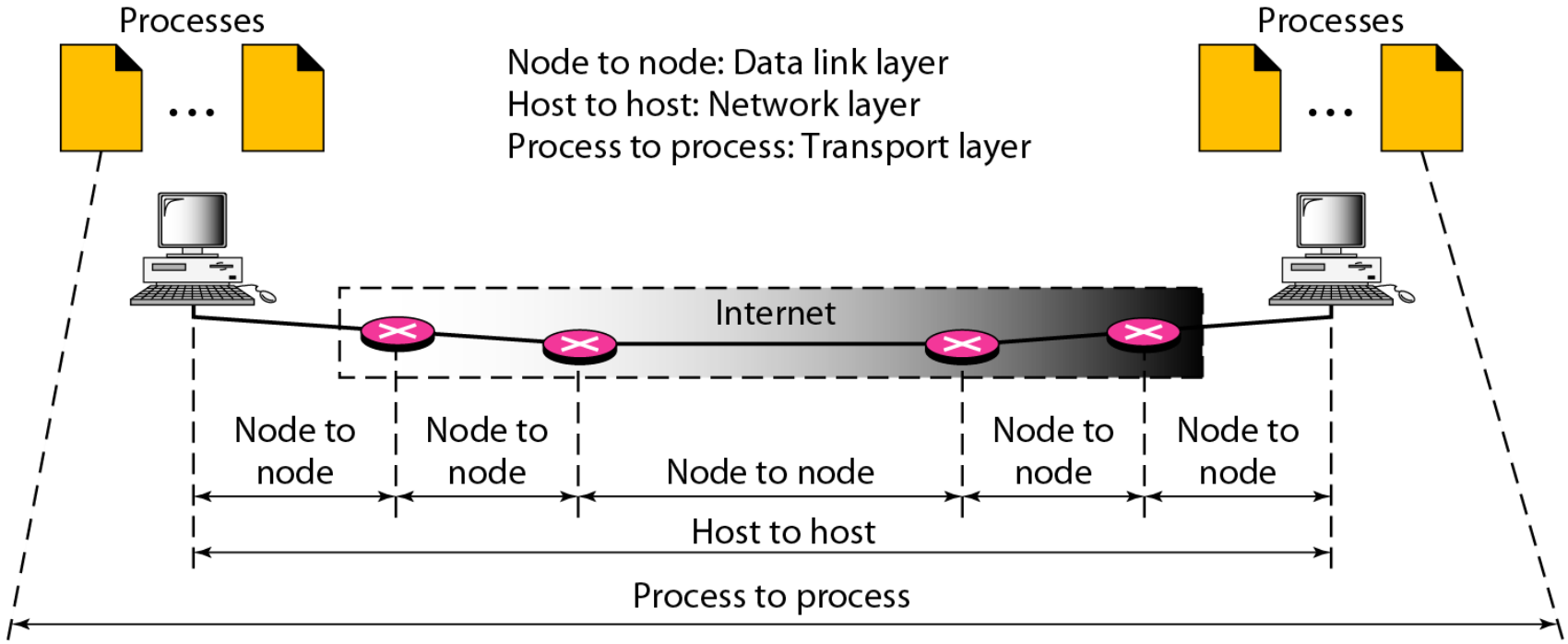
Protocolos de Transporte

Karima Velásquez
karima.velasquez@ciens.ucv.ve

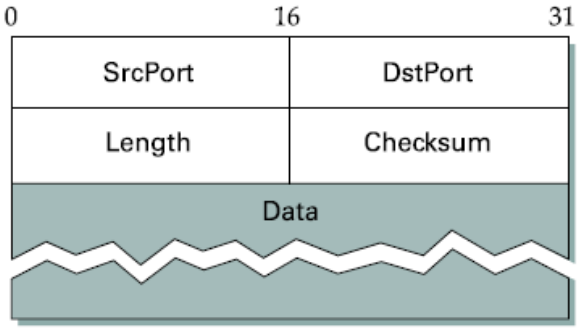
Capa de Transporte

- La capa de transporte es responsable del envío proceso a proceso.



Capa de Transporte: Propiedades

- Garantiza la entrega de mensajes.
- Entrega de mensajes en el mismo orden en que se envían.
- Envía un máximo de una copia de cada mensaje.
- Soporta arbitrariamente mensajes de gran tamaño.
- Admite la sincronización entre el emisor y el receptor.
- Permite al receptor aplicar control de flujo para el remitente.
- Soporta múltiples procesos de aplicación en cada host.



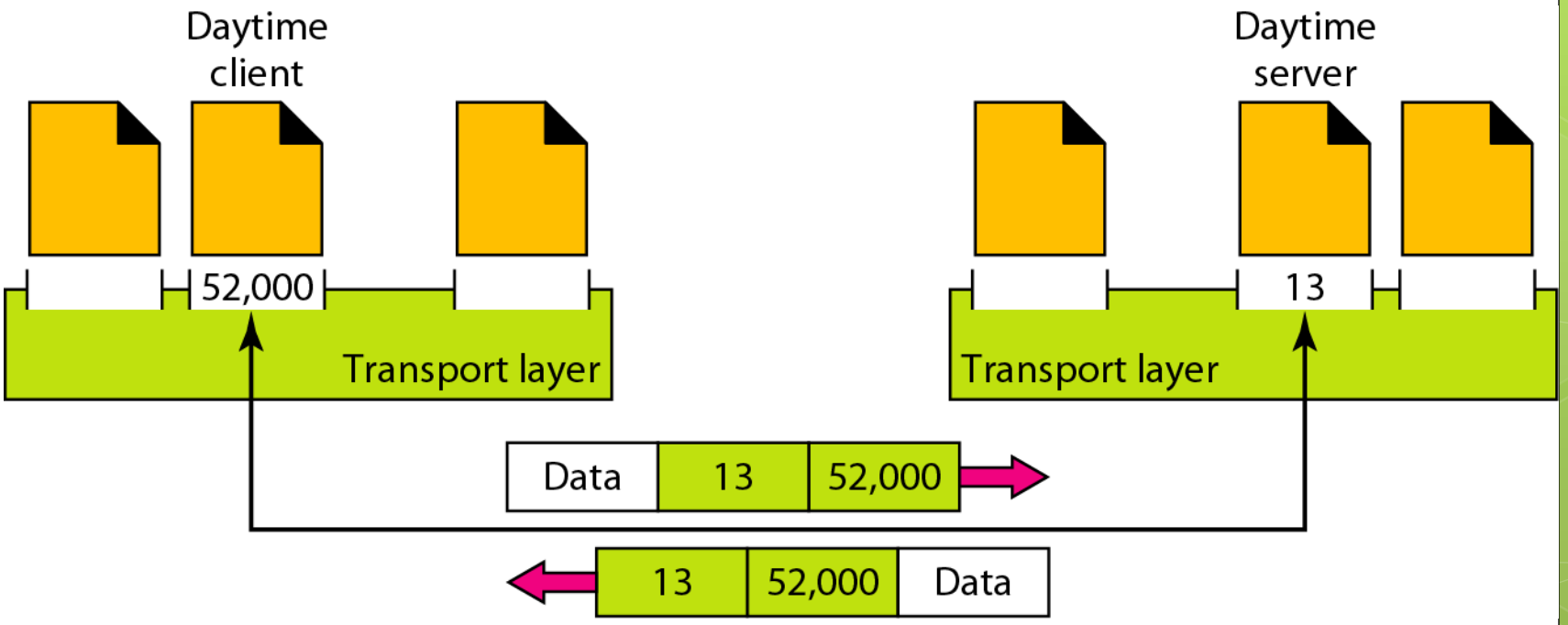
UDP

- Extiende el servicio de entrega host-a-host provisto por la red en un servicio de comunicación proceso a proceso.
- Es un ejemplo de protocolo que proporciona el servicio de transporte mas simple.
- Usa el sistema de direccionamiento indirecto basado en puertos.

UDP

- 64 K puertos pueden ser definidos.
- Alcanza ya que un puerto tiene significado en un simple host no en la Internet completa.
- Como los puntos finales conocen sus respectivos puertos de comunicación:
  - Servidor -> el puerto es anunciado en el mensaje del cliente.
  - Cliente -> el servidor usa puertos bien conocidos para identificar sus aplicaciones.

