

Capa de Transporte, TCP y UDP

Mg. Gabriel H. Tolosa

tolosoft@unlu.edu.ar

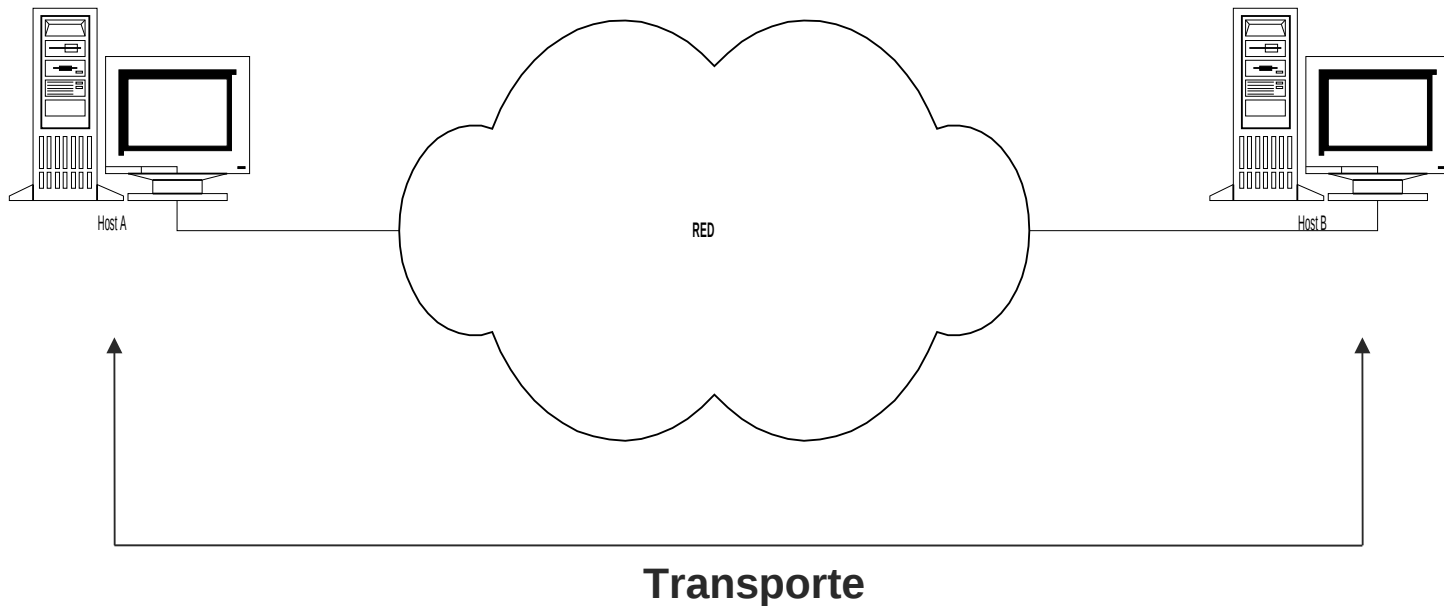
“Be conservative in what you do, be liberal in what you accept from others”

Jon Postel

Principio de Robustez

Capa de Transporte

Ofrece a sus usuarios un sistema transparente de intercambio de datos confiable, operando de extremo a extremo (end-to-end)“



Capa de Transporte

■ Servicios End-to-End

- Garantizar la entrega de los mensajes
- Entregarlos en el mismo orden en fueron enviados
- Eliminar duplicados
- Gestionar un flujo constante de bytes
- Realizar control de flujo
- Multiplexación de conexiones
- Envío de datos urgentes

