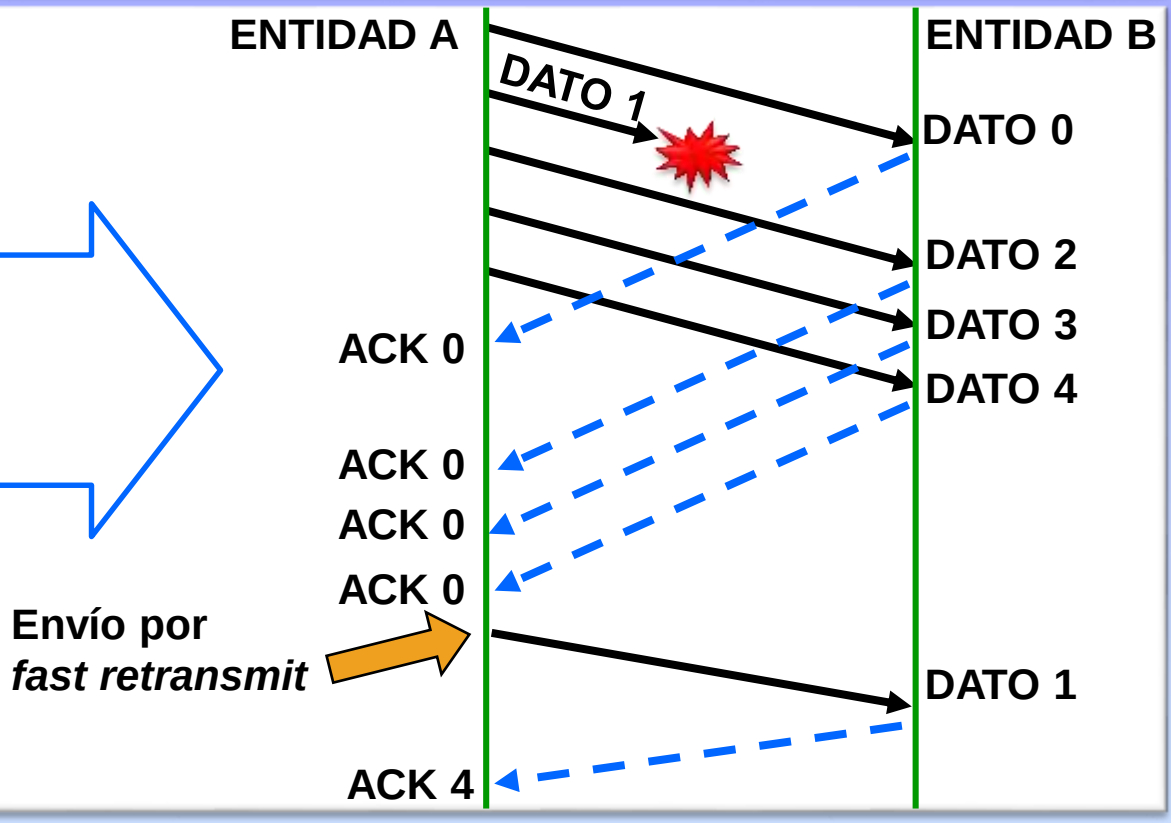
4. Protocolo de Control de Transmisión (TCP)

Control de congestión (TCP Reno)

Fast Restransmit y Fast Recovery

Fast Retransmit:

Este método permite reenviar un segmento cuando se reciben varios ACK repetidos, en concreto tres (este valor es empírico), sin tener que esperar a la expiración del temporizador, evitando ejecutar el algoritmo de arranque lento con la consecuente pérdida del aprovechamiento de la red



4. Protocolo de Control de Transmisión (TCP)

Control de congestión (TCP Reno)

Fast Restransmit y Fast Recovery

Fast Recovery (1/2):

Diseñado para el intervalo que transcurre entre que se detecta la pérdida de un segmento, a través de ACK duplicados y empieza el algoritmo *fast retrasnit*, y el momento en que llega un ACK no duplicado, es decir, cuando se asienten nuevos datos.

- Funcionamiento (Conjuntamente con fast retransmit)
  1. Establecer ***ssthresh*** al valor que resulta de la aplicación de la Ecuación 2.
  2. Retransmitir el segmento que se supone que se ha perdido y establecer ***cwnd*** al valor de ***ssthresh+3***. Este incremento refleja el hecho de que los tres ACK duplicados recibidos han abandonado la red y que el receptor de los segmentos que los originaron mantiene a éstos en memoria.

4. Protocolo de Control de Transmisión (TCP)

Control de congestión (TCP Reno)

Fast Restransmit y Fast Recovery

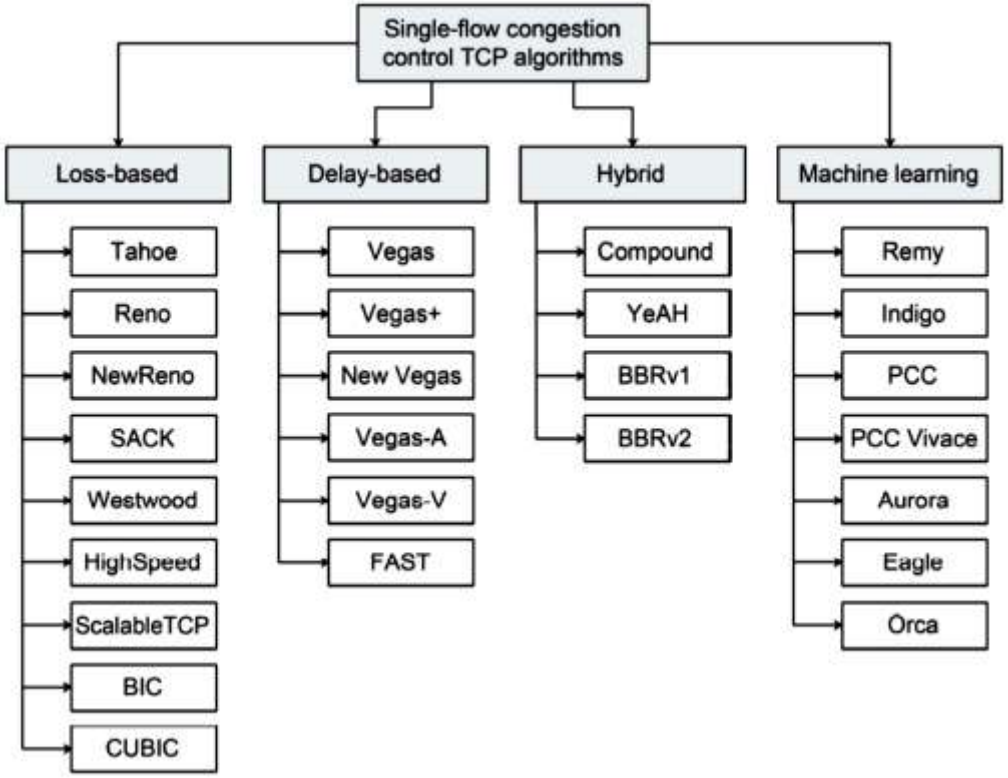
- Funcionamiento (Conjuntamente con *fast retransmit*)
  3. Para cada nuevo ACK duplicado se incrementa ***cwnd*** en el tamaño máximo de un segmento (SMSS), de esta forma se refleja el hecho de que un segmento ha dejado la red.
  4. Transmitir un segmento nuevo siempre que ***cwnd*** o la ventana de transmisión establecida por el receptor lo permita.
  5. Cuando llegue un ACK que asienta nuevos datos (no un ACK duplicado), este debe comprender el segmento perdido y todos los enviados después de este y hasta la recepción del tercer ACK duplicado. Entonces se establece el valor de ***cwnd*** al valor de ***ssthresh*** dado en el paso 1, finalizando el algoritmo de fast recovery.

4. Protocolo de Control de Transmisión (TCP)

Modificaciones al control de congestión de TCP

Evolución del control de congestión

- El control de congestión en TCP es un aspecto fundamental en el rendimiento de una conexión.
- Es objeto de estudios continuados y nuevas versiones.



