King Fahd University of Petroleum & Minerals Computer Engineering Dept

COE 541 – Design and Analysis of Local Area Networks

Term 041

Dr. Ashraf S. Hasan Mahmoud

Rm 22-144

Ext. 1724

Email: ashraf@ccse.kfupm.edu.sa

Revision – Fourier Transform

- **A "transformation" between the time domain and the frequency domain**

Time (t)		**Frequency (f)**
s(t)	**← →**	**S(f)**

$$S(f) = \int_{-\infty}^{\infty} s(t) e^{-j2\pi ft} dt$$ Fourier Transform

$$s(t) = \int_{-\infty}^{\infty} S(f) e^{+j2\pi ft} df$$ Inverse Fourier Transform

Revision – Fourier Transform (2)

- **F.T. can be used to find the BANDWIDTH of a signal or system**
  - **Bandwidth - system: range of frequencies passed (perhaps scaled) by system**
  - **Bandwidth – signal: range of (+ve) frequencies contained in the signal**

Revision – Fourier Transform (3)

- **Remember for periodic signals (i.e. s(t) = s(t+T) where T is the period) ➔ Fourier Series expansion:**

$$s(t) = \frac{A_0}{2} + \sum_{n=1}^{\infty} \left[A_n \cos(2\pi n f_0 t) + B_n \sin(2\pi n f_0 t) \right]$$

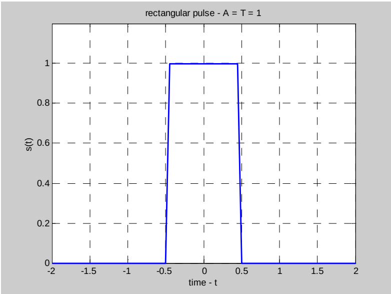
$$A_0 = \frac{2}{T}\int_0^T s(t)dt \qquad B_n = \frac{2}{T}\int_0^T s(t)\sin(2\pi n f_0 t)dt$$

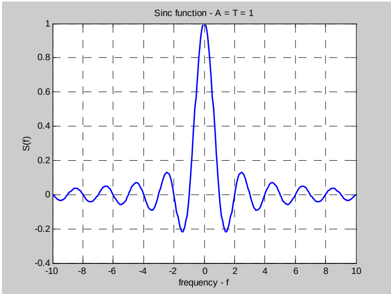
$$A_n = \frac{2}{T}\int_0^T s(t)\cos(2\pi n f_0 t)dt$$

f_0 is the fundamental frequency and is equal to 1/T

Revision – Fourier Transform (4)

- **Famous pairs – rectangular pulse (A =T = 1)**





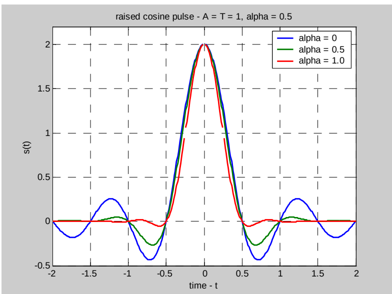
$$s(t) = \prod(t/T)$$

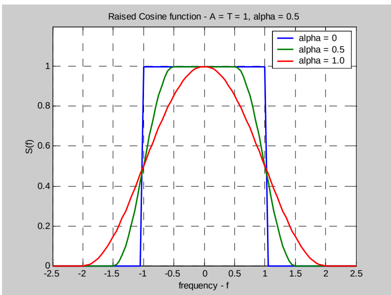
$$S(f) = AT\frac{\sin(\pi f T)}{\pi f T}$$

$S(f) = AT$ for $f = 0$
$= 0$ for $f = n/T$; n=+/- 1,2, …

Revision – Fourier Transform (5)

- **Famous pairs – Raised Cosine pulse (A =T = 1), as a function of α**





$$s(t) = \frac{(2A)}{T}\frac{\cos(2\pi\alpha t)}{1-(4\alpha t)^2}\frac{\sin(2\pi t/T)}{2\pi t/T}$$

$$S(f) = \begin{cases} A & |f| > \frac{1}{T} - \alpha \\ A\cos^2\left(\frac{\pi}{4\alpha}\left(|f| - \frac{1}{T} + \alpha\right)\right) & \frac{1}{T} - \alpha < |f| < \frac{1}{T} + \alpha \\ 0 & |f| > \frac{1}{T} + \alpha \end{cases}$$

Revision – Fourier Transform (6)

- **Raised Cosine Pulse: 0 < α < 1/T**

- **Note that s(t) = 0 for t = nT/2 where n = +/- 1,2, …**
  - **Very good for forming pulses**
  - **ZERO ISI for ideal situation**
- **BW for s(t) = 1/T + α**
  - **Maximum = 2 X 1/T (for α = 1/T)**
  - **Minimum = 1/T (for α = 0)**

Revision – Fourier Transform (7)

- **Matlab code:**

```
clear all  % clear all variables

A = 1;
T = 1;
alphas = [0 0.5 1];

for k = 1:length(alphas)
    alpha = alphas(k);

    t = -2:0.01:2;          % define the time axis
    s_t(k,:) = ((2*A)/T) * (cos(2*pi*alpha*t)./ …
               (1-(4*alpha*t).^2)) .*(sin(2*pi*t/T)./ …
               (2*pi*t/T));   % define s(t)

    f = -2.5:0.05:2.5;        % define the freq axis
    S_f(k,:) = zeros(size(f));
    i  = find(abs(f) <= (1/T-alpha));
    S_f(k,i) = A;
    i  = find((abs(f) <= (1/T+alpha)) & …
              (abs(f) > (1/T-alpha)));
    S_f(k,i) = A*(cos(pi/(4*alpha)* …
              (abs(f(i))-1/T+alpha))).^2;% define S(f)
end
```

```
figure(1);
plot(t, s_t);           % plot s(t)
title('raised cosine pulse - A = T = 1');
xlabel('time - t');
ylabel('s(t)');
legend('alpha = 0', 'alpha = 0.5', 'alpha = 1.0');
axis([-2 2 -0.5 2.2]);
grid

figure(2);
plot(f, S_f);           % plot S(f)
title('Raised Cosine function - A = T = 1');
xlabel('frequency - f');
ylabel('S(f)');
legend('alpha = 0', 'alpha = 0.5', 'alpha = 1.0');
axis([-2.5 2.5 0 1.2]);
grid
```

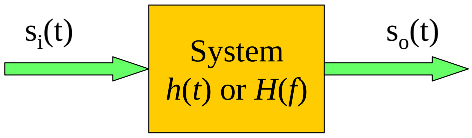
Signals and Systems

- **For linear Systems:**
  - **h(t) is the system's impulse response – i.e. $s_o(t) = h(t)$ when $s_i(t) = \delta(t)$**
  - **$S_i(t)$ is system input signal**
  - **$S_o(t)$ is system output signal**

$$s_0(t) = \int_{-\infty}^{\infty} s_i(\tau) h(t-\tau) d\tau$$

$$s_0(t) = s_i(t) * h(t)$$

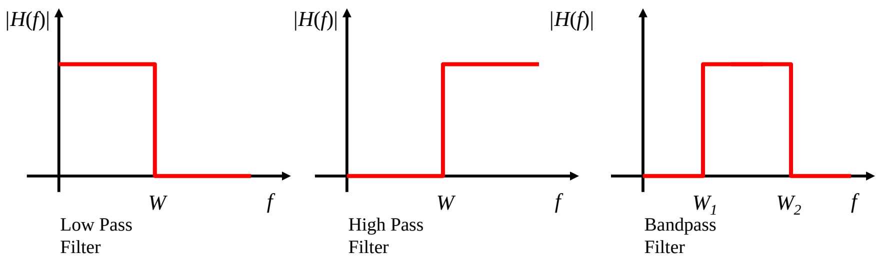
$$S_0(f) = S_i(f) H(f)$$



A good introduction into linear systems is found at
http://www.ece.utexas.edu/~bevans/courses/ee313/lectures/04_Convolution/lecture4.pdf

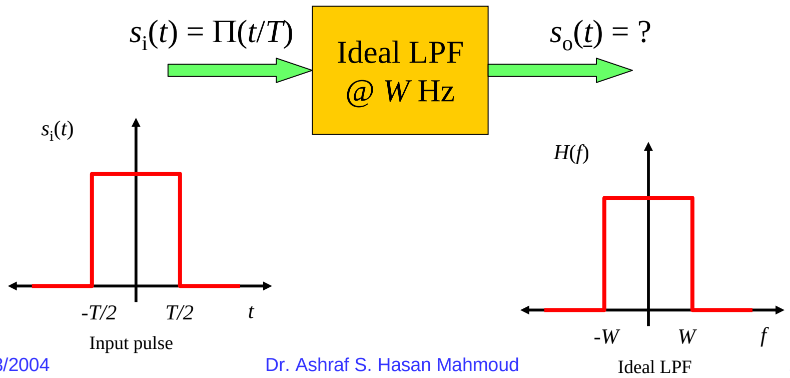
Signals and Systems (2)

