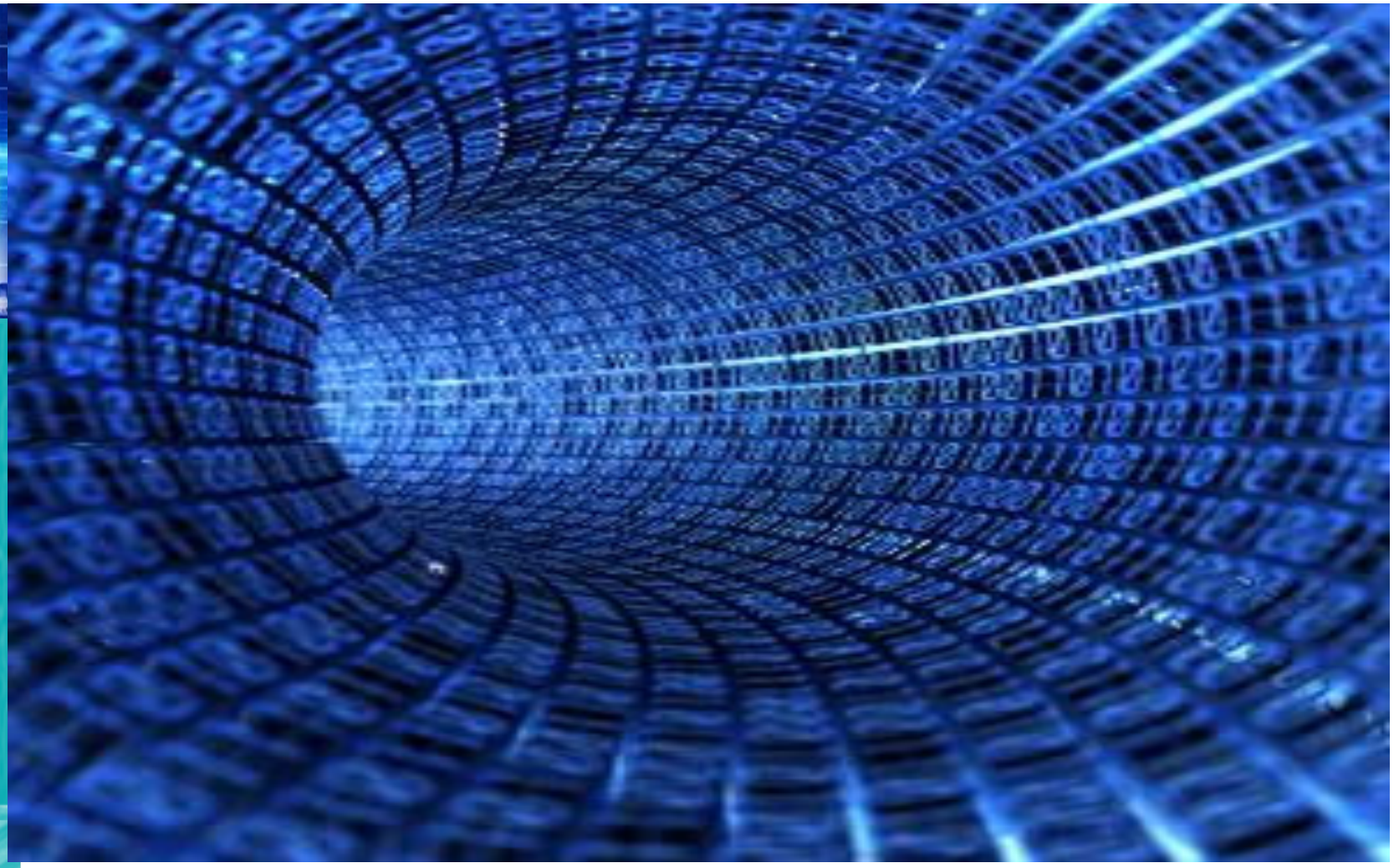


UNIDAD 3

PROTOCOLOS DE TRANSPORTE Y RUTEO DINÁMICO

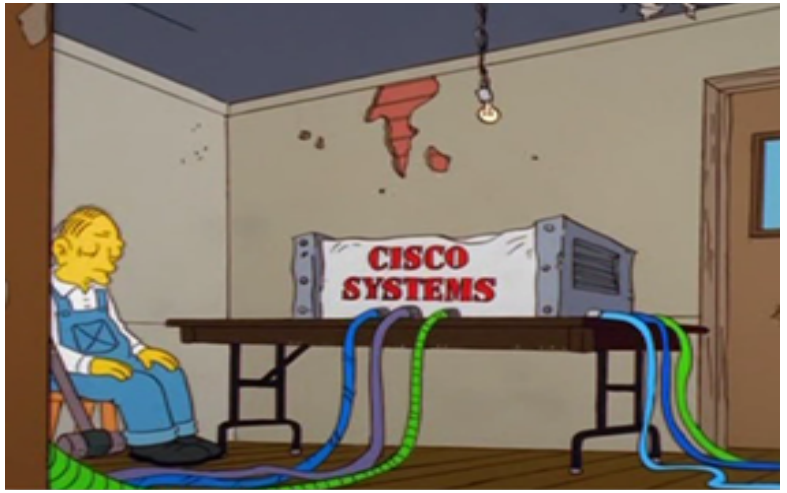
CC 4303 - Redes

Transporte y Ruteo Dinámico

CC4303 Redes

CONTENIDOS DE LA UNIDAD

- Protocolos y Encabezados End-to-End
- UDP
- Corrección de Errores
- TCP
- Anycast & Multicast
- Ruteo Interno
- Ruteo Externo
- Seguridad
- DNS



CC4303 Redes

END-TO-END ARGUMENT (1)

- Uno de los principios básicos del protocolo IP es realizar la inteligencia y el manejo de las conexiones en los extremos de la red de comunicaciones.
  - Por eso se dice que las redes TCP/IP son redes "tontas", que conectan dispositivos "inteligentes"
  - IP puede perder, duplicar y hasta alterar datos
- Lo anterior es lo que se conoce como *End-to-End Argument: "Todo lo que puede implementarse en los extremos en vez de en la red, se DEBE implementar SOLO en los extremos"*
- La primera vez que se introdujo este concepto fue en un artículo de 1981 de Saltzter, Reed y Clark del MIT.

END-TO-END ARGUMENT (2)

- En la práctica se traduce en que se trata de minimizar la intervención de los routers en los encabezados de los paquetes, dejando todo el procesamiento posible en los nodos extremos de la comunicación (el cliente y el servidor).
- La red telefónica era al revés: red “inteligente” para dispositivos “tontos”
- Hubo intentos de redes “inteligentes” para Internet, como la RDSI (ISDN en inglés) y ATM. Todas fracasaron.
- Las inteligencias no se suman bien y sí suman costos
- IP es la capa “tonta”, Transporte es la capa “inteligente”

CC4303 Redes

INTRODUCCIÓN (2)

