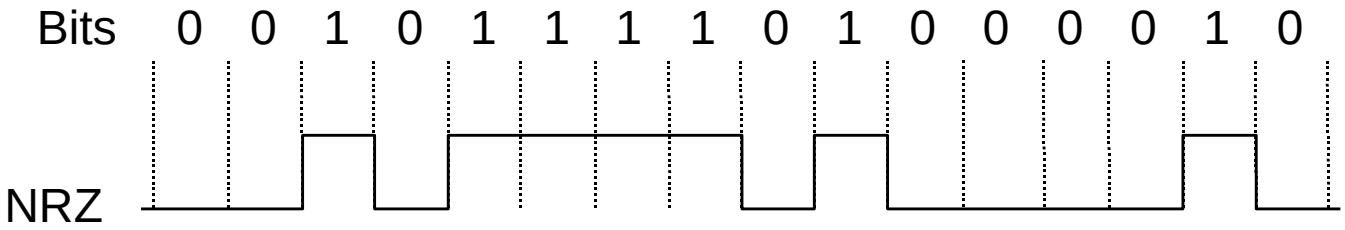
Enlaces Punto a Punto

Contenido

- Codificación
- Tramado (Framing)
- Detección de Errores
- Algoritmo Ventana Deslizante (Sliding Window Algorithm)

Codificación

- Las señales se propagan sobre un medio físico
  - Ondas electromagnéticas moduladas
  - Variaciones de voltaje
  - Referirse a Medios de transmisión
- Codificación de datos binarios en señales
  - Ej. 0 como señal baja y 1 como alto, tambien 0 como f0 y 1 como f1 (FSK)
  - Se conoce como codificación Non-Return to zero (NRZ)



Problema: 1s 0s Consecutivos

- Señal baja (0) podría ser interpretada como ausencia de señal
- Señal alta (1) podría conducir a perdida del nivel de referencia de señal
- Incapacidad para recuperar el reloj si no hay cambios garantizados
- Ejemplo: RS232

Codificaciones Alternativas

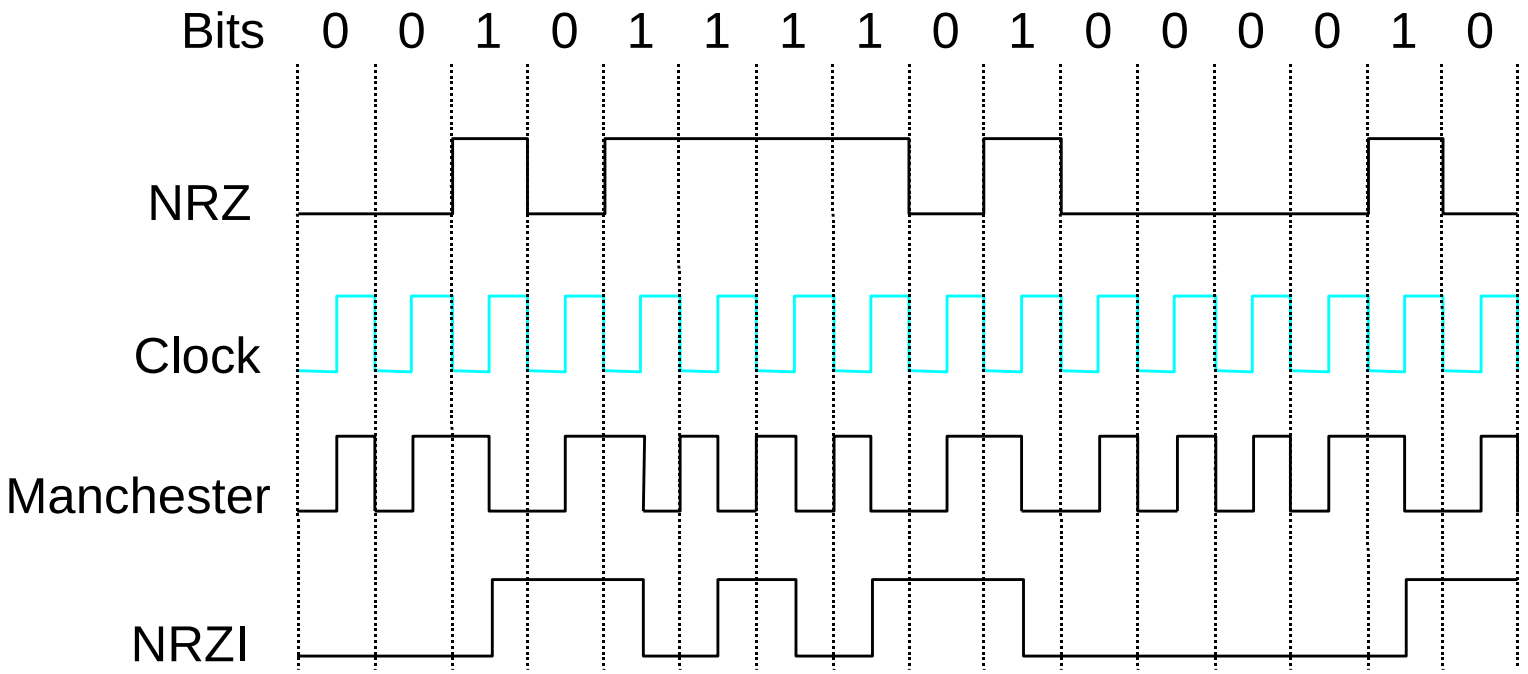
- Non-return to Zero Inverted (NRZI)
  - Genera una transición de la señal para codificar un uno; mantiene la señal sin cambio para codificar un cero.
  - Resuelve el problema de unos consecutivos.