Fuente: LORINCZ, Josip; KLARIN, Zvonimir; OŽEGOVIĆ, Julije. A Comprehensive Overview of TCP Congestion Control in 5G Networks: Research Challenges and Future Perspectives. *Sensors*, 2021, vol. 21, no 13, p. 4510.

4. Protocolo de Control de Transmisión (TCP)

Modificaciones al control de congestión de TCP

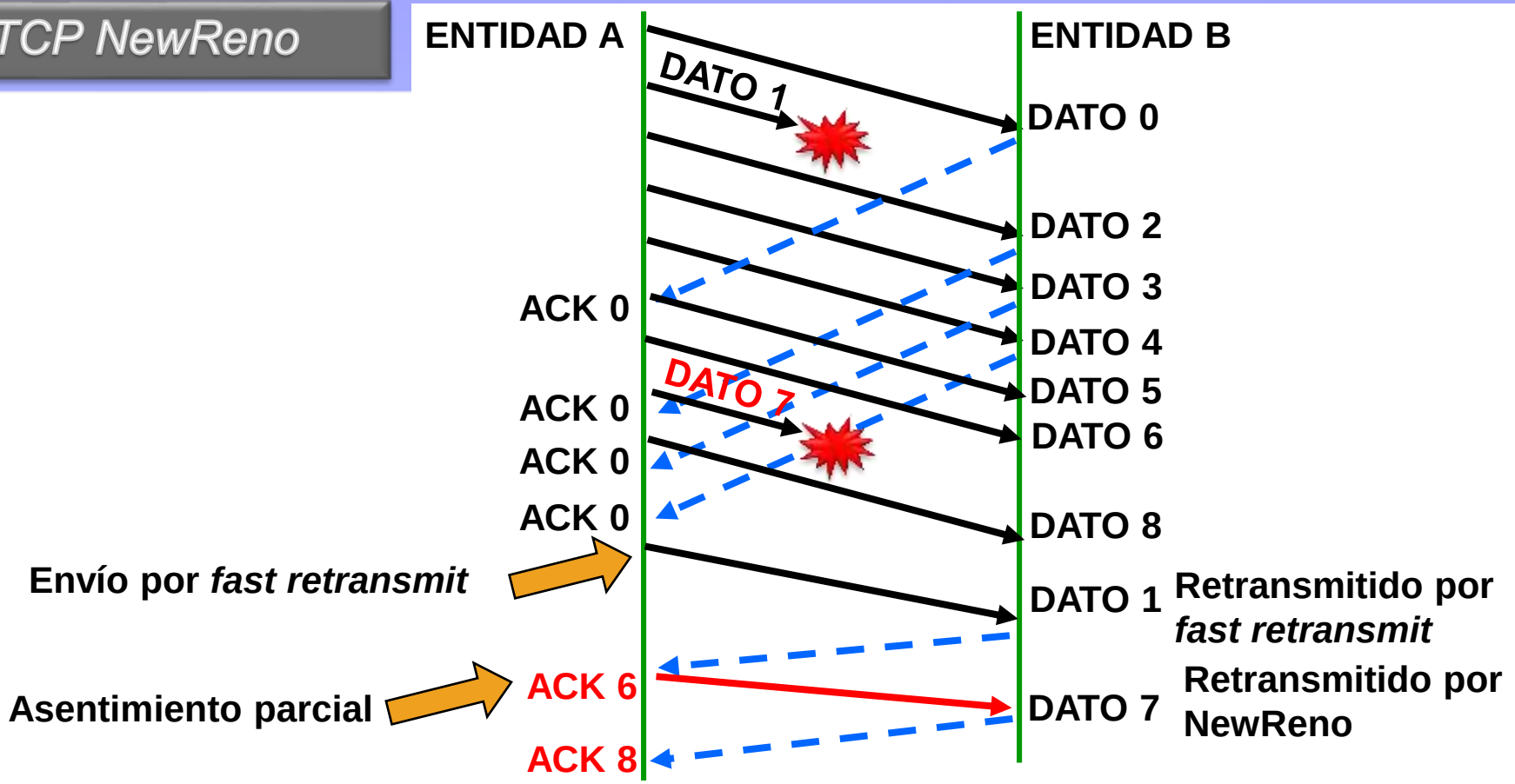
TCP NewReno

- **The NewReno Modification to TCP's Fast Recovery Algorithm (*RFC 6582*):** Modifica el comportamiento del algoritmo Fast Recovery, haciéndolo un poco más robusto frente a pérdidas múltiples para equipos que no tengan SACK dentro de su implementación de TCP o no lo usen.
  - Permite enviar nuevos segmentos para cubrir asentimientos parciales hasta que se asientan todos los datos pendientes.
    - El envío de este segmento supone que se ha perdido más de un segmento después del segmento que inició Fast Retransmit y Fast Recovery.
  - Al no alterar la modificación de la ventana (*inflation*) permite a la entidad de TCP enviar un segmento por cada ACK recibido.
  - Es más sencillo de implementar que SACK, aunque SACK es más eficiente, pero trae más problemas a la hora de gestionar la congestión.

4. Protocolo de Control de Transmisión (TCP)

Modificaciones al control de congestión de TCP

TCP NewReno



4. Protocolo de Control de Transmisión (TCP)

CUBIC

Evolución

- Especialmente diseñado para redes de alta capacidad y larga distancia, lo que se conoce como "*large bandwidth and delay product* (BDP)".
- Descrito inicialmente en: Ha, S., Rhee, I., and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", ACM SIGOPS Operating System Review, DOI 10.1145/1400097.1400105, July 2008.
  - También se detalla en una RFC informativa: RFC 8312 "*CUBIC for Fast Long-Distance Networks*" que se ha hecho estándar en la RFC 9438 de agosto de 2023.
- Es el mecanismo de control de congestión por defecto hoy en día en los sistemas Linux, Windows y Apple.
  - Con el comando "`Get-NetTCPSetting`" en el PowerShell de Windows 10/11 podemos ver esta configuración.
  - Versiones anteriores usaban Compound TCP y por compatibilidad se puede emplear también NewReno.

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Motivación

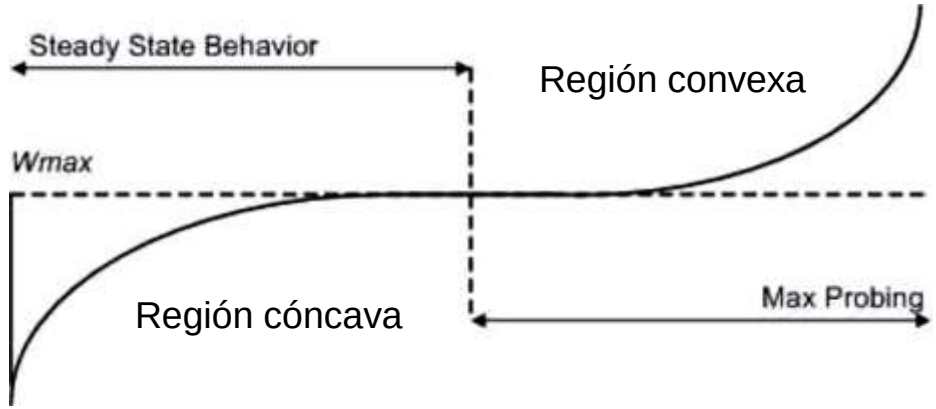
- Motivado por el lento crecimiento de la ventana de congestión, en mecanismos como TCP Reno y similares:
  - El crecimiento lento de la ventana *cwnd* tras un evento de congestión infrautiliza el caudal disponible en redes de alto producto ancho de banda (o *bandwidth-delay product* - BDP) y retardo (también conocidas como Long Fat Networks).
  - Para eso Cubic propone una **función cúbica de incremento de la ventana de congestión** para mejorar la escalabilidad y estabilidad en redes rápidas y de larga distancia.
  - También incorpora un mecanismo denominado **"Convergencia rápida"** para detectar si nuevos flujos se incorporan al canal y dejar ancho de banda libre.
- Es la evolución de *Binary Increase Congestion Control* (BIC-TCP) que fue adoptado por Linux en 2005.