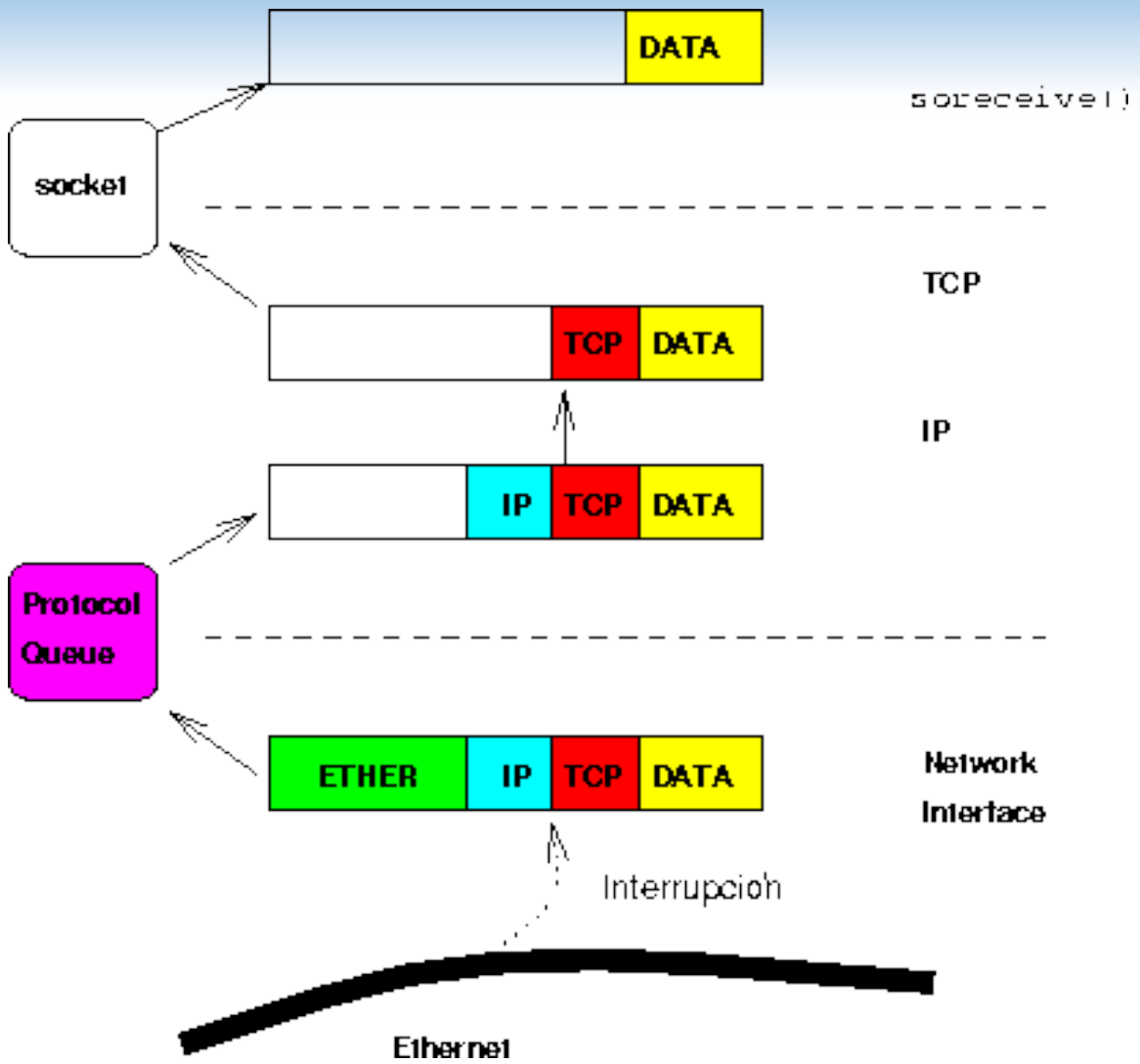
- Transporte es la capa que implementa sockets
  - Suponemos una red que conecta direcciones IP
  - Se permite enviar “paquetes” de datos
  - Las direcciones IP son únicas en el mundo
  - La red puede perder datos

CC4303 Redes

INTRODUCCIÓN (3)

- Existe dos líneas principales de servicios entregados por la capa de transporte (en el modelo de Internet)
  - TCP: Transmission Control Protocol (el inteligente)
    - Orientado al flujo confiable de bytes
    - Provee control de flujo, control de errores, orden.
  - UDP: User Datagram Protocol (el tonto)
    - Orientado principalmente a la mensajería.
    - Es asíncrono.
    - No confiable, sin control de flujo ni orden
    - Da la misma calidad que Internet (IP) puro

INTRODUCCIÓN (4)



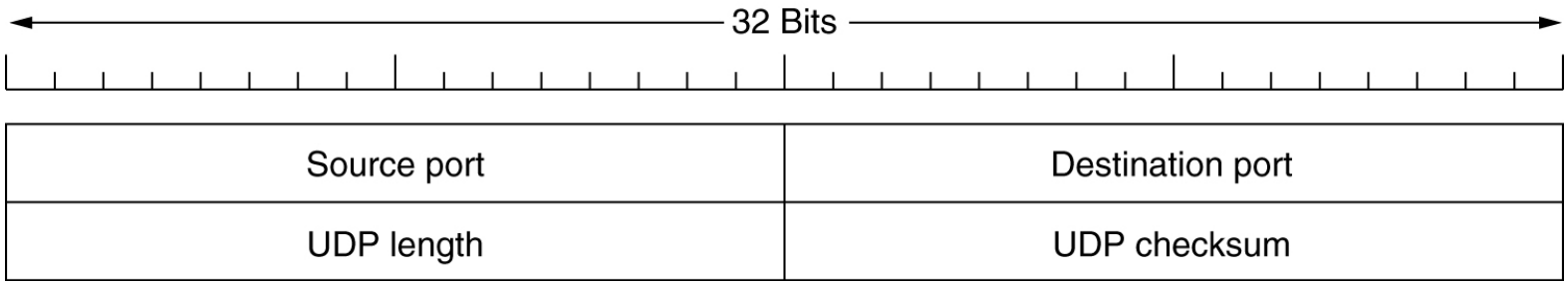
CC4303 Redes

PROTOCOLO UDP (1)

- Protocolo de transporte muy simple.
- Extiende las características de datagramas IP entre hosts a la comunicación entre procesos.
- Orientado a los mensajes y a los sistemas que funcionan en base a request/reply. En éstos casos, el establecimiento de una conexión es un costo que no vale la pena asumir.
- El control de flujo, orden y recuperación de errores queda en manos de la aplicación.
- Provee un checksum para detección de errores

PROTOCOLO UDP (2)

- El checksum se calcula sobre el header UDP mas un *pseudoheader*: campos del header IP como Protocolo, IP origen, IP destino y el largo del payload.