- Manchester
  - Transmite el XOR de los datos codificados NRZ y el reloj
  - Solo alcanza 50% de eficiencia en termino de bit por ancho de banda. En otras palabras ocupa el doble ancho de banda (o tasa de bits = 1/2 tasa de baudios)

Codificación (cont)

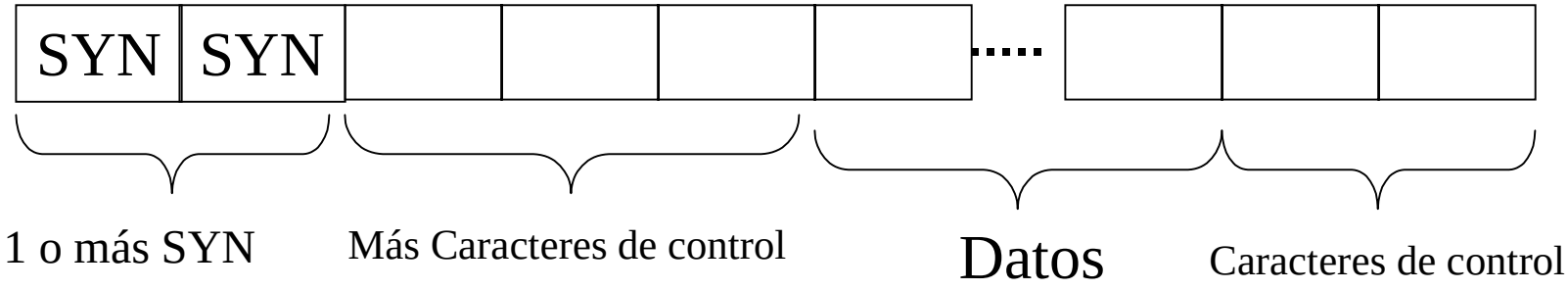
- 4B/5B
  - Cada 4 bits de datos se codifica en códigos de 5-bit
  - los códigos de 5 bits son seleccionados para tener no mas de un 0 inicial y no mas de dos 0s finales.
  - Así, nunca se tienen mas de tres 0s consecutivos
  - La palabra de código de 5 bit son transmitidas usando NRZI
  - Se logra 80% de eficiencia

Codificación (cont)