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Principios de diseño de CUBIC

- Principio 1: para un mejor uso y estabilidad, CUBIC emplea tanto el perfil cóncavo como el convexo de una función cúbica para incrementar la ventana de congestión, en vez de emplear solamente el convexo.



Crecimiento de la ventana en CUBIC
Fuente: Ha, S., Rhee, I., and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", ACM SIGOPS Operating System Review, DOI 10.1145/1400097.1400105, July 2008

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Principios de diseño de CUBIC

- Principio 2: para ser TCP-friendly*, CUBIC está diseñado para comportarse como TCP estándar en redes con RTTs cortos y bajo ancho de banda, situación en la que TCP estándar se comporta bien.
- Principio 3: CUBIC está diseñado para conseguir una compartición lineal del ancho de banda entre flujos con diferentes RTTs.
- Principio 4: CUBIC establece el factor de decrecimiento multiplicativo de la ventana en 0,7 para conseguir un equilibrio entre escalabilidad y velocidad de convergencia.

*: Coexistir con otras implementaciones de TCP en una red sin entrar en conflicto con ellas.

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Función de incremento de ventana

- CUBIC mantiene la temporización de los ACK de TCP y solo incrementa la ventana de congestión cuando se recibe un ACK.
- CUBIC no modifica *fast recovery* o *fast retransmit* en su versión NewReno [RFC 6682].
- La función de incremento de ventana de CUBIC solo opera en la fase de prevención de la congestión (*congestion advoidance*) y después de un evento de congestión:
  - **Evento de congestión**: un ACK duplicado o la congestión de red se detecta por la recepción de un ACK con el flag ECN-Echo activo, la detección de pérdida de QUIC u otros métodos como ACK-TLP para TCP [RFC8985]. **La expiración del temporizador de retransmisión no es un evento de congestión.**
- Cuando ***cwnd*** es menor de ***ssthresh***, CUBIC debe emplear un algoritmo de arranque lento, el cual debería ser HyStart++ [RFC9406], o en el caso de no estar disponible, podría usar el de TCP Reno [RFC5681].

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Función de incremento de ventana

$$W_{cubic}(t) = C\,(t-K)^3 + W_{max} \quad \textbf{(4)}$$

Donde:

- C es un parámetro de CUBIC que determina su agresividad en la competencia por el ancho de banda con otros algoritmos (se recomienda un valor de 0,4 [RFC 9438]).
  - Mientras más bajo sea C, más amigable será con TCP Reno, pero peor se adaptará a las redes de larga distancia.
- t es el tiempo transcurrido desde el comienzo de la etapa actual de prevención de la congestión: $t = t_{current} - t_{epoch}$
- K es el periodo de tiempo en el que la función *W(t)* toma en incrementar *W* hasta W_{max}, siempre que no haya más eventos de pérdida, que se calcula de la siguiente forma:

$$K = \sqrt[3]{\frac{W_{max} - cwd_{epoch}}{C}} \quad \textbf{(5)}$$

- $\boldsymbol{cwd_{epoch}}$ es el valor de la ventana de congestión al inicio de la actual etapa de prevención de la congestión.

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Función de incremento de ventana

- Cuando se recibe un nuevo ACK durante prevención de la congestión computa la ventana de congestión objetivo (***target***) después de un RTT (*smoothed round-trip time* que se verá más adelante) usando la ecuación 4, como sigue:

$$target = \begin{cases} cwnd & si\ W_{cubic}(t + RTT) < cwnd \\ 1{,}5\ \times cwnd & si\ W_{cubic}(t + RTT) > 1{,}5\ \times cwnd \\ W_{cubic}(t + RTT) & en\ otro\ caso \end{cases} \quad \textbf{(6)}$$

- Los límites en ***target*** se aseguran de que el ratio de incremento no decrece y es menor que el ratio de incremento de arranque lento.

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Regiones de funcionamiento de CUBIC

- En función del valor actual del tamaño de ***cwnd***, CUBIC puede funcionar en tres regiones diferentes:
  - La **región Reno-friendly**, en la cual CUBIC se asegura que consigue el mismo desempeño que el control de congestión Reno en redes de ciertas características.
  - La **región cóncava**, si CUBIC no está en la región Reno-friendly y ***cwnd*** es menor que W_{max}. En esta región ***cwnd*** crece rápidamente al inicio y se ralentiza conforme se aproxima a W_{max}.
  - La **región convexa**, si CUBIC no está en la región Reno-friendly y ***cwnd*** es mayor que W_{max}. En esta región se crece con cuidado al inicio buscando una nueva ventana máxima de manera más rápida que en Reno una vez se ***cwnd*** se aleja de W_{max}.
- Tanto en la región cóncava como en la convexa ***cwnd*** se incrementa de la misma manera, pero, como se ha comentado existen diferencias en cuanto al propósito de cada región.

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Región Reno-Friendly

- Como se ha comentado el control de congestión de TCP Reno funciona bien en ciertos tipos de redes con RTTs cortos y bajos anchos de banda (BDP pequeño).
  - En este tipo de redes CUBIC permanece en la región Reno-friendly para conseguir un desempeño similar que TCP Reno.
  - El crecimiento de la ventana no sigue la fórmula de $W_{cubic}(t)$ sino la que define $W_{est}(t)$ (ver a continuación en la ecuación 7), que asemeja el crecimiento estándar de TCP en prevención de la congestión.
  - Cuando en prevención de la congestión se recibe un ACK, se actualiza $W_{est}(t)$ y si $W_{cubic}(t) < W_{est}(t)$ CUBIC está en la región Reno-friendly y por lo tanto ***cwnd*** se establece a $W_{est}(t)$.

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Región Reno-Friendly

$$W_{est}(t) = W_{est} + \alpha_{cubic} * \frac{segments\_acked}{cwnd} \quad \textbf{(7)}$$

- α_{cubic} se calcula para que el comportamiento en esta fase sea similar al de Reno (ver sección 4.3 de la RFC 9438).
- Al principio de la fase de prevención de la congestión $\boldsymbol{W_{est}}$ se establece en el valor de $\boldsymbol{cwnd_{epoch}}$.

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Región cóncava

- Si CUBIC no está en la región Reno-friendly y ***cwnd*** es menor que $\boldsymbol{W_{max}}$, entonces CUBIC se encuentra en la región cóncava.
- Cuando se recibe un nuevo ACK en la fase de prevención de la congestión en la región cóncava cwnd se debe actualizar como sigue:

$$\frac{target - cwnd}{cwnd}$$

- El cálculo de ***target*** se hace conforme a la ecuación [6].

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Región convexa

- Si CUBIC no está en la región Reno-friendly y ***cwnd*** es mayor o igual que W_{max}, entonces CUBIC se encuentra en la región convexa.
  - En esta región se supone que las condiciones de la red pueden haber cambiado desde el último evento de congestión, y puede haber más ancho de banda disponible (flujos que dejan la red o fluctuaciones en Internet).
  - Esta región también se denomina la "*maximum probing phase*" ya que CUBIC está buscando un nuevo W_{max}.
- Cuando se recibe un nuevo ACK en la fase de prevención de la congestión en la región cóncava ***cwnd*** se debe actualizar como sigue:

$$\frac{target - cwnd}{cwnd}$$

- El cálculo de ***target*** se hace conforme a la ecuación [6].