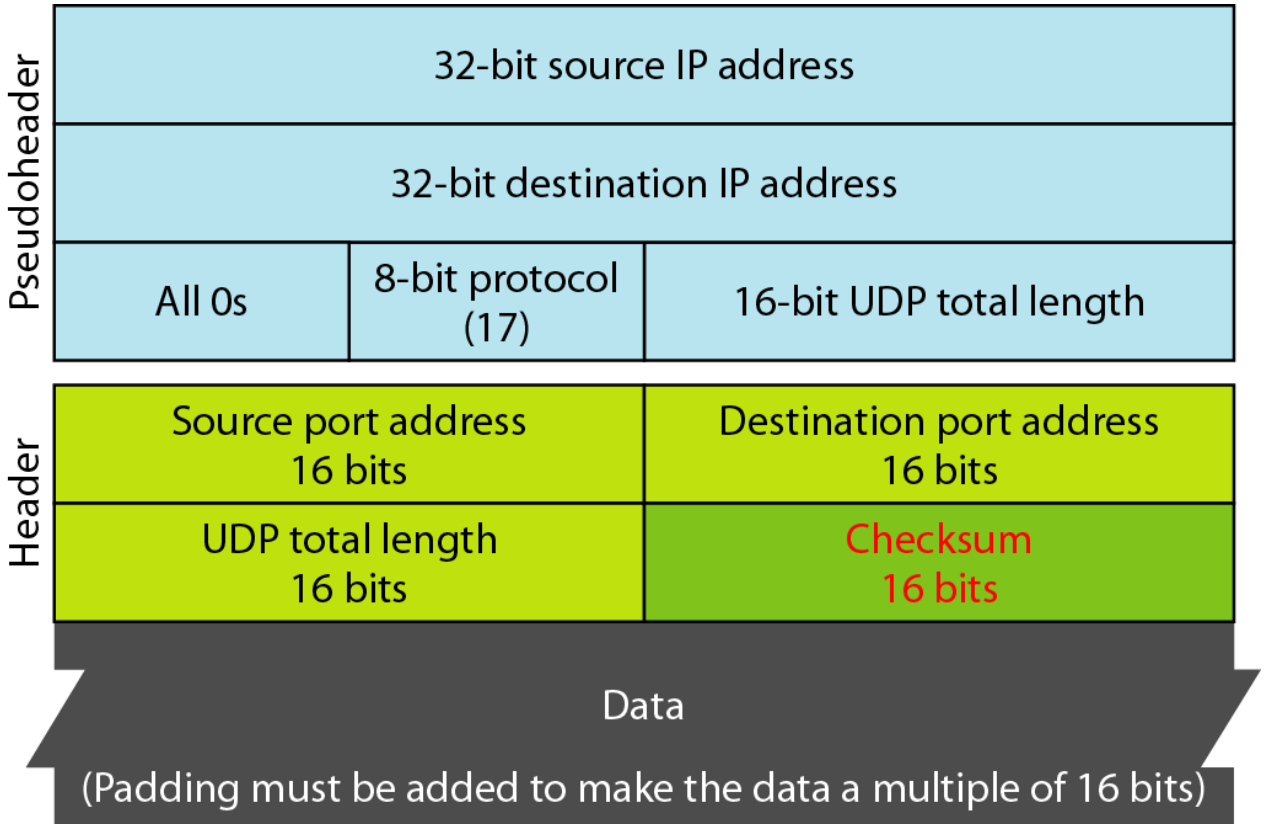
UDP: Números de puertos



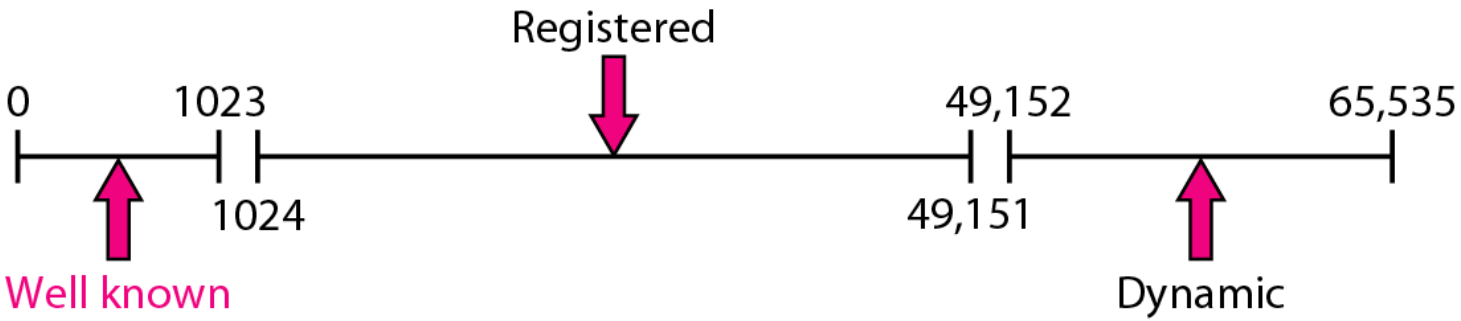
UDP

- Asegura correctitud del mensaje usando el checksum (opcional):
  - Checksum es calculado sobre:
  - Encabezado UDP.
  - Cuerpo mensaje UDP.
  - Pseudo header
    - Número de puerto
    - Dir IP fuente
    - Dir IP destino
    - Campo de long UDP (incluido dos veces)

UDP: Control de Errores: Checksum



Rangos de Puertos de Acuerdo a IANA



Puertos bien conocidos

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

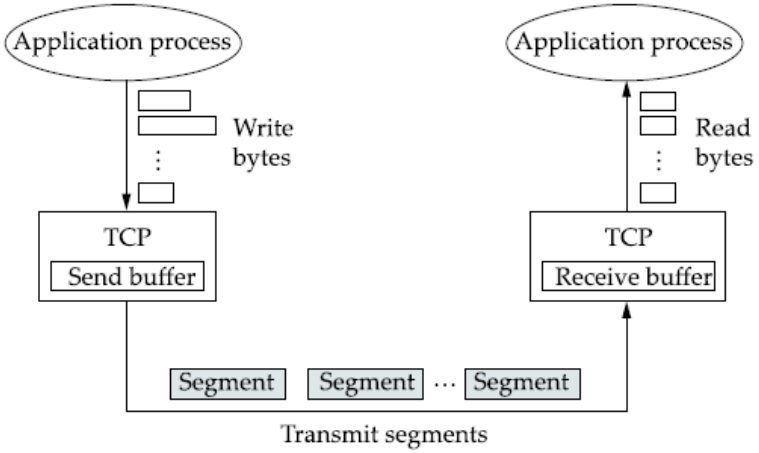
TCP

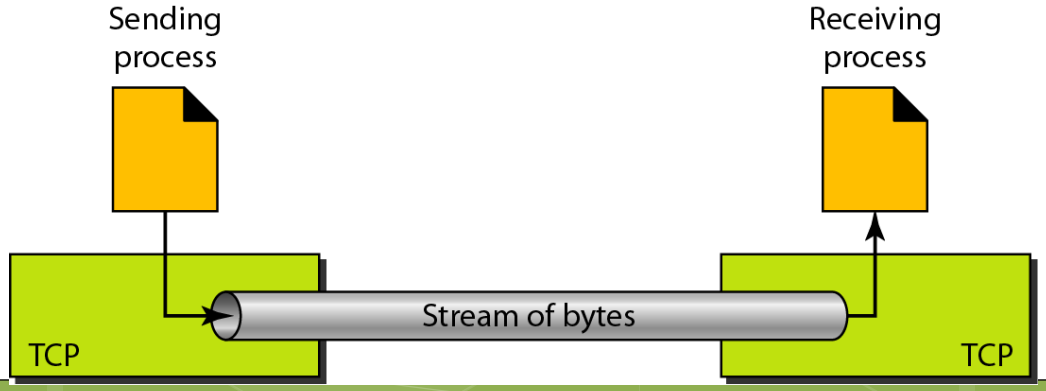
- Es orientado a conexión.
- Siendo las conexiones FDX.
- Servicio basado en stream de bytes.
- Proporciona entrega confiable y en orden de un chorro de bytes.
- Usa mecanismos de control de errores y control de flujo.
- Usa el mecanismo de multiplexaci ón basado en puertos descrito para UDP.
- Proporciona mecanismos de control de congestión.

TCP: Aspectos punto-a-punto

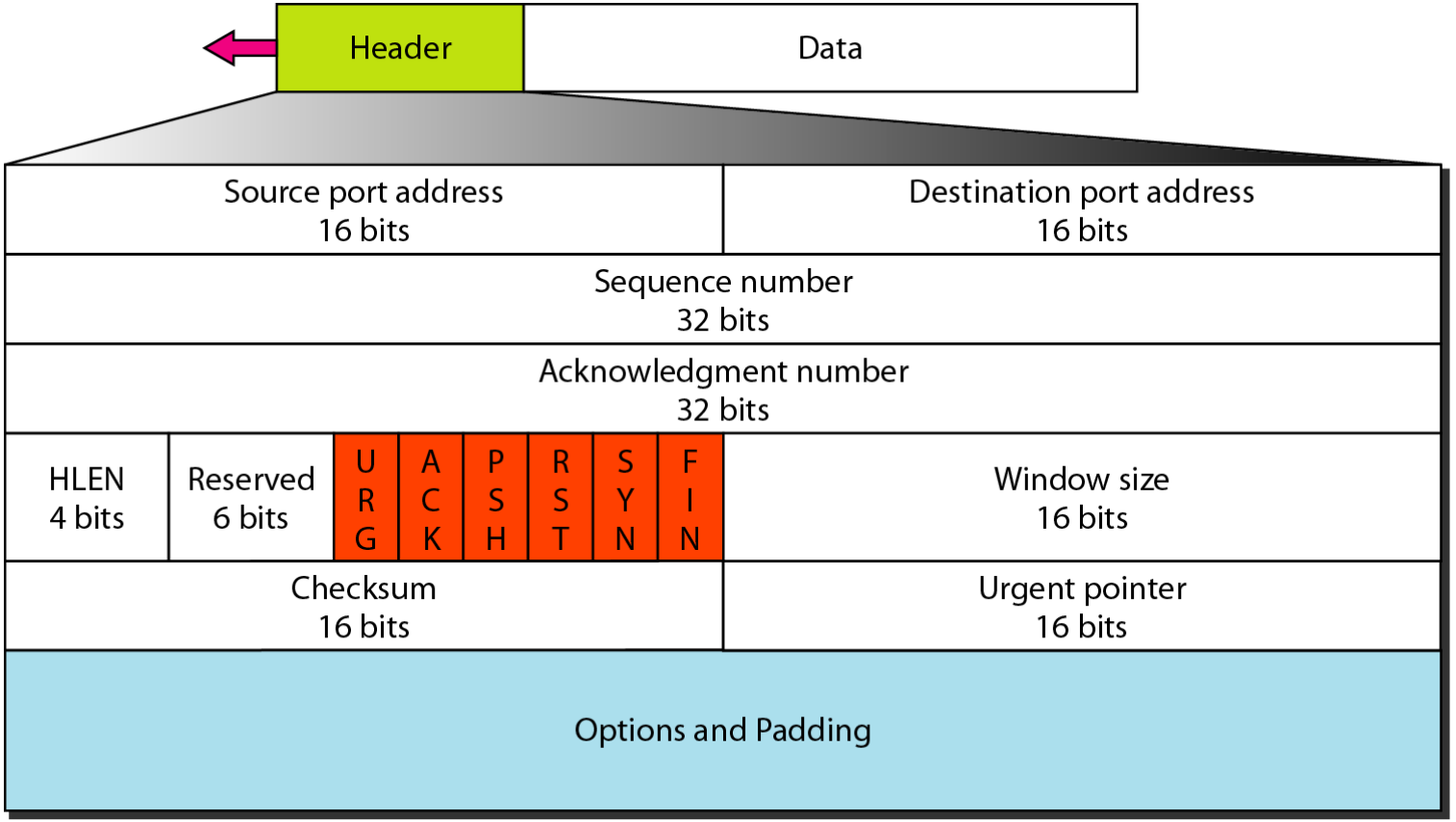
- TCP soporta conexiones lógicas entre procesos conectados en cualquier sitio en Internet.
- Las conexiones TCP pueden tener diferentes RTTs.
- Los paquetes pueden reordenarse en el camino.
- Los recursos para una conexión TCP pueden ser altamente variables.
- El emisor de una conexión no conoce los enlace por los cuales va a pasar los paquetes.