- **System bandwidth is determined by examining the Fourier transfer of the system function h(t), H(f)**
- **Example (transmission) systems:**



Signals and Systems - Example

- **Ideal Low Pass Filter – find the output signal for rectangular input pulse?**



Signals and Systems - Example

- **The input signal $s_i(t)$ is given by:**

$$s_i(t) = \begin{cases} A & |t| <= T/2 \\ 0 & \text{otherwise} \end{cases}$$

- **Where as it Fourier transform $S_i(f)$ is given by (note that $s_i(t)$ contains all frequencies from 0 till ∞ - refer to Fourier transform of rectangular pulse):**

$$S_i(f) = AT \frac{\sin(\pi f T)}{\pi f T} \quad \text{for all } f$$

- **The Fourier transform of the system impulse response, $H(f)$ is given by (note this transmission system limits frequencies to at most W Hz):**

$$H(f) = \begin{cases} 1 & |f| <= W \\ 0 & \text{otherwise} \end{cases}$$

Signals and Systems - Example

- **Therefore the Fourier transform of the output signal is given by:**
- $|S_o(f)| = |S_i(f)| \times |H(f)|$

$$= AT\frac{\sin(\pi f T)}{\pi f T} \quad \text{for } |f| < W$$

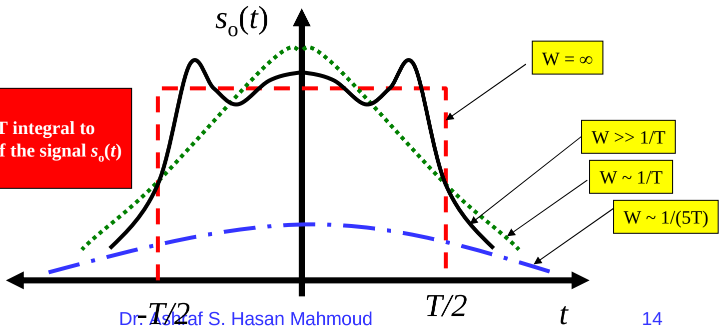
$$= 0 \quad \text{otherwise}$$

(**note the output signal has frequencies up to W Hz only**)

Signals and Systems - Example

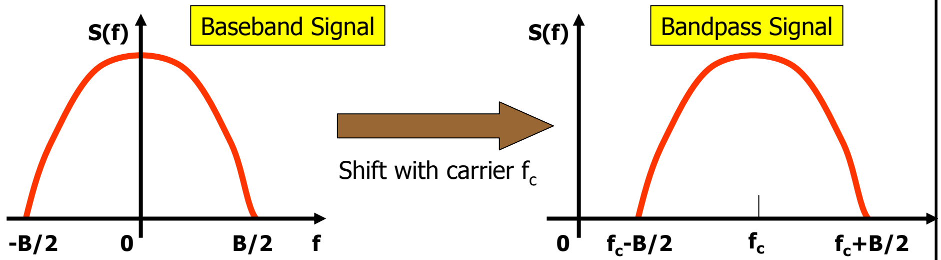
- **To find the output signal $s_o(t)$, one has to use the inverse Fourier transform on $S_o(f)$**
- **As the BW of the system is increased, the output signal approaches a rectangular pulse (copy of input)**

Exercise: Try to apply the inverse F.T integral to obtain an analytical form of the signal $s_o(t)$ as a function of W and T.



Baseband vs. Bandband

- **Baseband Signal:**
  - **Spectrum not centered around non zero frequency**
  - **May have a DC component**
- **Bandpass Signal:**
  - **Does not have a DC component**
  - **Finite bandwidth around or at f_c**



Modulation

- **Is used to shift the frequency content of a baseband signal**
  - **Basis for AM modulation**
  - **Basis for Frequency Division Multiplexing (FDM)**

Analog Communications

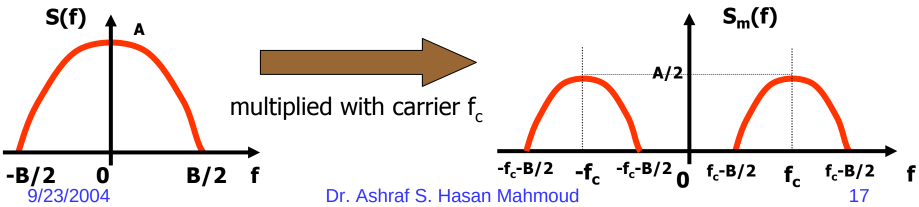
Modulation

- **Consider the signal s(t),**

$$s_m(t) = s(t) \times \cos(2\pi f t)$$

The spectrum for $s_m(t)$ is given by

$$S_m(f) = ½ \times \{S(f-f_c) + S(f+f_c)\}$$



Analog Communications

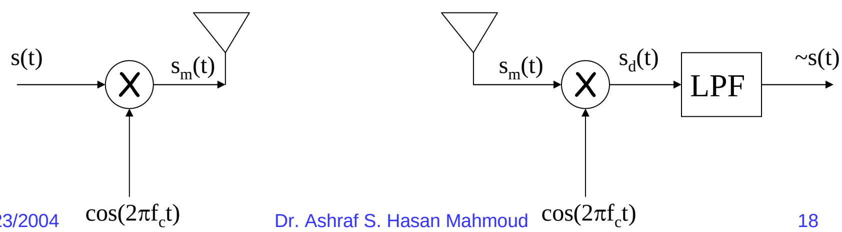
Modulation – Txer/Rxer

- **At the receiver side:**

$$s_d(t) = s_m(t) \times \cos(2\pi f_c t)$$
$$= s(t) \times \cos(2\pi f_c t) \times \cos(2\pi f_c t)$$
$$= ½\, s(t) + ½\, s(t) \times \cos(2\pi 2 \times f_c t)$$

desired term

undesired term – signal centered around $2f_c$ filtered out using the LPF



Digital Communications

Nyquist Bandwidth

- **For a noiseless channels of bandwidth B, the maximum attainable bit rate (or capacity) is given by**

$$C = 2B \log_2(M)$$

Where M is the size of the signaling set

Digital Communications

Shannon Capacity

- **Capacity of a channel of bandwidth B, in the presence of noise is given by**

$$C = B \log_2(1 + SNR)$$

where SNR is the ratio of signal power to noise power – a measure of the signal quality

Digital Communications

Example: Shannon Capacity

- **Consider a GSM system with BW = 200 kHz. If SNR is equal to 15 dB, find the channel capacity?**
- **Solution:**

SNR = 15 dB = $10^{(15/10)}$ = 31.6

C = $200 \times 10^3 \times \log_2(1+31.6)$

= 1005.6 kb/s

Note GSM operates at 273 kb/s which is ~27% of maximum capacity at SNR = 30 dB.

Digital Communications

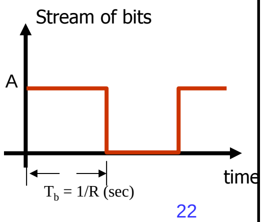
Eb/No Expression

- **An alternative representation of SNR**
- **Consider the bit stream shown in figure – for bit of rate R, then each bit duration is equal to T_b = 1/R seconds**
- **Energy of signal for the bit duration is equal to $A^2 \times T_b$, where its power is equal to bit energy / T_b or A^2.**
- **Noise power is equal to $N_0 \times B$ (refer to thermal noise section)**
- **Hence, SNR is given by signal power / noise power or**

$$SNR = \frac{signalpower}{N_0 B} = \frac{E_b}{N_0} \times \frac{R}{B}$$

- **One can also write**

$$\left(\frac{E_b}{N_0}\right)_{dB} = SignalPower(dBW) - 10\log R - 10\log k - 10\log T$$