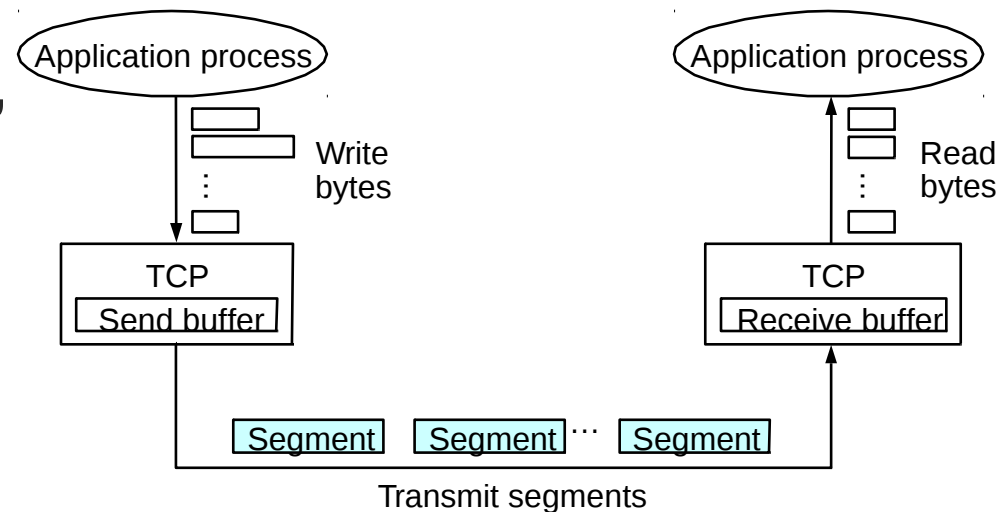
■ Servicios no orientados a la conexión

■ Servicios orientados a la conexión

- Establecimiento/Cierre de conexiones
- Transferencia fiable de T-PDUs

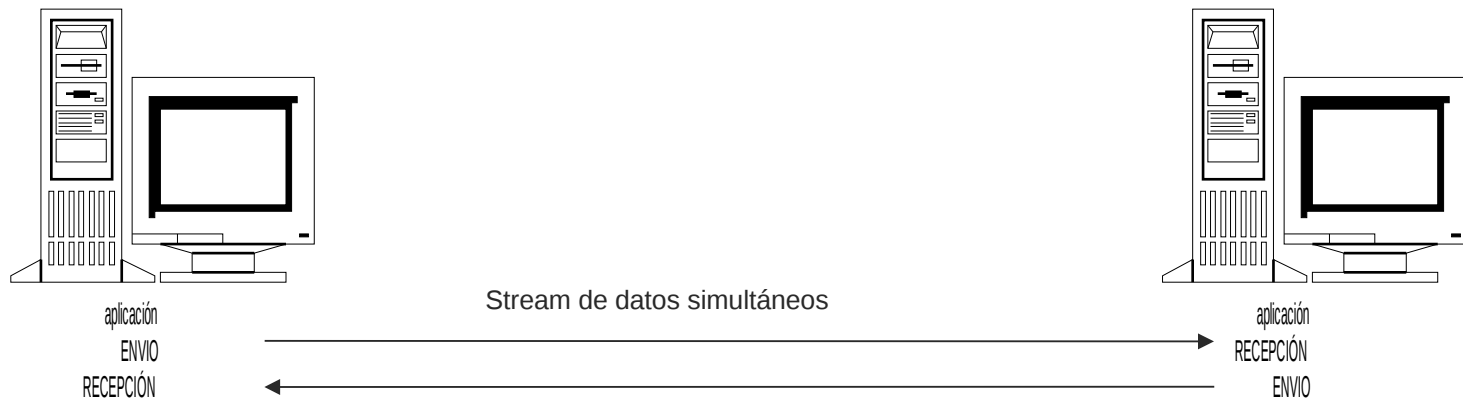
Capa de Transporte – TCP

- **Transmission Control Protocol**
- RFC 793
- Protocolo de transporte de Internet
- Servicio orientado a la conexión
- Transferencia de datos full-duplex
- Transporte confiable:
Control de transmisión,
de flujo, de errores,
de congestión de red

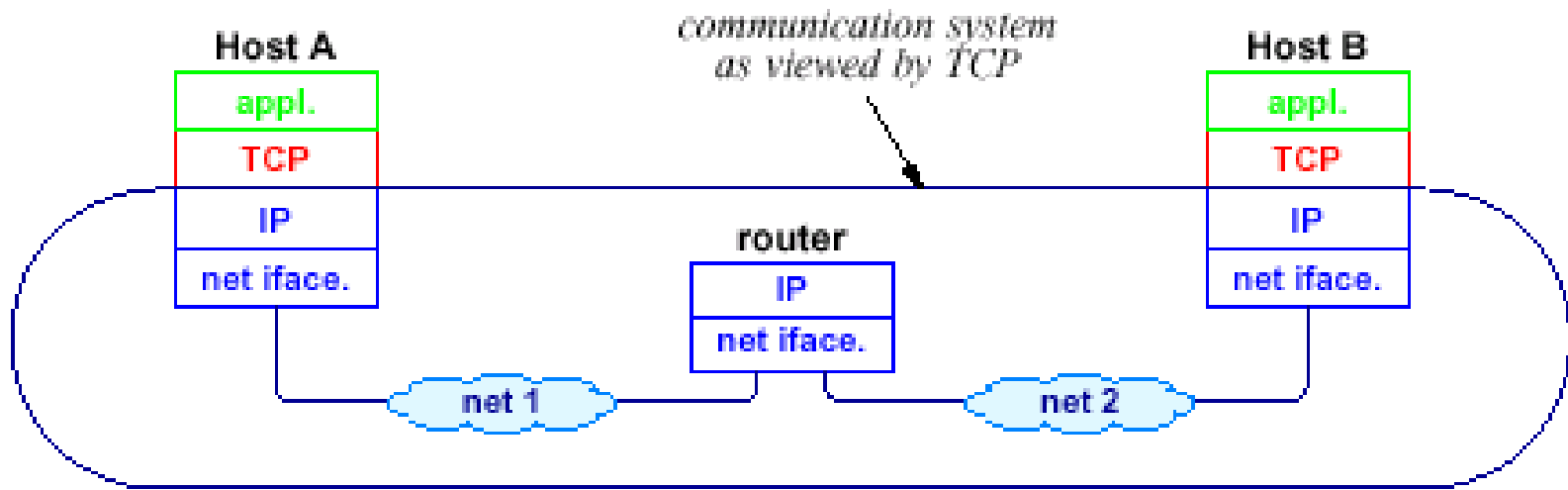


Capa de Transporte – TCP

- Servicio a la capa de aplicación
- Multiplexa/demultiplexa conexiones por aplicación
- Adaptable a LAN/WAN
- Transferencia de un stream de bytes entre sistemas finales (end-to-end)