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Decremento multiplicativo (multiplicative decrease)

- Cuando se detecta evento de congestión, CUBIC actualiza W_{max} y reduce ***cwnd*** y ***sstresh*** inmediatamente. La reducción de estos dos parámetros se describe a continuación:
  - $\text{ssthresh} = flight\_size * \beta_{cubic}$ ➔ nuevo umbral de arranque lento
  - $\text{cwnd}_{prior} = \text{cwnd}$ ➔ se guarda el valor de cwnd
  - $cwnd = \begin{cases} \max(\text{ssthresh}, 2) & \text{➔ } reducción\ cuando\ es\ una\ pérdida \geq 2\ SMSS \\ \max(\text{ssthresh}, 1) & \text{➔ } reducción\ al\ recibir\ ECE \geq 1\ SMSS \end{cases}$
  - $\text{ssthresh} = \max(\text{ssthresh}, 2);$ ➔ El umbral debe ser al menos 2 MSS
  - β_{cubic} debería valer 0,7

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Convergencia rápida (fast convergence)

- Como se ha comentado, CUBIC también actualiza W_{max} (antes de aplicar la reducción de decremento multiplicativo) cuando se detecta evento de congestión, esto se denomina convergencia rápida:

$$W_{max} = \begin{cases} cwnd * \frac{1+\beta_{cubic}}{2} & Si\ cwnd < W_{max} \quad si\ la\ convergencia\ rápida\ está\ activa, \\ & lo\ que\ reduce\ W_{max} \\ cwnd & en\ otro\ caso, y\ se\ recuerda\ cwnd\ antes\ de\ la\ reducción \end{cases}.$$

- Convergencia rápida se ha diseñado para entornos de red con varios flujos CUBIC.
  - Si solamente se tuviera un flujo CUBIC, sin otro tráfico, debería ser desactivada.

4. Protocolo de Control de Transmisión (TCP)

CUBIC

Expiración del temporizador de retransmisión

- En el caso de una expiración del temporizador de retransmisión CUBIC sigue Reno, salvo que ***ssthresh*** se actualiza usando β_{cubic} como en el decremento multiplicativo.

4. Protocolo de Control de Transmisión (TCP)

Notificación Explícita de Congestión (ECN)

- La **Notificación Explícita de Congestión** o ECN (*Explicit Congestion Notificaction*) es un mecanismo que permite a un router que lo soporte, avisar a las **entidades de transporte** involucradas en comunicaciones sobre él de que **es probable que se dé un escenario de congestión**.
  - Definida en la RFC 3168 (Actualizada por las RFCs: 4301, 6040, 8311).
  - Se espera que los routers que implementen mecanismos de gestión activa de colas (*Active Queue Management* -AQM) descrito en al RFC 7567, usen la ECN cuando detecten una situación de congestion suave o moderada.

4. Protocolo de Control de Transmisión (TCP)

Notificación Explícita de Congestión (ECN)

Funcionamiento

- El mecanismo de ECN dicta que cuando un router detecte la proximidad de la congestión (de manera persistente, no simplemente cuando se descarte un paquete), debe marcar el datagrama IP saliente en los dos bits reservados para tal fin:
  - Se ponen a uno los dos bits destinados para ello en la cabecera de IP: *Congestion Experienced* (CE).
  - Los dos bits destinados al mecanismo ECN coinciden con los bits reservados del antiguo campo de 8 bits Type of Service de IPv4 o Traffic Class de IPv6, los restantes bits son usados por el mecanismo de QoS Servicios Diferenciados (DiffServ) para su gestión.

ECT	CE	RFC 2481 nombre ECN[Obsoleto]
0	0	Not-Ect
0	1	ECT (1)
1	0	ECT(0)
1	1	CE

Equivalentes

ECT: ECN-Capable Transport e indica que las capas de transporte implementan ECN

Interpretación del campo ECN en IP (RFC 3168)

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

- TCP usa tres tipos de temporizadores para controlar la conexión:
  - Temporizador de retransmisión (RTO).
  - Temporizador de persistencia.
  - Temporizador de seguir con vida (*Keep alive timer*).
- Además, también existe un temporizador de espera al finalizar una conexión:
  - Temporizador del estado TIME-WAIT: usado para esperar que todos los paquetes involucrados en la comunicación desaparezcan de la red.
    - Este tiempo es de 2 MSL (*Maximum Segmen Lifetime* = 2 minutos).

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de retransmisión (RTO Timer)

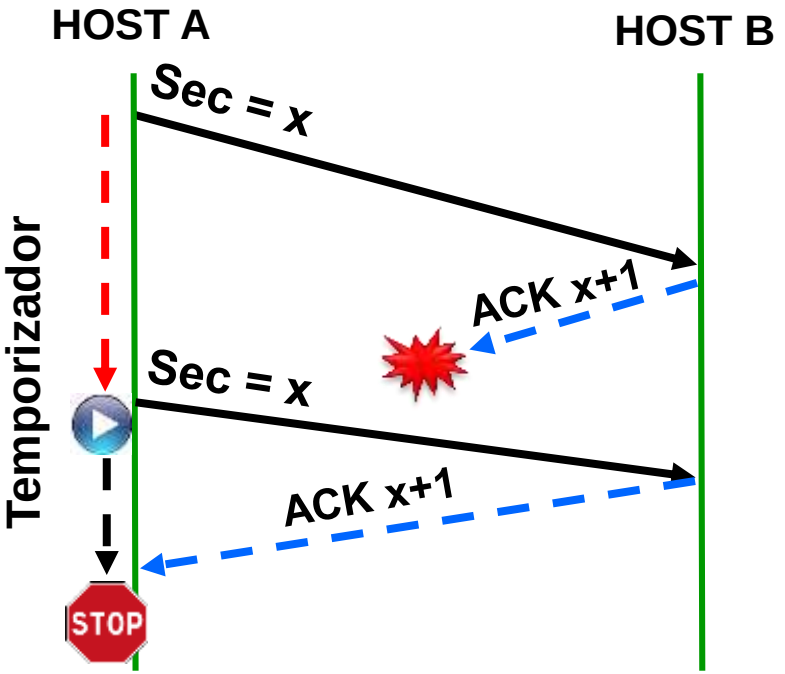
- Es el temporizador más importante dentro de una conexión, tiene como objetivo recuperar información perdida o dañada de manera transparente a las capas superiores.
- Al tiempo de espera se le conoce como RTO, que proviene de *Retransmission TimeOut*.
- Su cálculo se detalla en la RFC 6298 (que actualiza la RFC 1122).
  - Una implementación de TCP puede ser más conservadora que el algoritmo que se detalla en la RFC, pero nunca más agresiva.

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de retransmisión (RTO Timer)

- Algoritmo recomendado [RFC 6298]:
  - Al enviar un segmento con datos (incluso si es una retransmisión) se inicia el temporizador, si este no estaba activo, al valor actual de RTO.
  - Cuando todos los datos enviados estén asentidos, para el temporizador.
  - Cuando se reciba un ACK que asienta nuevos datos, reiniciar el temporizador al valor actual de RTO.



Ejemplo de retransmisión de un segmento

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de retransmisión (RTO Timer)

- El cálculo de este temporizador es complejo:
  - Se basa en el retardo de ida y vuelta o RTT (del inglés *Round Trip Time*)
  - El retardo de ida y vuelta del tráfico en la capa de transporte tiene una distribución muy dispersa en el tiempo.
  - Su media y varianza varían debido a las cambiantes condiciones de la red.
- Un error o desviación en el cálculo en el temporizador de retransmisión podría generar retransmisiones innecesarias si es demasiado corto, y uno demasiado largo retrasaría la detección de pérdidas reales de segmentos.

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de retransmisión (RTO Timer)

- Para solucionar esto se usa un algoritmo muy dinámico que se adapte a las evoluciones de la comunicación [Jac88].
  - [Jac88] Jacobson, V., "Congestion Avoidance and Control", Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988.
- Se mantienen dos variables de estado:
  - SRTT (*smoothed round-trip time*): RTT suavizado.
  - RTTVAR (*smoothed round-trip time variation*): Variación de RTT.
  - Se asume una granularidad de reloj G segundos.

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de retransmisión (RTO Timer)

Cálculo de RTO

- Hasta que se mide el primer RTT, RTO se establece a 1 segundo (anteriormente este valor era de 3 segundos, aunque sería aceptable cualquier valor mayor que 1 segundo).
  - Para más detalles sobre la bajada desde 3 se puede revisar el Anexo A de la RFC 6298.