Signal Elements or Pulses

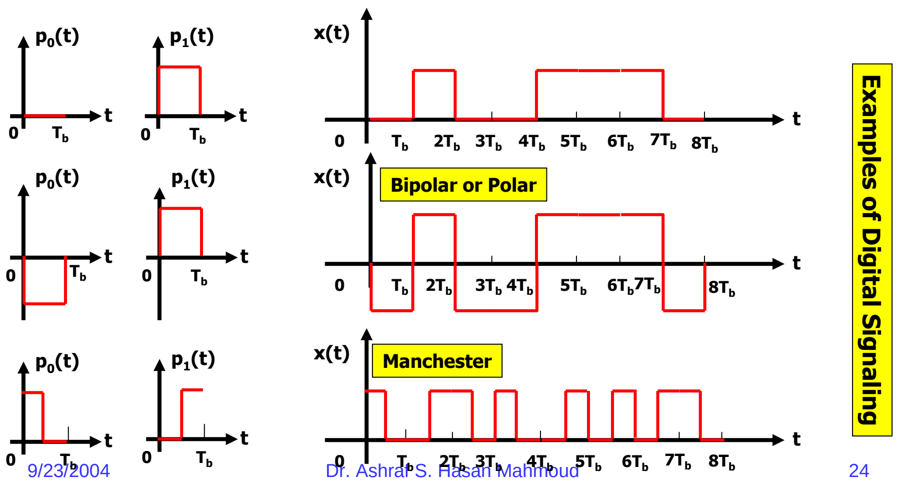
- **Unit of transmission – repeated to form the overall signal**
- ***Shape* of pulse determines the bandwidth of the transmitted signal**
- **Digital data is mapped or encoded to the different pulses or units of transmission**

- **Baud/Modulation or Symbol Rate (R_s)**
  - **The bit rate $R_b = R_s \log_2(M)$**

- **Please refer to earlier examples of pulses and the corresponding BW**

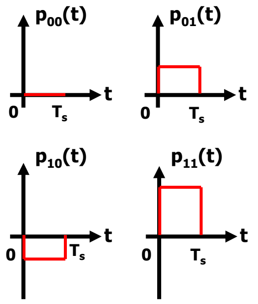
Signal Elements or Pulses

Definitions of Pulses **Encoded Signal: 0 1 0 0 1 1 1 0**

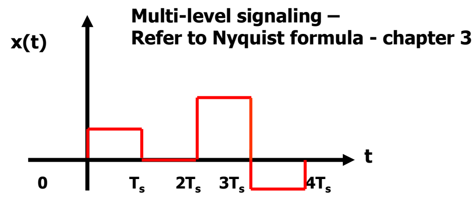


Signal Elements or Pulses

Pluses Definitions



Encoded Signal: 0 1 0 0 1 1 1 0

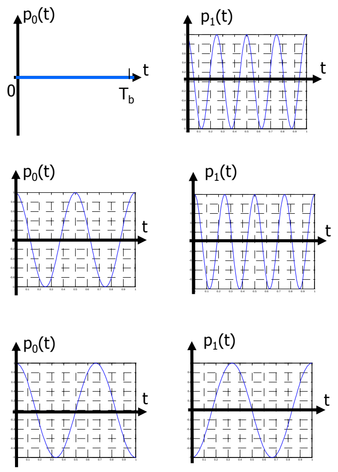


- Note that each symbol or pulse caries 2 bits
- Symbol duration is $T_s = 2T_b$
- Bit rate R equal to $1/T_b$
- Symbol rate or *baud rate* R_s equal to $1/T_s$ ➔ $R = 2R_s$
- In general to encode n bits per pulse, you need 2^n pulses

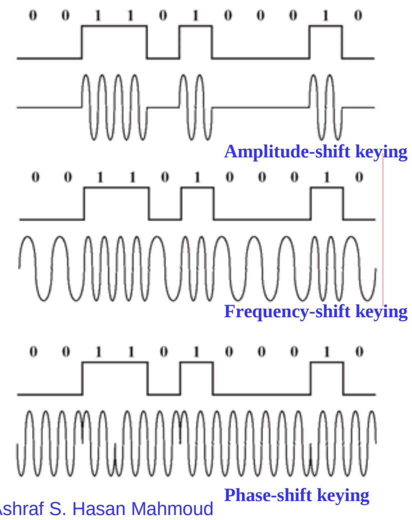
Example of Digital Signaling

Signal Elements or Pulses

Definitions of Pulses



Encoded Signal:

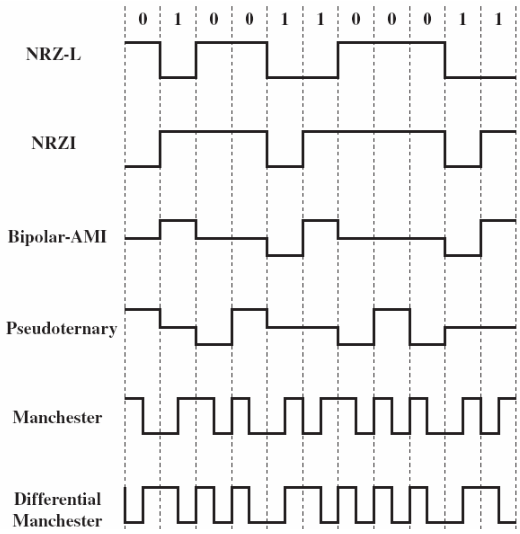


Example of Analog Signaling

Digital Signal Encoding Formats

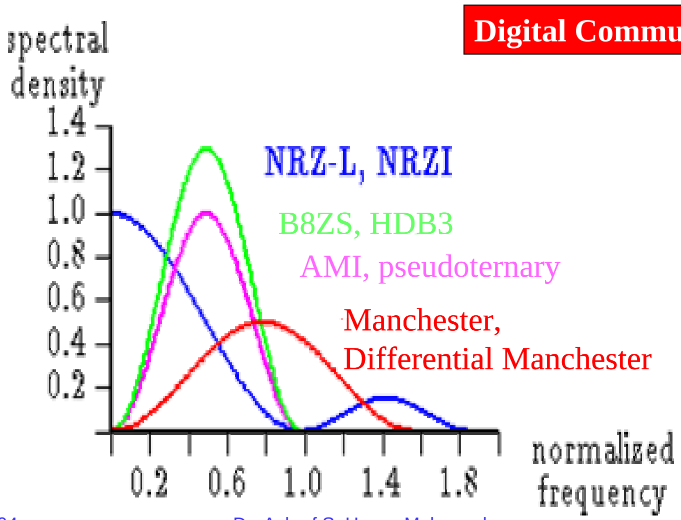
- **Nonreturn to Zero-Level (NRZ-L)**
  - **0 = high level**
  - **1 = low level**
- **Nonreturn to Zero Inverted (NRZI)**
  - **0 = no transition at beginning of interval**
  - **1 = transition at beginning of interval**
- **Bipolar-AMI**
  - **0 = no line signal**
  - **1 = +ve or –ve level; alternating successive ones**
- **Pseudoternary**
  - **0 = +ve or –ve level; alternating for successive ones**
  - **1 = no line signal**
- **Doubinary**
  - **0 = no line signal**
  - **1 = +ve or –ve level; depending on number of separating 0s (even – same polarity, odd – opposite polarity)**
- **Manchester**
  - **0 = transition from high to low in middle of interval**
  - **1 = transition from low to high in middle of interval**
- **Differential Manchester: Always transition in middle of interval**
  - **0 = transition at beginning of interval**
  - **1 = no transition at beginning of interval**

Digital Signal Encoding Formats



Spectrum Characteristics of Digital Encoding Schemes

Digital Communications



Asynchronous Data Transmission

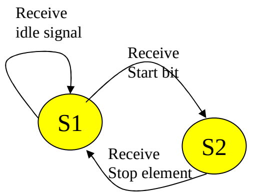
- **Digital Info:**
  - **Bits**
  - **Characters**
  - **Packets**
  - **Messages or files**
- **Serial vs. Parallel character**

Asynchronous Transmission

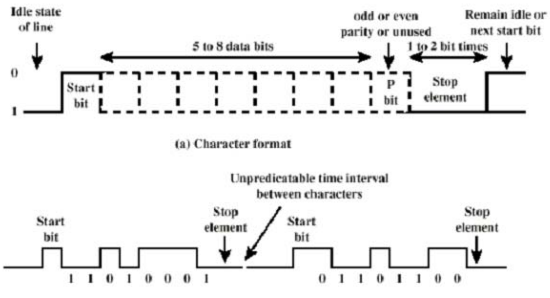
- Exploits: Rx-er can remain for short period in synch with Tx-er
- Used for short stream of bits – data transmitted one character (5 ~ 8 bits) at a time
- Synchronization is needed to be maintained for the length of short transmission
- Character is delimited (start & end) by known signal elements: start bit – stop element
- Rx-er re-synchs with the arrival of new character

Asynchronous Transmission

- Simple / Cheap
- Efficiency: transmit 1 start bit + 8 bit of data +2 stop bits ➔ Efficiency = 8/11 = 72% (or overhead = 3/11 = 28%)
- Good for data with large gaps (e.g. keyboard, etc)