Capa de Transporte – TCP

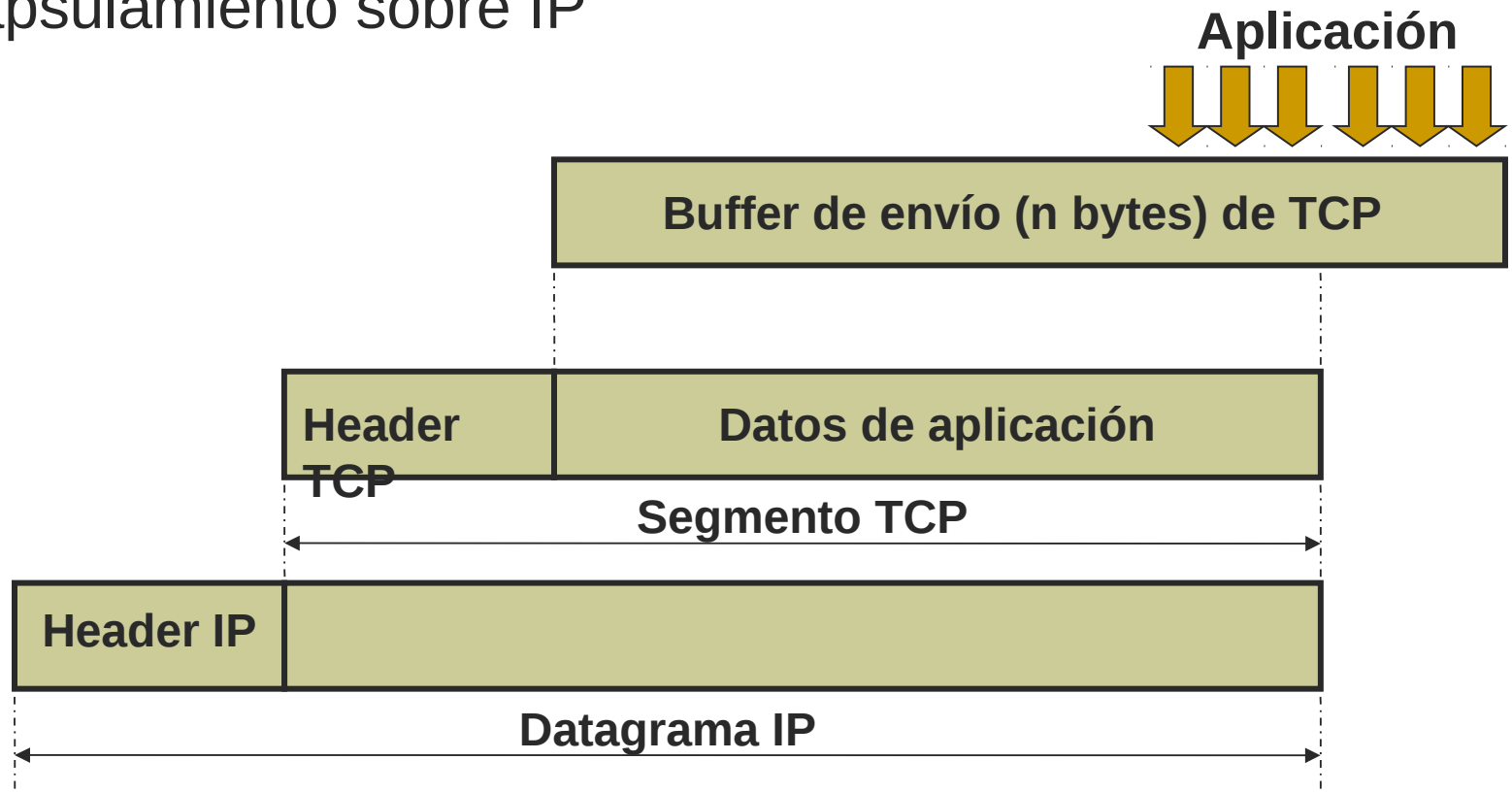


TCP provee un **Servicio de Transporte** completamente **confiable** sobre una red no confiable (o de mejor esfuerzo)

¿Cómo es posible?

Capa de Transporte – TCP

Encapsulamiento sobre IP



Capa de Transporte – TCP

■ Puertos

- Entero de 16 bits definidos a nivel de aplicación
- Usados para poder identificar cada aplicación
- Los procesos “servidores” usan generalmente un “puerto bien conocido” (well known ports)
 - Ej. FTP 20, HTTP 80, TELNET 23
- Los asigna la IANA (Internet Assigned Number Authority)

0	No usado
1-1023	Well Known Ports (servicios standard)
1024-65535	Usuarios (Clientes)
- En Unix se definen en `/etc/services`

Capa de Transporte – TCP

■ Sockets

- Cada host tiene al menos una dirección de red (IP) por interface
- Cada aplicación en un host opera en un puerto
- El par <dirección IP>, <Nro.de puerto> brinda una identificación única para servicios de capa de aplicación en un host
- Se conoce como SOCKET.

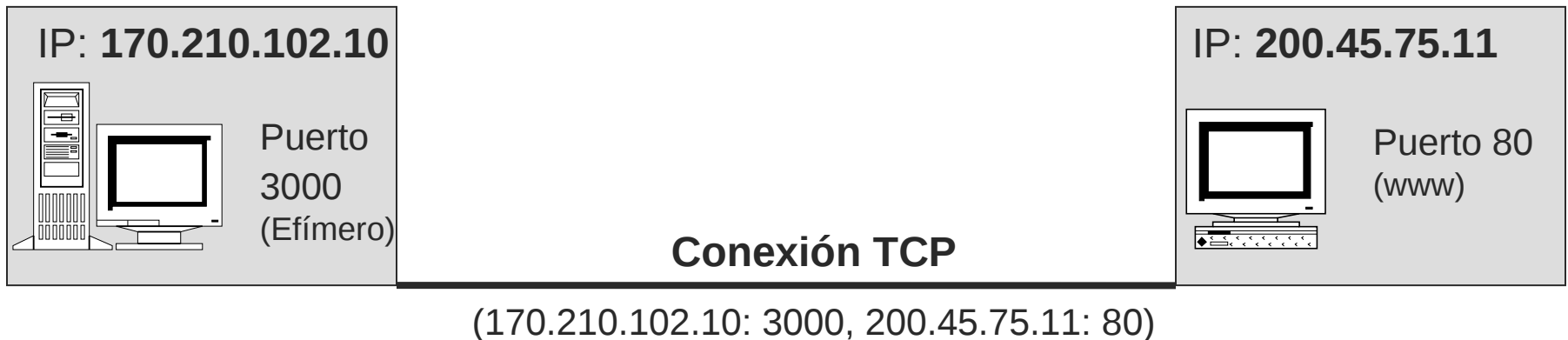


“La dirección IP es única a un nodo (interface), el número de puerto es único en un nodo (x servicio)”

Capa de Transporte – TCP

■ Identificación de Conexiones TCP

- Una conexión TCP está identificada por un par de sockets (del cliente y del servidor) a ambos extremos
- En el servidor se define un puerto donde la aplicación espera la conexión
- En el cliente se define un “puerto efímero” para crear el socket (>1024)



Capa de Transporte – TCP

■ Estado de Conexiones TCP

○ netstat

- Estado de las conexiones y estadísticas por protocolo (-a)
- Información acerca de las interfaces (-i)

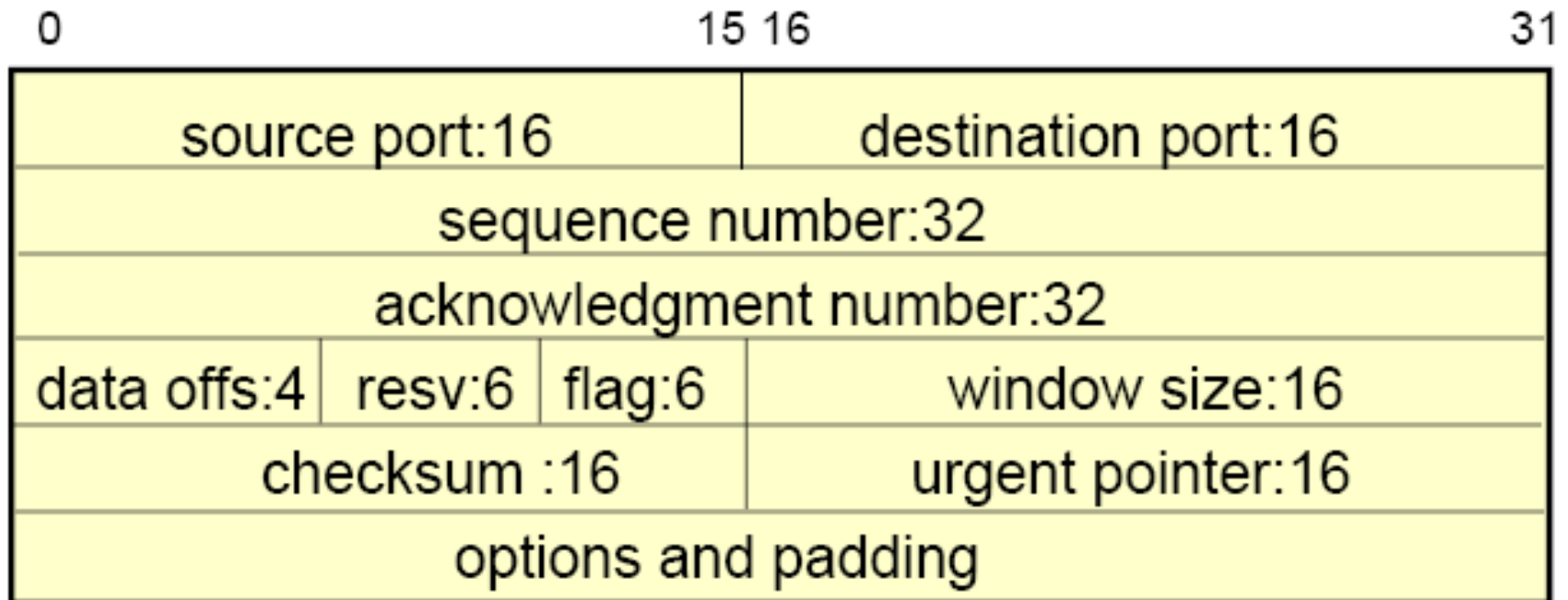
```
#netstat -a
```

```
Conexiones activas
```