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de retransmisión (RTO Timer)

Cálculo de RTO

- Cuando se mide el **primer RTT**, su valor se guarda en R y se actualizan las variables de estado de la siguiente manera:

```
SRTT <- R
RTTVAR <- R/2
RTO <- SRTT + max(G, K*RTTVAR)
```

Donde `K = 4`

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de retransmisión (RTO Timer)

Cálculo de RTO

- En la subsiguiente medidas de RTT, su valor se guarda en R' y se actualizan las variables de estado de la siguiente manera:

  (1°) RTTVAR <- (1-β) * RTTVAR + β* |SRTT - R'|

  (2°) SRTT <- (1 - α) * SRTT + α* R'

  Los valores de α y β deberían ser:

  - α = 1/8
  - β = 1/4

- Finalmente:

  RTO <- SRTT + max (G, K*RTTVAR)

- Se puede establecer un valor máximo, que debería ser de al menos 60 segundos.

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de retransmisión (RTO Timer)

Cálculo de RTO

- Granularidad del reloj (G).
  - No se establece un requisito concreto para este valor.
  - La experiencia ha mostrado que valores de G menores de 100 ms se comportan mejor que valores mayores.

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de retransmisión (RTO Timer)

Cálculo de RTO

- ¿Cuándo tomar una medida de RTT?
  - Al menos una medida cada RTT, salvo que esté activo el algoritmo de Karn (explicado a continuación).
  - Si se implementa la opción de *Timestamp*, cada ACK podría dar una medida.
  - Consejos:
    - Conexiones con ventanas de congestión muy grandes deberían tomar tantas medidas como sea posible.
    - Para ventanas de congestión pequeñas, medir el RTT para cada segmento no conduce a una mejora de la estimación de RTT.
    - Además, si se toman muchas muestras por cada RTT, los valores de α y β pueden generar una historia de RTT inadecuada.

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de retransmisión (RTO Timer)

- ¿Qué hacer ante una expiración del temporizador de retransmisión?
  - Retransmitir un segmento idéntico al más antiguo aún no asentido (cabeceras y datos).
  - `RTO <- RTO * 2` ("*back off the timer*") o algoritmo de Karn: Ante una pérdida por expiración del temporizador no actualizar RTT y duplicar la duración del temporizador de retransmisión, y eso se repite hasta que los segmentos se asienten de nuevo a la primera.
  - Iniciar el temporizador de retransmisión.
  - Si el temporizador expiró esperando el ACK de un segmento SYN, y la implementación de TCP usa un RTO menor de 3 segundos, el valor de RTO debe re-inicializarse a 3 segundos una vez la conexión se ha establecido.

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de persistencia

- Se activa cuando una entidad de transporte transmisora recibe una ventana de recetor con tamaño 0.
- Se usa para evitar el caso de que el segmento que permitiera de nuevo la transmisión se pierda y se quede bloqueada dicha entidad transmisora.
  - Por ejemplo cuando la notificación de un nuevo tamaño de ventana va en un ACK puro, los cuales no son retransmitidos si se pierden.

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de persistencia

- Una vez expirado el temporizador se envía un segmento de sondeo (*window probes* o sondas).
  - Las sondas son segmentos con un nuevo byte solamente.
- Si la respuesta tiene la ventana aún a cero, se reinicia el temporizador
- Si el receptor admite datos, se procede al envío normalmente y no se activa este temporizador.

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de persistencia

- El tiempo tomado como base es el del temporizador de retransmisión (RTO).
- A cada reintento este tiempo se duplica (por el algoritmo de Karn).
- Este temporizador no hace que la conexión se cierre, pudiendo estar en este estado indefinidamente.
  - Esto puede provocar un agotamiento de recursos, e incluso ha propiciado ataques basados en este temporizador.

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de seguir con vida

- Definido inicialmente en la RFC 1122 y no es obligatorio.
- Cuando una comunicación ha estado inactiva mucho tiempo este temporizador (`keepalive time`) puede expirar (La RFC 9293 establece que como mínimo debe ser de dos horas pero es configurable):
  - Se envían un número determinado de segmentos de sondeo (*keepalive probes*) a la otra entidad cada cierto tiempo (`keepalive interval`) solamente si no hay datos pendientes.
  - *Keepalive probe*: Mensaje ACK con el último byte ya asentido (no se envía información nueva)
  - Si no recibe respuesta a un número determinado de ellos (normalmente 9, pero pueden configurarse), se cierra la conexión.
    - Es posible que algunas implementaciones ni siquiera respondan a estos envíos.

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de seguir con vida

- Ejemplos de valores típicos (son configurables):

Sistema Operativo	keepalive time	keepalive interval	Configuración
MacOs	*7.200.000 milisegundos (2 horas)*	*75 segundos*	`sysctl` • `net.inet.tcp.keepidle` • `net.inet.tcp.keepintvl`
Windows	*7.200.000 milisegundos (2 horas)*	*1 segundo*	HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters – Parámetro: KeepAliveTime y KeepAliveInterval
Linux	*7.200.000 milisegundos (2 horas)*	*75 segundos*	`sysctl` • `net.ipv4.tcp_keepalive_time` • `net.ipv4.tcp_keepalive_probes = 9` • `net.ipv4.tcp_keepalive_intvl`

4. Protocolo de Control de Transmisión (TCP)

Administración de temporizadores en TCP

Temporizador de seguir con vida

- Este temporizador tiene algunos problemas:
  - Añade carga a la red con segmentos sin información nueva.
  - Puede terminar comunicaciones en buen estado, aunque inactivas, o por retardos en la red (si tuviera valores más bajos).
- De hecho, la RFC 1122 lo clasifica como totalmente opcional y configurable por el usuario.

5. Protocolo QUIC

- El protocolo QUIC es la apuesta de futuro para la WWW buscando una mayor eficiencia en la comunicaciones de este servicio con HTTP/3.
  - Se prevé que pueda ser usado por otras aplicaciones aparte de HTTP.
- Es un protocolo de transporte que se implementa sobre UDP para facilitar su despliegue.
- Posee mecanismos para la conexión rápida, multiplexación de streams y reusabilidad de conexiones.
- La protección se la aporta la integración con TLSv1.3.

IETF QUIC Working Group
https://quicwg.org/

5. Protocolo QUIC

Historia de QUIC

- Protocolo experimental, desplegado en Google a comienzos del 2014.
  - Empleado entre los servicios de Google y Chrome.
  - Su objetivo era mejorar la latencia de carga de las páginas y el re-buffer de los vídeos.
  - Hoy es un experimento exitoso.
  - Akamai lo desplegó en 2016.
- El grupo de trabajo QUIC del IETF se formó en el 2016.
  - Su función ha sido modular y estandarizar QUIC.
  - HTTP es su aplicación inicial.

5. Protocolo QUIC

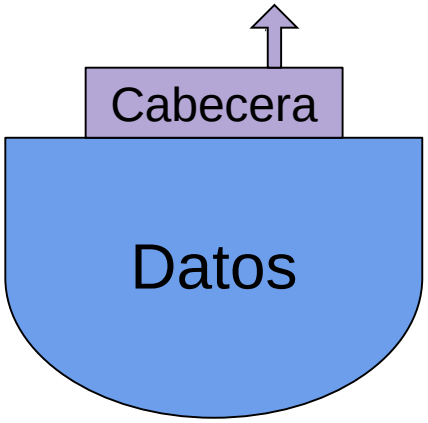
RFCs

- Thomson, M., "Version-Independent Properties of QUIC", **RFC 8999**, DOI 10.17487/RFC8999, May 2021, <https://www.rfc-editor.org/info/rfc8999>.
- Iyengar, J., Ed., and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", **RFC 9000**, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.
- Thomson, M., Ed., and S. Turner, Ed., "Using TLS to Secure QUIC", **RFC 9001**, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/info/rfc9001>.
- Iyengar, J., Ed., and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", **RFC 9002**, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/info/rfc9002>.