TCP: Orientación a byte





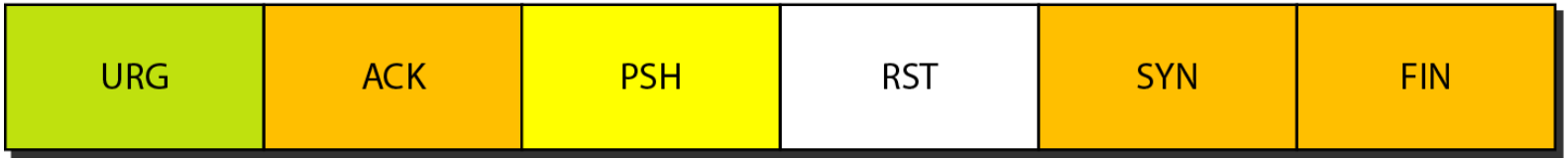
Formato del Segmento TCP



TCP: Campos de Control

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push

RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection



TCP: Banderas en el campo de control

Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

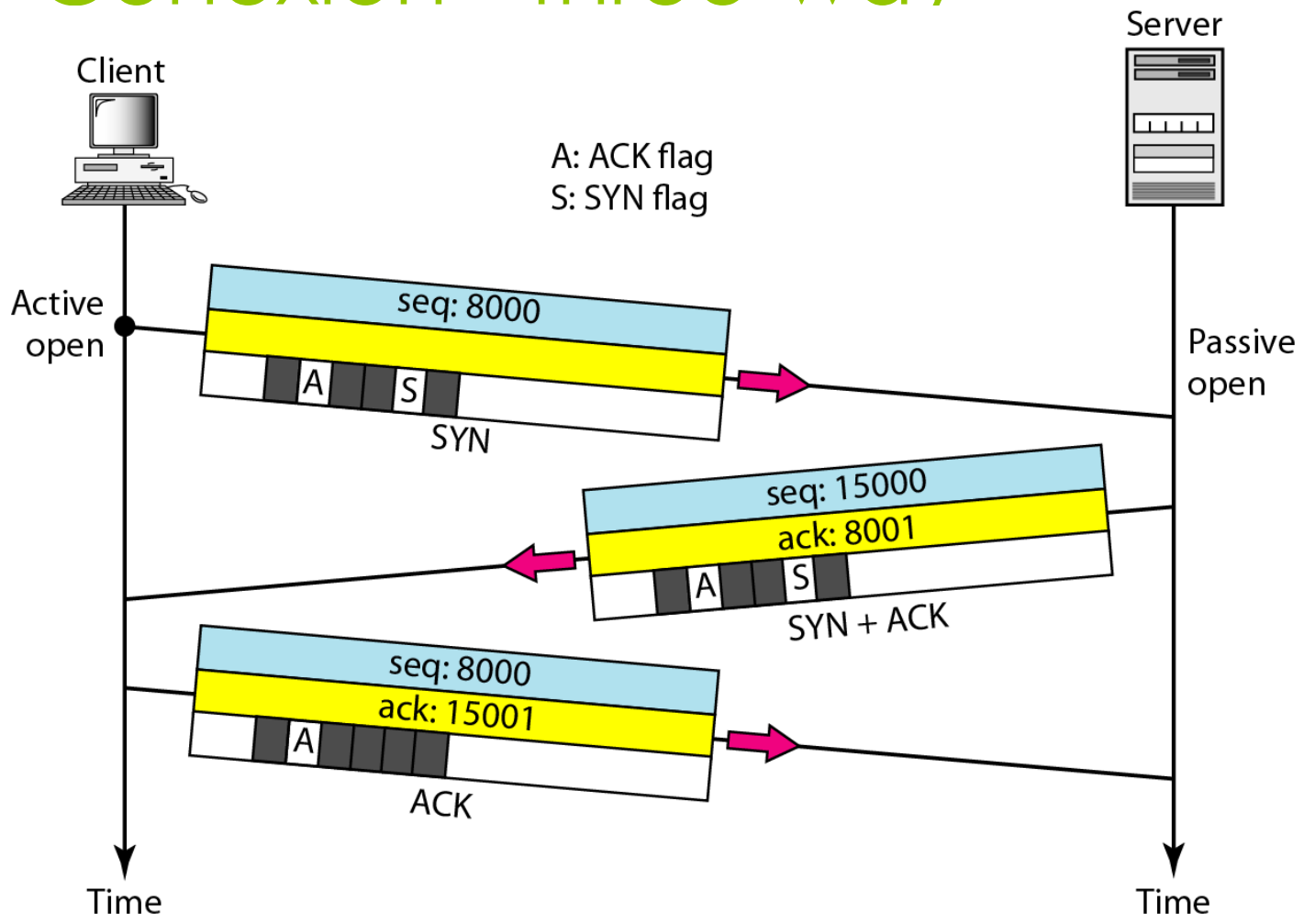
TCP

- Los bytes de data que serán transferidos en cada conexión son numerados por TCP.
- La numeración comienza por un valor generada aleatoriamente.
- El valor en el campo de número de secuencia de un segmento define el número del primer byte de data contenido en el segmento.

TCP

- El valor en el campo de reconocimiento define el número del próximo byte que un parte espera recibir.
- El reconocimiento es acumulativo.

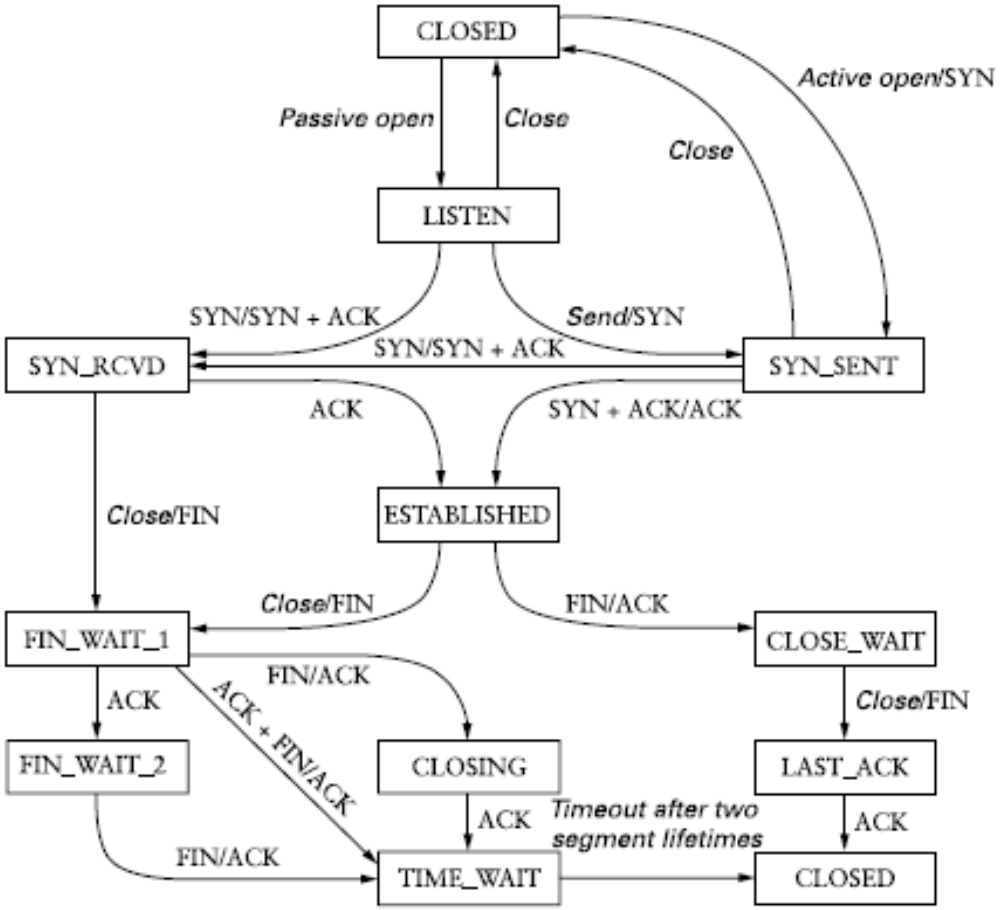
TCP: Establecimiento de la Conexión - three-way