S1: receiver in idle state
S2: receive is receiving character



(a) Character format

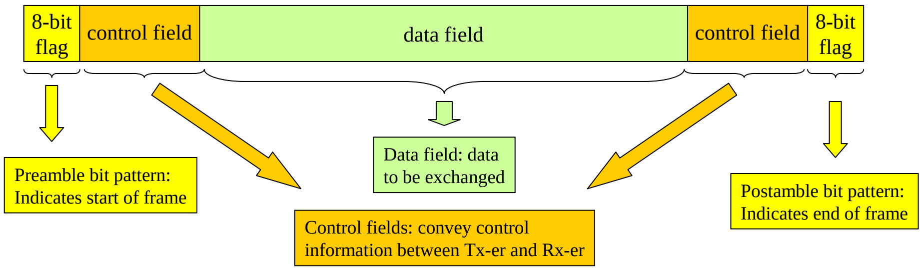
(b) 8-bit asynchronous character stream

Synchronous Data Transmission

- **Two ends remain in sync for significant period of time**
- **Use of SYNC or PREAMPLE characters**
- **Noise + Data = may create another SYNC character ➔ frame split**
  - **Solution – use two SYNC characters**
- **Rx-er must buffer incoming frames and search for SYNC character(s)**

Synchronous Frame Format

- Typical Frame Structure

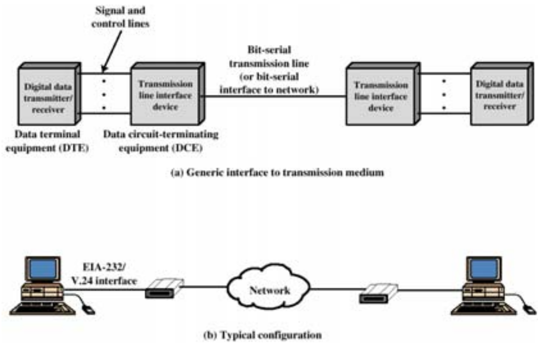


- For large data blocks, synchronous transmission is far more efficient than asynchronous:
  - E.g. HDLC frame (to be discussed in Chapter 7): 48 bits are used for control, preamble, and postamble – if 1000 bits are used for data ➔ efficiency = 99.4% (or overhead = 0.6%)

For more details refer to Chapter 6: The Data Communication Interface - Data and Computer Communications, Stallings 6th Edition, 2000

Interfacing

- Data Terminal Equipment (DTE): terminals or computers
- Data Circuit Equipment (DCE): modem
- Two DCEs exchanging data on behalf of DTEs must use exact same protocol



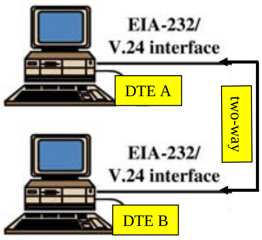
For more details refer to Chapter 6: The Data Communication Interface - Data and Computer Communications, Stallings 6th Edition, 2000

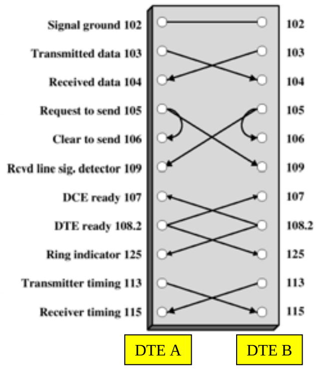
DTE-DCE Interface Definition

- Mechanical: physical specification of connection – type, dimensions, location of pins, etc
- Electrical: voltage levels and timing signals used
- Functional: specify functions that are performed for circuits – rx circuit, tx circuit, etc.
- Procedural: specification of sequence of event for transmitting data based on functional specification
- Two examples:
  - V.24/EIA-232-F, and
  - ISDN physical interface

V.24/EIA-232-F - Procedural Specification – Examples

- <u>Example:</u> Two terminals connected back-to-back through the V.24 interface BUT with no DCEs
- This is referred to as the NULL modem connection
- For short distance connections





Error Control

- Error Detection
  - Parity Checks
  - Cyclic Redundancy Check (CRC)

- For a channel of bit error rate (or BER) of P, the probability of m bits in error in a block of n bits (m <= n) is given by

$$\binom{n}{m} p^m (1-p)^{n-m}$$

or

$$\frac{n!}{(n-m)!\,m!} p^m (1-p)^{n-m}$$

The above simple formula assumes iid error probability across all bits in block – How realistic is that?

Error Control (2)

- The probability the frame or block is correct is given by

$$(1-p)^n$$

Therefore the probability, the frame is in error (one or more bits in error) is given by

$$1-(1-p)^n$$

The above quantity is referred to as FER

Error Control - Example

- Consider a channel with BER = 10^{-3}, for a block (packet of n = 100 bits), the probability of having one bit in error is equal to

$$100 \text{ X } P \text{ X } (1-P)^{99} = 9 \text{ X } 10^{-2}$$

While the probability of having 4 bits in error is equal to

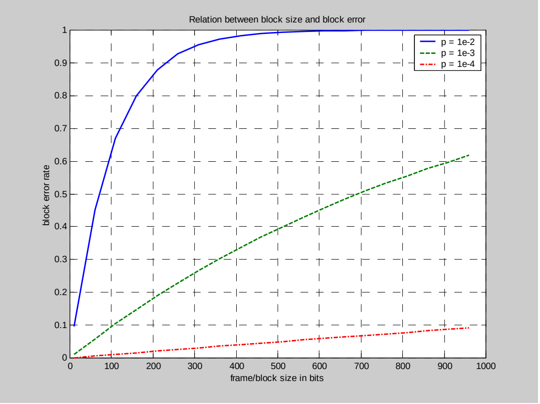
$$(100 \text{ choose } 4) \text{ X } P^4 \text{ X } (1-P)^{96} = 3.6\text{X}10^{-6}$$

The probability that the frame is correct is equal to

$(1-P)^{100} = 0.905$ ➔ i.e. ~10% of the time the frame is in error!!

Error Control – Example (2)

- Relation between block size (n) and frame error rate (FER)



Simple Parity Check

- Add one extra bit for each character such that:
  - Even Parity: no of 1s even
  - Odd Parity: no of 1s odd
- Simple
- Can not detect even no of errors in character
- Adding one extra bit to a group of n bits ➔ Excess redundancy = 1/(n+1)

VRC/LRC Parity Check

- Extension of simple parity: Vertical Redundancy Check (VRC) and Longitudinal Redundancy Check (LRC)

Original data to send

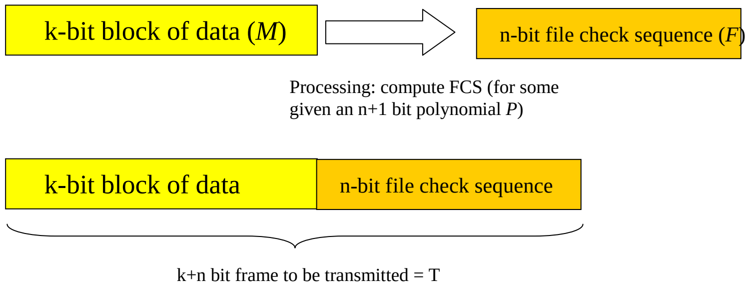
Parity check

Char 1	1	0	0	1	1	0	0	0	1
Char 2	0	1	1	1	0	1	0	1	1
Char 3	1	1	0	0	1	1	0	0	0
Char 4	1	0	0	0	1	0	0	0	0
Char 5	0	1	0	0	1	1	1	0	0
Checking char	1	1	1	0	0	1	1	1	0

VRC/LRC Parity Check (2)

- Can detect all odd errors – same as the simple parity check
- Can detect any combination of even error in characters that DO NOT result in even number of errors in a column
- Excess Redundancy: 14/(40+14) = 0.26
- There could be undetected errors – How?

Cyclic Redundancy Check (CRC)



- Modulo 2 arithmetic is used to generate the FCS:
  - 0 ± 0 = 0; 1 ± 0 = 1; 0 ± 1 = 1; 1 ± 1 = 0
  - 1 X 0 = 0; 0 X 1 = 0; 1 X 1 = 1

CRC – Mapping Binary Bits into Polynomials

- Consider the following k-bit word or frame and its polynomial equivalent:

$$b_{k-1}\, b_{k-2} \ldots b_2\, b_1\, b_0 \Rightarrow b_{k-1}x^{k-1} + b_{k-2}x^{k-2} + \ldots + b_1x^1 + b_0$$

where b_i ($k-1 \leq i \leq 0$) is either 1 or 0

CRC – Mapping Binary Bits into Polynomials- Examples

- Example1: an 8 bit word M = 11011001 is represented as $M(x) = x^7+x^6+x^4+x^3+1$

- Example2: What is $x^4M(x)$ equal to?