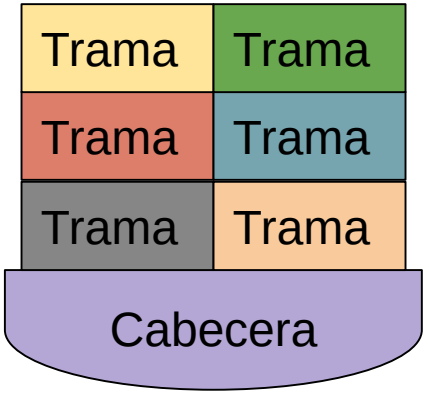
- Extensiones:
  - Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", **RFC 9221**, DOI 10.17487/RFC9221, March 2022, <https://www.rfc-editor.org/info/rfc9221>.
  - Thomson, M., "Greasing the QUIC Bit", **RFC 9287**, DOI 10.17487/RFC9287, August 2022, <https://www.rfc-editor.org/info/rfc9287>.

5. Protocolo QUIC

TCP

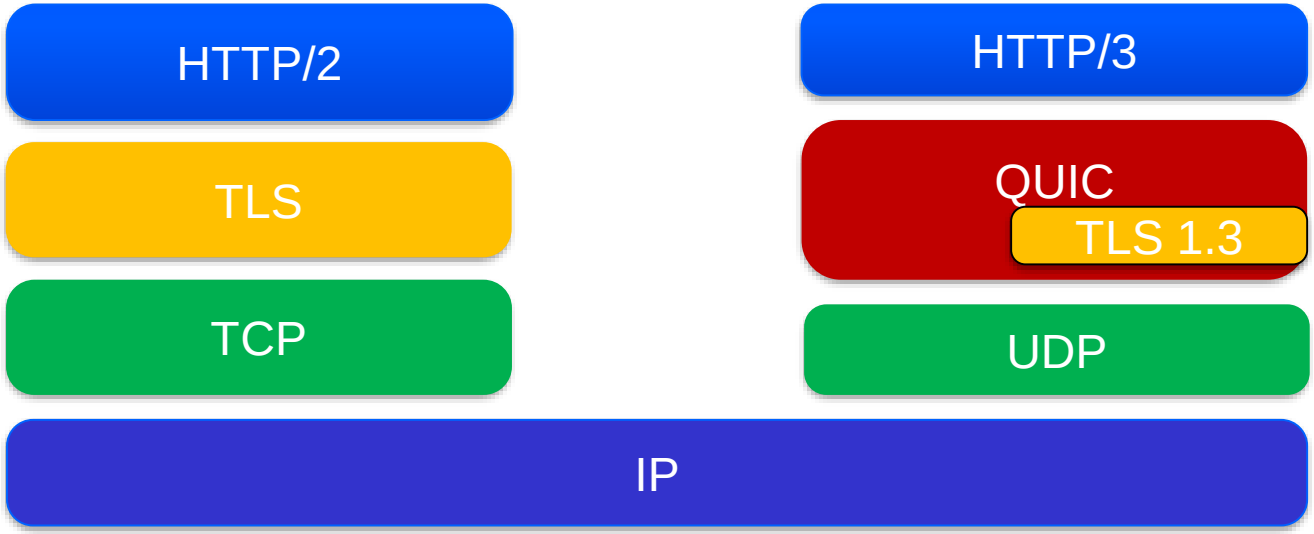


QUIC



5. Protocolo QUIC

Nueva torre de protocolos para el servicio web



5. Protocolo QUIC

¿Por qué QUIC?

- QUIC surge de la búsqueda constante de una menor latencia y mayor eficiencia en las comunicaciones del servicio web y la necesidad de solucionar algunos problemas de HTTP/2 sobre TCP y TLS, sobre todo:

Problema con HTTP/2: Bloqueo de encabezado de línea de TCP (TCP *head of line blocking*)

- Ante la pérdida de un segmento, TCP retiene los datos recibidos hasta que el segmento perdido se recibe correctamente, por lo que los streams de HTTP/2 se quedan bloqueados también.

5. Protocolo QUIC

¿Por qué QUIC?

¿Por qué sobre UDP?

- Crear un nuevo protocolo de transporte requeriría de que todos los sistemas operativos del mundo se actualizasen, además de añadir soporte en infinidad de equipos intermedios, y UDP ya está ahí con una cabecera ligera.
- Añadir comportamientos a TCP a través de su campo de opciones:
  - Puede derivar en que donde no se reconozcan las opciones no funcione el protocolo.
- HTTP/3 sobre SCTP [RFC 9260] (el protocolo que aportaba ya streams en capa de transporte): conexión a cuatro vías, sigue teniendo el bloqueo de cabecera, orientado a mensajes, sin una sólida trayectoria en seguridad con TLS y, además, QUIC puede migrar entre IPs pero SCTP no.

5. Protocolo QUIC

¿Por qué QUIC?

Osificación de Internet

- En Internet hay muchas versiones y tipos de dispositivos intermedios en Internet (*middle-boxes*), muchos de ellos pueden no estar actualizados, así que cuando uno de estos dispositivos interpretando un protocolo detecta algo que no va bien o no conoce, puede descartar o retrasar ese tráfico hasta niveles que hacen inútiles esas características para el usuario.
- ¿Cómo se combate la osificación?: encriptando tanta información como se pueda de una comunicación para que los *middle-boxes* no vean mucho del protocolo que los atraviesa.

5. Protocolo QUIC

¿Por qué QUIC?

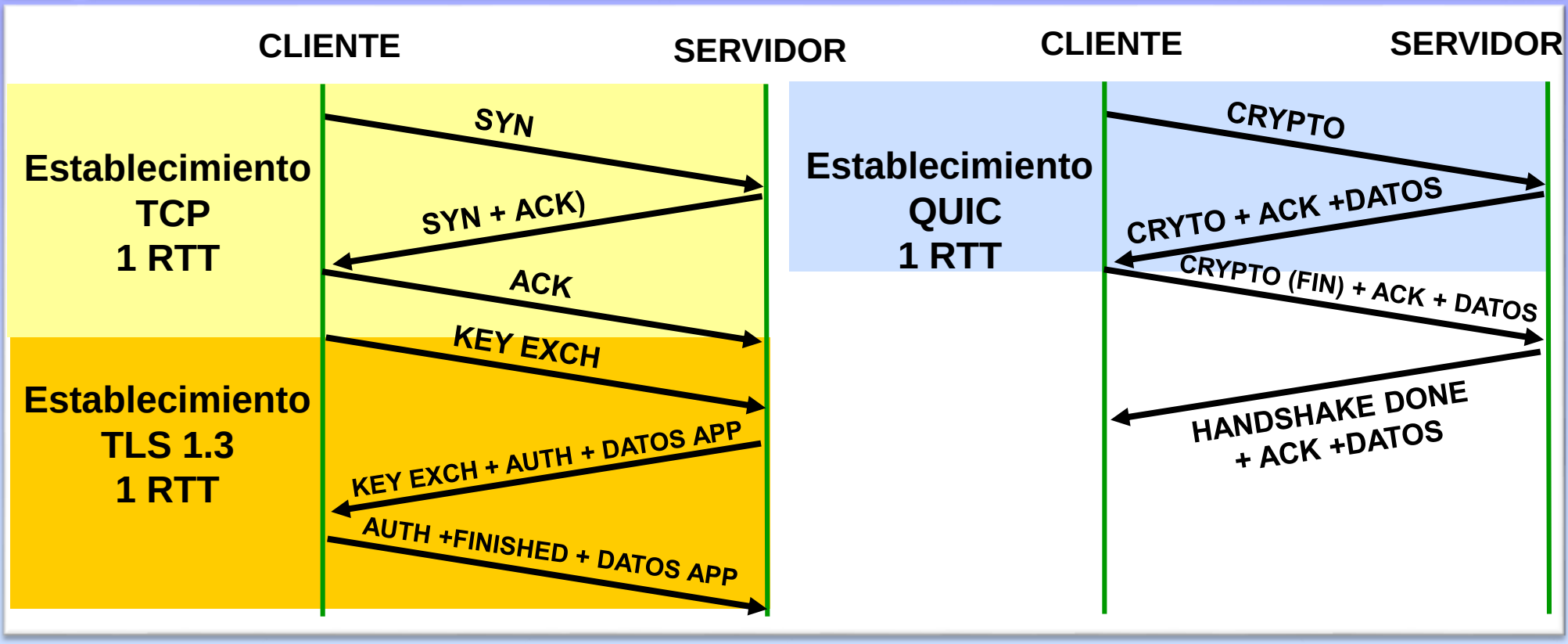
Reducción en la latencia de conexión

- QUIC, al tener incluida la negociación de TLS 1.3 consigue tener un proceso de conexión (o saludo, en inglés ***handshake***) de 1-RTT para una comunicación normal, e incluso un procedimiento de 0-RTT cuando se trata de retomar una comunicación previa.
  - QUIC consigue esta reducción al hacer una conexión con características criptográficas: *cryptographic handshake*.
  - TCP tiene el protocolo a tres vías, que más la negociación de TLS 1.3, se va a 2 RTTs.