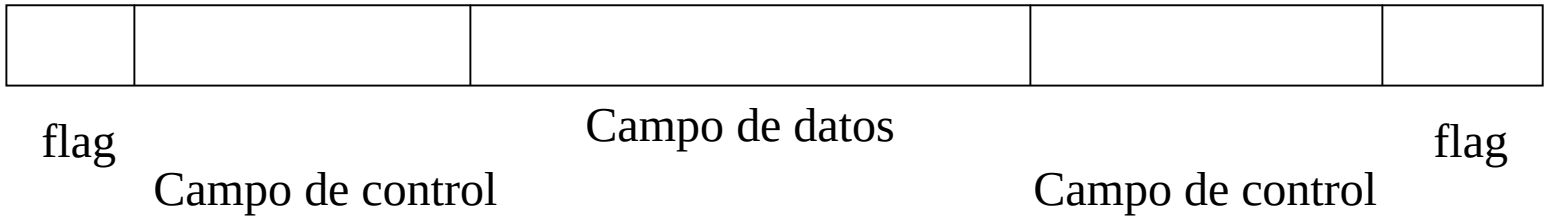


Transmisión orientada al carácter y al bit

- En la práctica se usan dos esquemas :
- La **transmisión síncrona orientada al carácter**
  - El bloque de datos es tratado como una sequencia de caracteres (usualmente de 8 bits).

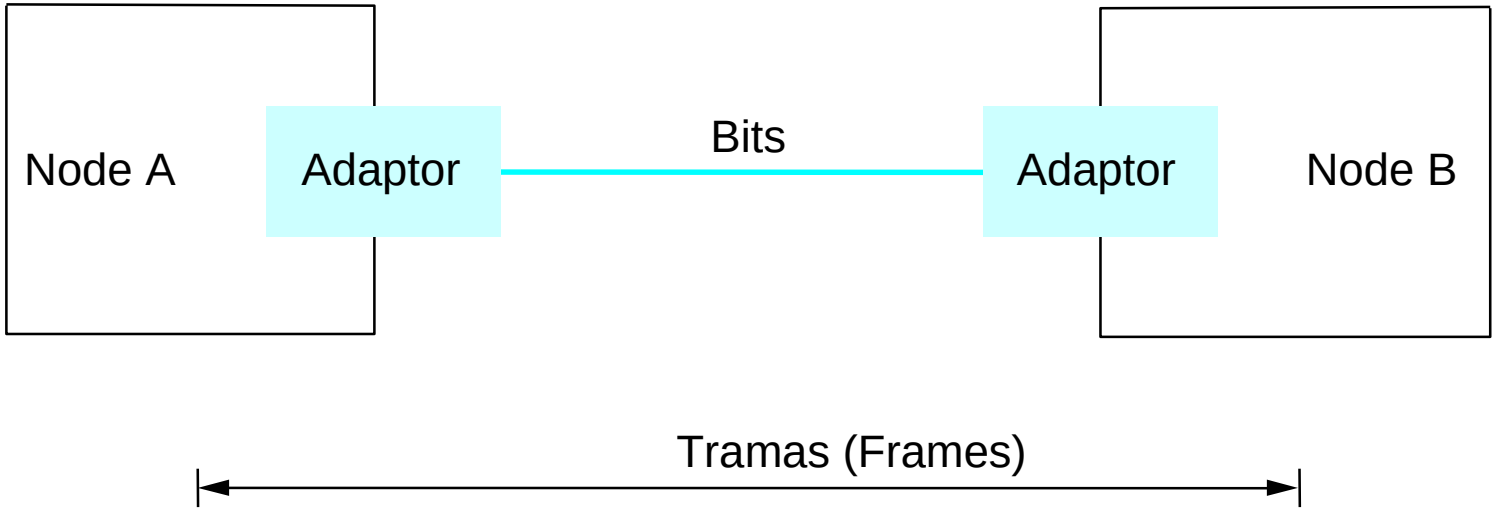


- La **transmisión síncrona orientada al bit**
  - El bloque de datos es tratado como una sequencia de bits



Entramado

- La secuencia de bits es organizada en tramas
- Esta función es típicamente implementada por el adaptador de red



