Capa de Transporte

Juegos en Red - Grado en Desarrollo de Videojuegos

Ruben Rodríguez
ruben.rodriguez@urjc.es
URJC

Natalia Madrueño
natalia.madrueno@urjc.es
URJC

2025-09-09

Universidad Rey Juan Carlos

DSLAB
DATA SCIENCE LABORATORY

Tabla de contenidos

- Introducción
- Funciones Principales
- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)
- Comparativa TCP vs UDP para Videojuegos
- Resumen

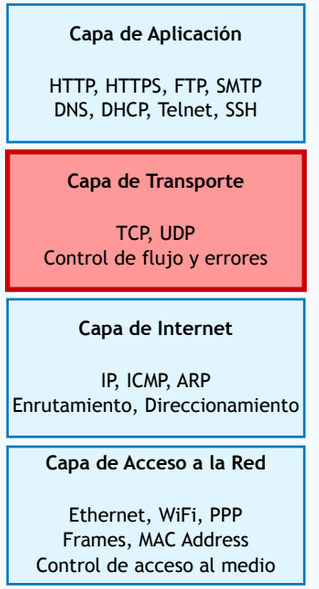
Introducción

¿Qué es la Capa de Transporte?

> La capa de transporte proporciona comunicación lógica entre procesos de aplicación que se ejecutan en diferentes hosts

Función principal:

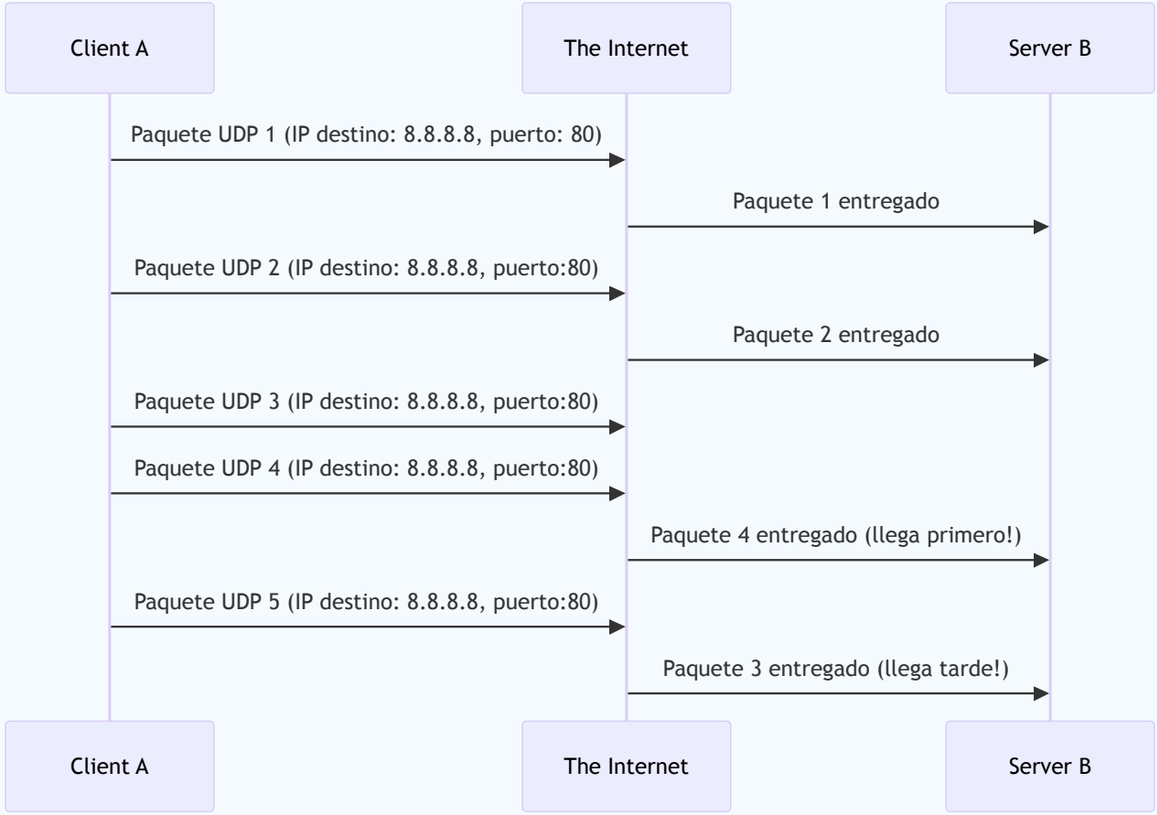
- Se ejecuta en hosts finales
- No en el núcleo de la red
- Divide mensajes en segmentos
- Recompone segmentos en el receptor