  $x^4M(x) = x^4(x^7+x^6+x^4+x^3+1) = x^{11}+x^{10}+x^8+x^7+x^4$, the equivalent bit pattern is 110110010000 (i.e. four zeros added to the right of the original M pattern)

- Example3: What is $x^4M(x) + (x^3+x+1)$?

  $x^4M(x) + (x^3+x+1) = x^{11}+x^{10}+x^8+x^7+x^4+ x^3+x+1$, the equivalent bit pattern is 110110011011 (i.e. pattern 1011 = x^3+x+1 added to the right of the original M pattern)

CRC Calculation

- T = (k+n)-bit frame to be tx-ed, n < k
- M = k-bit message, the first k bits of frame T
- F = n-bit FCS, the last n bits of frame T
- P = pattern of n+1 bits (a predetermined divisor)

T = (n+k)-bit frame

Note:
-T(x) is the polynomial (of k+n-1st degree or less) representation of frame T
-M(x) is the polynomial (of k-1st degree or less) representation of message M
- F(x) is the polynomial (of n-1st degree or less) representation of FCS
- P(x) is the polynomial (of nth degree or less) representation of the divisor P
-$T(x) = X^n M(x) + F(x)$ – refer to example 3 on previous slide

CRC Calculation (2)

- Design: frame T such that it divides the pattern P with no remainder?
- Solution: Since the first component of T, M, is the data part, it is required to find F (or the FCS) such that T divides P with no remainder

Using the polynomial equivalent:
$T(x) = X^n M(x) + F(x)$
One can show that $F(x)$ = remainder of $x^n M(x) / P(x)$
i.e if $x^n M(x) / P(x)$ is equal to $Q(x) + R(x)/P(x)$, then $F(X)$ is set to be equal to $R(X)$.

Note that:

$$\frac{\text{Polynomial of degree } k+n}{\text{Polynomial of degree } n} = \text{polynomial of degree } k + \text{remainder polynomial of degree } n \text{ or less}$$

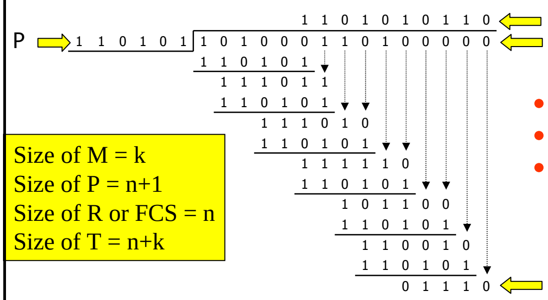
CRC Calculation - Procedure

1. **Shift pattern M n bits to the lift**
2. **Divide the new pattern $2^n M$ by the pattern P**
3. **The remainder of the division R (n bits) is set to be the FCS**
4. **The desired frame T is $2^n M$ plus the FCS bits**

Note:
$2^n M$ is the pattern resulting from shifting the pattern M n bits to the left. In other words, the polynomial equivalent of the pattern $2^n M$ is $x^n M(x)$

CRC Calculation – Example

- **Message M = 1010001101 (10 bits) ➔ k = 10**
  Pattern P = 110101 (6 bits – note 0th and nth bits are 1s)
  ➔n + 1 = 6 ➔ n = 5
  Find the frame T to be transmitted?
- **Solution:**

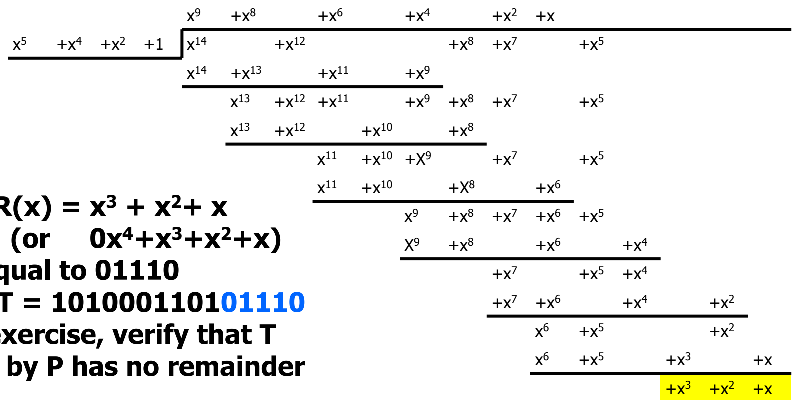


Size of M = k
Size of P = n+1
Size of R or FCS = n
Size of T = n+k

- **FCS = R is equal to 01110**
- **Frame T = 101000110101110**
- **As an exercise, verify that T divided by P has no remainder**

CRC Calculation – The previous example BUT using Polynomials

- **Message M = 1010001101 (10 bits) ➔ k = 10**
- **$M(x) = x^9 + x^7 + x^3 + x^2 + 1$ ➔ $x^5M(x) = x^{14} + x^{12} + x^8 + x^7 + x^5$**
  Pattern P = 110101 (6 bits – note 0th and nth bits are 1s) ➔n + 1 = 6 ➔ n = 5
- **$P(x) = x^5 + x^4 + x^2 + 1$**
  Find the frame T to be transmitted?
- **Solution:**



- **FCS = $R(x) = x^3 + x^2 + x$**
  (or $0x^4+x^3+x^2+x$)

➔ R is equal to 01110

- **Frame T = 1010001101**01110**
- **As an exercise, verify that T divided by P has no remainder**

CRC Calculation – The previous example BUT using Polynomials – cont'd

- Message M = 1010001101 (10 bits)
- ➔ M(x) = $x^9 + x^7 + x^3 + x^2 + 1$
- ➔ ➔ $x^5M(x) = x^{14} + x^{12} + x^8 + x^7 + x^5$
- Pattern P = 110101

➔ $P(x) = x^5 + x^4 + x^2 + 1$

- $R(x) = x^3 + x^2 + x$
- $Q(x) = x^9+x^8+x^6+x^4+x^2+x$
- $T(X) = x^5M(x) + R(x)$

  $= x^{14} + x^{12} + x^8 + x^7 + x^5 + x^3 + x^2 + x$, or

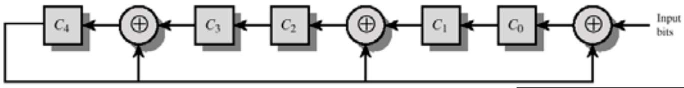
  T = 101000110101110

- <u>Exercise</u>: Verify that $Q(x) P(x) + R(x) = x^5 M(x)$

CRC – Receiver Procedure

- Tx-er transmits frame T
- Channel introduces error pattern E
- Rx-er receives frame $T_r = T \oplus E$ (note that if E = 000..000, then Tr is equal to T, i.e. error free transmission)
- T_r is divided by P, Remainder of division is R
- if R is ZERO, Rx-er assumes no errors in frame; else Rx-er assumes erroneous frame
- If an error occurs and T_r is still divisible by P ➔ UNDETECTABLE error (this means the E is also divisible by P)

CRC – Transmitter Circuit



▢ = 1-bit shift register ⊕ = Exclusive-OR circuit

(a) Shift-register implementation

Shift register circuit for dividing by
P = $X^5+X^4+X^2+1$

	C_4	C_3	C_2	C_1	C_0	$C_4 \oplus C_3$	$C_4 \oplus C_1$	$C_4 \oplus$ input	input
Initial	0	0	0	0	0	0	0	1	1
Step1	0	0	0	0	1	0	0	0	0
Step2	0	0	0	1	0	0	1	1	1
Step3	0	0	1	0	1	0	0	0	0
Step4	0	1	0	1	0	1	1	0	0
Step5	1	0	1	0	0	1	1	1	0
Step6	1	1	1	0	1	0	1	0	1
Step7	0	1	1	1	0	1	1	1	1
Step8	1	1	1	0	1	0	1	1	0
Step9	0	1	1	1	1	1	1	1	1
Step10	1	1	1	1	1	0	0	1	0
Step11	0	1	0	1	1	1	1	0	0
Step12	1	0	1	1	0	1	0	1	0
Step13	1	1	0	0	1	0	1	1	0
Step14	0	0	1	1	1	0	1	0	0
Step15	0	1	1	1	0	1	1	0	—