MULTIPLEXIÓN POR PUERTOS (1)

- Los servicios de la capa de transporte requieren que el emisor y receptor establezcan un socket. Cada socket se identifica por el par (IP, puerto).
- Las conexiones se identifican por los identificadores de los sockets a ambos lados.
- Estos permite asociar una aplicación con una conexión, mediante una tabla interna propia del S.O.

PID	Socket
1234	1
4563	2
8890	3

Socket	IP	Puerto
1	10.0.45.3	3535
2	10.0.45.4	5050
3	10.0.45.4	8080

CC4303 Redes

MULTIPLEXIÓN POR PUERTOS (2)

- Los puertos son números de 16 bits (por lo que existen 65535 posibilidades).
- Se dividen en dos categorías: Puertos bien conocidos (*well-known ports*, aquellos menores al 49.151) y puertos no privilegiados (todos los otros)
- Los puertos bien conocidos están reservados para los servicios estándares. Así, un cliente conoce exactamente a qué puerto dirigir un requerimiento.
- Cada flujo se identifica por la tupla (Dir_{origen}, $Puerto_{origen}$, $Dir_{destino}$, $Puerto_{Destino}$, Protocolo)

MULTIPLEXIÓN POR PUERTOS (3)

- Ejemplos de puertos

Servicio	Puerto	Protocolo	Servicio	Puerto	Protocolo
FTP	21	TCP	POP3	110	TCP
SSH	22	TCP	SunRPC	111	TCP y UDP
Telnet	23	TCP	NNTP	119	TCP
SMTP	25	TCP	NTP	123	TCP y UDP
Whois	43	TCP y UDP	SNMP	161	TCP y UDP
DNS	53	TCP y UDP	BGP	179	TCP
HTTP	80	TCP	HTTPS	443	TCP

EL5107 Tecnologías de Información y Comunicación

TRANSMISIÓN CONFIABLE (1)

- Transmisión Confiable se traduce en el mecanismo para asegurar que todo lo que se envía es recibido, en el mismo orden original.
- En UDP un error se puede detectar
- Cuando un error aparece, la capa debe descartar el paquete
- Transporte Confiable debe recuperarse posteriormente de esa situación.

CC4303 Redes

TRANSMISIÓN CONFIABLE (2)