Protocolos principales:

- **UDP**: Protocolo minimalista
- **TCP**: Protocolo complejo con garantías

Ejemplo: Cliente-Servidor con UDP



Funciones Principales

1. Multiplexación y Demultiplexación

Multiplexación: Recoger información de diferentes sockets y enviarla por un único medio.

Demultiplexación: Recibir segmentos y enviarlos a los sockets correspondientes.

Identificación de sockets:

- TCP: (IP y Puerto origen, IP y puerto destino)
- UDP: (IP y puerto origen, IP y puerto destino)

Puertos: Identificadores numéricos (1-65535)

- Servidores: asignación manual y fija
- Clientes: asignación aleatoria

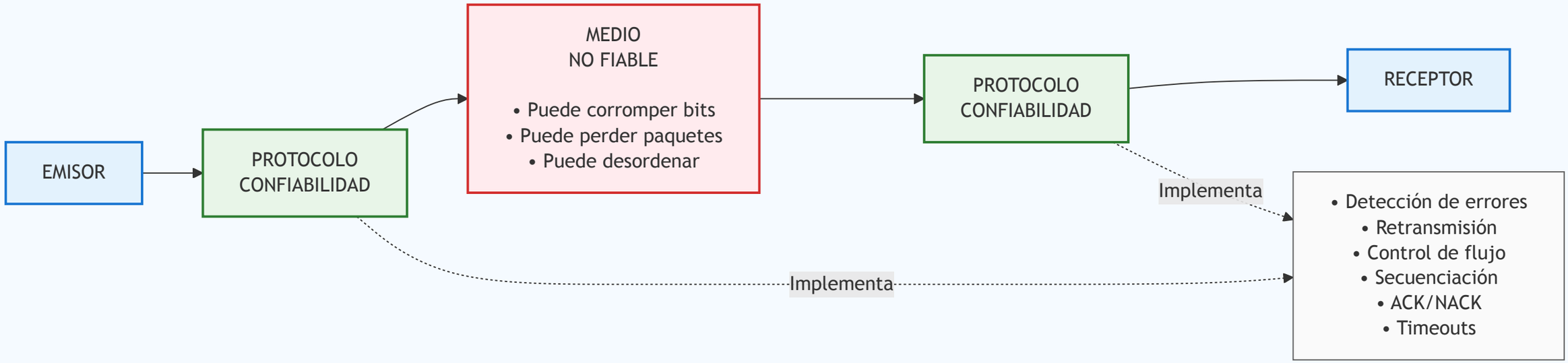
2. Transferencia Fiable

Características de una transferencia fiable:

- No se corrompe ningún bit
- No se pierde información (paquetes)
- La información se entrega en orden correcto

Opciones de implementación:

1. Usar protocolos fiables existentes (TCP)
2. Implementar características propias sobre protocolo no fiable (UDP + lógica aplicación)



Otros conceptos

- **Control de flujo**: Evita que el emisor sature al receptor limitando la velocidad de envío según la capacidad del destinatario
- **Control de congestión**: Ajusta la velocidad de transmisión para evitar saturar la red cuando detecta congestión
- **Temporización**: Proporciona el tiempo mínimo de entrega de datos entre emisor y receptor a través de la red
- **Tasa de transferencia mínima**: Garantiza una velocidad mínima de transmisión de datos para aplicaciones que requieren ancho de banda constante

UDP (User Datagram Protocol)