Initial–Step9: Message to be sent

Step10–Step14: Five zeros added

(b) Example with input of 1010001101

CRC – Receiver Circuit

- **Tx-er transmits frame T**

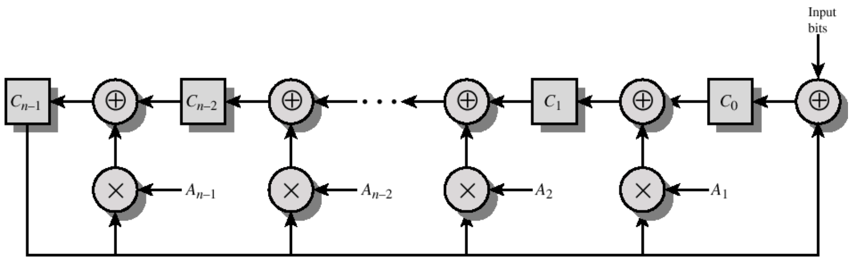
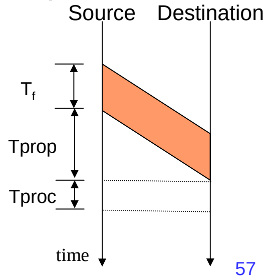


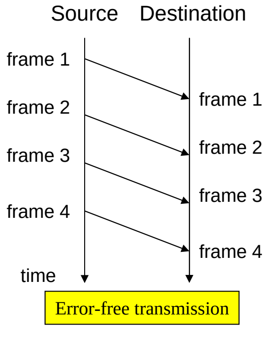
Figure 7.7 General CRC Architecture to Implement Divisor
$1 + A_1X + A_2X^2 + \ldots + A_{n-1}X^{n-1} + X^n$

Flow Control

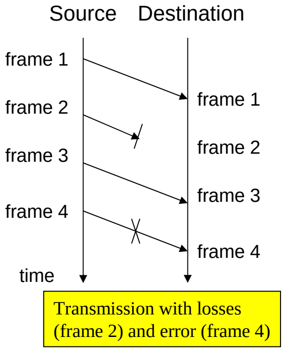
- A scheme to ensure that transmitter does not overwhelm receiver with data
- Transmission of one frame:
  - T_f: time to transmit frame
  - Tprop: time for signal to propagate
  - Tproc: time for destination to process received frame – small delay (usually ignored if not specified)
- Tproc may be ignored if not specified



Flow Control (2)



Models of Frame Transmission



- The destination has a limited buffer space. How will the source know that destination is ready to receive the next frame?
- In case of errors or lost frame, the source need to retransmit frames – i.e. a copy of transmitted frames must be kept. How will the source know when to discard copies of old frames?
- Etc.

Stop-and-Wait Protocol

- Protocol:
  - Source transmits a frame
  - After the destination receives frame, it sends ACK
  - Source, upon the receipt of ACK, can now send the next frame
- Destination can stop source by withholding the ACK
- Simple
- Animation for Stop-and-Wait
- NOTE: ONLY one frame can be in transit at any time

Stop-and-Wait Protocol: Efficiency

- After every frame, source must wait till acknowledgment ➔ Hence link propagation time is significant
- Total time to for one frame:

  T_total = Tf + 2Tprop + Tproc + Tack

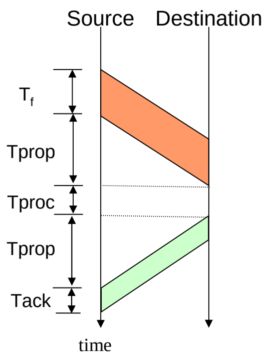
  if we ignore Tproc and Tack (usually very small)

  T_total = Tf + 2Tprop
- Link utilization, U is equal to

  U = Tf/ (T_total), or

  = 1 / (1+2(Tprop/Tf)) = 1 / (1 + 2 a)

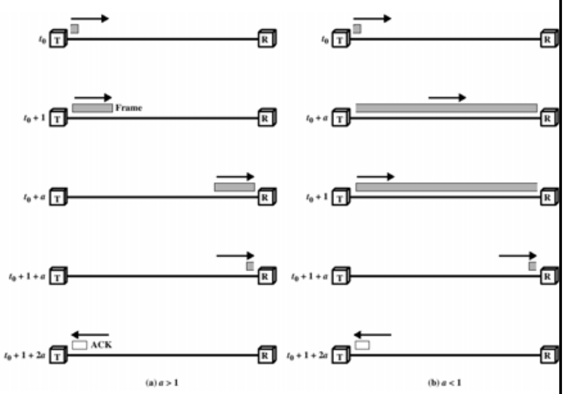
  where a = Tprop/Tf = length of link in bits
- If a < 1 (i.e. Tf > Tprop – when 1st transmitted bit reaches destination, source will still be transmitting ➔ U is close 100%
- If a > 1 (i.e. Tf < Tprop – frame transmission is completed before 1st bit reaches destination ➔U is low
- See figure 7.2



Stop-and-Wait Protocol: Efficiency (2)

- Remember: a = Tprop/Tf = length of link in bits

- If a < 1 (i.e. Tf > Tprop – when 1st transmitted bit reaches destination, source will still be transmitting ➔ U is close 100%
- If a > 1 (i.e. Tf < Tprop – frame transmission is completed before 1st bit reaches destination ➔U is low
- Stop-and-Wait is efficient for links where a << 1 (long frames compared to propagation time)



Sliding Window Protocol

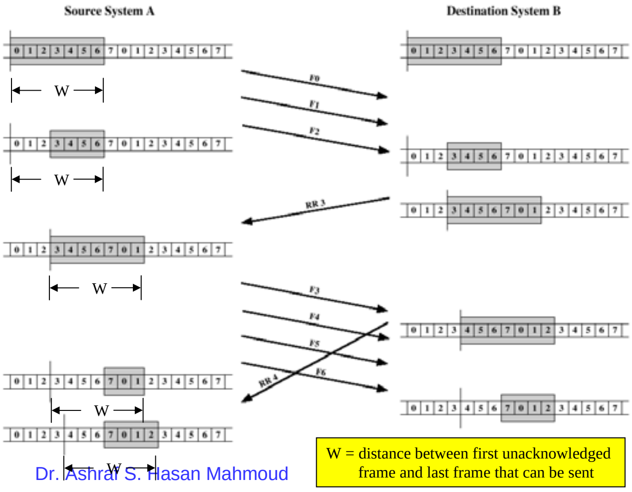
- Stop-and-Wait can be very inefficient when a > 1
- Protocol:
  - Assumes full duplex line
  - Source A and Destination B have buffers each of size W frames
  - For k-bit sequence numbers:
    - Frames are numbered: 0, 1, 2, …, 2^k-1, 0, 1, … (modulo 2^k)
    - ACKs (RRs) are numbered: 0, 1, 2, …, 2^k-1, 0, 1, … (modulo 2^k)
  - A is allowed to transmit up to W frames without waiting for an ACK
  - B can receive up to W consecutive frames
  - ACK J (or RR J), where 0<=J<= 2^k-1, sent by B means B is have received frames up to frame J-1 and is ready to receive frame J
  - B can also send RNR J: B have received all frames up to J-1 and is not ready to receive any more
- Window size, W can be less or equal to 2^k-1

Sliding Window Protocol (2)

- Example of Sliding-Window-Protocol: k = 3 bits, W = 7

Observations:

- A may tx W = 7 frames (F0, F1, …, F6)
- After F0, F1, & F2 are tx-ed, window is shrunk (i.e. can not transmit except F3, F4, …, F6)
- When B sends RR3, A knows F0, F1 & F2 have been received and B is ready to receive F3
- Window is advanced to cover 7 frames (starting with F3 up to F1)
- A sends F3, F4, F5, & F6
- B responds with RR4 when F3 is received – A advances the window by one position to include F2



W = distance between first unacknowledged frame and last frame that can be sent

Sliding Window Protocol - Piggybacking

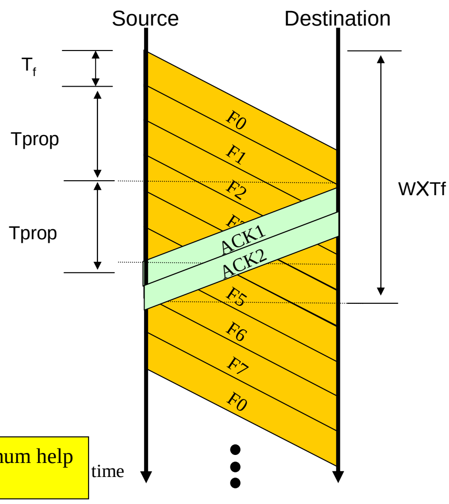
- When using sliding window protocol in full duplex connections:
  - Node A maintains its own transmit window
  - Node B maintains its own transmit window
  - A frame contains: data field + ACK field
  - There is a sequence number for the data field, and a sequence number for the ACK field