5. Protocolo QUIC

¿Por qué QUIC?

Reducción en la latencia de conexión



5. Protocolo QUIC

¿Por qué QUIC?

Resumen del establecimiento 1-RTT de QUIC según la RFC 9000

```
Client                                                  Server

Initial[0]: CRYPTO[CH] ->

                                     Initial[0]: CRYPTO[SH] ACK[0]
                           Handshake[0]: CRYPTO[EE, CERT, CV, FIN]
                                     <- 1-RTT[0]: STREAM[1, "..."]

Initial[1]: ACK[0]
Handshake[0]: CRYPTO[FIN], ACK[0]
1-RTT[0]: STREAM[0, "..."], ACK[0] ->

                                                 Handshake[1]: ACK[0]
                  <- 1-RTT[1]: HANDSHAKE_DONE, STREAM[3, "..."], ACK[0]
```

5. Protocolo QUIC

¿Por qué QUIC?

Resumen del establecimiento 0-RTT de QUIC según la RFC 9000

```
Client                                                  Server

Initial[0]: CRYPTO[CH]
0-RTT[0]: STREAM[0, "..."] ->

                                     Initial[0]: CRYPTO[SH] ACK[0]
                                      Handshake[0] CRYPTO[EE, FIN]
                           <- 1-RTT[0]: STREAM[1, "..."] ACK[0]

Initial[1]: ACK[0]
Handshake[0]: CRYPTO[FIN], ACK[0]
1-RTT[1]: STREAM[0, "..."] ACK[0] ->

                                                 Handshake[1]: ACK[0]
                 <- 1-RTT[1]: HANDSHAKE_DONE, STREAM[3, "..."], ACK[1]
```

5. Protocolo QUIC

Características generales

- Los extremos QUIC intercambian **paquetes QUIC enviados sobre datagramas UDP**.
  - Un datagrama UDP puede incluir uno o más paquetes QUIC.
- **Los paquetes llevan una o más tramas** con información de control y datos de aplicación entre ambos extremos.
  - Todos los paquetes tienen un identificador único que se incrementa monotónicamente que indica el orden de transmisión que va de 0 a 2^{62}-1 (debido a la codificación de enteros de QUIC vista en el tema anterior).
  - Todos los paquetes son autenticados y además van encriptados (no completamente, se encripta tanto como es posible para que sea práctico).
  - QUIC define 20 tipos diferentes de tramas.
- Los protocolos de aplicación intercambian información sobre una conexión QUIC a través de *streams*, los cuales son secuencias ordenadas de bytes.
  - Existen dos tipos de *streams*: bidireccionales y unidireccionales.
  - Se establece un sistema de crédito para limitar la creación de *streams* y la cantidad de datos que puede ser enviada.

5. Protocolo QUIC

Características generales

QUIC v2

- La RFC 9369 define una nueva versión de QUIC, la versión 2, pero el documento solamente añade detalles menores y en sí QUIC se sigue considerando igual, ya que en ningún caso su intención es dejar obsoleta la versión 1.
  - Un motivo para este cambio es evitar algunos casos de osificación y añadir la negociación de versiones.
  - Como curiosidad, el número 2 no se emplea como número de versión realmente, para evitar riesgos de osificación.
  - También cambian los tipos de los paquetes de la cabecera larga.

5. Protocolo QUIC

Características generales

Tipos y formato de los paquetes

- Paquetes de cabecera larga (*Long Header Packets*):
  - Se emplean en paquetes enviados antes del establecimiento de las claves con 1-RTT.
    - Los tipos son: Initial, 0-RTT, Handsake y Retry.
    - Hay campos específicos a cada tipo que se añaden en el payload, como por ejemplo, el número de paquete.
    - El campo de número de paquete va protegido con una clave derivada de la principal.

```
Long Header Packet {
  Header Form (1) = 1,
  Fixed Bit (1) = 1,
  Long Packet Type (2),
  Type-Specific Bits (4),
  Version (32),
  Destination Connection ID Length (8),
  Destination Connection ID (0..160),
  Source Connection ID Length (8),
  Source Connection ID (0..160),
  Type-Specific Payload (..),
}
```

Formato del paquete con cabecera larga

5. Protocolo QUIC

Características generales

Tipos y formato de los paquetes

- Paquetes de cabecera corta (*Short Header Packets*):
  - Una vez se establecen las claves del saludo 1-RTT el emisor cambia a paquetes con cabecera corta.
  - El objetivo de cambiar de cabecera es reducir la sobrecarga de cabecera un vez se han negociado las características de seguridad.

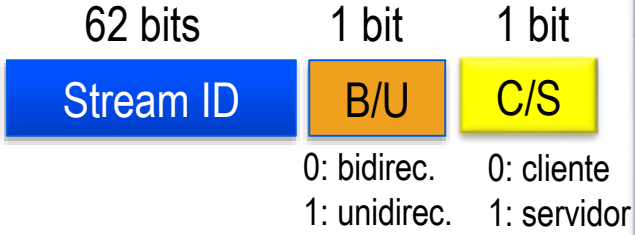
```
1-RTT Packet {
  Header Form (1) = 0,
  Fixed Bit (1) = 1,
  Spin Bit (1),
  Reserved Bits (2),
  Key Phase (1),
  Packet Number Length (2),
  Destination Connection ID (0..160),
  Packet Number (8..32),
  Packet Payload (8..),
}
```

Formato del paquete con cabecera corta

5. Protocolo QUIC

Streams

- Los streams proporcionan a la aplicación una forma ligera de mantener una abstracción de bytes ordenados dentro de cada uno de ellos.
  - Cada stream se parece a una conexión TCP.
- Un stream se inicia la primera vez que se envían datos sobre él.
- Los streams pueden ser unidireccionales o bidireccionales.
  - Los streams se identifican por un campo de 64 bits (stream ID).
    - El bit menos significativo indica quién inicia el stream, si el cliente(0), o el servidor (1).
    - El segundo bit menos significativo indica si el stream es bidireccional (0) o unidireccional (1).
    - Los 62 bits más significativos son el stream ID (0 a 2^{62}-1).