Proto	Dirección local	Dirección remota	Estado
TCP	0.0.0.0:21	*:21	LISTENING
TCP	127.0.0.1:1025	127.0.0.1:1026	ESTABLISHED
TCP	127.0.0.1:1026	127.0.0.1:1025	ESTABLISHED
TCP	170.210.102.10:80	0.0.0.0:*	LISTENING
TCP	170.210.102.10:137	0.0.0.0:*	LISTENING
TCP	170.210.102.10:138	0.0.0.0:*	LISTENING
TCP	170.210.102.10:139	0.0.0.0:*	LISTENING

Capa de Transporte – TCP

■ Estructura de Datos



Capa de Transporte – TCP

■ Estructura de Datos

- **Src/Dest Port:** Números de puerto de las aplicaciones origen y destino
- **Sequence #:** Indica el número del primer byte del segmento
- **Ack #:** Indica el número del próximo byte a recibir (válido solo si la bandera está activada)
- **Data Offset:** Indicación para el receptor en qué byte comienzan los datos de usuario
- **Flags:** Información de control
 - **URG:** Indicación si existen datos urgentes
 - **ACK:** El # de ACK es válido
 - **PSH:** Entrega inmediata
 - **RST:** Resetear la conexión
 - **SYN:** Solicitud de apertura de conexión
 - **FIN:** Solicitud de cierre de conexión

Capa de Transporte – TCP

■ Estructura de Datos

- **Window:** Tamaño de buffer de recepción (en bytes).
- **Checksum:** Secuencia de control
- **Urgent pointer:** Puntero a datos urgentes (válido solo si la bandera está activada)
- **Opciones:** Opciones. Por ejemplo, MSS (Tamaño máximo de segmento)

Capa de Transporte – TCP

■ Apertura de Conexión

- El solicitante envía un segmento con la flag SYN activada, dirigido a un puerto en destino, dando un número de secuencia inicial.
- El destino contesta con un segmento con las flag SYN y ACK activadas, y el número de secuencia incrementado en uno.
- El solicitante retorna un segmento con la flag de ACK activada, y el número de secuencia incrementado en uno.

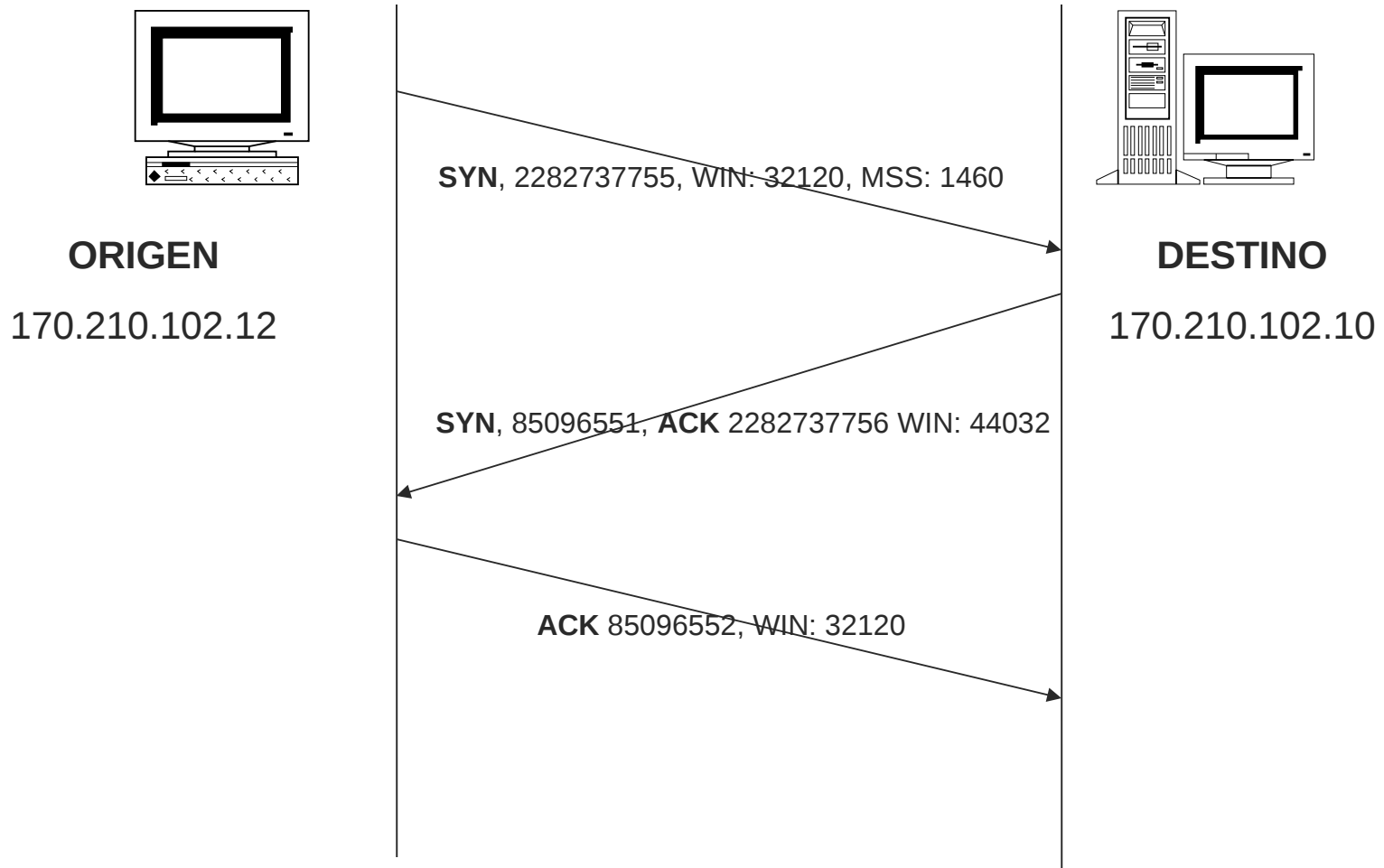
“Three-way handshake”