Sliding Window Protocol - Efficiency

- Again we can distinguish two cases:
- Case 1: W ≥ 2a + 1
- Case 2: W < 2a + 1

Sliding Window Protocol - Efficiency - Case 1

- Assume k=3, W = 7
  (ignoring Tack)
- Source can continuously keep transmitting!!
  - Because the ACK can arrive to source before the window is completed
- Utilization = 100%



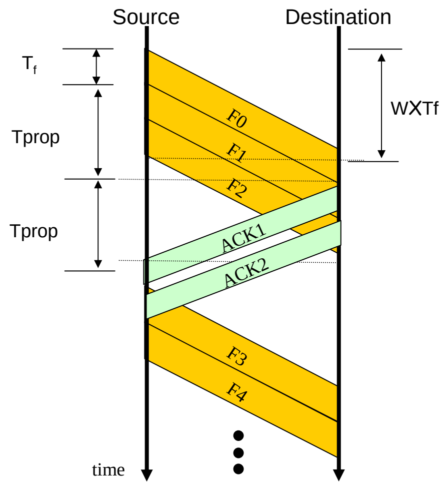
Sending ACK0 as soon as F0 is received is the maximum help the destination can do to increase utilization

Sliding Window Protocol - Efficiency - Case 2

- Assume k = 3, W = 3 (ignoring Tack)
- Source can NOT continuously keep transmitting!!
  - Because the ACK can NOT arrive to source before the window is completed

- Utilization = $\frac{W \times T_f}{T_f + 2 \times T_{prop}}$

  = $\frac{W}{1 + 2a}$



Sliding Window Protocol - Efficiency

- Refer to Appendix A
- When window size is W (for error free), link utilization, U, is given by

$$U = \begin{cases} 1 & W \geq (2a+1) \\ \dfrac{W}{2a+1} & W < (2a+1) \end{cases}$$

where a = Tprop/Tf or length of link in bits

- Sliding window protocol can achieve 100% utilization if W >= (2a + 1)

Sliding Window Protocol

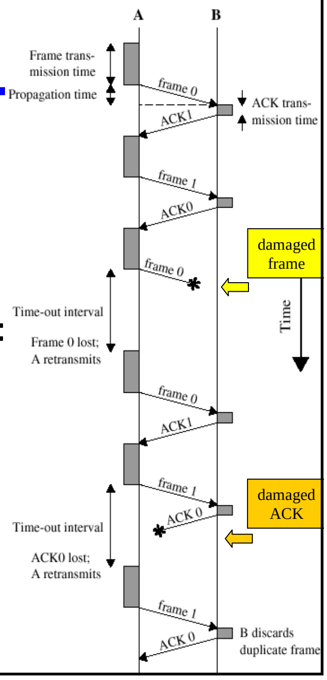
- Animation for Sliding Window protocol
- Sliding Window Protocol Simulation (http://www.cs.stir.ac.uk/~kjt/software/comms/jasper/SWP3.html)

Automatic Repeat Request - ARQ

- Types of Errors:
  - Lost frame
  - Damaged frame
- Error control Techniques:
  - Error detection – discussed previously
  - +ve ACK
  - Retransmission after timeout
  - -ve ACK and retransmission
- ARQ Procedures: convert an unreliable data link into a reliable one.
  - Stop-and-wait
  - Go-back-N
  - Selective-reject

Stop-and-Wait ARQ

- Based on the stop-and-wait control flow procedure
- Two types of errors:
  1. Frame lost or damaged – *Solution*: timeout timer
  2. Damaged or lost ACK – The timeout timer solves this problem



Sliding Window Protocol

- Stop-and-Wait can be very inefficient when a > 1
- Protocol:
  - Assumes full duplex line
  - Source A and Destination B have buffers each of size W frames
  - For k-bit sequence numbers:
    - Frames are numbered: 0, 1, 2, …, 2^k-1, 0, 1, … (modulo 2^k)
    - ACKs (RRs) are numbered: 0, 1, 2, …, 2^k-1, 0, 1, … (modulo 2^k)
  - A is allowed to transmit up to W frames without waiting for an ACK
  - B can receive up to W consecutive frames
  - ACK J (or RR J), where 0<=J<= 2^k-1, sent by B means B is have received frames up to frame J-1 and is ready to receive frame J
  - B can also send RNR J: B have received all frames up to J-1 and is not ready to receive any more
- Window size, W can be less or equal to 2^k-1

Go-Back-N ARQ

- Based on the sliding-window flow control procedure
- If the i^{th} frame is lost or deemed lost (i.e. $i+1^{st}$ is received before the i^{th} frame), the i^{th} frame and all subsequent frames are retransmitted

Selective-Reject ARQ

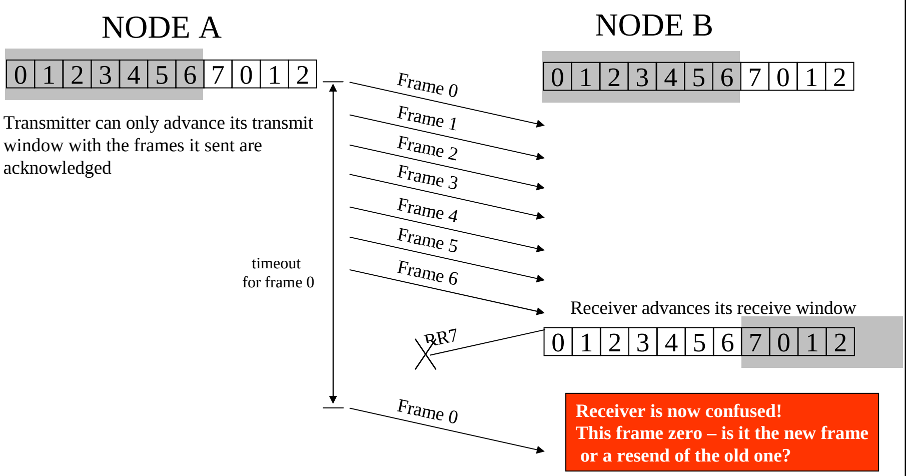
- In contrast to Go-Back-N, the only frames retransmitted are those that receive –ve ACK (called SREJ) or those that time out
- More efficient:
  - Rx-er must have large enough buffer to save *post*-SREJ frames
  - Buffer manipulation – re-insertion of out-of-order frames

Window Size for Selective-Reject ARQ – Why?

- Window size: should less or equal to half range of sequence numbers
  - For n-bit sequence numbers, Window size is $\leq 2^{n-1}$ (remember sequence numbers range from 0,1, …, 2^n-1)
- Why? See next example

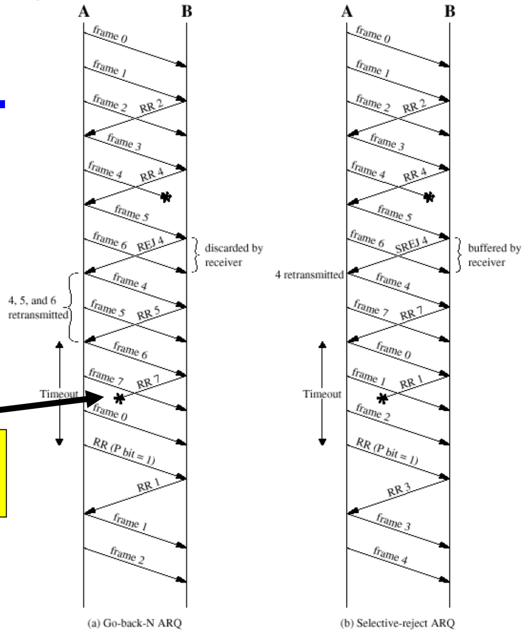
Window Size for Selective-Reject ARQ – Why? (2)

- Example: Consider 3-bit sequence number and window size of 7



Go-Back-N/Selective-Reject ARQ Examples

- With Go-back-N frames 4,5 and 6 are retransmitted
- With Selective-Reject only frame 4 is retransmitted



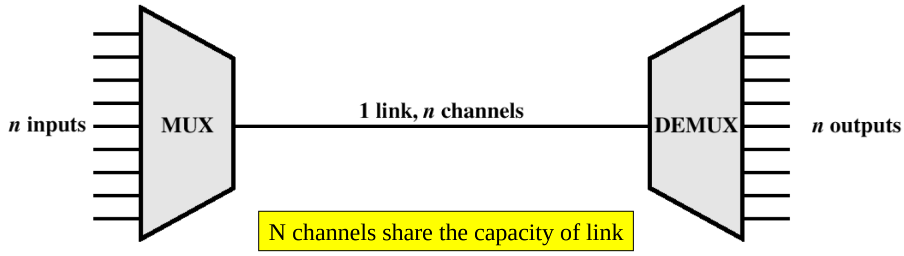
Did this lost RR7 affect flow?
How did the link recover?