Marcas de inicio y fin de trama

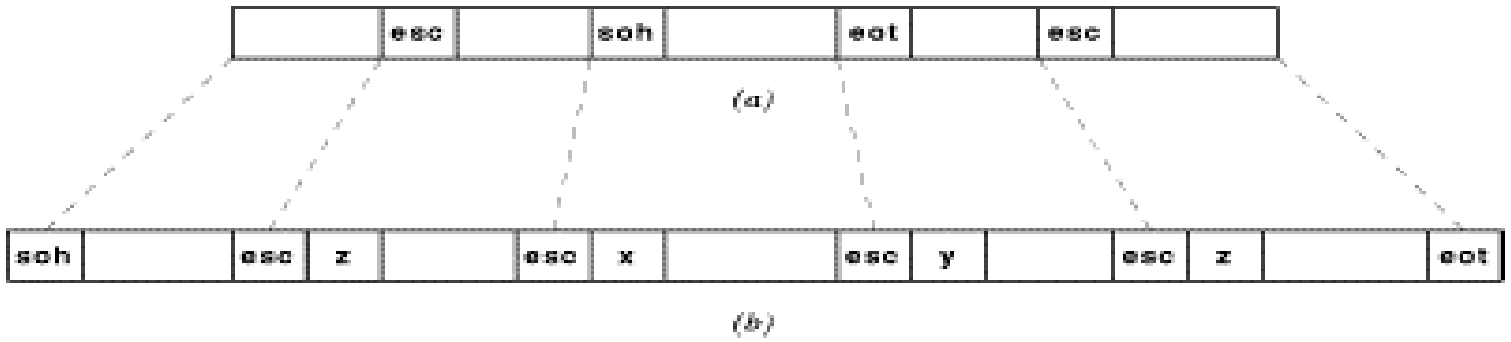
- Desventajas de poner marcas de inicio y fin de trama:
  - Overhead: i.e. El uso de símbolos que no portan información "útil". Considere secuencia de paquetes adyacentes.
- Ventaja:
  - permiten detectar fallas en los computadores y/o enlaces.
- Qué pasa si estos símbolos aparecen en los datos?

Bytes y bits de Relleno

- No podemos reservar dos símbolos para uso exclusivo de la red.
- El tx modifica levemente la secuencia que envía para asegurar que las marcas de inicio y término sean únicas.
- La red inserta bytes o bits extras cuando las marcas aparece en los datos. Esta técnica se conoce como ***byte stuffing*** o ***bit stuffing.***

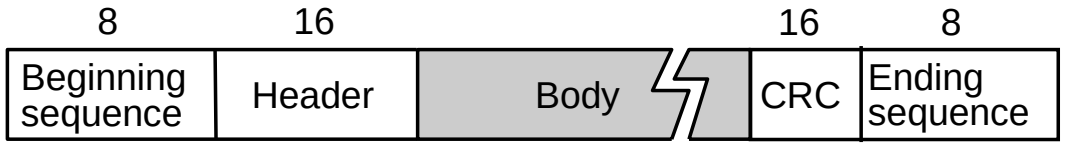
Ejemplo: byte stuffing

Character In Data	Characters sent
soh	esc x
eot	esc y
esc	esc z



Esquemas de Entramado

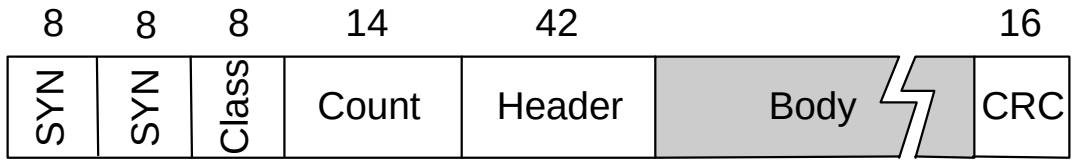
- Basado en centinela
  - Se delimitan las tramas con una patrón especial: 01111110
  - e.g., HDLC, SDLC, PPP



  - problema: el patrón especial también aparece en la carga (datos).
  - Una solución: *bit stuffing*
    - Tx: inserta 0 después de cinco 1s consecutivos
    - Rx: descarta 0 que sigue cinco 1s consecutivos

Esquemas de Entramado (cont)

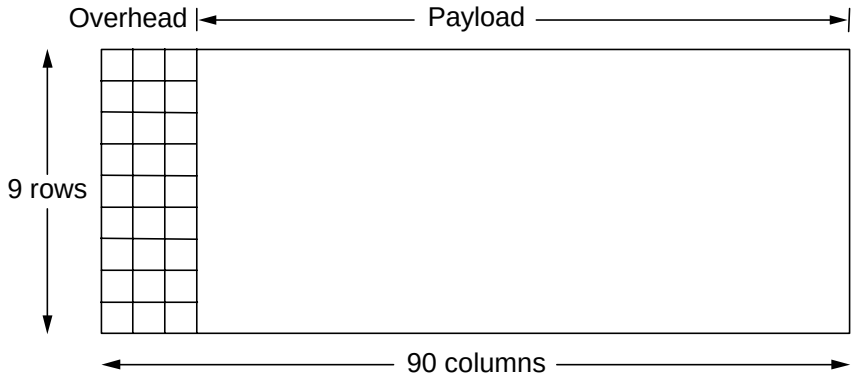
- Basado en cuenta o largo
  - En el encabezado se incluye el largo de la carga
  - e.g., DDCMP