Características de UDP

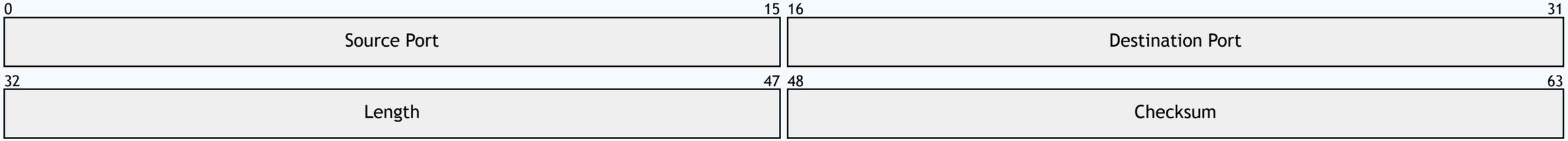
Protocolo minimalista [RFC 768]:

- Basado en best-effort (Fire-and-forget)
- No orientado a conexión
- Entrega no fiable y sin orden
- Integridad básica (checksum)
- Multiplexación y demultiplexación

Lo que NO proporciona:

- Control de flujo
- Control de congestión
- Temporización
- Tasa de transferencia mínima
- Seguridad

Estructura del Paquete UDP



- **Longitud:** Hasta 65535 bytes (limitado por MTU)
- **Checksum:** Verificación de integridad
- Estructura simple comparada con otros protocolos
- 8 bytes de cabecera fijos.

Checksum UDP

Proceso de cálculo:

1. Preparación: pseudo-cabecera IP + cabecera UDP + datos
2. División en palabras de 16 bits
3. Suma usando aritmética de complemento a uno
4. Complemento del resultado → campo checksum

Verificación en receptor:

- Mismo algoritmo incluyendo checksum recibido
- Resultado esperado: 0xFFFF
- Si difiere: datagrama descartado silenciosamente

Casos de Uso de UDP

Aplicaciones ideales para UDP:

- **Multimedia streaming:** Tolerante a pérdidas, sensible a interrupciones
- **DNS:** Respuestas rápidas necesarias
- **SNMP:** Administración de red
- **Gaming online:** Latencia baja crítica
- **QUIC/HTTP3:** Base para protocolos modernos optimizados

Ejemplo: Implementar protocolo propio sobre UDP para juegos

- Añadir número de paquete
- Descartar paquetes fuera de orden
- Ignorar duplicados
- Sobrecarga mínima

TCP (Transmission Control Protocol)

Características de TCP

Protocolo confiable [RFC 793]:

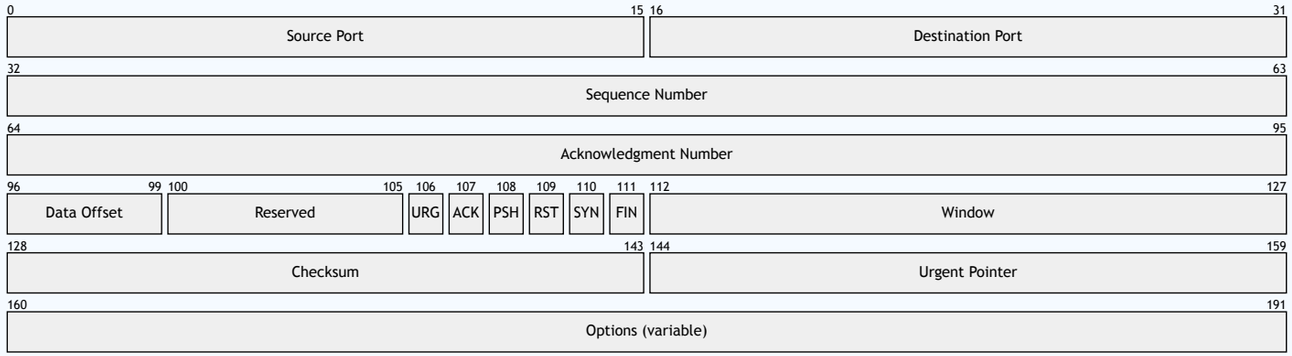
- Orientado a conexión
- Entrega fiable y ordenada (confiabilidad)
- Control de flujo
- Control de congestión
- Multiplexación y demultiplexación

Trade-off: Confiabilidad y orden sobre velocidad pura

Lo que NO proporciona:

- Temporización específica
- Tasa mínima garantizada
- Seguridad nativa (necesita TLS/SSL)