Switching

- Circuit Switching
  - Call Setup
  - Data Exchange
  - Call Termination
- Store-and-forward (Packet Switching)
  - Virtual Circuit
  - Datagram

What is MULTIPLEXING?

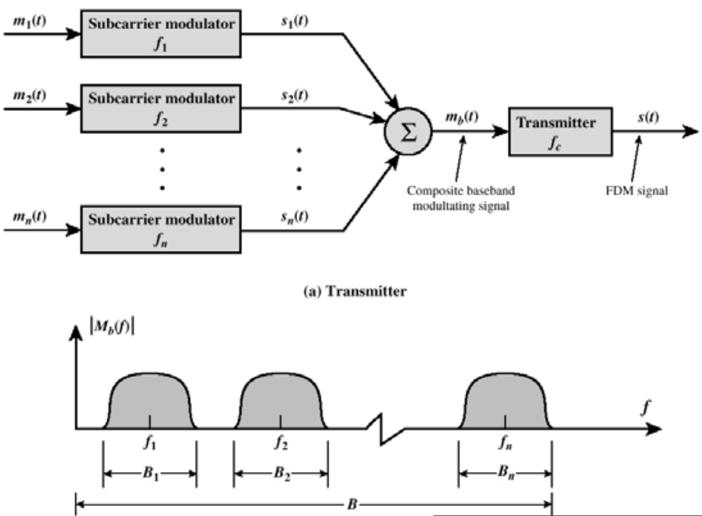
- A generic term used where more than one application or connection share the capacity of one link
- Why?
  - To achieve better utilization of resources



N channels share the capacity of link

Frequency-Division Multiplexing - Transmitter

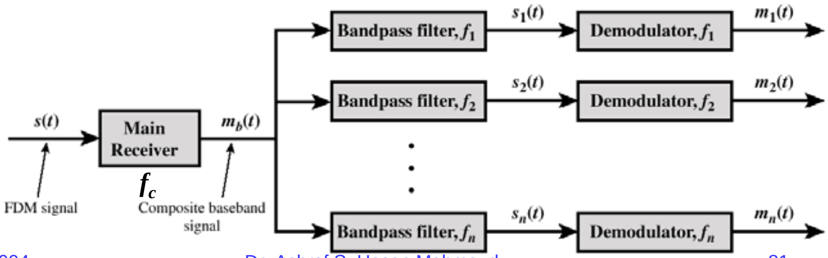
- $m_i(t)$: analog or digital information
- Modulated with subcarrier f_i ➔ $s_i(t)$
- $m_b(t)$ composite baseband modulating signal
- $m_b(t)$ modulated by f_c ➔The overall FDM signal s(t)



Spectrum function of composite baseband modulating signal $m_b(t)$

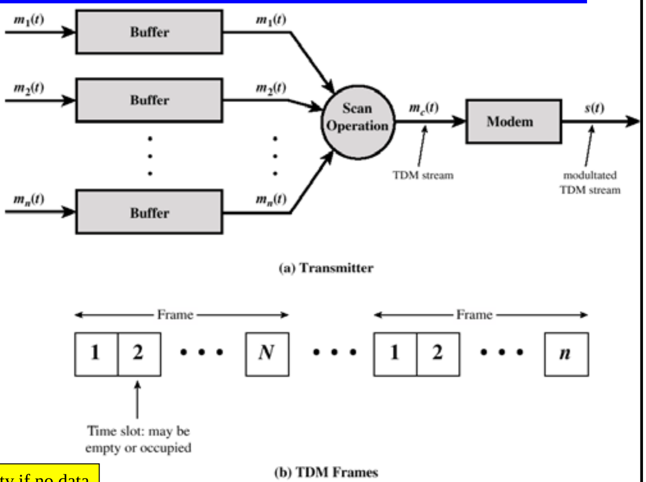
Frequency-Division Multiplexing - Receiver

- $m_b(t)$ is retrieved by demodulating the FDM signal s(t) using carrier f_c
- $m_b(t)$ is passed through a parallel bank of bandpass filters – centered around f_i
- The output of the i^{th} filter is the i^{th} signal $s_i(t)$
- $m_i(t)$ is retrieved by demodulating $s_i(t)$ using subcarrier f_i



Synchronous Time-Division Multiplexing - Transmitter

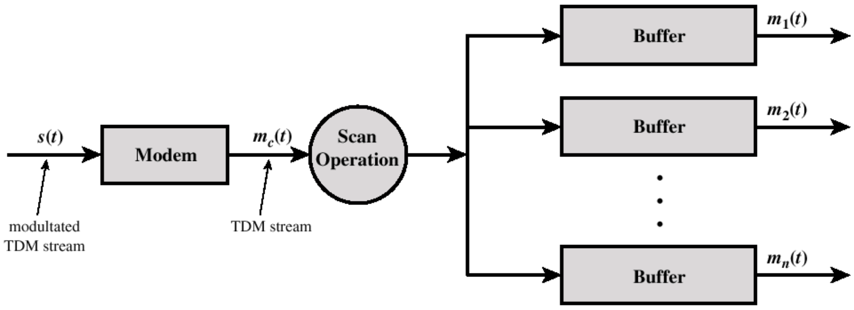
- Digital sources $m_i(t)$ – usually buffered
- A scanner samples sources in a cyclic manner to form a frame
- $m_c(t)$ is the TDM stream or frame ➔ frame structure is fixed
- Frame $m_c(t)$ is then transmitted using a modem ➔ resulting analog signal is s(t)



Fixed assignments – slot left empty if no data
No address requirement
Bit stuffing

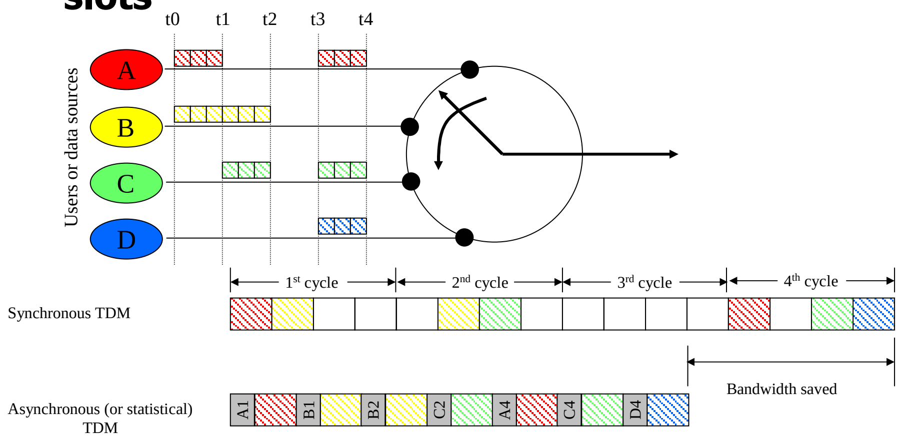
Synchronous Time-Division Multiplexing - Receiver

- TDM signal s(t) is demodulated ➔ result is TDM digital frame $m_c(t)$
- $m_c(t)$ is then scanned into n parallel buffers;
- The i^{th} buffer correspond to the original $m_i(t)$ digital information



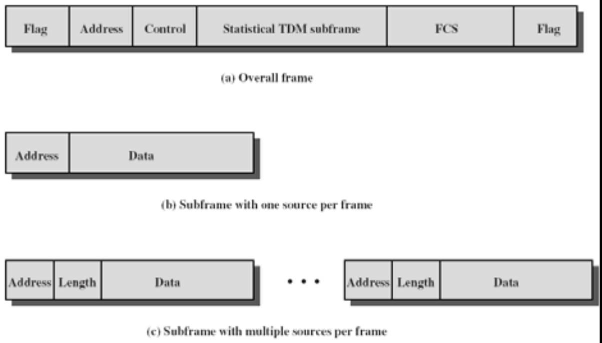
Statistical Time-Division Multiplexing

- **Dynamic and on-demand allocation of time slots**



Statistical Time-Division Multiplexing Frame Format

- **Clearly, the aim of statistical TDM is increase efficiency by not sending empty slots**
- **But it requires overhead info to work:**
  - **Address field**
  - **Length field**



Statistical Time-Division Multiplexing – Modeling

- **Data items (bits, bytes, etc) are generated at any time – source may be intermittent (bursty) not constant**
- **R b/s is the peak rate for single source**
  - **αR b/s is the average rate for single source (0 ≤ a ≤ 1)**
- **The *effective* multiplexing line rate is M b/s**
- **Each data item requires T_s sec to be served or tx-ed**
- **Data items may accumulate in buffer before server is able to transmit them → Queueing delay**