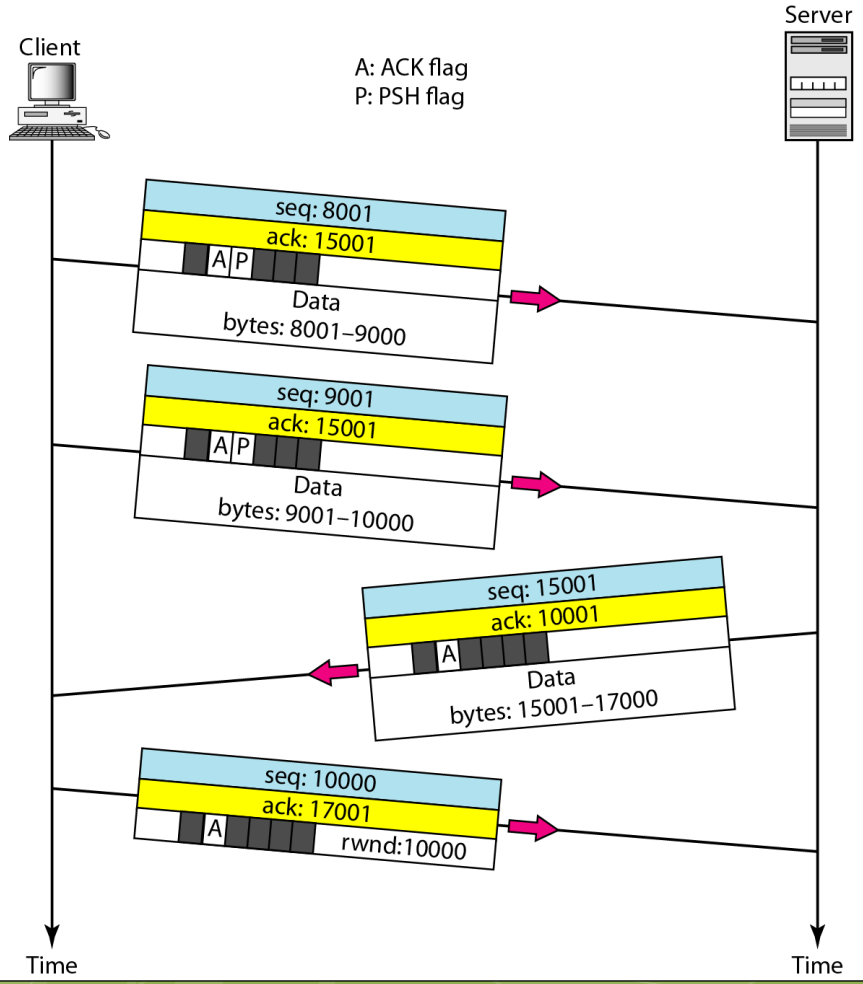
TCP: Establecimiento de la Conexión

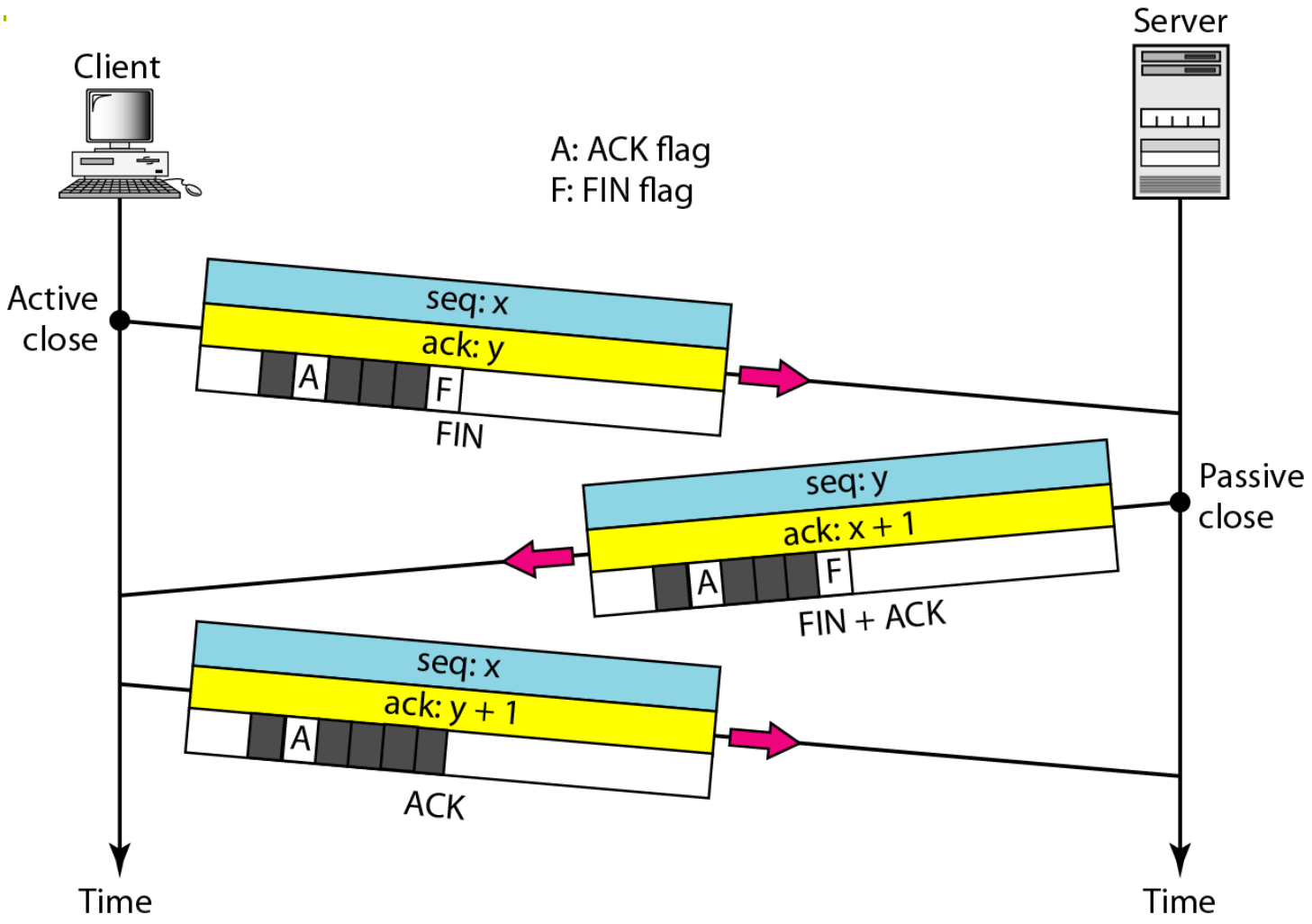
- Un segmento de SYN no transporta data, pero consume un número de secuencia.
- Un segmento de SYN+ACK no transporta data, pero consume un número de secuencia.
- Un segmento de ACK, si no transporta data, no consume número de secuencia.

TCP: Diagrama de estado

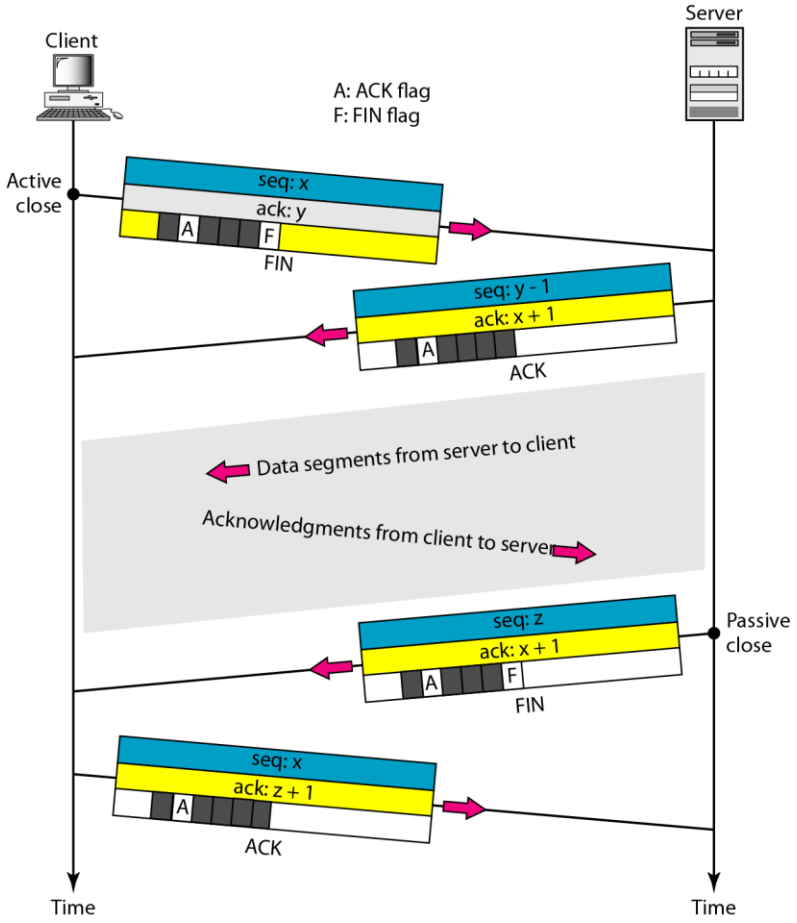


Transferencia de Data




Client
Server
A: ACK flag
F: FIN flag
Active close
seq: x
ack: y
A
F
FIN
seq: y
ack: x + 1
A
F
FIN + ACK
Passive close
seq: x
ack: y + 1
A
ACK
Time
Time

Half-Close

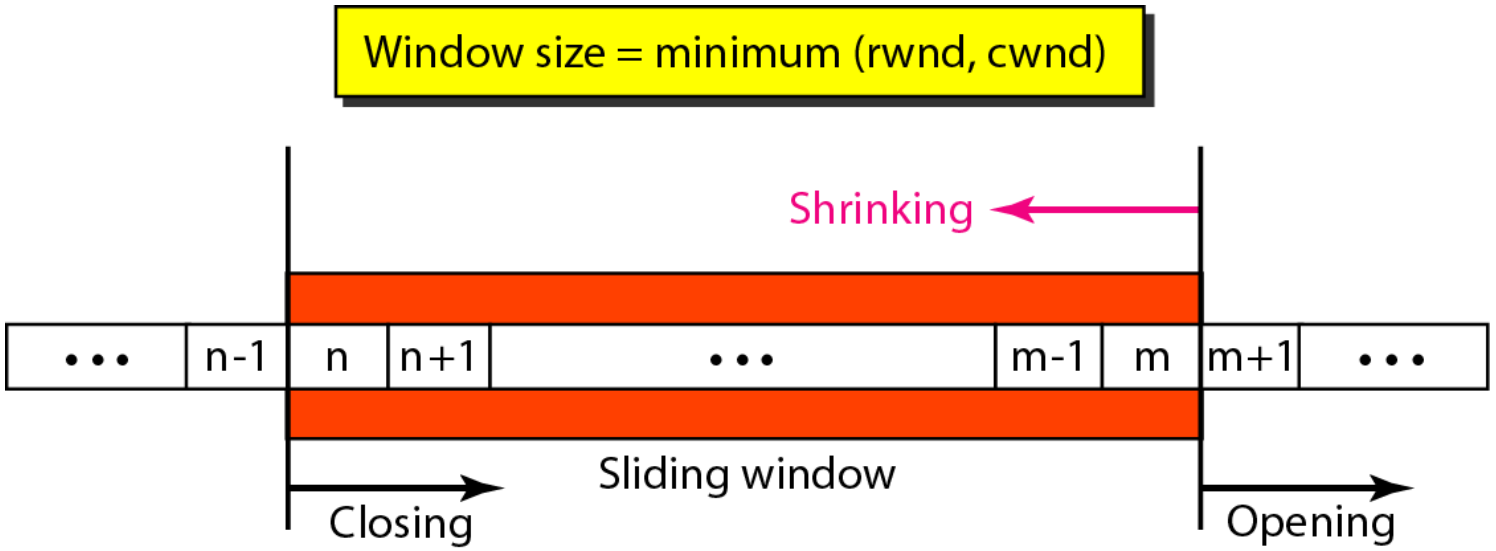

Client
Server
A: ACK flag
F: FIN flag
Active close
seq: x
ack: y
A
F
FIN
seq: y - 1
ack: x + 1
A
ACK
Data segments from server to client
Acknowledgments from client to server
seq: z
ack: x + 1
A
F
FIN
Passive close
seq: x
ack: z + 1
A
ACK
Time
Time