Estructura del Paquete TCP



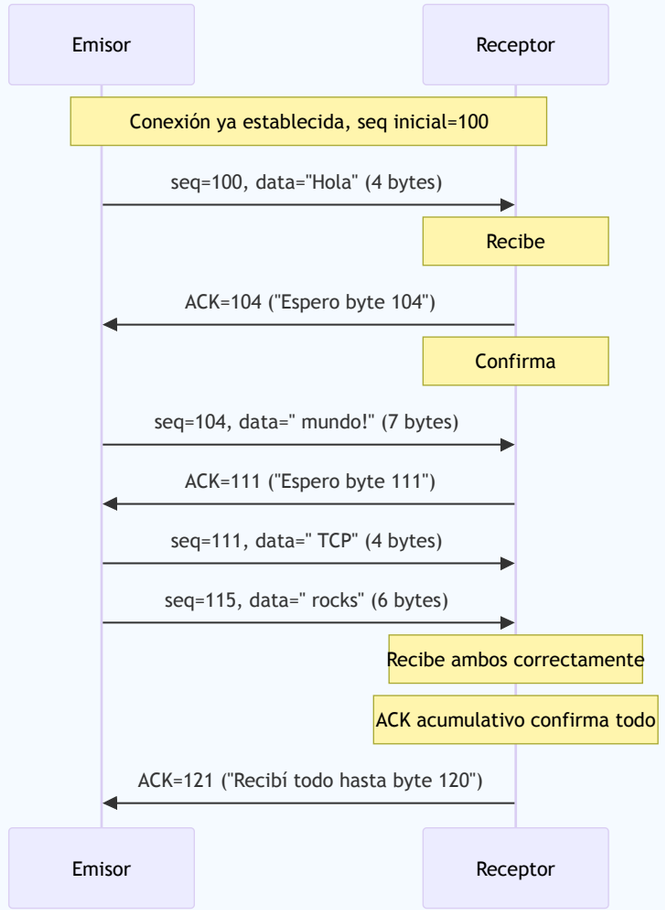
- **Sequence number**: Número que identifica la posición del primer byte de datos en el segmento dentro del flujo de datos.
- **Ack number**: Indica el próximo número de secuencia que el receptor espera recibir. Confirma la recepción correcta de datos anteriores.
- **Window:** Implementa control de flujo (bytes que receptor acepta)
- **checksum:** Muy similar a UDP.
- Tamaño variable de 20 a 60 bytes.

Mecanismos de Confiabilidad

> Los mecanismos de confiabilidad en TCP garantizan que los datos lleguen correctamente y en orden.

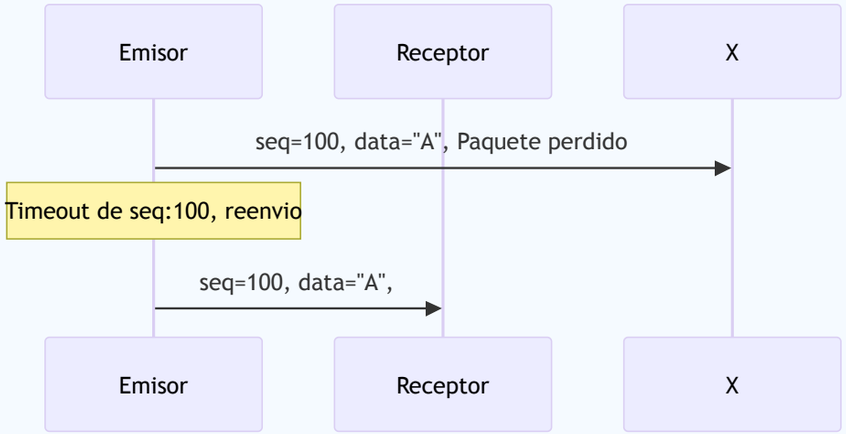
Números de secuencia y ACKs:

- Cada byte tiene número único
- Cuando enviamos información, está identificada por un número de secuencia (SEQ)
- Además, esperamos confirmación de que se ha recibido correctamente (ACK)
- ACKs acumulativos (ACK para byte N confirma hasta N-1)
- Los ACKs y SEQs permite detectar datos perdidos, duplicados o desordenados