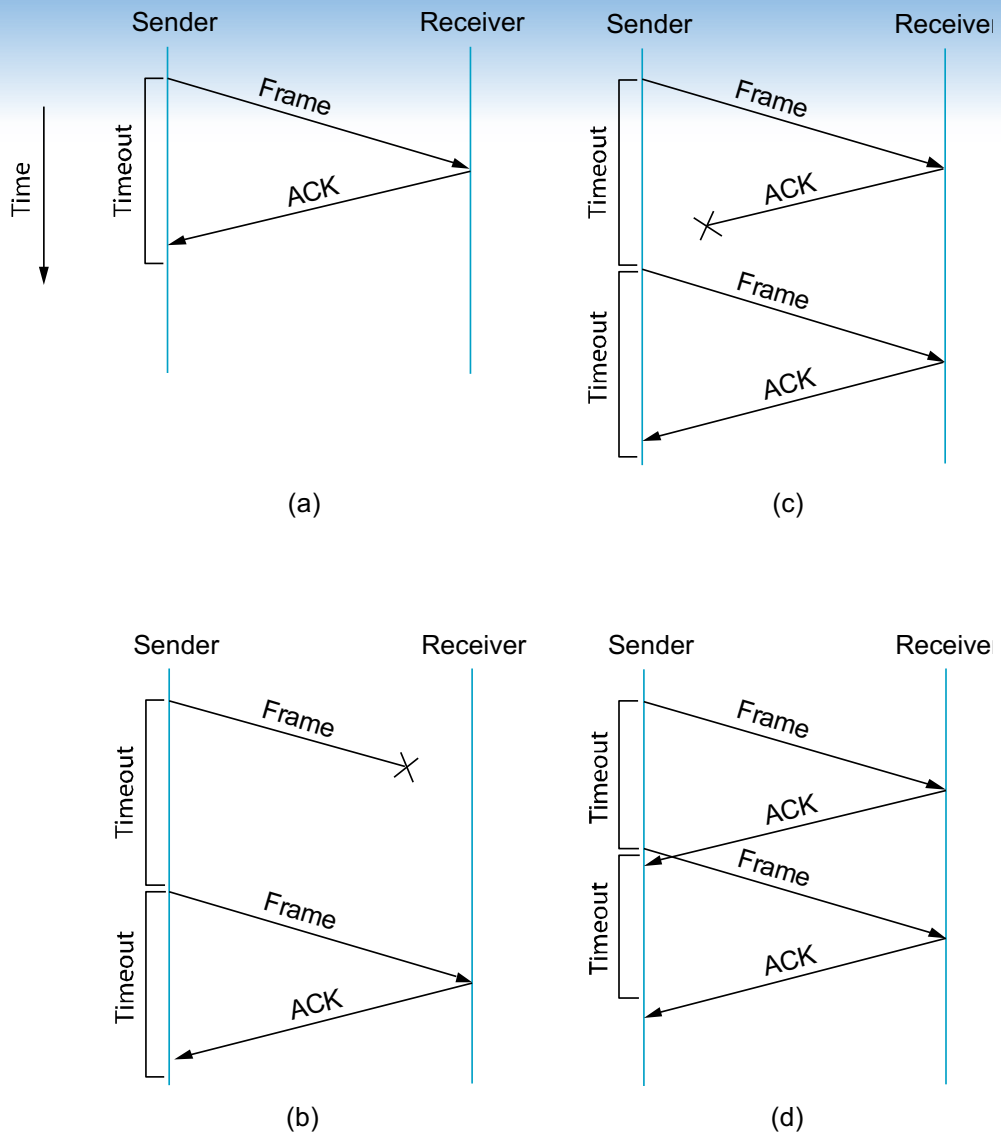
- Esto se logra mediante la combinación de los mecanismos: *acknowledgments (ACK)* y *timeout.*
- Los ACK son pequeños paquetes de control que un protocolo envía para emisor para notificar la recepción satisfactoria de un paquete previo.
  - Estos paquetes de control son sólo el encabezado (*header*) y no contienen datos. Por lo mismo, el receptor puede incluir un ACK en un paquete de datos que vaya en el sentido contrario. A este mecanismo se le conoce como *piggybacking*.

STOP-AND-WAIT (1)

- La idea es muy simple: Envío un paquete y me quedo esperando por el ACK respectivo antes de enviar otro paquete.
- Si después de cierta espera el ACK no llega, se retransmite.
- Revisemos algunos casos que se pueden producir durante la operación de este mecanismo.

CC4303 Redes

STOP-AND-WAIT (2)



STOP-AND-WAIT (3)

- En (a) se muestra el envío del paquete, y la posterior recepción del ACK.
- En (b) se muestra el envío de un paquete y su posterior pérdida, lo que genera una retransmisión correcta.
- En (c) se muestra el envío de un paquete, su recepción y generación del ACK, el cual se pierde antes de llegar al emisor, lo que genera una retransmisión incorrecta.
- En (d) se muestra un caso parecido a (c), pero el ACK no se pierde, sino que se retrasa. Esto genera una retransmisión incorrecta y un ACK duplicado.