5. Protocolo QUIC

Envío de información

- La información de la aplicación se intercambia en tramas de tipo STREAM.
  - Cada trama incorpora un identificador de stream (Stream ID) y un desplazamiento (Offset) que las identifica únicamente.
    - El offset es equivalente al número de secuencia de TCP.
  - El intercambio de datos debe ser ordenado, por lo que toda información que se reciba fuera de orden debe ser almacenada hasta que se complete.
  - Los límites de una trama STREAM no delimitan la información de aplicación y no se espera que sean mantenidos en la transmisión de datos, retransmisiones ante la pérdida de paquetes o al ser enviados a la aplicación tras la recepción.
  - Un extremo no debe enviar información al otro extremo sin asegurarse que los datos están dentro del límite impuesto por el control de flujo de su par.

```
STREAM Frame {
  Type (i) = 0x08..0x0f,
  Stream ID (i),
  [Offset (i)],
  [Length (i)],
  Stream Data (..),
}
```

Formato de la trama STREAM

Tipo: 00001XXX (00001 OFF LEN FIN)

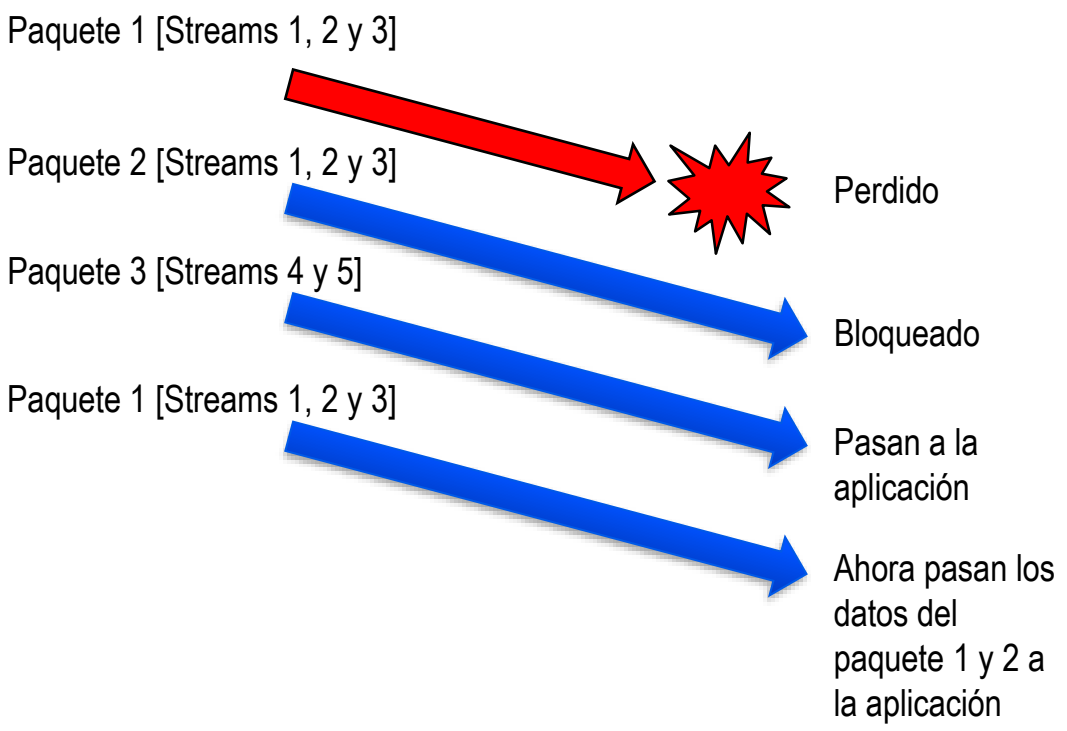
- Bit OFF (0x04): 1 presente offset, 0 no hay offset comienza en 0.
- Bit LEN (0x02): 1 campo de longitud presente, 0 el campo de datos se extiende al final del paquete.
- Bit FIN (0x01): a 1 indica que la trama marca el final del stream.

5. Protocolo QUIC

Envío de información

Solución del Head-of-line Blocking

- En QUIC, ante la pérdida de un paquete, solamente los streams que enviaron tramas en dicho paquete necesitarán esperar a la retransmisión.
  - Los demás streams podrán seguir progresando sin bloquearse.



5. Protocolo QUIC

Envío de información

Asentimiento de la información

- Dado que cada paquete enviado por QUIC tiene un único identificador que se incrementa, cuando un extremo asiente información envía el identificador más alto recibido sin huecos.
  - Todas las tramas menos las tramas ACK, PADDING y CONNECTION_CLOSE provocan el envío de ACK (*Ack-eliciting frames*). Por lo tanto, un paquete provoca el envío de ACK si porta alguna trama que genera ACK.
  - Si un paquete es asentido, esto implica que todas las tramas en dicho paquete se han recibido.
  - Además, en la trama ACK se pueden enviar hasta 256 huecos de asentimiento (como SACK en TCP, solo que en TCP solo se pueden enviar 3).

5. Protocolo QUIC

Envío de información

Detección y recuperación de pérdidas

- En QUIC un paquete se considera perdido si se cumplen las dos siguientes condiciones:
  - El paquete está sin asentir, en vuelo, y fue enviado antes que un paquete que sí ha sido asentido.
    - Esto se establece porque de esta manera se sabe que un paquete enviado después sí ha llegado al otro extremo.
  - El paquete fue enviado un número (*kPacketThreshold*) definido de paquetes antes que un paquete asentido, o fue enviado hace bastante tiempo en el pasado.
    - Estos umbrales se establecen para tolerar una posible reordenación de paquetes.

5. Protocolo QUIC

Envío de información

Detección y recuperación de pérdidas

- Es necesario detectar correctamente las pérdidas para evitar retransmisiones innecesarias y la degradación del rendimiento del control de congestión, el cual está vinculado a la detección de pérdidas.
- Para detectar la pérdida de paquetes finales se implementa un temporizador (*Probe Timeout*) que cuando expira sin recibir el ACK, se envía un paquete sonda para forzar un ACK.
  - Esta sonda puede llevar datos pendientes de asentir para reducir retransmisiones.
- La recuperación de una pérdida consiste en la retransmisión de las tramas consideradas perdidas en nuevos paquetes (con nuevo número de paquete sin correlación con los perdidos).

5. Protocolo QUIC

Envío de información

Cifrado de la información

- En los paquetes QUIC se cifra todo menos la cabecera de estos para poder procesarlos adecuadamente.
  - Es necesario tener en claro valores como el identificador de conexión, junto con ciertos flags necesarios para procesar la cabecera.

5. Protocolo QUIC

Envío de información

Envío no confiable de datagramas

- Para dar soporte a aplicaciones que no necesitan entrega confiable, como la transmisión de datos en tiempo real, QUIC proporciona un envío de datos no confiables, pero sí seguros (RFC 9221: An Unreliable Datagram Extension to QUIC).
  - Se emplean tramas DATAGRAM, las cuales no se retransmitirán en el caso de detectar una pérdida.
  - QUIC puede así proporcionar una conexión autenticada y confiable, junto con el envío de datos no confiables pero seguros.
  - Aunque en caso de pérdida los DATAGRAMAS no se reenvían, sí provocan el envío de ACKs y el emisor puede conocer si se ha recibido o no.

5. Protocolo QUIC

Control de flujo

- El control de flujo tiene como objetivo prevenir que un emisor rápido pueda sobrecargar a un receptor, o que un emisor malicioso consuma grandes cantidades de memoria.
- QUIC aplica el control de flujo individualmente a los streams o a la conexión en su conjunto.
  - También es posible limitar el número de streams (trama MAX_STREAMS) que el otro extremo puede iniciar.
- QUIC emplea un esquema de control de flujo basado en límites:
  - Un receptor avisa del límite total de bytes que está preparado para recibir, ya sea para un stream o para toda la conexión.
  - En el establecimiento de la conexión el receptor establece los límites iniciales para el control de flujo.
    - A partir de ahí emplea tramas MAX_STREAM_DATA (un stream) o tramas MAX_DATA (toda la conexión) para establecer límites superiores.
    - El valor que se establece es el máximo número de bytes que se pueden enviar por un stream o toda la conexión, según corresponda.

5. Protocolo QUIC

Control de congestión

- En QUIC el control de congestión está desacoplado del control de confiabilidad, así QUIC usa los números de los paquetes para el control de congestión y el offset de las tramas de un stream para la confiabilidad.
- El control de congestión de QUIC es similar a NewReno [RFC6582] de TCP.
  - QUIC proporciona señales genéricas para el control de congestión que habilitan para que un extremo implemente cualquier algoritmo diferente como por ejemplo CUBIC [RFC8312].
  - Soporta ECN [RFC3168] [RFC8311].
- Para evitar reducciones de la ventana de congestión innecesarias, QUIC solo colapsa la ventana de congestión cuando detecta **congestión persistente**:
  - Si dos paquetes que necesitan asentimiento se declaran como perdidos, se establece la **congestión persistente** si ninguno de los paquetes enviados entre ambos paquetes perdidos ha sido asentido.