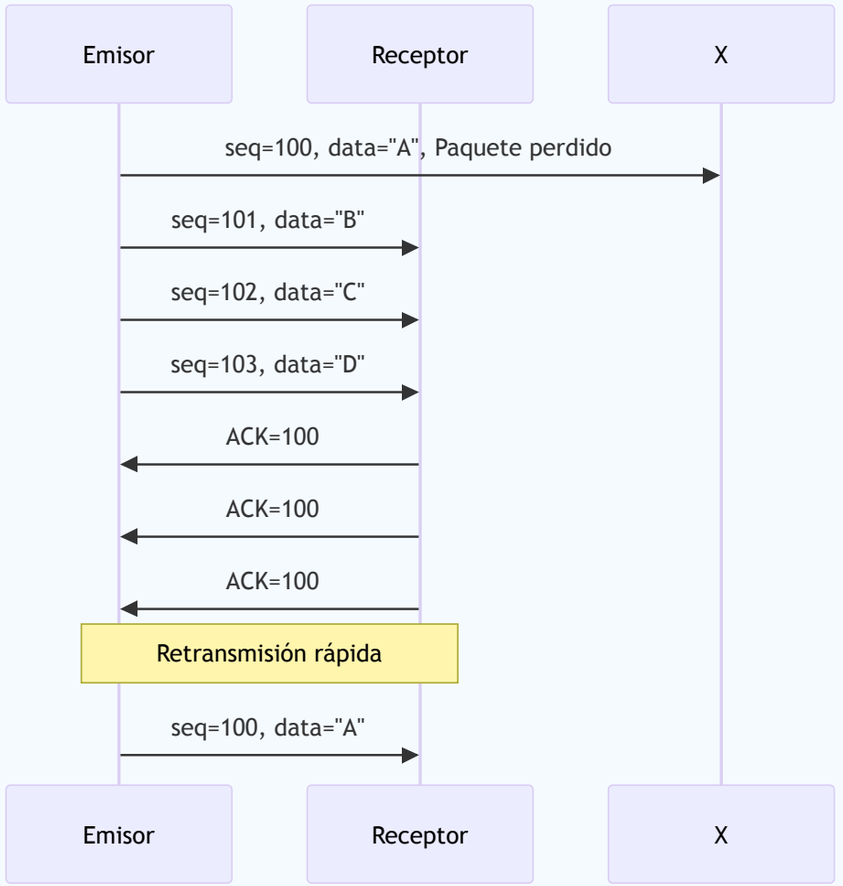
Detección de pérdidas

Pérdidas por timeout



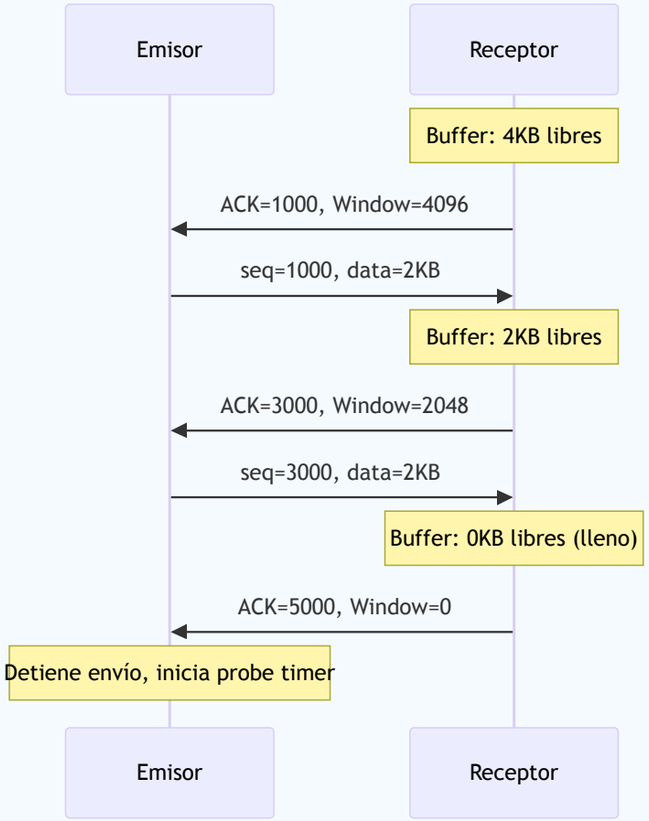
- Si no recibe ACK en tiempo determinado → asume pérdida y retransmite

Pérdidas por ACKs duplicados



Control de Flujo

El control de flujo en TCP es un mecanismo que evita que el emisor envíe más datos de los que el receptor puede procesar.