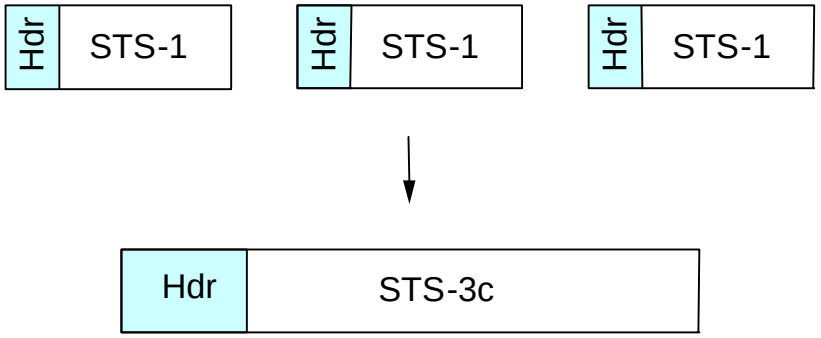


  - problema: El campo de cuenta se puede corromper
  - solución: Detectar cuando el CRC (Cyclic Redundancy Check) falla

Esquemas de Entramado (cont)

- Basados en reloj
  - Cada trama tiene una duración de 125us
  - ej., SONET: Synchronous Optical Network
  - STS-*n* (STS-1 = 51.84 Mbps)





Chequeo de Redundancia Cíclica (Cyclic Redundancy Check)

- Se agregan k bits de redundancia a los n-bit del mensaje
  - interesa que $k << n$
  - Ej., $k = 32$ y $n = 12{,}000$ (1500 bytes)
- Se representan n-bit de mensaje como un polinomio de grado n-1
  - Ej., MSG=10011010 => $M(x) = x^7 + x^4 + x^3 + x^1$
- Sea k el grado de algún polinomio divisor
  - Ej., $C(x) = x^3 + x^2 + 1$

CRC (cont)

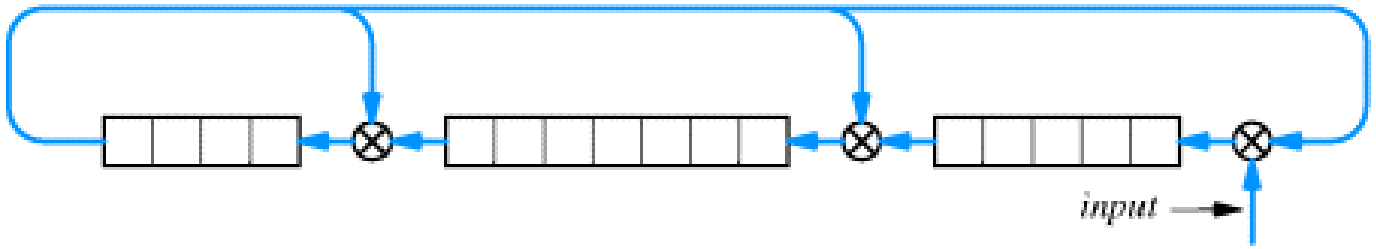
- Se transmite el polinomio $P(x)$ tal que sea divisible en forma exacta por $C(x)$
  - Se corre a la izquierda k bits, i.e., $M(x)x^k$
  - restar el resto de $M(x)x^k$ / $C(x)$ de $M(x)x^k$
- En general se recibe el polinomio $P(x) + E(x)$
  - $E(x) = 0$ implica ausencia de errores
- Se divide $(P(x) + E(x))$ por $C(x)$; esto cero si:
  - $E(x)$ fue cero (ningún error), o
  - $E(x)$ es exactamente divisible por $C(x)$

Seleccionando $C(x)$

- Para detectar:
- Todo error simple, x^k y x^0 deben tener coeficiente no cero.
- Todo error doble, $C(x)$ debe contener un factor con al menos tres términos
- Cualquier número impar de errores, $C(x)$ debe contener el factor $(x + 1)$
- Detecta cualquier “ráfaga” de errores (i.e., secuencia de bits consecutivos errados) para la cual el largo de la ráfaga es menor que k bits.
- La mayoría de ráfagas de largo mayor que k bits también pueden ser detectados.

