TCP y UDP

Álvaro González Sotillo

10 de febrero de 2018

Índice

1. Introducción

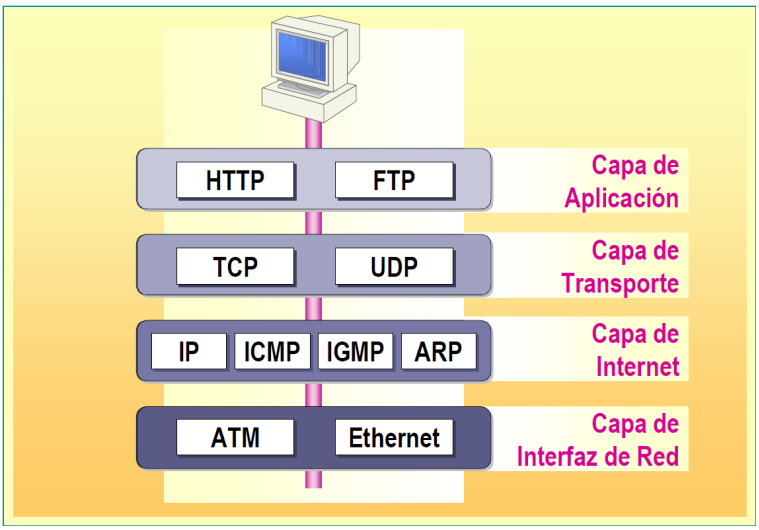


Figura 1: Esquema de niveles de red

- TCP y UDP son de la capa de transporte
- Es la primera que une dos entidades (procesos), en vez de dos hosts
- Suele ser la primera capa visible para los programadores de aplicaciones
    - IP se puede considerar la *frontera* entre los *administradores* y los *programadores*

2. UDP

- *User Datagram Protocol*
- Funciones:
  - Entregar un datagrama entre el emisor y receptor (procesos)
  - Detección de errores

2.1. Formato de trama UDP

	Bits 0 - 15	16 - 31
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

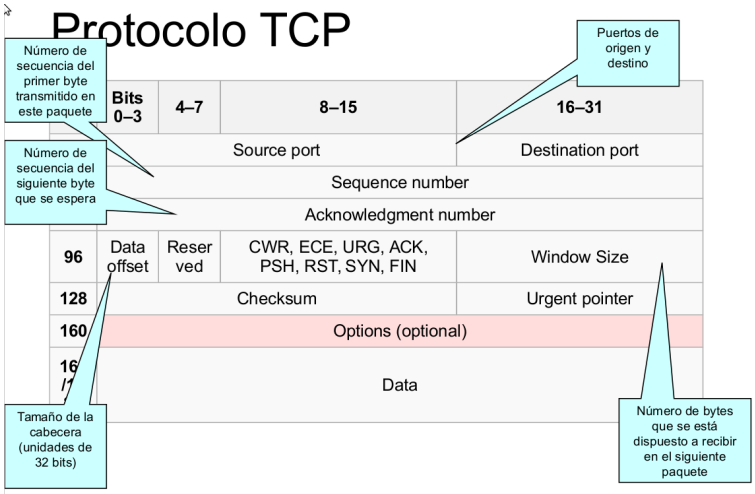
2.2. Características de UDP

- Los datagramas pueden llegar en un orden diferente al enviado (si IP elige rutas distintas para ellos)
- El emisor no tiene la seguridad de que los datagramas llegan al receptor
- Por tanto, no se utilizan **conexiones** ni es **confiable**. Cada datagrama se envía de forma independiente.

- ¿Cuántos puertos hay?
- ¿Cuál es el tamaño máximo de un datagrama UDP?

3. TCP

- *Transmission Control Protocol*
- Asegura que la transmisión se realiza por un medio fiable
- Garantiza la recepción de los mensajes en orden correcto
- Garantiza al emisor que los mensajes llegan correctamente al receptor
- Por tanto, es orientado a **conexión** y **confiable**.