- Receptor informa de su capacidad disponible, y se define:
- `VentanaRecepcion = BufferRecepcion – (UltimoByteRecibido – UltimoByteLeido)`
- El emisor se limita a esta ventana
- Si la capacidad es 0, se espera un tiempo y se vuelve a probar.

Control de Congestión

El control de congestión en TCP es un mecanismo que ajusta automáticamente la velocidad de envío (ventana de congestión) para evitar saturar la red cuando detecta pérdida de paquetes o retardos.

Ventana de congestión

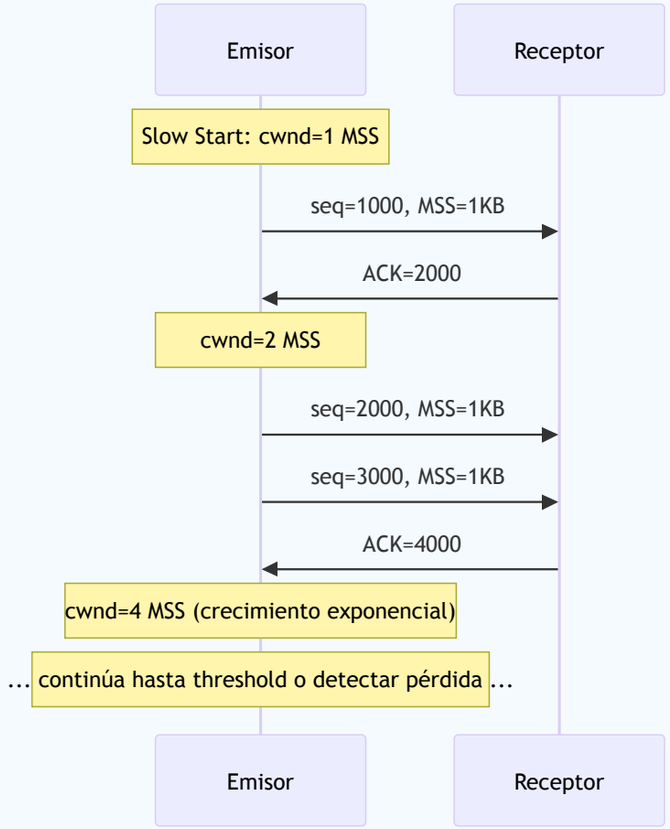
- Variable del emisor
- Bytes máximos en “el aire” (enviados sin ACK)
- Tasa efectiva = min(VentanaCongestion, VentanaRecepcion)

Eventos de Congestión (Pérdidas)

- **Timeout -> Modo slow start**
  - Pérdida severa
  - Ventana → 1 MSS
- **3 ACKs duplicados -> Modo congestion avoidance**
  - Pérdida moderada
  - Ventana → mitad