Implementación en hardware

- Si P = 1000100000100001
- En otras palabras: $P(X)=X^{16}+X^{12}+X^{5}+1$
- El circuito de hardware es como sigue:



- Al término del mensaje el resto es el valor del registro de desplazamiento

Algoritmo de suma de chequeo usado en Internet

- Se ve al mensaje como una secuencia de enteros de 16-bits; Se suman usando aritmética complemento 1 de 16-bit; finalmente se toma el complemento uno del resultado.

```
u_short
cksum(u_short *buf, int count)
{
   register u_long sum = 0;
   while (count--)
   {
      sum += *(buf++);
      if (sum & 0xFFFF0000)
      {
         /* carry occurred, so wrap around */
         sum &= 0xFFFF;
         sum++;
      }
   }
   return ~(sum & 0xFFFF);
}
```

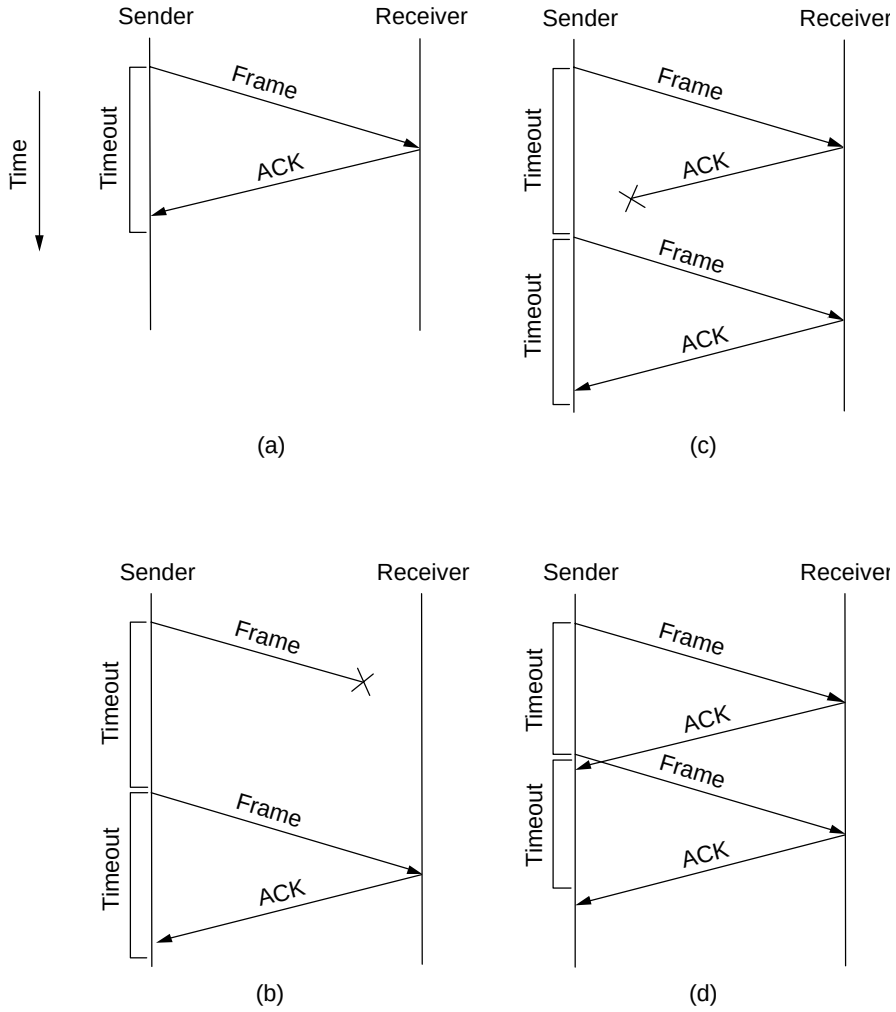
Ejemplo de calculo de CRC

- Supongamos: Código generador C(x)=x^3+x^2+1
- Mensaje: 10011010
- Codificación: k=3
-