3.1. Ventana y corrección de errores

- TCP necesita confirmación de cada mensaje enviado
  - Para garantizar la confiabilidad
- Opciones:
  - **Parada y espera**: Cada mensaje necesita confirmación
  - **Piggybacking**: La confirmación puede retrasarse algunos mensajes (ventana)
- La parada y espera es más simple, pero desaprovecha ancho de banda

3.1.1. Ejemplo de parada y espera

3.1.2. Ejemplo de *piggybacking*

3.2. *Sequence number* y *Acknowledgment number*

- **TCP** intenta que la comunicación se asemeje a un flujo de bytes
  - Todos los bytes que entran por un extremo
  - ... deben salir por el otro lado
- En **cada paquete** se envía
  - el número de secuencia del primer byte transmitido en el paquete
  - el número de secuencia del siguiente byte que se espera

3.3. Cómo se consigue la conexión

- Cada extremo de la comunicación debe saber:
  - Cuál es su siguiente byte a enviar (*sequence number*)
  - Cuál es su siguiente byte a recibir (*acknowledgment number*)

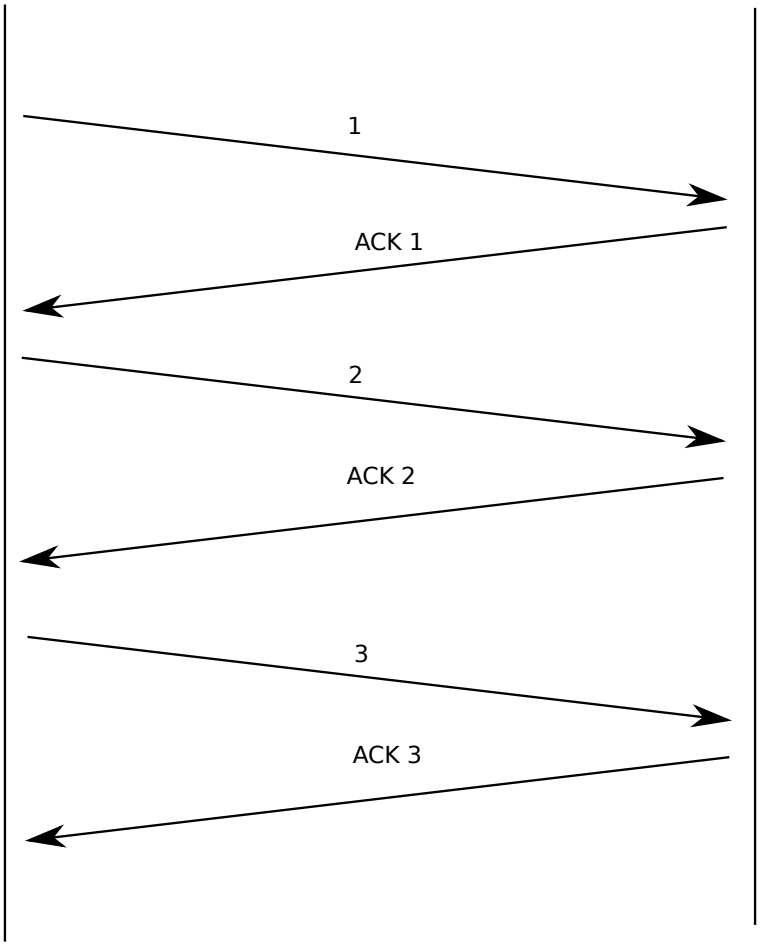


Figura 2: Ejemplo de parada y espera