Cierre de la Conexión

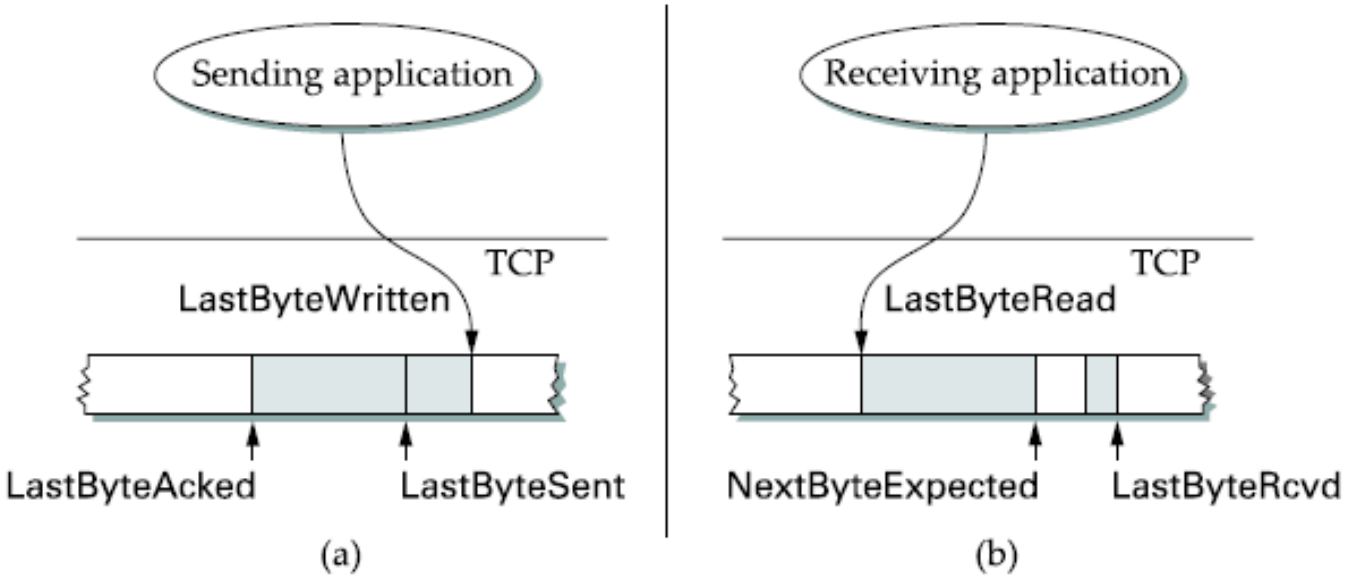
- El segmento FIN consume un número de secuencia si no lleva datos.
- El segmento FIN + ACK consume un número de secuencia si no lleva datos.

TCP: Control de Flujo

- Usa un esquema de ventanas deslizantes.
- Es orientado a byte.



TCP: Control de Flujo

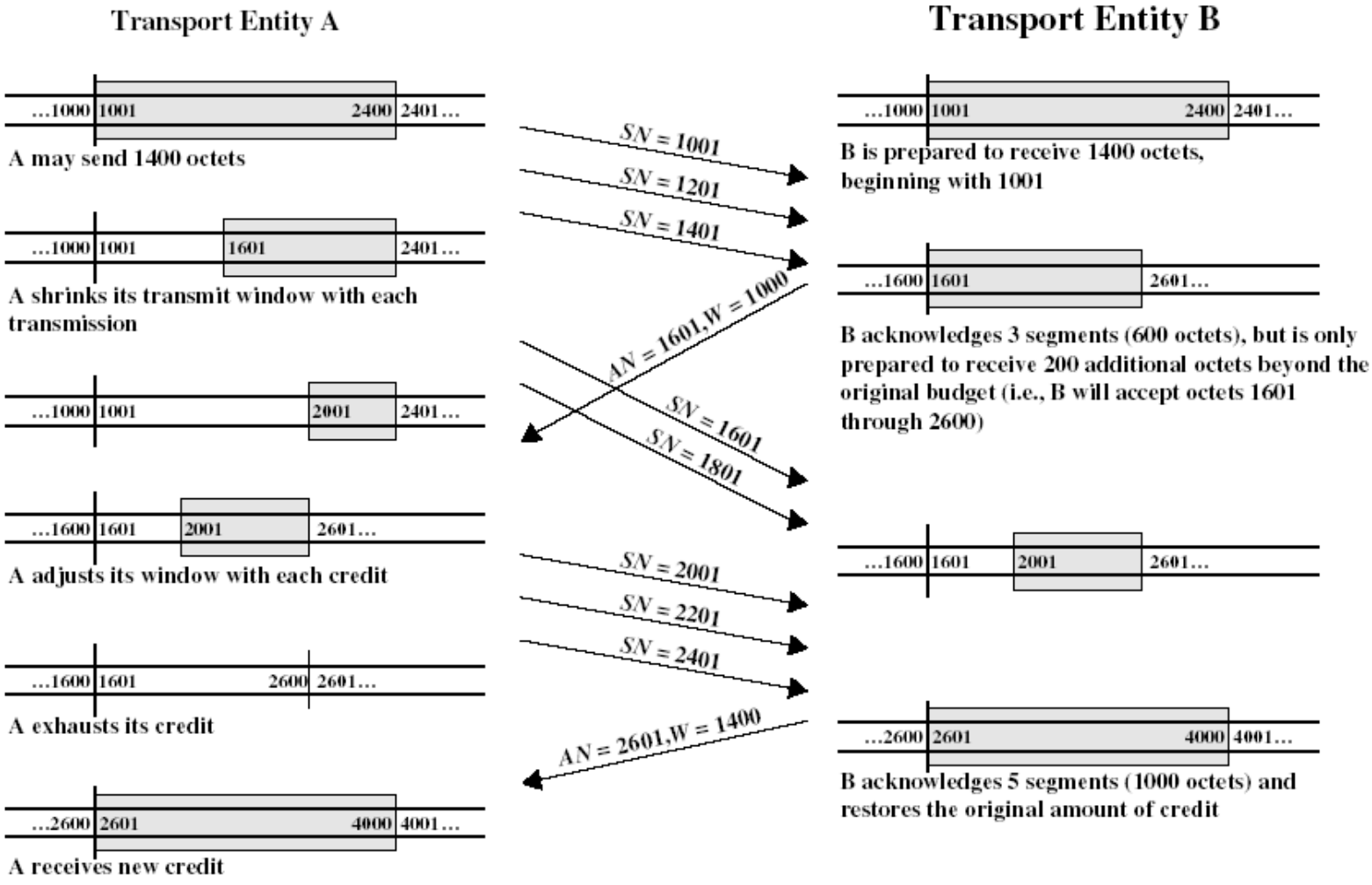


emisor

receptor

TCP: Control de Flujo

- TCP usa un esquema de créditos para proporcionar una transmisión confiable y el control de flujo.
- Este esquema es parecido al esquema de ventana deslizantes usado en la capa de enlace de datos y permite enviar varios segmentos de datos antes de recibir un reconocimiento.
- Este esquema trabaja a nivel de octetos y no de segmentos ni de paquete.
- Descripción de las ventanas deslizantes.


Transport Entity A
Transport Entity B
...1000 1001 2400 2401...
A may send 1400 octets
SN = 1001
SN = 1201
SN = 1401
...1000 1001 1601 2401...
A shrinks its transmit window with each transmission
...1000 1001 2001 2401...
AN = 1601,W = 1000
SN = 1601
SN = 1801
...1600 1601 2001 2601...
A adjusts its window with each credit
SN = 2001
SN = 2201
SN = 2401
...1600 1601 2600 2601...
A exhausts its credit
AN = 2601,W = 1400
...2600 2601 4000 4001...
A receives new credit
...1000 1001 2400 2401...
B is prepared to receive 1400 octets, beginning with 1001
...1600 1601 2601...
B acknowledges 3 segments (600 octets), but is only prepared to receive 200 additional octets beyond the original budget (i.e., B will accept octets 1601 through 2600)
...1600 1601 2001 2601...
...2600 2601 4000 4001...
B acknowledges 5 segments (1000 octets) and restores the original amount of credit

TCP: número de secuencias

- El número de secuencia puede comenzar de nuevo.
- En un futuro 32 bits para esta campo pueden ser pocos.

Bandwidth	Time until Wraparound
T1 (1.5 Mbps)	6.4 hours
Ethernet (10 Mbps)	57 minutes
T3 (45 Mbps)	13 minutes
Fast Ethernet (100 Mbps)	6 minutes
OC-3 (155 Mbps)	4 minutes
OC-12 (622 Mbps)	55 seconds
OC-48 (2.5 Gbps)	14 seconds

Table 5.1 Time until 32-bit sequence number space wraps around.