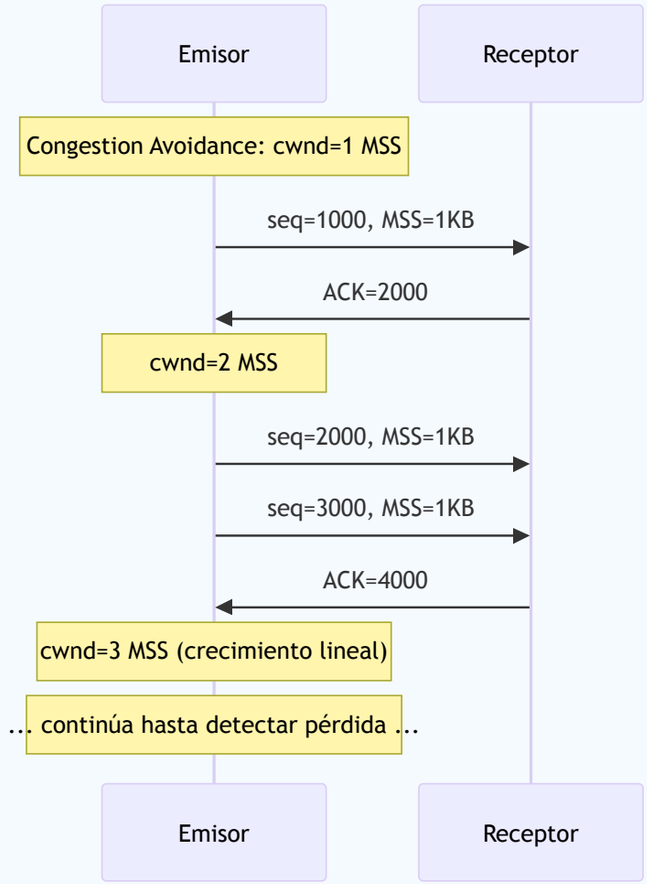
Mecanismos de control de congestión

Slow start



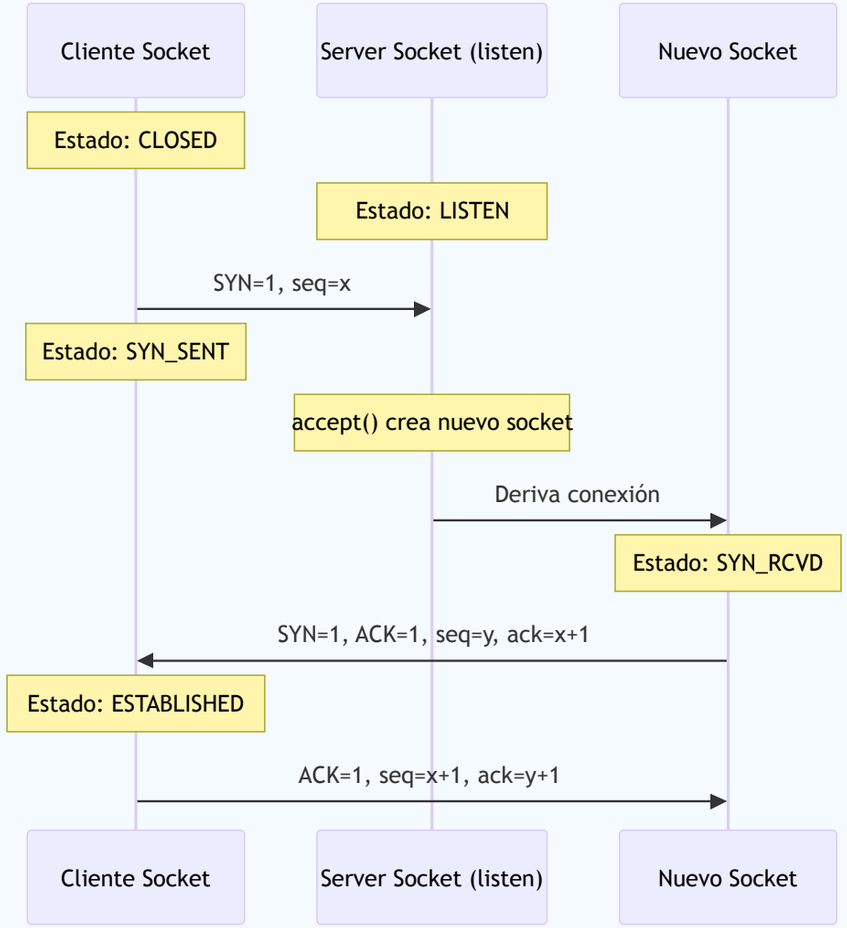
Duplicar ventana por cada RTT

Congestion Avoidance



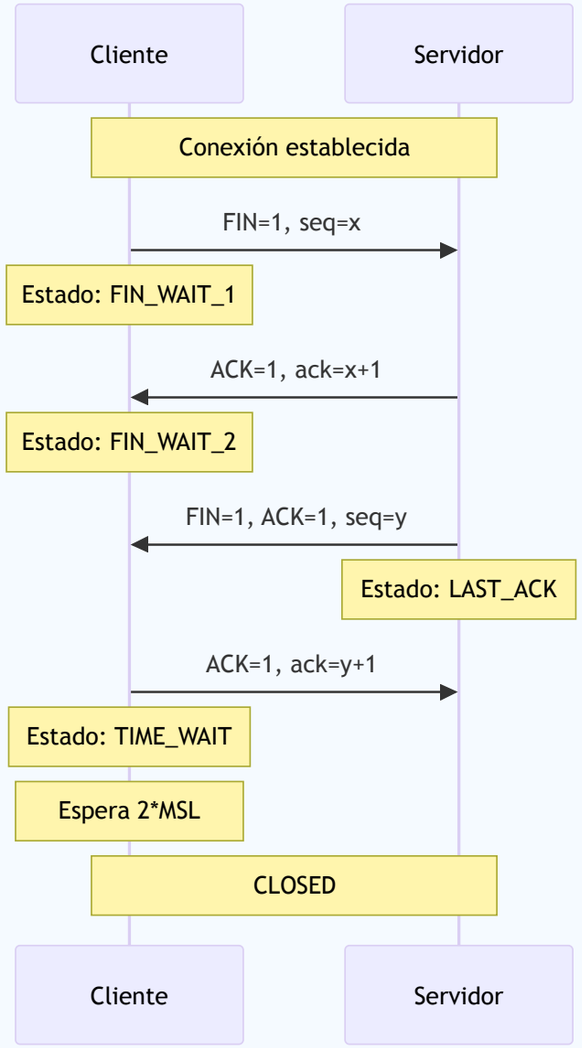
Incrementar en 1 la ventana en cada RTT

Handshake de Tres Fases



- Se negocian: MSS, opciones de ventana, extensiones TCP
- Los flags de las cabeceras también consumen bits.
- El serverSocket está en el servidor,
- una vez se establece la comunicación ambos sockets son iguales.

Terminación de Conexión



- Cada extremo debe enviar su propio FIN y recibir confirmación,
- permitiendo cierre unidireccional (half-close).
- El cliente espera 2*MSL para asegurar que su último ACK llegó,
- manejando retransmisiones tardías antes del cierre definitivo.

Equidad y Coexistencia

TCP es “fair”:

- N conexiones TCP comparten enlace equitativamente
- Cada una obtiene ~R/N del ancho de banda R
- Ver ejemplos en jergames (Bandwidth distribution)

Limitaciones:

- UDP no implementa control → puede monopolizar
- Aplicaciones con múltiples conexiones TCP
- Conexiones con menor RTT tienen ventaja

Comparativa TCP vs UDP para Videojuegos

Cuándo usar…

UDP

Requisitos para UDP:

- Latencias < 50ms
- Actualizaciones frecuentes
- Información nueva más valiosa que la vieja