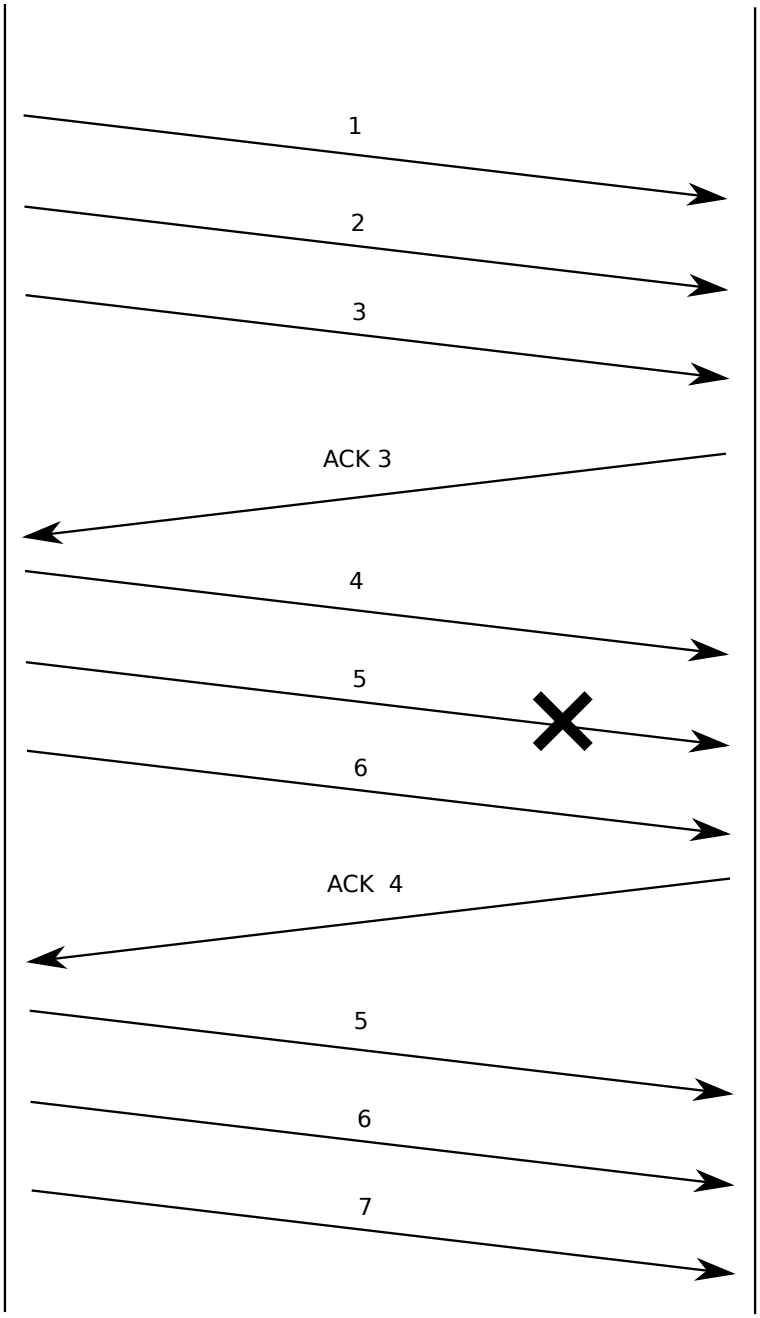


Figura 3: Ejemplo de piggybacking con ventana 3

- Además, también lleva la cuenta de su opinión acerca del *sequence number* y del *acknowledgment number* del otro extremo

3.4. Corrección de errores

- Si se recibe un *sequence number* posterior a nuestro *acknowledgment number*
  - Es un paquete **posterior** al que esperamos
    - Se puede guardar en la capa **TCP** hasta que lleguen los anteriores
    - Se puede ignorar, y reclamar los paquetes perdidos
- Si se recibe un *sequence number* anterior a nuestro *acknowledgment number*
  - Es un paquete ya recibido (se habrá duplicado)
  - Por tanto, se ignora
- Si se pierde un paquete, o llega con error
  - Enviaré un *acknowledgment number* menor que el que espera el otro lado
  - El otro lado reenviará los paquetes necesarios
- Si no tengo confirmación de un paquete enviado tras un *timeout*
  - Reenviaré el paquete

3.5. Ventana de transmisión

-

4. Referencias

- Formatos:
  - Transparencias
  - PDF
- Creado con:
  - Emacs
  - org-reveal