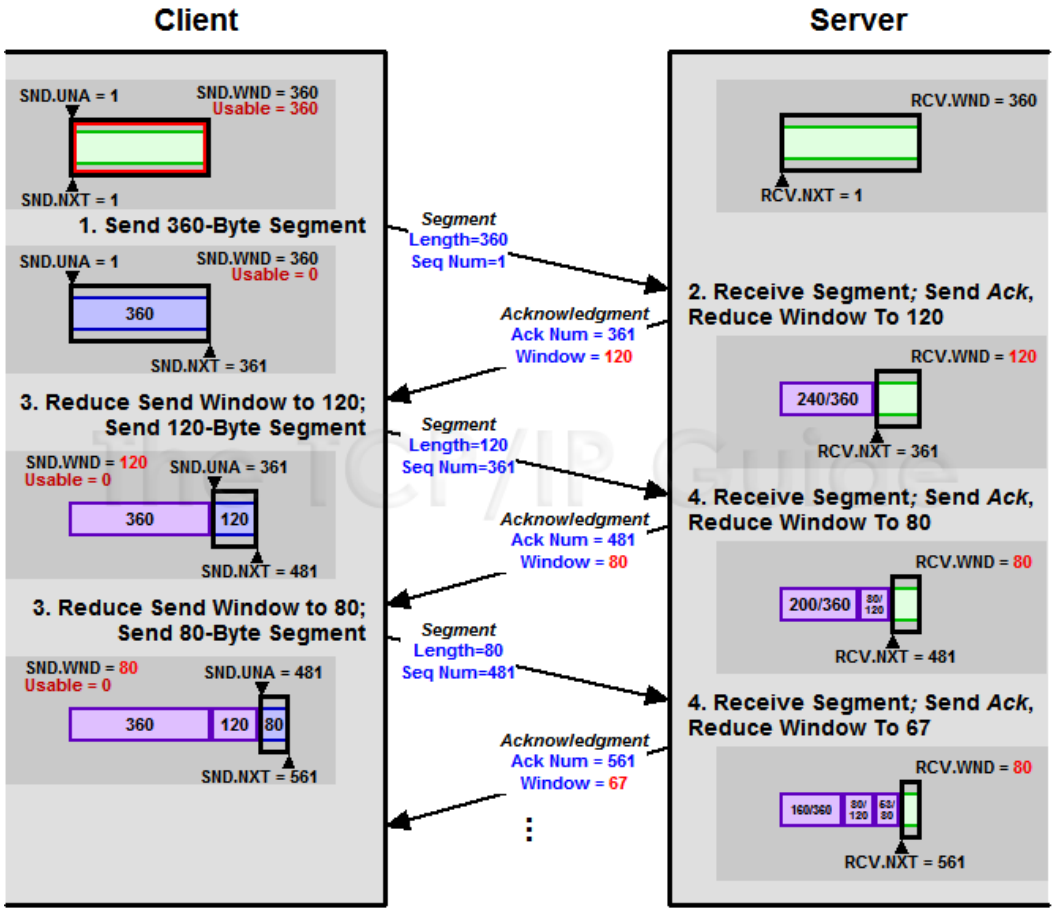
TCP: Envío de segmentos

- El campo de opción del encabezado TCP se puede usar para la negociación del tamaño máximo del segmento (MSS) que el receptor esta dispuesto a recibir.
- El tamaño máximo posible de un segmento es de 65535 octetos.
- Un segmento puede enviarse cuando:
  - Cuando se alcanza el MSS.
  - Como consecuencia de una operación de push.
  - El timer se activa.

Síndrome de Ventana Tonta (Silly Window)

- El receptor va anunciando tamaños de ventanas cada vez mas pequeñas impidiendo que el transmisor envíe volúmenes altos de información eficientemente.

Síndrome de Ventana Tonta (Silly Window)



Silly Window: Soluciones

- Lado receptor
  - Retardar ACKs
- Nagle's Algorithm
  - Emisor
    - Si ventana menos que MSS, esperar por enviar data.
    - Pero cuanto tiempo?

Silly Window: Nagle's Algorithm - self-clocking solution

- When the application produces data to send
  - if both the available data and the window >= MSS
    - send a full segment
  - else
    - if there is unACKed data in flight
      - buffer the new data until an ACK arrives
    - else
      - send all the new data now

TCP: Retransmisiones Adaptativas

- Los segmentos no reconocidos se retransmiten cuando se termina el time out.
- Para manejar retrasos variables que se encuentran en la red, TCP utiliza un Algoritmo Adaptable de Retransmisión.
- Este algoritmo usa el Round Trip Time (RTT).
- El RTT se calcula como:
- (hora que se envía el segmento - hora que se recibe su ACK)

TCP: Retransmisiones Adaptativas

- El nuevo valor del RTT se calcula de la forma siguiente:
- EstimatedRTT =

  $\propto$ * EstimatedRTT + (1-$\propto$) *SampleRTT
  - $0 \leq \propto \geq 1$
- TimeOut = 2×EstimatedRTT

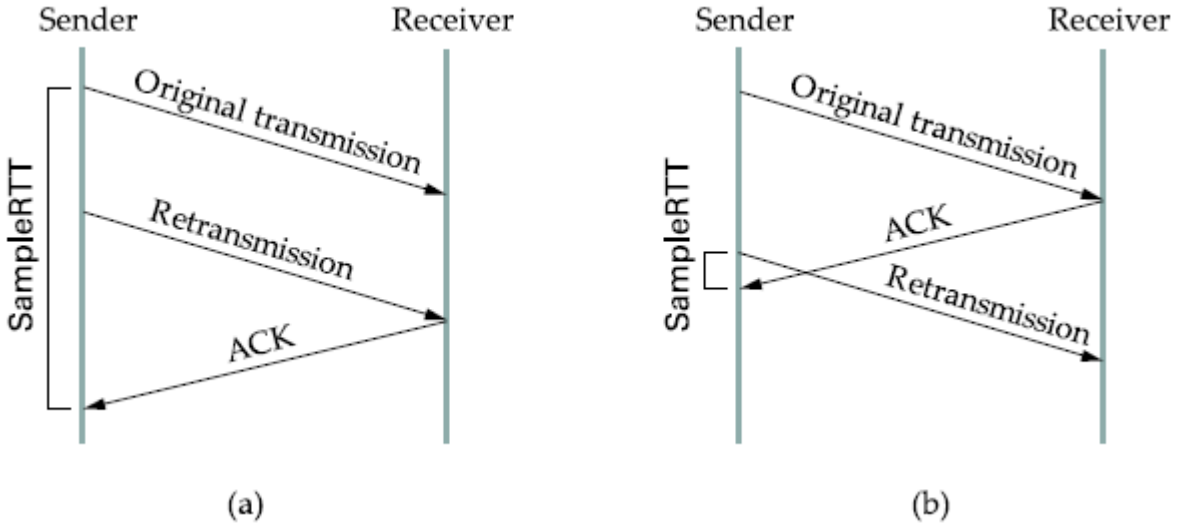
TCP: Algoritmo Karn/Partridge

- Analicemos que pasa con el valor del time out cuando los segmentos son retransmitidos.
- Cuando un el time out expira hay dos posibilidades:
  - El ACK reconoce al primer segmento en cuyo caso el RTO es simplemente mayor de lo esperado.
  - El ACK reconoce al segundo segmento.
  - Ya que TCP no puede distinguir entre estos dos casos, ¿con respecto a qué segmento, TCP debería calcular el nuevo RTO?

TCP: Algoritmo Karn/Partridge

- Si TCP calcula el RTT con respecto a la primera transmisión del segmento y el segundo caso es verdad, el nuevo RTT será muy grande. Peor aún este valor se irá propagando en los próximos cálculos del RTO.
- Si TCP calcula el RTT con respecto a la segunda transmisión del segmento hasta la recepción del ACK, entonces, si el ACK reconoce al primer segmento transmitido, el nuevo valor del RTT será muy corto, produciendo un valor pequeño del RTT.

TCP: Algoritmo Karn/Partridge



TCP: Algoritmo Karn/Partridge

- El algoritmo de Karn propone lo siguiente:
  - No usar el RTT de segmentos retransmitidos para calcular el nuevo RTT.
  - Calcular el backoff RTO cuando una retransmisión ocurre. El backoff RTO se calcula como:
    - RTO = q*RTO
- El valor mas usado de q es 2.

TCP: Algoritmo Karn/Partridge

- Usar el valor del backoff RTO hasta que arribe un reconocimiento para un segmento que no ha sido retransmitido.
- Entonces, calcular el RTO como se explicó inicialmente.

TCP: Algoritmo de Jacobson/Karels

$$\text{Difference} = \text{SampleRTT} - \text{EstimatedRTT}$$

$$\text{EstimatedRTT} = \text{EstimatedRTT} + (\delta \times \text{Difference})$$

$$\text{Deviation} = \text{Deviation} + \delta(|\text{Difference}| - \text{Deviation})$$

δ está entre 0 y 1

$$\text{TimeOut} = \mu \times \text{EstimatedRTT} + \phi \times \text{Deviation}$$

μ usualmente es 1 y ϕ es 4

TCP: Entrega de datos forzada

- Generalmente TCP puede decidir cuando tiene suficiente data para formar un segmento para ser enviado.
- Esto puede resultar ineficiente en el caso de data interactiva de longitud corta.
- El usuario de TCP puede requerirle a TCP que transmita el flujo de datos sin esperar que se almacene en memoria intermedia.
- El segmento también deberá tener el bit de PSH activado, para que se le entreguen los datos al usuario cuando lleguen al receptor.

TCP: Entrega fuera de banda

- TCP proporciona un mecanismo para especificar que cierta información esta viajando sin esperar que el receptor consuma todos los datos pertenecientes a un flujo de datos.
- Por ejemplo esto es útil para interrumpir o abortar una sesión en curso. La llegada de estos datos debe ser notificada al usuario tan pronto como llegue al receptor.
- Esto se logra usando el bit de URG y el puntero de urgencia que se describieron anteriormente.

TCP: Control de Congestión

- Que cada fuente determine cuanta capacidad está disponible en la red para saber cuantos paquetes pueden viajar de forma segura.