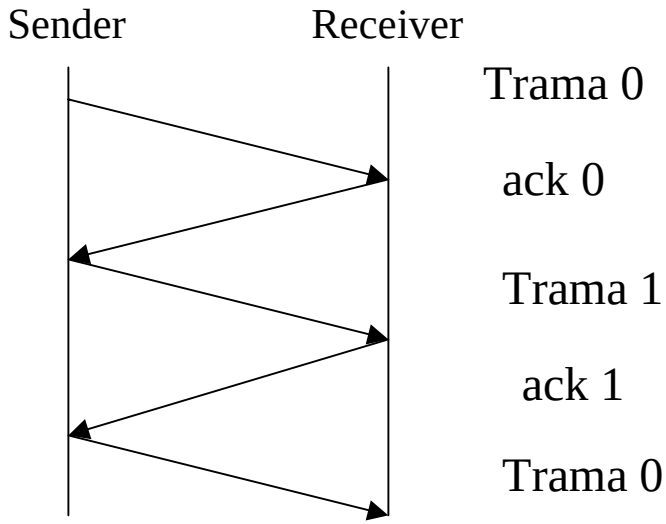
```
10011010000:1101=1111001
1101
 1001
 1101
  1000
  1101
   1011
   1101
    1100
    1101
     1000
     1101
      101  Resto
```

- Mensaje a transmitir: 10011011101

Acuses de Recibo (Acknowledgements) & Timeouts



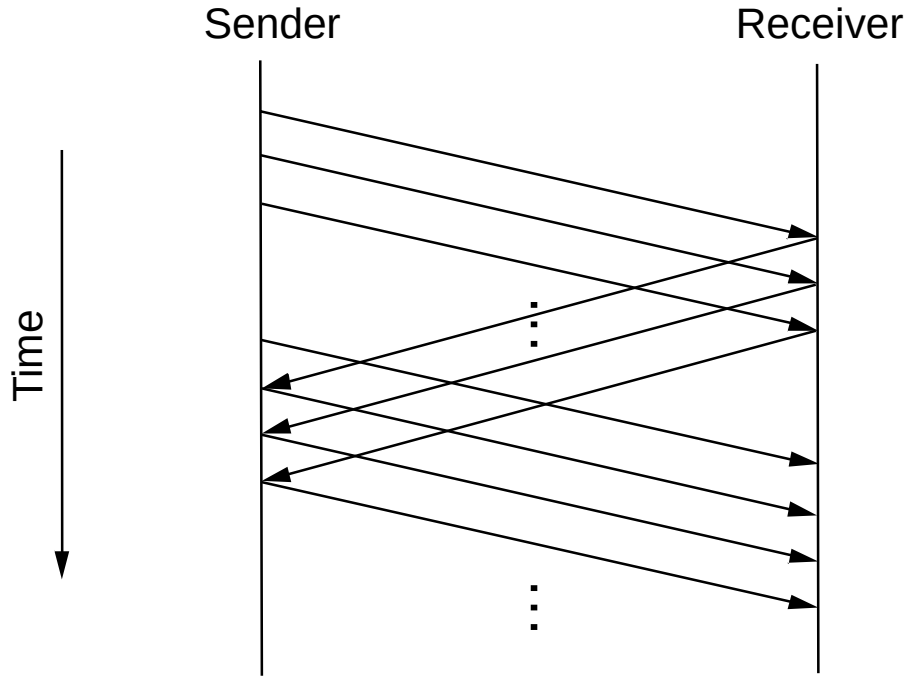
Protocolo Stop-and-Wait



- Usa un bit de numero de secuencia para detectar duplicados (cuando el ack se pierde).
- Problema: no mantiene la ruta ocupada (llena de datos).
- Ejemplo
  - Enlace de 1.5Mbps x 45ms RTT = 67.5Kb (8KB)
  - Tramas de 1KB implican 1/8 de utilización del enlace

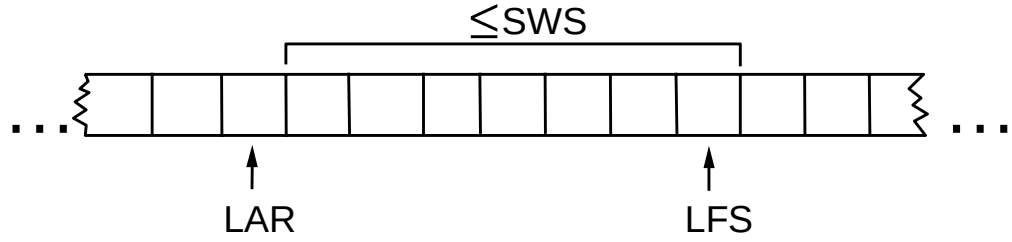
Protocolo Ventana Deslizante (Sliding Window, SW)