■ Cuestiones:

- Open Pasivo y Activo
- ISN y MSS
- Open Simultáneo

Capa de Transporte – TCP

■ Apertura de Conexión



Capa de Transporte – TCP

■ Cierre de Conexión

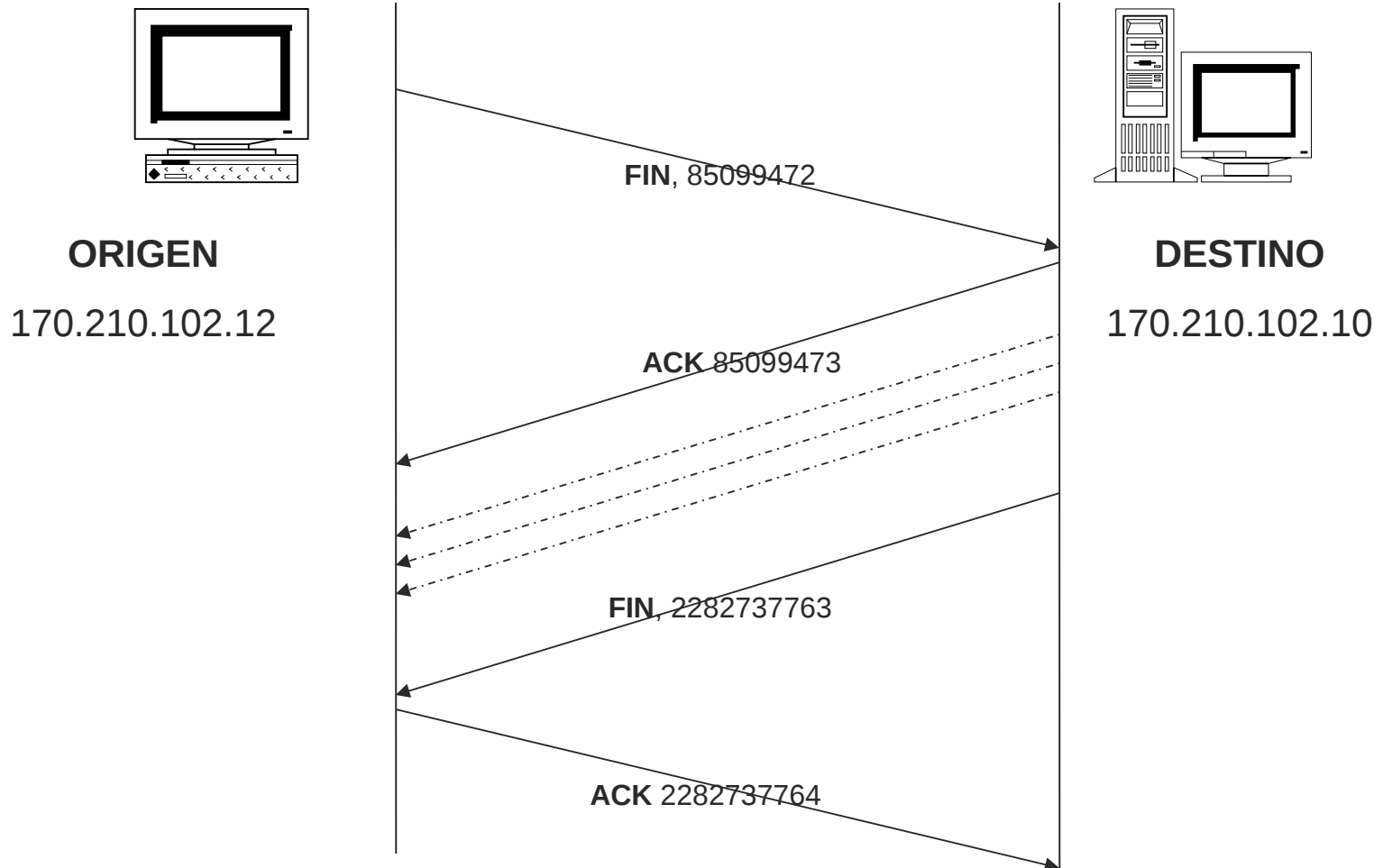
- El solicitante envía un segmento con la flag FIN activada
- El destino responde con un segmento confirmando el cierre (ACK) ⇒ Half-Close
- Se repite la secuencia en el otro sentido

■ Cuestiones:

- Half close
- Estado 2MSL Wait
- Estado FIN_WAIT
- Close simultáneo

Capa de Transporte – TCP

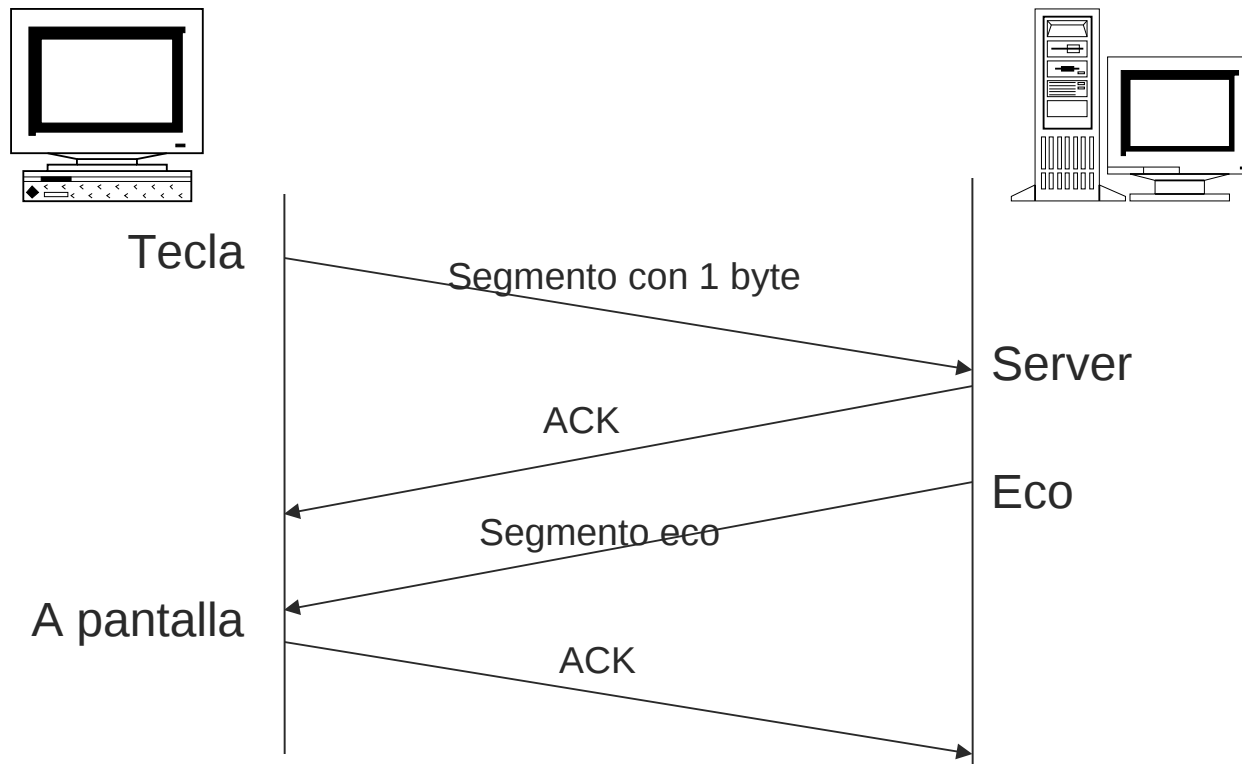
■ Cierre de Conexión



Capa de Transporte – TCP

■ Transferencia

- **Interactiva** (Ej: Telnet/RLogin) -> Delayed Ack, Algoritmo de Nagle



Capa de Transporte – TCP

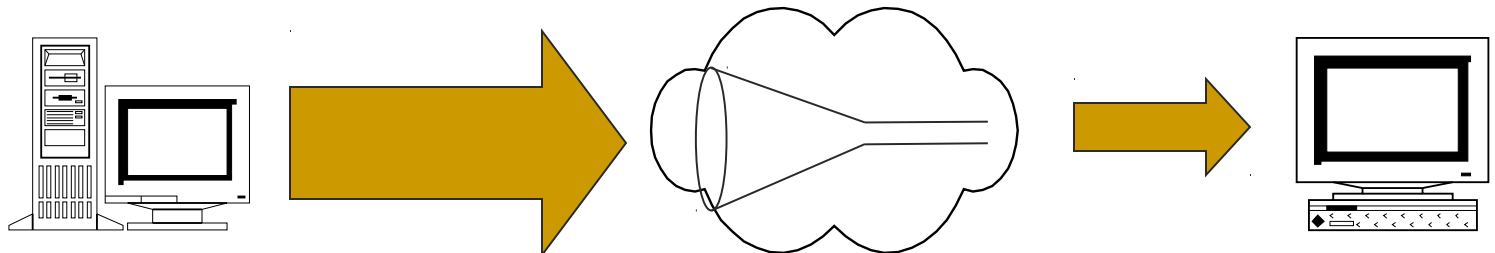
■ Transferencia

○ Masiva (Ej: FTP/Mail)

- Control de flujo (Ventanas deslizantes)