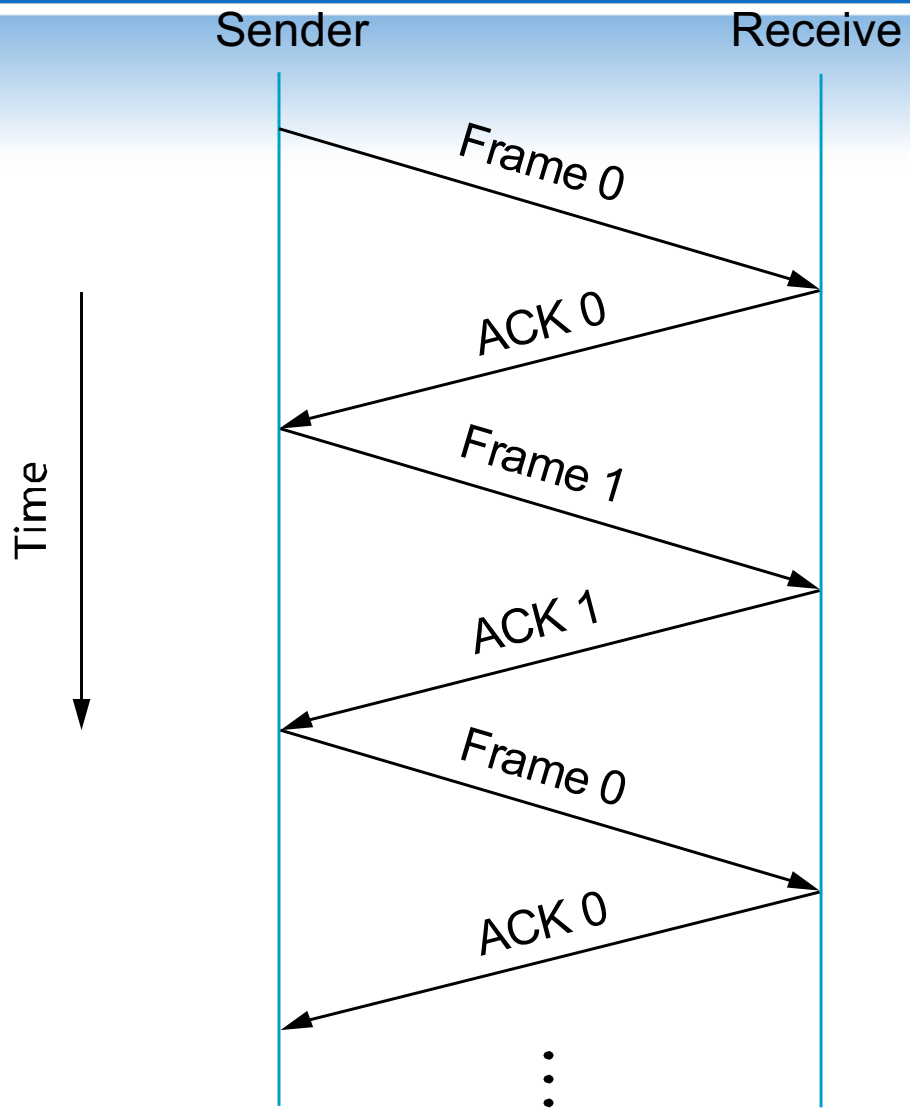
CC4303 Redes

STOP-AND-WAIT (4)

- (b), (c) o (d) indistinguibles
- En los casos (c) y (d) el emisor reenviará el paquete original, pero el receptor creerá que se trata de un nuevo paquete, lo que generará una duplicación.
- En (d) se duplica el ACK, haciendo creer al enviador que se recibió un nuevo paquete OK.
- Por ello, es necesario identificar el frame **Y** el ACK, por lo que se utilizan números de secuencia.
- Para este caso, basta un número de secuencia de 1 bit.

CC4303 Redes

STOP-AND-WAIT (5)



CC4303 Redes

STOP-AND-WAIT (6)

- La gran desventaja de Stop-and-Wait es que solo permite que un paquete de datos viaje al mismo tiempo, lo que se puede traducir en un uso ineficiente de la capacidad del enlace.
- Ejemplo:
  - Suponga un enlace de 1.5 Mbps con un RTT de 45 ms. El BDP (*bandwidth-delay product)* será de 67.5 Kb, aprox. 8KB.
  - Si un emisor envía un paquete de 1KB cada RTT, tendremos:

$$\text{BitsPorPaquete} \div \text{TiempoPorPaquete}$$

$$(1024 \times 8) \div 0.045$$

$$182 \text{ Kbps}$$

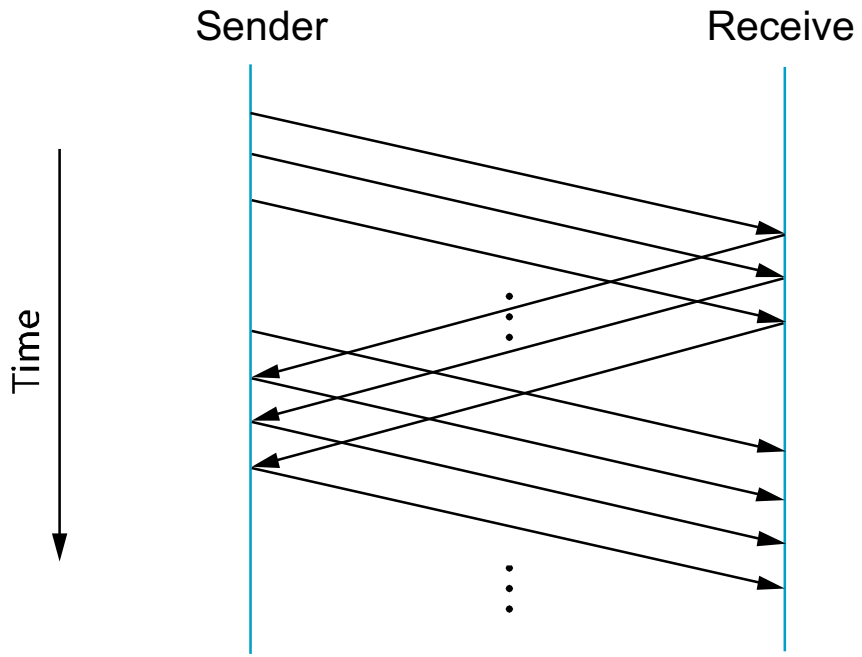
CC4303 Redes

STOP-AND-WAIT (7)