- Permite múltiples tramas no confirmadas (sin ACK)
- Hay un límite para el número de tramas no confirmadas o pendientes, llamado ventana (*window)*



SW: Transmisor

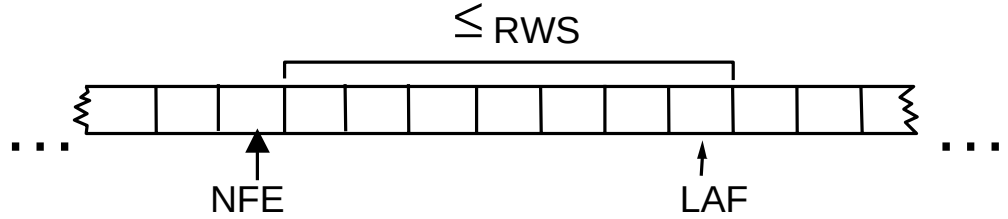
- Asigna una secuencia de números a cada trama (**`SeqNum`**)
- Mantiene tres variables de estado:
  - Tamaño de la ventana del tx (send window size,**`SWS`**)
  - último reconocimiento recibido (last acknowledgment received, **`LAR`**)
  - última trama enviada (last frame sent,**`LFS`**)
- Mantiene invariante: **`LFS - LAR`** <= **`SWS`**



- Avanza **`LAR`** cuando llegan ACK
- Se requiere de un buffer de hasta **`SWS`** tramas

SW: Receptor

- Mantiene tres variables de estado
  - Tamaño de la ventana receptora (receiver window size, **RWS**)
  - Trama(seqNum) mayor aceptable (largest acceptable frame, **LAF**)
  - Última trama recibida (last frame received, **LFR**)
- Mantiene invariante: **LAF - LFR** <= **RWS (<=SWS)**



- Cuando llega trama con **SeqNum**:
  - if **LFR** < **SeqNum** < = **LAF** Se acepta
  - if **SeqNum** < = **LFR** o **SeqNum** → **LAF** se descarta
- Se envía ACKs acumulativos →
- También se puede usar acuses negativos (NAK) o acuses selectivos. Dan mas información al tx.

Espacio de Números de Secuencia

- El campo **`SeqNum`** es finito; números de secuencia reaparecen (wrap around)
- El espacio de los números de secuencia debe ser mayor al numero de tramas pendientes
- **`SWS`** <= **`MaxSeqNum-1`** no es suficiente
  - supongamos campo de **`SeqNum`** de 3 bits (0..7)
  - **`SWS=RWS=7`**
  - Tx transmite tramas 0..6
  - llegan bien, pero se pierden los ACKs
  - Tx retransmite 0..6
  - Rx espera 7, 0..5, pero recibe segunda encarnación de tramas 0..5
- **`SWS`** < **`(MaxSeqNum+1)/2`** es la regla correcta
- Intuitivamente, **`SeqNum`** "se desliza" entre dos mitades del espacio de números de secuencia.

Control de perdida de paquetes y control de flujo

- El algoritmo de ventana deslizante es muy importante en redes y se puede usar para al menos tres roles:
- Recepción confiable de tramas
- Mantener el orden de transmisión de tramas.
- Control de flujo: Ajustando el SWS se puede limitar la tasa de tramas del TX. Así podemos limitar el tx para no sobrepasar la capacidad del Rx.

Canales lógicos concurrentes

- Multiplexa 8 canales lógicos sobre un único enlace
- Usa stop-and-wait en cada canal lógico
- Mantiene tres bits de estado por canal
  - Canal ocupado (busy)
  - Numero de secuencia actual de salida
  - Proximo numero de secuencia entrante
- Encabezado: 3-bit para numero de canal mas 1-bit de numero de secuencia
  - total 4-bits
  - equivalente al protocolo de ventana deslizante
- Separa *confiabilidad* de *orden*