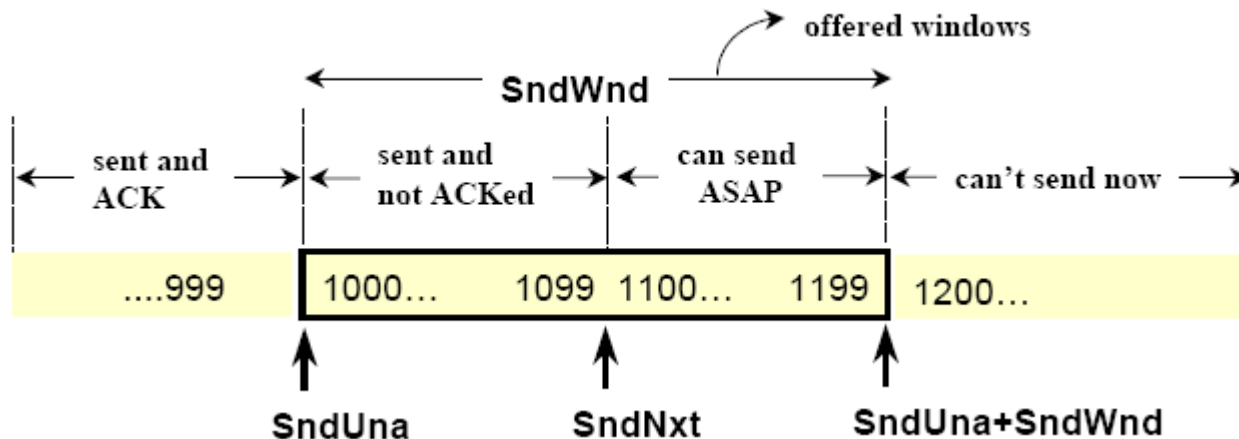
- Control de Congestión



Capa de Transporte – TCP

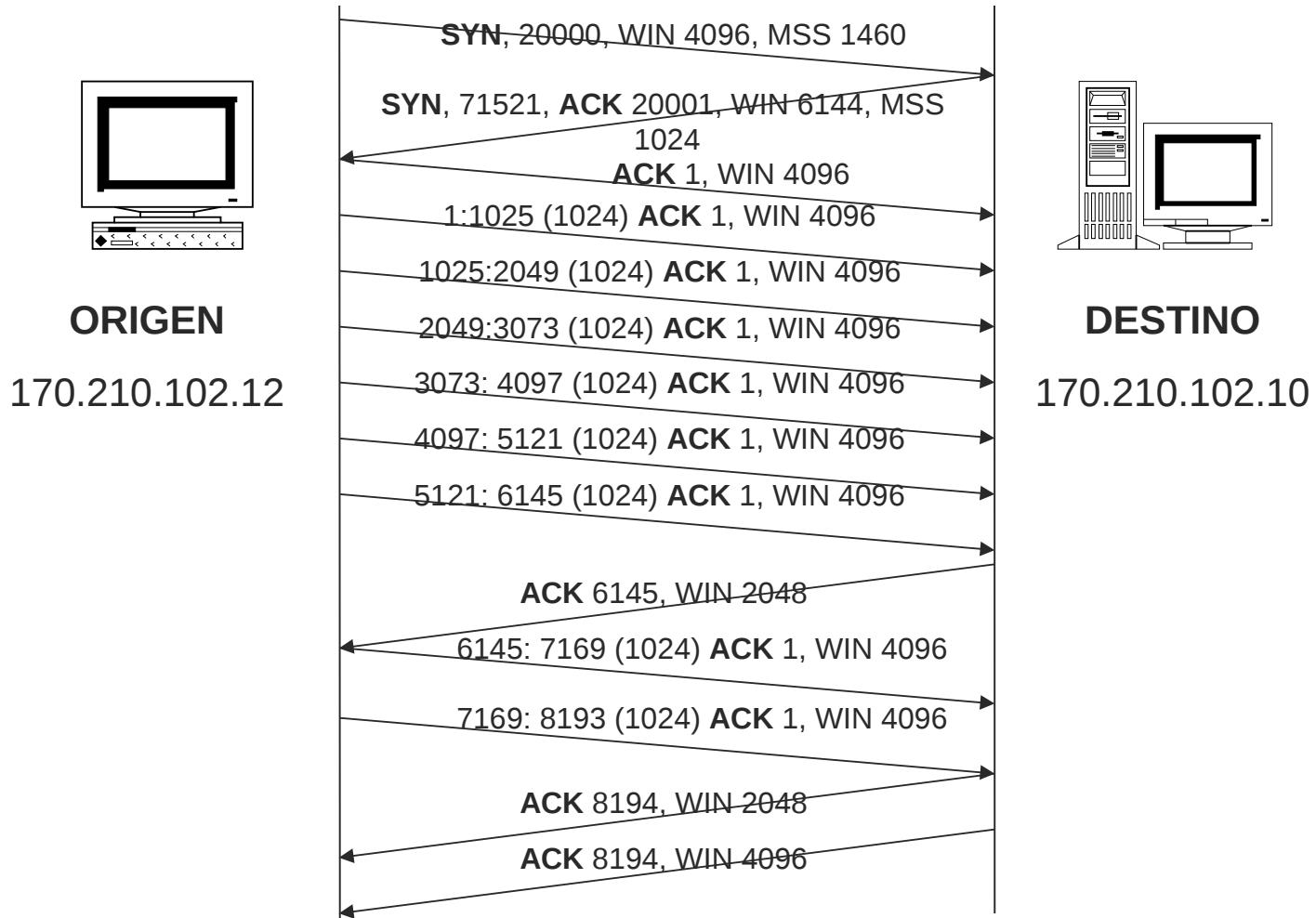
■ Ventanas Deslizantes

- Envío de n segmentos y espera de confirmación (ACK)
- Si no llega en un determinado tiempo, se retransmite
- El control de flujo se realiza determinando el tamaño adecuado de la ventana (bytes)
- En tamaño (buffer) se anuncia en cada segmento
- Emisor solo puede transmitir hasta “llenar” el buffer disponible