- El peor comportamiento lo tiene en *Long Fat Networks (LFN)*
- También llamadas “Elefantes”
- Ej: enlace satelital de 100Mbps, RTT de 600 ms
- BDP: ~6 Mbytes
- Es lo que cabría en el enlace hasta que llega el primer ACK
- Deberíamos entonces tener 'BDP' buffers de transmisión en espera de ACKs
- Enlace fibra 1Gbps, RTT 200ms -> 20 Mbytes

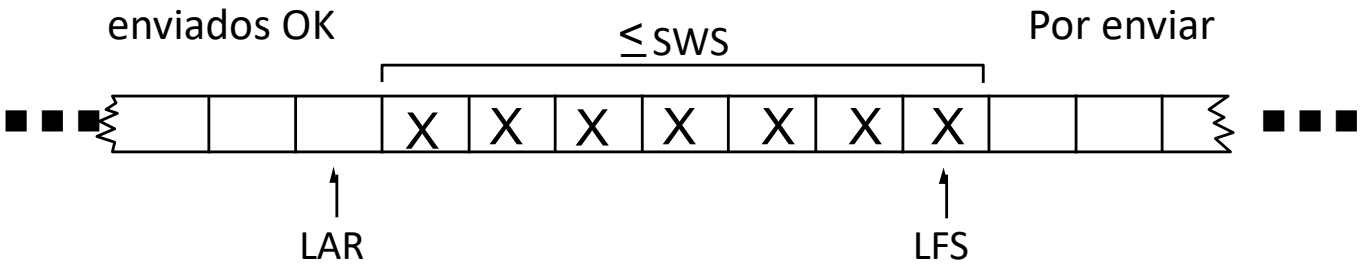
SLIDING WINDOW (1)

- En el caso de BDP 8KB y paquetes de 1KB quisiéramos enviar 8 paquetes a la vez, y poder enviar el noveno en cuanto llegue el ACK del primer paquete.



SLIDING WINDOW (2)

- El mecanismo de *Sliding Window* define lo siguientes elementos:
  - El emisor le asigna un *número de secuencia* a cada paquete que envía **SeqNum**.
  - El emisor mantiene tres variables. **SWS**=*send window size* (debiera aproximar BDP), que define el límite superior de paquetes sin confirmar que se pueden transmitir; **LAR**=*last ack received*, que indica el número de secuencia del último ACK recibido; **LFS**=*last frame sent*, que indica el número de secuencia del último paquete enviado.
  - El emisor mantiene el invariante:



CC4303 Redes

SLIDING WINDOW (3)

- Cuando se recibe un ACK, el emisor "mueve" LAR a la derecha, lo que permite enviar otro paquete.
- El emisor tiene un timer asociado a cada paquete enviado (aprox RTT).
- Si expira antes de la recepción del respectivo ACK, debe retransmitir. Esto se traduce en que el emisor debe ser capaz de mantener SWS paquetes en caso de que los necesite para retransmisión.

CC4303 Redes

SLIDING WINDOW (4) Go-back-N

- El receptor mantiene una ventana de tamaño 1:
  - ExpectedSeq == paquete esperado
  - Todo paquete distinto a ese es descartado
  - Pero siempre enviamos ACK para ExpectedSeq-1
- El emisor retransmite la ventana completa si hay un timeout (*Go-Back-N)*

CC4303 Redes

SLIDING WINDOW (4) Go-back-N

- Al recibir un ACKn con n > LAR+1
  - Puedo dar por recibidos OK todos los paquetes enviados desde LAR+1 hasta n (no me habrían enviado este ACK si no los hubieran recibido bien)
  - Se define ACKn ⟹ ACKi i <= n

Al recibir un paquete con número de secuencia > ExpectedSeq

- Puedo dar por perdidos los paquetes entre medio
- Enviar NACK para ExpectedSeq y apurar la retransmisión