Ventajas:

- Cabeceras pequeñas
- Sin tráfico de control
- Servidor necesita menos recursos
- No mantiene estado

Ejemplos: Shooters (Counter Strike), juegos de lucha

TCP

Requisitos para TCP:

- Tolerancia 100-200ms latencia
- Entrega ordenada garantizada
- Detección y corrección de errores

Consideraciones:

- Bloqueo cabeza de línea
- Latencias variables por retransmisiones
- Mayor tráfico de red

Ejemplos: MMORPGs (World of Warcraft), juegos

Ejemplos Concretos

World of Warcraft (TCP):

- Hechizos necesitan entrega garantizada
- Actualizaciones de inventario críticas
- Estado de misiones consistente
- MMORPGs toleran 100-200ms

Counter Strike (UDP):

- Retroalimentación inmediata crítica
- Actualizaciones posición/disparos
- Técnicas de interpolación en cliente
- Mitiga efecto paquetes perdidos

Resumen

Puntos Clave

- La capa de transporte proporciona comunicación lógica entre **procesos**
- **UDP:** Minimalista, best-effort, no orientado a conexión
- **TCP:** Confiable, ordenado, orientado a conexión
- **Multiplexación:** Múltiples sockets por un medio
- **Control de flujo:** Evita saturar al receptor
- **Control de congestión:** Responde a condiciones de red
- **Trade-off fundamental:** Confiabilidad vs velocidad
- Elección protocolo depende de requisitos aplicación