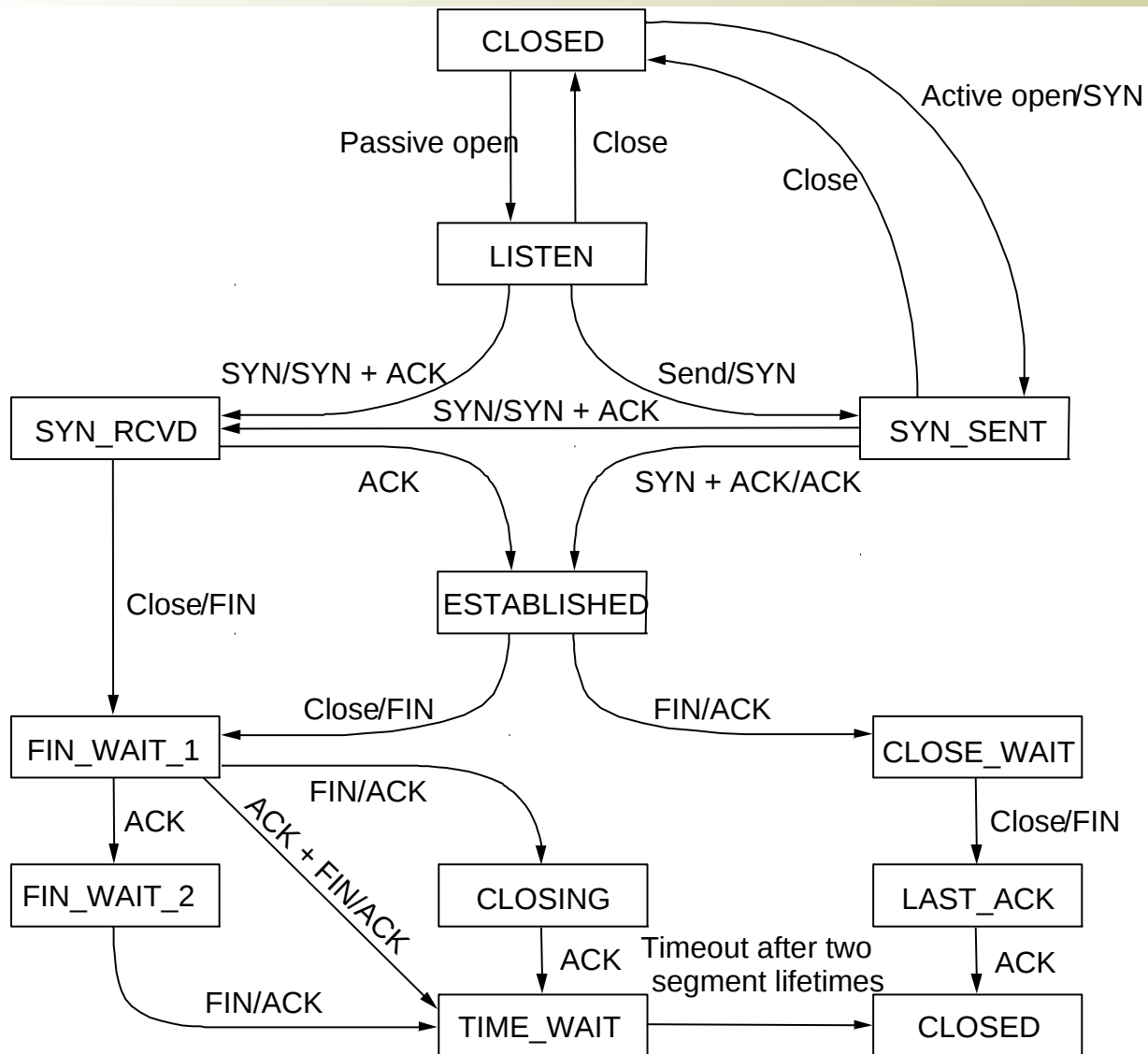


Capa de Transporte – TCP

■ Ventanas Deslizantes



Capa de Transporte – TCP

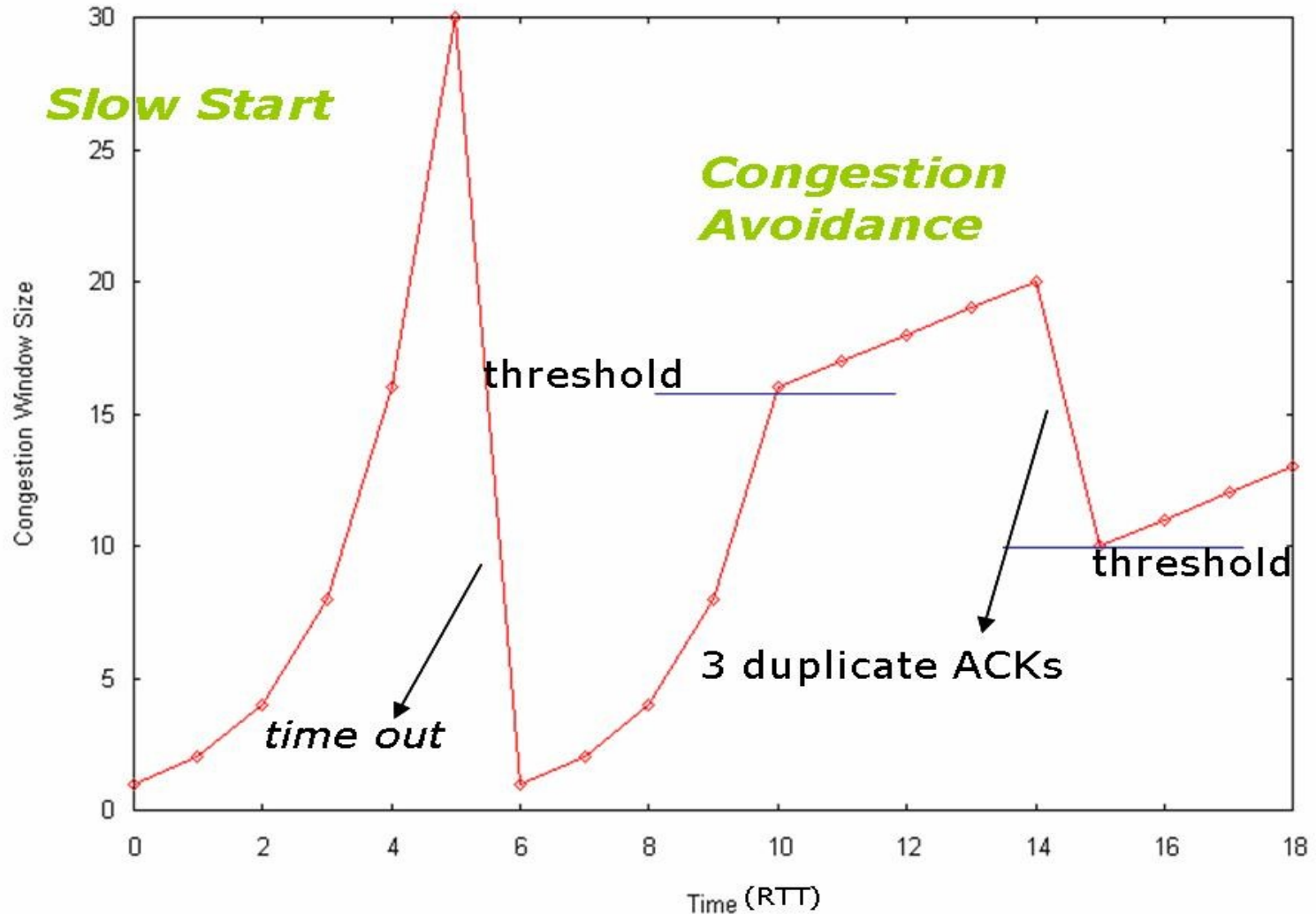


Capa de Transporte – TCP

■ Control de Congestión

- Congestión: La vía de salida tiene menos capacidad para enviar datos. Ej:
 - de LAN a WAN
 - Demasiadas entradas a un router con menos capacidad de salida
- Segmentos perdidos: Reenvío? Puede agravar la situación (La pérdida se asume a congestión dado que la tasa de error es muy baja, $< 1\%$)
- No se puede resolver, se puede evitar (indirectamente)
- Uso de los Algoritmos de Jacobson
 - Slow Start
 - Congestion Avoidance

Capa de Transporte – TCP



Capa de Transporte – TCP

■ Timeout y Retransmisión

■ Retransmission Timer

- Tiempo hasta recibir un ACK

■ Persist Timer

- Mantiene información sobre el tamaño de ventana

■ Keepalive Timer

- Detecta conexiones activas

■ 2MSL Timer

- Mide la duración del TIME_WAIT

Capa de Transporte – TCP

- **Retransmission Timer**
- **No puede tener un valor fijo**
 - Imposible soportar tráfico sobre LAN/WAN
 - Si es muy bajo \Rightarrow problemas de retransmisión
 - Si es muy alto \Rightarrow baja el throughput
- **TCP usa un RT adaptivo**
 - Varía según el RTT (segmento/ACK)
 - Usa el RTO de Jacobson
 - RTO cambia exponencialmente 1,3,6,12,24,48,64 segundos (se duplica por cada retransmisión \Rightarrow exponencial Backoff)
 - Reintenta hasta 9 minutos, luego RESET de la conexión

Capa de Transporte – TCP

■ Persist Timer

- Un extremo anuncia ventana = 0 para detener la transferencia
- Luego envía un segmento con el nuevo valor de ventana, pero éste se pierde
- El otro extremo maneja un timer de aprox. 500 ms, para solicitar una nueva actualización del valor de ventana. Para probar, envía un byte

■ Keep Alive

- Hay una conexión abierta pero no hay tráfico en curso
- Periódicamente se envía un segmento TCP para confirmar el estado de la conexión
- Si no hay confirmación después de varios reintentos la conexión se resetea
- El contenido de un segmento de keepalive no se pasa a la capa de aplicación