CC4303 Redes

SLIDING WINDOW (4) Go-back-N

- Mucho más eficiente que Stop&Wait si la ventana de emisión es suficiente para llenar el BDP de la conexión
- Frente a pérdidas: retransmite la ventana completa, problemas si es muy grande y hay muchos errores
- En enlaces de alto BDP y alta pérdida necesitamos una mejor solución: aprovechar los paquetes transmitidos OK y no re-transmitirlos.
- Soporta mejor la pérdida de ACKs que de datos

CC4303 Redes

SLIDING WINDOW (4) Selective Repeat

- Ahora el receptor mantiene ventana también, con tres variables:
  - **RWS**=receive window size, que define el límite máximo de paquetes desordenados que puede aceptar.
  - **LAF**=largest acceptable frame, que define el número de secuencia más grande que puede aceptar.
  - **LFR**=last frame received, que define el número de secuencia del último paquete recibido tal que todos los paquetes con secuencia <= LFR ya se recibieron OK.
- El receptor mantiene el siguiente invariante:

$$LAF - LFR \leq RWS$$

CC4303 Redes

SLIDING WINDOW (5) Selective Repeat

- Cuando un paquete con número SeqNum llega, el receptor hace lo siguiente:
  - Si SeqNum<=LFR o SeqNum>LAF, se considera fuera de la ventana de recepción y se descarta.
  - Si LFR<SeqNum<=LAF, entonces el paquete está dentro de la ventana de recepción y es aceptado.
  - SeqNum <= LFR es una retransmisión
  - SeqNum > LAF es un paquete que no puedo recibir ya que mi ventana de recepción es muy pequeña

CC4303 Redes

SLIDING WINDOW (6) Selective Repeat

- Cuando un paquete con número SeqNum llega, el receptor hace lo siguiente (Continuación):
  - El receptor envía el ACK para el paquete recibido (SeqNum) si SeqNum <= LAF
  - Si SeqNum > LAF, envío ACK para LRF
  - Si SeqNum == LRF+1, ajusta la ventana hasta el próximo paquete no recibido.
  - ***Ahora un ACKn no implica Acks a < n***

CC4303 Redes

SLIDING WINDOW (9)

- Notas
  - El tamaño de la ventana de emisión se calcula en base al BDP.
  - El tamaño de la ventana de recepción puede ser cualquiera.
    - Si RWS=1, el receptor no aceptará paquetes desordenados.
    - Si RWS=SWS, el receptor puede aceptar cualquier paquete que el emisor envíe.
    - Si RWS>SWS, no tiene mucho sentido, pues no pueden más de SWS paquetes desordenados.

EL5107 Tecnologías de Información y Comunicación

NÚMEROS DE SECUENCIA (1)

- Originalmente consideramos que no habían restricciones para el número de secuencia, pero en la práctica eso no es posible: debe ser un número finito.
- Por lo tanto, los números de secuencia se reutilizan de cuando en cuando, lo que agrega un nuevo problema: cómo diferenciar un paquete “antiguo” de uno “nuevo” si tienen el mismo número.
- Ello nos obliga a estar seguros que el número máximo de secuencia sea mayor que el número de paquetes en viaje.
  - Por ejemplo, en Stop-and-Wait tenemos como máximo dos números de secuencia y sólo uno en viaje a la vez.

EL5107 Tecnologías de Información y Comunicación

NÚMEROS DE SECUENCIA (2)

- Supongamos el caso de que SWS<=MaxSeqNum-1, donde MaxSeqNum es número de secuencias disponibles.
  - Si RWS=1, el valor es suficiente.
  - Si RWS=SWS, no es suficiente. Veamos un ejemplo
    - Sea MaxSeqNum=8, SWS=RWS=7.
    - El emisor envía los paquetes del 0 al 6, que son recibidos satisfactoriamente, pero los ACK se pierden.
    - El receptor espera ahora los paquetes 7 y 0..5, pero el emisor considera perdidos los paquetes 0..6 y los retransmite.
    - Al recibir el segundo bloque de paquetes, el receptor cree que está recibiendo la segunda encarnación de los paquetes, cuando en realidad son copias de los recibidos previamente.

EL5107 Tecnologías de Información y Comunicación

NÚMEROS DE SECUENCIA (3)

- La fórmula general está dada por Stop-and-Wait: que los números de secuencia alternen entre las dos mitades del espacio de números.
- Con ello podemos deducir que SWS<(MaxSeqNum+1)/2
- Todo número recibido fuera de LRF+SWS se considera “anterior” y enviamos ACK