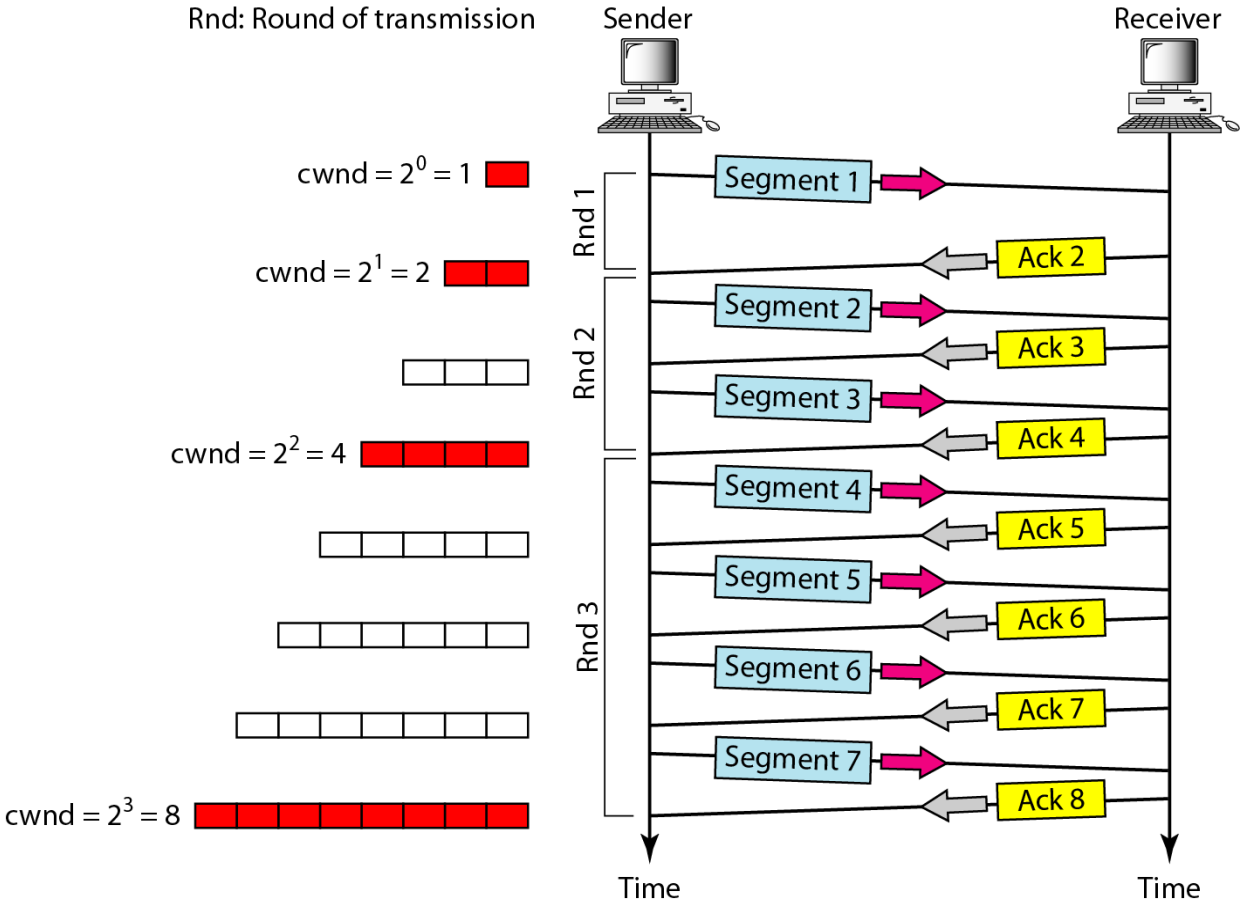
TCP: Arranque Lento/Disminución Multiplicativa

- awnd = min (cwnd, crédito)
- **Tamaño permitido de la ventana (awnd):** el número de segmentos que TCP puede enviar sin recibir reconocimientos.
- **Ventana de congestión (cwnd):** un nuevo límite introducido que se define al momento de inicio de una conexión y usado para reducir el mismo cuando se detecta congestión. Se expresa en segmentos.
- **Crédito (crédito):** la cantidad de créditos garantizados en el último reconocimiento calculado en número de segmentos. Este valor es calculado como Window/tamaño del segmento. Window es el campo que viaja en el TCP entrante.

TCP: Arranque Lento – Aumento Exponencial

- Cuando se establece la conexión, *cwnd* = 1 MSS (ie TCP solo puede enviar un segmento y esperar un reconocimiento).
- Cada vez que un reconocimiento es recibido, *cwnd* = *cwnd* + 1 MSS, hasta el máximo permitido.
- Por ejemplo, después que se envía un segmento y se recibe un reconocimiento *cwnd* = 2 MSS. Entonces el transmisor envía dos segmentos.
- Por cada reconocimiento recibido el aumenta el *cwnd* en 1 MSS. Así si recibe los dos reconocimientos correspondientes el *cwnd* se incrementa a 4 MSS.
- Realmente *cwnd* crece exponencialmente.

TCP: Arranque Lento – Aumento Exponencial



TCP: Arranque Lento

- TCP asume que la pérdida de un paquetes es una señal de congestión en alguna lugar de la red.
- Hay dos indicaciones de la pérdida de una paquete:
  - Time out ocurre.
  - Recepción de un ACK duplicado.

TCP: Detección de la Congestión -Disminución Multiplicativa

- El arranque lento puede impulsar la red a un punto de saturación muy difícil de recuperar.
- Se requiere combinarlo con una política de disminución multiplicativa.
- El mismo consiste en lo siguiente. Cuando el time out expira o se reciben ACKs duplicados :
  - Colocar el límite del arranque lento a la mitad de la ventana de congestión actual. Es decir
  - ssthresh = cwnd /2

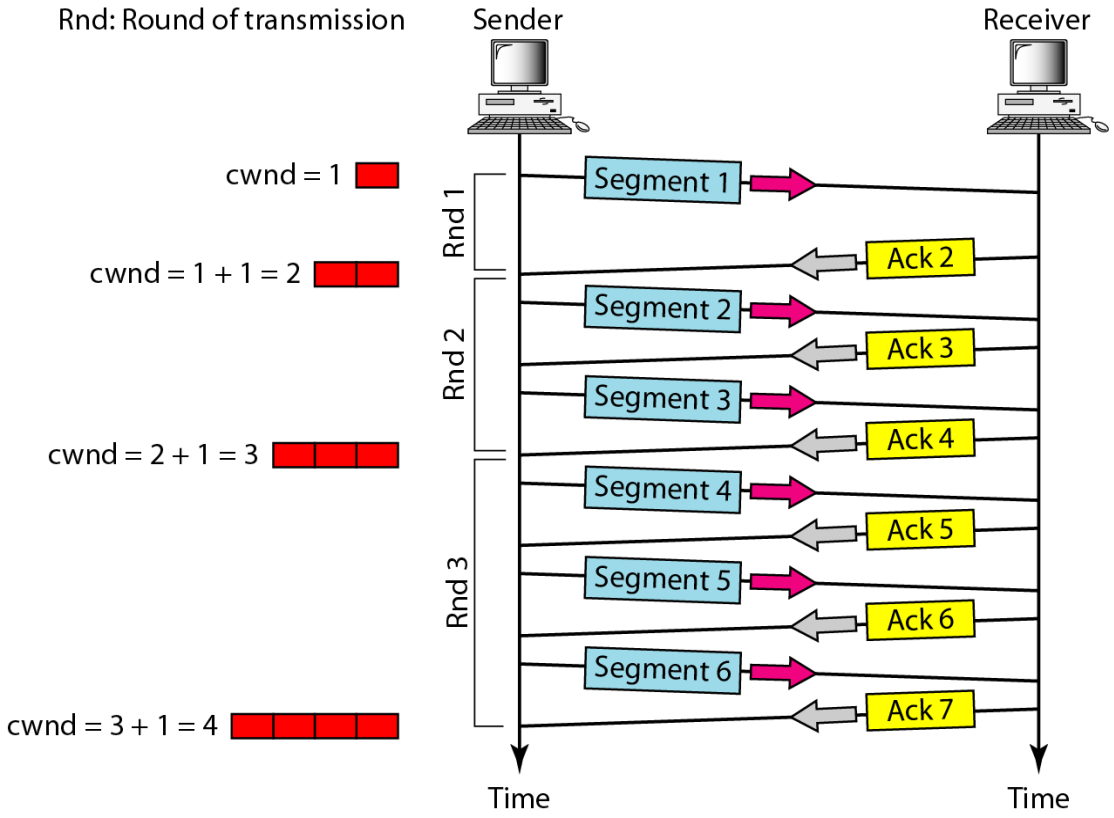
TCP: Detección de la Congestión -Disminución Multiplicativa

- Colocar el cwnd = 1(solo si se venció el time out)
- Aplicar el mecanismo de arranque lento hasta que cwnd = ssthresh.

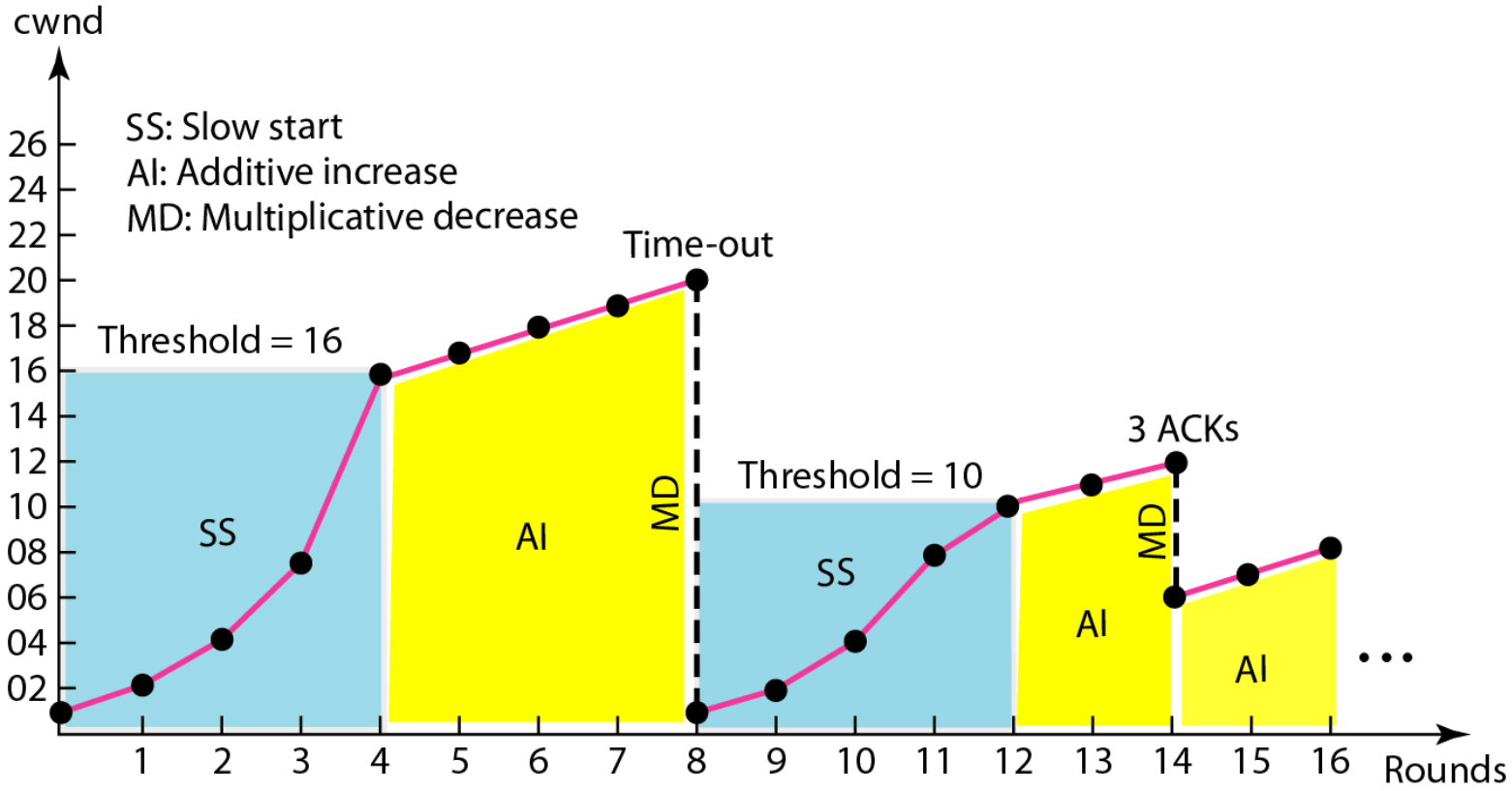
TCP: Prevención de congestión – Aumento Aditivo

- Cuando se alcanza el punto donde cwnd ≥ ssthresh, incrementar cwnd por cada viaje redondo, incrementar cwnd en 1 solamente si, para todos los segmentos en la ventana se ha recibido un reconocimiento.

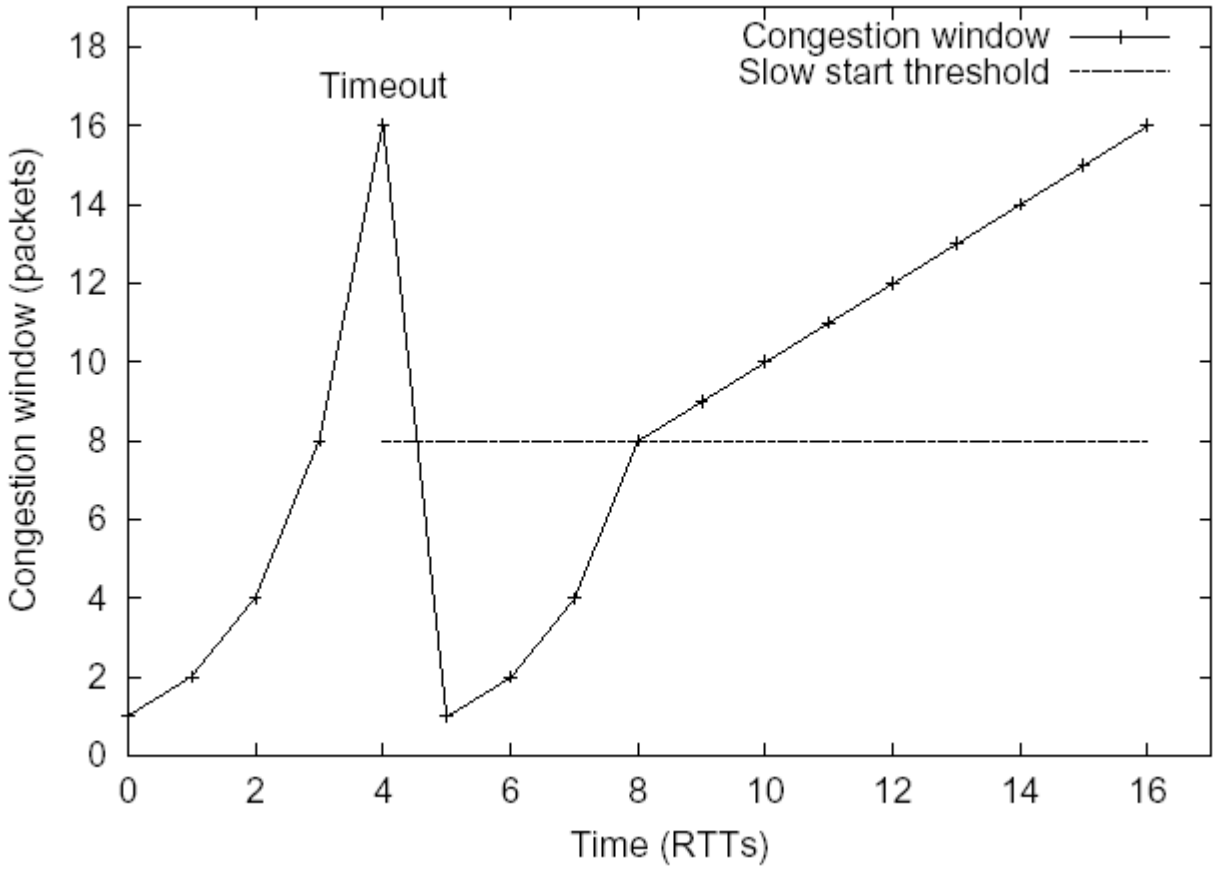
TCP: Prevención de congestión – Aumento Aditivo



TCP: Resumen de las Técnicas de Control de Congestión



TCP: Control de Congestión



TCP: Retransmisión Rápida y Recuperación Rápida

- En el funcionamiento anterior, se pueden esperar por largos períodos esperando que un timer expire.

TCP: Retransmisión Rápida

- TCP debe generar un ACK inmediato (ACK duplicado) cuando un segmento fuera de orden es recibido.
- Este ACK duplicado no debería ser retardado.
- Su propósito: indicar que el segmento esta duplicado y el número de secuencia esperado.

TCP: Recuperación Rápida

- Los algoritmos funcionan así:
  - Cuando un 3er ACK duplicado es recibido:
    - *ssthresh = cwnd/2*
    - Retransmitir el segmento pérdido.
    - *cwnd = ssthresh + 3*
  - Cada vez que un ACK duplicado llega:
    - *cwnd = cwnd + 1*
    - Transmitir paquete (si es permitido).

TCP: Recuperación Rápida

- Cuando próximo ACK llega (que reconoce nueva data).
  - *cwnd = ssthresh*
  - Este ACK debería reconocer el paquete retransmitido en paso 1 y paquetes intermedios.
- Este último paso es impedimento de congestión (no arranque lento).
- Entonces se conoce como recuperación rápida (*fast recobvery*).

Reconocimiento Selectivos (SACKs).

- TCP experimenta pobre rendimiento cuando múltiples paquetes de una ventana de datos se pierden.
- El receptor de segmentos TCP envía paquetes SACK al emisor.
- Para informar la data que ha recibido.

Reconocimiento Selectivos (SACKs)

- Opciones:
  - SACK permitido:
    - Enviado en un segmento SYN.
    - Indica que SACK puede ser usado durante la conexión.
  - SACK
    - Es la opción misma.

Reconocimiento Selectivos (SACKs)

- Opción Sack permitido:

Tipo = 4	Long = 2

Reconocimiento Selectivos (SACKs)

- Opción SACK

	Tipo =5	Long
Lado Izq del 1er bloque		
Lado Der del 1er bloque		
. . .		
Lado Izq del nesimo bloque		
Lado Der del nesimo bloque		

Reconocimiento Selectivos (SACKs)

- Opción SACK:
  - Se envía para informar sobre bloques de datos no contiguos recibidos.
  - No cambia significado de ACK.
  - Es informativa.

Reconocimiento Selectivos (SACKs)

- Comportamiento del receptor:
- Solo debe usar la opción SACK, si permitido (ver opción Sack permitido).
- Debe ser incluido en ACKs que no reconocen el paquete con el número de secuencia mas alto en la cola.
- Esto significa que ha habido pérdidas o re ordenamiento

Reconocimiento Selectivos (SACKs)

- Comportamiento de emisor:
- Una posible implementación sería:
  - Por cada segmento en la cola de retransmisión hay un *flag* SACKed.
  - Indica que el segmento ha sido reportado en la opción SACK.
  - El emisor prende la opción para aquellos segmento selectivamente reconocidos.
  - Cualquier segmento con el *flag* apagado y menor que el máximo segmento reconocido selectivamente es disponible para retransmisión.

Reconocimiento Selectivos (SACKs): Ejemplos

- Assume the left window edge is 5000 and that the data transmitter sends a burst of 8 segments, each containing 500 data bytes.
- Case 1: The first 4 segments are received but the last 4 are dropped.
- *The data receiver will return a normal TCP ACK segment acknowledging sequence number 7000, with no SACK option.*

Reconocimiento Selectivos (SACKs): Ejemplos

- Case 2: The first segment is dropped but the remaining 7 are received.
- *Upon receiving each of the last seven packets, the data receiver will return a TCP ACK segment that acknowledges sequence number 5000 and contains a SACK option specifying one block of queued data:*

Triggering	ACK Segment	Left Edge	Right Edge
5000	(lost)		
5500	5000	5500	6000
6000	5000	5500	6500
6500	5000	5500	7000
7000	5000	5500	7500
7500	5000	5500	8000
8000	5000	5500	8500
8500	5000	5500	9000

Reconocimiento Selectivos (SACKs): Ejemplos

- Case 3: The 2nd, 4th, 6th, and 8th (last) segments are dropped.
- *The data receiver ACKs the first packet normally. The third, fifth, and seventh packets trigger SACK options as follows:*

Triggering	ACK Segment	First Block Left Edge	First Block Right Edge	2nd Block Left Edge	2nd Block Right Edge	3rd Block Left Edge	3rd Block Right Edge
5000	5500						
5500	(lost)						
6000	5500	6000	6500				
6500	(lost)						
7000	5500	7000	7500	6000	6500		
7500	(lost)						
8000	5500	8000	8500	7000	7500	6000	6500
8500	(lost)						

Reconocimiento Selectivos (SACKs): Ejemplos

- *Suppose at this point, the 4th packet is received out of order. (This could either be because the data was badly misordered in the network, or because the 2nd packet was retransmitted and lost, and then the 4th packet was retransmitted). At this point the data receiver has only two SACK blocks to report. The data receiver replies with the following Selective Acknowledgment:*

Triggering Segment	ACK	First Block Left Edge	First Block Right Edge	2nd Block Left Edge	2nd Block Right Edge	3rd Block Left Edge	3rd Block Right Edge
6500	5500	6000	7500	8000	8500		

Reconocimiento Selectivos (SACKs): Ejemplos

- *Suppose at this point, the 2nd segment is received. The data receiver then replies with the following Selective Acknowledgment:*

Triggering Segment	ACK	First Block Left Edge	First Block Right Edge	2nd Block Left Edge	2nd Block Right Edge	3rd Block Left Edge	3rd Block Right Edge
5500	7500	8000	8500				